

From Experimental Artifacts to Solution Lineages

Jin Li^{1,*}, Ahmed Murtadha^{1,*}, Zhiyu Wang^{1,*}, Qiwen Chen^{1,5,*}, William Chen^{1,*}, Yifei Wu^{1,*}, Guan Wang^{1,*}, Andy L. Siy¹, Jiayi Yang¹, Mengsha Huang³, Wenhao Li^{1,2}, Yixuan Liu³, Shuailin Pan^{1,4}, Mingli Yuan¹, Sen Song³, Yuhao Sun^{1,†}

¹Sapient Intelligence ²Nanyang Technological University ³Tsinghua University
⁴Carnegie Mellon University ⁵University of Pennsylvania

github.com/sapientinc/praxist praxist.sapient.inc

Abstract. Autonomous R&D agents now write, run, and improve executable artifacts under automated evaluation—but largely as laboratory instruments: shown on curated benchmarks, with gains that are hard to trace to a cause and costs well above what sustained engineering practice absorbs. The limitation is structural. Most systems treat each attempt as nearly self-contained, so logs, memories, and search trees record what happened without establishing which design element produced an improvement, whether its evidence survived validation, or how it recombines with others. Long campaigns therefore keep re-learning the same lessons. We introduce PRAXIST, a lineage-centered generational system that converts reproducible artifacts and evaluator outcomes into a typed evidence graph of findings, lane-structured frontiers, and agendas. Separating local artifact construction from cohort-level evidence synthesis lets later attempts inherit validated mechanisms, unresolved claims, and useful constraints, and leaves results attached to an inspectable lineage. On the standardized 75-task MLE-bench suite, the finalized official-grader results give PRAXIST 60 medals (80.0%), 49 of them gold, against 55 medals (73.3%) and 34 gold for a Claude Code baseline on Claude Opus 4.8—at a recorded model spend of US\$3,054 versus US\$38,370, roughly a twelfth of the cost. Four case studies—quantitative trading, LiDAR-inertial-visual SLAM, tokamak magnetic control, and rocket landing—carry the same process into open-ended engineering problems, improving on each task-native baseline in headline accuracy, survival, or resource cost, with the discovery path on record. Stronger artifacts at an order of magnitude less spend, each backed by an auditable lineage, are, to our knowledge, first brought together here: the operating profile production research requires, not the one a benchmark demonstration establishes.



Overview of PRAXIST: selective evidence inheritance turns distributed, evaluated artifacts into lineage-grounded research state, improving results on MLE-bench and in open-ended R&D case studies.

[†] Corresponding author. * Equal Contribution. Contact: praxist@sapient.inc.

1 Introduction

Complex engineering and scientific breakthroughs are rarely found in a single attempt; they are constructed cumulatively from validated intermediate discoveries. Recognizing this, recent autonomous R&D systems solve problems by constructing executable artifacts, evaluating them with task-grounded feedback, and using the results to guide later attempts^{1,2,3,4,5,6}. The same pattern underpins machine-learning engineering benchmarks^{7,8,9,10,11,12}, agents for software engineering^{13,14,15}, and scientific reproduction benchmarks^{16,17}: language models sustain long-horizon discovery when generation is coupled to automated evaluation.

As these systems move from isolated attempts to long-running campaigns, they must decide how previous attempts shape future construction. Most preserve campaign state as a search tree over candidate solutions^{6,1,2,4,5}. As the tree grows, each node artifact becomes increasingly compositional, combining data choices, algorithms, hyper-parameters, implementation techniques, and diagnostics from many attempts. Since a mechanism’s value may emerge only through combination, artifact-level search can prune components before their utility becomes visible. Campaigns therefore generate extensive evaluated experience, yet only a limited portion of what they learn is retained in a form that later attempts can build on.

Assembly theory suggests a different primitive: it characterizes complex objects through the formation histories that build them from reusable substructures, with selection determining which

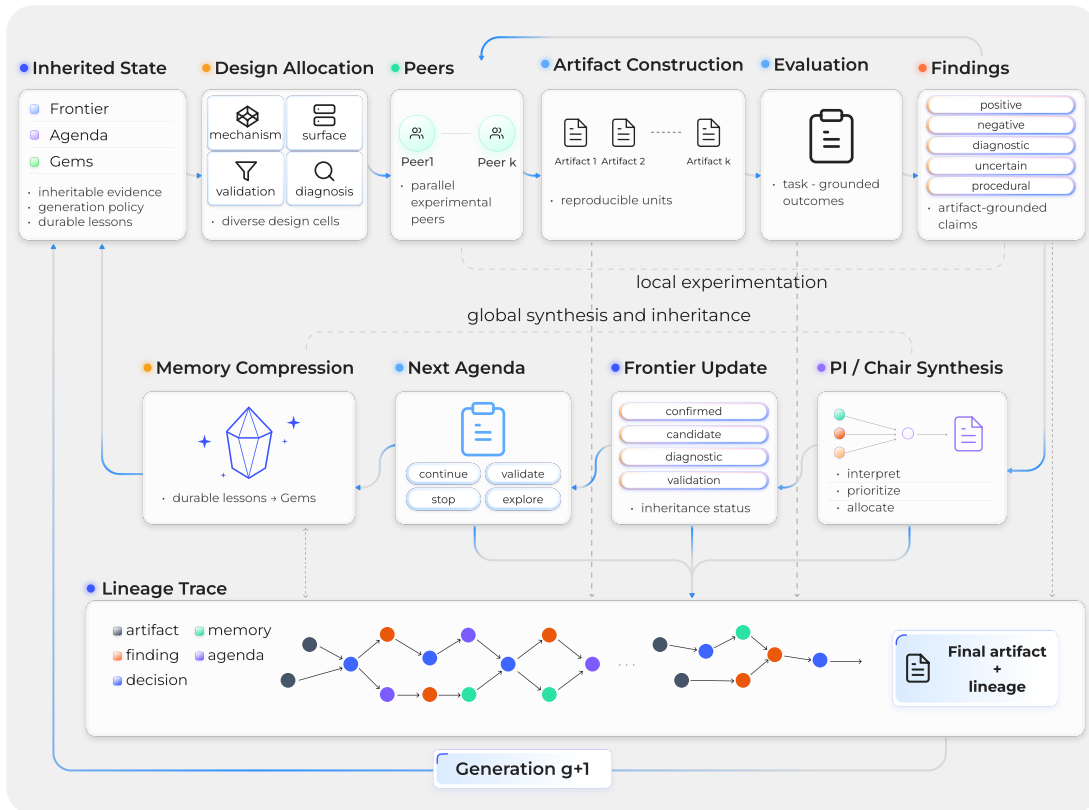


Figure 1: One generation of the PRAXIST loop. Agents inherit the accumulated evidence and run deliberately different experiments, each producing a reproducible artifact scored by an external evaluator (top). Outcomes become typed claims, pooled into updated evidence and the next generation’s plan: continue, stop, validate, or explore (middle). A lineage records every step (bottom).

persist^{18,19,20}. Because every step operates on parts already retained, cost scales with the number of assembly operations rather than the size of the space, and each element is amortized across later constructions. Search then need not foresee which variant will matter, only retain the right elements in the right roles and keep them recombining. We call this conversion of evaluated outcomes into actionable, artifact-grounded state evidence inheritance.

Several traditions supply relevant capabilities: reflection and memory reuse observations across attempts^{21,22,23,24}; multi-agent systems distribute planning and revision across roles^{25,26,27}; graph structures organize agent computation, reasoning, and memory^{28,29,30,31,32}; quality-diversity search maintains coverage across high-performing regions^{33,34,35}; and provenance systems link outputs to the processes producing them^{36,37,38,39}. Cumulative construction also requires a common interface between evaluation and inheritance: each result must state what was learned, how strongly it is supported, what grounds it, and its role in later work.

We introduce PRAXIST, a lineage-centered system that implements this interface for evaluator-grounded autonomous R&D. PRAXIST organizes each campaign as an artifact-to-lineage process that repeats over generations, separating parallel artifact construction from cohort-level evidence synthesis; Figure 1 traces one such generation. Each generation begins from inherited evidence and produces *design contracts*: pre-artifact plans fixing a mechanism, intervention surface, research intent, parent lineage, evidence signature, and validation hook. A cohort-level allocation rule, *Quantified Diversity* (QD), spreads these contracts across distinct design cells of mechanism family, intervention surface, and intent, keeping coverage broad.

A cohort of *peers* constructs reproducible artifacts against the assigned contracts, an external evaluator scores each one, and the outcomes are interpreted into typed *findings* that PI roles read from complementary perspectives. Synthesis promotes eligible findings onto a *frontier* with confirmed, candidate, diagnostic, and validation lanes, and a Chair turns the panel memos and updated frontier into the next generation’s *agenda* of continue, stop, validate, and explore decisions. Durable cross-generation lessons are compressed into *Gems*. Together they form a lineage graph that stays active throughout the campaign and accompanies the final artifact as a record of its formation.

We evaluate PRAXIST on all 75 MLE-bench tasks against a local Claude Code baseline, earning 60 medals (80.0%) and 49 golds against 55 (73.3%) and 34 for Claude Code on Opus 4.8. In four open-ended case studies—quantitative trading, LiDAR-inertial-visual SLAM, tokamak magnetic control, and rocket landing—it improves the task-native baseline’s headline metric in two domains, leads on survival but not on full-horizon precision in tokamak control, and, in the fourth, cuts the baseline’s visual-compute cost at unchanged accuracy, while the accompanying lineages document mechanism discovery, validation, recombination, and boundary testing, so that each reported gain arrives with an auditable account of how it was reached.

Our contributions are:

- We formulate *evidence inheritance* as a systems requirement for long-horizon evaluator-grounded R&D: prior evidence must be selectively retained and recombined rather than merely stored, as assembly theory describes objects built from reusable parts under selection.
- We develop PRAXIST, a generational system that converts evaluated artifacts into typed, inheritable evidence—findings, frontier lanes, agendas, and Gems—and keeps the resulting lineage graph active during research and inspectable afterwards.
- We evaluate PRAXIST on all 75 MLE-bench tasks and four open-ended case studies, scoring outcomes, lineage process measures, and cost against local baselines.

