Robust Observation Planning via Facet Reasoning

Jennifer Santos[*1] , **Damien Van Meerbeeck** [*1] , **Mika Skjelnes**[1] ,
Arnaud Lequen[1] and **Daniel Gnad**[1,2]
[1]Linköping University
[2]Heidelberg University
{jennifer.santos, damien.van.meerbeeck, mika.skjelnes, arnaud.lequen}@liu.se,
daniel.gnad@uni-heidelberg.de

Abstract

Observation planning is the task of scheduling the operations of a satellite constellation to take pictures of a set of regions of interest (ROIs) and sending these observations down to Earth. With large constellations, uncertain observation conditions, e.g. due to clouds, and limited availability of downstream capacity, it becomes increasingly hard to coordinate the operations. We address the setting where satellites have limited memory, and pictures need to be sent through the constellation to a satellite that can transmit them to a ground station. A key challenge in this problem is the highly dynamic environment, which requires a robust planning system that can quickly adapt to changes. We propose a framework based on *facet reasoning*, which, unlike traditional planners that provide a single fixed solution, allows for dynamic updates to the current solution without replanning from scratch. We show empirically that facets offer a scalable approach for constellation planning and illustrate the robustness in a case study.

1 Introduction

Earth observation (EO) missions rely on scheduling limited satellite resources to collect images of regions of interests (ROIs) and deliver the resulting data to ground stations. As recent surveys emphasize, this family of problems includes both imaging and communication decisions and remains a central optimization challenge for modern systems [Bianchessi *et al.*, 2007; Ferrari *et al.*, 2025; Zilberstein *et al.*, 2025], where multiple satellites need to be coordinated jointly.

For operational EO constellations, image acquisition cannot be planned independently from data delivery. Downlink opportunities are scarce, on-board memory is limited, and transmission constraints interact directly with acquisition choices [Bianchessi and Righini, 2008; Chen *et al.*, 2020]. In the setting considered here, these communication constraints are even more restrictive: a satellite may need to forward an image through the constellation before it can reach a ground station. This makes the problem naturally one of automated planning, where sensing, routing, and downlink decisions must be synchronized over time.

A further challenge is that observation opportunities are uncertain and can change rapidly. Cloud cover and related weather effects degrade observation quality and are a standard source of uncertainty in EO scheduling [Xiao *et al.*, 2019; Wang *et al.*, 2021]. More broadly, dynamic environments motivate planning systems that can revise a current plan quickly rather than restart from scratch each time conditions change [Fox *et al.*, 2006; Little and Thiébaux, 2007]. Our goal is therefore to support robust observation planning for satellite constellations in a form that can be updated interactively as new constraints or failures arise.

We propose to model constellation observation planning in PlanPilot [Gnad *et al.*, 2025b], a recently proposed tool for the interactive exploration of plan spaces. Rather than committing to a single solution, PlanPilot encodes a planning task in Answer-Set Programming (ASP) and lets a user navigate the set of all valid plans of a bounded horizon by iteratively enforcing or forbidding individual actions, called *facets*. We show how to use this faceted reasoning to represent operational constraints on the plan. To this effect, we adapt PlanPilot's encoding in order to allow temporal reasoning. In addition, unlike a conventional one-shot planner that returns a single fixed plan, our approach supports iterative refinement of the current solution and fast adaptation to changing conditions without having to go through a full replanning phase.

Our paper is organized as follows. Section 2 provides background on satellite observation planning and the PlanPilot framework. Section 3 formalizes the constellation observation planning problem, including the communication constraints that distinguish our setting. Section 4 presents our ASP encoding that extends PlanPilot to handle temporal reasoning and satellite-specific constraints. Section 5 demonstrates our approach on benchmark instances of varying scale and complexity. Section 6 analyzes the experimental results and discusses the trade-offs between solution quality and computational efficiency. Finally, Section 7 discusses our results and outlines future research directions.

