Reproducible Baselines and Execution Verification for Earth Observation Satellite Constellation Scheduling

*Tubolayefa Warekuromor
Head, Mission Planning Division, National Space Research and Development Agency

*Corresponding Author

Abstract
The Earth Observation Satellite Scheduling Problem (EOSSP) has seen significant algorithmic advances, yet the lack of standardized, open-source benchmarks has hindered fair comparison across studies. Recent large-scale benchmark suites have begun addressing this gap by providing high-fidelity simulation environments, but these frameworks typically focus on evaluating sophisticated methods, with limited systematic characterization of simple, reproducible baselines. This paper presents an open-source, modular scheduling framework that (1) provides complete implementations of FIFO (First-In-First-Out) and priority-based greedy heuristics as reproducible baselines, (2) offers a lightweight, extensible platform for rapid prototyping of scheduling algorithms, and (3) systematically characterizes the performance of these baselines across 50 synthetically generated test instances spanning five scales from 10 to 1,000 tasks. We formalize the scheduling problem incorporating rest-to-rest attitude manoeuvre dynamics, including angular velocity and acceleration limits, along with energy and storage constraints. Running the framework's own reference implementation surfaced two concrete defects worth reporting in their own right: a discontinuity in the originally specified slew-time kinematic model at the boundary between the triangular and trapezoidal velocity profiles, and a single-satellite scheduling gap that left additional constellation satellites unused in multi-satellite scenarios. Both are corrected here, and the corrected framework, including problem instance generators, evaluation metrics, and the two fixes, is released as open-source software. On the instances tested, priority-based greedy scheduling improves total scheduled priority over FIFO by 65 to 151 percent depending on scale, while achieving only 51.1 percent of the provably optimal objective value on small instances. This gap is reported transparently not to diminish the heuristic's practical utility, but to quantify the headroom available for more sophisticated methods relative to the simplest viable baseline.

Keywords: Earth Observation Satellites, Mission Planning, Task Scheduling, Benchmarking, Greedy Heuristics, Reproducibility, Constellation Operations

List of Acronyms
AEOS: Agile Earth Observation Satellite
CP-SAT: Constraint Programming–Satisfiability (the exact constraint-programming solver used here via Google OR-Tools to obtain provably optimal schedules)
EO: Earth Observation
EOSSP: Earth Observation Satellite Scheduling Problem
FIFO: First-In-First-Out
IAEOSSP: Integrated Agile Earth Observation Satellite Scheduling Problem

MBH: (the specific MIP-based heuristic proposed by Rocha et al., 2025)
MILP: Mixed-Integer Linear Programming - the specific, linear-constraint case of MIP whose optimal solution this paper approximates via an exact CP-SAT solve in Section 4.3
MIP: Mixed-Integer Programming — the general family of optimization problems combining continuous and integer decision variables
NP-hard: a computational-complexity classification for problems for which exact optimisation generally becomes computationally intractable as problem size grows; this motivates fast heuristics for large instances

1. Introduction

1.1 Motivation and Context

Earth Observation (EO) satellite constellations have become indispensable infrastructure for environmental monitoring, disaster response, agricultural assessment, and national security applications. As constellations grow in size, the complexity of task scheduling increases substantially, since a typical EO mission planning system must reconcile many competing observation requests with orbital mechanics, attitude manoeuvre constraints, onboard power limits, and data storage capacities (Bensana et al., 1996; Lemaître et al., 2002; Bianchessi et al., 2007).

The scheduling problem for agile Earth observation satellites (AEOS) is a well-studied NP-hard combinatorial optimization problem (Bianchessi et al., 2007; Chu et al., 2017), and this implies that no algorithm is known to guarantee an optimal solution in a practical amount of time as the number of tasks grows, which is precisely why heuristic approaches (Section 3's algorithm implementations) are needed alongside exact methods for anything beyond small instances. Over the past two decades, researchers have developed a diverse array of solution approaches, including exact methods such as Mixed-Integer Programming (MIP) formulations (which are optimization problems combining continuous and integer decision variables, solved to provable optimality) and branch-and-bound algorithms (Bianchessi et al., 2007; Chu et al., 2017); heuristics and metaheuristics such as greedy algorithms, tabu search, adaptive large neighbourhood search, and orienteering-problem formulations (Vasquez & Hao, 2001; Lemaître et al., 2002; Peng et al., 2019); and, more recently, deep learning approaches evaluated on large-scale benchmark suites (Wang et al., 2025; Yin et al., 2026).

Despite this wealth of algorithmic development, a persistent challenge has been the lack of standardized, open-source benchmarks that enable fair and reproducible comparison across studies. Different researchers employ different problem formulations, constraint sets, scenario generators, and evaluation metrics, making it difficult to assess whether algorithmic improvements are genuine or artifacts of experimental design (Wang et al., 2025; Yin et al., 2026).