2 Method

2.1 Overview and Formalization

PRAXIST runs autonomous R&D as an *artifact-to-lineage* process. Every attempt is materialized as a reproducible *artifact*; an evaluated artifact is interpreted into one or more *findings*; findings are promoted onto a *frontier* of inheritable evidence; the frontier is synthesized into an *agenda* that directs the next round of attempts; and a campaign concludes by reporting a final artifact together with the *lineage* records of how it was produced:

$$\text{Artifact} \rightarrow \text{Finding} \rightarrow \text{Frontier} \rightarrow \text{Agenda} \rightarrow \text{Lineage}.$$

The central design choice is that evidence is never inherited as a raw transcript or a scalar score. Before any evidence is allowed to influence future work, PRAXIST assigns it an explicit *operational role*: a mature parent to build on, a fragile candidate to validate, a failure to avoid, or a lesson to remember. Systems that carry state as score-ranked artifact collections—evolving program databases, solution trees—inherit an untyped object^{4,5,6,1,2}; PRAXIST inherits typed findings whose frontier lanes make validation status explicit, agenda decisions that assign each element an operational role, and, where enabled, Gems that compress durable lessons across generations.

Problem formalization. We formalize PRAXIST as a generational state-transition process. Let $g = 0, 1, \dots, G-1$ index generations. A generation is the synchronization boundary at which peer-level evidence becomes shared research state. The state inherited into generation g is

$$\mathcal{S}_g = (\mathcal{F}_g, \mathcal{A}_g, \mathcal{G}_g, \mathcal{L}_g), \quad (1)$$

where $\mathcal{F}_g$ is the frontier, $\mathcal{A}_g$ is the agenda for generation g , $\mathcal{G}_g$ is the set of Gems (durable cross-generation lessons), and $\mathcal{L}_g$ is the lineage trace accumulated so far. Table 1 collects the notation. Each generation reads $\mathcal{S}_g$, executes a cohort of C parallel *peers*, and emits the successor state $\mathcal{S}_{g+1}$; Algorithm 1 states the full cycle depicted in Figure 1, with the peer-owned span—peers through findings—bracketed as *local experimentation* in the figure (design allocation is itself a cohort-level decision). Full field-level schemas for every object are collected in Appendix A and referenced by compact pointers below; the main text remains self-contained. One convention applies throughout: the vocabularies named below—finding types, inheritance actions, frontier lanes, evidence stages, and agenda dispositions—are the *method-level* roles that inheritance is defined over, not the literal field values of any one deployment. The implementation exposes a task-facing enum for finding records and lets each task realization declare its own frontier lanes—or none—and its own evidence stages, which the campaign then maps onto these roles; Appendix A records that correspondence and where a task’s own labels are the operative ones.

One generation at a glance. Figure 1 shows a single generation end to end, and the rest of this section follows it box by box. The **top row** is *local experimentation*: it turns inherited evidence into new evaluated evidence. A generation starts from the *inherited state* on the left—the *frontier* of evidence eligible to influence this generation, the *agenda* that says what this generation should do, and the *Gems* that carry durable lessons across many generations. *Design allocation* (§2.2) turns that state into one design contract per peer and deliberately spreads the cohort across distinct design cells—different mechanism families, intervention surfaces, and intents—so that a generation tests several hypotheses rather than crowding the one direction that currently looks best. The *peers* are C autonomous experimental workers that run concurrently, each owning a single contract. Each peer

Table 1: Notation used in the method; lane names and vocabularies are those of the configuration used in this work. Detailed schemas are given in Appendix A.

Symbol	Meaning
g, G	generation index and number of generations
$\mathcal{S}_g = (\mathcal{F}_g, \mathcal{A}_g, \mathcal{G}_g, \mathcal{L}_g)$	research state inherited into generation g
C	cohort size: number of parallel peers per generation
$p_i, i \in \{1, \dots, C\}$	the i -th peer of a generation
d_i	design contract assigned to peer p_i
$c_i \in \mathcal{C}$	design cell occupied by d_i in the behavior space $\mathcal{C}$
a_i	reproducible artifact constructed by peer p_i
$e_i = \text{EVAL}(a_i)$	external-evaluator outcome for artifact a_i
ϕ	a finding; $\Phi_g = \bigcup_i \Phi_i$ the generation’s findings
$\mathcal{F} = \mathcal{F}^{\text{cf}} \cup \mathcal{F}^{\text{cd}} \cup \mathcal{F}^{\text{dg}} \cup \mathcal{F}^{\text{vl}}$	lanes used here: confirmed, candidate, diagnostic, validation
μ_r	memo emitted by PI role r
$\mathcal{A}$	agenda: per-direction directives and peer contracts
$\mathcal{G}$	Gems: durable compressed lessons
ρ	compression period, where Gem compression is enabled
$\mathcal{L}$	lineage trace: typed records of research events
a^*	final reported artifact

builds a *reproducible artifact*, the self-contained unit of work that makes an attempt re-runnable and inspectable, and an *external evaluator*—the task’s own metric, never the agent’s self-assessment—scores it. The evaluated artifact is then interpreted *against the contract that produced it* into typed *findings* (§2.3), so that a result is recorded as an artifact-grounded claim with a status (positive, negative, diagnostic, uncertain, procedural) rather than as a number. The arrow returning across the top of the row is the peer’s own inner loop: within a generation, a peer may revise its artifact against its own evaluated findings before the generation boundary.

The **bottom row** is *global synthesis and inheritance*: it decides what the run keeps. Reading right to left, a panel of *PI* roles and a *Chair* (§2.4) interpret the generation’s pooled findings from complementary perspectives and arbitrate between them. The *frontier update* then sorts the surviving evidence into inheritance lanes—confirmed, candidate, diagnostic, and validation in the configuration used here—so that a later generation inherits not only what was measured but how far it can be trusted and what should be done with it. The *next agenda* converts that into a policy for the following generation, issuing per-direction verdicts to continue, validate, stop, or explore. Where memory compression is enabled, every ρ generations it distills recurring lessons into Gems, keeping long runs within a bounded context. Together, the updated frontier, agenda, and Gems form the state inherited by generation $g+1$, closing the loop back to the left of the figure. Running beneath both rows, the *lineage trace* (§2.5) records research objects as they are produced—artifacts, findings, decisions, agendas, memory—and the typed relations between them; it is the run’s second deliverable, reported alongside the final artifact as an account of how the outcome was reached.

2.2 Inherited State and Design Allocation

Inherited state. Each generation begins from the inherited state $\mathcal{S}_g$ of Eq. 1 (leftmost box of Figure 1), which supplies three kinds of prior evidence. The *frontier* $\mathcal{F}_g$ is the inheritable evidence

Algorithm 1 PRAXIST generational research cycle. Section references point to the paragraph that defines each operator; appendix pointers for field-level schemas appear in those paragraphs.

Input: task with external evaluator $\text{EVAL}(\cdot)$; cohort size C ; generations G ; compression period ρ if Gems are enabled.

Init: $\mathcal{F}_0 \leftarrow \emptyset$, $\mathcal{A}_0 \leftarrow \emptyset$, $\mathcal{G}_0 \leftarrow \emptyset$, $\mathcal{L}_0 \leftarrow \emptyset$.

for $g = 0$ **to** $G - 1$ **do**

$\{d_i\}_{i=1}^C \leftarrow \text{ALLOCATE}(\mathcal{A}_g, \mathcal{F}_g, \mathcal{G}_g)$ // design allocation: one design cell c_i per peer (§2.2)

parallel for $i = 1$ **to** C **do** // local experimentation

$a_i \leftarrow \text{BUILD}(d_i, \mathcal{S}_g)$ // peer constructs a reproducible artifact (§2.3)

$e_i \leftarrow \text{EVAL}(a_i)$ // external evaluation (§2.3)

$\Phi_i \leftarrow \text{INTERPRET}(a_i, e_i, d_i)$ // artifact-to-finding extraction (§2.3)

$\Phi_g \leftarrow \bigcup_{i=1}^C \Phi_i$

$\{\mu_r\}_r \leftarrow \{\text{PI}_r(\Phi_g, \mathcal{F}_g)\}_r$ // PI panel memos on the generation’s findings (§2.4)

$\mathcal{F}_{g+1} \leftarrow \text{PROMOTE}(\mathcal{F}_g, \Phi_g, \{\mu_r\}_r)$ // frontier update into the task’s lanes (§2.4)

$\mathcal{A}_{g+1} \leftarrow \text{CHAIR}(\{\mu_r\}_r, \mathcal{F}_{g+1})$ // Chair arbitration $\rightarrow$ next agenda (§2.4)

if $g \bmod \rho = \rho - 1$ **then** $\mathcal{G}_{g+1} \leftarrow \text{COMPRESS}(\mathcal{F}_{g+1}, \mathcal{G}_g)$ **else** $\mathcal{G}_{g+1} \leftarrow \mathcal{G}_g$ // Gems, only if enabled (§2.5)

$\mathcal{L}_{g+1} \leftarrow \mathcal{L}_g \cup \text{RECORD}(\{a_i\}, \Phi_g, \mathcal{F}_{g+1}, \mathcal{A}_{g+1}, \mathcal{G}_{g+1})$ // lineage trace (§2.5)

write out $\mathcal{F}_G, \mathcal{L}_G$, run summary; **report** $a^* \leftarrow \arg \max_{a \in \text{art}(\mathcal{F}^{\text{cf}})} \text{SCORE}(a)$ with $\mathcal{L}_G$ // selected at reporting time (§2.5)

eligible to influence this generation; the *agenda* $\mathcal{A}_g$ is the generation policy produced by the previous synthesis step; and the *Gems* $\mathcal{G}_g$ are durable lessons retained across many generations. Because a long run produces far more evidence than any single prompt context can hold, inheritance is *selective*: a peer receives only the frontier entries, Gems, and parent lineage relevant to its assigned direction, the PI panel receives generation-level evidence, and the Chair receives the panel’s memos (Appendix A.6). The unit that grounds all of this evidence is the *reproducible artifact*, defined next.

Reproducible artifacts. The **reproducible artifact** a is the minimal file set required to reproduce, inspect, or evaluate one attempt. In the machine-learning-engineering (MLE) task family used as our running example⁸, an artifact comprises the submission file, the code that produced it, and any supporting files needed for reproduction. Artifacts are what ground evaluator outcomes and downstream research claims in concrete experimental products, mirroring provenance models that connect entities, activities, and derived results³⁶ and ML experiment-tracking systems that record code, parameters, and metrics^{38,39} (artifact schema and status labels: Appendix A.2).

