Mesh Dispatching for Area Coverage using Several Earth Observation Systems

Cédric Pralet,[1] Gauthier Picard,[1] Cyrille de Lussy,[2] Jonathan Guerra[2]

[1] DTIS, ONERA, Université de Toulouse, France
[2] Airbus Defence and Space, Toulouse, France
cedric.pralet@onera.fr, gauthier.picard@onera.fr, cyrille.de-lussy@airbus.com, jonathan.guerra@airbus.com

Abstract

In this paper, we consider area coverage requests that can be fulfilled using a set of Earth observation systems, where each system disposes of its own satellites and its own mission center. The approach proposed involves a federation layer that dispatches meshes of various sizes to the different systems in a seamless way for the end-users. The dispatch strategy is optimized based on heuristic search and large neighborhood search. Weather and system workload uncertainties being hard to estimate, re-dispatch operations are also used to re-optimize the allocation of meshes to systems depending on the current progress. Experiments are performed on scenarios where the goal is to collect images over countries covering up to hundreds of thousands square kilometers, such as France.

1 Introduction

Nowadays, *Earth Observation* (EO) satellites are exploited for various applications, *e.g.* disaster management, land monitoring, or conflict monitoring. For such applications, the end-users demand scalable, responsive, and cost-effective solutions to observe possibly large areas that may cover several tens or hundreds of thousands square kilometers, whereas each individual observation performed by a satellite may cover only a few tens of square kilometers. As multiple EO systems have been developed over the years to fulfill specific observation needs, each end-user can post its observation requests to different candidate *missions*. Here, a mission refers to a satellite or a set of satellites managed by a single operator (or mission center) that uses its own planning engine to optimize the satellite acquisition plans given the set of observation requests received so far.

But nowadays, to collect images over large areas as quickly as possible, it is quite natural to try and benefit from the capabilities of *multiple* existing EO missions in parallel (Farges et al. 2024). This leads us to develop an extra *federation layer* whose role is to continuously receive observation requests, dispatch observation tasks to the missions, get the observation data back, and deliver the corresponding images to the end-users (see Fig. 1). Doing so, each end-user benefits from a seamless access to numerous satellite resources through a single entry point, without having to care about the details. On the other side, the federation layer optimizes the decomposition of the workload among several missions to speed up the achievement of coverage requests, while possibly merging redundant demands posted by multiple end-users. Globally, the federation layer must tackle several challenges:

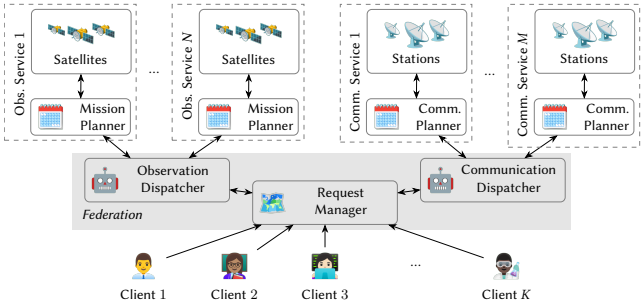

Figure 1: Federated architecture improving performance by dispatching requests from multiple end-users to multiple systems in a seamless way (Farges et al. 2024)

- It must solve a *highly combinatorial problem* (hundreds of requests, tens of satellites or more, many ways to partition a large area among different missions, etc.).
- It must handle *multiple objectives* (minimization of the completion time of the coverage requests, minimization of the total workload of the systems, etc).
- It does not fully control the satellite plans since a specific planning engine is used by each mission. It may even ignore the orderbook of some external missions.
- It must deal with various sources of uncertainty, e.g. the presence of clouds leading to failed images. This requires to regularly revise the dispatch strategy to take into account the current progress of the coverage tasks.

The rest of the paper is organized as follows. After having discussed related works, we formalize the problem solved by the federation layer. This problem consists in dispatching *meshes* among the missions available. We then detail optimization techniques for quickly dispatching and re-dispatching meshes over different missions. Last, we give experimental results and provide some perspectives. This work is performed in the context of the DOMINO-E Eu-

ropean project whose goal is to empower EO operators to intelligently coordinate their imagery needs across different mission-specific systems (https://domino-e.eu/).

2 Related works

Covering large areas using several Earth observation satellites is a problem that already received attention in the literature. On this topic, some existing works discretize the large areas to get a finite set of relevant (discrete) points within the area. This is the case for the CLASP system used for several NASA missions (Maillard, Chien, and Wells 2021). In this system, a set of observation opportunities is computed for each discrete point given the satellite passes, different candidate slewing angles, and the on/off times of the instruments. The problem to solve then becomes an observation opportunity selection problem where the goal is to maximize the number of points for which a given coverage quality is reached. For this, the authors propose a greedy scheduler that exploits a strict priority order between the observation opportunities coupled with the exploration of different priority orders. On the same line, Liu and Hodgson (2013) consider a set of points for covering a large area for disaster emergency response, but they exploit techniques for extracting a restricted number of relevant points within the large area. Moreover, their observation opportunity selection strategy aims at optimizing a weighted sum of four objective functions respectively related to the spatial resolution of the opportunities selected, their spectral resolution, the data collection day, and the off-nadir angles. These techniques are applied to scenarios involving 8 satellites.

