Solving the Agile Earth Observation Satellite Scheduling Problem with CP and Local Search

Valentin Antuori
LAAS – CNRS, Toulouse, France

Damien T. Wojtowicz
LAAS – CNRS, Toulouse, France

Emmanuel Hebrard
LAAS – CNRS, Toulouse, France

Abstract

The increasing hunger for remote sensing data fuels a boom in satellite imagery, leading to larger agile Earth observation satellite (AEOS) constellations. Therefore, instances of the AEOS scheduling problem (AEOSSP) has become harder to solve. As most existing approaches to solve AEOSSP are designed for a single spacecraft or smaller constellations in mind, they are not tailored to the need of our industrial partner that is about to launch a constellation of 20 AEOSs. Hence, we designed a local search solver able to schedule observations and downloads at such a scale. It relies on solving a series of sub-problems as travelling salesman problem with time windows (TSPTW), first greedily, then using a CP-SAT exact solver in order to find a solution when the greedy insertion fails. Lastly, it schedules downloads and enforces memory constraints with greedy algorithms. Experiments were carried out on instances from the literature as well as generated instances from a simulator we designed. Our experiments show that using CP to solve the sub-problem significantly improve the solutions, and overall our method is slightly better than state-of-the-art approaches.

2012 ACM Subject Classification Computing methodologies → Planning and scheduling; Applied computing → Operations research

Keywords and phrases Local Search, Greedy Algorithms, Aerospace Applications

Digital Object Identifier 10.4230/LIPIcs.CP.2025.3

Supplementary Material *Dataset*: `https://github.com/ssquilla/Earth_Observing_Satellites_benchmarks` [35]; archived at `swh:1:dir:c1c87cf9c3a3e7ffe616bdffdc9c52af9e30cb35`

Funding France Relance grant (project JAPETUS)

1 Introduction

Fueled by an increasingly easier access to space, technological advances, and an ever-growing hunger for up-to-date information, satellite imagery is booming. Since the introduction of the Pléiades project [13], satellite surveillance has typically been carried out using agile Earth observation satellites (AEOS), because of their ability to observe an area within a large horizon thanks to their yawing, pitching and rolling capabilities (see Fig. 1). Those satellites are typically deployed within constellations of a few spacecrafts. Current existing projects, such as JAPETUS, lead by our industrial partner Prométhée[1], aim at launching constellations of several dozen satellites.

As a consequence of the growing size of instances and constellations, the already NP-hard AEOS Scheduling Problem (AEOSSP) becomes even harder to solve in practice. This is a major problem for AEOS constellation operators, as low-quality observation plans would

[1] See `https://promethee.earth/`.

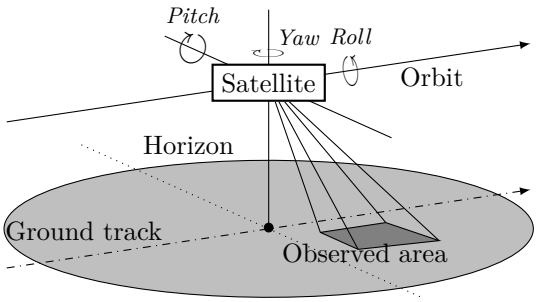

Figure 1 Schematic view of an agile satellite's observation capabilities.

compromise their profitability. Moreover, their business model is increasingly based on reactivity, allowing their customers to request images and have them delivered within a few hours. AEOSSP must then be solved as fast as possible, since it is part of broader processes that are all subject to tightening deadlines. As a side effect, image download planning has to be efficient in order to minimize the overall processing time of a request. In the past two decades, extensive work has been carried out on solving several versions of the AEOSSP [34]. Many methods have been proposed so far, including exact methods, heuristics and metaheuristics. However, most of these proposals are designed with a single spacecraft or smaller constellations in mind, and without considering download planning or a reactive business model.

In this paper, we introduce the industrial application of an acquisition planning method using several techniques from CP, that can deal with large scale instances including complex requests while performing download planning. Our method can quickly find good solutions for a constellation of twenty AEOSs on instances reaching up to a thousand complex requests over a planning horizon of six hours. Our approach is similar to the algorithm designed by Squillaci, Pralet and Roussel [25] in that it relies on a local search approach hybridized with an external solver. However, our method not only uses a SAT-based CP solver to solve sub-problems rather than solely relying on greedy algorithms, but also considers satellite memory constraints and takes download decisions instead of pre-assigning time intervals dedicated to downloads.

It solves the AEOSSP by decomposing it into an acquisition planning problem and a download planning problem. The first one is solved via local search whose moves aims at building a sequence of acquisitions for a single satellite over a given time period consisting of several overlapping acquisition opportunities, which often correspond to a daylight portion of the satellite's orbit over land. Those moves are performed using calls to a CP solver and greedy algorithms. The download planning problem takes an acquisition plan as input and is solved using a greedy algorithm.

Experiments were carried out over a set of instances from the literature as well as instances generated using a simulator we designed. Results shows that our solver is comparatively faster and finds better solutions than both the state-of-the-art aforementioned LNS method tailored for AEOSSP and a baseline model implemented with IBM's CP Optimizer.

This paper is organized as follows. AEOSSP is defined in Sect. 2. Then, a brief overview of related work is provided in Sect. 3. Our approach is explained in Sect. 4, then evaluated and compared with the literature in Sect. 5. Finally, this paper is concluded in Sect. 6.

2 Problem Statement

The AEOSSP problem aims at scheduling observations and data transfers in order to maximize the profit of the constellation's operator. Together with our industrial partner, we adopted an integrated planning definition, i.e. including constraints on AEOSs' memory and computing not only an observation planning but also a download planning. In this respect, it is similar to that adopted in some of the latest research [11, 10]. Since our partner emphasizes reactivity, AEOSSP is periodically solved over a few-hour rolling planning horizon. Hereafter, we describe the elements of the problem before formalizing it.

2.1 Input Data

Let $\mathcal{R}$ be a set of customer *requests*. A request r involves observing an area split into a set $\mathcal{M}_r$ of *meshes*. Requests can be repeated, hence we define the set $\mathcal{P}_r$ of those *repetitions*. For instance, there can be a request to observe the area around Glasgow, which is decomposed into a grid of meshes covering the city, with a two-hour period during the CP 2025 conference.

The constellation we consider has a set $\mathcal{S}$ of *satellites*, that can be used to observe a mesh for a given period. Such an action is called an *acquisition opportunity*, although we shall use *acquisition* for short. Let $\mathcal{A}$ be the set of acquisitions. We define the subset $\mathcal{A}_{r,t,m}$ as the set of possible acquisitions of the request r for the period t and the mesh m, and $\mathcal{A}^s$ as the subset of acquisitions which can be made by satellite s. Each acquisition a has a duration p_a to make that observation, which must occur during a time window $[r_a, d_a]$. It is associated to a profit v_a, and it consumes a quantity of memory γ_a. Satellites may have some acquisitions in memory at the beginning of the planning period. For each satellite s, $\mathcal{F}^s$ is the set of such acquisitions, and we note $\mathcal{F} = \cup_{s\in\mathcal{S}}\mathcal{F}^s$. There can be many alternative acquisitions for the same observation *task* (i.e. a mesh for a given period), since the same image can be shot by different satellites or the same satellite on different orbits. Therefore, the goal is to select acquisitions among the many alternatives, so that as many requests as possible are fulfilled.

Then, the acquisitions must be scheduled. As satellites need to maneuver in order to aim at their targets (i.e. to observe a mesh), there are *transition times* between successive acquisitions performed by the same satellite. In agreement with our industrial partner and in line with other works in the literature [3], we assume that transition times are time-independent. Thus, $T(a_1, a_2)$ is the transition time between two acquisitions $\{a_1, a_2\} \subseteq \mathcal{A}^s$ on a satellite s.

Finally, a network of *ground stations* (GS) are used to download the observations. For each acquisition a, there is a set $\mathcal{W}_a$ of possible *download windows*. A download time window w is a time window $[r_w, d_w]$ during which acquisitions can be downloaded. It has a total capacity γ_w which is the product of its bandwidth by its duration. Each satellite s has a memory capacity C_s.

Memory management is the second part of the problem. It consists of assigning each selected acquisition in the plan to a download time windows, while ensuring that onboard memory and download capacity constraints are respected.

2.2 Decision Variables and Constraints

The following model formally describes the problem. For each acquisition $i \in \mathcal{A}$, we have a boolean selection variable $x_a \in \{0, 1\}$ and a start time variable $s_a \in \{r_a, d_a - p_a\}$. Moreover, for each acquisition $a \in \mathcal{A} \cup \mathcal{F}$ and for each of its download options $w \in \mathcal{W}_a$, there is a boolean selection variable $y_{a,w} \in \{0, 1\}$.