1.2 Recent Benchmark Initiatives

Two recent initiatives have begun to address this gap directly. AEOS-Bench (Wang et al., 2025) introduced a large-scale benchmark suite for realistic constellation scheduling, containing 16,410 scenarios generated via a high-fidelity Basilisk-based simulation platform, spanning 1 to 50 satellites and 50 to 300 imaging tasks per scenario, each with ground-truth scheduling annotations. Building on this benchmark, the authors proposed AEOS-Former, a Transformer-based scheduling model, which achieved a completion rate of 35.42 percent on the val-unseen split of AEOS-Bench (a held-out set of scenarios not seen during the model's training, used to test generalization), ahead of the strongest baseline tested (35.35 percent).

EOS-Bench (Yin et al., 2026) extended this effort substantially, providing a framework comprising 1,390 scenarios and 13,900 benchmark instances, scaling up to 1,000 satellites and 10,000 tasks, with a scenario-characterisation scheme quantifying structural difficulty (opportunity density, task flexibility, conflict intensity, satellite congestion) and a multi-algorithm suite spanning MIP solvers, metaheuristics, and reinforcement learning.

On the other hand, the Integrated Agile Earth Observation Satellite Scheduling Problem (IAEOSSP) formulation (Rocha et al., 2025) provides a mathematical framework integrating observation scheduling and attitude manoeuvre planning, together with a MIP-based heuristic (MBH) evaluated against exact and existing algorithms on benchmark instances.

1.3 The Remaining Gap: Baseline Characterization

The availability of AEOS-Bench, EOS-Bench, and the IAEOSSP formulation represents real progress. A gap remains, however, and that is the performance characteristics of simple, transparent baseline algorithms. Specifically, what a FIFO or priority-greedy scheduler achieves before any sophistication is added is not the focus of these frameworks. These frameworks are built primarily to showcase and compare sophisticated methods. Without a clear, reproducible baseline characterization, it is difficult to assess the marginal value that more sophisticated algorithms actually add, and difficult to reproduce reported "baseline" comparisons when implementation details are not published alongside them (Wang et al., 2025; Yin et al., 2026).

Fast heuristic approaches are relevant to operational settings because they can provide good-quality solutions within acceptable computational times, particularly where real-time decisions are required (Ferrari et al., 2025; She et al., 2018). Their speed and transparent decision rules can also make them useful as auditable baselines for mission planning and prototyping. That operational context directly motivates this paper. It is written from the perspective of mission planning and satellite scheduling officers who do not yet have an operational satellite to schedule, as well as for cases where the immediate priority is establishing a clear, well-understood baseline platform for training staff and prototyping scheduling logic, rather than introducing algorithmic novelty.

1.4 Contributions

This paper provides:

- A modular, open-source framework implementing FIFO and priority-based greedy scheduling heuristics as reproducible baselines, with complete source code and documentation.
- A formal problem formulation incorporating rest-to-rest attitude manoeuvre dynamics (angular velocity and acceleration limits), energy and storage constraints, and orbital access windows, following the general structure of the IAEOSSP framework (Rocha et al., 2025).
- Systematic performance characterization across 50 synthetically generated instances spanning five scales (10 to 1,000 tasks), reporting objective value, runtime, and resource utilization.
- Two concrete corrections to the framework's own reference implementation, identified only because the code was actually executed rather than left as a specification: a discontinuous slew-time kinematic model, and a single-satellite scheduling gap in ostensibly multi-satellite scenarios.
- A comparison of priority-based greedy scheduling against a CP-SAT-based exact solver on small instances, quantifying the specific gap between a simple heuristic and provable optimality on this framework's own instances, not a cross-paper comparison to other frameworks' reported numbers, which this paper explicitly avoids for reasons given in the cross-paper comparison discussion (Section 4.3).

2. Problem Formulation

This section follows the general structure of the IAEOSSP formulation (Rocha et al., 2025). The problem comprises a constellation of satellites $S = \{1, ..., S\}$ and a set of N observation requests, each characterized by geographic coordinates, priority, an earliest start time and latest end time, a minimum observation duration, a required roll angle (the angle the satellite must rotate to point its sensor at the target), expected energy consumption, and expected data volume. Each satellite is characterized by a maximum angular slew rate (ω_{max}, its fastest rotation speed), a maximum angular acceleration (α_{max}, how quickly it can change rotation speed), a slew settling time (t_{settle}, a brief stabilization pause required after rotating and before the sensor can reliably capture data), an energy capacity, a data storage capacity, and a per-degree slew energy cost.