Instead of considering observation opportunities over discrete points, some existing works study the multi-satellite coverage of a polygon using 2D-strips. Ntagiou et al. (2018) consider a finite set of candidate strips, defined by discrete slewing angles, together with memory constraints and an objective function requiring to maximize the coverage percentage at the end of the planning horizon. They solve this problem using ant colony optimization, for a disaster monitoring constellation involving 3 satellites. Similarly, Niu, Tang, and Wu (2018) compute the set of satellite passes over the area as well as a finite set of candidate strips based on a finite set of slewing angles. Their goal is then to select strips among the candidate ones in order to maximize the coverage rate, minimize the completion time, maximize the average spatial resolution, and minimize the average slew. This multi-objective optimization problem is solved by a genetic algorithm, for a scenario involving 16 satellites from different missions. Zhibo et al. (2021) also use a finite set of candidate strips and a genetic algorithm, but they take into account the duration of the maneuvers between strips as well as memory and energy limitations, for a scenario involving 15 satellites. To explore a larger solution space, Chen et al. (2020) consider 0/1 strip selection variables but continuous candidate slewing angles for each strip. They also exploit a genetic algorithm to both maximize the area covered and minimize the number of selected strips, for scenarios involving 4 satellites. Lenzen et al. (2021) propose an intermediate approach where the candidate slewing angles are iteratively discretized, for a scenario involving a single satellite. Last, Berger, Lo, and Barkaoui (2020) consider candidate observation strips that may have heterogeneous orientations.

In another direction, for missions involving agile satellites that can point in any direction (left, right, forward, backward), some authors developed area scanning strategies either for a single satellite (Shao et al. 2018) or for multiple satellites (Ji and Huang 2019). Finally, there exist works on planning algorithms managing several coverage requests, with an objective function favoring the completion of the ongoing requests based on a non-linear partial reward function (Lemaître et al. 2002).

With regard to these existing works, our ambition is to avoid considering a finite set of arbitrary discrete points or long strips of arbitrary lengths. To do this, we reason about the meshes (smaller than strips) associated with each mission. This allows us to model situations where during a single pass over an area, an agile satellite may observe several adjacent meshes not aligned with the ground track. Moreover, we aim at dealing with both the coverage requests formulated at the level of the federation layer and the requests directly included in the orderbook of each mission, and with the fact that the specific planning engine used for each mission is not directly controllable. Last, we aim at revising the dispatch strategy given the current progress.

3 Model of the Mesh Dispatch Problem

We first describe the mathematical model developed to dispatch coverage requests to different missions possibly using different mesh sizes. Downlink issues are not handled here.

3.1 Input data

Observation systems and requests The model first considers the following input data:

- R: set of observation requests received by the federation layer; each request covers an area defined as a union of polygons;
- S: set of observation systems (or missions) available;
- $\forall s \in S,\ MeshSize_s \in \mathbb{R}$: size of the meshes of system s (depending on the swath width of the satellite sensors);
- $\forall s \in S,\ R_s$: set of requests in the orderbook of demands received directly by system s (independently of the federation layer); for external systems, these requests may be hidden to the federation layer (in this case, $R_s = \emptyset$);
- $\forall r \in R \cup R_s,\ Pr_r \in \mathbb{N}$: priority of request r; as a convention, priority 0 corresponds to the highest priority.

Decomposition of the areas of interest For each system, the ground areas to observe are decomposed into a set of meshes. In this work, we assume that all the systems work on meshes aligned with the North-South and East-West directions. More precisely, they use a so-called *world layer split* that decomposes the Earth surface into a set of layers placed at successive latitudes. Each layer contains adjacent meshes covering the set of possible longitudes, and the number of meshes in a layer decreases from the Equator to the poles. From the world layer split associated with each system, we compute the following input data related to the spatial description of the areas of interest:

- $\forall s \in S$, M_s: for system s, set of meshes that have a non-empty intersection with one of the requests in $R \cup R_s$; the dimension of each mesh in M_s is compatible with the swath width of the sensor of each satellite in s;
- C: a set of so-called *cells*, corresponding to an atomic decomposition of the areas of interest for the requests in $R \cup (\cup_{s \in S} R_s)$; basically, each cell is a maximum polygon such that for each observation system, all points in the cell are contained in the same system mesh (see Fig. 2);
- $\forall c \in C$, $CellSize_c \in \mathbb{R}$: size of cell c;
- $\forall c \in C$, $UsefulCellSize_c \in \mathbb{R}$: size of the intersection between cell c and the areas of interest; this size is equal to $CellSize_c$ if and only if cell c is fully included in the areas of interest;
- $\forall s \in S, \forall i \in M_s$, $C_{si} \subseteq C$: set of cells covered by mesh i of system s; each mesh may cover more than one cell; in Fig. 2, some meshes in orange cover up to 9 cells.

Concerning these input data, we use a specific algorithm (not detailed here) to quickly compute the relevant cells given the meshing strategy exploited by each system. The cell-decomposition approach allows us to reason about a minimum set of discrete entities while bringing strong guarantees that the coverage of all the cells implies the coverage of all the areas of interest (no hole in the final coverage). The previous input data could also be derived from decompositions aligned with the ground track of the satellites, but this point is left for future works.

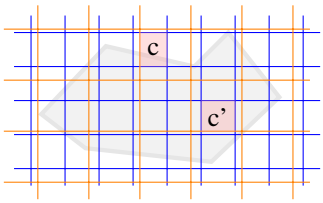

Figure 2: Examples of cells (in red) for an area of interest (in gray), given the meshes of two systems (grids in blue and orange); for cell c on the top, we have $UsefulCellSize_c < CellSize_c$, while for cell c' on the right we have $UsefulCellSize_{c'} = CellSize_{c'}$

Capacities of the satellites Given the complexity of the dispatch problem in terms of number of meshes and satellites, we manipulate a coarse-grain model of the observation capacities of each satellite. In this model, the passes of the satellites over the areas of interest are decomposed into successive *satellite slots*, each covering X mesh layers in the world layer split. See Fig. 3 for a scenario involving 6 satellite passes (arrows in blue), with the corresponding successive slots for one of them (green rectangles covering $X = 5$ mesh layers). Within each slot, the satellite can observe up to N meshes (e.g., $N = 5$) among a set of visible meshes defined from the coordinates of the mesh centers and the position of the satellite over its orbit during the slot (in Fig. 3, possibility to observe all hatched meshes during the slot in yellow). Additionally, to get a compact model, we take into account the fact that each slot of a given satellite can be repeatedly used after each orbital cycle (e.g., every 26 days). If there is no orbital cycle for a given satellite, it is possible to try and compute an approximate cycle time based on a given precision on the longitude of the ascending node of each orbit. Formally, for each system $s \in S$, we consider the following input data:

- K_s: set of satellite slots associated with system s;
- $\forall k \in K_s$, $Capacity_{sk} \in \mathbb{N}$: maximum number of meshes observable during slot k;
- $\forall k \in K_s$, $M_{sk} \subseteq M_s$: set of meshes visible during slot k; these meshes are obtained from geometric conditions on the satellite-zenith and Sun-zenith angles;
- $\forall k \in K_s$, $SlotTime_{sk} \in \mathbb{R}^+$: date of slot k;
- $\forall k \in K_s$, $Cycle_{sk} \in \mathbb{R}^+$: duration after which slot k becomes available again (cycle time); hence, slot k is available at times $t_1 = SlotTime_{sk}$, $t_2 = SlotTime_{sk} + Cycle_{sk}$, $t_3 = SlotTime_{sk} + 2 \cdot Cycle_{sk}$, and so on.

Note that if the next meteorological forecast for slot k of system s is not satisfactory owing to a cloud cover threshold, it is possible to update input data $SlotTime_{sk}$ by $SlotTime_{sk} \leftarrow SlotTime_{sk} + CycleTime_{sk}$. Also, the input data introduced contain several parameters whose precise values must be set depending on the actual features of the satellites considered.

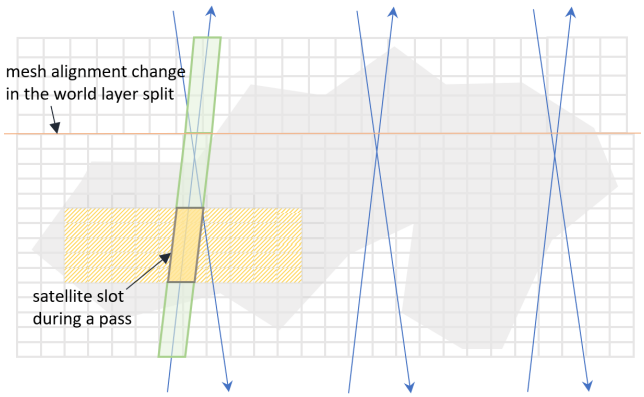

Figure 3: Satellite slots defined from satellite passes

Mesh neighborhood In practice, the number of meshes that can be observed by a satellite during a pass actually depends on the spatial dispersion of the meshes allocated to the satellite slots, because a higher spatial dispersion increases the maneuver times between successive observations. Also, the satellites considered are more agile around the pitch axis than around the roll axis, and it is therefore preferable to assign to each satellite strips of meshes aligned with the direction of the ground track of the satellite. For these reasons, we define a mesh neighborhood structure that is used by the algorithms to try and allocate strips of meshes to the systems, when possible, rather than meshes spread over the area of interest. To do this, we introduce the following input data for each system $s \in S$:

- $Ngh_s \subseteq M_s \times M_s$: set of mesh pairs (i, i') such that i and i' are considered as *neighbors* for system s, that is as meshes that are close to each other;
- $\forall (i, i') \in Ngh_s$, $SpatialRwd_{sii'} \in \mathbb{R}^+$: *mesh grouping reward* obtained if meshes i and i' are both allocated to system s.

The neighborhood obtained for each mesh is shown in Fig. 4, that illustrates both the case where the current mesh layer and the next one have the same alignment in the world layer split and the case where a mesh layer and the next one have different alignments. For instance, we collect a reward of 0.4 if meshes i_0 and i_5 are both selected, and a reward of $0.16 + 0.4 = 0.56$ if "strip" $\{i_2, i_0, i_5\}$ is selected. From a global point of view, this approach allows us to prefer solutions where the meshes allocated to a given system are grouped, as illustrated in Fig. 5. Other settings could be used for the values of the mesh grouping rewards, for instance to better deal with the case of inclined orbits.

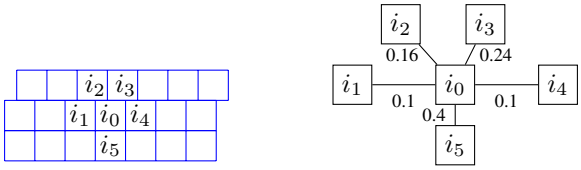

Figure 4: Neighborhood of mesh i_0 and rewards obtained if neighboring meshes are selected together with i_0

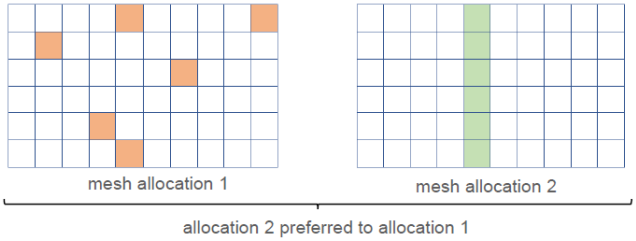

Figure 5: Two allocations of meshes to a given system

3.2 Decision variables

To model the mesh dispatch problem, we introduce the following decision variables:

- $\forall s \in S$, $\forall i \in M_s$, $x_{si} \in \{0, 1\}$: Boolean variable taking value 1 if and only if an observation of mesh i is requested to system s;
- $\forall s \in S$, $\forall k \in K_s$, $\forall i \in M_{sk}$, $y_{ski} \in \{0, 1\}$: Boolean variable taking value 1 if and only if mesh i is allocated to satellite slot k of system s;
- $\forall s \in S$, $\forall k \in K_s$, $z_{sk} \in \mathbb{N}$: number of cycles required to observe all the meshes allocated to slot k of system s;
- $\forall s \in S$, $\forall k \in K_s$, $t_{sk} \in \mathbb{R}^+$: completion time associated with slot k of system s.