The problem is to compute a mission, i.e. select and schedule the acquisitions and downloads for each satellite in order to maximize the profit, satisfying the following constraints:

Acquisitions time windows: Selected acquisitions must occur wholly within their time windows.

$$x_a \implies r_a \leq s_a \leq d_a - p_a \quad \forall a \in \mathcal{A} \tag{1}$$

Acquisitions no-overlap: On a satellite $s \in \mathcal{S}$, the selected acquisitions must not overlap in time.

$$x_a \wedge x_b \implies s_a + p_a + T(a,b) \leq s_b \vee s_b + p_b + T(b,a) \leq s_a \quad \forall \{a,b\} \subseteq \mathcal{A}^s \tag{2}$$

Acquisitions alternatives: No more than one acquisition among alternatives for each observation target shall be done.

$$\sum_{a \in \mathcal{A}_{r,t,m}} x_a \leq 1 \quad \forall r \in \mathcal{R}, \forall t \in \mathcal{P}_r, \forall m \in \mathcal{M}_r \tag{3}$$

Acquisitions download: Selected acquisitions cannot be downloaded more than once, they must be finished before being downloaded[2].

$$\sum_{w \in \mathcal{W}_a} y_{a,w} \leq x_a \qquad \forall a \in \mathcal{A} \tag{4}$$

$$y_{a,w} \implies s_a + p_a \leq r_w \quad \forall a \in \mathcal{A}, \forall w \in \mathcal{W}_a \tag{5}$$

Satellite memory: At any given time t, the quantity of data stored on a satellite $s \in \mathcal{S}$ (equal to the total acquired before t minus the total download before t) must not exceed its capacity[3].

$$\sum_{a \in \mathcal{A}^s | x_a \wedge s_a < t} \gamma_a - \sum_{a \in \mathcal{A}^s, w \in \mathcal{W}_a | y_{a,w} \wedge d_w < t} \gamma_a \leq C_s \quad \forall t \tag{6}$$

Downloads capacity: The total quantity of data downloaded during a time window must not exceed its capacity.

$$\sum_{a \in \mathcal{A}} y_{a,w} \gamma_a \leq \gamma_w \quad \forall w \in \mathcal{W} \tag{7}$$

The objective is then simply to maximize the total profit:

$$\sum_{a \in \mathcal{A}} x_a v_a \tag{8}$$

This objective function might seem overly simple. However, from the point of view of the operators, this is relatively easy way to steer the plan toward their true objectives. In particular, some efforts have been put into complex requests with non-additive rewards [25]. For instance, imagine that you want a complete map of Scotland. It requires to perform a large set of acquisitions, hence having only a subset of the observations may be worthless, or at least the profit should not be linear with the ratio of the observations that were made. However, such complex requests are likely to span over several planning horizons. Therefore, modeling a complex cost system might be detrimental if the whole request can never fit within one planning horizon. On the other hand, with a simple additive objective, the operators may for instance start with attaching low profits to each observation of a complex request, and then increase the weight associated to missing observations on subsequent planning sequences, depending on how close to completion they are.

3 Related Work

Initial work on AEOSSP was carried out as part of the development of the Pléiades constellation [13]. Since then, there have been numerous articles on this topic [34, 24], considering several variations of the problem, and exploring a wide array of methods that are generally in line with related aerospace problems [36]. In this section, we provide a brief overview of the state of the art on AEOSSP.

[2] We make the conservative assumption that an acquisition can be downloaded only if the acquisition is finished at the start of the download time window.

[3] We make the conservative assumption that memory is released at the end of the download time windows.

3.1 AEOSSP variations

Substantial differences exist in the scope of the AEOSSP used throughout the literature, mainly on extensions to the core observation scheduling problem [13]. This core is NP-hard [13]. Moreover, exhaustively taking into account all possible extensions of the AEOSSP can make it too complex, therefore intractable for existing methods. As a consequence, all the publications that we are aware of about AEOSSP consider only a subset of the constraints with more or less radical simplifications. We also defined with our industrial partner a somewhat simplified version of AEOSSP in this paper, as previously discussed in Sect. 2.

Some models [28, 6, 14] consider the breakdown into meshes and strips (i.e. long polygons imaged while the satellite is moving) of the requests' areas of interest as decision. As in most of the literature, we assume that this step is part of some preprocessing and therefore outside of the scope of our work.

Depending on the AEOSs' capabilities, the nature of the requests themselves can vary. Indeed, they can be simple, one-shot images of a given area, strips covering long areas [1], repetitive, stereoscopic [20], video requests [5], or even be instrument-specific when several sensors are on the spacecrafts [18, 17, 37]. Our solver is agnostic to the nature of the acquisitions; expressing punctual and stereoscopic requests can be done at preprocessing time using repetitions.

The *quality* of the images is time-dependent [21, 16, 29, 33, 22] and multifaceted (shooting angle, luminosity, cloud cover, customer preferences, etc.), leading some authors to consider AEOSSP as a multi-objective problem [16, 15]. Hence, in addition to profit, those works seek to maximize image quality or use of the satellites. *Profit* itself can be time-dependent [28] since it is quality-dependent. It can also be affected by urgency and priority policies among requests [29, 27], and even be sequence-dependent in the case of repetitive or stereoscopic requests [4, 12, 14, 20]. In our work, we have a black-box approach to profit as it is determined for each acquisition by our partner: quality and urgency are therefore sidelined as they are embedded within the profit, hence removing its time-dependence.

There are also operational constraints. Firstly, *transition times*, which is the time the satellite needs to maneuver towards its target. They are by nature dependent on the time and sequence of acquisitions made by AEOS [3, 22, 2]. They are either modeled as such, or approximated by piecewise linear functions or constants, or simply ignored. Our approach also approximates transition time using a transition time matrix between categories of attitude adopted by the AEOSs when performing the acquisitions. Hence, it allows to take into account time and sequence dependencies at a granularity defined at preprocessing time while formally removing time-dependence from our model.

Then there can be constraints about onboard *energy* management. The energy comes from solar panels, and the rate at which batteries are refilled therefore depends on both the time (the exposition to the sun depends on the orbit of the satellites) and the tasks performed by the satellite since it constrains the orientation of the solar panels [32, 33]. However, on-board energy management is rarely a bottleneck, and hence it is usually not taken into account when planning the observations. In accordance with the needs expressed by our industrial partner, this aspect is not covered in our work.

Finally, there is the management of onboard *memory* and its corollary: the management of *downloads* [21, 10, 11, 19]. Scheduling both acquisitions and downloads is often referred to *integrated scheduling* in the literature [34]. Memory can be looked at in two ways. By ignoring the files, it becomes a reservoir that is filled and emptied according to linear functions. If the files are taken into account, then it is a set of blocks that must be distributed between the download windows according to their capacity. We take the second approach, as it best fits our partner's processes.

3.2 Resolution methods

The wide range of variations in AEOSSP has resulted in the study of numerous resolution methods.

Exact methods have been extensively used to solve the AEOSSP. For instance, a branch-and-bound algorithm taking into account time-dependent constraints was proposed and shown to be effective on small instances [3]. Several MILP formulations have also been proposed [29, 12], including in the context of integrated scheduling [10, 11]. Some work only focus on data download [19] and are more in line with broader work on the Antenna-Satellite scheduling problem. Although MILP was successfully used to perform integrated scheduling with a commercial solver for Planet Lab's constellation of over a hundred spacecrafts [23], they are not *agile* satellites and hence the scheduling problem is a lot easier than the AEOSSP. Moreover, while projects with a problem statement roughly similar to ours [11] does use MILP, their instances are way smaller (4 satellites, up to 3 ground stations and 200 requests), do not take into account transition times, and have a time limit orders of magnitude too high (approx. 3 h) to be used in our industrial partner's processes. For all these reasons, we have not retained the possibility of using CP to solve the problem as a whole, but rather to solve sub-problems as a local search move.

Much of the work on AEOSSP has chosen to abandon optimality guarantees and propose methods based on heuristics and metaheuristics. Several methods solely based on heuristics have been proposed to solve AEOSSP. The scheduler of the COSMO-SkyMed constellation, comprising four spacecrafts, simply selects as much acquisitions as it can by descending order of priority [1]. Heuristics can also be used to compute the acquisitions given areas of interest in the context of an online scheduler [14]. A method using deep reinforcement learning and heuristic algorithm has also recently been proposed to solve AEOSSP for a single satellite [2]. Although those heuristics-based methods are easy to implement and yield good results for a single satellite or smaller constellation, they are not expected to be as good when applied to a whole constellation because of the larger number of alternative acquisitions. Methods using machine learning are also of great interest to solve AEOSSP, as satellite operators accumulate many instances over time, but the size of the instances and the lack of real-world data from our industrial partner led us away from learned heuristics.

Many metaheuristics-based methods has also been proposed since the first studies on AEOSSP, such as taboo search [4]. Genetic algorithms have been widely used to tackle multi-objective versions of the AEOSSP, for instance when considering operator's preferences [16], load balancing [7], or solution stability in an online setting [15] in addition to profit. They were also applied to decompose sub-problems. In a first example, it is used for single-satellite scheduling within a broader game theory framework where satellites are players [20]. In a second example, it is used to choose between overlapping acquisitions within multi-level clusters computing using priority metrics [8]. Agent-based methods are also interesting to solve small-scale instances of AEOSSP [5].

