Managing complex requests for a constellation of Earth observing satellites

Samuel Squillaci, Stéphanie Roussel, Cédric Pralet

Managing complex requests for a constellation of Earth observing satellites

Samuel Squillaci, Stéphanie Roussel, Cédric Pralet
[1]ONERA/DTIS, Université de Toulouse
F-31055 Toulouse, France
{samuel.squillaci,stephanie.roussel,cedric.pralet}@onera.fr

Abstract

Nowadays, the development of Earth observation systems involving multiple satellites and multiple ground stations allows to satisfy in parallel many observation requests formulated by end-users. One difficulty though is that in this context, the mission planning system must often manage various kinds of observation requests (one-shot requests, periodic requests, *etc.*), and various kinds of requirements over each request (collect data as quickly as possible, download data only towards specific stations, transmit stereoscopic observations during the same downlink window, *etc.*). This paper introduces a generic approach to deal with such complex requests, based on so-called *modes*, where each mode explicitly defines a set of observation and download activities that allows a request to be fulfilled. The goals are then to select a subset of requests among the candidate ones, to select a mode for each of these requests, and to schedule all activities covered by the selected modes. For this, the paper first introduces a MILP model and then a fast incomplete search procedure based on iterative request insertion and plan repair. The experiments performed show that the approach scales well on realistic instances involving 16 or 32 satellites.

1 Introduction

In this paper, we consider an Earth observation system under development that will be composed of about 30 satellites, one objective being to decrease the duration between two possible successive visits of a given ground area. The system will receive observation requests coming from several end-users, some of which might have their own reception stations in addition to the main ground station facilities. For this new system, a key point concerns the development of a mission planning tool taking as an input a set of candidate imaging requests and returning as an output a mission plan for each satellite over the next time period.

Several techniques were proposed in the last decade to define such an observation selection and scheduling tool for a satellite constellation. Some authors proposed to use a unique global optimization problem to define the activities of the satellites. Such approaches are based on *Branch & Price* (Han et al. 2018-12-01; Hu 2019) or MILP (Kim et al. 2019; Xiao et al. 2019). Several local search based methods were also tested such as tabu search (Bianchessi et al. 2007), evolutionary algorithms (Wang et al. 2015; Eddy and Kochenderfer 2020), or simulated annealing (Vongsantivanich et al. 2018). Pareto Search is also considered when multiple objective functions must be optimized (Sun 2012; Li and Li 2019). Other approaches explicitly identify on one hand an assignment problem that consists in determining the satellite that should be used to perform a given request, and on the other hand a scheduling problem that consists in ordering all activities assigned to a given satellite. For the assignment part, a first approach selects at each step an observation task that maximizes a given scoring function and assigns this task to a satellite whose workload is the lowest (Dishan et al. 2013). Another approach assigns tasks to satellites based on randomized heuristics that take into account various features (size of the observation windows, Nadir position, *etc.*), and couples this process with an Adaptive Large Neighborhood Search (ALNS) that uses destroy and repair operations to try and decrease the number of unscheduled tasks (He et al. 2018). A last approach assigns each task to the satellite that offers the highest scheduling success probability according to a neural network success prediction model, and then performs tabu search to build a plan from the task assignment strategy (Du et al. 2020). Finally, strategies for on-line planning have also been studied. An approach consists in using heuristics for measuring the difficulty of inserting an observation in the plan of a given satellite (He et al. 2019).

With regards to these previous works, one issue is that for the system under development that we consider, the central mission planner must be able to deal with observation requests that are more complex than those encountered in the literature. More specifically, this planner must be able to deal with several kinds of requests that are relevant for the end-users, including standard one-shot simple observation requests, one-shot stereoscopic requests (two observations of a given ground area, using a forward and a backward pointing respectively), periodic imaging requests (observation every X hours over a given area), periodic and stereoscopic imaging requests, *etc.*

The mission planning system to define must also deal with various constraints and preferences on the way observation and data downlink should be performed. For instance, there

might be constraints on the choice of reception stations for downloading collected data. Also, for stereoscopic requests requiring two observations, there might be a need to download the two associated data files during the same downlink window. As a result, in addition to standard constraints holding on each individual activity, there can be constraints holding over several activities composing a given request. The same remark holds for preferences in the sense that there might be preferences holding on individual activities (*e.g.*, preference related to cloud cover conditions) or preferences holding on multiple activities (preference for a minimum temporal distance between observation and data transmission, preference for the smallest deviation from the period for a periodic observation request, *etc.*).

One contribution of this paper in the frame of Earth Observation Satellites (EOSs) consists in the use of so-called *request modes* to manage such a variety of requirements. Basically, each mode corresponds to a set of observation and download activities that allow to cover a given request while satisfying the request observation and download constraints. To cover complex preferences, a reward is also associated with each mode to measure the quality of service obtained for the request with that mode. This reward can aggregate features related to cloud cover forecast or preferences over the temporal distance between activities covered by the mode. One key advantage of this approach is that instead of having to explicitly manage user constraints and preferences, the generic global planning system only needs to reason about the modes and the constraints expressing the physical capabilities of each satellite (kinematic constraints and resource constraints). Doing so, it is easy to integrate new kinds of requests if needed during the lifetime of the system, without changing the core mission planner.

The basic idea of using modes for describing alternatives to carry out an activity is not new in the scheduling domain. For instance, in *Multi-mode Resource Constraint Scheduling Problems* (MRCPSP (Talbot 1982; Coelho and Vanhoucke 2015)), the goal is to schedule a set of activities over a set of machines that have limited capacities, given that each activity has a set of possible modes. Each mode requires using a subset of the machines during a certain duration and with a certain consumption level for each machine. In MRCPSP, the activities to perform are however not as complex as observation requests encountered for EOSs. In another direction, the concept of *switch groups* recently introduced for the Mars 2020 mission also allows to defined several ways of performing an activity (Agrawal et al. 2021). Each switch group consists of a set of execution alternatives that consume more or less time and resources and that are more or less preferred, and switch groups can be very quickly used onboard to adapt the plan to the real context.

