AstroAgentBench: Evaluating Agentic Planning on Space Mission Planning Tasks

Weiyi Wang[1,5], **Xinchi Chen**[1,3,4,5†], **Jingjing Gong**[2,5], **Xuanjing Huang**[1,4], **Xipeng Qiu**[1,2,3,4,5†]

[1]Institute of Trustworthy Embodied AI, Fudan University [2]Shanghai Innovation Institute

[3]Shanghai Key Laboratory of Multimodal Embodied AI

[4]College of Computer Science and Artificial Intelligence, Fudan University

[5]OpenMOSS Team

wywang26@m.fudan.edu.cn, xc_chen@fudan.edu.cn, jjgong@sii.edu.cn

xjhuang@fudan.edu.cn, xpqiu@fudan.edu.cn

Abstract

Recent LLM-for-Space systems address mission planning, scheduling, operations support, simulator control, and autonomy, but their evaluations use different task contracts, control settings, simulators, and success criteria. We introduce AstroAgentBench,[1] a seven-family benchmark for executable space mission planning in the domains of scheduling, observation planning, constellation design, and relay support. For each case, an agent submits a planning artifact that is checked by an external verifier for schema, timing, geometry, resources, and mission value. Results report validity and normalized scores, with comparisons to task-specific solver references. Across five LLM agent systems and 35 held-out cases, the strongest systems approach or exceed solver-reference scores on several families, while weaker systems often fail to produce high-value valid plans and even strong systems lose quality on geometric, product-level, or design-heavy tasks. Trace analyses separate two failure points: task-contract misformulation and weak solution construction. Successful runs instead calibrate agent-written implementations against verifier feedback and adapt search to case-specific structure. Ablations show that procedure injection and memory accumulation help selectively, when they supply the missing formulation, calibration, or search support.

1 Introduction

Space mission planning is both consequential and difficult. It turns mission goals into executable decisions over spacecraft, ground assets, communication links, energy, data, and time. These decisions are coupled over long horizons and constrained by physical state, operational priorities, and limited resources. A plan that sounds plausible can still fail because it violates visibility, timing, or resource constraints. This difficulty is reflected in decades of work on AI planning, operations research, spacecraft autonomy, and satellite scheduling, which have produced specialized models and solvers for particular mission settings (Johnston and Clement, 2006; Lemaître et al., 2002; He et al., 2018; Lee et al., 2024). Recent LLM-for-Space work extends the same ambition toward more flexible planning and operations support, including formation mission planning, satellite scheduling, simulated operations assistance, and orbital autonomy (Zhang et al., 2025; Chen et al., 2025a; Campanelli and Hiebl, 2025; Mousist, 2025). These efforts remain fragmented across task formulations, simulators, and success criteria. General agentic planning under executable, physically grounded mission constraints remains an open evaluation problem.

Modern LLM agents have moved beyond single-turn answer generation toward systems that reason, act, and revise through interaction. Reasoning-and-acting frameworks interleave deliberation with tool use. Embodied and web agents extend this loop to persistent environments. Software agents expose repositories, command execution, and executable code as the action surface (Yao et al., 2023; Wang et al., 2024a; Zhou et al., 2024; Yang et al., 2024; Wang et al., 2024b, 2025). Space mission planning has a similar executable surface: handoff documents, machine-readable cases, computational libraries, schedules, and checkers. This lets us ask whether an agent can translate natural-language mission goals and technical documentation into an executable planning procedure, and revise it using feedback.

We ask three questions about LLM agents in ex-

†Corresponding authors.

[1]Cases, generators, verifiers, solver references, experiment code, and run traces are publicly available at https://github.com/Mtrya/AstroAgentBench.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1561–1590

November 6-10, 2026 ©2026 Association for Computational Linguistics

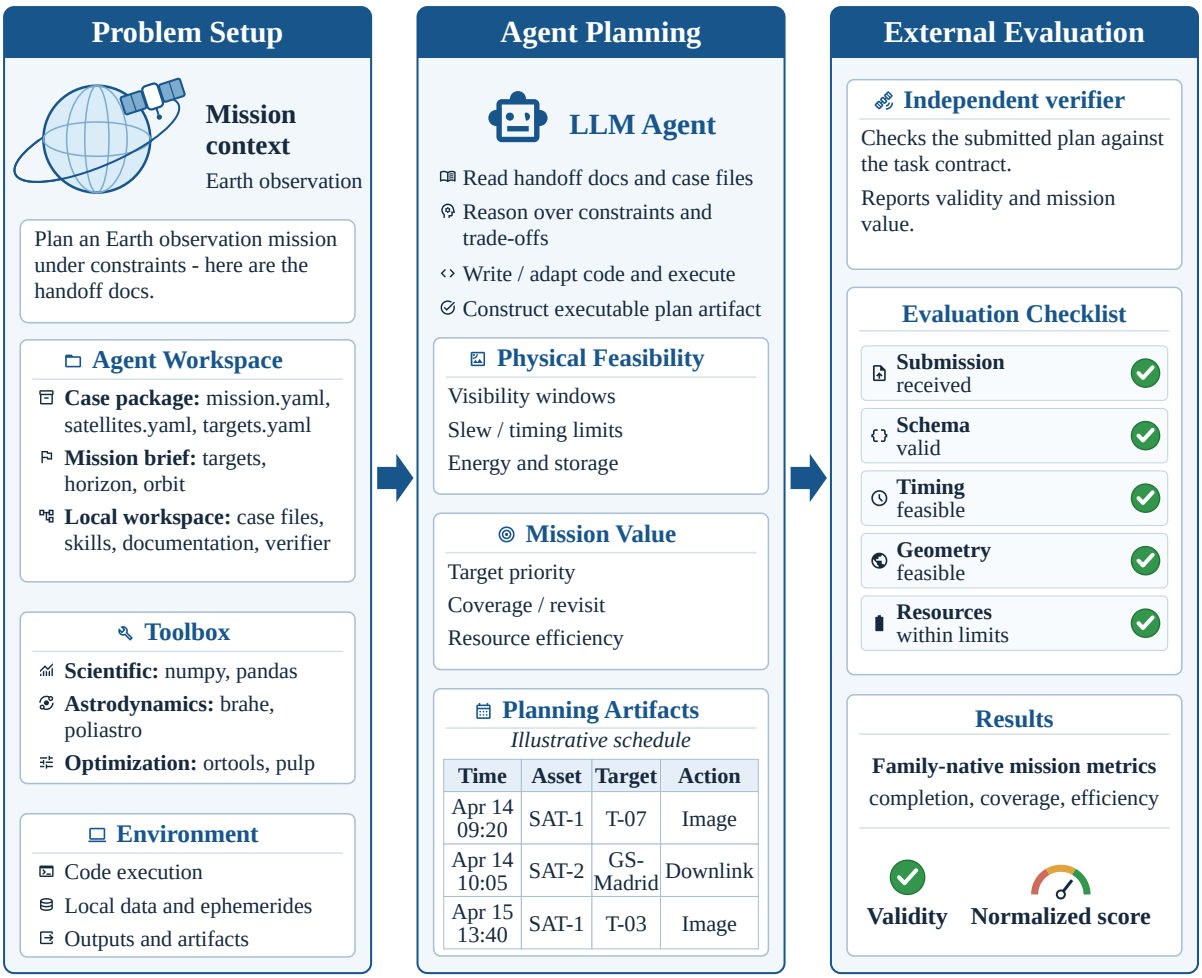


Figure 1: Overview of AstroAgentBench. A case package and mission brief define the agent workspace; the agent constructs an executable planning artifact; an independent verifier checks feasibility and scores mission value.

ecutable space mission planning. First, can they produce plans that pass external feasibility checks rather than merely describe plausible strategies? Second, when a submission is valid, how close is its quality to that of specialized solver references? Third, which system conditions explain variation in validity and scores, including verifier use, domain guidance, and accumulated experience? The distinction matters because validity alone is too weak. A plan can satisfy a schema and still observe no valuable targets, serve no demand, or cover little area. A benchmark for this setting must therefore evaluate both feasibility and graded mission value.

We introduce **AstroAgentBench**, a verifier-backed suite covering seven executable space mission-planning task families across scheduling, observation planning, constellation design, and relay support. Each case provides mission files, documentation, libraries, a solution schema, and a verifier (Figure 1). The agent must construct a planning procedure rather than call a pre-built domain planner. By combining heterogeneous task families, external feasibility checks, graded mission value, and system-level attribution, the benchmark tests agent construction of executable plans under physical constraints.

We evaluate five LLM agent systems on AstroAgentBench and compare their outputs with per-family solver references. The main pattern is uneven capability: stronger systems often produce verifier-valid plans and sometimes approach solver-reference scores, but their success is task- and system-dependent, while weaker systems often fail before producing valid executable artifacts. The same invalid or low-value outcome can arise from different parts of the run, from task interpretation to plan construction and final submission.

We further study two forms of workspace support for solution construction. Procedure injection tests whether compact domain guidance supplies

useful abstractions or brittle templates. Memory accumulation tests whether experience accumulated on training cases transfers to held-out test cases. AstroAgentBench therefore treats grounded planning as an agent-system property rather than a model-only property.

The paper makes three contributions. First, it introduces a seven-family verifier-backed benchmark for executable space mission planning, with schemas, instances, verifiers, and solver-reference comparisons. Second, it provides an empirical evaluation of contemporary LLM agent systems on this benchmark, measuring feasibility and comparing normalized scores with task-specific solver references. Third, it analyzes how trace-level failure modes, verifier use, procedure injection, and memory accumulation shape performance. AstroAgentBench gives emerging LLM-for-Space work a controlled evaluation setting grounded in physical and astrodynamic checks of feasibility and mission value.

2 Related Work

2.1 LLM-Based Agents

LLM-based agents combine language-model reasoning with tool use and environment interaction (Yao et al., 2023; Wang et al., 2024a). Coding agents extend this approach to repository navigation, code editing, and execution, with systems such as SWE-agent, CodeAct, OpenHands, and AutoCodeRover providing different interfaces and scaffolds for these activities (Yang et al., 2024; Wang et al., 2024b, 2025; Zhang et al., 2024).

2.2 Space Mission Planning and LLM-for-Space Work

Classical space mission planning is organized around specialized optimization problems. Deep Space Network allocation, agile satellite observation scheduling, stereo imaging, area coverage, constellation design, and satellite-network routing each define their own states, decision variables, physical constraints, and objective functions (Johnston and Clement, 2006; Goh et al., 2022; Lemaître et al., 2002; He et al., 2018; Zezhong et al., 2023; Lee et al., 2024; Lyu et al., 2024). This literature supplies mature solvers for individual mission-planning families, where candidate plans are meaningful only through domain-specific feasibility and value criteria.

LLM-for-Space work brings language-model systems into planning, scheduling, operations, and control workflows. Formation mission-planning work uses LLMs to decompose and organize spacecraft coordination tasks, while Earth-observation scheduling work uses multi-agent LLM systems or LLM-assisted search to design scheduling algorithms (Zhang et al., 2025; Chen et al., 2025a; Wang et al., 2026a). Satellite-operations work uses LLM agents for telemetry monitoring, procedure retrieval, anomaly explanation, and human-confirmed command execution in simulated or operational settings (Campanelli and Hiebl, 2025; Affaitati, 2024). Spacecraft-operation studies use LLMs or vision-language models as controllers in rendezvous and simulator environments, and on-orbit autonomy work explores LLM supervision of control policies under spacecraft resource constraints (Carrasco et al., 2025; Mousist, 2025). These studies span the mission lifecycle, but use different task scopes and control settings.

2.3 Agent Benchmarks

Agent benchmarks have expanded from symbolic planning tasks to interactive and code-grounded environments. PlanBench, planning-ability studies, and temporal-constraint benchmarks focus on action structure, state change, and goal satisfaction, while TravelPlanner studies itinerary construction under realistic travel constraints (Valmeekam et al., 2023b,a; Ding et al., 2025; Xie et al., 2024). WebArena places agents in stateful digital environments where success depends on navigation, tool use, and recovery from partial observations (Zhou et al., 2024). SWE-bench, SciCode, ScienceAgentBench, and BLADE require repository edits, executable programs, or scientific analyses that are checked by tests and programmatic metrics (Jimenez et al., 2024; Tian et al., 2024; Chen et al., 2025b; Gu et al., 2024). These benchmarks make agent evaluation interactive and externally checked through web-state success, software tests, scientific code tests, or analysis metrics. What remains uncommon is physically grounded planning whose submitted artifact is verifiable against a domain model.

3 AstroAgentBench

3.1 Overview

AstroAgentBench is a seven-family benchmark for executable space mission planning with LLM agents, spanning scheduling, imaging, and constel-

Task family	Horizon	Primary input entries per case
AEOSSP	12 h	20–28 satellites; 1,614–1,970 tasks
SatNet	7 d	257–333 requests; 2,513–3,370 view periods
SPOT-5	/	8–1,057 candidate photographs
Stereo	48 h	10–12 satellites; 121–144 targets
Regional	72 h	6–12 satellites; 10,492–18,325 grid cells
Revisit	48 h	24–32 targets
Relay	96 h	6–10 satellites; 4–8 demand windows

Table 1: Per-case input scale in AstroAgentBench. Entries summarize the agent-facing mission objects that define each planning instance.

lation design. A case gives the agent a mission instance, machine-readable files, documentation, libraries, and a required solution format. The submitted artifact is a schedule, assignment, geometric observation plan, constellation design, or contact plan that is checked after the run by an external verifier.

The suite contains 116 generated and normalized cases spanning train/test splits and seven task families. Agile Earth-observation scheduling, SatNet, and SPOT-5 center on selecting or assigning precomputed opportunities. Stereo Imaging and Regional Coverage require geometric observation plans. Revisit Constellation and Relay Constellation add constellation-design variables together with operating schedules. Table 1 summarizes the per-case input scale, and Appendix C gives the per-family train/test breakdown (Table 8) and the family-specific task definitions.

AstroAgentBench is related to solver-facing satellite-planning benchmarks such as SPOT-5, SatNet, AEOS-Bench, and EOS-Bench, which each focus on one planning formulation, and to broader agent benchmarks outside space mission planning. In this landscape, AstroAgentBench places seven space mission-planning families in an agent-facing setting. Appendix A gives a compact comparison.