2 Background

Automated Planning. Classical planning tasks are typically specified in PDDL [McDermott *et al.*, 1998], a lifted

[*]Equal contribution

representation of the task. Most modern planners go through a grounding phase, which translates PDDL tasks into the (ground) SAS$^+$ formalism [Bäckström and Nebel, 1995].

A SAS$^+$ planning task is a tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$, where $\mathcal{V}$ is a finite set of *state variables*, each variable $v \in \mathcal{V}$ having a finite domain $dom(v)$. An *atom* is a pair $\langle v, d \rangle$ with $v \in \mathcal{V}$ and $d \in dom(v)$. A *partial state* is a function s defined on a subset $\mathcal{V}(s) \subseteq \mathcal{V}$ such that $s[v] \in dom(v)$ for every $v \in \mathcal{V}(s)$; it is a *state* when $\mathcal{V}(s) = \mathcal{V}$. We often identify partial states with the corresponding sets of atoms, so that $\langle v, d \rangle \in s$ is equivalent to $s[v] = d$. We write $\mathcal{S}$ for the set of all states.

Each action $a \in \mathcal{O}$ is a pair $a = \langle pre(a), \mathit{eff}(a) \rangle$, where $pre(a)$ and $\mathit{eff}(a)$ are partial states called the *precondition* and *effect* of a. Action a is *applicable* in a state s whenever $pre(a) \subseteq s$, and applying it yields the state $s[\![a]\!]$ that agrees with $\mathit{eff}(a)$ on $\mathcal{V}(\mathit{eff}(a))$ and with s elsewhere. The state I is the *initial state* and the partial state G is the *goal*. A *plan* for Π is a sequence of actions $\langle a_1, \ldots, a_n \rangle$ that is iteratively applicable from I and whose final state s_n satisfies $G \subseteq s_n$. Given a horizon $h \in \mathbb{N}$, we denote by $Plans_h(\Pi)$ the set of plans for Π of length exactly h

Faceted Reasoning. Faceted reasoning has its roots in propositional logic, where a *facet* of a theory is an atom that appears in some but not all of its models [Fichte *et al.*, 2022]. Activating a facet restricts the set of models without rendering the theory unsatisfiable, and therefore provides a meaningful unit of variability that can be enforced or forbidden by the user. Gnad *et al.* [2025a] adapted this notion to classical planning by considering action occurrences as facets. Given a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ and a horizon h, the pair (a, t) with $a \in \mathcal{O}$ and $1 \leq t \leq h$ is a facet if there exists at least one plan in $Plans_h(\Pi)$ in which action a is applied at step t and at least one plan in $Plans_h(\Pi)$ in which it is not. A *route* is a finite sequence of such enforcing or prohibiting steps that records the user's navigation history through the plan space and that can be undone or extended at any point.

PlanPilot. PlanPilot [Gnad *et al.*, 2025b] is an interactive plan-space navigation tool that operationalises the above ideas. It takes as input a planning task specified in PDDL or SAS$^+$ together with a horizon h, and translates it into an ASP encoding via the `plasp` compiler [Dimopoulos *et al.*, 2019]. The resulting program is passed to `fasb` [Fichte *et al.*, 2022], which, using `clingo`[1] as its underlying solver, computes the available facets and reasons about the remaining plan space after a facet is (de)activated. At each interaction step, the user can query PlanPilot for the set of facets that are still available, the number of plans that satisfy the currently active route, or an enumeration of these plans; activating a facet restricts the plan space accordingly, while deactivating one rolls back the corresponding restriction. Beyond enforcing or forbidding the occurrence of an action at a specific time step, PlanPilot also supports *abstract time-step* facets of the form $occurs\_sometime(a)$, which require that action a occurs at *some* step along the plan without fixing which one. Because reasoning over facets is significantly cheaper than reasoning over plans [Speck *et al.*, 2025], this design enables responsive interaction even when the underlying plan space is very large, which we leverage in this work to make replanning efficient.