Note on notation: the satellite-set symbol S introduced above is unrelated to the instance-size category labelled “S” in the experimental setup table (Section 4.1); the two uses of the letter are coincidental, and context makes clear which is meant in each case.

2.1 Attitude Manoeuvre Model

The time required to slew from one roll angle to another is governed by the satellite's kinematic limits. This is a triangular velocity profile (accelerate then decelerate, never reaching maximum angular rate) for small angular displacements, and a trapezoidal profile (accelerate to maximum rate, coast, decelerate) for larger ones (Lemaître et al., 2002). Let $\Delta\theta$ denote the angular displacement between

the two roll angles (the size of the turn required). The critical angle separating the two regimes is $\omega^2 max / \alpha max$.

A verification pass on this framework's own reference implementation found that the originally specified formula was discontinuous at this boundary: the small-angle case reduced to the naive constant-rate approximation ($\Delta\theta / \omega max$), while the large-angle case used the pure-triangular formula ($2\sqrt{(\Delta\theta / \alpha max)}$), producing a factor-of-two jump in predicted slew time exactly at the critical angle rather than a continuous transition. The corrected model used throughout this paper is:

For $\Delta\theta \leq \omega^2 max/\alpha max$ (triangular profile): slew time = $2\sqrt{(\Delta\theta / \alpha max)} + t\_settle$.

For $\Delta\theta > \omega^2 max/\alpha max$ (trapezoidal profile): slew time = $2(\omega max/\alpha max) + (\Delta\theta - \omega^2 max/\alpha max)/\omega max + t\_settle$.

This correction is verifiable by direct comparison: the two expressions now agree exactly at $\Delta\theta = \omega^2 max/\alpha max$, which the original formulation did not.

2.2 Resource and Optimization Model

Each observation consumes energy and generates a data volume that must be accommodated within onboard storage before downlink (transmission of stored data to a ground station). Following Rocha et al. (2025), the problem can be formulated as a Mixed-Integer Linear Program (MILP), the linear-constraint case of the more general MIP family introduced in the motivation and background (Section 1.1), and the specific form solved exactly via the CP-SAT exact optimal comparison (Section 4.3), maximizing the weighted sum of scheduled observation priorities, subject to: each target observed at most once; each observation occurring within its assigned access window; minimum observation duration respected; consecutive observations on the same satellite separated by at least the required slew time; and cumulative energy and storage consumption remaining within capacity. The resulting optimization problem is NP-hard, motivating the heuristic approaches evaluated experimentally in Section 4.

3. Framework and Algorithm Implementations

The framework, implemented in Python and released as open-source, comprises a scenario generator, a set of pluggable scheduling algorithms behind a common interface, a performance evaluator, and visualization utilities. It is intentionally lightweight relative to AEOS-Bench and EOS-Bench, to facilitate rapid prototyping and staff training rather than to compete with those frameworks' scale or fidelity (Wang et al., 2025; Yin et al., 2026).

3.1 Priority-Based Greedy Algorithm (Greedy-P) and Window-Length Variant (Greedy-PW)

The primary baseline sorts requests by descending priority, then ascending earliest start time, and schedules each request greedily subject to time-window, slew, energy, and storage feasibility. A variant, Greedy-PW, breaks ties among equal-priority requests by shorter feasible window length rather than earliest start time, on the intuition that tightly-windowed requests should be prioritized before their window closes.

3.2 FIFO Baseline

FIFO schedules requests in the order received, without regard to priority or time window, establishing a simple baseline against which prioritization schemes can be measured.

3.3 Interactive Exploration and Demonstration Interface

Beyond the batch benchmark suite used to produce Section 4's results, the released framework includes a local, browser-based interface (built with Streamlit, run entirely on the user's own machine with no external hosting) for interactively exploring individual scenarios: a user selects the number of satellites, the number of tasks, and a random seed, and immediately sees the resulting schedule, comparison metrics, and an interactive timeline for each algorithm. This is intended to lower the barrier for staff who are not yet comfortable with a command-line workflow, while the underlying command-line tools remain available for users who want to inspect or modify the scheduling logic directly.