Finally, local search and large neighborhood search (LNS) methods have also been extensively studied, including in the context of multi-objective optimization [28, 37] and online solvers [27]. Moves proposed in the literature rely on a wide array of algorithms. On the more complex side, there is a move based on computing maximum independent sets within acquisitions interval graphs using an evolutionary algorithm [6]. Unsupervised learning has also been used as a preprocessing tool for destructive LNS moves [21]. However, most of the constructive LNS moves are simpler, as they widely rely on random choices [9] and greedy algorithms [9, 22, 25]. It should be noted that these local search and LNS methods are similar to what can be found in the team orienteering problem [30], since it shares many characteristics of the acquisition selection sub-problem of AEOSSP.

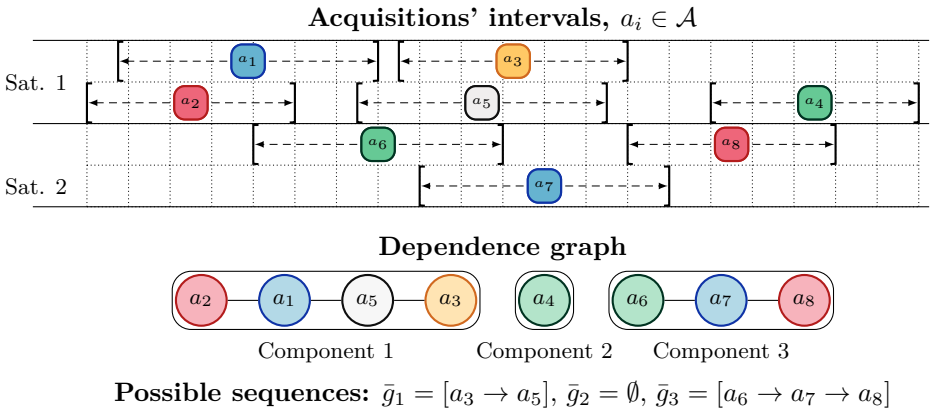

Figure 2 Example of sequences defined from the dependence graph built from a set of acquisistions. Acquisitions with the same colour are alternatives to cover the same mesh for the same task of the same request. The interval graph contains three connected components.

The solver we present in this article is also in this line of work, as local search methods are particularly effective for solving complex instances in satellite constellations, while making it easier to break down the problem into acquisition planning and download management.

4 A Local Search Approach

In order to solve our version of AEOSSP, we leverage the structure of the problem to decompose it into two parts. Firstly, the selection and allocation of the acquisitions – the master problem – is solved using a local search method that computes sequences of acquisitions for each satellite. Secondly, one planning of file downloads – the sub-problem – is solved using a greedy algorithm. Those two parts are repeated until a time limit.

4.1 Acquisition Planning

The acquisition planning phase aims at building a sequence of acquisitions for each satellite maximizing the profit under the aforementioned constraints 1, 2 and 3.

Our method uses the fact that there exists subsets of acquisitions of a given satellite that are temporally independent of each other [6, 26]. This can happen when the satellite flies over a large area with little interest (e.g. an ocean, cloudy areas, etc.), or simply because of the satellite's orbital nighttime. Formally, for each satellite $s \in \mathcal{S}$ we can define a dependence graph whose nodes are the acquisitions $a \in \mathcal{A}^s$. There is an edge between two acquisitions $\{a, b\} \subseteq \mathcal{A}^s$ if $d_a + T(a, b) > r_b \vee d_b + T(b, a) > r_a$, i.e., when their time windows overlap while taking into account transition time. As a consequence, each connected component of these graphs are temporally independent of each other (see example in Fig. 2). Hence, local optimization may be performed over each connected component in order to build the plan in the form of sequences of acquisitions.

Our algorithm for acquisition planning has two phases. Firstly, it builds a solution from scratch with a greedy algorithm. Secondly, a local search phase aims at improving the solution until a local minimum is reached.

4.1.1 Greedy Initial Solution Construction

The greedy algorithm iteratively tries to insert an acquisition. It uses a utility criterion $U(a) = v_a/p_a$ defined as the ratio of an acquisition a's profit over its duration to select the next acquisition to insert. The algorithm iterates over the acquisitions whose associated mesh and task are not currently covered by descending order of their utility, by trying to insert them in their satellite's plan to minimize the increment of transition times.

The time complexity to find the best valid position is linear in the length of the current sequence and to insert the acquisition is linear in the size of the sequence under construction.

Therefore, the algorithm runs in $O(|\mathcal{A}|^2)$ time overall for a sequence of acquisition $\mathcal{A}$.

Let $\mathcal{A} = a_0, a_1, \ldots, a_k, a_{k+1}$ be a feasible sequence of acquisitions with a_0 and a_{k+1} dummy acquisitions of null duration, null transition time with all other acquisitions and such that $r_{a_0} = 0$ and $d_{a_{k+1}} = \infty$. The greedy insertion algorithm returns the position $i \in [0, k]$ such that inserting a between acquisitions a_i and a_{i+1} yields a feasible sequence and $T(a_i, a) + T(a, a_{i+1})$ is minimum.

The following quantities can be accessed in constant time:

$$s_{a_i} = \max(r_{a_i}, s_{a_{i-1}} + p_{a_{i-1}} + T(a_{i-1}, a_i)) \qquad \forall 1 \leq i \leq k \tag{9}$$

$$slack(a_i) = d_{a_i} - s_{a_i} - p_{a_i} \qquad \forall 1 \leq i \leq k \tag{10}$$

$$gap(a_i) = s_{a_i} - s_{a_{i-1}} + p_{a_{i-1}} + T(a_{i-1}, a_i) \qquad \forall 1 \leq i \leq k \tag{11}$$

Then, we store the values of the cumulative gap up until acquisition a_i in linear space:

$$cgap[a_i] = \sum_{j=0}^{i-1} gap(a_j) \qquad \forall 1 \leq i \leq k \tag{12}$$

Therefore, we can access in constant time $F(a_i, a_j)$ the time length available between tasks a_i and a_j (with $j > i$) in the sequence:

$$F(a_i, a_j) = cgap[a_j] - cgap[a_i] + slack(a_j) \qquad \forall 1 \leq i \leq k \tag{13}$$

Finally, we store the time length available for a task insertion immediately after task a_i using linear space:

$$I[a_i] = min_{i<j\leq k} F(a_i, a_j) \qquad \forall 1 \leq i \leq k \tag{14}$$

When trying to insert acquisition a, the algorithm makes a linear search over the sequence $a_0, a_1, \ldots, a_k$ to find the position $i \in [0, k]$ that minimizes $T(a_i, a) + T(a, a_{i+1})$, whilst ensuring that the insertion is feasible using the value of $I[a_i]$. Then, after the actual insertion, the values of s, $cgap$ and I are updated in linear time by once again traversing the acquisition in the sequence. An insertion therefore requires two linear-time traversals of the sequence.

Finally, we introduce randomization. Indeed, instead of taking the next acquisition in utility order, there is a small probability to take a random one instead. Once this algorithm has produced an initial solution, the local search starts.

4.1.2 Local Search

Our method locally optimizes the plan by iteratively recomputing sub-sequences. A move for our local search consists in solving a kind of *orienteering problem* over the complete graph formed by the acquisitions of a given connected component. In this graph, each node is an acquisition with the same time window, and is associated to a profit, which we will describe

below. The transition time between two acquisitions is the length of the path between the two associated nodes. The goal is to find a path in this graph that maximizes the sum of the profit of the visited nodes, while respecting time windows (i.e. finding a maximum-profit sequence of acquisitions).

The profit of selecting an acquisition a is defined as follows. It uses the marginal profit $\mathcal{P}$ that is the profit increment that would occur if an acquisition was inserted in the plan instead of another alternative acquisition (i.e., the cost of swapping acquisition a for an alternative acquisition b if it was in the current plan, that is, if $x_b = 1$).

$$\mathcal{P}(a) = v_a - \sum_{b \in \mathcal{A}_{r,t,m} \setminus a} x_b v_b \quad r \in \mathcal{R}, t \in \mathcal{P}_r, m \in \mathcal{M}_r \mid a \in \mathcal{A}_{r,t,m} \tag{15}$$

Should the move find a solution with a higher overall profit than the current plan, the new sequence that was built for the considered connected component substitutes the one in the plan. The latter is updated as a whole, by deselecting already selected alternative acquisitions from the new sequence.

To solve this sub-problem, we use the same procedure as for the initialization: for each acquisition in decreasing order of marginal profit, it tries to insert acquisitions within the sequence. This procedure stops when the first acquisition with a non positive marginal profit is reached. Two insertion techniques are used. The first tries to insert the acquisition in the sequence so that transition times are minimized, just as in the initial solution construction. The second technique is used when the first fails: an external solver is called to perform the insertion of the new acquisition by rescheduling the whole connected component.