3.2 Feasibility Constraints

AstroAgentBench checks feasibility by reconstructing the submitted plan over the mission horizon. The verifier combines submitted decisions with the case files and simulation model, then applies the validity components in Table 2. Appendix C gives the family-specific validity rules, and Appendix B gives the simulation and physical models.

The validity layer combines submission-format checks with physical reconstruction. File-presence and schema checks determine whether the verifier can interpret the decision object. Time, geometry, and resource checks then evaluate that object as a coupled mission plan: actions must be placed in admissible intervals, supported by the spatial configuration at those intervals, and compatible with shared assets and evolving capacities. Validity is therefore an aggregate property of the submitted plan, not only a local property of individual actions.

3.3 Benchmark Construction Pipeline

AstroAgentBench uses two construction routes. Five families are generated as new mission-planning cases, while SatNet and SPOT-5 are normalized from existing planning benchmarks. The generated route builds mission settings; the legacy route brings established instances into the same agent-facing setting.

Generated cases begin from grounded source material. Earth-observation, stereo-imaging, and regional-coverage cases sample satellites from fixed CelesTrak TLE snapshots, so orbital motion comes from real cataloged objects. Ground targets, city targets, regions, and relay endpoints are drawn from public city, site, or region libraries with global coverage. Missing sensor, agility, power, and link parameters are assigned from representative ranges.

Difficulty controls shape the candidate before physical filtering. For each generated family, they set the scale of assets and service objects, mission-horizon length or placement, geographic spread, geometry thresholds, resource budgets, and demand density. The sampled candidate contains the assets available to the agent, the service objects that create mission value, and the family-specific parameters that define successful service.

Physical filtering removes candidates that are physically empty, degenerate, unrealistic, or nearly automatic. Propagation, line-of-sight checks, access-window computation, coverage grids, lookup tables, and family-specific audits retain cases with feasible opportunities that still interact through time, geometry, and resources.

SatNet and SPOT-5 contribute published communication-scheduling and photography-selection instances. They are converted into the same evaluation shape as the generated families: the agent receives a planning instance, submits a family-specific artifact, and is scored by verifier-computed feasibility and value.

Component	Description
	Validity
Submission	The agent submits the required planning artifact before timeout.
Schema	The artifact parses into the family-specific decision format.
Time	Actions lie inside horizons, access or service windows, and required temporal gaps.
Geometry	The verifier recomputes visibility, pointing, range, footprint, illumination, or line of sight.
Resources	The plan respects concurrency limits, stateful consumables, and capacity or demand budgets.
	Performance
Native value	Family metrics are computed for valid artifacts, such as service, coverage, product quality, revisit gap, latency, or unmet demand.
Normalized score	Native values are mapped to a higher-is-better cross-family scale; missing or invalid submissions receive zero.

Table 2: Evaluation layers in AstroAgentBench. Validity is checked before mission value is interpreted.

3.4 Evaluation

The main evaluation uses 35 held-out test cases, five from each task family. In each system-case run, the agent receives the task prompt, brief, case files, and construction materials. The submitted planning artifact is scored after the run by an external verifier.

Evaluation follows the two-layer structure in Table 2: validity first, then performance. The validity layer asks whether the agent completed an executable handoff under the required schema and constraints. The performance layer assesses solution quality using family-native metrics and summarizes performance with normalized scores. Agent and solver-reference results use the same family-specific scoring rules. This separates whether an agent can produce a feasible mission plan from how much case-specific value the feasible plan recovers.

For example, on one AEOSSP test case, a valid schedule from Claude Code + Claude Opus 4.6 completes 1,415 of 1,947 tasks and recovers 68.54% of the total task weight. Combining these completion measures with turnaround time and energy consumption yields a score of 70.87; the system's five-case mean is 58.45. Appendix C gives the numerical calculation.

3.5 Agent-System Setup

AstroAgentBench evaluates complete agent systems: a configured model, a harness, and its workspace. Systems that share a model but differ in harness, tool access, memory, or feedback channel are treated as distinct evaluated systems.

Each run is framed by an agent-facing prompt package: a task prompt, a family-specific brief, case files, and a local validation helper. During a run, agents read mission data, choose a planning abstraction, write or adapt code, execute it, inspect feedback, and revise the produced plan.

Each run uses a controlled execution environment under a fixed budget. The protocol evaluates the submitted artifact produced within that budget, without manual intervention or post-hoc repair.

4 Experiments

We evaluate the five agent systems under a common execution budget. Each run uses the same case package, installed scientific, astrodynamics, and operations-research libraries, and a fixed two-hour wall-clock budget on 8 AMD Ryzen 7 9700X cores and 32 GB memory.

The evaluated systems are Claude Code + Claude Opus 4.6, Codex CLI + GPT-5.4, Kimi CLI + Kimi K2.6, OpenCode + MiniMax M2.7, and OpenCode + DeepSeek V4 Pro. Appendix E gives the full environment, workspace, and system-configuration details.

We additionally report a SPOT-5 evaluation of OpenCode with self-hosted Qwen3.6-27B in Appendix H.

4.1 Baselines

We compare agent systems with task-specific solver references under the same family scoring rules. Five families use implemented solvers whose outputs are checked by the benchmark verifiers; SatNet uses literature-reported results, and SPOT-5 uses archived reference solutions. For each case, we select the best and, where available, second-best reference scores, then average them within each family to obtain the solver rows in Table 3. Appendix D details the methods, reference sources, and aggregation procedure.

4.2 Main Results

Table 3 summarizes the main result matrix. Every system has at least some verifier-valid plans, and two are valid on all 35 held-out cases. Normalized scores nevertheless remain far from saturated

Method	Valid	AEOSSP	Regional	Relay	Revisit	SatNet	SPOT-5	Stereo
Claude Code + Claude Opus 4.6	26/35	58.45	21.22	12.25	**76.50**	60.36	43.96	19.16
Codex CLI + GPT-5.4	**35/35**	**72.91**	72.08	**64.91**	69.58	**69.45**	**61.92**	**68.91**
Kimi CLI + Kimi K2.6	**35/35**	72.03	**73.15**	57.77	73.36	66.14	**61.92**	0.25
OpenCode + MiniMax M2.7	23/35	36.44	17.73	0.00	0.00	37.54	37.47	0.00
OpenCode + DeepSeek V4 Pro	34/35	71.44	35.79	35.67	70.95	60.23	60.91	13.00
Best solver baseline	/	**75.95**	**86.44**	61.79	70.98	61.59	60.92	**96.05**
Second solver baseline	/	71.65	76.26	58.20	63.33	55.95	/	92.00

Table 3: Mean normalized score by method and task family. Scores are averaged over five held-out test cases per family; higher is better. The *Valid* column counts verifier-valid agent submissions out of 35 test cases. Bold values mark the best solver-or-agent score in each family and, when different, the best agent score. For solver-baseline rows, the best and second-best solver references are selected separately for each case and then averaged within each family; SPOT-5 has only one solver-reference row.

across several families, especially Regional Coverage and Stereo Imaging. Validity and mission value are therefore distinct measurements. With five held-out cases per family, these results provide an initial comparison of the evaluated systems. Appendix G reports 95% case-bootstrap confidence intervals for the mean scores.

The score gap between agents and solver references varies across task families. On AEOSSP, the strongest agent row scores 3.04 points below the best solver-reference row; on SPOT-5, Relay Constellation, Revisit Constellation, and SatNet, the strongest agent row scores above the corresponding reference row by 1.00, 3.12, 5.52, and 7.86 points, respectively. Regional Coverage and Stereo Imaging remain more separated: the best solver baselines average 86.44 and 96.05, while the strongest agent rows average 73.15 and 68.91.

Scores also vary widely within systems. A row that is strong on one family can leave large gaps on geometry-heavy or design-heavy families, and the family leaders differ across agents.

5 In-Depth Analysis

The result matrix in Section 4 shows uneven capability rather than uniform failure. We analyze failures in task formulation and solution construction, identify two mechanisms behind strong runs, and use verifier-call patterns plus workspace-support ablations to probe those mechanisms. Figure 2 previews the trace-level evidence.

5.1 Failure Modes

Trace evidence separates two failure layers: formulating the task from the available materials, and constructing a high-value plan after the task has been formulated. In **task-formulation failures**, the system may read the task document but fail to integrate the physical model, coupled constraints, scored objective, and required submitted artifact. Stereo Imaging gives the most compact example. Access-window selection is only the first step: raw observations must remain valid under hard constraints, form eligible stereo products, satisfy footprint overlap, convergence, and pixel-scale checks, and preserve final validity. The observed failures break different links in this pipeline. Kimi CLI + Kimi K2.6 produces valid observations whose footprints often do not create products. Claude Code + Claude Opus 4.6 sometimes treats rich case files as a complete task definition, builds a plausible self-imposed problem, and optimizes an inferred product-level schema and hallucinated objective. Some Stereo Imaging failures also reflect ambiguity in the documented cross-track sign convention (Appendix K), so these outcomes measure sensitivity to the supplied task specification as well as planning capability.

In **solution-construction failures**, the system has enough of the task contract to generate candidates, but cannot reliably build, improve, and preserve a high-value executable plan. The failure appears as weak search, brittle code, late verifier use, or silent goal drift during optimization. OpenCode + MiniMax M2.7 is representative. It often contacts the task document or verifier, yet still accepts valid but zero-value relay submissions, trusts agent-written simulators after verifier disagreement, or treats feasibility as the stopping condition. OpenCode + DeepSeek V4 Pro shows a related pattern on Stereo Imaging: a run may find useful product geometry, but lose the final score by failing to preserve both product value and hard-constraint validity. Table 4 shows the aggregate counterpart for selected families: Claude Code + Claude Opus 4.6 and OpenCode + MiniMax M2.7 include several no-call cases, so those runs could not use verifier feedback to correct an agent-inferred task model or

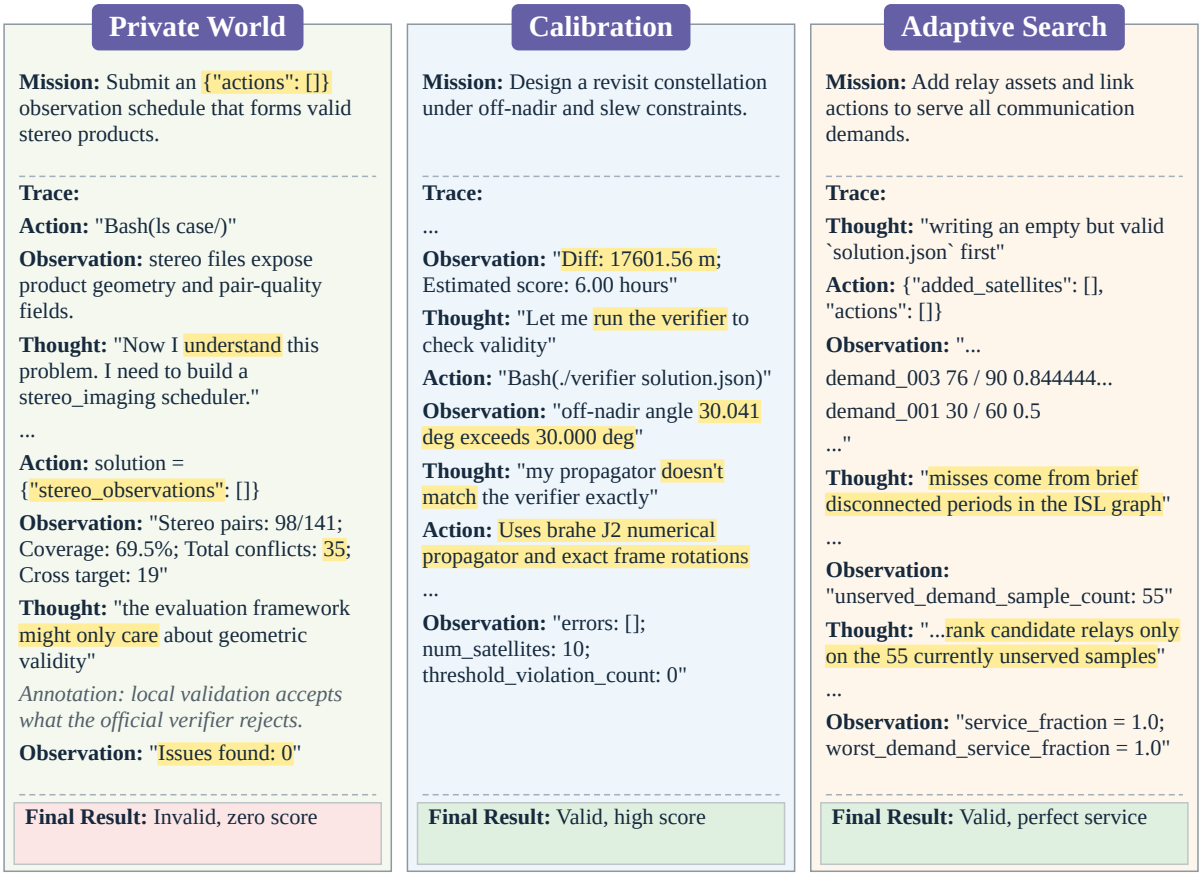


Figure 2: Trace excerpts for three mechanisms in Section 5. Left: a Stereo Imaging run builds a self-imposed task contract, writes a product-level artifact, and accepts local validation even though the official result is invalid with zero score. Middle: a Revisit Constellation run uses verifier feedback to align propagation and frame handling with off-nadir and slew constraints. Right: a Relay Constellation run preserves a valid baseline and focuses search on unserved demand samples until service is complete.

simulator during construction. Appendix L quantifies the process-level counterparts of these failures across all 175 main-experiment runs: 12.6% end with an invalid submission, 9.1% submit a valid artifact that scores zero, and 22.3% commit to a durable artifact before reading the root workspace contract.