The rest of the paper is organized as follows. Section 2 describes the problem considered and formalizes the request modes. Section 3 defines a MILP model for this problem. Section 4 introduces a fast incomplete search strategy based on iterative request insertion and plan repair. Section 5 provides experimental results on scenarios involving heterogeneous requests to be performed by 16 or 32 satellites. Section 6 gives some perspectives for this work.

2 Problem description

We consider a constellation of Low Earth Orbit satellites that can perform observations and download collected data to ground reception stations. The optical instrument and the data emission antenna are body-mounted on each satellite, meaning that during each observation, a satellite must be pointed to the associated ground target, and during a data download activity, it must be pointed to a reception station. As a result, acquisition and data transfer cannot be performed in parallel. In the following, we denote by $\mathcal{S}$ the set of satellites, and the objective is to plan the activities of the satellites over a temporal horizon $[0, H]$.

2.1 Observation requests

At a given planning phase, the mission centre considers a set of observation requests $\mathcal{R}$. Each request $r \in \mathcal{R}$ is associated with a specific ground area. Satisfying a request means (1) performing observations over the corresponding ground area based on the available *observation opportunities*, (2) downloading observation data towards ground reception stations based on the available *download opportunities*. As explained later, we consider a set of alternatives to decompose each request into basic observation and download activities, each alternative being referred to as a *mode* (more details below).

2.2 Observation opportunities

For each satellite $s \in \mathcal{S}$, there is a set of observation opportunities $\mathcal{O}_s$ over relevant ground targets. Each observation opportunity $o \in \mathcal{O}_s$ is defined by:

- a window $[Start_o, End_o]$ during which the ground target is visible from satellite s;
- an observation duration $\Delta_o \in]0, End_o - Start_o]$;
- a download duration $\Theta_o \in \mathbb{R}^+$, representing the time required for downloading data collected during opportunity o if the latter is used;
- a memory usage $Mem_o \in \mathbb{N}$, representing the memory space used by the data collected if an observation is actually performed during opportunity o.

2.3 Download opportunities

For each satellite $s \in \mathcal{S}$, there is a set of download opportunities $\mathcal{D}_s$. Each download opportunity $d \in \mathcal{D}_s$ corresponds to a time window $[Start_d, End_d]$ during which data can be transmitted to a ground reception station.

2.4 Past observations

The activities of satellites are regularly planned over a rolling horizon. At a given step, for each satellite $s \in \mathcal{S}$, there exists a set of past observations $\mathcal{P}_s$ that are stored onboard but not downloaded to a ground station yet. These past observations must be taken into account when planning the activities of the satellites over the next scheduling period. Each past observation $o \in \mathcal{P}_s$ is defined by:

- a download duration $\Theta_o \in \mathbb{R}^+$, representing the time required for downloading o;

- a memory usage $Mem_o \in \mathbb{N}$, representing the memory space used for storing o onboard the satellite.

2.5 Requests modes

We now describe the link between observation requests, observation opportunities, download opportunities, and past observations. Basically, for each observation request $r \in \mathcal{R}$, there exists a set of candidate modes $\mathcal{M}_r$. Each mode represents an alternative for completing request r. A mode is a set of pairs (o, d) where:

- o is an observation opportunity or a past observation contributing to request r;
- d is a download opportunity to be used for downloading data associated with o (with the assumption that d is associated with the same satellite as o).

More formally, each mode $m \in \mathcal{M}_r$ is a set of pairs $(o, d) \in \bigcup_{s \in \mathcal{S}} ((\mathcal{O}_s \cup \mathcal{P}_s) \times \mathcal{D}_s)$, with the assumption that an observation opportunity or a past observation o does not appear more than once in a single mode (contrarily to a download opportunity, that can be used for several observations). Note that an observation or download opportunity can be part of several modes for a given request.

A request r is completed as soon as all observation and download opportunities associated with one mode $m \in \mathcal{M}_r$ are used for collecting and transmitting data for r. Figure 1 illustrates modes for a toy example involving two satellites s_1 and s_2. Satellite s_1 has two observation opportunities (o_1 and o_2) and one download opportunity (d_1). Satellite s_2 has three observation opportunities (o_3, o_4 and o_5), two download opportunities (d_2 and d_3) and one past observation (p_1). We consider two requests r_1 and r_2 that have two modes each:

$$r_1 \begin{cases} m_1 = \{(o_1, d_1)\} \\ m_2 = \{(o_4, d_3)\} \end{cases} \quad r_2 \begin{cases} m_3 = \{(p_1, d_2), (o_2, d_1), (o_3, d_2)\} \\ m_4 = \{(p_1, d_2), (o_2, d_1), (o_5, d_3)\} \end{cases}$$

This means that to satisfy request r_1, it is possible (1) either to perform an observation within observation opportunity o_1 and download the associated data within download opportunity d_1, (2) or to perform an observation within opportunity o_4 and download the associated data within opportunity d_3. Similarly, to satisfy request r_2, (1) past observation p_1 must be downloaded during opportunity d_2, (2) an observation must be performed during opportunity o_2 and downloaded during opportunity d_1, and (3) one of the observation-download pairs (o_3, d_2), (o_5, d_3) must be used. As mentioned in the introduction, the concept of mode allows to cover one-shot and periodic requests, monoscopic and stereoscopic observations, and more generally any constraint over the observation-download pattern.

Last, a reward $\omega_m \in \mathbb{R}^+$ is associated with each mode $m \in \mathcal{M}_r$. Such a reward can cover any preferences on features like the weather conditions for the observations, the time at which the observation and download activities take place, the ground station to which data is downloaded, *etc.* Compared to many models used in the literature, attaching the rewards to modes and not to the basic observation and download activities is more expressive, since the reward of a request might not be the sum of the rewards associated with its basic activities. One typical example is given by periodic observations that should be separated by a temporal distance that is neither to low nor to high.