3.3 Constraints

The constraints of the problem are listed in Equations 1-4. Equation 1 expresses that all cells must be covered by a selected mesh, so that all the areas are covered. Equation 2 expresses that a mesh is allocated to system s if and only if it is allocated to a (unique) satellite slot associated with s. Equation 3 defines the minimum number of cycles required by a slot to observe all the meshes allocated to that slot. This number is derived from the slot load and the slot capacity. Equation 4 uses this number of cycles to compute an estimated completion time from each slot.

$$\forall c \in C, \sum_{s \in S, i \in M_s \mid c \in C_{si}} x_{si} \geq 1 \quad (1)$$

$$\forall s \in S, \forall i \in M_s, \; x_{si} = \sum_{k \in K_s} y_{ski} \quad (2)$$

$$\forall s \in S, \forall k \in K_s, \; z_{sk} = \left\lceil \frac{\sum_{i \in M_{sk}} y_{ski}}{Capacity_{sk}} \right\rceil \quad (3)$$

$$\forall s \in S, \forall k \in K_s, \quad (4)$$

$$t_{sk} = \begin{cases} 0 \text{ if } z_{sk} = 0 \\ SlotTime_{sk} + Cycle_{sk}\,(z_{sk} - 1) \text{ otherwise} \end{cases}$$

3.4 Objective functions

The first objective (Equation 5) consists in minimizing the maximum completion time obtained over the different slots. The second one (Equation 6) consists in maximizing the reward obtained by allocating neighboring meshes to the same system, so as to limit the maneuver times for the satellites. The third objective (Equation 7) is to minimize the size of the total area over which images are captured. This is equivalent to minimizing the size wasted due to the selection of overlapping meshes associated with different systems, or due to the selection of meshes that overlap the frontiers of the areas of interest. The fourth objective (Equation 8) consists in minimizing the sum of the completion times of the different slots, the idea being to complete each observation task as soon as possible instead of just minimizing the completion time of the last slot.

$$\text{minimize} \quad \max_{s \in S, k \in K_s} t_{sk} \quad (5)$$

$$\text{maximize} \quad \sum_{s \in S, (i,i') \in Ngh_s} SpatialRwd_{sii'} \cdot x_{si} \cdot x_{si'} \quad (6)$$

$$\text{minimize} \quad \sum_{s \in S, i \in M_k} MeshSize_s \cdot x_{si} \quad (7)$$

$$\text{minimize} \quad \sum_{s \in S, k \in K_s} t_{sk} \quad (8)$$

These objective functions lead to a preference for different solutions. For instance, to maximize the total mesh grouping reward, it is better to allocate all the areas to cover to a single system, while to minimize the completion time, it is better to share the workload between the systems. To jointly optimize the different objectives, we could use a weighted sum or compute non-dominated solutions, e.g. based on an epsilon-constrained method. In the following, we simply optimize the objective functions in the order

given by Equations 5-8. Once a set of meshes $M'_s \subseteq M_s$ is selected for each system $s \in S$, the federation layer sends requests to system s for demanding the observation of all the meshes in M'_s. System s then tries to achieve the correspond demands in the plans built by its own planning engine.

In the model obtained, the main decision actually consists in allocating meshes to systems. Indeed, it can be shown that once these decisions are made, the part of the problem consisting in allocating meshes to slots using a given number of cycles is actually close to a classical optimization problem known as a *maximum flow problem* in a graph. This aspect is not detailed due to space limitations. Last, the non-linear constraints and objective functions of the model could be linearized by introducing additional variables.

4 Optimization Techniques

We now present the algorithms developed for tackling the mesh dispatch problem, including a greedy heuristic algorithm and a Large Neighborhood Search (LNS). We also describe methods for re-dispatching the meshes depending on the current progress of the coverage tasks.

4.1 Greedy Heuristic Search

In order to build good-quality solutions, we first propose a greedy search process. As illustrated in Fig 6, this process starts from a solution where the set of meshes allocated to each system is empty and then iteratively fills the passes of satellites over the areas of interest. The satellite passes are filled in a chronological order, that is the next pass to fill is the pass that starts at the earliest possible time. Here, filling a pass of a satellite of system s means adding no more than $Capacity_{sk}$ mesh observations during each slot k of this pass. The meshes added to slot k are chosen among the set of meshes in M_{sk} that are not already covered. Moreover, if slot k becomes full following the mesh selections, we open a new candidate slot at the next cycle, following input data $Cycle_{sk}$. The process continues until all the areas of interest are covered. After that, some post-processing is performed to remove redundant mesh allocations that may have been introduced (last step in Fig. 6). In the following, we detail some algorithmic settings.