5.2 Success Mechanisms

In the inspected high-scoring traces, agents often succeed by building a case-specific solver during the run and calibrating it against verifier feedback. The first mechanism is **implementation calibration**. Successful systems write their own propagation, geometry, scheduling, or routing code, but they use verifier disagreement to correct that code before trusting it. In Revisit Constellation, Claude Code + Claude Opus 4.6 first builds a close but invalid agent-written orbital model. Verifier errors expose off-nadir and slew-gap mismatches, and the run repairs its implementation around benchmark-compatible propagation, frame rotation, and transition timing. The complementary pattern appears in Table 4: Codex CLI + GPT-5.4, Kimi CLI + Kimi K2.6, and OpenCode + DeepSeek V4 Pro call the verifier in every shown case group, giving their agent-written solvers repeated opportunities to align with that model.

The second mechanism is **case-adaptive search with incumbent preservation**. Strong systems do not only apply a family-level recipe; they infer the local structure of the particular instance. In Revisit Constellation, the useful search changes once the required revisit floor is reached: additional observations no longer improve the primary metric, so the run turns to reducing satellite count while preserving a verified incumbent. In Relay Constellation, Codex CLI + GPT-5.4 searches for bottleneck bridge relays and keeps verified service plans while testing higher-risk link activations. In Regional Coverage memory transfer, useful prior

System	Revisit	Stereo	SatNet	Reg.	Relay
Claude	13.6(0)	20.2(2)	19.0(1)	0.0(5)	1.0(4)
Codex	5.8(0)	8.8(0)	6.0(0)	8.2(0)	6.4(0)
Kimi	9.2(0)	10.4(0)	16.0(0)	11.6(0)	8.8(0)
OC+MM	3.0(4)	18.0(3)	19.4(0)	25.8(1)	19.4(0)
OC+DS	6.0(0)	47.2(0)	10.0(0)	18.2(0)	12.4(0)

Table 4: Verifier use on selected held-out task families. Each cell gives mean verifier calls per case over five cases; parentheses give no-call cases, and shading marks cells with at least one. No-call cases indicate runs without verifier feedback for calibration. Abbreviations follow Table 3; OC denotes OpenCode, MM/DS denote MiniMax M2.7/DeepSeek V4 Pro, and Reg. denotes Regional Coverage.

experience broadens the search across regions and helps preserve the best verified candidate when later variants regress.

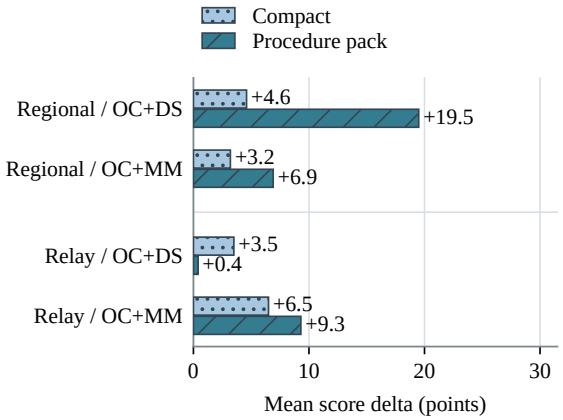
This case-adaptive behavior is consistent with the cases where agent rows score above solver-reference rows. The baselines are specialized and systematic, but their adaptation policy is fixed before the run. In the inspected traces, strong runs synthesize a local search policy during the run, using case files and verifier feedback to identify the current bottleneck and choose the next repair or search move. The same loop can fail when verified incumbents are not preserved or submitted.

The ablations below focus on whether workspace support improves the solution-construction side of this loop.

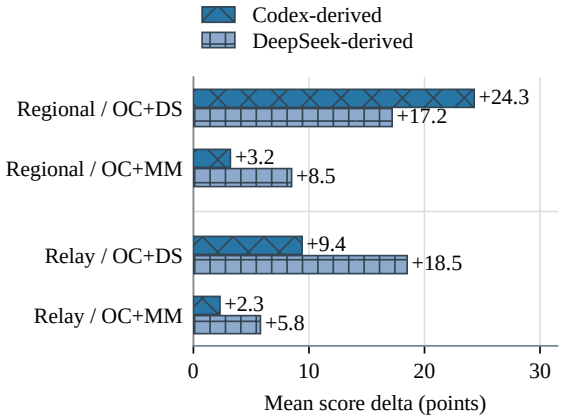
5.3 Ablations

We test two prompt/workspace supports that target solution construction for OpenCode + DeepSeek V4 Pro and OpenCode + MiniMax M2.7. We choose these two systems because they share the OpenCode harness, and these two families because they place OpenCode + DeepSeek V4 Pro in a middle band, well below the stronger systems but short of total failure, where workspace support has room to register a measurable change. **Procedure injection** adds human-written task procedures to the workspace. **Memory accumulation** adds notes distilled from prior agent runs on training cases. Figure 3 reports five-case mean normalized-score deltas on Regional Coverage and Relay Constellation. The appendix reports the underlying system-level rows.

Human-written domain procedures help most when they turn construction into local verifier-guided search. For OpenCode + DeepSeek V4 Pro, the full procedure pack gives the largest Regional



(A) Procedure injection



(B) Memory accumulation

Figure 3: Workspace-support ablations for two OpenCode systems on Regional Coverage and Relay Constellation. Bars show five-case mean normalized-score deltas relative to the system- and family-specific baseline: no procedure in Panel A and no memory in Panel B. Positive values indicate higher normalized score under the support condition.

Coverage gain, consistent with a loop that generates candidate strips, scores marginal coverage, and preserves verified incumbents. Relay Constellation is more conjunctive: placement, routing, and scheduling must align before service appears, so procedure support gives only small gains for DeepSeek. For OpenCode + MiniMax M2.7, procedures produce smaller Regional gains and larger Relay gains from a zero-score baseline. This contrast suggests that procedures can reduce empty-service collapses, but they do not by themselves supply the coordinated search needed for high relay value.

Memory notes transfer agent-derived candidate families, verifier-backed acceptance thresholds, and repair habits. The effect is strongest for

OpenCode + DeepSeek V4 Pro, especially in Regional Coverage and in Relay Constellation with DeepSeek-derived memory. OpenCode + MiniMax M2.7 also improves under memory on both families, but the gains are smaller and the resulting scores remain bounded by the system's ability to instantiate and repair the remembered plan family.

Across both supports, the main pattern is conditional transfer rather than monotone improvement. Prompt/workspace support helps when it improves search discipline, incumbent preservation, or reusable repair for the active bottleneck. It does not replace task formulation, implementation calibration, or case-specific verifier evidence.

6 Conclusion

AstroAgentBench evaluates LLM agent systems on executable space mission planning tasks whose submissions are checked for feasibility and scored by mission value. Across seven task families, the results show nontrivial planning ability and clear brittleness: the best agent rows approach or exceed solver-reference scores on several families, but weaker systems often fail to produce high-value valid plans, and even strong systems lose quality on geometric, product-level, or design-heavy tasks.

The trace and ablation results point to the same pattern. High-scoring runs calibrate agent-written implementations against verifier feedback, preserve verified incumbents, and adapt search to case-specific bottlenecks. Prompt/workspace support helps when it supplies the missing part of this loop. Current agent systems can sometimes build useful case-specific solvers, but their performance depends on keeping task formulation, physical-model calibration, search, and final handoff aligned.

Limitations

The main empirical limitation is scale. The reported evaluation covers five held-out cases for each of the seven task families, 35 cases in total, across five agent systems. Broader coverage over more cases, systems, seeds, and difficulty settings would require many more long interactive agent runs. LLM API cost is the primary constraint; wall-clock time and local compute are secondary.

The solver-reference rows are quality anchors rather than certified optima. The underlying planning families are combinatorial and include hard scheduling, coverage, routing, and design subproblems, so finding universal optima at benchmark scale is often impractical. When an agent approaches or outscores a solver row, the comparison is to the implemented reference set under the normalized scorer, not to a universal optimum.

The ablations are diagnostic rather than exhaustive. Procedure injection and memory accumulation target two plausible sources of workspace support, but they do not cover all harness designs, feedback channels, prompt structures, search tools, memory policies, or human-in-the-loop workflows. Their effects should therefore be read as evidence about the active failure layers in this setup.

The benchmark reports one sampled difficulty regime. The generators could create larger constellations, denser demand patterns, longer horizons, and higher asset counts. We do not report scaling curves across those regimes, so the results do not establish how the evaluated systems would behave as mission scale increases.

Ethical Considerations

Space mission-planning technology has dual-use potential. Better automated planning can support scientific observation, disaster response, communications, and operations analysis, but similar capabilities could also support surveillance, military planning, or strategic space operations. AstroAgentBench is designed as an evaluation benchmark rather than an operational autonomy stack: it uses compact physical models, public or representative inputs, and offline verifiers, and it does not provide procedures for commanding spacecraft or ground systems. Extensions toward full-fidelity mission simulation or direct operational interfaces would change this risk profile.

The results also argue against direct operational reliance on current LLM agents. Several systems produce plausible artifacts that fail external checks or recover little mission value. Any use of agent-generated plans in real missions would require domain-expert review, certified planning software, independent verification, and human authority over operational decisions.

The benchmark uses generated cases, public source material, and normalized public planning instances, and its curation respects the applicable licenses for public sources. The released code and data are under the MIT License, and normalized instances remain under their original distribution terms. It does not collect personal data or human-subject annotations. Future releases should main-

tain these boundaries.

To improve engineering efficiency and writing clarity, we use AI-powered tools for automated code completion and language refinement. Nevertheless, all benchmark implementations, experimental scripts, and published results undergo thorough manual verification by the authors. We also perform random audits to confirm that released materials are free of sensitive information, personally identifiable data, and harmful content.

Acknowledgments

This work is in part supported by the New Generation Artificial Intelligence-National Science and Technology Major Project (2025ZD0123502).

References

Francesco Affaitati. 2024. Automation in operations for earth bound satellites–state of the art and prospective. In *AIAA AVIATION FORUM AND ASCEND 2024*, page 4806.

Valentin Antuori, Damien T. Wojtowicz, and Emmanuel Hebrard. 2025. Solving the agile Earth observation satellite scheduling problem with CP and local search. In *31st International Conference on Principles and Practice of Constraint Programming (CP 2025)*, volume 340 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:22. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik.

Michele Campanelli and David Hiebl. 2025. Conceptual use and architecture of LLM-agents in satellite operations. In *18th International Conference on Space Operations*.

Alejandro Carrasco, Victor Rodriguez-Fernandez, and Richard Linares. 2025. Large language models as autonomous spacecraft operators in Kerbal Space Program. *Advances in Space Research*.

Jiawei Chen, Yingguo Chen, Duc Truong Pham, Yanjie Song, Jian Wu, Lining Xing, and Yingwu Chen. 2025a. A large language model-based multi-agent framework to autonomously design algorithms for Earth observation satellite scheduling problem. *Engineering*.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. 2025b. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, volume 2025, pages 96934–96990.

Thomas Claudet, Ryan Alimo, Edwin Goh, Mark D Johnston, Ramtin Madani, and Brian Wilson. 2022. Δ-MILP: Deep Space Network scheduling via mixed-integer linear programming. *IEEE Access*, 10:41330–41340.

Zifeng Ding, Sikuan Yan, Moy Yuan, Xianglong Hu, Fangru Lin, and Andreas Vlachos. 2025. TCP: a benchmark for temporal constraint-based planning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 22463–22486.

Duncan Eddy and Mykel J. Kochenderfer. 2021. A maximum independent set method for scheduling Earth-observing satellite constellations. *Journal of Spacecraft and Rockets*, 58(5):1416–1429.

Jason Gerard, Juan A. Fraire, and Sandra Céspedes. 2026. Contact plan design for optical interplanetary communications. *Ad Hoc Networks*, 194:104393.

Edwin Goh, Hamsa Shwetha Venkataram, Bharathan Balaji, Brian D Wilson, and Mark D Johnston. 2022. SatNet: A benchmark for satellite scheduling optimization. In *AAAI-22 Workshop on Machine Learning for Operations Research (ML4OR)*.

Paul Grislain, Nicolas Pelissier, François Lamothe, Oana Hotescu, Jérôme Lacan, Emmanuel Lochin, and José Radzik. 2022. Rethinking LEO constellations routing with the unsplittable multi-commodity flows problem. In *2022 11th Advanced Satellite Multimedia Systems Conference and 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, pages 1–8. IEEE.

Ken Gu, Ruoxi Shang, Ruien Jiang, Keying Kuang, Richard-John Lin, Donghe Lyu, Yue Mao, Youran Pan, Teng Wu, Jiaqian Yu, Yikun Zhang, Tianmai M Zhang, Lanyi Zhu, Mike A Merrill, Jeffrey Heer, and Tim Althoff. 2024. BLADE: Benchmarking language model agents for data-driven science. In *Findings of the association for computational linguistics: EMNLP 2024*, pages 13936–13971.

Lei He, Xiaolu Liu, Gilbert Laporte, Yingwu Chen, and Yingguo Chen. 2018. An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. *Computers & Operations Research*, 100:12–25.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, volume 2024, pages 54107–54157.

Mark D Johnston and Bradley J Clement. 2006. Automating deep space network scheduling and conflict resolution. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1483–1489.

Samir Khuller, Anna Moss, and Joseph Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45.

Junhong Kim, Jaemyung Ahn, Han-Lim Choi, and Doo-Hyun Cho. 2020. Task scheduling of agile satellites with transition time and stereoscopic imaging constraints. *Journal of Aerospace Information Systems*, 17(6):285–293.

François Lamothe, Emmanuel Rachelson, Alain Haït, Cédric Baudoin, and Jean-Baptiste Dupé. 2023. Dynamic unsplittable flows with path-change penalties: New formulations and solution schemes for large instances. *Computers & Operations Research*, 152:106154.

Hang Woon Lee, Seiichi Shimizu, Shoji Yoshikawa, and Koki Ho. 2020. Satellite constellation pattern optimization for complex regional coverage. *Journal of Spacecraft and Rockets*, 57(6):1309–1327.

Soung Sub Lee, Jong Pil Kim, Eungnoh You, Jae-Hyuk Youn, and Ho-Hyun Shin. 2024. Satellite constellation method to achieve desired revisit performance for multiple targets. *Journal of Applied Remote Sensing*, 18(2):024509–024509.

Michel Lemaître, Gérard Verfaillie, Frank Jouhaud, Jean-Michel Lachiver, and Nicolas Bataille. 2002. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6(5):367–381.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429.