[1] https://github.com/potassco/clingo

3 Satellite Constellation Problem

Assumptions and Specification of the Domain The domain we study is motivated by integrated EO scheduling, in which imaging decisions and communication decisions must be solved together rather than in isolation [Bianchessi and Righini, 2008; Chen *et al.*, 2020; Xiao *et al.*, 2019]. This is particularly relevant for constellation settings, where several satellites may cooperate to satisfy requests under limited downlink capacity and changing observation conditions [Bianchessi *et al.*, 2007; Wang *et al.*, 2021; Zilberstein *et al.*, 2025]. We adopt the following assumptions to obtain a clean planning model while preserving the core operational bottlenecks:

- A satellite has a restricted set of tools, e.g. camera types.
- A satellite has a memory capacity of one picture.
- The satellites are at a fixed distance from each other, assuming a gridlike formation.
- There is a fixed number of ROIs, and a fixed number of ground stations.
- Satellites can transmit pictures to Earth if they are in range of the ground station, and can transmit pictures to adjacent satellites.

These assumptions capture the aspect of the application that most strongly motivates our approach: feasible plans depend on narrow temporal windows for both observation and transmission, and these windows may become unusable or less valuable as conditions evolve, in the long or short term. Weather, faults, or communication restrictions can invalidate planned actions and force the system to revise the current plan quickly [Wang *et al.*, 2021; Xiao *et al.*, 2019]. In such settings, repairing or refining an existing plan is often preferable to recomputing a new plan from scratch [Fox *et al.*, 2006; Little and Thiébaux, 2007].

Two features of this setting drive the design choices we make in the rest of the paper. The first is that the actions available to a satellite at any given time step are heavily restricted, both by foreseen and by unforeseen constraints. Foreseen constraints follow directly from orbital geometry: a satellite can only acquire an image of a ROI while it is in view of that region, and can only downlink to the ground while it is in line of sight of a ground station, so observation and transmission opportunities come in narrow, periodic windows. Unforeseen constraints arise on top of this static schedule, when conditions evolve in ways that the original plan did not anticipate. For instance, a cloud cover event can invalidate an observation window, a hardware fault can disable a satellite, or a communication restriction can close a ground link. In both cases the consequence is the same: only a small subset of the actions that the planning model allows in principle are actually applicable at runtime.

The second feature is that, because of these restrictions, replanning is the norm rather than the exception. Plans for this domain rarely survive their intended horizon untouched:

an unusable window or a failed transmission typically forces the operator to revise the current schedule. Recomputing a plan from scratch each time such an event occurs is wasteful when most of the previous solution remains valid, and is often undesirable operationally, since it may discard manual adjustments made by the user. This motivates an approach in which the current plan can be incrementally refined under new constraints, which is exactly the kind of interaction that faceted reasoning over plan spaces is designed to support.

4 Encoding into PlanPilot

4.1 PDDL Model

PDDL Domain We encode the constellation observation problem as a PDDL domain with two object types, $satellite$ and $image$, and six predicates. Three predicates describe the dynamic state of the constellation: $has\text{-}image(s, i)$ indicates that satellite s currently holds image i in its on-board memory; $is\text{-}full(s)$ marks that memory as occupied (recall that each satellite holds at most one image at a time); and $transmitted(i)$ records that image i has reached the ground.

The remaining three predicates act as static *gates* whose role is to expose, at the planning level, the opportunities that the operational environment offers at each time step: $connected(s_1, s_2)$ captures the intra-grid neighbourhood between satellites and is treated as time-invariant in this model; $can\text{-}take\text{-}image(s, i)$ is true whenever satellite s may acquire image i; and $can\text{-}earth\text{-}transmit(s)$ is true whenever satellite s is in line of sight of a ground station.

Actions There are three actions in our domain. Action $take\text{-}picture(s, i)$ requires $can\text{-}take\text{-}image(s, i)$ together with an empty memory and produces $has\text{-}image(s, i)$ and $is\text{-}full(s)$. Action $transmit\text{-}satellite(s_1, s_2, i)$ forwards an image to a connected neighbour with free memory, freeing the source satellite. Action $transmit\text{-}earth(s, i)$ requires $can\text{-}earth\text{-}transmit(s)$ and delivers the image to the ground, asserting $transmitted(i)$. The goal of every problem instance is that every image be transmitted to the ground.