Mesh selection heuristics When filling a satellite pass, we need to select meshes for each satellite slot involved in the pass. To guide the selection with regard to the objective functions given in Equations 5-8, the algorithm maintains several features for each mesh i of each system $s \in S$:

- a mesh grouping score $grpScore_{si}$ that gives the increase obtained in the mesh grouping objective function if mesh i is selected; formally, grouping score $grpScore_{si}$ is set to 0 initially for all the meshes, and each time a new mesh i is selected, the scores of its neighbors i' (such that $(i, i') \in Ngh_s$) are updated by $grpScore_{si'} \leftarrow grpScore_{si'} + SpatialRwd_{sii'}$;
- a coverage score $covScore_{si}$ corresponding to the size of the subarea of mesh i that is still useful given the meshes already allocated to satellite passes of other systems; initially, this coverage score is equal to the sum of the useful areas of the cells covered by i, that is $covScore_{si} \leftarrow \sum_{c \in C_{si}} UsefulCellSize_c$; then, each time a mesh i of system s is selected, we consider each cell c covered by i (*i.e.*, each cell c in C_{si}), and for each mesh i' of a system $s' \neq s$ such that $c \in C_{s'i'}$, we update the coverage score by $covScore_{s'i'} \leftarrow covScore_{s'i'} - UsefulCellSize_c$;
- a so-called penalized mesh priority level $Pm_{si} \in \mathbb{R}$ that takes into account the highest priority of a request to which mesh i contributes for system s, penalized by the presence of the area covered by i in the orderbook of other systems. More precisely, we can compute a basic mesh priority level $Pm'_{si} = \min_{r \in R \cup R_s \mid i \text{ contributes to } r} Pr_r$. Then, the penalized mesh priority level is given by $Pm_{si} = Pm'_{si} + 0.5$ if the area covered by mesh i has a non-null intersection with the orderbook of another system $s' \neq s$, and $Pm_{si} = Pm'_{si}$ otherwise.

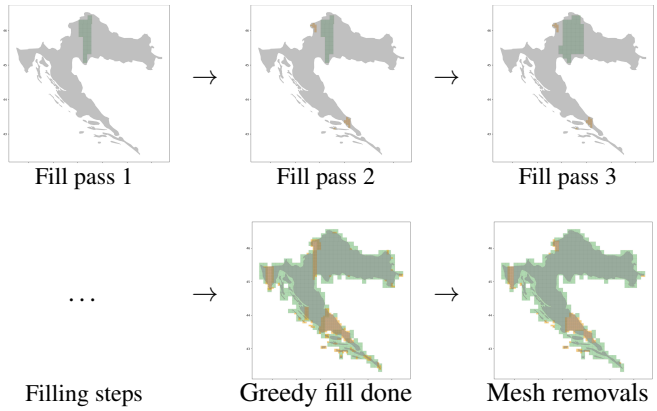

Figure 6: Iterations of the greedy search procedure selecting meshes step-by-step to cover the area of interest, for a scenario involving two systems (system 1 using the green meshes and system 2 using the yellow meshes)

Then, at each mesh selection step, we use the following heuristic:

- select a mesh that has a positive coverage score ($covScore_{si} > 0$) to avoid useless observations;
- in case of ties, select a mesh whose penalized priority level is the smallest; the idea is to prefer the selection of meshes associated with high-priority requests and that are not present in the orderbook of other systems;
- in case of ties, select a mesh that has the highest coverage score; this penalizes the selection of meshes overlapping meshes selected for other systems;
- in case of ties, select a mesh that gives the highest mesh grouping reward, to decrease the maneuver times;
- in case of ties, select a random mesh.

Removal of useless meshes During the greedy fill process, it may happen that a mesh i of system s is selected at some step and then a mesh i' containing i and associated with another system s' is selected at another step. In this case, the selection of mesh i can be canceled. To identify the useless meshes, we maintain for each cell $c \in C$ the set of

meshes $covMeshes_c$ covering c that are selected in the current solution. This set is initialized by $covMeshes_c \leftarrow \emptyset$ and updated each time a new mesh i covering c is selected for a system s ($covMeshes_c \leftarrow covMeshes_c \cup \{i\}$ in this case).

Then, to remove all useless meshes, we traverse the satellite passes of the current solution in the reverse chronological order. For each mesh i of system s allocated to the pass considered and such that $covScore_{si} = 0$, we remove i from the current solution and apply operation $covMeshes_c \leftarrow covMeshes_c \setminus \{i\}$ for each cell $c \in C_{si}$. Moreover, if set $covMeshes_c$ becomes a singleton containing a unique mesh i' associated with a system s', the coverage score of i' is increased by $covScore_{s'i'} \leftarrow covScore_{s'i'} + UsefulCellSize_c$. This leads to $covScore_{s'i'} > 0$, hence mesh i' will never be deactivated during the mesh removal process, or in other words it becomes mandatory.

Solution compression Due to mesh deactivations, there may exist residual capacities in some satellite slots of the current solution. To exploit these residual capacities, we traverse the satellite slots of each system s by increasing completion times. For each mesh i allocated to a slot k, if possible, we move i to the first slot k' such that (1) k' has a residual capacity, (2) $i \in M_{sk'}$, and (3) adding i to k' leads to a lower completion time for i. In other words, we perform a backward move exploiting a residual capacity in k'.

4.2 Large Neighborhood Search (LNS)

The greedy fill process described before can be iterated to generate several candidate solutions, since the mesh selection heuristic involves random choices in case of ties. A multi-start strategy can be useful to diversify the set of solutions explored, but it is also relevant to intensify search around the solution found after each greedy filling phase. This is why we also propose to use LNS (Pisinger and Røpke 2010). In the LNS metaheuristic, the main idea is to perform successive destroy and repair operations over the current solution to try and optimize its quality. In our case, a destroy operation consists in removing x% of the meshes allocated to each system, and a repair operation consists in using the chronological pass-filling process to cover the whole area again. These mechanisms are illustrated in Fig. 7 over a few steps. When the destroy and repair operations do not improve the current solution during several iterations (parameter referred to as *nItNoImprovement*), the algorithm restarts from an empty plan. The search continues until a maximum CPU time is reached. Finally, the best solution found is returned. On this point, to compare the solutions found during the search process, we use the four metrics of Equations 5-8 in a lexicographic order.

For the destroy phase, several mesh selection heuristics can be used, such as giving a preference to the removal of meshes having a low coverage score, a low mesh grouping score, or a large estimated completion time. But to add diversity in the search process, the meshes deactivated at each destroy phase are chosen randomly among the set of meshes that have non-selected neighbors in the current solution, or in other words randomly among the set of meshes placed at the frontier of a group of meshes.