Deep Innovation Gate (DIG). Before code is written, PRAXIST applies a pre-artifact *innovation gate* that makes exploration intentional (second box of Figure 1). Its scope is configurable: by default the gate runs as the campaign’s opening allocation, and a task may enable it for every generation. The rocket campaign reported here uses the default, so its systematic pre-artifact contracts belong to the first generation, after which the Chair’s agenda carries the same fields forward. The allocation operator $\{d_1, \dots, d_C\} = \text{ALLOCATE}(\mathcal{A}_g, \mathcal{F}_g, \mathcal{G}_g)$ returns one **design contract** d_i for each peer p_i . Its peer-level gate is the **Deep Innovation Gate (DIG)**: DIG requires a peer to inspect the inherited state, identify a testable mechanism, and fix the intended intervention before construction begins. The resulting contract specifies the mechanism family being tested, the intervention surface it changes, the parent lineage it extends, the evidence signature that would support or weaken it, the validation or ablation hook that makes the result interpretable, and the forbidden changes that

would invalidate the test. DIG is read-only: it does not run experiments, create variants, or write result artifacts. Its purpose is to make the later artifact interpretable by letting finding extraction compare the *intended* experiment against the artifact’s *observed* behavior. The full contract field list appears in Appendix A.3 (Table 13).

Quantified Diversity (QD). The cohort-level component of ALLOCATE is **Quantified Diversity (QD)**. Each contract d_i occupies a **design cell** c_i in a behavior space $\mathcal{C}$, where a cell is the tuple

$$c = (\text{mechanism family, intervention surface, intent}), \quad (2)$$

and QD distributes the C contracts to *cover* distinct cells rather than concentrating effort on one apparent direction. Diversity is therefore represented by explicit cell coordinates and controlled by caps over mechanism family, intervention surface, and intent. The caps are hard constraints in the opening allocation, where a greedy allocator enforces them; in later generations the Chair plans the same cell coordinates under the same caps as soft targets, and the agenda validator warns on missing or unknown planning dimensions rather than rejecting the agenda. This mechanism is related to quality-diversity search, which maintains diverse high-quality candidates across a feature space^{33,34,35}, but PRAXIST applies the idea to research design rather than to final artifacts. Within one generation, QD can allocate peers to exploit a confirmed direction, validate a fragile candidate, diagnose a failure mode, and explore a distinct mechanism family (the four design foci “mechanism, surface, validation, diagnosis” shown in the figure). The allocation rule and its caps are detailed in Appendix A.3.

2.3 Artifact Construction, Evaluation, and Findings

Peers and artifact construction. A **peer** p_i is one parallel experimental worker: an autonomous agent that owns a single design contract d_i and drives it to an evaluated artifact (third and fourth boxes of Figure 1). Given its contract and the selectively inherited state, peer p_i constructs an artifact $a_i = \text{BUILD}(d_i, \mathcal{S}_g)$, iterating internally—writing code, running experiments, and revising—until the artifact reaches a reportable stage (the return arrow across the top of Figure 1). The C peers of a generation run concurrently and coordinate only through published evidence, never by editing shared state directly; this separates local experimentation from the global allocation and promotion decisions that occur at the generation boundary. Storing and reusing experience in this way follows the broader view of agents that persist, synthesize, and retrieve memory across tasks^{23,24}. Peer experiments are executed on shared hardware by a thin resource-scheduling layer that paces launches so that each generation completes enough mature evidence before its wall-clock horizon; Appendix A.8 gives its specification and the supporting paired event simulation.

External evaluation. Each artifact is scored by the task’s **external evaluator**, $e_i = \text{EVAL}(a_i)$, which returns a task-grounded outcome together with a validity state and an *evidence stage*. The evidence stage records how much of the evaluation an artifact has passed, on an ordered ladder from cheap sanity checks (*smoke*), through partial probes (*scout*), to a full scored evaluation and its replications. The core distinguishes those three levels generically; the names of the intermediate rungs are task-owned, so the rocket evaluator reports *canary*, *development*, and *complete* where the trading evaluator reports *smoke*, *scout*, and three replication tiers. Evidence stage is deliberately distinct from score: a high score obtained only at a preliminary stage is treated as immature and cannot crowd out a complete result during promotion (evaluator interface, validity states, and the exact evidence-stage ladder: Appendix A.2).

Findings. An evaluated artifact is not yet reusable knowledge; the **finding** is the reusable unit (rightmost box of the top row). The extraction operator $\Phi_i = \text{INTERPRET}(a_i, e_i, d_i)$ converts each artifact, its outcome, and its intended design into one or more artifact-grounded claims. A finding is the tuple

$$\phi = (\text{intervention}, \text{outcome}, \text{evidence}, \tau, m, \alpha), \quad (3)$$

recording what was attempted, what happened, the evidence, and three labels that govern inheritance: the finding *type* τ , the evidence *maturity* m , and the recommended inheritance *action* α . The type takes one of five values: *positive* (an improvement or reusable mechanism), *negative* (a weakened assumption or intervention), *diagnostic* (a constraint, failure mode, or invalidity condition), *uncertain* (evidence worth keeping pending further checks), or *procedural* (a constraint on how future experiments must run). The maturity m inherits the artifact’s evidence stage, so interpretation confidence stays tied to how thoroughly it was evaluated. The action $\alpha \in \{\text{reuse}, \text{validate}, \text{avoid}, \text{diagnose}, \text{preserve}, \text{archive}\}$ is a recommendation for what the next generation should do with it. Both τ and α name the operational roles the method is defined over; the implementation files a finding under a task-facing record enum (`result`, `hypothesis`, `insight`, `challenge`, `error`) and carries the inheritance recommendation in next-step-intent metadata rather than in a single α column (field and vocabulary tables: Appendix A.2).

Failures as first-class evidence. Negative and diagnostic findings are first-class evidence, not discarded failures. A failed artifact can reveal an invalid assumption, an evaluator constraint, a data-processing error, or a fragile score, and encoding this as a finding lets later generations inherit the failure as a constraint rather than rediscovering it independently. This treatment of feedback as persistent, reusable state generalizes verbal-reflection and self-refinement mechanisms in language agents^{21,22} from within-episode iteration to cross-generation inheritance. The pooled findings of the generation are $\Phi_g = \bigcup_{i=1}^C \Phi_i$.

2.4 Global Synthesis: Frontier Update and Next Agenda

PI/Chair synthesis. The bottom row of Figure 1 turns the pooled findings Φ_g into the next generation’s inheritable state. It begins with **PI/Chair synthesis**, which converts peer-local evidence into generation-level research policy. A panel of *Principal-Investigator* (PI) roles independently interprets Φ_g from complementary perspectives—a *Builder* that assembles the strongest evidence-backed mainline, a *Skeptic* that audits unsupported or fragile claims, and a *Portfolio* role that balances effort across approach families; in *high-stakes mode*, used when final claims or reproducibility risks require extra scrutiny, the panel adds an *External-validity* role that checks reproducibility and evidence boundaries—each role emitting a memo $\mu_r = \text{PI}_r(\Phi_g, \mathcal{F}_g)$, after which a *Chair* merges the memos with the updated frontier into a single agenda, $\mathcal{A}_{g+1} = \text{CHAIR}(\{\mu_r\}_r, \mathcal{F}_{g+1})$. Role specialization of this kind is widely used in multi-agent LLM collaboration^{25,26,27}; PRAXIST applies it specifically to research governance, so that scores, reliability, and research utility are weighed against one another before inheritance. The panel proceeds through an evidence-freeze step, parallel independent memos, and an anonymized cross-review, after which the Chair arbitrates (role definitions, panel modes, and round structure: Appendix A.4).

Frontier update. Synthesis drives the **frontier update** (the “Frontier Update” box of Figure 1), the operation $\mathcal{F}_{g+1} = \text{PROMOTE}(\mathcal{F}_g, \Phi_g, \{\mu_r\}_r)$ that decides which evidence becomes inheritable

and in what capacity. In the configuration used throughout this work, the frontier is partitioned into four *lanes* by operational role,

$$\mathcal{F} = \mathcal{F}^{\text{cf}} \cup \mathcal{F}^{\text{cd}} \cup \mathcal{F}^{\text{dg}} \cup \mathcal{F}^{\text{vl}}, \quad (4)$$

namely *confirmed* ($\mathcal{F}^{\text{cf}}$): mature enough to serve as a parent or constraint; *candidate* ($\mathcal{F}^{\text{cd}}$): promising but immature evidence; *diagnostic* ($\mathcal{F}^{\text{dg}}$): failures, controls, and failure modes that shape exploration; and *validation* ($\mathcal{F}^{\text{vl}}$): evidence scheduled for reproduction, ablation, or checking before promotion. Separating the frontier this way lets PRAXIST inherit not just the best score but also reliability, diagnostic value, and validation priority. Lanes are task-defined rather than built into the system: a task realization declares whichever lanes it needs—the trading campaign uses *confirmed alpha*, *alpha incubator*, *benchmark floor*, and *diagnostic control*—and a task that declares none falls back to a single-metric frontier. Promotion admits, per lane, the top findings under that lane’s criterion, gating out immature evidence (smoke/scout stages) and non-promotable results (promotion criteria, per-lane caps, and gating rules: Appendix A.5).

Next agenda. The frontier and the panel memos yield the **next agenda** $\mathcal{A}_{g+1}$ (the “Next Agenda” box), the control object for the next generation. It labels each inherited direction with one of four dispositions—*continue* a confirmed direction, *stop* a weakened one, *validate* a fragile candidate, or *explore* an underexplored cell—and attaches, for each of the C peers, a *peer contract* that specifies its role, hypothesis, success signal, and forbidden actions. As with the finding vocabulary, the four dispositions are method-level roles; the emitted agenda records them through a finer next-step-intent field that distinguishes, for example, repairing a failure mode from ablating one. ALLOCATE (§2.2) consumes the agenda at the start of generation $g+1$, closing the loop between global synthesis and local experimentation (agenda schema and lane-to-disposition mapping: Appendix A.5).

2.5 Memory, Lineage, and Final Output

Memory compression and Gems. Frontier entries carry the operational status of *current* evidence, but a long campaign also accumulates lessons that should survive after their originating evidence is archived. Where a campaign enables it, every ρ generations PRAXIST performs a **memory compression** $\mathcal{G}_{g+1} = \text{COMPRESS}(\mathcal{F}_{g+1}, \mathcal{G}_g)$ that distills the balanced frontier into **Gems** (the “Memory Compression” box). A Gem is a compact, durable lesson—a validated mechanism, a rejected assumption, a recurring failure mode, or a procedural constraint—retained across the reset boundary so that later peers keep the benefit of past evidence without its bulk. Compression is lane-balanced (preserving control and diagnostic lessons, not only high scorers) and bounded (only a small active Gem set is kept). The facility is off by default: among the case studies reported here only the trading campaign runs it, with a period of six generations and at most four active Gems. This extends experience-storage-and-retrieval memory patterns^{23,24,21} to evaluator-grounded campaigns (compression trigger, selection policy, and Gem schema: Appendix A.6).