Instance A problem instance is described by a file that specifies the geometry and the schedule of observation opportunities. The constellation is organised in L parallel layers of m satellites each, with the convention that satellite $s_{i,j}$ sits at position $i \in \{1, \ldots, m\}$ in layer $j \in \{1, \ldots, L\}$; the $connected$ relation is then derived from this grid. The file lists n ROIs and specifies, for each image, the (satellite, offset) pairs at which a picture may be taken, and, for each satellite, the offsets at which it can transmit to the ground. These offsets are interpreted modulo the orbital period P, which is the number of time steps after which the constellation returns to its initial geometric configuration and the visibility windows recur: an offset list $\langle o_1, \ldots, o_r \rangle$ paired with period P activates the corresponding gate at every time step t such that $t \equiv o_l + 1 \pmod{P}$ for some $l \in \{1, \ldots, r\}$.

A non-obvious aspect of this model is that the underlying temporal constraints (i.e. when a satellite can image a region or downlink to the ground) are *not* expressed in the PDDL domain itself, which contains neither durative actions nor explicit time literals. Instead, the $can\text{-}take\text{-}image$ and $can\text{-}earth\text{-}transmit$ predicates serve as proxies whose truth values across the planning horizon are scheduled outside PDDL, in the ASP encoding described below. This separation keeps the PDDL domain compact and reusable across instances, while delegating all temporal reasoning to the layer on which PlanPilot operates.

4.2 Extending PlanPilot with Temporal Constraints

PlanPilot's stock encoding produces sequential plans in which exactly one action is applied per time step. For our setting, this is too restrictive in two respects. First, distinct satellites may act in parallel within the same orbital step (for instance, one taking a picture while another forwards a previously acquired image) and forcing these actions into separate steps inflates the horizon and the size of the plan space. Second, the periodic visibility windows of the instance only constrain *when* a gate predicate may be true, not which satellite is acting at that step, so the planning horizon is more naturally expressed in terms of orbital steps than of action steps.

We therefore extend the encoding to admit *parallel* action occurrences per step under $\forall$-step semantics, where a set of actions may fire jointly at step t provided that every linearisation of the set is applicable in the state reached at step t and produces the same valid state at step $t + 1$. We then drive the predicates $can\text{-}take\text{-}image$ and $can\text{-}earth\text{-}transmit$ from the periodic schedule of the instance directly inside the ASP program. Facets, which originally identified a single action occurrence $occurs(a, t)$, are then extended to range over the parallel occurrences and over the predicate schedule itself, which is what allows the user to enforce or forbid temporal constraints on observation windows from within PlanPilot.

In the case where no action occurs at a time step, for instance because no satellite can perform a meaningful action at this specific point in time, the solver does not assign any action to that step of the plan.

5 Demonstration

We evaluate the capabilities of our approach empirically by building a set of benchmarks modeling EO problems as described before, and solving them using our adaptation of PlanPilot.

5.1 Building the Benchmark Sets

We construct three benchmark sets, each containing six instances. The first two assess scalability with respect to the number of images to be acquired, with the second one introducing a sensor constraint. The third assesses scalability with respect to the size of the satellite constellation.

Satellite constellation As illustrated in Figure 1, our satellite constellation model consists of multiple orbital layers, each containing a fixed number of m satellites arranged in a symmetric configuration. The constellation includes a ground station that serves as the final destination for all collected imagery data. Communication capabilities vary across satellites: while all satellites can establish links with their immediate neighbors (either in adjacent layers at the same orbital position, or within the same layer at adjacent positions), only