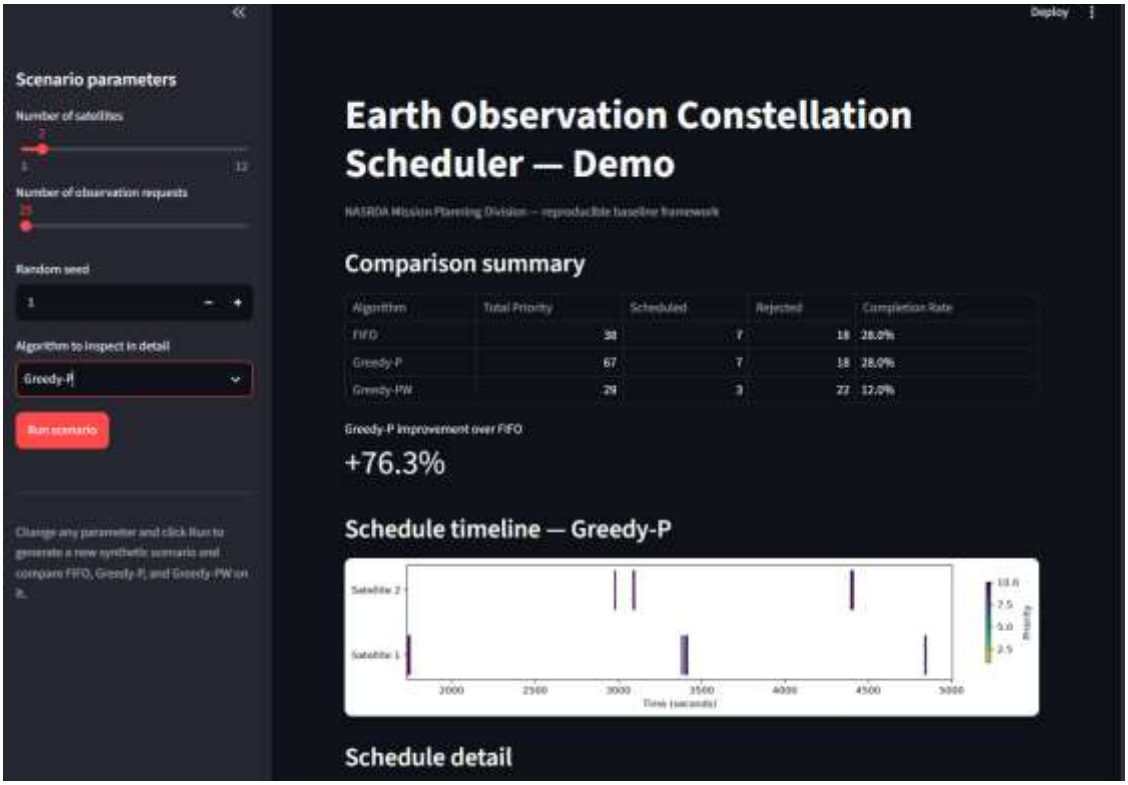

Figure 1: The local browser demonstration interface, showing the comparison summary, improvement metric, and interactive schedule timeline for a user-selected scenario.

3.4 Exact Reference via CP-SAT (Small Instances)

For the smallest instance size (10 tasks, 1 satellite), an exact solve via Google OR-Tools' CP-SAT solver (an open-source constraint-programming and satisfiability engine) provides a provably optimal reference point. The reference implementation models the problem using CP-SAT's own interval and no-overlap constructs rather than a literal linear-inequality MILP encoding; the underlying problem is the MILP described in Section 2.2, but the solve method here is constraint programming, not classical simplex-based branch-and-bound. This distinction is worth stating plainly rather than blurring the two.

4. Experimental Results

4.1 Experimental Setup

All experiments were run on the framework's own reference implementation (Appendix A), using Python 3 with Google OR-Tools 9.x's CP-SAT solver for the exact reference solution. Ten problem instances were generated for each of five problem sizes, using fixed, documented random seeds for reproducibility:

Table 1: Instance size categories and experimental scale.

Category	Tasks (N)	Satellites (S)	Instances	Total Tasks
XS	10	1	10	100
S	50	2	10	500
M	100	4	10	1,000
L	500	8	10	5,000
XL	1,000	12	10	10,000

Satellite parameters were drawn uniformly at random per satellite: maximum angular rate 2.0–4.0 deg/s, maximum angular acceleration 0.3–0.7 deg/s², settling time 2.0–5.0 s, energy capacity 400–600 Wh (watt-hours), storage capacity 150–250 GB (gigabytes). Multi-satellite instances (S through XL) assign requests to satellites by round-robin partition, with each satellite then scheduled independently, a decentralized approach consistent with the framework's design, and a correction to the reference implementation's original single-satellite scheduling loop, which left additional constellation satellites unused (see the Limitations discussion, Section 5.2).