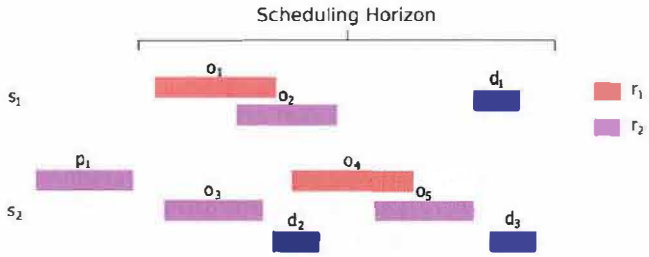


Figure 1: Toy example involving 2 satellites, 2 requests, 5 observation opportunities, 3 download opportunities, and 1 past observation

A potential drawback of the approach is however that the number of possible modes for a given request might be quite large. For this, we can consider only the most relevant combinations of observation-download pairs for each request, and new modes might be generated on-the-fly if needed.

2.6 Maneuver model

Each satellite s performs a sequence of observation and download activities. The temporal distance between successive activities must be consistent with the transition time required by s to maneuver between the successive target pointings. To model this point, let us denote by $\mathcal{A}_s$ the set of activity opportunities available for satellite s ($\mathcal{A}_s = \mathcal{O}_s \cup \mathcal{D}_s$). Then, for each pair of activities $(a, b) \in \mathcal{A}_s^2$, we define a minimum transition duration $\tau_{a,b} \in \mathbb{R}^+$ that represents the minimum maneuver time from the satellite attitude required for performing a to the satellite attitude required for performing b.

The previous definition can be extended to the case where a is equal to a fictitious activity α_s placed at the beginning of the plan of s, and to the case where b is equal to a fictitious activity β_s placed at the end of the plan of s.

2.7 Memory resource model

At any time, the total mass memory used by a satellite must not exceed a limit capacity $MemCap_s \in \mathbb{N}$. More formally, for each satellite s and each time t over the scheduling horizon, the total memory used by the observations that have been fully performed before t but not fully downloaded yet at t must not exceed $MemCap_s$. For the system considered, this capacity is not the main bottleneck of the problem.

2.8 Solution plan

A solution plan σ is defined by:

- a subset $\mathcal{R}(\sigma) \subseteq \mathcal{R}$ of requests that are selected in σ;
- for each request $r \in \mathcal{R}(\sigma)$, a mode $m_r(\sigma)$ chosen for r in σ;
- for each satellite $s \in \mathcal{S}$, a sequence of opportunities used $\Pi_s(\sigma) = [\alpha_s, \Pi_{s,1}, \ldots, \Pi_{s,k}, \beta_s]$, where each opportunity $\Pi_{s,i}$ belongs to $\mathcal{A}_s$, and where the set of activities

involved in sequences $\Pi_s(\sigma)$ is consistent with the set of opportunities covered by the modes selected for the requests in $\mathcal{R}(\sigma)$.

From this, the earliest start time of the first activity $a = \Pi_{s,1}$ of a given satellite s is given by $earliest_a(\sigma) = Start_a$, and the earliest start time of an activity $a = \Pi_{s,i}$ that follows an activity $b = \Pi_{s,i-1}$ is recursively computed by:

$$earliest_a(\sigma) = \max(Start_a, earliest_b(\sigma) + dur_b(\sigma) + \tau_{b,a})$$

where $dur_b(\sigma)$ stands for the duration of activity b (either the observation duration, when b corresponds to an observation opportunity, or the sum of the download duration for data downloaded within b, when b corresponds to a download opportunity).

A plan is said to be *feasible* if and only if, for each satellite,

- data observation precedes data download for every observation-download pair (o, d) covered by a mode used;
- the mass memory capacity constraint is satisfied;
- each activity a can be finished within its allocated time window ($earliest_a(\sigma) + dur_a(\sigma) \leq End_a$).

A feasible plan σ is said to be *optimal* if it maximizes the reward $\omega(\sigma)$ defined as the sum of the rewards provided by the selected request modes ($\omega(\sigma) = \sum_{r \in \mathcal{R}(\sigma)} \omega_{m_r(\sigma)}$). The problem obtained is referred to as an EOSCPM problem (*EOS Constellation Planning with Modes*).

3 MILP model

In this section, we provide a MILP model that can be used to find optimal solutions to small EOSCPM instances.

Additional notations To make the modeling easier, we introduce a few additional notations:

- $\mathcal{O} = \bigcup_{s \in \mathcal{S}} \mathcal{O}_s$ denotes the set of observation opportunities;
- $\mathcal{D} = \bigcup_{s \in \mathcal{S}} \mathcal{D}_s$ denotes the set of download opportunities;
- $\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}_s$ denotes the set of activity opportunities;
- $\mathcal{P} = \bigcup_{s \in \mathcal{S}} \mathcal{P}_s$ denotes the set of past observations;
- $\mathcal{M} = \bigcup_{r \in \mathcal{R}} \mathcal{M}_r$ denotes the set of request modes;
- for each activity opportunity $a \in \mathcal{A}_s$ for a satellite s, we denote by $\mathcal{A}_a^+$ (resp. $\mathcal{A}_a^-$) the set of activity opportunities that can be performed just after (resp. just before) a in sequence Π_s. Formally, $\mathcal{A}_a^+ = \{b \in \mathcal{A}_s | Start_a + \tau_{a,b} \leq End_b\}$ and $\mathcal{A}_a^- = \{b \in \mathcal{A}_s | Start_b + \tau_{b,a} \leq End_a\}$; on the example provided in Figure 1, if we suppose that the transition time between activities is null, $\mathcal{A}_{o_4}^+ = \{d_2, o_5, d_3\}$ and $\mathcal{A}_{o_4}^- = \{o_3, d_2, o_5\}$;
- for each mode m, $\mathcal{O}_m$ and $\mathcal{D}_m$ denote the sets of observation and download opportunities involved in mode m respectively. Formally, $\mathcal{O}_m = \{o \in \mathcal{O} | \exists (o, d) \in m\}$ and $\mathcal{D}_m = \{d \in \mathcal{D} | \exists (o, d) \in m\}$;
- for a mode $m \in \mathcal{M}$ and a satellite $s \in \mathcal{S}$, m_s is the set of observation-download pairs (o, d) in m that are associated with satellite s. Formally, $m_s = \{(o, d) \in m | o \in \mathcal{O}_s \cup \mathcal{P}_s, d \in \mathcal{D}_s\}$.