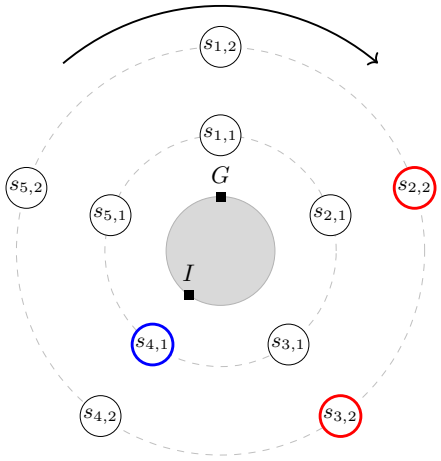

Figure 1: Satellite constellation of `prob-5-2-1`, with G marking the ground station and I the ROI. Satellites highlighted in red are equipped with cameras, while the satellite highlighted in blue is equipped with the communication arrays required to send data back to Earth. The arrow indicates the direction of orbital motion.

a subset is equipped with specialized communication arrays capable of transmitting data directly to the ground station.

The satellites are heterogeneously equipped with different EO instruments, including optical and infrared cameras, reflecting constraints where sensor types are distributed across the constellation. The primary objective is to successfully observe all designated ROIs and ensure the acquired imagery reaches the ground station within the mission timeline.

The satellites follow periodic orbital trajectories around Earth, with each satellite returning to its initial position after exactly m time steps.

Set 1: Expanding Images In the first benchmark set, we isolate the effect of increasing the number of images. The constellation is fixed at $L = 3$ layers, with $m = 8$ satellites per layer and period $P = 8$, while the number of images $n \in \{2, \ldots, 7\}$ varies. Each image can be acquired by at most two satellites and transmitted to Earth by a third satellite.

Set 2: Infrared Images The second benchmark set uses the same constellation parameters as the first: $L = 3$ layers, $m = 8$ satellites per layer, period $P = 8$ and $n \in \{2, \ldots, 7\}$. Unlike the previous set, the images are divided into two categories: optical and infrared. This models a scenario in which satellites are equipped with different sensors, namely a standard camera or an infrared camera. Accordingly, one satellite is restricted to acquiring optical images, while the other is restricted to acquiring infrared images. For each value of n, the number of infrared images is set to $\lfloor n/2 \rfloor$ and the remaining images are assigned to the optical category.

Set 3: Expanding Satellites In the final benchmark set, we fix the number of images and scale the size of the constellation. Specifically, we set $L = 3$ layers and $n = 2$ images, while varying the number of satellites per layer $m \in \{9, \ldots, 14\}$ with the period equal to the number of satellites, $P = m$. Following the previous sets, only one satellite can transmit images to Earth.

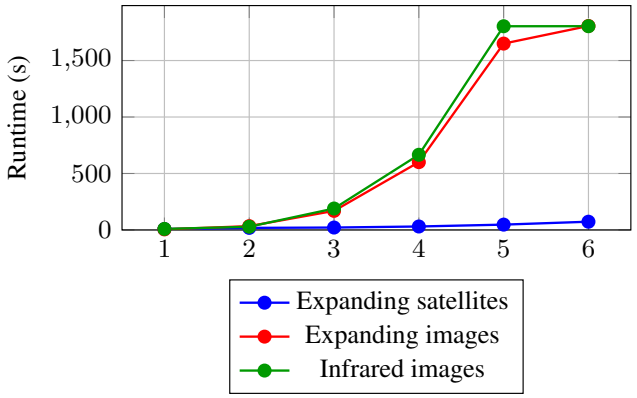

Figure 2: Runtime of PlanPilot for each solved instance across the three benchmark sets. Instances are ordered by increasing order of difficulty within their set.

5.2 Experimental Setup

We evaluate the performance of PlanPilot on the described benchmarks using the Lab toolkit [Seipp *et al.*, 2017]. For each instance, the search horizon is fixed to the precomputed optimal horizon h^*. Obtaining the horizon is generally simpler than facet reasoning. Each run is limited to 30 minutes and 4 GiB of memory. All experiments were run on a laptop with AMD Ryzen 7 250 CPU and 32GB of RAM, running Ubuntu 24.04 with two parallel processes. The instance is solved using PlanPilot with a script that instructs the solver to return the first solution found.