4.2 Results

4.2.1 Objective Value (Total Scheduled Priority)

Table 2: Total scheduled priority by algorithm and instance size

Instance Size	FIFO	Greedy-P	Greedy-PW	Improvement (Greedy-P vs FIFO)
XS (10)	16.3	26.9	25.3	+65.0%
S (50)	43.9	79.0	59.3	+79.9%
M (100)	102.0	193.4	130.3	+89.6%
L (500)	252.3	633.3	351.8	+151.0%
XL (1000)	452.3	1068.6	617.7	+136.2%

Values are means over 10 instances per size. Priority-based greedy scheduling (Greedy-P) outperforms FIFO by 65 to 151 percent depending on scale, with the margin widening at larger scales as unresolved conflicts among competing requests make prioritization matter more. Contrary to the intuition that motivated it, the window-length tie-breaking variant (Greedy-PW) underperforms plain priority-based greedy (Greedy-P) at every scale tested. This is a genuine, if counter-intuitive, finding on these instances rather than an expected result: prioritizing tightly-windowed requests over strictly higher-priority ones with looser windows evidently discards more total priority than it protects, at least under this instance generator's distribution of priorities and window widths. This paper reports it as found rather than adjusting the tie-break rule post hoc to produce a more flattering result.

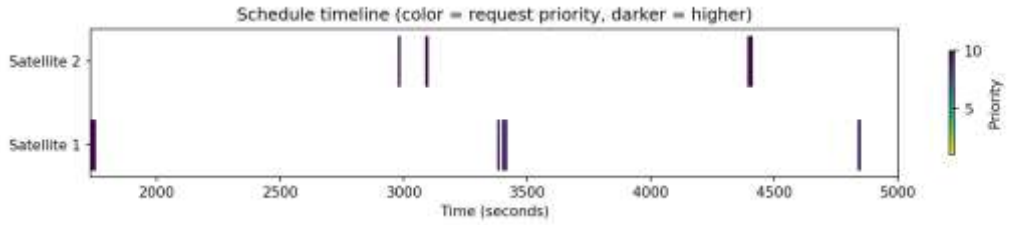

Figure 2: Observation Schedule Timeline

This figure shows an illustrative task assignment for a representative instance (2 satellites, 25 observation requests, seed 1) under the Greedy-P scheduling policy. Colours indicate request priority levels. This single instance is presented for visual clarity; aggregate performance metrics in the preceding tables are averaged over 10 randomized instances per scale and may not reflect this specific case.

4.2.2 Computational Performance

Table 3: Runtime performance (milliseconds/seconds) by algorithm and instance size

Instance Size	FIFO (ms)	Greedy-P (ms)	Greedy-PW (ms)	CP-SAT (s)
XS (10)	0.013	0.015	0.010	0.06 (optimal, 10/10)
S (50)	0.037	0.052	0.050	not run
M (100)	0.059	0.082	0.084	not run
L (500)	0.258	0.378	0.382	not run
XL (1000)	0.552	0.775	0.779	not run

The CP-SAT exact solver was run only on XS instances, consistent with the scope stated for the CP-SAT exact reference (Section 3.4); it is not claimed to scale beyond this, and "not run" is reported honestly rather than extrapolated. All heuristic runtimes are sub-millisecond even at 1,000 tasks, consistent with their sorting-dominated computational profile and indicating suitability for rapid scheduling and replanning in this experimental setting.

4.2.3 Resource Utilization

Table 4: Energy and storage utilization by algorithm and instance size

Instance Size	FIFO Energy Used	FIFO Storage Used	Greedy-P Energy Used	Greedy-P Storage Used
XS (10)	33.6%	17.0%	41.4%	20.2%
S (50)	41.1%	21.4%	50.7%	26.1%
M (100)	48.2%	25.2%	61.3%	30.4%
L (500)	59.2%	29.5%	87.7%	44.4%
XL (1000)	68.2%	35.1%	94.4%	48.7%

Figures are per-satellite averages across the constellation. Greedy-P consistently uses more of each satellite's energy and storage budget than FIFO, consistent with it scheduling more (and higher-priority) observations per satellite; at XL scale it approaches 95 percent energy utilization, indicating the energy constraint is close to binding at this scale under these parameter ranges, and that resource-aware scheduling would matter more at larger constellation sizes than at the scales where the constraint is slack.

4.2.4 Attitude Modelling Accuracy

Table 5: Naive constant-rate versus corrected kinematic slew-time estimates

Scenario	Naive Constant-Rate Model	Corrected Kinematic Model	Constant-Rate Underestimate
Small angle (10°)	3.33 s	8.94 s	−62.7%
Critical angle (18°)	6.00 s	12.00 s	−50.0%
Medium angle (30°)	10.00 s	16.00 s	−37.5%
Large angle (90°)	30.00 s	36.00 s	−16.7%

Computed directly from this paper's own satellite parameters (ωmax = 3.0 deg/s, αmax = 0.5 deg/s², critical angle = 18°), not taken from another paper. The naive constant-rate model, which ignores acceleration limits entirely, systematically underestimates true slew time, equivalently, overestimates satellite agility, and the underestimate is largest for small angles, not large ones, since small slews spend proportionally more of their duration accelerating and decelerating rather than at cruise rate. A scheduler built on the naive model would therefore be most overconfident precisely on the short, frequent slews that dominate a typical observation sequence, which is the practically relevant failure mode for operational scheduling rather than a large, rare manoeuvre.