Yifeng Lyu, Han Hu, Rongfei Fan, Zhi Liu, Jianping An, and Shiwen Mao. 2024. Dynamic routing for integrated satellite-terrestrial networks: A constrained multi-agent reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*, 42(5):1204–1218.

Antonio M Mercado-Martínez, Beatriz Soret, and Antonio Jurado-Navas. 2025. Scheduling agile earth observation satellites with onboard processing and real-time monitoring. In *GLOBECOM 2025-2025 IEEE Global Communications Conference*, pages 2023–2029. IEEE.

Alejandro D Mousist. 2025. ASTREA: Introducing agentic intelligence for orbital thermal autonomy. *arXiv preprint arXiv:2509.13380*.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294.

Akashah Shabbir, Muhammad Akhtar Munir, Akshay Dudhane, Muhammad Umer Sheikh, Muhammad Haris Khan, Paolo Fraccaro, Juan Bernabe Moreno, Fahad Shahbaz Khan, and Salman Khan. 2025. ThinkGeo: Evaluating tool-augmented agents for remote sensing tasks. *arXiv preprint arXiv:2505.23752*.

Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, Shengyan Liu, Di Luo, Yutao Ma, Hao Tong, Kha Trinh, Chenyu Tian, Zihan Wang, Bohao Wu, Yanyu Xiong, and 11 others. 2024. SciCode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 37:30624–30650.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023a. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36:38975–38987.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023b. On the planning abilities of large language models-a critical investigation. *Advances in neural information processing systems*, 36:75993–76005.

Feiran Wang, Jiawei Chen, Yonghao Du, Yanjie Song, Yingwu Chen, Rammohan Mallipeddi, and Witold Pedrycz. 2026a. LLM-assisted adaptive large neighborhood search for agile earth observation satellite scheduling. *Engineering Management*, 13(1):213–239.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Luting Wang, Yinghao Xiang, Hongliang Huang, Dongjun Li, Chen Gao, and Si Liu. 2026b. Towards realistic earth-observation constellation scheduling: Benchmark and methodology. *Advances in Neural Information Processing Systems*, 38:85923–85944.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024b. Executable code actions elicit better LLM agents. In *Forty-first International Conference on Machine Learning*.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, and 5 others. 2025. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pages 65882–65919.

David O Williams Rogers, Dongshik Won, Dongwook Koh, Kyungwoo Hong, and Hang Woon Lee. 2026. Optimal satellite constellation configuration design: A collection of mixed integer linear programs. *Journal of Spacecraft and Rockets*, pages 1–18.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. TravelPlanner: A benchmark for real-world planning with language agents. In *International Conference on Machine Learning*, pages 54590–54613. PMLR.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. SWE-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Qian Yin, Jiaxing Li, Jiaqi Cheng, Qizhang Luo, Annalisa Riccardi, Abhijit Chatterjee, Rafael Vazquez, Carlo Novara, Michalis Mavrovouniotis, Ponnuthurai Nagaratnam Suganthan, Shengzhou Bai, Xiaoxuan Hu, Lining Xing, Ming Xu, Shuang Li, Zixuan Zheng, Xin Shen, Xiaoyu Chen, Yi Gu, and 7 others. 2026. EOS-Bench: A comprehensive benchmark for Earth observation satellite scheduling. *arXiv preprint arXiv:2604.25782*.

LU Zezhong, Xin Shen, LI Deren, Dilong Li, Yaxin Chen, Di Wang, and Shuai Shen. 2023. Multiple super-agile satellite collaborative mission planning for area target imaging. *International Journal of Applied Earth Observation and Geoinformation*, 117:103211.

Chen Zhang, Jin Jin, Linling Kuang, and Jian Yan. 2018. LEO constellation design methodology for observing multi-targets. *Astrodynamics*, 2(2):121–131.

Yuhang Zhang, Bohan Jiao, and Zhaohui Dang. 2025. A large language model-based approach to spacecraft formation mission planning. *IFAC-PapersOnLine*, 59(20):2166–2170.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. 2024. AutoCodeRover: Autonomous program improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1592–1604.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pages 15585–15606.

A Benchmark Landscape

Table 5 compares AstroAgentBench with related benchmarks by setting, task-family breadth, and verification mechanism. The neighboring benchmarks cover symbolic or temporal planning, single-family satellite scheduling, and remote-sensing tool-use tasks.

Benchmark	Setting	Task families	Verification
SPOT-5 / ROADEF (Lemaître et al., 2002)	Agile-satellite photograph selection	One photography-selection family	Combinatorial constraints
SatNet (Goh et al., 2022)	Ground-station contact assignment	One communication-scheduling family	Contact windows, antenna occupancy, demand metrics
AEOS-Bench (Wang et al., 2026b)	Realistic Earth-observation constellation scheduling	One scheduling family at large scale	Orbital dynamics, resources, scheduling metrics
EOS-Bench (Yin et al., 2026)	Earth-observation satellite scheduling	One scheduling family with scale regimes	Orbital dynamics, platform constraints, scheduling metrics
Planning benchmarks (Valmeekam et al., 2023b,a; Ding et al., 2025)	Symbolic action planning and temporal constraints	Multiple symbolic domains	Formal action and temporal-validity checks
Remote-sensing agent benchmarks (Shabbir et al., 2025)	Tool-augmented remote-sensing analysis	Multiple remote-sensing tasks	Tool outputs and answer checks
AstroAgentBench	Executable space mission planning	Seven mission-planning families	Schema, timing, geometry, resources, mission-value scoring

Table 5: Benchmark-landscape comparison. Blue rows are agent-facing; yellow rows are solver-facing. Agent-facing means the benchmark presents tasks through prompts or a prepared workspace; verification summarizes how submitted outputs are checked.

B Physical and Astrodynamics Models

This appendix records the physical and astrodynamics assumptions used by each task family. The benchmark does not impose a single simulator across families: some tasks use precomputed windows or abstract conflict variables, while others evaluate orbit propagation, frame conversion, access geometry, pointing limits, resource accounting, and service scoring.

For families that compute geometry from orbital state, feasibility is evaluated in a fixed physical order: propagate spacecraft state, transform inertial states into an Earth-fixed frame, then compute visibility, pointing, surface intersection, or link geometry.

Three modeling regimes recur across Table 6. SatNet and SPOT-5 are abstraction-heavy: their physical history enters the task through view periods, maintenance windows, forbidden tuples, camera domains, and capacity fields. AEOSSP, Stereo Imaging, and Regional Coverage keep the satellite set fixed and compute observation geometry from submitted actions. Revisit and Relay expose architecture variables: a solution supplies satellite states, and the model propagates those states before evaluating target visibility or communication service.

Propagation and Frames. AEOSSP, Stereo Imaging, and Regional Coverage use two-line elements (TLEs): compact mean-element records for cataloged Earth satellites. Their satellite states are propagated with SGP4, the standard TLE propagator, which natively returns position and velocity in the TEME frame; the propagation libraries convert TEME states into the GCRF inertial and ITRF Earth-fixed frames before any geometry is computed. Revisit and Relay instead propagate submitted or added GCRF states with deterministic J2 dynamics, which model central-body gravity plus the dominant oblateness perturbation. For AEOSSP, Regional Coverage, Revisit, and Relay, the resulting GCRF state is transformed into ITRF/ECEF before it is compared with Earth-fixed targets, grid samples, ground stations, or relay endpoints; Stereo Imaging queries Earth-fixed states from its propagator directly.

Access and Pointing. For satellite i, ground object j, and time t, let $\epsilon_{ij}(t)$ be the elevation angle of the satellite above the local horizon at j, $R_{ij}(t)$ the slant range, and $\theta_{ij}(t)$ the payload pointing angle

Aspect	AEOSSP	SatNet	SPOT-5	Stereo	Regional	Revisit	Relay
State and reference geometry							
Propagation	SGP4 (TEME)	precomputed	abstracted	SGP4 (TEME)	SGP4 (TEME)	J2	J2
Frame conversion	GCRF–ITRF	/	/	GCRF–ITRF	GCRF–ITRF	GCRF–ITRF	GCRF–ITRF
Surface model	WGS84	/	/	WGS84	WGS84	spherical R_E	spherical R_E
Geometry and action feasibility							
Pointing variable	off-nadir	/	/	along/across	roll	off-nadir	/
Transition check	slew; settle	setup; teardown	tuple conflicts	slew; settle	roll slew	slew; settle	/
Resources							
Energy resource	battery	/	/	/	battery	battery	/
Capacity resource	/	antenna occupancy	recorder cap	/	duty limit	/	link capacity

Table 6: Physical and astrodynamics model ingredients by family. A slash marks an inactive layer because the family uses a higher-level abstraction for that part of the physical problem. SGP4 natively produces TEME states, which the propagation wrappers convert into GCRF and ITRF. Revisit and Relay convert geodetic targets or endpoints into Earth-fixed coordinates; the surface-model row names the Earth body used for altitude and blockage checks.

away from the allowed boresight or nadir direction. Let $\epsilon_j^{\min}$ be the required minimum elevation at the target or ground endpoint, $R_j^{\max}$ the maximum usable range when a range cap is modeled, and $\theta_i^{\max}$ the payload's maximum pointing angle. A target observation is feasible only if the line of sight satisfies

$$\epsilon_{ij}(t) \geq \epsilon_j^{\min}, \qquad R_{ij}(t) \leq R_j^{\max}, \quad \theta_{ij}(t) \leq \theta_i^{\max}.$$

Stereo imaging adds product-level constraints on overlap, convergence angle, temporal separation, and pixel-scale ratio. Regional coverage derives strip footprints from roll angle and sensor field of view, then scores weighted sample coverage.

Slew and Settling. Same-satellite action sequences must leave enough time for retargeting. For a scalar angular change $\Delta\theta$, maximum angular rate $\omega_{\max}$, maximum angular acceleration $\alpha_{\max}$, settling time t_s, and threshold $\theta_c = \omega_{\max}^2/\alpha_{\max}$, use the first branch below when $\Delta\theta \leq \theta_c$ and the second otherwise:

$$T_{\text{slew}}(\Delta\theta) = \begin{cases} 2\sqrt{\Delta\theta/\alpha_{\max}}, \\ \Delta\theta/\omega_{\max} + \omega_{\max}/\alpha_{\max}. \end{cases}$$

Consecutive actions a_k, a_{k+1} on the same satellite require

$$s_{k+1} - e_k \geq T_{\text{slew}}(\Delta\theta_{k,k+1}) + t_s.$$

Energy. Battery-constrained families integrate state of charge over model time segments. Let $E_i(t)$ be the battery state of charge, $E_i^{\max}$ its capacity, Δt a segment duration in seconds, and $P_i^{\text{chg}}(t)$ and $P_i^{\text{load}}(t)$ the charge and load powers. A generic update is

$$E_i(t + \Delta t) = \min(E_i^{\max},\ E_i(t) + \Delta E_i(t)),$$
$$\Delta E_i(t) = \frac{(P_i^{\text{chg}}(t) - P_i^{\text{load}}(t))\Delta t}{3600}.$$

Hard feasibility requires

$$E_i(t) \geq 0 \qquad \forall i, t.$$

Load terms depend on the family and may include idle, imaging, and slew or maneuver power.

Communication and Routing. SatNet separates antenna occupation from useful communication time. Setup and teardown consume the occupied interval O_k, while only $[s_k, e_k]$ contributes service. Relay constellation separates submitted link activations from routed service. The service model builds $G_t(S)$, allocates feasible paths, and computes latency as

$$\ell_{d,t} = 1000\, L_{d,t}/c,$$

where $L_{d,t}$ is the selected path length and c is the speed of light.

C Task Contracts and Metrics

Let I denote a benchmark instance, S a submitted solution, and f a task family. The family validity indicator is

$$V_f(I, S) \in \{0, 1\}.$$

Here $V_f(I, S) = 1$ means that the submitted solution satisfies the hard constraints for family f. Invalid or missing submissions receive score zero in aggregate tables. For any scalar x, let

Family	Submitted solution	Hard-validity layers	Native metrics
AEOSSP	point-observation actions	task window; required duration; sensor type; visibility; off-nadir; same-satellite overlap; slew; battery	completion; turnaround; energy
SatNet	antenna-track rows	view-period containment; setup; teardown; antenna occupancy; maintenance; request/resource match; minimum duration	unsatisfied demand; satisfied requests; tracking hours
SPOT-5	one camera-mode assignment per photograph	domain membership; binary conflicts; ternary conflicts; multi-orbit memory cap	profit; memory use
Stereo Imaging	raw observation actions	timing; access interval; solar elevation; off-nadir; overlap; slew; stereo-product geometry	target coverage; product quality
Regional Coverage	roll-only strip actions	time grid; duration; sensor band; strip intersection; overlap; roll slew; battery; duty limit; minimum regional coverage	weighted coverage; actions; battery
Revisit Constellation	initial satellite states; observation actions	satellite cap; orbit bounds; visibility; range; off-nadir; timing; overlap; slew; battery	revisit gap; satellite count
Relay Constellation	added satellite states; link activations	orbit bounds; ground-link geometry; inter-satellite geometry; overlap; endpoint and satellite link caps; Earth blockage	service; latency; added satellites

Table 7: Task constraints and metrics inventory. Each row names what a valid submitted solution can contain, which hard-validity layers are applied, and which native metrics are reported for valid submissions.

Family	Total	Train	Test
AEOSSP	30	10	5
SatNet	5	0	5
SPOT-5	21	10	5
Stereo	15	10	5
Regional	15	10	5
Revisit	15	10	5
Relay	15	10	5
All	116	60	35

Table 8: Per-family case counts. Test is the held-out split used by the main evaluation and all ablations. AEOSSP additionally provides 15 auxiliary-split cases (test_easy, test_hard, test_horizon_2022); SatNet contributes five published instances, all held out; SPOT-5's train and test lists overlap on two instances by design, and eight further published instances are in neither split.

$[x]_0^1 = \min(1, \max(0, x))$. The higher-is-better score reported in cross-family tables is

$$Q_f(I,S) = \begin{cases} N_f(I,S), & V_f(I,S) = 1, \\ 0, & \text{otherwise}, \end{cases}$$

where N_f maps the family-native metric to the normalized scale defined below. Native metrics remain family-specific; Q_f is only the common reporting score.