Variables. We introduce the following variables:

- $\forall r \in \mathcal{R}, \forall m \in \mathcal{M}_r$, $\mathbf{y}_m \in \{0, 1\}$ specifies whether mode m is selected for performing request r;
- $\forall a \in \mathcal{A}$:
  - $\mathbf{z}_a \in \{0, 1\}$ takes value 1 if and only if activity opportunity a is used;
  - $\mathbf{t}_a \in [Start_a, End_a]$ represents the start time of the activity performed during opportunity a;
  - $\mathbf{dur}_a \in [0, End_a - Start_a]$ represents the duration of the activity performed within opportunity a;
- $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}_s \cup \{\alpha_s\}, \forall b \in \mathcal{A}_a^+ \cup \{\beta_s\}$, $\mathbf{x}_{a,b} \in \{0, 1\}$ indicates whether b is the successor of a in the sequence of activity opportunities used for s.

MILP model The MILP model is given in Equations 1 to 15. The objective function (Eq. 1) consists in maximizing the global reward provided by the selected modes. Constraint 2 imposes that at most one mode can be selected for each request. Constraint 3 imposes that each activity must be finished before the end of its associated time window. The duration of an observation is equal to the duration given in the input data (Eq. 4). The download duration within a download opportunity d corresponds to the sum of download durations for observation opportunities and past observations that are associated with d in the selected modes (Eq. 5). Note that contrarily to observation activities, the duration of the download activities is not fixed beforehand. Constraints 6-7 enforce the consistency between the presence of activities in the plan ($\mathbf{z}_a$ variables) and the sequences of activities used for each satellite ($\mathbf{x}_{a,b}$ variables). Constraints 8-9 impose that the initial and final activities for each satellite s are α_s and β_s. Constraint 10 expresses that an observation opportunity is present if and only one mode containing it is selected. Constraints 11-12 ensure the consistency between the presence of download opportunities in the plan ($\mathbf{z}_a$ variables) and the selection of modes ($\mathbf{y}_m$ variables): if a mode is selected, then every download opportunity of this mode is present, and a download opportunity is present in the plan only if at least one mode containing it is selected. Constraint 13 expresses that the start and end dates of two successive activities must be consistent with the minimum maneuver times (big-M formulation using constant $\Gamma_{a,b} = End_a + \tau_{a,b} - Start_b$ to ensure that the constraint is inactive when $\mathbf{x}_{a,b}$ takes values 0). Constraint 14 imposes that observation must precede data download (big-M formulation using constants $\Gamma_{a,b}$ again to ensure that the constraint is inactive when $\mathbf{y}_m$ takes values 0). Constraint 15 models the memory capacity limitation for each satellite: for each download opportunity d, the memory required to store all observations possibly performed but possibly not downloaded yet must not exceed the mass memory capacity. Checking the memory capacity between two successive download slots is useless since the memory consumption peeks are necessarily located at the beginning or during download windows. Note that this constraint is conservative as it uses dates of visibility windows of activities instead of start and end dates decided in the plan.

$$\textit{maximize} \sum_{r \in \mathcal{R}} \sum_{m \in \mathcal{M}_r} \omega_m \mathbf{y}_m \quad (1)$$

subject to :

$$\forall r \in \mathcal{R}, \quad \textstyle\sum_{m \in \mathcal{M}_r} \mathbf{y}_m \leq 1 \quad (2)$$

$$\forall a \in \mathcal{A}, \quad \mathbf{t}_a + \mathbf{dur}_a \leq End_a \quad (3)$$

$$\forall o \in \mathcal{O}, \quad \mathbf{dur}_o = \Delta_o \quad (4)$$

$$\forall d \in \mathcal{D}, \quad \mathbf{dur}_d = \textstyle\sum_{m \in \mathcal{M}} \mathbf{y}_m \sum_{(o,d) \in m} \Theta_o \quad (5)$$

$$\forall s \in \mathcal{S}, \forall a \in \mathcal{A}_s, \quad \textstyle\sum_{b \in \mathcal{A}_a^+ \cup \{\beta_s\}} \mathbf{x}_{a,b} = \mathbf{z}_a \quad (6)$$

$$\forall s \in \mathcal{S}, \forall a \in \mathcal{A}_s, \quad \textstyle\sum_{b \in \mathcal{A}_a^- \cup \{\alpha_s\}} \mathbf{x}_{b,a} = \mathbf{z}_a \quad (7)$$

$$\forall s \in \mathcal{S}, \quad \textstyle\sum_{a \in \mathcal{A}_s \cup \{\beta_s\}} \mathbf{x}_{\alpha_s,a} = 1 \quad (8)$$

$$\forall s \in \mathcal{S}, \quad \textstyle\sum_{a \in \mathcal{A}_s \cup \{\alpha_s\}} \mathbf{x}_{a,\beta_s} = 1 \quad (9)$$

$$\forall o \in \mathcal{O}, \quad \mathbf{z}_o = \textstyle\sum_{m \in \mathcal{M}_{r_o}} \mathbf{y}_m \quad (10)$$

$$\forall m \in \mathcal{M}, \forall d \in \mathcal{D}_m, \quad \mathbf{z}_d \geq \mathbf{y}_m \quad (11)$$

$$\forall d \in \mathcal{D}, \quad \mathbf{z}_d \leq \textstyle\sum_{m \in \mathcal{M} | d \in \mathcal{D}_m} \mathbf{y}_m \quad (12)$$

$$\forall a \in \mathcal{A}, \forall b \in \mathcal{A}_a^+ \text{ s.t. } \Gamma_{a,b} > 0,$$
$$\mathbf{t}_a + \mathbf{dur}_a + \tau_{a,b} \leq \mathbf{t}_b + \Gamma_{a,b}(1 - \mathbf{x}_{a,b}) \quad (13)$$

$$\forall m \in \mathcal{M}, \forall (o,d) \in m \text{ s.t. } o \in \mathcal{O} \text{ and } \Gamma_{o,d} > 0,$$
$$\mathbf{t}_o + \Delta_o + \tau_{o,d} \leq \mathbf{t}_d + \Gamma_{o,d}(1 - \mathbf{y}_m) \quad (14)$$

$$\forall s \in \mathcal{S}, \forall d \in \mathcal{D}_s,$$
$$\sum_{m \in \mathcal{M}} \mathbf{y}_m \sum_{\substack{(o,d') \in m_s \\ End_{d'} > Start_d \\ Start_o + \Delta_o \leq End_d \text{ if } o \in \mathcal{O}}} Mem_o \leq MemCap_s \quad (15)$$