Lineage trace. Throughout the cycle, research events are appended to the **lineage trace** (the “Lineage Trace” band along the bottom of Figure 1):

$$\mathcal{L}_{g+1} = \mathcal{L}_g \cup \text{RECORD}(\{a_i\}, \Phi_g, \mathcal{F}_{g+1}, \mathcal{A}_{g+1}, \mathcal{G}_{g+1}).$$

The lineage records the research objects defined above—artifacts, findings, synthesis decisions, agendas, and memory (Gem) updates—together with typed relations among them such as *derived-from*, *supports*, *challenges*, and *updates*. It is materialized as several correlated ledgers that share

identifiers—an edge list over findings, an artifact index, an event trajectory, and per-generation frontier, agenda, and Gem state—rather than as one object graph, and a chain is traversed by following those shared identifiers across them. It is accumulated *during* the run rather than reconstructed after the fact: each artifact points to the parents, contract, and findings it descended from, and each promotion or agenda decision records why evidence became inheritable. During execution the lineage determines what later peers inherit; after execution it is the structured account of the research trajectory (node and edge vocabulary: Appendix A.7).

Final output. After G generations, the outcome of a PRAXIST campaign is reported as a final artifact together with its lineage,

$$a^* = \arg \max_{a \in \text{art}(\mathcal{F}^{\text{cf}})} \text{SCORE}(a), \quad \text{output} = (a^*, \mathcal{L}_G), \quad (5)$$

where $\text{art}(\mathcal{F}^{\text{cf}})$ denotes the artifacts that ground the confirmed-lane findings: under the default rule the final artifact is the best-scoring member of the confirmed frontier. Eq. 5 states that rule, not an automatic runtime call. The loop itself terminates by writing out the frontier, the lineage ledgers, and a run summary; the selection is applied over those outputs at reporting time, and each case study below states which artifact it selected and under which rule—the Rocket and Quant studies report an artifact selected post-run from outside the confirmed lane, and say so where they report it. The output is thus not only an evaluated artifact but the *solution lineage* that explains how it was obtained—which evidence supported it, which failures constrained it, which candidates required validation, and which agendas shaped its construction—providing reusable state for future researchers or autonomous systems.

3 Experiments

3.1 Overview

Our evaluation has two parts. The first is a large-scale comparative benchmark: we run PRAXIST on MLE-bench⁸, a suite of 75 Kaggle-derived machine-learning-engineering competitions with an external grading harness, and compare it head-to-head against a Claude Code baseline that we run ourselves under an identical protocol (Section 3.2). MLE-bench is our primary evaluation because it is the setting in which directly comparable cross-system claims are possible: every system faces the same tasks, the same data, and the same medal thresholds.

The second part probes generality beyond competition-style ML engineering through four in-depth case studies on harder, more open-ended R&D problems: rocket design (Section 3.3), quantitative trading (Section 3.4), LiDAR-inertial-visual SLAM (Section 3.5), and tokamak magnetic control for fusion (Section 3.6). We emphasize that the case studies are not standardized cross-system comparisons in the sense of MLE-bench: each is scored by its own domain-native evaluator against a task-grounded baseline, and we analyze both the resulting artifact and its discovery process. The Rocket study additionally reports an external autonomous code optimizer run on the same task and the same success metric, an outside reference point rather than a matched-model or matched-compute comparison. In the Rocket study, PRAXIST constructs a deterministic first-contact landing controller that reaches 12,288/12,288 (100%) on the frozen protocol, against 17.12% for Weco, an autonomous code optimizer run on the same task and objective from the same 4.03% starting point; a separate post-run all-row audit of the three fixed source banks records 40,959, 40,959, and 40,960

successes out of 40,960 rows per bank. In the Quant study, it discovers a recurrent, execution-aware trading policy whose 53% walk-forward CAGR is 2.3 times the 23% of its paired all-eligible equal-weight baseline. In the SLAM study, it shows that state-of-the-art LIVO systems over-spend visual computation on redundant frames and repeated map observations. Replacing the uniform visual-update policy with an observability-aware scheduler and a geometry-aware map-admission gate cuts the evaluator-captured visual-path processing time by roughly 72% across fourteen sequences at no cost in trajectory accuracy; the two arms stamp their poses differently, so Section 3.5 reports the accuracy columns side by side rather than as a gain. In the Fusion study, it synthesizes a tokamak magnetic-control law that achieves higher aggregate survival and lower common-horizon tracking error than a task-native controller based on the plasma control system deployed on the MAST-U device, while that baseline keeps the edge on the benchmark’s original full-horizon metric and on completion rate. Appendix B records the execution layer for all five studies: the repetitions behind each headline number, the hardware/time envelope, and the PRAXIST campaign setup.

3.2 MLE-bench

3.2.1 Experimental Setup

Benchmark. MLE-bench⁸ consists of 75 Kaggle competitions spanning tabular modeling, computer vision, natural-language processing, time-series forecasting, and signal-processing tasks. An agent must produce a valid submission file for each competition; submissions are scored by the official grading harness against the competition’s private leaderboard, and a run earns a bronze, silver, or gold medal by clearing the corresponding human-leaderboard threshold. Tasks are stratified by complexity into a *Low* tier (22 tasks; this tier constitutes the MLE-bench Lite subset), a *Medium* tier (38 tasks), and a *High* tier (15 tasks). We evaluate on the full 75-task set and report results per tier. Medals are determined by the official MLE-bench grading harness from each submission’s private-leaderboard score, and a task contributes a medal only if its recorded score clears the corresponding MLE-bench threshold. Entries without a benchmark-available final score are kept in the denominator and counted as no medal. Every row we report is a local 75-task evaluation graded by that same harness on the same hardware pool, which makes the rows directly comparable to one another; we therefore do not place them beside the public MLE-bench leaderboard, whose entries are produced under different run conventions and reported as multi-seed run-group averages. Each arm is one full 75-task sweep, so the medal rates are single-sweep outcomes rather than variance estimates over repeated sweeps. These results nevertheless cover all 75 benchmark competitions, providing broad evidence about cross-task reliability under the stated reporting protocol.

Setup. We report one finalized result per task from the release ledger. That ledger records an explicit integrity adjudication for each of the 75 tasks: every reported PRAXIST result is the highest-scoring official attempt that passes the adjudication, with the selected submission verified by SHA-256 against the run journal, and attempts drawn from a contaminated lineage are excluded outright rather than down-weighted (90,423 attempts were rejected across the campaign). On 9 of the 75 tasks the whole lineage behind the previously best attempt was excluded and a clean actor’s submission was substituted in its place, which lowers the medal on four of those nine, and 6 further tasks were re-reviewed and cleared; Appendix C reports the per-task statuses. This adjudication is internal to the PRAXIST ledger and is separate from the benchmark’s own post-run screen for severe-cheating violations, which applies to the Claude Code baseline described below. PRAXIST runs deepseek-v4-pro with a 1M-token context window as its research-agent model on all 75 competitions and evaluates each competition using the procedure described in Section 2. We compare

Table 2: MLE-bench Any Medal rate (%) by complexity tier and base LLM, with all-task medal composition. Bold marks the best value in each column.

Agent	Base LLM	Any Medal (%)				All-task G/S/B
		Low (22)	Medium (38)	High (15)	All (75)	
Claude Code	Claude Opus 4.8	81.8	76.3	53.3	73.3	34/16/5
PRAXIST (ours)	deepseek-v4-pro	90.9	81.6	60.0	80.0	49/10/1

it against a Claude Code baseline swept over all 75 tasks locally under the same grading harness. The baseline runs **Claude Opus 4.8** with maximum thinking budget; after the benchmark’s post-run screen for severe-cheating violations, its finalized ledger retains 70 accepted scores, marks five tasks not scored, and records a dollar-equivalent cost of US\$38,370 (Appendix C gives its per-task results). Both arms ran on the same pool of H100 80GB GPUs, with each experiment scheduled as a single-GPU job and up to eight experiments in flight per task; 70 tasks ran under a 24-hour per-task wall cap and the remaining five under a 36-hour cap. Appendix A.9 reports the generation, cohort, and budget settings of the single PRAXIST campaign, and Appendix B.1 records the execution environment and the campaign totals behind the finalized ledger.

3.2.2 Main Results

Table 2 reports the base LLM for each arm, Any Medal rates per complexity tier, and all-task medal composition. The audited Claude Code + Opus 4.8 sweep records 55 accepted medals across the 75-task suite (73.3%), with five tasks not scored. PRAXIST exceeds the Opus 4.8 sweep on every tier—90.9% versus 81.8% on Low, 81.6% versus 76.3% on Medium, and 60.0% versus 53.3% on High—and has the larger gold count in every tier, 49 versus 34 overall, with 19 of its 20 Low-tier medals gold. The available dollar-equivalent ledgers add a separate efficiency contrast: the recorded PRAXIST 75-task spend is approximately US\$3,054, whereas the Opus 4.8 sweep records US\$38,370, roughly an order of magnitude apart on the same task suite.

The medal composition is the most distinctive feature of the PRAXIST result. Of PRAXIST’s 60 medals, 49 (81.7%) are gold, compared with 34 of 55 accepted medals (61.8%) for Claude Code on Opus 4.8. Because both arms are single locally measured runs rather than official multi-seed leaderboard rows, the margins should be interpreted as benchmark-wide outcomes under the stated reporting protocol rather than seed-averaged estimates of a distribution over repeated runs.

3.2.3 Analysis by Task Category

Beyond the per-tier aggregates, we compare PRAXIST head-to-head against the Claude Code + Opus 4.8 baseline competition by competition. Table 3 shows this comparison on 19 competitions drawn from the Medium- and High-complexity tiers, where several category-level differences are visible. The selection spans both categories where PRAXIST improves the medal outcome and categories where Claude Code has the stronger score; the complete 75-task comparison, including the Low tier, is given in Appendix C.

Where PRAXIST improves the medal outcome. In the selected 19-task subset, PRAXIST records the better score on 12 tasks and the stronger gold count: 11 gold outcomes versus 5 for Claude Code. Each arm medals on 14 of the 19, but the medal is strictly better for PRAXIST on eight tasks and strictly better for the baseline on four, so the difference is concentrated in medal grade

Table 3: Per-task comparison of the Claude Code + Opus 4.8 baseline and PRAXIST on 19 representative Medium- and High-tier MLE-bench competitions. Shading encodes the medal (gold, silver, bronze; unshaded = none); bold marks the better score in the metric’s direction. The # column indexes the full 75-task listing of Appendix C.