We use *Tempo*[4], a CP-SAT solver that we are developing as part of a distinct project. It is asked to solve a traveling salesman problem with time windows (TSPTW), where the goal is to find a valid tour in the graph formed by the acquisitions already in the sequence extended with the acquisition we are trying to insert. The time window of a node is the time window of the acquisition, and the transition time between two nodes is the transition time between the two associated acquisitions. We model this problem with one interval variable for each acquisition i. The release and due dates of the interval variable of an acquisition i are r_a and d_a respectively, and its duration is p_a. Then the only constraint of the model is a NoOverlap on all the acquisitions, taking into account the transition times. There is no objective function as we just seek a valid sequence of acquisitions. Tempo's search is limited by a fail limit. The main benefit of this approach is that it allows to reorder the sequence when a solution is found, possibly leaving space for further acquisition insertions.

Finally, the neighborhood of a given plan is defined by the number of connected components. Indeed, we have one possible move per connected component, that eventually leads to an improving neighbor. Therefore we choose to perform a simple descent, choosing at each iteration the first improving neighbor among all randomly ordered neighbors, until a local minimum is reached, i.e. none of move can improve the current plan.

4.2 Download Planning

The download planning phase takes place when a local minimum is found. Its aim is twofold. It first computes a download plan by affecting download windows to the planned acquisitions and the initial memory elements aboard the satellite. Then, it fixes potential remaining memory overloads on the satellites by progressively deselecting acquisitions.

[4] See `https://gitlab.laas.fr/roc/emmanuel-hebrard/tempo`.

For each satellite, the algorithm iterates over the download windows by increasing order of their starting time, in order to greedily affect acquisitions until the download window's capacity is reached or there is no remaining acquisition to download. This greedy algorithm starts by collecting the acquisitions that are still on board at the starting time of the download time window, and sort them by decreasing order of their memory consumption. Then acquisitions are assigned in this order to the window if they fit.

Despite download decisions being taken, memory constraints may still be violated as the algorithm is blind to memory overloads. Hence, a memory overload repairing procedure is applied to satellites with overloads. Since we assume that the memory is freed at the end of the download window, overloads are checked at these time points. When one is found, a knapsack problem is solved over the onboard acquisitions. The knapsack capacity is the memory capacity of the satellite, the weights are the memory consumption of the acquisitions, and the values are the profits of the acquisitions. We use a simple greedy algorithm with a profit over size ratio utility to solve the knapsack problem.

It should be noted that while both procedures are using greedy algorithms, it is not an issue as download planning is often an easy task and memory is rarely a bottleneck. Finally, it should be noted that since our overall method is solved on a rolling horizon, some images may not be downloaded during the current planning horizon. They will be the content of the set $\mathcal{F}$ of the next planning horizon.

5 Experiments and Results

We evaluated our approach against a state-of-the-art commercial solver and recent proposals from the literature, over instances from the literature as well as generated by a simulator of our own design.

5.1 Experimental setting

Experiments were run on a cluster of computers with 2,1 GHz Intel CPUs and 256 GB RAM. Results are the aggregation of 5 runs with different seeds and a 10 min timeout. We implemented our solver with C++ 20.

5.1.1 Instances

We used two instance sets to evaluate our work.

The first instance set, hereafter referred to as "Squillaci et al.", comes from recent work[5] [25]. It contains 32 instances containing up to 1140 requests. Details of the instances are provided in Table 7 (Appendix. A). They are regional instances focusing on Europe, half of them are generated using a uniform target distribution and the other half focus on fifty large European airports, containing one-shot, video, stereoscopic and periodic requests. It should be noted that video requests are like one-shot requests but longer and more profitable. Moreover, a stereoscopic request implies observing the same area twice with the same satellite with two different angles. Therefore, we slightly adapted our method in order to support stereoscopic requests, by trying to insert the second acquisition right after the first one has been inserted into its satellite's sequence. If the second insertion fails, then the first acquisition's selection is canceled. The marginal profit of an acquisition from a stereoscopic

[5] Instances retrieved from `https://github.com/ssquilla/Earth_Observing_Satellites_benchmarks`.

Table 1 Satellites orbital parameters for the simulation, including plane inclination i (deg), altitude a (km), longitude of ascending node Ω (deg), mean anomaly M and period P (min).

(a) Inclined plane orbits.

i	a	Ω	M	P
90	500	-180	$\{-180, -90, 0, 90\}$	90
90	500	-90	$\{-180, -90, 0, 90\}$	90
90	500	0	$\{-180, -90, 0, 90\}$	90
90	500	90	$\{-180, -90, 0, 90\}$	90

(b) Heliosynchronous orbits.

i	a	Ω	M	P
98	550	-144	-180	90
98	550	-60	-90	90
98	550	60	0	90
98	550	144	90	90

request is the sum of both's marginal profits. We did not use the profit function proposed in [25] for periodic requests since we have an all-or-nothing approach for the requests. Finally, as the instance files did not provide transition times, we followed the method of Squillaci et al. to compute it as a function of the euclidean distance between the targets.

The second instance set we used was produced by an instance generator we designed[6]. The latter's input is a set of coordinates corresponding to points of interest (POIs) distributed over the Earth, then calculates meshes covering them using a greedy algorithm. It then simulates satellite orbits to determine the visibility time windows for each mesh and ground station over a given planning horizon. Requests are then generated by randomly selecting a subset of the meshes, with a probability of also selecting their neighbors in order to simulate requests over large geographical areas. The number of periods depends on the view accesses associated with the selected meshes, and is also randomly selected.

We produced a set of 20 instances[7], containing up to 500 requests over a planning horizon of 6 h, assuming a constellation of twenty satellites evenly spaced on four parallel inclined orbits and an heliosynchronous orbit (see orbital parameters in Table 1). We assumed a network of 6 ground stations with characteristics similar to KSATlite[8]. As in many other work, we used non-uniform POI distribution at regional scale [1, 9, 25] – hereafter referred to as "Antilles", "Sahara" and "Fountains" – and worldwide scale – "Military" and "World Heritage" – that we extracted from OpenStreetMap. For each POI set, we generated 4 instances from 50 to 500 requests. Further details about the instances are in Table 6 (Appendix. A).

In order to make a comparison, we use the same hypothesis to compute transition time as in Squillaci et al. instances. For a given target, we choose a base profit and a decreasing slope randomly, each acquisition has then a profit that depends on its start date, the later it is, the lower will be the profit. In addition to that, each acquisition has a chance to have a large penalty in order to simulate bad weather.

5.2 Comparison points

We compared our algorithm with the results of the aforementioned LNS method from the literature, as well as a baseline model with IBM's CP Optimizer (CPO).

[6] See source code at `https://gitlab.laas.fr/roc/damien-wojtowicz/aeossp_instance_generator`.
[7] Instances available at `https://gitlab.laas.fr/roc/damien-wojtowicz/aeossp_instances`.
[8] See `https://www.ksat.no/ground-network-services/ksatlite/`.

5.2.1 LNS Method

We choose Squillaci, Roussel and Pralet's [25] LNS method because, to our knowledge, it is the most similar to ours in the literature. It uses dedicated moves to relax sets of requests before reinserting the requests in the plan.

Its destructive phases use the same concept of connected components within an acquisitions' time windows graph as our method, as well as other work in the literature [6]. Those phases randomly select a subset of connected components, then relax a subset of acquisitions within, so as to have more choice to reinsert a request after relaxation.

Repair phases use CPO to solve an orienteering problem over the subset of the selected connected components. Although this algorithm can run in a multi-core parallel mode, we run the experiments with a single core in order to do a fair comparison with our algorithm.

5.2.2 CP Model with CPO

Finally, we implemented a baseline CP model with CPO version 20.1.0 using its C++ API, solved with a single thread and running on the same machines as our solver. Under the default parameter setting, CPO runs LNS until a local optimum is detected and then switches to complete search, namely, Failure Directed Search [31]. We defined three sets of interval variables to model this problem. First, there is an optional interval $X_a := \langle s_a, e_a \rangle$ for each acquisition $a \in \mathcal{A}$, with $s_a \geq r_a$, $e_a \leq d_a$ and $e_a - s_a = p_a$. Then, there is an interval $Y_w := \langle s_w, e_w \rangle$ for each download window $w \in \mathcal{W}$ with $s_w \geq r_w$ and $e_w \leq d_w$. Lastly, there is an optional interval $Z_{a,w} := \langle s_{a,w}, e_{a,w} \rangle$ for each download window $w \in \mathcal{W}_a$ for each acquisition $a \in \mathcal{A}$ with $s_{a,w} \geq s_w \wedge s_{a,w} \geq e_a$, $e_{a,w} \leq e_w$ and $e_{a,w} - s_{a,w} = m_a / b_w$.