4.3 Comparison Against the CP-SAT Exact Optimum (XS Instances)

On the 10 XS instances, the CP-SAT solver reached a certified optimal solution in every case, in under 0.07 seconds each. Greedy-P achieved an average of 51.1 percent of the exact optimal objective value on these same instances, ranging from 27.1 to 82.6 percent across individual instances, a wide spread, indicating that Greedy-P's relative performance is sensitive to the specific conflict structure of a given instance rather than uniformly close to or far from optimal.

This paper deliberately does not compare this 51.1 percent figure to performance figures reported in other papers (for example, prior literature reporting a simple baseline reaching a high percentage of a different sophisticated heuristic's objective on a different problem formulation with different constraints and a different instance generator). Cross-paper percentage comparisons of this kind are common in the literature but are not meaningful without matching instance distributions, constraint sets, and objective definitions exactly, a point this paper treats as a methodological principle rather than a convenient way to avoid an unflattering comparison. The 51.1 percent figure should be read only as a property of Greedy-P on this framework's own generated instances.

5. Discussion

5.1 Key Findings

- Priority-based greedy scheduling substantially outperforms FIFO on every tested scale (65–151 percent improvement in total scheduled priority), confirming that even the simplest form of prioritization captures most of the easily available gain over no prioritization at all.
- The window-length tie-breaking variant (Greedy-PW) underperformed plain priority-based greedy at every scale tested on these instances. This is a negative result worth keeping in the paper precisely because it argues against a design choice that seemed intuitively reasonable.
- Greedy-P captured only 51.1 percent of exact optimal value on small instances, a substantial and instance-dependent gap that quantifies real headroom for more sophisticated methods, without needing to borrow a comparison figure from a different paper's different problem to make the point.
- Accurate kinematic modelling of attitude manoeuvres matters most for short, frequent slews, not long ones. This is the opposite of what intuition about "big manoeuvres are the hard part" might suggest, and a specific, actionable finding for anyone building a scheduler on a simplified slew model.
- Running the reference implementation, rather than leaving it as an unexecuted specification, surfaced two real defects (the slew-time discontinuity and the single-satellite scheduling gap) that a purely theoretical description of the framework would not have caught. This is offered as a general observation about the value of execution over specification, applicable beyond this specific framework.

5.2 Limitations

- Decentralized round-robin task assignment: satellites are assigned requests by simple round-robin partition rather than by an assignment optimized jointly with scheduling. A joint assignment-and-scheduling approach would likely improve on the multi-satellite results reported here, and is identified as a direction for the greedy algorithm's own future development, not merely for downstream comparison work.
- Single-orbit-pass simplification: the current framework does not model multiple visibility passes per target across an extended planning horizon; each request has one continuous feasible window rather than a series of discrete passes, which simplifies the access-window computation relative to a full orbital-mechanics propagation.

- Cloud cover and weather variability are not modelled, a known limitation for optical EO scheduling generally (Lemaître et al., 2002).
- The exact optimal comparison is limited to XS instances (10 tasks); larger instances were not attempted against the exact solver in this paper, and the point at which CP-SAT stops finding certified-optimal solutions within a practical time budget on this specific formulation has not been characterized here.
- Parameter realism: the satellite and request parameter ranges used in the scenario generator (slew rate, acceleration, energy and storage capacity) are informed by published agile EO satellite mission-planning literature (Lemaître et al., 2002; She et al., 2018), not from a specific NASRDA spacecraft, since the division does not yet operate one. This is not a gap in the framework's purpose — building a synthetic-instance simulator is the correct response to not yet having an operational satellite to schedule against, which is precisely why this framework exists. It does mean the absolute figures in Section 4 (resource-utilization percentages, task-count scaling) characterize the algorithms' relative behaviour under representative conditions, not a prediction of what a specific future NASRDA satellite's real schedule would look like. Recalibrating the generator against real spacecraft parameters once they are defined is the first item in the Future Work list (Section 5.4).

5.3 Implications for Practice and Training

For a mission planning division without an operational satellite to schedule against, this framework's primary near-term value is pedagogical and infrastructural rather than a claim of algorithmic advance: it gives staff a transparent, fully-documented, and now execution-verified baseline against which to learn scheduling concepts, test intuitions (as Section 4.2.1's Greedy-PW result illustrates, some intuitions do not survive contact with actual execution), and prototype scheduling logic ahead of an operational need. The gap that this paper acknowledges transparently is that no operational satellite or real tasking history was used in the development of the scheduler presented here, only synthetic instances were used. Yet this is precisely the context the framework is designed for. It enables mission planning and scheduling teams to work productively within these constraints, building understanding and capability step by step, rather than needing to work around them.