Table 7 lists each family's submitted object, hard-validity layers, and native metrics. The derived quantities below are computed from submitted primitive actions, states, assignments, or link intervals. Table 8 gives the per-family case counts behind the 116-case suite and the 35-case held-out evaluation.

AEOSSP. Let $\mathcal{J}$ be the task set. Task $j \in \mathcal{J}$ has release time r_j, deadline d_j, required duration δ_j, required sensor type σ_j, and weight w_j. Satellite i has sensor type σ_i. A submitted point-observation action is

$$a_k = (i_k, j_k, s_k, e_k),$$

where satellite i_k observes task j_k from start time s_k to end time e_k. The schedule must satisfy

$$r_{j_k} \le s_k < e_k \le d_{j_k},$$
$$e_k - s_k = \delta_{j_k}, \qquad \sigma_{i_k} = \sigma_{j_k},$$

as well as continuous visibility, off-nadir, same-satellite non-overlap, slew-plus-settle, and battery constraints. Let $C(S) \subseteq \mathcal{J}$ be the tasks completed by at least one valid action, and let t_j^{done} be the earliest completion time for completed task j. The completed-weight fraction WCR and completed-task fraction CR are

$$\text{WCR}(S) = \frac{\sum_{j \in C(S)} w_j}{\sum_{j \in \mathcal{J}} w_j},$$
$$\text{CR}(S) = \frac{|C(S)|}{|\mathcal{J}|}.$$

Turnaround time TAT is the mean $t_j^{\text{done}} - r_j$ over $j \in C(S)$, and power consumption PC is gross watt-hour use over the horizon. With mission horizon H and case energy budget E_{bud}, the reporting score is

$$N_{\text{AEOSSP}} = 100(0.45\,\text{WCR} + 0.20\,\text{CR} + 0.20[1 - \text{TAT}/H]_0^1 + 0.15[1 - \text{PC}/E_{\text{bud}}]_0^1).$$

Here E_{bud} is the sum of satellite battery capacities, used as an energy normalization constant. If no task is completed, TAT is undefined and the reporting score is zero.

Worked example. On AEOSSP test case 1, the verifier accepts the schedule from Claude Code + Claude Opus 4.6: it completes 1,415 of 1,947 tasks, with completed weight 3,669 out of 5,353. The verifier reports TAT = 1071.23 s and PC =

18621.35 Wh. With $H = 43200$ s and $E_{\mathrm{bud}} = 31000$ Wh, substitution gives

$$\begin{aligned} N_{\mathrm{AEOSSP}} \approx 100\Big(&0.45\,\frac{3669}{5353} \\ &+ 0.20\,\frac{1415}{1947} \\ &+ 0.20\Big(1 - \frac{1071.23}{43200}\Big) \\ &+ 0.15\Big(1 - \frac{18621.35}{31000}\Big)\Big) \\ \approx\ & 70.87. \end{aligned}$$

The five case scores are 70.87, 74.45, 73.89, 73.05, and 0.00, whose arithmetic mean is 58.45 at the reported precision. The fifth submission is parsed as an empty schedule and receives zero.

SatNet. Let $\mathcal{J}$ be the request set. Request j belongs to a subject group $g(j)$, requests communication duration D_j, and defines setup and teardown times α_j and β_j. A submitted track is

$$a_k = (j_k, C_k, s_k, e_k),$$

where request j_k uses antenna or antenna-array resource C_k during transmission interval $[s_k, e_k]$. The occupied antenna interval includes setup and teardown:

$$O_k = [s_k - \alpha_{j_k},\, e_k + \beta_{j_k}).$$

The transmission interval must lie inside a valid view period, all occupied intervals sharing an antenna must be disjoint, occupied intervals must avoid maintenance, the resource must be compatible with the request, and $e_k - s_k$ must meet the per-track minimum duration rule. Let A_j be the allocated communication time for request j, capped at D_j. Let $\mathcal{G}$ be the subject groups and $\mathcal{J}_g = \{j \in \mathcal{J} : g(j) = g\}$. The unsatisfied fraction for group g is

$$u_g = \frac{\max\left(\sum_{j\in\mathcal{J}_g} D_j - \sum_{j\in\mathcal{J}_g} A_j, 0\right)}{\sum_{j\in\mathcal{J}_g} D_j}.$$

SatNet reports

$$U_{\mathrm{rms}} = \left(\frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}} u_g^2\right)^{1/2}, \qquad U_{\max} = \max_{g\in\mathcal{G}} u_g,$$

with lower values preferred. The reporting score is

$$\begin{aligned} N_{\mathrm{SatNet}} = 100(&0.75[1 - U_{\mathrm{rms}}]_0^1 \\ &+ 0.25[1 - U_{\max}]_0^1). \end{aligned}$$

SPOT-5. Let $\mathcal{P}$ be the candidate photograph set. Photograph p has profit v_p and allowed camera-mode domain D_p, which contains only nonzero camera modes; submitting value 0, equivalently the all-zero assignment, denotes not selecting the photograph. Binary variable

$$x_{p,m} \in \{0, 1\}, \qquad p \in \mathcal{P},\ m \in D_p,$$

encodes whether p is selected with mode m. A valid assignment chooses at most one nonzero mode for each photograph:

$$\sum_{m\in D_p} x_{p,m} \le 1$$

and must satisfy every listed binary or ternary conflict. For a forbidden tuple of arity r over photographs $p_1, \ldots, p_r$, with forbidden modes $m_1, \ldots, m_r$,

$$\sum_{\ell=1}^{r} x_{p_\ell, m_\ell} \le r - 1.$$

In multi-orbit instances, $w_{p,m}$ is the memory consumption of selecting photograph p with mode m, divided by a fixed normalizing constant and rounded, and memory use is capped by

$$\sum_{p\in\mathcal{P}} \sum_{m\in D_p} w_{p,m} x_{p,m} \le 200.$$

The objective metric is selected profit,

$$P(S) = \sum_{p\in\mathcal{P}} v_p \sum_{m\in D_p} x_{p,m}.$$

If P_{all} is the unconstrained sum of available photograph profits, the reporting score is

$$N_{\mathrm{SPOT5}} = 100\left[\frac{P(S)}{0.5P_{\mathrm{all}}}\right]_0^1.$$

Stereo Imaging. Let $\mathcal{J}$ be the target set. A submitted raw observation is

$$a_k = (i_k, j_k, s_k, e_k, \alpha_k, \beta_k),$$

where satellite i_k observes target j_k during $[s_k, e_k]$, and α_k and β_k are along-track and cross-track steering angles. Let $t_k = (s_k + e_k)/2$ be the midpoint time. Action-level validity checks timing, access interval, solar elevation, off-nadir pointing, same-satellite overlap, and slew. A pair (a_p, a_q) can

become a stereo product only when the two observations share a target and also satisfy the following product-level constraints. Here o_{pq} is overlap fraction, γ_{pq} is convergence angle, η_p and η_q are effective pixel scales, and the subscripted constants are case parameters:

$$\begin{aligned} |t_p - t_q| &\le \Delta_{\max}, \\ o_{pq} &\ge o_{\min}, \\ \gamma_{\min} \le \gamma_{pq} &\le \gamma_{\max}, \\ \frac{\max(\eta_p, \eta_q)}{\min(\eta_p, \eta_q)} &\le \eta_{\max}. \end{aligned}$$

Tri-stereo products additionally require three compatible observations and a near-nadir anchor. Let $\mathcal{P}_j(S)$ be the valid stereo or tri-stereo products covering target j, and let $Q(p)$ be product quality for product p. Per-target quality is

$$Q_j(S) = \begin{cases} \max_{p \in \mathcal{P}_j(S)} Q(p), & \mathcal{P}_j(S) \neq \emptyset, \\ 0, & \mathcal{P}_j(S) = \emptyset, \end{cases}$$

The native quality $\bar{Q}$ is the mean of $Q_j(S)$ over targets, and coverage is the fraction of targets with $Q_j(S) > 0$. The reporting score is $N_{\text{Stereo}} = 100[\bar{Q}]_0^1$.

Regional Coverage. A submitted strip action is

$$a_k = (i_k, s_k, \Delta_k, \rho_k),$$

where satellite i_k images for duration Δ_k starting at s_k, and ρ_k is the signed roll angle. Validity checks time-grid alignment, duration and roll-band bounds, strip-Earth intersection, same-satellite overlap, roll slew, battery, duty limits, and optional minimum regional coverage. Let $\mathcal{P}_{\text{cov}}$ be the weighted coverage-sample set, w_p the weight of sample p, and $c_p(S)$ the number of valid strips covering p. Unique weighted coverage is

$$\text{Cov}_w(S) = \frac{\sum_{p \in \mathcal{P}_{\text{cov}}} w_p \, \mathbf{1}\{c_p(S) \ge 1\}}{\sum_{p \in \mathcal{P}_{\text{cov}}} w_p}.$$

Let $\mathcal{R}$ be the configured region set. For region $r \in \mathcal{R}$, let $\mathcal{P}_r \subseteq \mathcal{P}_{\text{cov}}$ be its samples. The per-region covered fraction is

$$\text{Cov}_r(S) = \frac{\sum_{p \in \mathcal{P}_r} w_p \, \mathbf{1}\{c_p(S) \ge 1\}}{\sum_{p \in \mathcal{P}_r} w_p},$$

and the reported coverage ratio averages regions with configured weights λ_r:

$$\text{Cov}(S) = \frac{\sum_{r \in \mathcal{R}} \lambda_r \text{Cov}_r(S)}{\sum_{r \in \mathcal{R}} \lambda_r}.$$

Let $A(S)$ be the action count, $A_{\max}$ the action cap, $E_{\min}$ the minimum remaining battery, and $E_{\max}$ the representative battery capacity. The reporting score is

$$\begin{aligned} N_{\text{Regional}} = 100(&0.50\text{Cov}_w + 0.20\text{Cov} \\ &+ 0.15[1 - A/A_{\max}]_0^1 \\ &+ 0.15[E_{\min}/E_{\max}]_0^1). \end{aligned}$$

Revisit Constellation. Let $\mathcal{J}$ be the target set and let the mission run from t_0 to t_1. Target j has expected revisit period g_j^{req}. A solution contains n satellite initial states,

$$X = \{(\mathbf{r}_i(t_0), \mathbf{v}_i(t_0))\}_{i=1}^n$$

and observation actions. The validity checks include satellite-count caps, orbit bounds, visibility, slant range, off-nadir pointing, timing, overlap, slew, and battery. Let $T_j(S) = (\tau_{j,0}, \ldots, \tau_{j,L_j})$ be the ordered sequence containing t_0, the successful observation midpoint times for target j, and t_1. The maximum revisit gap for target j is

$$g_j(S) = \max_{0 \le \ell < L_j} (\tau_{j,\ell+1} - \tau_{j,\ell}).$$

The reported primary metric is the mean requirement-floored gap,

$$G(S) = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \max(g_j(S), \, g_j^{\text{req}}),$$

with lower values preferred, followed by satellite count. Let $H = t_1 - t_0$. For a scalar gap g and requirement g_{req}, the target-level reporting curve is

$$\phi_{\text{gap}}(g) = \begin{cases} 1, & g \le g_{\text{req}}, \\ 0, & g \ge H, \\ \left(\frac{H-g}{H-g_{\text{req}}}\right)^2, & \text{otherwise.} \end{cases}$$

The gap component averages this curve over targets:

$$q_{\text{gap}} = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \phi_{\text{gap}}(g_j(S)).$$

Let $n_{\min}$ and $n_{\max}$ be the configured lower and upper satellite-count anchors. The scarcity bonus is

$$q_{\text{sat}} = [(n_{\max} - n)/(n_{\max} - n_{\min})]_0^1,$$

and it is applied only after every target reaches its revisit requirement:

$$N_{\text{Revisit}} = \begin{cases} 70 q_{\text{gap}}, & q_{\text{gap}} < 1, \\ 70 + 30 q_{\text{sat}}^2, & q_{\text{gap}} = 1. \end{cases}$$

Relay Constellation. Let $\mathcal{D}$ be the demand set. Demand d has a weight ω_d, two endpoints, and requested sample times $\mathcal{T}_d$. A solution contains added satellite states and link-activation intervals. At sample time t, the active validated links define a graph

$$G_t(S) = (\mathcal{N}, E_t(S)),$$

where $\mathcal{N}$ contains endpoints and satellites, and $E_t(S)$ contains validated ground or inter-satellite links. Demand d is served at time t if its endpoints are connected in $G_t(S)$ under the route-allocation rules. The per-demand service fraction is

$$\text{service}_d(S) = \frac{|\{t \in \mathcal{T}_d : \ d \text{ is served at } t\}|}{|\mathcal{T}_d|}.$$

The aggregate service fraction s is the demand-weighted mean,

$$s = \frac{\sum_{d\in\mathcal{D}} \omega_d \, \text{service}_d(S)}{\sum_{d\in\mathcal{D}} \omega_d},$$

and $s_{\min} = \min_{d\in\mathcal{D}} \text{service}_d(S)$. The remaining native metrics are the number of added satellites n, mean latency $\bar{\ell}$, and 95th-percentile latency ℓ_{95}, computed over served samples. The service core is

$$q_{\text{svc}} = 0.75s + 0.25s_{\min}.$$

If $s < 1$, then $N_{\text{Relay}} = 70q_{\text{svc}}$. For $s = 1$, use added-satellite anchors $n_{\min}, n_{\max}$ and latency cap $L_{\max}$:

$$u_n = \left[\frac{n_{\max} - n}{n_{\max} - n_{\min}}\right]_0^1, \qquad q_{\text{sat}} = u_n^2,$$

$$q_{\text{lat}} = 0.60\left[1 - \frac{\bar{\ell}}{L_{\max}}\right]_0^1 + 0.40\left[1 - \frac{\ell_{95}}{L_{\max}}\right]_0^1,$$

$$N_{\text{Relay}} = 70 + 30(0.75q_{\text{sat}} + 0.25q_{\text{lat}}).$$

D Solver Baselines

The solver-reference rows in Table 3 are casewise quality anchors under the same normalization as the agent rows. For each held-out case, we score every available reference with the family rule in Appendix C. The first solver row averages the best reference per case; when a second reference exists, the second row averages the second-best reference. A row may therefore combine different named references across cases. SPOT-5 has one solver row because each held-out case uses one archived reference-solution source.