The objective function is stated as follow:

$$\max \sum_{a \in \mathcal{A}} \textsc{PresenceOf}(X_a) \times v_a \tag{16}$$

We defined the following constraints. Constraint 17 ensures that acquisitions do not overlap in the plan for each satellite, taking into account a transition matrix T. Constraint 18 ensures uniqueness of observations. Constraint 19 defines a profile of memory usage, and Constraint 20 sets a limit on this profile. It should be noted that this is the typical way to model a reservoir using CPO, which we chose over a cumulative global constraint in order to avoid adding a task that starts with the acquisitions and ends at the download. Constraints 21 and 22 enforce consistency between acquisitions and downloads: only acquisitions in the plan can be downloaded and only one download can be done for each acquisition. Finally, Constraint 23 ensures that the capacity of each download window is not exceeded.

$$\textsc{NoOverlap}(\{X_a \mid S_a = s\}, T) \qquad \forall s \in \mathcal{S} \tag{17}$$

$$\textsc{Alternative}(\{X_a \mid R_a = r \wedge T_a = t \wedge M_a = m\}) \qquad \forall r \in \mathcal{R}, \forall t \in \mathcal{P}_r, \forall m \in \mathcal{M}_r \tag{18}$$

$$p_s := \sum_{a \in \mathcal{A}_s} step(X_a, m_a) - \sum_{w \in \mathcal{W}_s} step(Z_{a,w}, m_a) \qquad \forall s \in \mathcal{S} \tag{19}$$

$$\textsc{AlwaysIn}(p_s, 0, C_s) \qquad \forall s \in \mathcal{S} \tag{20}$$

$$\textsc{PresenceOf}(Z_{a,w}) \implies \textsc{PresenceOf}(X_a) \qquad \forall a \in \mathcal{A}, \forall w \in \mathcal{W}_a \tag{21}$$

$$\textsc{Alternative}(\{Z_{a,w} \mid w \in \mathcal{W}_a\}) \qquad \forall a \in \mathcal{A} \tag{22}$$

$$\textsc{NoOverlap}(\{Z_{a,w} \mid \forall a \in \mathcal{A}_w\}) \qquad \forall w \in \mathcal{W} \tag{23}$$

It should be noted that, for Squillaci et al. instances, this model has to be adapted by removing all constraints and variables dealing with the memory and the downloads, treating all the downloads as acquisitions except that their intervals are not optional, and by adding an equality constraint between the presence variable of stereoscopic pairs of acquisitions.

5.3 Results and Discussion

We now present and discuss the experimental results, by first comparing our method with the aforementioned comparison approaches. Then, we assess the impact of our use of the CP-SAT solver Tempo within the local search move, as well as the consequences of how our method manages satellite's memory.

5.3.1 Comparison with LNS and CPO

We compared our method (noted as `LS-tempo`, with a fail limit of 50 when using the solver) against the LNS from the literature (`LNS`) and the CPO model (`CPO`) over Squillaci et al.'s instances (see Table 2) and over our set of instances in (see Table 3).

The baseline `CPO` model is the most competitive for the smaller instances in every instance set. However, it does not scale up to large instances as well as the two other methods. On Squillaci et al.'s instances, it has good results up to 500 requests, is the best on 4 instances and can also prove optimality on 4 instances. On our instances, it is the best up to 100 requests and proves optimality on 3 instances. Auxiliary experiments on Squillaci's instances showed that CPO performs better with a 1 hour time budget (up to 27 % increase in profit), but still stays behind other methods (with 10 minutes budgets), especially on large instances (for example the 0-1000-0-0 instances: 22 % and 17 % behind our method). Nevertheless, our industrial context does not allow such a time budget as the resolution is part of a larger process.

Secondly, `LS-tempo` shows good performances and has often the best solutions on both instance sets (21 times the best solution, compared to 16 for `CPO` and 7 for `LNS`). On Squillaci et al. instances, while our solver is the worst over instances with periodic requests only, it is especially good for the 1000 video requests instance of concentrated target (16 % gap versus 21 % for `LNS`). `LNS` has the most balanced performances, is particularly good for the periodic-only instances, and has the best gap in average. Over our instances, `LS-tempo` is the best solver in average (0.5 % gap better). Therefore, the overall performance difference between `LNS` and `LS-tempo` is not clear since `LNS` seems to be good in average, but `LS-tempo` finds the best solution more often.

It should be noted that the low efficiency of `LS-tempo` on periodic-only instances is the consequence of a comparatively smaller number of successful acquisition insertion using the CP-SAT solver. Even when allowing our method to run for up to an hour and with a larger fail limit for the solver, results on the large periodic-only instances solely improved the gap by at most 1 %.

5.3.2 Impact of the CP-SAT Solver

We assessed the advantages of using Tempo inside our approach. Hereafter, `LS-tempo` refers to our method when it includes calls to Tempo and `Greedy` is our method without calls to Tempo.

Over Squillaci et al. instances, `LS-tempo`, the impact is significant. We observes a 0.9 % gap improvement in average, with `LS-tempo` giving better results on all but two instances. Averaging the gap makes the differences between the two solvers smaller, as 15 instances over the 32 are solved optimally with the base solver (see Table 4 for data on a subset of instances).

Over our set of instances, however, the advantage of using Tempo is not as clear. We observe no difference in the average gap of `LS-tempo` and `Greedy`. Indeed, the method makes a very high number of calls to Tempo with a very low success rate. In other words, the

Table 2 Profit and gap of the solutions found by `CPO`, `LNS` and `LS-tempo` over Squillaci et al.'s instances. Optimal profit is noted with a star (*). For the sake of clarity, only instances where at least one solver has not found the optimal solution are shown.

$\|\mathcal{R}\|$	OS	V	S	P	CPO		LNS		LS-tempo	
					profit	gap	profit	gap	profit	gap
500	500	0	0	0	**5070***	0.00	5069	<0.01	**5070***	0.00
1000	1000	0	0	0	10165	0.13	11382	0.03	**11536**	0.01
500	0	500	0	0	21176	0.01	20988	0.02	**21200**	0.01
1000	0	1000	0	0	40280	0.39	52024	0.21	**55367**	0.16
250	0	0	0	250	**4100**	0.03	4018	0.05	3946	0.07
500	0	0	0	500	10856	0.12	**11072**	0.10	10512	0.15
1000	0	0	0	1000	14643	0.33	**15415**	0.29	13953	0.36
570	120	150	270	30	22379	0.02	22409	0.02	**22585**	0.01
1140	240	300	540	60	13075	0.19	**16223**	<0.01	16220	<0.01
500	500	0	0	0	**9270***	0.00	9263	<0.01	9267	<0.01
1000	1000	0	0	0	13409	0.26	17920	0.02	**18055**	0.01
250	0	250	0	0	**18500***	0.00	18456	<0.01	**18500***	0.00
500	0	500	0	0	**36592**	0.01	36304	0.02	36420	0.02
1000	0	1000	0	0	51732	0.29	67868	0.07	**68564**	0.06
250	0	0	0	250	**6445***	0.00	6442	<0.01	6388	0.01
500	0	0	0	500	12024	0.10	**12307**	0.08	11662	0.13
1000	0	0	0	1000	17410	0.35	**19385**	0.27	18055	0.32
570	120	150	270	30	22132	<0.01	22152	<0.01	**22187**	<0.01
1140	240	300	540	60	31666	0.29	43828	0.02	**43899**	0.02
				Average		0.134		0.064		0.070

Table 3 Profit and gap of the solutions found by `CPO`, `LNS` and `LS-tempo` over our instances. Optimal profit is noted with a star (*).

Set	$\|\mathcal{R}\|$	CPO		LNS		LS-tempo	
		profit	gap	profit	gap	profit	gap
`antilles`	50	**20104**	<0,01	20050,6	0,01	20090	<0,01
	100	**36480**	0,02	36192,6	0,03	36328	0,02
	250	71324	0,30	73033,8	0,28	**74035**	0,27
	500	81745	0,60	**89858,4**	0,56	88222	0,57
`sahara`	50	**17751**	0,01	17725	0,01	17743	0,01
	100	**38702**	0,03	38393,6	0,04	38570	0,03
	250	73972	0,32	75885,2	0,30	**76020**	0,30
	500	76581	0,61	**87462,4**	0,56	85559	0,57
`fountains`	50	**20304**	0,01	20265,4	0,01	20287	0,01
	100	**37048**	0,02	36897,8	0,02	37014	0,02
	250	90934	0,15	95060,6	0,11	**97721**	0,08
	500	106102	0,52	141014,6	0,37	**141331**	0,37
`military`	50	**22467***	0.00	22442,2	<0,01	**22467***	0.00
	100	**38489***	0.00	38447	<0,01	**38489***	0.00
	250	102849	<0,01	101945,6	0,01	**102868**	<0,01
	500	206813	0,03	207729,2	0,03	**211080**	0,01
`world heritage`	50	**18222***	0.00	18213,8	<0,01	**18222***	0.00
	100	**36699**	<0,01	36655,4	<0,01	36691	<0,01
	250	103648	0,02	103253,8	0,02	**103860**	0,01
	500	185196	0,09	192808,4	0,05	**197782**	0,03
		Average	0.136		0.120		0.115

TSPTW sub-problems are either easily solved by the greedy insertion, or are very hard to solve, and hence using a complete solver is not worth the effort. Full details on those results are given in appendix B.