6 Results

Performance Table 1 reports coverage and runtime of PlanPilot for each instance. Figure 2 shows the runtime behavior when scaling the instance size. When extending the image set, all instances up to $n = 6$ are solved, but runtime increases steeply as the number of images grows. The infrared benchmark set follows a similar trend, although the first unsolved instance occurs earlier, at $n = 5$. Even though the additional sensor constraint makes the problem more difficult, non-trivial instances of that benchmark set are still within reach of our approach. In addition, the last set of experiments shows that a higher number of satellites does not make the problem significantly harder to handle, as all instances in that benchmark set are solved, with runtime increasing only slightly as m grows.

Constraints Modeling Unexpected Events To illustrate how additional constraints can be incorporated interactively to account for unforeseen events, we consider the instance `prob-5-2-1`, as shown on Figure 1, and demonstrate the capabilities of our method. This instance has $m = 5$ satellites per layer, $L = 2$ layers, $n = 1$ image and period $P = 5$.

In this constellation, only satellite 2 in the second layer ($s2 - l2$) and satellite 3 in the second layer ($s3 - l2$) can acquire the image, while satellite 4 in layer 1 ($s4 - l1$) is the only satellite that can transmit images to Earth. The ROI is located three time steps from the starting point, and the ground station is located at the starting point, see Figure 1. For this planning problem, the optimal horizon is $h^* = 7$.

Benchmark set	Instance	$\|\pi^*\|$	Solved	Time (s)
Expanding	`prob-8-3-2`	17	✓	5.21
image	`prob-8-3-3`	25	✓	33.88
	`prob-8-3-4`	33	✓	169.06
	`prob-8-3-5`	41	✓	600.44
	`prob-8-3-6`	49	✓	1649.99
	`prob-8-3-7`	57	×	–
Infrared	`prob-8-3-2-inf`	17	✓	9.09
image	`prob-8-3-3-inf`	25	✓	28.56
	`prob-8-3-4-inf`	33	✓	189.73
	`prob-8-3-5-inf`	41	✓	665.66
	`prob-8-3-6-inf`	49	×	–
	`prob-8-3-7-inf`	57	×	–
Expanding	`prob-9-3-2`	19	✓	7.95
satellites	`prob-10-3-2`	21	✓	18.31
	`prob-11-3-2`	23	✓	21.41
	`prob-12-3-2`	25	✓	30.66
	`prob-13-3-2`	27	✓	47.19
	`prob-14-3-2`	29	✓	72.98

Table 1: Detailed results on the benchmark set. Instances are named `prob-`m`-`L`-`n, where m denotes the number of satellites per layer, L the number of layers, and n the number of images. A "`-inf`" postfix indicates two image categories: normal and infrared. $|\pi^*|$ is the minimal solvable horizon. Solved indicates whether the instance is solved (✓) or not (×). The last column shows the runtime.

We run the framework in interactive mode and obtain one plan using the command `! 1` (see Figure 3). In the plan, both eligible satellites acquire the image while passing over the ROI. However, only the copy acquired by $s3-l2$ is communicated to $s4-l1$ and transmitted to Earth. The copy acquired by $s2-l2$ continues to be communicated through the constellation, but does not reach the ground station within the horizon. Suppose now that a cloud is predicted to cover the ROI at time step 2. We can block $s3-l2$ from acquiring an image at that time step using command + ∼occurs (...) and request a new solution (see Figure 4). The resulting plan makes $s2-l2$ acquire the image at time step 3. The satellite $s3-l2$ still acquires the image at time step 7, since the added constraint only prevents acquisition at time step 2. However, this later copy does not reach the ground station within the horizon.