Five families use implemented references that write the same submitted artifact type as agents and are checked by the family verifier. Table 9 lists these public rows and their core abstractions. SatNet uses literature-reported Delta-MILP and PPO values for the same oversubscribed weeks (Claudet et al., 2022; Goh et al., 2022). SPOT-5 matches held-out instances to archived challenge solutions and recomputes their profit and resource weight before normalization.

Score Relationship. Appendix C defines the family score N_f. Higher-is-better native metrics map directly to higher normalized scores, lower-is-better metrics are inverted, and Revisit and Relay keep their gates: satellite-count or latency terms enter only after the revisit or service requirement is met.

E Agent Environment

This appendix records the run configuration used for the agent rows in Table 3. An evaluated system consists of a configured model, a harness, and the prepared workspace in which the run occurs. The main-experiment workspace supplies the task brief, case files, output contract, agent-local instructions, and verifier helper before the harness is launched. Table 10 summarizes the layers exposed to the run.

The main-experiment workspace is initialized with the following structure:

```
workspace/
  .agents/skills/ (or .claude/skills)
  case/
  AGENTS.md (or CLAUDE.md)
  README.md
  verifier
```

The official scorer consumes the final artifact after the run stops. The workspace always includes a runnable verifier helper as an opaque binary; ablation runs may add procedure or memory materials, whose condition definitions appear with the corresponding ablation rows in Appendix I.

Table 11 reports the harness package version and reasoning configuration for the five evaluated systems, and Table 12 records the pinned Python libraries in the base runtime. The evaluated-system names carry the model identities.

Family	Reference	Core abstraction
AEOSSP	MWIS conflict graph (Eddy and Kochenderfer, 2021)	Enumerates feasible observation candidates, links incompatible candidates in a conflict graph, and selects a weighted independent set before verifier-facing repair.
AEOSSP	Greedy LNS (Antuori et al., 2025)	Builds a satellite-local schedule by greedy candidate insertion, then reinserts bounded neighborhoods to improve completion and turnaround.
SatNet	Delta-MILP (Claudet et al., 2022)	Uses a mixed-integer contact-assignment model to allocate antenna time while controlling request-level unsatisfied demand.
SatNet	PPO (Goh et al., 2022)	Uses a trained reinforcement-learning policy to choose contact assignments under the SatNet request-service metric.
SPOT-5	Reference lookup (Lemaître et al., 2002)	Matches known held-out instances to archived challenge solutions and recomputes their profit under the benchmark verifier.
Stereo Imaging	CP/local search (Lemaître et al., 2002)	Constructs a library of pair and tri-stereo products, then inserts and repairs products as coupled scheduling objects.
Stereo Imaging	Pruned MILP (Kim et al., 2020)	Prunes observation windows and stereo products before solving a coverage-first mixed-integer selection model.
Regional Coverage	CP local search (Antuori et al., 2025)	Generates roll-only strip candidates, scores marginal weighted coverage, and repairs sequence neighborhoods with bounded CP.
Regional Coverage	CELF selection (Leskovec et al., 2007; Nemhauser et al., 1978; Khuller et al., 1999)	Treats fixed strip candidates as a submodular coverage set and applies lazy marginal-gain selection with schedule filtering.
Revisit Constellation	J2 RGT set cover (Lee et al., 2024)	Searches J2 repeat-ground-track shells, expands RAAN candidates, and schedules confirmed target assignments to reduce revisit gaps.
Revisit Constellation	RGT/APC constructive (Zhang et al., 2018; Lee et al., 2020; Mercado-Martínez et al., 2025)	Builds repeat-ground-track access profiles, selects gap-improving satellites, and schedules observations by target freshness.
Relay Constellation	MCLP+TEG (Williams Rogers et al., 2026; Gerard et al., 2026)	Selects relay candidates by contact opportunity coverage, then emits route-aware link activations over a time-expanded graph.
Relay Constellation	UMCF/SRR (Grislain et al., 2022; Lamothe et al., 2023)	Builds dynamic communication graphs, solves path-restricted flow relaxations, and rounds paths into verifier-compatible link intervals.

Table 9: Solver-reference inventory for Table 3. The table names the public reference rows and states the core algorithmic idea for each row; evidence status and score mapping are described in the surrounding text.

Layer	Contents exposed to the run
Task package	Rendered family brief, rendered task prompt, case files, required output contract, and the runnable opaque verifier helper.
Runtime tools	Shared Linux container with Python 3.13, Node.js 24.x, OpenJDK 17, shell tools, Git, JSON utilities, file-search utilities, and the evaluated agent CLI packages.
Python libraries	Pinned astrodynamics, geometry, optimization, and data libraries, including Brahe, Basilisk, Orekit bindings, Skyfield, OR-Tools, PuLP, NetworkX, NumPy, SciPy, pandas, Shapely, pyproj, and plotting utilities.
Agent-local guidance	The Brahe skill document is installed under `.agents/skills/` for each harness where supported. Procedure-injection and memory-accumulation experiments add only the configured procedures or prior-run notes for the corresponding condition.
Execution controls	A fixed two-hour wall-clock limit, 8 CPU allocation, 32 GB memory limit, 16 GB shared-memory allocation, no human repair during the run, and post-run official scoring of the final submitted artifact.
Collected evidence	Final submitted artifact, official verifier output, aggregate metrics, and the harness-specific session logs needed for later trace analysis.

Table 10: Prepared run environment. The benchmark controls the task package and official scorer; the evaluated system controls the interaction policy inside the prepared workspace.

Representative fragments from the agent-facing task material are shown in Appendix F. The full materials are longer because each family also includes field-level schemas and modeling details.

Agents may iterate locally during the run, but the recorded artifact is the submitted solution file present at termination. Missing files, invalid files, and valid low-quality plans are separated by the protocol. Results are interpreted at the agent-system level: model, harness, workspace, and submitted artifact together define the measured behavior.

Evaluated system	Harness package	Reasoning configuration
Claude Code + Claude Opus 4.6	`@anthropic-ai/claude-code` 2.1.123	high
Codex CLI + GPT-5.4	`@openai/codex` 0.125.0	high
Kimi CLI + Kimi K2.6	`kimi-cli` 1.40.0	thinking
OpenCode + MiniMax M2.7	`opencode-ai` 1.14.30	thinking
OpenCode + DeepSeek V4 Pro	`opencode-ai` 1.14.30	max

Table 11: Harness package and reasoning configuration for the five evaluated systems. Versions are taken from the shared base runtime.

Layer	Pinned versions in the base runtime
Astrodynamics	`bsk==2.9.0`; `brahe==1.4.2`; `orekit-jpype==13.1.4.0`; `shapely==2.1.2`; `skyfield==1.54`; `pyproj==3.7.2`; `pygmo==2.19.8`.
Optimization	`ortools==9.15.6755`; `pulp==3.3.0`; `networkx==3.6.1`; `numpy==2.2.6`; `scipy==1.17.1`; `sympy==1.14.0`.
Scientific	`pandas==2.3.3`; `pydantic==2.13.3`; `matplotlib==3.10.7`; `plotly==6.7.0`; `scikit-learn==1.8.0`; `datasets==4.8.4`; `huggingface-hub==1.12.0`; `kagglehub==1.0.0`; `pyyaml==6.0.3`; `pytest==9.0.3`; `tqdm==4.67.1`; `pyinstaller==6.20.0`.

Table 12: Base runtime Python library versions available to all evaluated systems inside the shared container.

F Representative Prompt Fragments

This appendix records selected fragments from the agent-facing task material, chosen for transparency about the contract seen by agents. The complete materials also include schemas, file descriptions, examples, and family-specific details, which are too long to reproduce here. The excerpts below preserve operative wording; line breaks may be wrapped for print, ellipses mark omitted neighboring text, and workspace-relative tokens are retained when they are part of the prompt contract.

Shared Workspace Rules (AGENTS.md).

```
...
- Treat `solution.json` as the required final deliverable unless the
workspace
  says otherwise.
- If the workspace exposes a verifier helper, use it for local iteration
when
  helpful.
- The run may be stopped after 2 hours. As soon as you have any valid or
  likely-valid answer, write it to `solution.json` and keep that file
  valid
  while you continue improving it.
- Prefer incremental improvement: preserve the best working
`solution.json`
  you have, and only replace it after the replacement is written
  completely
  and is at least as likely to verify.
...
```

Brahe Skill.

```
---
name: brahe
description: |
  ...
---

# Brahe Skill

Curated documentation and runnable examples for the Brahe Python
library.

...

## Module Map

See more examples and documents on how to use brahe:

| Topic | Reference |
|-------|-----------|
| **Time & EOP** |
[docs/learn/time/index.md](docs/learn/time/index.md),
  [docs/learn/eop/index.md](docs/learn/eop/index.md) |
| **Coordinates** |
[docs/learn/coordinates/index.md](docs/learn/coordinates/index.md) |
| **Propagation** | [docs/learn/orbit_propagation/index.md](docs/lear⌋
n/orbit_propagation/index.md),
  [docs/learn/orbit_propagation/numerical_propagation/index.md](docs/⌋
  learn/orbit_propagation/numerical_propagation/index.md) |
| **Access** | [docs/learn/access_computation/index.md](docs/learn/ac⌋
cess_computation/index.md) |
...
```

Short Task Instructions.

```
Stereo Imaging:
Please solve the prepared stereo-imaging case using the files in
`case/`.
...
Write `solution.json` at the workspace root. You have a 2-hour timeout,
so
preserve a valid or likely-valid `solution.json` as soon as possible
and keep
improving it in place. Prioritize normalized stereo quality first, then
improve valid stereo coverage where you can.

Relay Constellation:
Please solve the prepared relay-network augmentation case using the
files in
`case/`.
...
Write `solution.json` at the workspace root. You have a 2-hour timeout,
so
preserve a valid or likely-valid `solution.json` as soon as possible
and keep
improving it in place. Build a feasible augmentation and link plan
first, then
push service up and latency down.

Revisit Constellation:
Please solve the prepared revisit-driven constellation planning case
using the
files in `case/`.
...
Write `solution.json` at the workspace root. You have a 2-hour timeout,
so
preserve a valid or likely-valid `solution.json` as soon as possible
and keep
improving it in place. Produce a feasible constellation and observation
schedule, then push the revisit gaps down as far as you can.
...
```

AEOSSP.

```
...
Do not submit visibility claims, maneuver windows, power traces, or
completion
claims.
Those are derived during validation.
...
```

Stereo Imaging.

```
The output should schedule raw observations only. Stereo pairing,
tri-stereo grouping, overlap checks, convergence checks, and quality
scoring
are derived during validation.
...
The satellite set is fixed by TLEs in `case/satellites.yaml`; you
choose only
raw observation windows and two boresight steering angles.
`off_nadir_along_deg` tilts along the flight direction and
`off_nadir_across_deg` tilts cross-track in the satellite local frame.
...
Target access is derived from the target center at `longitude_deg`,
`latitude_deg`, and `elevation_ref_m`; it is not a submitted claim.
...
```

Regional Coverage.

```
Your job is to produce `solution.json`, a schedule of
`strip_observation`
actions that maximizes unique weighted regional coverage while
remaining valid.
...
Do not submit your own strip polygons, coverage claims, or access-window
identifiers. Strip geometry and coverage are derived during validation.
...
the only attitude command is signed `roll_deg`, the cross-track
off-nadir
angle of the strip center. Positive and negative signs look to opposite
sides
of the ground track.
...
```

Revisit Constellation.

```
- a proposed constellation at mission start
- a schedule of observation actions for that constellation
...
The goal is to keep revisit gaps small across the targets while
respecting the
orbit, visibility, timing, slew, and power constraints.
...
In practical terms, feasibility comes first. After that, the solution
should
drive revisit gaps down as much as possible, and efficient
constellation size
matters once the required revisit quality is achieved.
...
```

Relay Constellation.

```
The modeled decision is: add a limited number of relay satellites and
choose
when physical links are active. You do not submit routes, per-demand
service
assignments, or latency calculations. Instead, the validator builds a
time-varying communication graph from your active links and then
computes how
much demand can actually be served through that graph.
...
With multiple active demands, allocation maximizes total served demand
`weight`, then minimizes total latency, then uses deterministic path
ordering
as a tie-breaker.
...
```

Injected Procedures.

Regional coverage procedure.

```
| Valid nonzero but weak | Build a small candidate table and add one
fresh-coverage strip at a time. |
...
When comparing candidate moves, use this order after each verifier run:
1. validity
2. higher `weighted_coverage_ratio`
...
```

Relay constellation procedure.

```
If `valid=true` and `service_fraction=0`, the next move is not latency
optimization. The next move is to complete one served path.
...
Private propagation or visibility code is a candidate generator, not
proof. If
the verifier disagrees, change the candidate.
...
```

SatNet.

```
Prioritize request satisfaction and mission-level fairness: minimize
`U_rms`
first, minimize `U_max` next, then use satisfied request count and valid
communication time as tie-breakers.
...
U_rms = sqrt(mean(U_i^2))
U_max = max(U_i)
...
```

SPOT-5.

```
This problem is modeled as a constrained assignment-and-selection
problem
rather than as orbital propagation.
...
Assignment `0` rejects the photograph. A nonzero assignment selects it
and
must be in that variable's domain. Values `1`, `2`, and `3` are
mono-camera
choices. Value `13` is the dual-camera choice and is legal only when the
variable's domain explicitly contains `13`.
...
Header mismatches in `claimed_profit` or `claimed_weight` may produce
warnings
rather than immediate invalidity, but the assignment vector itself must
still
satisfy all domain, conflict, and capacity rules.
...
computed_profit = sum(profit_i for i where assignments[i] != 0)
computed_weight = sum(normalized_weight_i for selected variables)
computed_weight <= 200
...
```

G Uncertainty in the Main Results

Table 13 reports 95% percentile case-bootstrap confidence intervals for the agent-system mean scores. For each family, we enumerate all $5^5 = 3125$ ordered samples of five cases drawn with replacement, using the same case indices across systems. We recompute the mean for each sample and take the 2.5th and 97.5th percentiles, with linear interpolation, as the interval endpoints. All original zero-score outcomes are retained. The recorded run for each system–case pair is held fixed, so these intervals summarize case-resampling variation and do not estimate variability across repeated agent runs. With only five cases per family, they provide a limited uncertainty summary; an all-zero interval reflects five observed zero scores and does not establish zero performance on unseen cases.