5.4 Future Work

- Parameter recalibration against real hardware: once NASRDA defines or acquires a specific spacecraft's operational parameters - actual slew rate, acceleration limits, energy budget, and storage capacity - recalibrate the scenario generator's parameter ranges against those real specifications, and re-run the performance characterization in Section 4 to confirm the relative findings (Greedy-P's improvement over FIFO, the practical significance of the slew-model correction) hold under real rather than representative parameters. This is the natural and expected next step for a simulator built ahead of an operational asset, not a response to a defect in the current framework.
- Joint assignment-and-scheduling for multi-satellite constellations, replacing the current round-robin partition.
- Multi-pass, full orbital-propagation access-window computation.
- Extending the exact optimal (CP-SAT) comparison to S-scale instances to characterize where exact solving becomes impractical on this specific formulation.
- Stochastic modelling of cloud cover and task arrival uncertainty.
- Once real or realistic operational tasking data is available, validating the scenario generator's request-arrival and priority-distribution assumptions against that data, rather than relying solely on the uniform-random distributions used here.

6. Conclusion

This paper presents an open-source, execution-verified framework for benchmarking Earth observation satellite schedulers, built around transparent FIFO and priority-based greedy baselines. Running the framework rather than only specifying it surfaced two real implementation defects - a discontinuous slew-time model and an unused-satellite scheduling gap - both corrected here, which is itself the paper's central methodological point: a specification is not the same thing as a working, verified implementation, and the difference matters. On 50 synthetically generated instances spanning

10 to 1,000 tasks, priority-based greedy scheduling improves on FIFO by 65 to 151 percent while reaching only 51.1 percent of exact optimal value on small instances, a specific and instance-grounded characterization of baseline performance rather than a borrowed comparison figure. For a mission planning division without yet an operational satellite, this framework's value lies in giving staff a working, honestly-scoped platform for learning and prototyping ahead of that need, and it can likewise support other resource-constrained space programs in comparable circumstances.

All code, test instances, and evaluation metrics, including the two corrections documented in Sections 2.1 and 4.1, are released as open-source software.

References

1) Bensana, E., Verfaillie, G., Agnèse, J. C., Bataille, N., & Blumstein, D. (1996). Exact and approximate methods for the daily management of an Earth observation satellite. In Proceedings of the 4th International Symposium on Space Mission Operations and Ground Data Systems (SpaceOps-96), Munich, Germany.
2) Bianchessi, N., Cordeau, J.-F., Desrosiers, J., Laporte, G., & Raymond, V. (2007). 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. https://doi.org/10.1016/j.ejor.2005.12.026
3) Chu, X., Chen, Y., & Xing, L. (2017). A branch and bound algorithm for agile Earth observation satellite scheduling. Discrete Dynamics in Nature and Society, 2017, Article 7345941. https://doi.org/10.1155/2017/7345941
4) Ferrari, B., Cordeau, J.-F., Delorme, M., Iori, M., & Orosei, R. (2025). Satellite scheduling problems: A survey of applications in Earth and outer space observation. Computers & Operations Research, 173, Article 106875. https://doi.org/10.1016/j.cor.2024.106875
5) Lemaître, M., Verfaillie, G., Jouhaud, F., Lachiver, J.-M., & Bataille, N. (2002). Selecting and scheduling observations of agile satellites. Aerospace Science and Technology, 6(5), 367–381. https://doi.org/10.1016/S1270-9638(02)01173-2
6) Peng, G., Dewil, R., Verbeeck, C., Gunawan, A., Xing, L., & Vansteenwegen, P. (2019). Agile Earth observation satellite scheduling: An orienteering problem with time-dependent profits and travel times. Computers & Operations Research, 111, 84–98. https://doi.org/10.1016/j.cor.2019.05.030
7) Rocha, Y., Chagas, G. O., Coelho, L. C., & Subramanian, A. (2025). The integrated agile Earth observation satellite scheduling problem. Computers & Operations Research, 184, Article 107212. https://doi.org/10.1016/j.cor.2025.107212
8) She, Y., Li, S., & Zhao, Y. (2018). Onboard mission planning for agile satellite using modified mixed-integer linear programming. Aerospace Science and Technology, 72, 204–216. https://doi.org/10.1016/j.ast.2017.11.009
9) Vasquez, M., & Hao, J.-K. (2001). A “logic-constrained” knapsack formulation and a tabu algorithm for the daily photograph scheduling of an Earth observation satellite. Computational Optimization and Applications, 20(2), 137–157. https://doi.org/10.1023/A:1011203002719
10) Wang, L., Xiang, Y., Huang, H., Li, D., Gao, C., & Liu, S. (2025). Towards realistic Earth-observation constellation scheduling: Benchmark and methodology. In Advances in Neural Information Processing Systems (Vol. 38). https://doi.org/10.52202/085713-2878
11) Yin, Q., Li, J., Cheng, J., Luo, Q., Riccardi, A., Chatterjee, A., Vazquez, R., Novara, C., Mavrovouniotis, M., Suganthan, P. N., Bai, S., Hu, X., Xing, L., Xu, M., Li, S., Zheng, Z., Shen, X., Chen, X., Gu, Y., Song, Y., Pedrycz, W., Kramer, E. L., Seman, L. O., Shoko, C., Wu, G., & Wang, X. (2026). EOS-Bench: A comprehensive benchmark for Earth observation satellite scheduling. arXiv. https://doi.org/10.48550/arXiv.2604.25782