4 Iterative insertion and repair

The MILP model defined in the previous section can be used as a reference approach on instances containing a few requests. For realistic instances however, there is a need for algorithms that can quickly deliver good-quality solutions. This is why we introduce a fast iterative search procedure. The latter starts from an empty plan and tries to insert at each step one more request r into the plan, using the best candidate mode left for r. Requests are inserted until the solution becomes infeasible. In this case, the solution is repaired either by changing the ordering of activities over some satellites, or by removing requests from the current plan. This repair procedure is applied until the solution becomes feasible again. To choose a request to insert and remove at each step, the algorithm respectively uses an insertion heuristic and a repair heuristic. On top of that, a perturbation phase is introduced to try and find better solutions. The search scheme obtained can be seen as a Large Neighborhood Search (LNS) that alternates between constructive and destructive search phases. We provide below a more detailed description of each component of the algorithm. The approach directly searches for a plan in the form proposed in Section 2.8 (no manipulation of the MILP variables defined before).

4.1 Request insertion procedure

Algorithm 1 gives the basic procedure used to insert one request r into a solution plan σ using one mode from a set of allowed modes $\mu_r \subseteq \mathcal{M}_r$. The algorithm first gets the best mode available in μ_r based on function *getBestMode* (line 2). The latter returns a best mode according to features like the mode reward or the current load of each satellite (more details later on these points). The mode chosen for r is stored in a global variable referred to as $m_r(\sigma)$.

Algorithm 1 then inserts one by one all activities covered by mode $m_r(\sigma)$. More precisely, for each observation-download pair (o,d) covered by the mode,

- function *greedyInsert* is used to insert o into the sequence of opportunities used by the satellite associated with o; this function follows an insertion heuristic H^{ins} (line 4) that selects a position for o in sequence Π_s so as to minimize the sum of maneuver times for s;
- if download opportunity d is already used in the plan, the procedure simply updates the current duration of d in σ (line 5); otherwise, download opportunity d is added to the sequence of opportunities used by satellite s, based on function *greedyInsert* again (line 8).

Algorithm 1 Insert a request

```
1: function INSERTREQUEST(σ, r, μ_r, H^ins)
2:     m_r(σ) ← getBestMode(r, μ_r, σ)
3:     for all (o, d) ∈ m_r(σ) do
4:         σ ← greedyInsert(σ, o, H^ins)
5:         if d ∈ σ then dur_d(σ) ← dur_d(σ) + Θ_o
6:         else
7:             dur_d(σ) ← Θ_o
8:             σ ← greedyInsert(σ, d, H^ins)
9:     return σ
```

4.2 Request removal procedure

At some steps, the search strategy needs to remove requests from the current plan. This is done based on Algorithm 2. The latter ensures that when a request r is removed, all observation and download activities associated with the current mode $m_r(\sigma)$ of r are removed from the plan. For an observation-download pair (o,d) in $m_r(\sigma)$, removing the download activity means first decreasing the duration associated with d, and then removing d if its duration becomes null (case in which no more download activity is using d). Function *removeActivity* simply removes an activity from the sequence of activities of a satellite, without changing the ordering among the other activities.

Algorithm 2 Remove a request

```
1: function REMOVEREQUEST(σ, r)
2:     for all (o, d) ∈ m_r(σ) do
3:         σ ← removeActivity(σ, o)
4:         dur_d(σ) ← dur_d(σ) − Θ_o
5:         if dur_d(σ) = 0 then s ← removeActivity(σ, d)
6:     return σ
```

4.3 Plan repair procedure

As mentioned before, after inserting a given request r, the current solution plan σ might become infeasible. In this case, we iteratively use two strategies to repair the solution following the insertion of r (Algorithm 3).

- The first one performs a local search on the sequences of opportunities used by the satellites whose plan is infeasible (line 4). This strategy does not change the content of the solution as no activity is removed or inserted.
- The second strategy removes one request from the plan following a repair heuristic denoted H^{rep} (lines 6-8).

The algorithm returns the new solution plan σ and the set of requests removed from the plan to repair the inconsistency. With this procedure, requests inserted into the plan at a given step can be removed for the sake of the requests that are not satisfied yet.

Algorithm 3 Repair

```
1: function REPAIR(σ, r, H^rep)
2:     E ← ∅
3:     while ¬feasible(σ) do
4:         σ ← localSearch(σ, H^rep)
5:         if ¬feasible(σ) then
6:             r' ← selectRemoval(σ, r, H^rep)
7:             σ ← removeRequest(σ, r')
8:             E ← E ∪ {r'}
9:     return (σ, E)
```

4.4 Iterative insertion and repair

Algorithm 4 gives an iterative algorithm built from the previous basic components. The algorithm takes as an input a current plan σ (possibly an empty plan), an insertion heuristic H^{ins}, and a repair heuristic H^{rep}.

Initially, there is a set of requests $\mathcal{R}(\sigma)$ that are already integrated into the plan, and all requests that are not in $\mathcal{R}(\sigma)$ are put in the set of requests $\mathcal{C}$ that are candidate for insertion. For every candidate request r in $\mathcal{C}$, all modes are available initially (line 3).