H Supplementary Self-Hosted System Evaluation

We evaluated OpenCode 1.14.19 with self-hosted Qwen3.6-27B through an OpenAI-compatible endpoint on the five held-out SPOT-5 cases in three batches. Each agent workspace was configured with 8 CPUs, 32 GB memory, and a 7,200-second run budget. Table 14 reports the normalized scores under the same validity and scoring rules as the main evaluation. Eight of the 15 recorded attempts produced verifier-valid submissions; six produced no submission and one produced an invalid submission. This supplement is limited to the tested system configuration and SPOT-5 cases.

Family	Claude	Codex	Kimi	OC+MM	OC+DS
AEOSSP	58.45 [29.06, 73.95]	72.91 [69.87, 76.15]	72.03 [69.43, 74.14]	36.44 [16.60, 52.71]	71.44 [69.15, 73.73]
Regional	21.22 [20.71, 22.22]	72.08 [60.04, 80.37]	73.15 [67.40, 77.38]	17.73 [9.98, 24.82]	35.79 [24.15, 44.92]
Relay	12.25 [0.00, 36.75]	64.91 [55.81, 77.71]	57.77 [42.85, 72.63]	0.00 [0.00, 0.00]	35.67 [11.55, 59.78]
Revisit	76.50 [73.72, 78.85]	69.58 [61.90, 76.01]	73.36 [69.78, 78.13]	0.00 [0.00, 0.00]	70.95 [69.98, 71.93]
SatNet	60.36 [29.25, 79.41]	69.45 [61.78, 77.13]	66.14 [58.88, 74.96]	37.54 [16.55, 55.85]	60.23 [50.07, 70.39]
SPOT-5	43.96 [11.09, 76.83]	61.92 [44.58, 82.18]	61.92 [44.58, 82.18]	37.47 [7.15, 71.09]	60.91 [43.99, 82.17]
Stereo	19.16 [0.31, 55.67]	68.91 [33.64, 93.67]	0.25 [0.00, 0.75]	0.00 [0.00, 0.00]	13.00 [0.00, 38.99]

Table 13: Mean normalized scores with 95% case-bootstrap confidence intervals beneath them, computed from five held-out cases per family. Systems and model versions are those in Table 3; OC+MM and OC+DS denote OpenCode with MiniMax and DeepSeek, respectively. Intervals describe each system's mean and are not pairwise significance tests.

Repetition	Case 8	Case 28	Case 1021	Case 1403	Case 1506	Valid	Mean
1	100.00	0.00	54.80	0.00†	64.36	3/5	43.83
2	100.00	34.37	0.00	0.00	0.00	2/5	26.87
3	100.00	34.37	0.00	35.60	0.00	3/5	33.99

Table 14: Supplementary SPOT-5 normalized scores for OpenCode with self-hosted Qwen3.6-27B. Missing or invalid submissions receive zero, and each mean includes all five cases. † The recorded attempt failed during provider/model lookup before inference and produced no submission; its zero is retained under the missing-submission rule.

I Family-Native Metrics and Ablations

This appendix reports the family-native metrics behind Table 3 and the per-system ablation rows behind Section 5.3. The first subsection gives the raw mission metrics the source formulations optimize, so the normalized mapping in Appendix C can be read against the quantities it aggregates. The procedure-injection and memory-accumulation subsections then report five-case rows the main body only summarizes, with missing or invalid submissions scored zero: procedure injection tests whether written task procedures change search behavior, and memory accumulation tests whether prior-run notes transfer. Each ablation covers the systems and families needed for its intervention.

I.1 Family-Native Metrics — Main Experiment

Table 15 reports the raw mission metrics behind the normalized results in Table 3, so the mapping in Appendix C can be read against the quantities each source formulation optimizes. The families divide into three scoring types. Three combine several native terms under fixed weights: AEOSSP over completed-weight ratio, completion ratio, turnaround, and power; SatNet over two unsatisfied-demand statistics; Regional over two coverage terms with an action-count and a battery margin. Two are gated rather than linear: Revisit scores a revisit-gap curve and adds a satellite-count bonus only once every target meets its requirement, and Relay scores demand service and worst-group service and adds a latency and satellite bonus only at full service. The last two are single-objective, where the score is a monotone rescaling of one metric: selected profit for SPOT-5 and product quality for Stereo Imaging. A normalized scale is needed where a family folds several mission terms into one comparable number, which is five of the seven families.

SatNet is the sharpest test of whether normalization flatters the agent rows. Under its published unsatisfied-demand metrics, the strongest agent system reaches $U_{\mathrm{rms}} = 0.234$ and $U_{\max} = 0.519$, improving on both the mixed-integer baseline $(0.302, 0.673)$ and the reinforcement-learning baseline $(0.316, 0.772)$. The normalized scorer records a narrower lead, 69.45 against the 61.59 solver-reference row, because it compresses the unsatisfied-demand range onto a bounded scale.

The native columns also locate where the agent rows genuinely trail, on the raw quantity rather than as an effect of the mapping. Regional Coverage separates on its weighted coverage term, where the strongest agent reaches 0.944 against references at 0.995 and 0.968, and where one agent returns valid but empty schedules at zero coverage. That system's nonzero normalized score comes entirely from the action-count and battery-margin terms, so its best-marked zero action count reflects not acting at all rather than better task performance. Stereo Imaging separates on product quality, 0.689 for the strongest agent against 0.961 and 0.920 for the references; because Stereo is single-objective, its native and normalized values agree by construction, so this gap is a real quality deficit and not a normalization artifact.

Read across families, the native columns reproduce the strong-to-weak ordering of Table 3. The divergences fall on the weighted and gated families, whose score folds in secondary mission terms beyond the leading native metric, such as SatNet's worst-group ratio, the Regional action and battery margins, or the Relay worst-group service. The normalized scale aggregates mission value across these terms rather than relabeling a single metric, which is why it can be read against the native columns without reducing to them.

Family / metric	Agent systems					Reference	
	Claude	Codex	Kimi	OC+MM	OC+DS	Ref1	Ref2
AEOSSP (weighted)							
valid (/5)	5	5	5	5	5	5	5
WCR (.45) ↑	0.570	0.708	0.692	0.201	0.682	**0.758**	0.682
CR (.20) ↑	0.594	0.736	0.720	0.176	0.710	**0.790**	0.721
TAT (s) (.20) ↓	9477	1025	**973**	9369	987	1018	1128
PC (kWh) (.15) ↓	16.5	19.1	18.8	**9.8**	18.7	19.8	18.5
SatNet (weighted)							
valid (/5)	4	5	5	4	5	5	5
U_{rms} (.75) ↓	0.354	**0.234**	0.248	0.550	0.268	0.302	0.316
$U_{\max}$ (.25) ↓	0.525	**0.519**	0.611	0.849	0.787	0.673	0.772
Regional Coverage (weighted)							
valid (/5)	5	5	5	5	5	5	5
Cov_w (.50) ↑	0.000	0.880	0.944	0.044	0.397	**0.995**	0.968
Cov (.20) ↑	0.000	0.881	0.945	0.039	0.374	**0.995**	0.970
actions (.15) ↓ †	**0.0**	45.8	60.4	27.6	54.2	—	—
min batt (Wh) (.15) ↑ †	**495**	494	494	**495**	492	**495**	494
Revisit Constellation (gated)							
valid (/5)	5	5	5	0	5	5	5
gap (h) ↓ (gate)	**6.80**	7.67	6.84	48.00	6.84	**6.80**	8.83
sats ↓ (bonus) †	**11.4**	16.0	14.4	—	17.0	17.2	19.8
Relay Constellation (gated)							
valid (/5)	1	5	5	4	5	5	5
service ↑	0.189	**0.935**	0.868	0.000	0.559	0.933	0.928
worst-grp ↑	0.133	**0.684**	0.522	0.000	0.360	0.633	0.643
SPOT-5 (single objective)							
valid (/5)	3	5	5	4	5	5	—
profit (k) ↑	68.9	**115.3**	**115.3**	45.5	112.1	112.3	—
selected ‡	125.2	178.2	177.8	62.4	173.4	—	—
Stereo Imaging (single objective)							
valid (/5)	3	5	5	1	4	5	5
quality ↑	0.192	0.689	0.003	0.000	0.130	**0.961**	0.920
coverage ‡	0.197	0.723	0.003	0.000	0.188	0.971	0.944

Table 15: Family-native metrics behind Table 3, in raw units, for the five agent systems (Claude = Claude Code + Claude Opus 4.6; Codex = Codex CLI + GPT-5.4; Kimi = Kimi CLI + Kimi K2.6; OC+MM = OpenCode + MiniMax M2.7; OC+DS = OpenCode + DeepSeek V4 Pro) and the two strongest reference solvers per family. Pale green, blue, and yellow column shading separate metric labels, agent-system columns, and reference columns; darker bold cells mark the best value in rows with a displayed preference direction, with ties all marked. Each row is one native term the family score uses; the parenthetical weight is its coefficient in the normalized mapping of Appendix C and the arrow its preferred direction (↑ higher better, ↓ lower better). Families divide into three types: *weighted* families combine the listed terms linearly; *gated* families score a revisit-gap or service core and add a post-gate bonus, so the terms enter nonlinearly and no single weight applies; *single-objective* families have a score that is a monotone rescaling of one metric. Cells are penalized five-case means: invalid or missing cases take the score-zero native value, namely zero for higher-is-better metrics, one for the bounded SatNet ratios, the 48-hour horizon for the revisit gap, and the 12-hour horizon for AEOSSP turnaround on cases with no completed task. *valid (/5)* counts verifier-valid cases. † marks count and battery-margin terms reported as valid-only means, since a penalty for a count is ill-defined; ‡ marks unscored context metrics. Profit is in thousands of objective points, power in kWh, turnaround in seconds, the revisit gap in hours, battery in Wh. Reference columns hold fixed named solvers rather than Table 3's per-case envelope: Ref1 and Ref2 are MWIS conflict-graph and greedy-LNS (AEOSSP), Δ-MILP and PPO (SatNet), CP local-search and CELF (Regional), J2-RGT set-cover and RGT-APC constructive (Revisit), MCLP-TEG and UMCF-SRR (Relay), CP local-search and pruned MILP (Stereo), and reference lookup (SPOT-5, one reference). The Relay full-service latency and satellite bonus is gated per case at full service: Codex and Kimi each reach full service on one of the five cases, so their Relay means include that bonus, while the reference rows never reach full service.

I.2 Family-Native Metrics — Ablations

The procedure-injection and memory-accumulation ablations are reported here in the same native units as Table 15, for the two families common to both interventions, Regional Coverage and Relay Constellation, and the two OpenCode systems they target. Tables 16 and 17 give each family's primary native metric, weighted coverage ratio for Regional Coverage and demand service fraction for Relay Constellation, as a penalized five-case mean with invalid cases scored zero; both are higher better. The no-procedure and no-memory columns are the same baseline runs as the corresponding rows of Table 15.

Procedure injection. The compact procedure gives a short domain workflow and the procedure pack adds broader search and domain guidance. OpenCode + DeepSeek V4 Pro shows the clearest effect on Regional Coverage, where the full pack raises weighted coverage from 0.40 to 0.64 while using fewer actions on average than the no-procedure baseline, consistent with a construction loop that broadens candidate strips, ranks them by marginal coverage, and preserves the best verified incumbent. Relay Constellation is more conjunctive, since placement, ground and inter-satellite links, endpoint capacity, simultaneous visibility, and demand-window routing must align before service appears: service rises from 0.56 to 0.66 under the compact procedure but does not improve further under the full pack. OpenCode + MiniMax M2.7 gains on both families from a low base, reaching 0.23 weighted coverage and 0.17 service, and its worst-served demand window stays at zero throughout, so the procedure widens partial service without closing the hardest demand.

Memory accumulation. Each memory condition mounts a frozen note set. A donor system (Codex CLI + GPT-5.4, or OpenCode + DeepSeek V4 Pro) accumulated notes over twelve training cases, two per family for the six families with a training split, and the resulting snapshot is mounted unchanged into every evaluated run: no note derives from a held-out test case, and nothing accumulates across test runs. The mounted set always contains notes from all six donor families, so the reported conditions mix same-family training notes with cross-family notes. Memory is more specific than a static procedure and higher variance, since prior-run notes may carry concrete candidate families, warnings from earlier verifier failures, or habits for preserving a valid solution. On Relay Constellation, OpenCode + DeepSeek V4 Pro improves with memory from 0.56 service with no memory to 0.68 with Codex-derived notes and 0.83 with its own prior-run notes, the largest native gain in either ablation. Regional Coverage shows the transfer-mismatch boundary: Codex-derived notes raise weighted coverage to 0.72, above the 0.65 from the reader's own notes, so a donor candidate family can transfer better than self-derived memory when it matches the new geometry. OpenCode + MiniMax M2.7 again gains only marginally and stays below 0.18 on both families; on Relay its DeepSeek-derived condition also drops two of five cases to invalid, so the penalized service of 0.11 reflects reliability loss rather than quality alone.

J Temporal-Robustness Check

The AEOSSP temporal check asks whether the held-out result is tied to one particular orbital epoch. It compares the default held-out split with a shifted-horizon split built from a later source epoch while keeping the task family and scoring convention fixed. Codex CLI + GPT-5.4 remains close across the two splits, with mean weighted completion ratio changing from 0.7075 to 0.6949. OpenCode + DeepSeek V4 Pro changes from 0.6820 to 0.6816. The two solver references are similarly stable: conflict-graph scheduling changes from 0.7582 to 0.7667, and greedy large-neighborhood search from 0.6816 to 0.6874.

This check isolates one source of temporal sensitivity: the orbital epoch used to generate AEOSSP cases. Under the shifted horizon, both agent systems and solver references preserve their relative behavior, indicating that the AEOSSP comparison is not tied to one frozen time window.