Table 4 Average gap, average number of restart (r), average number of calls to Tempo (t), average success rate when calling Tempo (s), time spent in the solver (Ts) and time ratio spent in modelling the sub-problem with the solver (Tm) for various fail limits. For Squillaci et al. instances, we only used the instances where at least one of the solvers did not find the optimal solution

Fail limit	Squillaci et al. instances						Our instances					
	Gap	r	t	s	Ts	Tm	Gap	r	t	s	Ts	Tm
0	0.089	2507	-	-	-	-	0.115	45772	-	-	-	-
50	0.070	337	153300	47.6%	32%	15%	0.115	5302	1341790	7.68%	21%	54%
100	0.071	264	104603	48.0%	36%	13%	0.115	3047	851625	7.81%	26%	48%
500	0.071	251	122940	50.1%	51%	8%	0.115	5418	1046200	8.10%	41%	40%
1000	0.074	233	114420	50.6%	56%	7%	0.115	4649	946012	8.15%	48%	34%
10000	0.083	208	123406	50,9%	65%	5%	0.116	3665	525106	8.20%	66%	19%
100000	0.089	188	113339	51,5%	69%	5%	0.118	3364	414748	8.26%	73%	14%

Table 4 shows the impact of the fail limit given to Tempo. We can see this parameter is not sensitive, since as a fail limit of 50 and 1000 yields roughly the same results. Beyond this value, the gap increases as well as the time spent calling the solver. More importantly, we note that the simple fact of using Tempo with a small fail limit dramatically improves the gap. It means that solutions to the TSPTW sub-problem solutions can be easily solved with a small tree search in about half the cases. Therefore, there is no use in increasing the fail limit too much, indicating that heuristics designed to solve TSPTW could yield even better performances than using a complete CP-SAT solver.

5.3.3 Memory Management Impact

Finally, Table 5 shows the comparison between the profit and the gap yielded by `CPO`, `LS-tempo` and `Greedy` when taking into account the memory constraints and computing a download plan. We can see that `CPO` loses 3.5 % on the average gap when adding these constraints, while our solver performance does not change. It means that memory constraints are not tight, and that the 6 ground stations are sufficient for the parameters we used for our instance generator. Those results are consistent with other work in the literature, which explains why memory management is often overlooked or simplified when dealing with AEOSSP [34].

6 Conclusion

In this paper, we presented a solver for the AEOSSP that is designed for our industrial partner Prométhée's the upcoming constellation of 20 AEOSs. This solver is able to tackle two sub-problems of AEOSSP, namely acquisition selection and download planning. Acquisition selection is solved by a local search procedure, which is based on greedily solving a sequence of TSPTW, with occasional help from a CP solver in order to get a maximal-profit sequence of acquisitions for each satellite. Download planning is seen as a sequence of knapsack problems, that are solved greedily. When we include the download planning, experiments showed that our solver find better solution than IBM CPO on average, especially on larger instances. Overall, with and without the download planning, IBM CPO unsurprisingly does not scale very much. We show that including calls to a CP-SAT solver inside our solver lead to an improvement on average, but these improvements depend on the type of instances. Also, the type of instances is important to choose which solver to use. Especially we show that while our solver is not good on period requests only, it exhibits state-of-the-art capabilities on other types of instances.

Table 5 Result on our set of instance with memory management

Set	$\|\mathcal{R}\|$	CPO		Greedy		LS-tempo	
		profit	gap	profit	gap	profit	gap
antilles	50	**20102**	<0.01	20089	<0.01	20087	0.01
	100	36323	0.02	36308	0.02	**36326**	0.02
	250	69674	0.31	**74115**	0.27	73929	0.27
	500	67979	0.67	**89069**	0.56	88329	0.57
sahara	50	**17748**	0.01	17736	0.01	17744	0.01
	100	38517	0.03	**38568**	0.03	38564	0.03
	250	73805	0.32	**76101**	0.30	76036	0.30
	500	68814	0.65	**86572**	0.56	85834	0.57
fountains	50	**20301**	0.01	20282	0.01	20289	0.01
	100	36939	0.02	36991	0.02	**37012**	0.02
	250	84771	0.21	97456	0.09	**97703**	0.09
	500	93629	0.58	**142525**	0.36	141493	0.36
military	50	**22467***	0.00	**22467***	0.00	**22467***	0.00
	100	**38489***	0.00	38488	<0.01	**38489***	0.00
	250	100592	0.03	102838	<0.01	**102865**	<0.01
	500	165914	0.23	211001	0.02	**211088**	0.01
world heritage	50	**18222***	0.00	18217	<0.01	**18222***	0.00
	100	**36692**	<0.01	36686	<0.01	36691	<0.01
	250	98776	0.06	103766	0.01	**103856**	0.01
	500	152770	0.25	197722	0.03	**197794**	0.03
	Average	0.170		0.115		0.116	

Finally, further work could focus on adding different moves and variations to our solver. We did experiment the resolution of the orienteering problem with our CP solver, but it does not scale for this problem today. We also would like to use it for the insertion of several acquisitions, in order to make its use more efficient by decreasing the number of calls.

References

1 Nicola Bianchessi and Giovanni Righini. Planning and Scheduling Algorithms for the COSMO-SkyMed Constellation. *Aerospace Science and Technology*, 12(7):535–544, 2008. doi:10.1016/j.ast.2008.01.001.

2 Jiawei Chen, Ming Chen, Jun Wen, Lei He, and Xiaolu Liu. A Heuristic Construction Neural Network Method for the Time-Dependent Agile Earth Observation Satellite Scheduling Problem. *Mathematics*, 10(19):3498, 2022. doi:10.3390/math10193498.

3 Xiaogeng Chu, Yuning Chen, and Lining Xing. A Branch and Bound Algorithm for Agile Earth Observation Satellite Scheduling. *Discrete Dynamics in Nature and Society*, 2017, 2017. doi:10.1155/2017/7345941.

4 J. F. Cordeau and G. Laporte. Maximizing the Value of an Earth Observation Satellite Orbit. *The Journal of the Operational Research Society*, 56(8):962–968, 2005. doi:10.1057/PALGRAVE.JORS.2601926.

5 Kaikai Cui, Junhua Xiang, and Yulin Zhang. Mission Planning Optimization of Video Satellite for Ground Multi-Object Staring Imaging. *Advances in Space Research*, 61(6):1476–1489, 2018. doi:10.1016/j.asr.2017.10.056.

6 Duncan Eddy and Mykel J. Kochenderfer. A Maximum Independent Set Method for Scheduling Earth-Observing Satellite Constellations. *Journal of Spacecraft and Rockets*, 58(5):1416–1429, 2021. doi:10.2514/1.A34931.

7 Yanxiang Feng, Ruipeng Zhang, Sida Ren, Shuailin Zhu, and Yikang Yang. A Distributed Approach for Time-Dependent Observation Scheduling Problem in the Agile Earth Observation Satellite Constellation. *Remote Sensing*, 15(7):1761, 2023. doi:10.3390/rs15071761.

8 Mohamed Elamine Galloua, Shuai Li, and Jiahao Cui. Earth Observation Satellite Imaging Task Scheduling with Metaheuristics: Multi-Level Clustering and Priority-Driven Pre-Scheduling. *Advances in Space Research*, 2024. doi:10.1016/j.asr.2024.11.023.

9 Lei He, Xiaolu Liu, Gilbert Laporte, Yingwu Chen, and Yingguo Chen. An Improved Adaptive Large Neighborhood Search Algorithm for Multiple Agile Satellites Scheduling. *Computers & Operations Research*, 100:12–25, 2018. doi:10.1016/j.cor.2018.06.020.

10 Lijun He, Ben Liang, Jiandong Li, and Min Sheng. Joint Observation and Transmission Scheduling in Agile Satellite Networks. *IEEE Transactions on Mobile Computing*, 21(12):4381–4396, 2022. doi:10.1109/TMC.2021.3076088.

11 Chae-Hyeon Kim, Sung Jun Kim, and Han-Lim Choi. Optimal Integrated Scheduling of Observation and Download Tasks for Multiple Satellites with Memory Constraints. In *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, pages 2287–2294, 2024. doi:10.1109/CASE59546.2024.10711478.

12 Junhong Kim, Doo-Hyun Cho, Jaemyung Ahn, and Han-Lim Choi. Task Scheduling of Multiple Agile Satellites with Transition Time and Stereo Imaging Constraints, 2019. doi:10.48550/arXiv.1912.00374.

13 Michel Lemaître, Gérard Verfaillie, Frank Jouhaud, Jean-Michel Lachiver, and Nicolas Bataille. Selecting and Scheduling Observations of Agile Satellites. *Aerospace Science and Technology*, 6(5):367–381, 2002. doi:10.1016/S1270-9638(02)01173-2.

14 Rich Levinson, Sreeja Nag, and Vinay Ravindra. Agile Satellite Planning for Multi-Payload Observations for Earth Science, 2021. doi:10.48550/arXiv.2111.07042.