Then, at each step, the algorithm selects one candidate request r in $\mathcal{C}$ and inserts this request into the plan, following insertion heuristic H^{ins} (line 5-6). Request r is removed from the set of candidate requests, and if the solution obtained is not feasible, it is repaired based on the *Repair* function presented before. The latter removes a set of requests $\mathcal{E}$ from the current plan to restore consistency (line 9). The current mode $m_e(\sigma)$ of each request $e \in \mathcal{E}$ is then removed from the set of modes available for e, and if there is a mode left for e, the latter becomes a candidate for insertion again. The algorithm can be stopped if a maximum CPU time is reached, and it terminates since at least one more mode is accepted or removed after each repair (and there is a finite set of modes).

4.5 LNS algorithm

Finally, Algorithm 5 corresponds to the high-level algorithm that we use for EOS constellation planning with modes. This

Algorithm 4 IterativeInsertRepair with H^{ins} = insertion heuristic and H^{rep} = repair heuristic

```
1: function ITERATIVEINSERTREPAIR(σ, H^ins, H^rep)
2:     C ← R\R(σ)
3:     for all r ∈ C do μ_r ← M_r
4:     while C ≠ ∅ do
5:         r ← selectInsertion(σ, C, H^ins)
6:         s ← insertRequest(σ, r, H^ins)
7:         C ← C \ {r}
8:         if ¬feasible(σ) then
9:             (σ, E) ← Repair(σ, r, H^rep)
10:            for all e ∈ E do
11:                μ_r ← μ_r \ {m_e(σ)}
12:                if μ_r ≠ ∅ then C ← C ∪ {e}
13:    return σ
```

algorithm is a kind of Large Neighborhood Search (LNS), that alternates between constructive and destructive phases.

More precisely, the algorithm starts from an empty plan and quickly builds a first solution by using the *IterativeInsertionRepair* procedure seen before together with a simple insertion heuristics H_0^{ins} that systematically selects a request whose best mode has the highest reward, and a simple repair heuristic H_0^{rep} that systematically removes the last request inserted to repair the plan (line 2).

After that, while there is some CPU time left, the algorithm tries to apply the iterative insertion and repair procedure again, based on insertion and repair heuristics H^{ins} and H^{rep} that are typically less greedy than H_0^{ins} and H_0^{rep} (line 6). In our experiments, H^{ins} is the *lowest satellite load heuristic* detailed in Section 4.6 and H^{rep} is the *highest conflict-index* heuristic detailed in Section 4.7.

If the new solution σ obtained is strictly better that the best solution found since the last destructive phase, the number of iterations without improvement (it) is reset to 0 (line 8) and the best solution found is updated if needed (line 9). Otherwise, the number of iterations without improvement is updated and if it is greater than parameter *maxStableIt* given as an input, function *Perturb* is called to randomly remove some requests from current solution σ (line 13). This allows to diversify search. Finally, the best solution found along all iterations is returned.

4.6 Insertion heuristics

Insertion heuristic H^{ins} mentioned before is used to select a candidate request at each step. We consider two versions of this heuristic.

- *Highest reward heuristic*: this heuristic chooses a request r whose best allowed mode m offers the highest reward ω_m. This criteria does not depend on the current state of the solution.
- *Lowest satellite load heuristic*: this heuristic takes into account the load of the satellites during the time windows associated with the candidate modes of each candidate request. The main idea is that a low satellite load increases the chance to get a feasible solution after insertion and decreases the chance to get conflicts with modes inserted in

Algorithm 5 Large Neighborhood Search

```
1: function LNS(H_0^ins, H_0^rep, H^ins, H^rep, maxStableIt, H^perturb, MaxTime)
2:     σ ← IterativeInsertRepair(∅, H_0^ins, H_0^rep)
3:     σ* ← σ
4:     (ω^Stable, it) ← (ω(σ), 0)
5:     while cpuTime() < MaxTime do
6:         σ ← IterativeInsertRepair(σ, H^ins, H^rep)
7:         if ω(σ) > ω^Stable then
8:             (ω^Stable, it) ← (ω(σ), 0)
9:             if ω(σ) > ω(σ*) then σ* ← σ
10:        else
11:            it ← it + 1
12:            if it > MaxStableIt then
13:                σ ← Perturb(σ, H^perturb)
14:                (ω^Stable, it) ← (ω(σ), 0)
15:    return σ*
```

the future. Basically, the average load of satellite s at time t in solution σ is defined by:

$$Load_s(t,\sigma) = \sum_{a \in \mathcal{A}_s(\sigma)} \frac{dur_a(\sigma)}{End_a - Start_a} \mathbb{1}_{[Start_a, End_a[}(t)$$

In other words, over its allowed time window, each activity a consumes a resource ratio that depends on the duration of a in σ. Without going into further implementation details, the satellite load heuristic selects at each step a request r^* and a mode m^* so as to minimize the height of the maximum peek that is obtained in the satellite load profiles after inserting the mode. This criteria depends on the content of the current solution plan σ.

4.7 Repair heuristics

Repair heuristic H^{rep} is used to choose a mode to remove when the current solution becomes infeasible. We consider three versions of this heuristic.

- *Highest tardiness heuristic*: when the solution becomes infeasible due to temporal constraints, it is possible to define, for each activity a of the plan, a tardiness measure

$$\tau_a(\sigma) = \max(0, earliest_a(\sigma) + dur_a(\sigma) - End_a)$$

This tardiness quantifies by how much activity a is late with regards to the end time of its allowed time window. Then, the tardiness of a mode m used in the plan can be defined as:

$$\tau_m(\sigma) = \sum\nolimits_{o \in \mathcal{O}_m} \tau_o(\sigma) + \sum\nolimits_{d \in \mathcal{O}_m} \tau_d(\sigma)$$

The highest tardiness heuristic removes a request whose mode $m_r(\sigma)$ in the current solution has the highest tardiness.

- *Lowest reward heuristic*: among the requests whose mode is involved in the infeasibility explanation ($\tau_m(\sigma) > 0$), this heuristic removes a request r whose mode $m_r(\sigma)$ has the lowest reward.