Similarly, we can model a temporary maintenance constraint. Suppose satellite 5 in layer 2 ($s5-l2$) is unavailable for communication at time step 5. We prevent communication to and from this satellite at that time step and request a new solution (see Figure 5). The new plan avoids communication involving $s5-l2$ at time step 5, while still achieving the goal within the horizon.

```
! 1
solution 1:
occurs([take-pic,s3-l2,i1],2)
occurs([take-pic,s2-l2,i1],3)
occurs([transmit-sat,s2-l2,s1-l2,i1],4)
occurs([transmit-sat,s3-l2,s3-l1,i1],4)
occurs([transmit-sat,s1-l2,s5-l2,i1],5)
occurs([transmit-sat,s3-l1,s4-l1,i1],5)
occurs([transmit-sat,s5-l2,s5-l1,i1],6)
occurs([transmit-earth,s4-l1,i1],7)
occurs([transmit-sat,s5-l1,s1-l1,i1],7)
```

Figure 3: Initial solution for instance `prob-5-2-1`.

```
+ ∼occurs([take-pic,s3-l2,i1],2)
! 1
solution 1:
occurs([take-pic,s2-l2,i1],3)
occurs([transmit-sat,s2-l2,s3-l2,i1],4)
occurs([transmit-sat,s3-l2,s3-l1,i1],5)
occurs([transmit-sat,s3-l1,s4-l1,i1],6)
occurs([take-pic,s3-l2,i1],7)
occurs([transmit-earth,s4-l1,i1],7)
```

Figure 4: Solution with weather constraint blocking image acquisition at time step 2, shown in the first line.

7 Discussion

This work demonstrates the application of faceted planning to satellite constellation mission planning, showing how dynamic constraints can be effectively incorporated into the planning process. The framework successfully handles both environmental constraints (such as weather conditions) and operational constraints (such as maintenance schedules) while maintaining computational efficiency. The current framework provides a solid foundation that can be extended in several meaningful directions.

Enhanced satellite modeling Satellites can be encoded to have more sophisticated memory management capabilities. New predicates can be introduced to track the occupied storage space on each satellite, enabling more realistic modeling of data storage limitations and buffering strategies. This would allow for more complex mission scenarios where satellites must manage limited onboard storage while balancing image acquisition and data transmission priorities. Also note that additional sensors can be added, and that we are not restricted to different types of cameras.

Flexible constellation topologies The framework naturally supports diverse satellite constellation topologies through the facet mechanism. More complex topologies can be easily implemented, including dynamic topologies where the constellation structure changes over time due to orbital mechanics or mission requirements. The topology can be altered by using the exact same facet mechanism, providing unprecedented flexibility in constellation design and mission adaptation.

```
+ ∼occurs([transmit-sat,s4-l2,s5-l2,i1],5)
+ ∼occurs([transmit-sat,s5-l2,s4-l2,i1],5)
+ ∼occurs([transmit-sat,s5-l2,s1-l2,i1],5)
+ ∼occurs([transmit-sat,s1-l2,s5-l2,i1],5)
+ ∼occurs([transmit-sat,s5-l2,s5-l1,i1],5)
+ ∼occurs([transmit-sat,s5-l1,s5-l2,i1],5)
! 1
solution 1:
occurs([take-pic,s3-l2,i1],2)
occurs([take-pic,s2-l2,i1],3)
occurs([transmit-sat,s3-l2,s4-l2,i1],3)
occurs([transmit-sat,s2-l2,s2-l1,i1],4)
occurs([transmit-sat,s4-l2,s5-l2,i1],4)
occurs([transmit-sat,s2-l1,s3-l1,i1],5)
occurs([transmit-sat,s3-l1,s4-l1,i1],6)
occurs([transmit-sat,s5-l2,s4-l2,i1],6)
occurs([transmit-earth,s4-l1,i1],7)
occurs([transmit-sat,s4-l2,s5-l2,i1],7)
```

Figure 5: Solution with maintenance constraint preventing communication involving s5-l2 at time step 5.