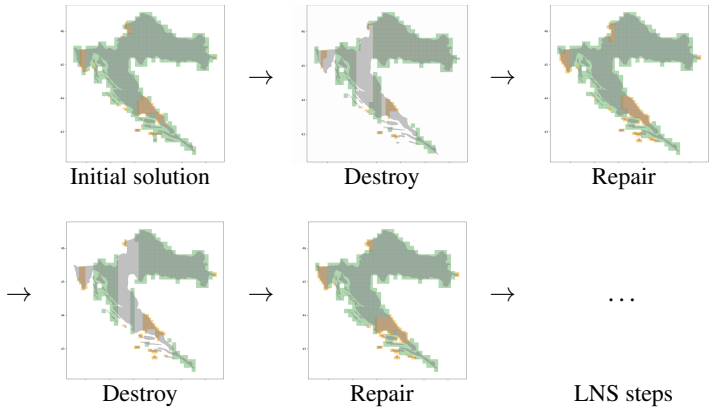

Figure 7: LNS procedure using successive destroy and repair operations, together with restart mechanisms

4.3 Re-dispatch algorithm

Each planning step of the federation layer produces a dispatch strategy that allocates meshes to different observation systems. One issue is that the set of observations actually achieved by each system may significantly differ from the set of observations expected by the federation layer. Indeed, the latter only takes into account a coarse-grain model of the observation capacities of the satellites, it does not model the precise behavior of the specific planner used by each individual mission, and it does not take into account the arrival of new requests. Moreover, the cloud cover forecast may differ from the actual cloud cover, hence some meshes may have to be observed again to get valid images. Last, it is relevant to regularly take into account potential deviations concerning the actual orbital positions of the satellites. For all these reasons, we regularly adapt the mesh dispatch strategy. To do this, we directly reuse the LNS algorithm detailed before, starting either from an empty solution, or from the solution obtained after the last dispatch phase of the federation layer. During the re-dispatch phase, we also update the input data, by adding observation requests received since the last optimization step and removing the areas over which valid images have been collected. The optimization of plan stability metrics can be relevant but is left for future works.

5 Experimental results

This section provides experimental results obtained for both mesh dispatching and re-dispatching scenarios. We consider a unique coverage request and empty orderbooks, even if the algorithms defined can deal with multiple requests and non-empty orderbooks. The results are obtained over processors Intel(R) Core(TM) i5-8400H CPU 2.50GHz for the dispatch case and processors Intel(R) Xeon(R) CPU E5-2660-v3 2.60GHz for the re-dispatch case.

5.1 Dispatch case

Scenarios We consider coverage requests over six areas of various shapes and sizes: Netherlands, Denmark, Croatia, Occitanie (French region), France (Metropolitan), and Chile.

To achieve the coverage tasks, two observation systems using sun-synchronous orbits are available:

- System S1 (called *CO3D*) composed of 4 satellites (semi-major axis: 6880km; mesh size: 5km×7km).
- System S2 (called *PNEO*) composed of 2 satellites (semi-major axis: 7000km; mesh size: 13km×13km).

From the full set of orbital parameters, we compute the time periods during which each satellite overflies the area of interest and may observe relevant meshes given maximum observation angle constraints. Moreover, for each satellite of each system, the cycle time is equal to 26 days, meaning that in a geocentric coordinate frame, each satellite comes back to its initial position after this duration. The observation time of each satellite is divided into successive time slots. For the experiments, the duration of each slot is set to 20 seconds. Each satellite of S1 can observe up to 10 meshes per slot ($Capacity_{sk} = 10$) and each satellite of S2 can observe up to 3 meshes per slot ($Capacity_{sk} = 3$).

Results Fig. 8 gives solutions produced by the LNS algorithm for the six areas of interest, using a maximum CPU time of 2 minutes except for instances Chile and France, where this CPU time is arbitrarily raised to 5 minutes because these instances contain a higher number of cells and meshes, and better solutions are found between 2 minutes and 5 minutes of computational time. Globally, we can see that meshes are dispatched to the two observation systems available (S1 in green, S2 in yellow), the meshes allocated to a system are rather grouped, and the overlapping between meshes allocated to different systems is rather low.

Table 1 gives details about the number of candidate meshes for the two systems (**#msh-S1** and **#msh-S2**), the number of cells (**#cells**), the time required to decompose the coverage area into meshes and cells and initialize the satellite slots (**tIni**), the time required to get a first solution using the chronological greedy pass-filling algorithm (**tGr**), the mean number of destroy and repair operations performed per second for LNS (**#it/s**), and the estimated number of days required to cover the area by using both S1 and S2 (**#days-Sall**) or each system alone (**#days-S1**, **#days-S2**). Globally, the table shows that the algorithm manages to produce solutions for instances involving several thousands of cells in a few seconds, and solutions for instances involving tens of thousands of cells in about one minute. They also show that using the two systems together is beneficial to minimize the estimated completion time of the coverage.

For the Croatia benchmark, Fig. 9 shows, at each iteration of the algorithm, the values obtained for the four solution evaluation metrics (to be minimized in a lexicographic order): (1) coverage completion time in days, (2) loss in the mesh grouping reward, (3) area wastage due to overlapping meshes or portions of meshes located outside of the area of interest, (4) average mesh completion time. Here, an iteration refers to the application of one *destroy and repair* step of the LNS algorithm. The results show that the iteration at which the best solution is found (dashed red bar) is not systematically one of the first iterations. This means that restarting from empty solutions after a number of search steps without improvement helps finding better solutions.