- *Highest conflict-index heuristic*: instead of measuring the precise tardiness associated with a mode, it is possible to count the number of times each mode occurs in an infeasible part of the solution. To do this, for each activity $a = \Pi_{s,i}$ of the plan such that $\tau_a(\sigma) > 0$, the conflict-set explaining the tardiness of a can be defined as the shortest subsequence $exp_a(\sigma) = [\Pi_{s,i-k}, \ldots, \Pi_{s,i}]$ such that $earliest_{\Pi_{s,i-k}}(\sigma) = Start_{\Pi_{s,i-k}}$. From this, the conflict-index of an activity a in solution σ is defined as the number of times it belongs to a conflict-set:

$$CI_a(\sigma) = card\{b \in \mathcal{A}(\sigma) \mid a \in exp_b(\sigma)\}$$

where $\mathcal{A}(\sigma)$ stands for the set of activity opportunities included in σ. Then, the conflict-index of each mode used in the plan is defined by:

$$CI_m(\sigma) = \sum\nolimits_{o \in \mathcal{O}_m} CI_o(\sigma) + \sum\nolimits_{d \in \mathcal{O}_m} CI_d(\sigma)$$

The highest conflict-index heuristic removes a request whose mode has the highest conflict-index.

5 Experiments

5.1 Instances description

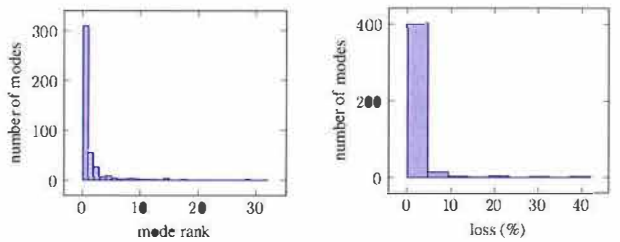
The instances generated correspond to constellations composed of 16 and 32 satellites. The scheduling horizon covers 48 hours, the last 24 hours being dedicated only to download opportunities. Areas of interest for the requests are all located in Europe and partitioned into 4 types: (1) *monotonic requests* requiring a single observation; (2) *stereoscopic requests* requiring two observations performed using two different pointing angle ranges; (3) *systematic requests* requiring the scheduling of all possible observations of the target; (4) *periodic requests* requiring the scheduling of one observation within each period (4 periods are considered in the instances). We make the number of requests vary from 200 to 500. Each instance is named *R1-R2-R3-R4* where *Ri* represents the number of requests of type i as described previously. The number of modes and the size of the modes directly depend on the type of the associated request. In order to limit the number of modes for each request, we only consider the 10 best modes for each request. Moreover, we only consider one download opportunity for each observation. For each request, we randomly generate two bounds a and b in $[0, 1]$, and a random reward in $[a, b]$ is assigned to each mode of this request.

5.2 Results analysis

We have implemented the LNS approach in Python 3.6.9 and encoded the MILP approach in CPLEX 20.1. Both approaches have been run on 20-core Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz, 62GB RAM. For the LNS approach, we respectively use the *Lowest satellite load heuristic* and the *Highest conflict-index heuristic* as insertion and repair heuristics (H^{ins} and H^{rep}). $H^{perturb}$ consists in removing 5% of the modes. We arbitrarily choose $MaxStableIt = 1$ and the maximum time for the LNS algorithm is set to $MaxTime = 5$ minutes. Maximum time for the MILP approach is set to 10 minutes.

Instance	IIR_0	LNS	MILP
16 satellites			
40-0-0-0	30.97 (40)	30.97 (40)	**30.97 (40)**
0-40-0-0	30.41 (40)	30.41 (40)	**30.43 (40)**
0-0-40-0	8.24 (15)	9.62 (16)	0 (0)
0-0-0-40	25.15 (32)	29.71 (40)	**30.5 (40)**
10-10-10-10	22.24 (30)	25.32 (35)	6.71 (9)
100-0-0-0	75.76 (100)	75.76 (100)	**75.76 (100)**
0-100-0-0	74.11 (100)	74.11 (100)	**74.26 (100)**
0-0-0-100	49.96 (65)	69.64 (100)	0 (0)
50-50-0-0	74.79 (100)	74.79 (100)	**75.20 (100)**
25-25-25-25	55.79 (73)	60.87 (84)	3.20 (5)
200-0-0-0	150.74 (200)	150.74 (200)	0 (0)
0-200-0-0	146.07 (200)	146.07 (200)	142.21 (192)
0-0-200-0	8.68 (11)	12.9 (20)	0 (0)
0-0-0-200	72.37 (92)	92.92 (131)	0 (0)
100-100-0-0	148.49 (200)	148.49 (200)	142.30 (191)
50-50-50-50	99.69 (133)	111.8 (157)	0 (0)
250-250-0-0	336.35 (452)	349.29 (481)	0 (0)
125-125-125-125	209.41 (280)	225.75 (308)	–
32 satellites			
250-250-0-0	371.45 (500)	371.45 (500)	0 (0)
125-125-125-125	239.16 (320)	254.95 (348)	0 (0)

Table 1: Global reward obtained for each approach along with the number of satisfied requests in parentheses.



(a) Modes ranks (b) Loss w.r.t. the best mode

Figure 2: Selected modes for a 500-request instance

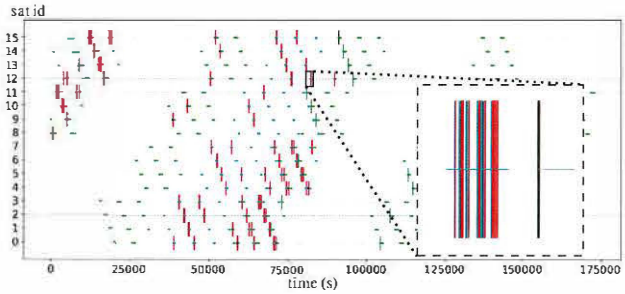


Figure 3: Solution for a 500 requests instance and a 16 satellites constellation.