Category	#	Task	Metric	Claude Code	PRAXIST (ours)
Image Classification	2	alaska2-image-steganalysis	weighted AUROC $\uparrow$	0.9260	0.9140
	4	cassava-leaf-disease-classification	accuracy $\uparrow$	0.9010	0.9021
	17	kuzushiji-recognition	F1 $\uparrow$	0.9714	0.9741
	23	rsna-breast-cancer-detection	probabilistic F1 $\uparrow$	0.4719	0.2752
	25	statoil-iceberg-classifier-challenge	log loss $\downarrow$	0.1451	0.1271
Image (Other)	40	petfinder-pawpularity-score	RMSE $\downarrow$	16.77	16.86
	41	rsna-miccai-brain-tumor-radiogenomic-classification	AUROC $\uparrow$	0.5341	0.6588
Text Classification	29	AI4Code	Kendall τ $\uparrow$	0.8606	0.8227
	31	facebook-recruiting-iii-keyword-extraction	micro F1 $\uparrow$	0.7851	0.7959
	33	jigsaw-unintended-bias-in-toxicity-classification	bias-weighted AUC $\uparrow$	0.8635	0.8640
	34	learning-agency-lab-automated-essay-scoring-2	quadratic kappa $\uparrow$	0.8312	0.8384
	35	lmsys-chatbot-arena	log loss $\downarrow$	0.8718	0.9812
	38	tweet-sentiment-extraction	Jaccard $\uparrow$	0.7225	0.7239
Training LLMs	58	chaii-hindi-and-tamil-question-answering	word Jaccard $\uparrow$	0.7585	0.8894
Forecasting	60	osic-pulmonary-fibrosis-progression	Laplace log-lik. $\uparrow$	-7.036	-5.6407
Tabular	48	icecube-neutrinos-in-deep-ice	angular error $\downarrow$	1.005	1.0563
Image Segmentation	64	hubmap-kidney-segmentation	Dice $\uparrow$	0.9487	0.9488
	67	uw-madison-gi-tract-image-segmentation	Dice-Hausdorff $\uparrow$	0.6274	0.8722
Audio Classification	42	freesound-audio-tagging-2019	LRAP $\uparrow$	0.7471	0.7445

rather than medal count. The clearest gains are conversions from no medal at all: brain-tumor radiogenomic classification (#41) moves from the baseline’s 0.5341 AUROC to 0.6588, automated essay scoring (#34) from 0.8312 to 0.8384 quadratic kappa, and pulmonary-fibrosis progression (#60) from -7.04 to -5.64 Laplace log-likelihood, each crossing from no medal to gold. A second family upgrades a baseline that already medals: on cassava leaf disease (#4), keyword extraction (#31), and Hindi/Tamil question answering (#58, where word Jaccard rises from 0.7585 to 0.8894), Claude Code earns silver and PRAXIST reaches gold, while GI-tract segmentation (#67) rises from no medal to silver and iceberg classification (#25) from bronze to silver. This is the per-task analogue of the medal-composition pattern in Section 3.2.2: the frontier can retain strong parents and schedule additional attempts after a first medal-clearing artifact has been found.

Where Claude Code remains stronger. Claude Code has the better score on the remaining seven of the selected tasks, and the raw-score picture over the full suite is closer still. On the 70 tasks where both arms record a score, the baseline has the better raw score on 36 and PRAXIST on 33, with one tie; PRAXIST additionally returns a score on the five tasks the baseline leaves unscored. Its benchmark advantage therefore lies in which side of a leaderboard threshold a run lands on, not in winning the majority of head-to-head score comparisons. Four rows of Table 3 cost it a medal outright: image steganalysis (#2), notebook-cell ordering (#29), and IceCube event reconstruction (#48) each leave a baseline silver unmatched, and screening mammography (#23) falls from the baseline’s 0.4719 probabilistic F1 to 0.2752. The full table in Appendix C shows the same pattern more broadly for several large-label-space recognition tasks. These competitions are often compute- and pipeline-intensive, so fewer complete training/evaluation cycles may fit inside a fixed campaign budget. On tasks where both systems clear gold—audio tagging (#42), PetFinder Pawpularity (#40), and LMSYS Chatbot Arena (#35)—residual score differences remain informative for score-level analysis but do not change the benchmark medal outcome.

Cost accounting. For the two arms with finalized model-spend ledgers, PRAXIST records a 75-task cost of about US\$3,054, converted from CNY 20,695 using the Federal Reserve H.10 exchange rate of 6.7766 CNY per US dollar on 10 July 2026.¹ The finalized Claude Code + Opus 4.8 ledger records US\$38,370 for the same suite, twelve times the PRAXIST figure. Cost figures provide resource context only; medals in Tables 2 and 3 are determined by task scores and MLE-bench thresholds rather than cost.

3.3 Case Study: Rocket (Reusable Rocket Landing)

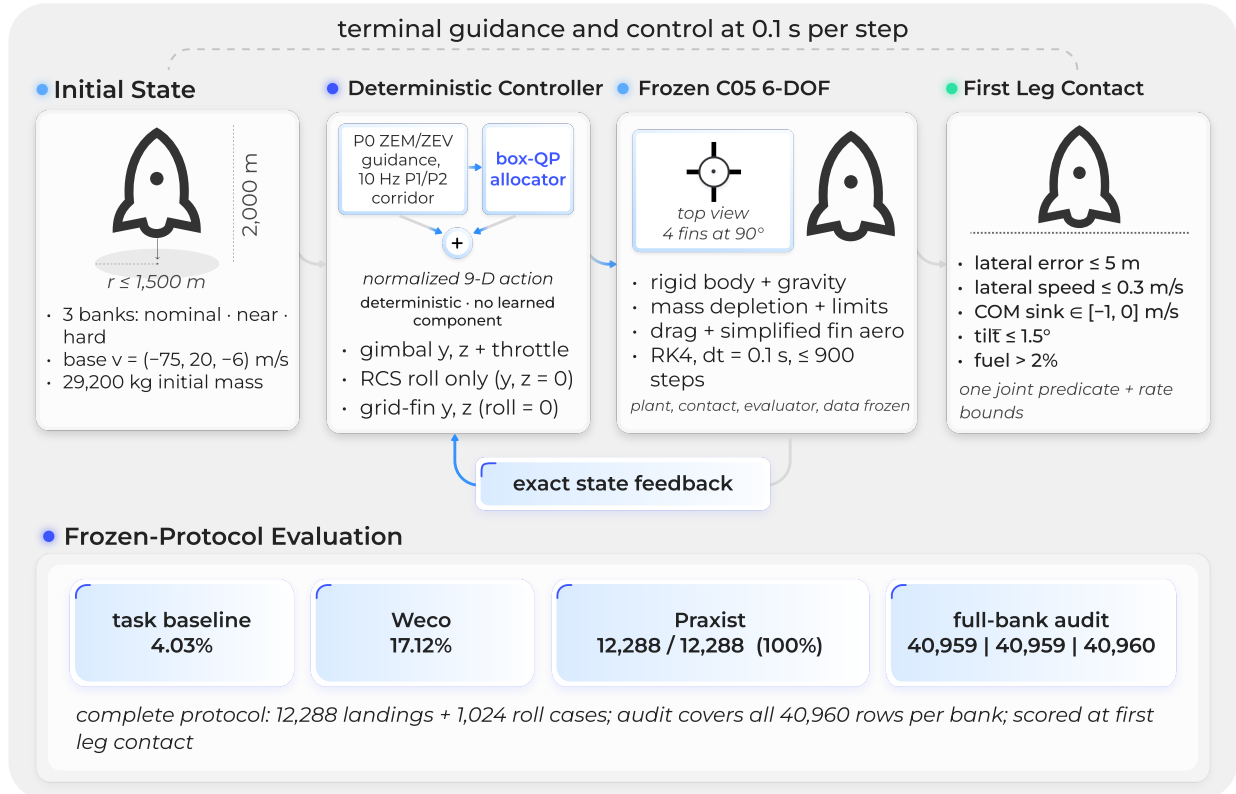


Figure 2: Rocket task overview. Initial states are drawn from three fixed banks—a nominal area-uniform disk, a row-paired slower near-OOD bank, and a faster hard-OOD outer annulus—and are flown by a deterministic hybrid controller: rolling ZEM/ZEV guidance in phase P0, fuel-commit and phase guards, and a P1/P2 terminal-descent corridor. Geometric attitude control feeds a closed-form pitch/yaw box QP that splits each torque demand between gimbal and grid fins, with the RCS restricted to roll. The frozen C05 6DoF plant advances under RK4 at 0.1-s steps, and every trajectory is scored once, at the interpolated first landing-leg contact, by a single joint success predicate. The evidence strip records results on the shared task and objective—the 4.03% starting artifact, Weco’s reported 17.12%, and the selected controller at 12,288/12,288—and, separately, its full-bank audit (40,959, 40,959, and 40,960 successes per 40,960-row bank).

Problem setup. The task is terminal guidance and control for a reusable, vertically landing rocket (Figure 2). The vehicle is the frozen Swordfish C05 six-degree-of-freedom rigid-body plant, starting at 2,000 m with 22,200 kg of dry mass and 7,000 kg of main propellant, or 29,200 kg of initial

¹https://www.federalreserve.gov/releases/h10/hist/dat00_ch.htm

total mass, and advancing at 0.1 s steps under exact state feedback. The controller commands two gimbal deflections, throttle, three reaction-control-system (RCS) channels, and three grid-fin channels, but the contract hard-locks the pitch and yaw RCS channels and the grid-fin roll channel to zero, so RCS is restricted to roll while pitch and yaw remain actuated by the gimbal and the grid fins, whose effectiveness depends strongly on geometry and flight regime⁴⁰. Plant, contact model, integrator, evaluator, and data are frozen for the whole campaign: a search may change only the controller, and cannot buy apparent gains by loosening the physics or moving the scoring endpoint. Appendix B.2 gives the full protocol.

Evaluation protocol. Initial states come from three fixed banks that share a base velocity of $(-75, 20, -6) \text{ m s}^{-1}$: a nominal area-uniform disk of radius 1,500 m, a near-OOD bank pairing those positions row by row with velocities scaled by $s \sim U[0.92, 0.98)$, and a hard-OOD bank drawing an annulus at $1,500 \leq r < 1,650 \text{ m}$ with $s \sim U[1.02, 1.08)$. The banks are fixed and adaptively reused rather than held out. Success is one joint task-native predicate evaluated at the interpolated state where the first landing leg tips the ground—bounding lateral error, sink speed, lateral speed, tilt, angular rates, and remaining propellant simultaneously—so post-contact spring-damper response cannot improve the scored state. Two evidence scopes stay separate throughout: a complete private validation of 4,096 trajectories per bank, hence 12,288 landing trajectories plus 1,024 roll cases, used for selection; and a separate post-run audit over every row of all three banks.