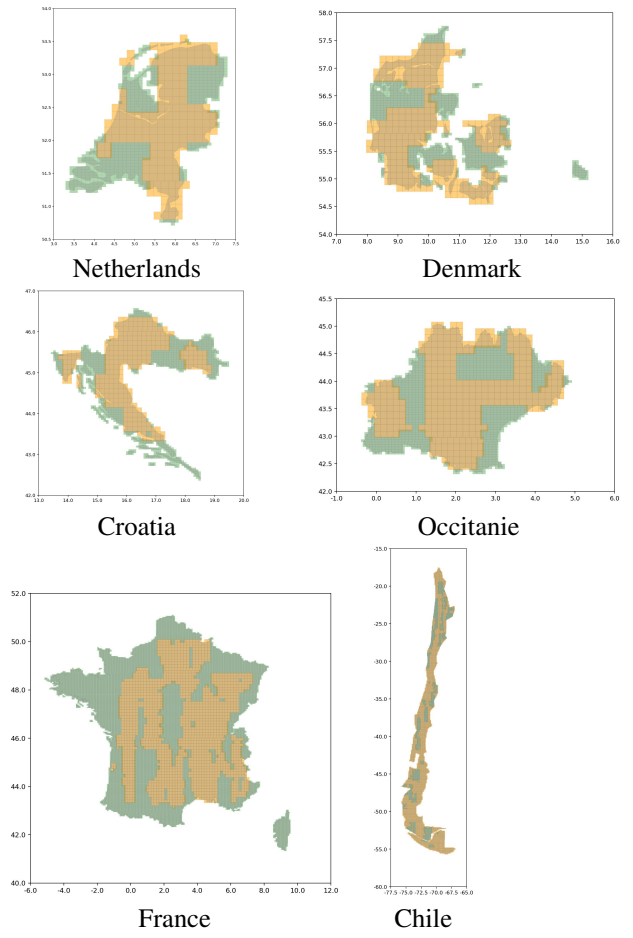

Figure 8: Dispatch solutions found for several areas (meshes of S1 in green, meshes of S2 in orange); configuration: destroy percentage = 20%, nItNoImprovement = 15, maxCpuTime = 2 min for instances Netherlands, Denmark, Croatia, Occitanie, and 5 min for France, Chile

instance	#msh		#cells	tIni(s)	tGr(s)	#it/s	#days		
	S1	S2					Sall	S1	S2
Nthlands	298	1255	2400	2.31	0.83	8.35	12	39	27
Denmark	407	1621	3047	3.21	1.05	6.27	12	40	30
Croatia	497	2059	3903	2.55	1.32	3.56	20	62	48
Occitanie	517	2322	4572	3.16	2.08	2.49	24	77	59
France	3628	16836	33336	30.82	38.77	0.12	69	189	130
Chile	5497	25236	49492	21.73	57.98	0.03	18	34	24

Table 1: Results for the dispatch case on different instances; configuration: destroy percentage = 20%, nItNoImprovement = 15, maxCpuTime = 2 min for instances Netherlands, Denmark, Croatia, Occitanie, and 5 min for France, Chile

5.2 Re-dispatch case

Scenarios To evaluate the method proposed in the re-dispatch case, we simulate (1) the mission planning engine associated with each system, that actually plans the observation of meshes given the current orderbook, and (2) the rejection of mesh observations due to the cloud cover condi-

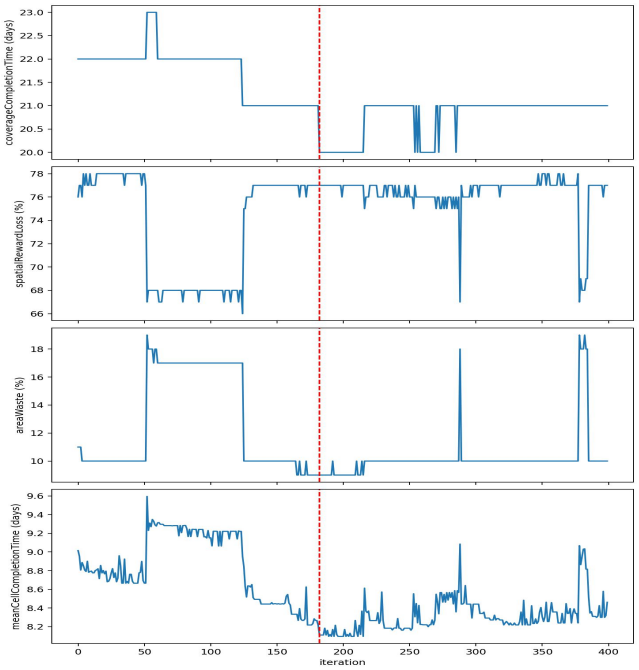

Figure 9: Evolution of the solution evaluation metrics along the iterations of LNS; configuration: destroy percentage = 20%, nItNoImprovement = 15, maxCpuTime = 2 min

tions. As a result, we simulate one day of operations of each satellite as follows:

- for each slot k of a satellite of system s over the day, we randomly choose $Capacity_{sk}$ meshes that are visible during slot k and not observed yet (random choice using a uniform probability distribution); if the number of meshes satisfying these conditions is less than $Capacity_{sk}$, we select all the candidate meshes;
- we consider that each mesh selected is planned and validated with a certain probability. For the experiments we consider a coarse-grain simulation based on two acceptance scenarios:
  - Scenario A "High workload in the orderbook of S1": mesh acceptance probability randomly chosen in $[0.1, 0.5]$ for S1 and $[0.6, 1.0]$ for S2;
  - Scenario B "High workload in the orderbook of S2": mesh acceptance probability randomly chosen in $[0.6, 1.0]$ for S1 and $[0.1, 0.5]$ for S2.