Representative results are presented in Table 1. The reward value of the first solution obtained by *IterativeInsertRepair* (line 2 in Algorithm 5) is given in the column named IIR_0. A bold font is used for optimal solutions (MILP approach).

Results show that the MILP approach is effective on small instances but does not scale up. Indeed, for most instances involving more than 200 requests, the approach returns an empty plan or does even finish pre-processing. In comparison, LNS produces results for all generated instances, whatever the types of the requests considered. The global reward varies with respect to the types of requests in the instance since satisfying requests having small modes (monotonic and stereoscopic requests) is easier than satisfying requests having large modes (systematic or periodic requests). For instance, the LNS approach satisfies all the requests for instance *250-250-0-0* but not for instance *125-125-125-125*. Moreover, the global reward increases with the number of observation opportunities (more satellites) or with the number of non-systematic requests. Solutions returned by the LNS approach are quite close from the optimal solutions for small instances and have a better global reward for instances with 200 requests or more. Note that the global reward obtained by the first iterative insert repair (IIR_0) is improved during LNS. As illustrated in Figure 2a, most requests that are satisfied are fulfilled through one of their best modes but some requests are fulfilled through their 30th best mode. Requests that are not fulfilled with their best mode have their associated reward decreased. As illustrated in Figure 2b, the reward loss is smaller than 40% *w.r.t.* the best mode.

Figure 3 illustrates the scheduling of 500 requests for a constellation of 16 satellites. The observation, download and maneuver activities are represented by rectangles that are respectively blue, green and red. The horizontal blue and green lines correspond respectively to the observations and downloads opportunity windows. Note that the sparse structure of the plan comes the fact that requests are focused on Europe.

6 Conclusion and future works

This paper presented a mode-based approach for managing complex requests for a constellation of satellites, together with two algorithmic approaches (MILP and LNS). Experiments performed on generated instances show that LNS scales well, which is promising with respect to large catalogs of requests expected for future satellite constellations.

A first perspective of this work is to improve the LNS approach. To do so, it would be possible to limit the number of modes and allow on-the-fly creation of new modes for a given request when current modes for this request cannot be inserted. Parallelization techniques could also be used by splitting the scheduling horizon in order to obtain smaller independent problems. A second perspective consists in making the problem more realistic by considering time-dependent maneuver duration models or by using reward models that take into account weather forecasts, delivery time for the observation, requests priority, *etc.*

Acknowledgments

This work has been performed with the support of the French government in the context of the "Programme d'Investissements d'Avenir", namely by the BPI PSPC project LiChIE led by AIRBUS Defence and Space and involving several partners, including ONERA.

References

Agrawal, J.; Chi, W.; Chien, S.; Rabideau, G.; Kuhn, S.; Gaines, D.; Vaquero, T.; and Bhaskaran, S. 2021. Enabling Limited Resource-Bounded Disjunction in Scheduling. *Journal of Aerospace Information Systems* .

Bianchessi, N.; Cordeau, J.-F.; Desrosiers, J.; Laporte, G.; and Raymond, V. 2007. A heuristic for the multi-satellite, multi-orbit and multi-user management of Earth observation satellites. *European Journal of Operational Research* .

Coelho, J.; and Vanhoucke, M. 2015. The Multi-Mode Resource-Constrained Project Scheduling Problem. In Schwindt, C.; and Zimmermann, J., eds., *Handbook on Project Management and Scheduling Vol.1*, International Handbooks on Information Systems, 491–511. Springer.

Dishan, Q.; Chuan, H.; Jin, L.; and Manhao, M. 2013. A Dynamic Scheduling Method of Earth-Observing Satellites by Employing Rolling Horizon Strategy. *The Scientific World Journal* .

Du, Y.; Wang, T.; Xin, B.; Wang, L.; Chen, Y.; and Xing, L. 2020. A Data-Driven Parallel Scheduling Approach for Multiple Agile Earth Observation Satellites. *IEEE Transactions on Evolutionary Computation* .

Eddy, D.; and Kochenderfer, M. J. 2020. A Maximum Independent Set Method for Scheduling Earth Observing Satellite Constellations. *arXiv:2008. 08446[cs, eess]* .

Han, C.; Wang, X.; Song, G.; and Leus, R. 2018-12-01. Scheduling multiple agile Earth observation satellites with multiple observations. *arXiv:1812.00203 [astro-ph]* .

He, L.; Liu, X.; Laporte, G.; Chen, Y.; and Chen, Y. 2018. An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. *Computers & Operations Research* .

He, Y.; Chen, Y.; Lu, J.; Chen, C.; and Wu, G. 2019. Scheduling multiple agile earth observation satellites with an edge computing framework and a constructive heuristic algorithm. *Journal of Systems Architecture* .

Hu, X. 2019. A branch and price algorithm for EOS constellation imaging and downloading integrated scheduling problem .

Kim, J.; Ahn, J.; Choi, H.-L.; and Cho, D.-H. 2019. Task Scheduling of Multiple Agile Satellites with Transition Time and Stereo Imaging Constraints .

Li, Z.; and Li, X. 2019. A multi-objective binary-encoding differential evolution algorithm for proactive scheduling of agile earth observation satellites. *Advances in Space Research* 63(10): 3258–3269.

Sun, K. 2012. Multi-objective mission planning problem of a constellation agile satellites.

Talbot, F. 1982. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science* 28(10): 1197–1210.

Vongsantivanich, W.; Holvoet, N.; Chaimatanan, S.; and Delahaye, D. 2018. Mission planning for non-homogeneous Earth observation satellites constellation for disaster response .

Wang, J.; Zhu, X.; Yang, L. T.; Zhu, J.; and Ma, M. 2015. Towards dynamic real-time scheduling for multiple earth observation satellites. *Journal of Computer and System Sciences* .

Xiao, Y.; Zhang, S.; Yang, P.; You, M.; and Huang, J. 2019. A two-stage flow-shop scheme for the multi-satellite observation and data-downlink scheduling problem considering weather uncertainties. *Reliability Engineering & System Safety* .