Baseline. Our point of comparison is another autonomous research system, not merely the artifact the task ships with. We ran Weco’s LLM-driven code optimizer—built on the AIDE tree search over code⁶—on the same task package,² with the same `landing_success_rate` objective and the same starting artifact, which scores 495/12,288 trajectories, or 4.03%, under our complete protocol. Weco evaluated 793 candidate controllers in 2 h 37 min and reached a best score of 17.12%, a $4.25\times$ relative gain that nonetheless leaves the task unsolved⁴¹. That budget is not small: its 793 evaluations amount to roughly 800 optimization steps, which in PRAXIST’s generational structure would correspond to about 100 generations running eight parallel candidate experiments each—an illustration of scale, not a description of the rocket run, which committed 12 generations of 16 peers. Recorded model spend runs the opposite way from the scores: the PRAXIST run cost US\$196.05 against US\$1,009.66, over its own 697 completed evaluation jobs, so the higher success rate was not bought with a larger budget. Table 4 places both systems against the shared starting artifact. Weco’s 17.12% is the best score reported by its run dashboard on that task and objective; PRAXIST’s 100% is measured by the frozen 12,288-trajectory protocol reported here. The systems use different underlying language models and wall-clock budgets, and Weco’s editable search surface additionally included the task-variant manifest whereas the PRAXIST run could edit the controller alone, so Weco serves as an external autonomous-optimizer reference rather than a controlled cross-model comparison.

Discovered controller. The selected artifact is deterministic throughout—no neural network, reinforcement learning, learned residual, or training seed appears anywhere in it. Rolling zero-effort-miss/zero-effort-velocity guidance⁴² sets the descent in phase P0, a fuel-commit governor and phase guards handle the transition, a terminal descent corridor carries the vehicle to first leg contact, and geometric attitude control⁴³ drives the airframe while a damped roll loop absorbs the roll channel. The newest method contribution, committed at generation 11, is a closed-form box-constrained allocator that splits each pitch/yaw torque request between gimbal and grid fins by

²https://dashboard.weco.ai/share/xZJAbpYNN7aJCvyiZuuZFsp_hzQdPqTy

solving $\min (a u_f + b u_g - d)^2 + w u_f^2$ under amplitude and rate boxes, with only the grid deflection regularized so the cost is gimbal-primary^{44,45}. It is not an iterative solver: the implementation enumerates clipped KKT candidates and returns the minimum-cost feasible point. The evaluated instance is that controller’s generation-12 configuration, byte-identical in code and differing in a single value, which is the run’s current complete-evidence champion, selected post-run rather than promoted as a committed frontier artifact.

Table 4: Rocket case study: landing success of the shared starting artifact, the Weco autonomous-optimizer baseline^{6,41}, and PRAXIST on the same task package and `landing_success_rate` objective. Relative gains are taken against the starting artifact, which is the given controller and so has no search budget of its own. Evaluation counts are each system’s own unit of work: candidate controllers scored for Weco, completed evaluation jobs for PRAXIST.

System	Landing success	Relative gain	Evaluations	Recorded spend
Starting artifact	4.03%	—	—	—
Weco	17.12%	4.25×	793	US\$1,009.66
PRAXIST	100%	24.8×	697	US\$196.05

Results. Under the frozen complete protocol, the selected controller lands 12,288/12,288 trajectories (100%), compared with the shared starting artifact’s 4.03% and Weco’s reported 17.12% on the shared task and objective, with a descriptive Wilson 95% lower bound of 99.9687% on that fixed set (Table 31). Per-bank rates rise from 5.6641%, 6.4209%, and 0% to 100% throughout, and the worst radius bin from 0% to 100%. The improvement is one of contact quality rather than contact occurrence: the baseline already reaches first contact in every trajectory, and what changes is the state in which that contact happens. The 95th-percentile sink speed falls from 66.3928 to 0.32636 m s^{−1}, lateral speed from 1.40592 to 0.04705 m s^{−1}, and tilt from 4.56062° to 0.35184°, while fuel depletion drops from 88.6393% of trajectories to none. Two metrics move adversely and we retain them: gimbal total variation rises 80.43% and roll-to-pitch/yaw coupling 87.55%, consistent with moving pitch/yaw activity onto the gimbal.

A separate post-run evaluation over all 40,960 rows of each bank records 40,959, 40,959, and 40,960 successes (Table 32). We report these per bank because the three banks are differently defined and are not one IID reliability sample. Both failures violated only the lateral-speed conjunct and lie in the [0, 450) m initial-radius bin; they are residual-motion failures rather than impacts or fuel exhaustion. The hard bank’s 100% count should not be read as broad margin, since its minimum successful fuel reserve is only 2.8908%, just 0.8908 points above the gate.

Lineage and claim boundary. The gain accumulated along the parent chain rather than arriving in one step: a fuel-commit governor and settled-release gate first, then P0 attitude authority and a low-altitude bandwidth schedule, then the initial-radius and slew branch with a sink guard, which together closed the set at 100% by generation 9, and only then the box-QP allocator at generation 11 (Table 34). This is cumulative direction-of-progress evidence, not an additive causal decomposition. The allocator’s own attribution comes from a one-key ablation: it lowers sink and lateral-speed P95 and removes the residual gimbal saturation, but its immediate parent was already at 100%, so it did not cause the success gain. That gain belongs to the accumulated lineage, whose mechanisms were established, validated, and recombined across generations. The result is bounded

by a frozen low-order simulator, by fixed banks that are adaptively reused and so carry selection-overfitting risk, by exact state feedback, and by scoring that stops at first contact; nothing is claimed about post-contact dwell, bounce, or leg loads, and the measured rates are not real-world landing reliability. Appendix B.2.1 gives the full protocol, matched results, audit, ablation, and lineage.

3.4 Case Study: Quantitative Finance (Quant)

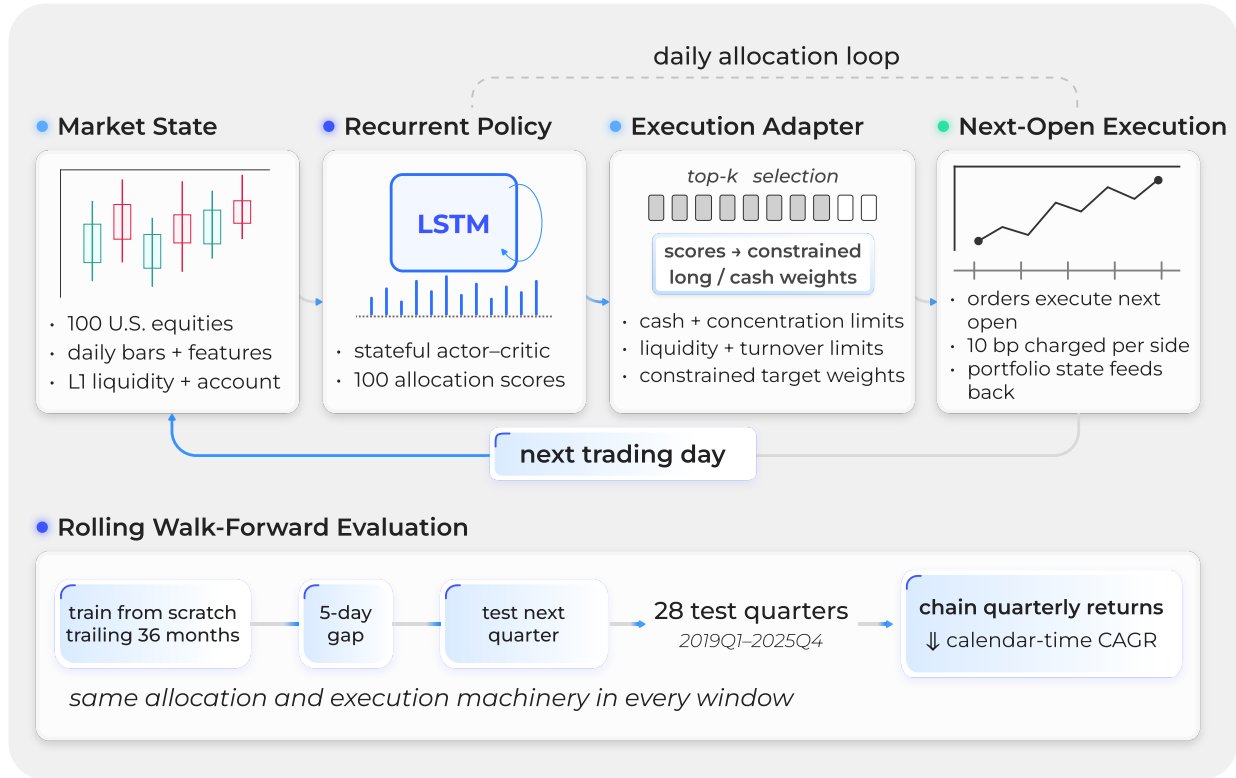


Figure 3: Quant task overview. Daily market, liquidity, and account state are mapped to 100 allocation scores; a deterministic execution adapter converts those scores into constrained long/cash weights, and orders execute at the next open with per-side costs. The evaluator retrains the policy on each trailing 36-month window, leaves a five-trading-day gap, tests the next quarter, and chains all 28 quarterly returns into calendar-time CAGR.

Problem setup. The task is daily stock allocation (Figure 3): every trading day, decide what share of the portfolio to hold in each of a fixed 100-stock, multi-sector U.S. universe (technology and semiconductors, health care, transport and logistics, materials, REITs, and large banks) and what share to leave in cash. Positions are long or flat, never short. Inputs are task-local Alpaca SIP split-adjusted daily bars, expanded per-symbol features, daily top-of-book L1 features, and the current state of the account, including its cash balance and open positions.

Evaluation protocol and baseline. A policy is never tested on data it was trained on. Evaluation walks forward through the 28 consecutive quarters from 2019Q1 to 2025Q4: for each quarter, a model is trained from scratch on the preceding 36 months, a five-trading-day gap is left between the end of that training window and the start of trading, and that quarter is then traded once. The environment also excludes target columns and rejects forward-looking feature names, so the protocol is structurally causal by construction; we did not additionally audit every engineered feature

column for leakage. Orders execute at the next day’s open with 10 basis points charged per side, which is more conservative than a frictionless close-to-close backtest. The headline metric chains all 28 quarterly returns into a calendar-time compound annual growth rate (CAGR). The **baseline** is an all-eligible equal-weight buy-and-hold comparator over the same fixed universe. In each independently cash-reset window, it invests 98% of the account equally across every protocol-eligible stock at the next open using fractional shares, charges the same 10 basis-point buy cost, and holds those shares unchanged through the final close. Chaining the 28 window endpoints gives a 22.80% paired-evaluation CAGR over the same period. Every window retrains from scratch, so a full five-seed evaluation of one candidate is 145 independent training runs; that is the campaign’s top evaluation tier, and the artifact reported below was evaluated at the first tier, a single seed over 29 cells. Appendix B.3 gives the evaluation tiers, the per-cell training cost, and the campaign configuration.