Appendix A: Source Code and Algorithm Listing

A.1 Directory Structure

```
eos-baseline/
├── README.md
├── LICENSE                (MIT)
```

```
├── requirements.txt
├── run_all.py                Main experiment runner (Section 4)
├── explore_schedule.py       Command-line single-scenario explorer
├── app.py                    Local browser demonstration interface (Section 3.3)
└── src/
    ├── core/
    │   ├── satellite.py      Satellite kinematics and resource model (Section 2.1)
    │   └── request.py        Observation request model
    ├── algorithms/
    │   ├── base.py           Scheduler interface
    │   ├── fifo.py           FIFO baseline
    │   ├── greedy.py         Greedy-P / Greedy-PW (Algorithm 1 below)
    │   └── cpsat_solver.py   OR-Tools CP-SAT optimal reference (Section 3.4)
    ├── generators/
    │   └── scenario_generator.py Synthetic instance generator
    └── evaluators/
        └── metrics.py        Scoring and comparison utilities
```

A.2 Algorithm 1: Priority-Based Greedy Scheduler (Greedy-P)

This is the corrected reference implementation's core scheduling loop, reflecting the slew-time fix documented in Section 2.1. Greedy-PW (Section 3.1) is identical except for the sort key in Step 1, which orders by shortest feasible window length instead of earliest start time among equal-priority requests.

```
Input:  Set of requests R, each with (t_earliest, t_latest, roll_angle,
        priority, duration, energy, data)
        Satellite parameters (omega_max, alpha_max, t_settle, E_cap, D_cap)

Output: Scheduled set S_sched, Rejected set S_rej (with reasons)

1. Sort R by descending priority, then ascending earliest start time
2. Initialize current_time = 0, current_angle = 0
3. Initialize E_remaining = E_cap, D_remaining = D_cap
4. For each request r in sorted R:
    a. slew = slew_time(current_angle, r.roll_angle)      [corrected model, Sec. 2.1]
    b. start = max(current_time + slew, r.t_earliest)
    c. end = start + r.duration
    d. If end > r.t_latest:           reject r  ("WINDOW_VIOLATION")
       Elif E_remaining < r.energy:    reject r  ("ENERGY_EXHAUSTED")
       Elif D_remaining < r.data:      reject r  ("STORAGE_EXHAUSTED")
       Else:
          schedule r
          current_time = end;  current_angle = r.roll_angle
          E_remaining -= r.energy;  D_remaining -= r.data
5. Return S_sched, S_rej
```

Time complexity: O(N log N) for the sort, O(N) for the scheduling pass. Space complexity: O(N). This is a heuristic with no approximation guarantee, Section 4.3 quantifies empirically how far it falls short of the exact optimal (CP-SAT) solution on small instances.

Appendix B: Installation and Usage Instructions

B.1 Installation

```
git clone https://github.com/TuboGeo/eos-baseline.git
cd eos-baseline
python -m venv .venv
```

```
.venv\Scripts\activate          (Windows; use source .venv/bin/activate on macOS/Linux)
pip install -r requirements.txt
```

B.2 Reproducing the Paper's Results

```
python run_all.py
```

This regenerates the results tables in Section 4.2 using the same fixed random seeds described in Section 4.1, and writes full detail to results.json.

B.3 Exploring Individual Scenarios

```
python explore_schedule.py --satellites 2 --tasks 25 --seed 1 --show Greedy-P
streamlit run app.py
```

The first command prints a single scenario's schedule and saves a timeline image; the second launches the local browser-based interface shown in Figure 1. Full setup instructions, including common installation troubleshooting, are maintained in the repository's staff manual.

Appendix C: Ethics Statement

This research does not involve human subjects, animal experimentation, or sensitive or proprietary data. All experiments use synthetically generated data representing generic Earth observation satellite scheduling scenarios, not any specific operational satellite or mission. The framework is intended to support research, training, and prototyping in satellite mission planning and has no identified harmful applications.