15 Hai Li, Yongjun Li, Yuanhao Liu, Kai Zhang, Xin Li, Yu Li, and Shanghong Zhao. A Multi-Objective Dynamic Mission-Scheduling Algorithm Considering Perturbations for Earth Observation Satellites. *Aerospace*, 11(8):643, 2024. doi:10.3390/aerospace11080643.

16 Longmei Li, Feng Yao, Ning Jing, and Michael Emmerich. Preference Incorporation to Solve Multi-Objective Mission Planning of Agile Earth Observation Satellites. In *Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1366–1373, 2017. doi:10.1109/CEC.2017.7969463.

17 Wei-Chen Lin and Da-Yin Liao. A Tabu Search Algorithm for Satellite Imaging Scheduling. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1601–1606, 2004. doi:10.1109/ICSMC.2004.1399860.

18 Wei-Cheng Lin, Chung-Yang Liu, Da-Yin Liao, and Yung-Yao Lee. Daily Imaging Scheduling of an Earth Observation Satellite. In *Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, volume 2, pages 1886–1891 vol.2, 2003. doi:10.1109/ICSMC.2003.1244686.

19 Lorena Linares, Rafael Vazquez, Federico Perea, and Jorge Galán-Vioque. A Mixed Integer Linear Programming Model for Resolution of the Antenna-Satellite Scheduling Problem. *IEEE Transactions on Aerospace and Electronic Systems*, 60(1):463–473, 2024. doi:10.1109/TAES.2023.3326422.

20 Lihao Liu, Zhenghong Dong, Haoxiang Su, and Dingzhan Yu. A Study of Distributed Earth Observation Satellites Mission Scheduling Method Based on Game-Negotiation Mechanism. *Sensors*, 21(19):6660, 2021. doi:10.3390/s21196660.

21 Xiaolu Liu, Gilbert Laporte, Yingwu Chen, and Renjie He. An Adaptive Large Neighborhood Search Metaheuristic for Agile Satellite Scheduling with Time-Dependent Transition Time. *Computers & Operations Research*, 86:41–53, 2017. doi:10.1016/j.cor.2017.04.006.

22 Guansheng Peng, Guopeng Song, Yongming He, Jing Yu, Shang Xiang, Lining Xing, and Pieter Vansteenwegen. Solving the Agile Earth Observation Satellite Scheduling Problem With Time-Dependent Transition Times. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(3):1614–1625, 2022. doi:10.1109/TSMC.2020.3031738.

23 Vishwa Shah, Vivek Vittaldev, Leon Stepan, and Cyrus Foster. Scheduling the World's Largest Earth-Observing Fleet of Medium-Resolution Imaging Satellites. In *11th International Workshop on Planning and Scheduling for Space (IWPSS)*, page 6, 2019.

24 Samuel Squillaci. *Mission plan optimization for a constellation of Earth observation satellites (Optimisation de plans de mission pour une constellation de satellites d'observation de la Terre)*. PhD thesis, Office national d'études et de recherches aérospatiales (ONERA) and Institut supérieur de l'aéronautique et de l'espace (ISAE-SUPAERO), Toulouse, France, 2023.

25 Samuel Squillaci, Cédric Pralet, and Stéphanie Roussel. Scheduling Complex Observation Requests for a Constellation of Satellites: Large Neighborhood Search Approaches. In *Proceedings of the 20th International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR)*, 2023. doi:10.1007/978-3-031-33271-5_29.

26 Samuel Squillaci, Stéphanie Roussel, and Cédric Pralet. Parallel Scheduling of Complex Requests for a Constellation of Earth Observing Satellites. In *PAIS 2022*, pages 100–113. IOS Press, 2022. doi:10.3233/FAIA220068.

27 Haiquan Sun, Wei Xia, Zhilong Wang, and Xiaoxuan Hu. Agile Earth Observation Satellite Scheduling Algorithm for Emergency Tasks Based on Multiple Strategies. *Journal of Systems Science and Systems Engineering*, 30(5):626–646, 2021. doi:10.1007/s11518-021-5506-4.

28 Panwadee Tangpattanakul, Nicolas Jozefowiez, and Pierre Lopez. A Multi-Objective Local Search Heuristic for Scheduling Earth Observations Taken by an Agile Satellite. *European Journal of Operational Research*, 245(2):542–554, 2015. doi:10.1016/j.ejor.2015.03.011.

29 Christopher G. Valicka, Deanna Garcia, Andrea Staid, Jean-Paul Watson, Gabriel Hackebeil, Sivakumar Rathinam, and Lewis Ntaimo. Mixed-Integer Programming Models for Optimal Constellation Scheduling given Cloud Cover Uncertainty. *European Journal of Operational Research*, 275(2):431–445, 2019. doi:10.1016/j.ejor.2018.11.043.

30 Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. Iterated Local Search for the Team Orienteering Problem with Time Windows. *Computers & Operations Research*, 36(12):3281–3290, 2009. doi:10.1016/j.cor.2009.03.008.

31 Petr Vilím, Philippe Laborie, and Paul Shaw. Failure-Directed Search for Constraint-Based Scheduling. In *Proceedings of the 12th International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR)*, pages 437–453, 2015. doi:10.1007/978-3-319-18008-3_30.

32 Jianjiang Wang, Erik Demeulemeester, and Dishan Qiu. A Pure Proactive Scheduling Algorithm for Multiple Earth Observation Satellites under Uncertainties of Clouds. *Computers & Operations Research*, 74:1–13, 2016. doi:10.1016/j.cor.2016.04.014.

33 Xinwei Wang, Guopeng Song, Roel Leus, and Chao Han. Robust Earth Observation Satellite Scheduling With Uncertainty of Cloud Coverage. *IEEE Transactions on Aerospace and Electronic Systems*, 56(3):2450–2461, 2020. doi:10.1109/TAES.2019.2947978.

34 Xinwei Wang, Guohua Wu, Lining Xing, and Witold Pedrycz. Agile Earth Observation Satellite Scheduling Over 20 Years: Formulations, Methods, and Future Directions. *IEEE Systems Journal*, 15(3):3881–3892, 2021. doi:10.1109/JSYST.2020.2997050.

35 Damien Wojtowicz. AEOSSP Instance Generator. Dataset, France Relance grant (project JAPETUS), swhId: swh:1:dir:c1c87cf9c3a3e7ffe616bdffdc9c52af9e30cb35 (visited on 2025-07-24). URL: https://github.com/ssquilla/Earth_Observing_Satellites_benchmarks, doi:10.4230/artifacts.24101.

36 Fatos Xhafa and Andrew W.H. Ip. Optimisation Problems and Resolution Methods in Satellite Scheduling and Space-Craft Operation: A Survey. *Enterprise Information Systems*, 15(8):1022–1045, 2021. doi:10.1080/17517575.2019.1593508.

37 Wenyuan Zhang and Gangtie Zheng. Scheduling an Agile Multipayload Earth-Observing Satellite. *Journal of Spacecraft and Rockets*, 61(1):143–156, 2024. doi:10.2514/1.A35726.

A Instances Details

Table 6 Generated instances details. It shows, for each instance, its set of POIs, the number of requests $|\mathcal{R}|$, of periodic requests P, of multi-mesh requests M, of downloads $|\mathcal{W}|$, of acquisition opportunities $|\mathcal{A}|$, of acquisitions needed to satisfy all the requests $s(\mathcal{A})$, of connected components nc, the average size of the components sc, and the average number of acquisition opportunities per target ar.

Set	$\|\mathcal{R}\|$	P	M	$\|\mathcal{W}\|$	$\|\mathcal{A}\|$	$s(\mathcal{A})$	nc	sc	ar
antilles	50	11	2	60	1017	70	19	54	15
	100	18	4	60	1993	133	19	105	15
	250	54	17	60	5313	370	19	280	14
	500	88	29	60	10510	717	19	553	15
sahara	50	3	4	74	872	59	22	40	15
	100	18	10	74	1857	147	23	81	13
	250	43	19	89	4455	361	23	193	13
	500	89	39	74	9043	721	23	393	13
fountains	50	9	3	82	1692	67	31	55	25
	100	14	4	82	3273	133	32	102	25
	250	51	14	82	8372	359	32	262	23
	500	112	33	82	17037	748	32	532	23
military	50	10	6	159	1503	83	104	14	18
	100	22	2	164	2648	140	108	24	19
	250	48	4	165	6735	347	141	48	19
	500	97	40	163	14039	741	163	86	19
world heritage	50	4	4	158	1729	67	119	14	26
	100	17	1	160	2952	130	135	22	23
	250	40	20	164	8153	360	136	59	23
	500	85	38	166	15979	717	133	120	22

Table 7 Squillaci et al. instances details. It shows, for each instance, the number of request $|\mathcal{R}|$, of one-shot requests OS, video requests V, stereoscopic requests S and periodic requests P, of downloads $|\mathcal{W}|$, of acquisition opportunities $|\mathcal{A}|$, of acquisitions needed to satisfy all the requests s($\mathcal{A}$), of connected components nc, the average size of the components sc(when $sc > 1$), and of acquisition opportunities per target ar. The first part of this table is about the concentrated instances and the second part is about the spread instances.