Discovered artifact. The artifact reported here is a recurrent, execution-aware policy: a flat LSTM actor–critic⁴⁶ trained with PPO⁴⁷, bounded replay of recent recurrent rollouts, supervised cross-sectional warm-up objectives, and a deterministic execution adapter that converts the 100 policy scores into long/cash weights under top- k selection, cash, concentration, liquidity, and turnover constraints. Its lineage traces to an earlier LSTM–PPO parent with recent-rollout replay and behavior-cloning warm-up; a later repair step added effective-number regularization and maximum-weight penalties to that parent while keeping its return. We select it post hoc, as the highest walk-forward CAGR recorded anywhere in the campaign; it is not the campaign’s own promoted artifact. Under the task’s promotion rule it sits in the incubator lane—first-tier evidence on one seed, with three hard constraint violations—while the highest confirmed-lane result is a different, generation-19 cross-sectional attention policy that completed the full five-seed tier cleanly at a lower CAGR. The numbers below therefore characterize a case-study artifact chosen by the reported metric, not an artifact that cleared the campaign’s clean-promotion gate.

Results. Table 5 reports the walk-forward comparison. Over the 28 windows the discovered policy compounds to a 1,864.5% cumulative return, a **53% calendar-time CAGR versus 23% for the paired baseline**—a 2.3-fold ratio in growth rate and an advantage of roughly 30 percentage points. The policy is positive in 26 of 28 quarters, with a 1.56 quarterly zero-rate Sharpe ratio, a 12.75% mean and 33.48% worst quarterly-window maximum drawdown, and beats its baseline in every calendar year of the test period. On a strictly training-excluded 2026 validation (model trained only through 23 December 2025, evaluated 2 January–21 May 2026), the policy returns 21.85% with a 10.97% maximum drawdown and a 1.73 daily zero-rate Sharpe. The result is also robust to execution costs: under an additional 50 basis points per executed side, the mean quarterly return declines from 12.28% to 7.09% but remains positive.

Qualitative analysis. The discovery trajectory illustrates the generational dynamics of Section 2. The best-so-far CAGR rises through intermediate recurrent-policy variants before reaching 53%, and progress is deliberately non-monotonic at the attempt level: later generations spend much of their effort on exploration, ablation, and validation rather than on chasing the incumbent’s score. Subsequent candidates include falsification probes, positive controls, and cost-penalty variants that never exceed the incumbent but harden the evidence around it. The selected mechanism is coherent rather than accidental: the frontier retained the recurrent-PPO parent for its return, a diagnostic finding flagged its concentration risk, and the repair intervention targeted diversification directly—whose effect is visible in the reported policy’s realized statistics (8.01 mean effective names, 21.69% maximum mean single-name weight, 7.95% mean daily turnover, 9.57% mean

Table 5: Quant case study: rolling walk-forward results of the discovered policy versus the paired all-eligible equal-weight baseline (2019Q1–2025Q4, 28 quarterly windows). Calendar-year rows compound that year’s four independently cash-reset quarterly windows; worst quarter and positive quarters refer to the PRAXIST policy.

Year	PRAXIST policy	Baseline	Worst quarter	Positive quarters
2019	64.17%	31.20%	5.59%	4/4
2020	194.51%	53.34%	7.34%	4/4
2021	47.59%	31.76%	0.81%	4/4
2022	−19.66%	−22.88%	−29.14%	3/4
2023	62.24%	34.41%	5.18%	4/4
2024	39.96%	19.98%	3.58%	4/4
2025	50.90%	27.63%	−7.21%	3/4
Cumulative return (2019–2025)	1,864.5%	320.8%		
Calendar-time CAGR	53.07%	22.80%		

cash). The repair moved concentration in the intended direction but fell short of its own pre-registered success signals of more than ten effective names and a maximum weight below 0.15, and the strong setting was retained for its return rather than for meeting those targets.

3.5 Case Study: SLAM

Problem setup. We consider online LiDAR–inertial–visual odometry (LIVO), where a moving platform estimates its six-degree-of-freedom pose by sequentially fusing inertial, LiDAR, and camera measurements into a shared state estimate (Figure 4). Our starting point is FAST-LIVO2⁴⁸, a state-of-the-art tightly coupled LIVO system built around a shared error-state iterated Kalman filter (ESIKF). FAST-LIVO2 applies a fixed estimation sequence: inertial propagation, LiDAR point-to-plane update, and sparse-direct visual update. This design gives a strong reference pipeline, but it also imposes a uniform visual-computation policy: the visual update is attempted on each incoming camera frame, regardless of whether that frame provides substantial marginal information beyond the current LiDAR-constrained state and visual map. We use aligned absolute pose error (APE) RMSE as the primary accuracy metric, computed against Leica ground truth under the evaluator’s rigid Umeyama alignment with 0.02 s association, on the UAV-borne NTU-VIRAL benchmark⁴⁹. The baseline is the stock FAST-LIVO2 implementation as built for this campaign—close to, but not byte-identical to, that build with COVSCHED disabled—and it achieves a mean full-sequence APE RMSE of 0.0937 m across the evaluated sequences. The campaign is deterministic and carries no training seed, so we report no repetition variance. Appendix B.4 gives the full evaluation protocol, the timing environment, the campaign configuration, and the run-acceptance caveat that qualifies which of the campaign’s repeated replays reach the table.

Discovered mechanism. The mechanism selected by PRAXIST, COVSCHED, treats the sparse-direct visual update as a resource to be allocated rather than as an unconditional step in the estimation loop. It adds two visual-pathway controls to the otherwise unchanged FAST-LIVO2 pipeline: a frame-level marginal-value scheduler that decides whether an incoming camera frame triggers a sparse-direct visual update, and a deduplicating map-admission policy that limits redundant map growth by suppressing candidate landmarks which re-observe geometry already represented in the

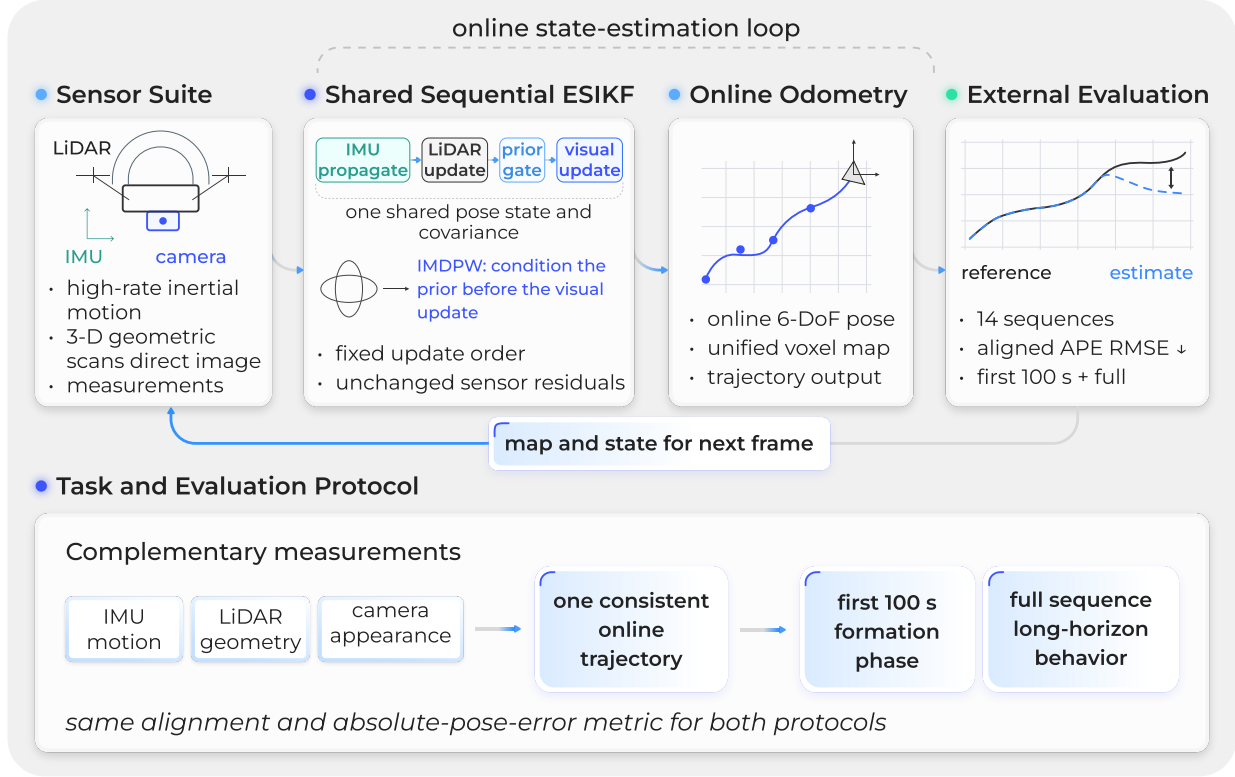


Figure 4: SLAM task overview. Synchronized IMU, LiDAR, and camera measurements update one shared ESIKF state in sequence, producing an online 6-DoF trajectory and map. COVSCHED, the blue gate, intervenes only at the visual pathway: it schedules sparse-direct visual updates by marginal value and admits map points only when they add new geometry, leaving the update order and the sensor residuals unchanged. Evaluation aligns the estimated and reference trajectories and reports full-sequence APE RMSE.

sparse map. IMU propagation, the LiDAR point-to-plane update, and every residual definition are left unchanged, so no estimator is retuned. The two builds are not, however, identical outside the visual pathway: the COVSCHED build publishes odometry stamped with the LiDAR measurement time, whereas unmodified FAST-LIVO2 stamps its output with wall-clock publication time. Because APE associates estimated and reference poses inside a 20 ms window, that difference is itself an accuracy factor, and we quantify it below.