K Additional Case Studies

This appendix gives trace-level evidence for the mechanisms summarized in Section 5 and Figure 2. Each example links an agent-facing material choice, such as task wording, workspace packaging, injected procedure, memory note, or final-artifact convention, to an observed run behavior and official outcome. The examples cover prompt salience, signed-frame binding, procedure granularity, memory-prior transfer, and canonical deliverable handoff. Figure 4 previews the prompt-salience, signed-frame, and memory-transfer exam-

System	No procedure	Compact	Procedure pack
Regional Coverage (weighted coverage ratio ↑)			
OpenCode + DeepSeek V4 Pro	0.397	0.415	0.644
OpenCode + MiniMax M2.7	0.044	0.127	0.225
Relay Constellation (service fraction ↑)			
OpenCode + DeepSeek V4 Pro	0.559	0.658	0.601
OpenCode + MiniMax M2.7	0.000	0.124	0.174

Table 16: Procedure-injection ablation in native units. Entries are penalized five-case means of each family's primary native metric, both higher better, for the compact-procedure and full procedure-pack conditions against the no-procedure baseline. Procedures help most when they turn construction into verifier-aligned candidate generation and incumbent preservation, as in Regional Coverage; in Relay Constellation broader guidance reduces empty-service collapses but does not remove the coupled placement, link-timing, and routing burden.

System	No memory	Codex-derived	DeepSeek-derived
Regional Coverage (weighted coverage ratio ↑)			
OpenCode + DeepSeek V4 Pro	0.397	0.725	0.646
OpenCode + MiniMax M2.7	0.044	0.097	0.175
Relay Constellation (service fraction ↑)			
OpenCode + DeepSeek V4 Pro	0.559	0.681	0.833
OpenCode + MiniMax M2.7	0.000	0.043	$0.110^{\dagger}$

Table 17: Memory-accumulation ablation in native units. Entries are penalized five-case means of each family's primary native metric, comparing no-memory runs with memory accumulated by a Codex donor and by the OpenCode + DeepSeek V4 Pro donor. †Two of five cases are invalid in this cell, so the penalized mean substitutes zero for them; the valid-only mean is 0.184. Memory transfers concrete solver habits but is high variance: a prior candidate family can match the new case and improve search or mismatch it and regress.

ples.

Each case study starts from the official outcome and reconstructs which source the evaluated system treated as defining the task: the short task prompt, root task document, local verifier, injected procedure, memory note, or self-written checker. Prompt fragments are shown before interpretation because the prompt contract is part of the evidence. The interpretations then explain how the observed trace behavior follows from those fragments.

Prompt Salience and Contract Closure. The Regional Coverage failure starts from schema selection, but the mechanism is more specific than incomplete reading of the task materials. The short tasking cue points the system toward case files, while the task document defines the required action type and field names.

```
Please solve the prepared regional strip-coverage case using
the files in `case/`.
...
Your job is to produce `solution.json`, a schedule of
`strip_observation`
actions that maximizes unique weighted regional coverage
while remaining valid.
...
Do not submit your own strip polygons, coverage claims, or
access-window
identifiers. Strip geometry and coverage are derived during
validation.
...
Actions with another `type` are not strip observations and do
not create
coverage; use only `strip_observation` entries.
```

The trace follows the case-file cue almost literally: it starts with "using the files in case/", lists only the case directory, and later checks an agent-invented field named `roll_angle_deg`. Those checks can be internally coherent, and the final JSON can contain many rows, but the official parser recognizes `strip_observation` rows with `roll_deg`. The aggregate symptom is therefore valid but empty: all five files are valid JSON schedules after parsing, yet the parsed action count and weighted coverage are zero. The mechanism is premature contract closure. After the first durable schema is written, solvers, validators, and progress messages all reinforce that schema; the later search is real work, but it is work inside the wrong contract.

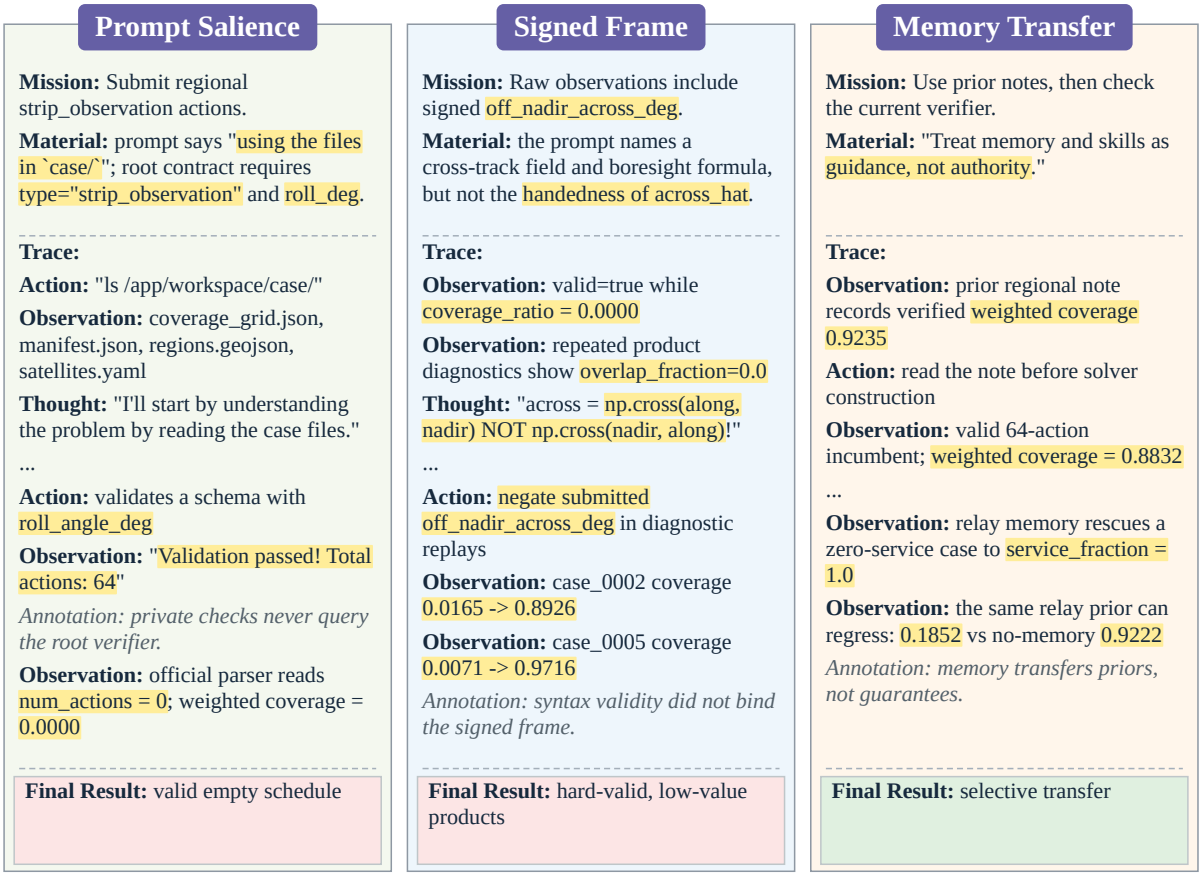


Figure 4: Selected Appendix K mechanisms connecting agent-facing material, trace behavior, and final outcome. Left: prompt salience leads a Regional Coverage run to bind to case files and an agent-inferred `roll_angle_deg` schema, producing a valid empty schedule under the official parser. Middle: a Stereo Imaging run produces hard-valid actions but loses product overlap because the signed cross-track convention is not bound. Right: memory transfers concrete search priors that help under structural match and regress under mismatch.

Signed-Frame Binding. Kimi CLI + Kimi K2.6 shows a different failure in agent-facing material: the action schema is often correct, but the signed local frame is not bound tightly enough before optimization.

```
The satellite set is fixed by TLEs in `case/satellites.yaml`;
you choose only
raw observation windows and two boresight steering angles.
`off_nadir_along_deg` tilts along the flight direction and
`off_nadir_across_deg` tilts cross-track in the satellite
local frame.
The boresight vector is proportional to:
nadir_hat
+ tan(off_nadir_along_deg) * along_hat
+ tan(off_nadir_across_deg) * across_hat
```

The system reads the task brief and uses the verifier, and the final files can pass action-level checks. The loss comes after parsing: the submitted cross-track steering signs are expressed in the opposite handedness from the verifier's local frame, so image strips land on the wrong side of the ground track. Product overlap then collapses even when convergence and timing look plausible. Diagnostic replays show that negating the across-track sign converts two near-zero schedules into high-coverage schedules. The prompt content is correct but under-specified. A short signed-frame description can still leave the handedness ambiguous for agents writing their own geometry code.

Procedure Granularity and Coupling Burden. Procedure injection helps unevenly, because a procedure improves a run only when it matches the action space. Unlike the main text's case-adaptive-search mechanism, the mechanism here is the width of the prompt-provided procedure and how much coupled state it asks the agent to coordinate.

```
Valid nonzero but weak | Build a small candidate table and
add one
fresh-coverage strip at a time. |
...
Build a small candidate table before dense sweeps:
- satellite id
- start time on `manifest.time_step_s`
- duration
- signed roll
...
```

```
If `valid=true` and `service_fraction=0`, the next move is
not latency
optimization. The next move is to complete one served path.
Private propagation or visibility code is a candidate
generator, not proof.
```

In Regional Coverage, the full procedure changes the run from recovering basic geometry into broad candidate generation, marginal unique-coverage scoring, local swaps, and verifier-backed best-so-far retention. The result is the largest procedure-injection gain among the tested families for OpenCode + DeepSeek V4 Pro, raising its mean normalized score from 35.79 to 55.25. Relay Constellation is different. It also benefits from service-first reminders, especially avoiding valid zero-service handoffs, but the broader skill pack expands a coupled search over added relays, ground links, inter-satellite links, endpoint caps, satellite link caps, timing, and routed demand windows. The compact procedure has the better mean score because it keeps the run closer to one served path and one verified candidate at a time. The same external guidance helps when it creates a local marginal loop and can hurt when it widens a conjunctive design problem faster than the agent can coordinate it.

Memory Prior Match and Mismatch. Memory accumulation shows how natural-language context can influence behavior beyond the immediate task brief: the agent reads prior run notes and imports concrete solver habits into the next case. The useful side is concrete: regional memory can transfer exact candidate-generation recipes and verified thresholds; relay memory can transfer a tested constellation family and a per-sample routing abstraction. This produces large rescues, including a no-memory Relay case with zero service becoming full service under memory. The boundary is equally concrete. The same relay prior can underserve a different demand-window pattern, and one memory-conditioned case scores far below its no-memory counterpart. Memory is therefore best interpreted as an empirical prior over solver construction. It changes where the search starts and what repair moves are salient; it does not replace case-specific verifier evidence.

Canonical Deliverable Handoff. Some failures occur after the solver has already found a better artifact. The prompt and shared workspace rules make the canonical final path part of the contract, not merely a convenience. Incumbent preservation concerns retaining a good candidate during search; this mechanism concerns whether that candidate is copied into the collected deliverable before the run ends.

```
- Treat `solution.json` as the required final deliverable
unless the workspace
  says otherwise.
- The run may be stopped after 2 hours. As soon as you have
any valid or
  likely-valid answer, write it to `solution.json` and keep
  that file valid
  while you continue improving it.
...
- Prefer incremental improvement: preserve the best working
`solution.json`
  you have, and only replace it after the replacement is
  written completely
  and is at least as likely to verify.
```

Kimi CLI + Kimi K2.6 on Revisit Constellation shows why this matters. The trace can find verified lower-satellite variants while the collected file remains weaker. Another run reaches the revisit floor, starts a riskier lower-satellite search, and terminates with a collected file that loses the primary gate. This is final-state management under a timeout, not weak physical reasoning: each improvement must be copied back before the next risky branch.

L Hallucination and Ungrounded-Submission Metrics

The failure modes in Section 5.1 and the case studies in Appendix K include hallucination-type behaviors: agents build self-imposed task contracts, optimize hallucinated objectives, trust agent-written validators over the official verifier, or hand off confident but empty artifacts. Table 18 quantifies three process-level counterparts over the full main-experiment matrix (175 runs, 35 per system). *Invalid submissions* are runs whose final collected artifact is rejected by the official verifier, typically through fabricated schema fields or malformed artifacts; 22 runs (12.6%) end this way. *Valid-but-zero submissions* pass the official verifier but score zero: the artifact is real but operationally empty; 16 runs (9.1%). *Ungrounded commitments* are runs that write a durable candidate or deliverable artifact

before reading the root workspace contract, the behavioral precondition for self-imposed task models; 39 runs (22.3%), and 8 further runs make no detectable durable commitment. These counts key on explicit read events of the root task document; across the recorded traces, no harness auto-injects that document into the model's context, although Codex CLI auto-injects the generic workspace rules file, which is not the task contract.

System	Invalid	Valid-but-zero	Ungrounded
Claude	9	1	17
Codex	0	0	1
Kimi	0	4	6
OC+MM	12	6	15
OC+DS	1	5	0
All	22	16	39

Table 18: Hallucination-related process metrics over the 175 main-experiment runs (35 per system). Invalid: the final collected artifact is rejected by the official verifier. Valid-but-zero: the artifact verifies but scores zero. Ungrounded: a durable artifact commitment occurs before the root workspace contract is read. Abbreviations follow Table 3.

Two qualifications accompany these counts. First, valid-but-zero submissions have mixed causes: three of the four Kimi CLI + Kimi K2.6 cases on Stereo Imaging stem from an across-track sign-convention mismatch that leaves the artifact valid but with zero footprint overlap, whereas the Relay Constellation cases are zero-service handoffs. Second, an ungrounded commitment is a risk factor, not a verdict: runs can still recover once they acquire the workspace contract and let the official verifier override the private model. Even so, these metrics separate systems whose outcome scores are otherwise close. Codex CLI + GPT-5.4 and OpenCode + DeepSeek V4 Pro almost never commit before reading the contract (1 and 0 of 35 runs, respectively), while Claude Code + Claude Opus 4.6 and OpenCode + MiniMax M2.7 do so in roughly half of their runs, consistent with the trace evidence in Section 5.1.