We perform experiments for three reassessment periods (1 day, 7 days, and 14 days) together with the no-reassessment case. At each reassessment phase, we recompute the mesh dispatching strategy, and to limit the duration of the simulations we use a maximum CPU time of one minute for each reassessment phase. We do not consider instances Chile and France that contain a higher number of meshes and cells and require higher simulation times to test all the configurations.

Results Fig. 10 shows the evolution of the completion rate over time in Scenario A for different areas of interest and different reassessment periods (case R1000 corresponds to the no-reassessment case). The results for Scenario B are similar. Globally, we can see that using a shorter reassessment period is beneficial, especially during the second half of the coverage period. This is because during the first reassessment periods, each system somehow always has some workload to do, while during the last reassessment periods, one of the systems may be idle while another one is struggling with a high workload. Using a shorter reassessment period, e.g. 1 day, helps such situations to be avoided. Conversely, as shown by configuration "Croatia instance, Scenario A", it may happen that reassessing too often decreases the performance, possibly because keeping a stable dispatch strategy is sometimes better than systematically reacting to small disturbances in the current progress.

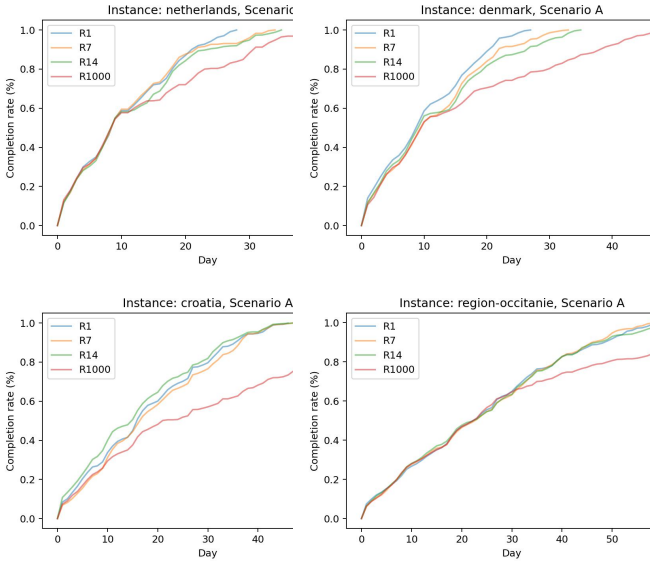

Figure 10: Impact of the reassessment period on the evolution of the completion rate

6 Conclusion and Perspectives

This paper introduced a model of a mesh dispatch problem for a federation layer, together with heuristic and large neighborhood search methods. There are several perspectives for this work. First, further experiments must be performed in the multi-request case. Second, historical weather data could be exploited to better deal with sub-areas that are hard to capture due to bad local meteorological conditions on average; such sub-areas could be allocated to several systems in parallel. Third, the case of meshes that do not have a North-South orientation (e.g., for inclined orbits) should be tackled. Last, experiments must be performed with the actual planning engines used by the individual missions.

7 Acknowledgments

This work has been performed in the DOMINO-E European project led by Airbus Defence and Space and involving Onera, Airbus, and CapGemini for the tasks related to the definition of a federation layer dealing with coverage requests.

References

Berger, J.; Lo, N.; and Barkaoui, M. 2020. QUEST – A new quadratic decision model for the multi-satellite scheduling problem. *Computers & Operations Research*, 115: 104822.

Chen, Y.; Xu, M.; Shen, X.; Zhang, G.; Lu, Z.; and Xu, J. 2020. A Multi-Objective Modeling Method of Multi Satellite Imaging Task Planning for Large Regional Mapping. *Remote sensing*, 12(3).

Farges, J.-L.; Perotto, F.; Picard, G.; Pralet, C.; de Lussy, C.; Guerra, J.; Pavero, P.; and Planchou, F. 2024. Going Beyond Mono-Mission Earth Observation: Using the Multi-Agent Paradigm to Federate Multiple Missions. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2674–2678.

Ji, H.; and Huang, D. 2019. A mission planning method for multi-satellite wide area observation. *International Journal of Advanced Robotic Systems*, 16(6).

Lemaître, M.; Verfaillie, G.; Jouhaud, F.; Lachiver, J.-M.; and Bataille, N. 2002. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6(5): 367–381.

Lenzen, C.; Dauth, M.; Fruth, T.; Petrak, A.; and Gross, E. 2021. Planning Area Coverage with Low Priority. In *International Workshop on Planning and Scheduling for Space*.

Liu, S.; and Hodgson, M. 2013. Optimizing large area coverage from multiple satellite sensors. *GI Science & Remote Sensing*, 50.

Maillard, A.; Chien, S.; and Wells, C. 2021. Planning the Coverage of Solar System Bodies Under Geometric Constraints. *Journal of Aerospace Information Systems*, 18(5).

Niu, X.; Tang, H.; and Wu, L. 2018. Satellite scheduling of large areal tasks for rapid response to natural disaster using a multi-objective genetic algorithm. *Journal of Disaster Risk Reduction*, 28.

Ntagiou, E. V.; Armellin, R.; Iacopino, C.; Policella, N.; and Donati, A. 2018. Ant Based Mission Planning: Two Examples. In *15th International Conference on Space Operations*.

Pisinger, D.; and Røpke, S. 2010. *Large Neighborhood Search*, 399–420. Springer, 2 edition.

Shao, E.; Byon, A.; Davies, C.; Davis, E.; Knight, R.; Lewellen, G.; Trowbridge, M.; and Chien, S. 2018. Area Coverage Planning with 3-axis Steerable, 2D Framing Sensors. In *International Conference on Automated Planning and Scheduling*.

Zhibo, E.; Shi, R.; Gan, L.; Baoyin, H.; and Li, J. 2021. Multi-satellites imaging scheduling using individual reconfiguration based integer coding genetic algorithm. *Acta Astronautica*, 178: 645–657.