$\|\mathcal{R}\|$	OS	V	S	P	$\|\mathcal{W}\|$	$\|\mathcal{A}\|$	s($\mathcal{A}$)	nc	sc	ar
50	50	0	0	0	123	2607	50	124	23	50
250	250	0	0	0	123	13149	250	127	112	52
500	500	0	0	0	123	25646	500	116	221	51
1000	1000	0	0	0	123	51189	1000	117	437	51
50	0	50	0	0	123	2436	50	115	21	46
250	0	250	0	0	123	12162	250	114	107	48
500	0	500	0	0	123	25554	500	118	216	51
1000	0	1000	0	0	123	49149	1000	116	424	49
50	0	0	0	50	123	576	150	26	18	3
250	0	0	0	250	123	2367	750	25	91	3
500	0	0	0	500	123	4794	1501	28	168	3
1000	0	0	0	1000	123	9777	3000	27	359	3
57	12	15	27	3	123	2125	90	112	19	22
285	60	75	135	1	123	10327	450	115	90	23
570	120	150	270	30	123	19203	900	117	164	21
1140	240	300	540	60	123	45985	1800	111	414	25
50	50	0	0	0	123	2438	50	115	21	46
250	250	0	0	0	123	12269	250	118	104	49
500	500	0	0	0	123	24475	500	119	206	49
1000	1000	0	0	0	123	49035	1000	121	405	50
50	0	50	0	0	123	2427	50	115	21	46
250	0	250	0	0	123	12190	250	118	103	48
500	0	500	0	0	123	24326	500	119	204	48
1000	0	1000	0	0	123	48686	1000	120	406	49
50	0	0	0	50	123	568	150	27	17	3
250	0	0	0	250	123	2348	750	27	83	3
500	0	0	0	500	123	4594	1501	28	161	3
1000	0	0	0	1000	123	9162	3004	28	324	3
57	12	15	27	3	123	2025	90	115	18	21
285	60	75	135	15	123	10467	451	119	88	23
570	120	150	270	30	123	20820	900	118	176	23
1140	240	300	540	60	123	41693	1800	119	350	23

B Results Tables

Table 8 shows the profit and the gap yielded by both methods on Squillaci et al. instances, while Table 8shows it on our instances, both using a fail limit of 50 for Tempo. The gap was computed using an upper bound calculated with the CPO model. Thanks to a naive upper bound (sum of the profit of the best opportunities for each request), the solver can prove optimality on trivial instances.

Over Squillaci et al. instances, `LS-tempo` has better results on all instances but 2, with an improvement of 0.9 % gap in average. Averaging the gap make the differences between the two solvers smaller, as 15 instances over the 32 are solved optimal with the base solver (see Table 4 for data on a subset of instances). We can see the number of restart drops with the

number of requests, especially for instances with periodic requests only. The same remark holds for the Tempo calls success rate. We note that for instance with 1000 video requests with concentrated target, tempo has only 4 % of success, nevertheless the result of `LS-tempo` is much better than `Greedy`.

Over our set of instances, the use advantage of using Tempo is not as clear. At first glance, it seems that there is a difference in the results between regional and worldwide instances. Using Tempo always seems to give the best solution, but the difference are in fact very small and the average gap for `LS-tempo` and `Greedy` are the same. Compared to the other instance set the number of restart and calls to Tempo is very high. On the contrary the success rate is very low, especially on regional datasets. It seems that instance of the sub-problem are either easily solve with the greedy insertion, or are very hard and cannot be solved with a small fail limit.

Table 8 Profit, gap and number of restarts (r) found using `LS-tempo` and `Greedy` over our instances. Average values over 5 runs. A star (*) means the profit is optimal, a dash ("–") means the instance was solved optimally during the initial descent.

$\|\mathcal{R}\|$	OS	V	S	P	Greedy			LS-tempo				
					profit	gap	r	profit	gap	r	t	s
50	50	0	0	0	**390***	0.00	–	**390***	0.00	–	–	–
250	250	0	0	0	**1655***	0.00	–	**1655***	0.00	–	12	100%
500	500	0	0	0	**5070***	0.00	–	**5070***	0.00	–	26	100%
1000	1000	0	0	0	11440	0.02	601	**11536**	0.01	322	68427	79%
50	0	50	0	0	**4060***	0.00	–	**4060***	0.00	–	1	100%
250	0	250	0	0	**7440***	0.00	–	**7440***	0.00	–	13	100%
500	0	500	0	0	21040	0.01	3054	**21200**	0.01	1841	98269	89%
1000	0	1000	0	0	51524	0.22	441	**55367**	0.16	76	1439698	4%
50	0	0	0	50	**780***	0.00	11	**780***	0.00	–	3	100%
250	0	0	0	250	3777	0.11	9475	**3946**	0.07	39	37802	14%
500	0	0	0	500	10120	0.18	4997	**10512**	0.15	5	36189	5%
1000	0	0	0	1000	**13996**	0.36	1992	13953	0.36	1	34562	1%
57	12	15	27	3	**2010***	0.00	–	**2010***	0.00	–	1	100%
285	60	75	135	1	**6265***	0.00	3	**6265***	0.00	–	21	100%
570	120	150	270	30	22092	0.03	2414	**22585**	0.01	524	97259	60%
1140	240	300	540	60	16126	0.01	234	**16220**	<0.01	17	24299	36%
50	50	0	0	0	**720***	0.00	3	**720***	0.00	–	4	100%
250	250	0	0	0	**4625***	0.00	81	**4625***	0.00	–	15	100%
500	500	0	0	0	9138	0.01	1952	**9267**	<0.01	870	64068	81%
1000	1000	0	0	0	17833	0.02	444	**18055**	0.01	214	66872	69%
50	0	50	0	0	**2880***	0.00	7	**2880***	0.00	–	2	100%
250	0	250	0	0	**18500***	0.00	65	**18500***	0.00	7	137	89%
500	0	500	0	0	35416	0.04	3103	**36420**	0.02	1443	223688	65%
1000	0	1000	0	0	66660	0.08	317	**68564**	0.06	155	436114	17%
50	0	0	0	50	**1060***	0.00	–	**1060***	0.00	–	4	100%
250	0	0	0	250	6139	0.05	9243	**6388**	0.01	82	31641	41%
500	0	0	0	500	11177	0.17	4397	**11662**	0.13	5	28955	7%
1000	0	0	0	1000	**18078**	0.32	1964	18055	0.32	1	28728	2%
57	12	15	27	3	**1765***	0.00	–	**1765***	0.00	–	1	100%
285	60	75	135	15	**12210***	0.00	24	**12210***	0.00	1	43	95%
570	120	150	270	30	22048	0.01	2468	**22187**	<0.01	701	110152	84%
1140	240	300	540	60	42908	0.04	480	**43899**	0.02	105	85822	62%
				Average		0.053			0.042			67.6%

Table 9 Profit, gap and number of restarts (r) found using `LS-tempo` and `Greedy` over our instances. Average values over 5 runs. A star (*) means the profit is optimal, a dash ("-") means the instance was solved optimally during the initial descent.

POIs	$\|\mathcal{R}\|$	Greedy			LS-tempo				
		profit	gap	r	profit	gap	r	t	s
antilles	50	**20090**	<0.01	167166	**20090**	<0.01	13186	3055040	6%
	100	36307	0.02	82066	**36328**	0.02	2396	2296988	4%
	250	**74129**	0.27	22835	74035	0.27	255	2263862	1%
	500	**88913**	0.56	10785	88222	0.57	119	1983807	0%
sahara	50	17736	0.01	242973	**17743**	0.01	11743	1785669	13%
	100	**38575**	0.03	79356	38570	0.03	1504	1400336	4%
	250	75919	0.30	19381	**76020**	0.30	212	1898979	1%
	500	**86522**	0.56	10690	85559	0.57	85	1532961	0%
fountains	50	20282	0.01	122542	**20287**	0.01	10309	1986404	7%
	100	36989	0.02	39253	**37014**	0.02	1806	1824849	4%
	250	97433	0.09	8422	**97721**	0.08	112	891194	2%
	500	**142314**	0.36	3564	141331	0.37	36	1342948	0%
military	50	**22467***	0.00	44987	**22467***	0.00	30372	585242	19%
	100	38488	<0.01	15250	**38489***	0.00	9135	418030	29%
	250	102838	<0.01	3742	**102868**	<0.01	391	324473	12%
	500	211019	0.01	1116	**211080**	0.01	35	186169	6%
world heritage	50	18221	<0.01	23588	**18222***	0.00	19290	760620	24%
	100	36684	<0.01	13974	**36691**	<0.01	4706	1214798	13%
	250	103770	0.01	2714	**103860**	0.01	310	717050	6%
	500	197736	0.03	1040	**197782**	0.03	39	366371	3%
	Average		0.115			0.115			7.7%