8 Conclusion

We demonstrate the application of faceted planning to satellite constellation mission planning through three comprehensive benchmark sets, showcasing the scalability across multiple dimensions. Our approach successfully handles dynamic environmental constraints such as weather conditions and operational constraints like satellite maintenance schedules, showing how facets enable rapid replanning when conditions change. Our interactive framework allows real-time constraint addition and plan modification, demonstrating practical applicability for mission operations. Future work includes taking more explicit constraints into account, for instance by enforcing that some observations must be made at specific times, or that data has to be sent to Earth before a deadline, and enhancing scalability to larger constellations.

Ethical Statement

No satellite was harmed in the making of this paper.

Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

Christer Bäckström and Bernhard Nebel. Complexity results for SAS^+ planning. *Computational Intelligence*, 11(4):625–655, 1995.

Nicola Bianchessi and Giovanni Righini. Planning and scheduling algorithms for the COSMO-SkyMed constellation. *Aerospace Science and Technology*, 12(7):535–544, 2008.

Nicola Bianchessi, Jean-François Cordeau, Jacques Desrosiers, Gilbert Laporte, and Vincent Raymond. A heuristic for the multi-satellite, multi-orbit and multi-user management of earth observation satellites. *European Journal of Operational Research*, 177(2):750–762, 2007.

Yuning Chen, Ji Lu, Renjie He, and Junwei Ou. An efficient local search heuristic for earth observation satellite integrated scheduling. *Applied Sciences*, 10(16):5616, 2020.

Yannis Dimopoulos, Martin Gebser, Patrick Lühne, Javier Romero, and Torsten Schaub. plasp 3: Towards effective ASP planning. *Theory and Practice of Logic Programming*, 19(3):477–504, 2019.

Benedetta Ferrari, Jean-François Cordeau, Maxence Delorme, Manuel Iori, and Roberto Orosei. Satellite scheduling problems: A survey of applications in earth and outer space observation. *Computers & Operations Research*, 173:106875, 2025.

Johannes Klaus Fichte, Sarah Alice Gaggl, and Dominik Rusovac. Rushing and strolling among answer sets - navigation made easy. In *Proc. AAAI 2022*, pages 5651–5659, 2022.

Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan stability: Replanning versus plan repair. In *Proc. ICAPS 2006*, pages 212–221, 2006.

Daniel Gnad, Lee-or Alon, Eyal Weiss, and Alexander Shleyfman. PDBs go numeric: Pattern-database heuristics for simple numeric planning. In *Proc. AAAI 2025*, pages 26507–26515, 2025.

Daniel Gnad, Markus Hecher, Sarah Gaggl, Dominik Rusovac, David Speck, and Johannes K. Fichte. Interactive exploration of plan spaces. In *Proc. KR 2025*, pages 599–609, 2025.

Ian Little and Sylvie Thiébaux. Probabilistic planning vs replanning. In *ICAPS 2007 Workshop on the International Planning Competition: Past, Present and Future*, 2007.

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL – The Planning Domain Definition Language – Version 1.2. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University, 1998.

Jendrik Seipp, Florian Pommerening, Silvan Sievers, and Malte Helmert. Downward Lab. https://doi.org/10.5281/zenodo.790461, 2017.

David Speck, Markus Hecher, Daniel Gnad, Johannes K. Fichte, and Augusto B. Corrêa. Counting and reasoning with plans. In *Proc. AAAI 2025*, pages 26688–26696, 2025.

Xinwei Wang, Yi Gu, Guohua Wu, and John J. Woodward. Robust scheduling for multiple agile earth observation satellites under cloud coverage uncertainty. *Computers & Industrial Engineering*, 156:107292, 2021.

Yiyong Xiao, Siyue Zhang, Pei Yang, You Meng, and Jiaoying Huang. A two-stage flow-shop scheme for the multi-satellite observation and data-downlink scheduling problem considering weather uncertainties. *Reliability Engineering & System Safety*, 188:263–275, 2019.

Itai Zilberstein, Ananya Rao, Matthew Salis, and Steve Chien. Decentralized, decomposition-based observation scheduling for a large-scale satellite constellation. *Journal of Artificial Intelligence Research*, 82:169–208, 2025.