In the champion configuration the scheduler (PVTR_MODE=8) keys on a LiDAR translation-observability signal $o_{\min} = \lambda_{\min}(\sum_i \mathbf{n}_i \mathbf{n}_i^T)$, where $\mathbf{n}_i$ is the normal of a LiDAR point-to-plane correspondence. Large $o_{\min}$ indicates that the LiDAR geometry already constrains the local translation subspace well, so the marginal value of a visual update is low; small $o_{\min}$ indicates a weakly constrained translation direction, where visual information is most valuable. COVSCHED combines this instantaneous redundancy signal with a persistent low-observability dwell counter and integrates the resulting demand through a saturating skip budget with a hard cap of three consecutively skipped frames, which is what prevents permanent visual starvation. The recorded schedules are strongly protective of the frames that need vision: on nya_03, high- $o_{\min}$ frames are skipped 74.9% of the time against 0.3% for low- $o_{\min}$ frames. The second component (VMAP_DEDUP=1) discards a candidate visual point as a re-observation rather than admitting it as new

Table 6: SLAM case study: full-sequence APE RMSE (m) of the FAST-LIVO2 baseline versus the discovered COVSCHED module on all fourteen NTU-VIRAL sequences. *Coverage* is the ratio of COVSCHED to baseline ground-truth-associated samples, and the last column the reduction in the evaluator-captured amortized visual-path (VIO) processing time; both mean-row entries there are macro averages of per-sequence values, not ratios of means. Because the two arms stamp their poses differently, the APE columns are reported side by side with no relative-improvement column; Section 3.5 bounds the difference between them.

Sequence	FAST-LIVO2 RMSE	COVSCHED RMSE	Coverage	Evaluator-captured VIO time
eee_01	0.0582	0.0279	1.03	77.8%
eee_02	0.0602	0.0408	0.90	75.6%
eee_03	0.0725	0.0322	0.79	70.2%
nya_01	0.0537	0.0308	0.99	73.6%
nya_02	0.0856	0.0353	1.76	75.6%
nya_03	0.0848	0.0339	0.99	75.9%
rtp_01	0.0959	0.0525	1.23	81.6%
sbs_01	0.0524	0.0300	0.98	73.7%
sbs_02	0.0567	0.0341	0.99	75.6%
sbs_03	0.0614	0.0308	0.98	74.1%
spms_01	0.1795	0.0925	0.93	74.8%
spms_02	0.2407	0.1132	0.92	43.2%
spms_03	0.1118	0.0500	1.15	70.1%
tnp_03	0.0978	0.0979	0.84	71.6%
Mean (14 seq.)	0.0937	0.0501	1.03	72.4%

coverage when it lies within 0.08 m of an existing map point in the same voxel with a normal deviation below 15° . COVSCHED thus governs both when visual computation is spent and which visual observations are allowed to grow the map; Appendix B.4 records the parameter settings and two qualifications they carry.

Results. Table 6 reports the full-sequence comparison against FAST-LIVO2. Across the fourteen NTU-VIRAL sequences the two arms record mean APE RMSE of 0.0937 m and 0.0501 m respectively, with the COVSCHED value lower on thirteen sequences and effectively tied on tnp_03 (0.0978 m versus 0.0979 m). We deliberately do not report that gap as an accuracy improvement. The two arms stamp their published poses under different conventions, and re-associating them under a single rule removes the gap almost entirely; the caveats below give the measurement and the bound it implies. The accuracy claim we do make is the weaker one: on this benchmark, spending far less visual computation does not cost trajectory accuracy.

The main measured efficiency gain is on the visual-processing path. Across the same fourteen runs, the evaluator-captured amortized VIO processing time decreases by 72.4% on average, with a median reduction of 74.4%; every sequence shows a reduction, ranging from 43.2% on spms_02 to 81.6% on rtp_01. Because FAST-LIVO2’s LiDAR mapping and visual processing are reported by separate asynchronous timing streams, we do not interpret this number as a 72.4% reduction in end-to-end SLAM latency, throughput time, or CPU load. The LiDAR-side timing is essentially unchanged. As a conservative aggregate proxy, summing the internally reported LIO and VIO thread-wall workloads gives a 22.1% reduction. The supported resource claim is therefore that

COVSCHED substantially reduces the measured visual-path workload.

These results support the resource-allocation reading of the mechanism: COVSCHED does not trade accuracy for speed by uniformly thinning the visual stream, but conditions visual computation on state-dependent marginal value and limits redundant sparse-map growth. The saving appears on every sequence, whereas the residual APE difference does not track task difficulty ($r = 0.13$ between baseline APE and the per-sequence gap). That is the pattern the timestamp analysis below predicts, and it is consistent with the hypothesis motivating the mechanism—that many baseline visual updates are redundant given the current LiDAR-constrained state and visual map—while leaving the accuracy side of the comparison uncommitted.

The claim is nevertheless bounded, most consequentially by a timestamp confound. The two arms of Table 6 do not share a pose-timestamp convention: COVSCHED publishes measurement-time stamps and the unmodified baseline publishes wall-clock stamps, and a same-binary control that toggles only that setting on a fixed trajectory moves APE by a factor of 1.74–1.80. Re-associating the baseline poses under the COVSCHED rule—possible without interpolation on the ten sequences where both arms emit the same number of poses—leaves the mean relative APE change at -0.09% rather than the $+45\%$ the raw columns suggest. The gap between the two accuracy columns is therefore dominated by the timestamp and association convention, which is why we report no improvement column and make no accuracy-gain claim. Three further caveats, detailed in Appendix B.4, qualify the rest: the ground-truth-associated pose sets differ between the arms, so the comparison must be read together with coverage ($0.79\times$ to $1.76\times$); relative pose error over short 1–30 s windows is mixed rather than uniformly improved; and the component controls are too thin, and too confounded with the map-admission filter, to attribute the result to either half of the mechanism. Establishing the mechanism’s true effect on accuracy and supplying the missing factorial both require the same outstanding experiment—a paired re-run of all fourteen sequences with both arms built from the same source and stamped the same way. The current evidence therefore supports COVSCHED as a structured, bounded, map-aware visual-resource policy, without assigning the improvement to either component alone.

Qualitative analysis. COVSCHED is a change in visual-processing policy rather than in sensor modeling: FAST-LIVO2 implicitly assumes that every incoming visual frame should enter the sparse-direct update, and COVSCHED makes that decision conditional on the current state of the LiDAR geometry and the visual map. The distinction also shaped the evidence package. Because the estimated trajectories are visually close under a direct overlay, the run reports distributional APE, executed and skipped visual-update schedules, visual-map growth, runtime, and coverage rather than relying on trajectory plots, and—following the evidence contracts of Section 2—it declines to claim a duplicate-merge statistic that its instrumentation never recorded, reporting the supported traces and preserving the coverage ratios and rate-matched skipping control that qualify the comparison. That restraint matters because a visual-resource mechanism is easy to overclaim: it may improve APE by reducing computation, by suppressing harmful redundant updates, by changing the evaluated pose set, or by some combination of the three. The current evidence supports COVSCHED as a strong structured resource policy, and the outstanding factorial would sharpen the causal attribution.

3.6 Case Study: Fusion (Tokamak Magnetic Control)

Problem setup. The task is real-time magnetic control of a tokamak: holding a ring of plasma in the right shape and place inside a MAST-U-like spherical fusion device (Figure 5). The plasma is

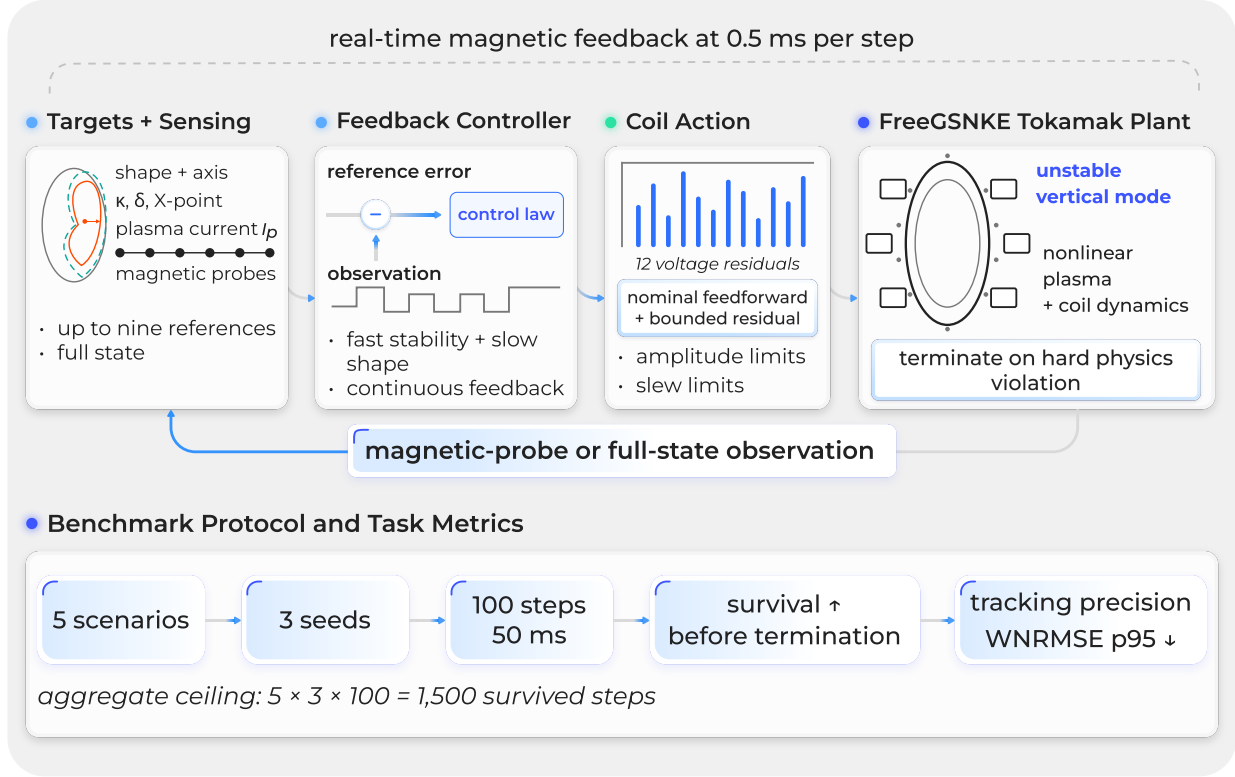


Figure 5: Fusion task overview. At every 0.5-ms step the controller compares plasma targets with the scenario’s observation and adds 12 bounded voltage residuals to nominal coil feedforward; the two closed-loop controllers evaluated here read the target values from the harness rather than from the probe observation, as noted in the text. FreeGSNKE advances the nonlinear tokamak state and terminates on hard physics violations; the benchmark scores 15 episodes by aggregate survival and tracking precision, which we report on both the benchmark’s full horizon and the common horizon defined in the text.

simulated by FreeGSNKE, a time-dependent free-boundary Grad–Shafranov solver⁵⁰, wrapped as a reinforcement-learning environment. Every 0.5 ms it adjusts 12 magnetic-coil voltages, added to the scenario’s nominal schedule under hard amplitude and slew limits, and must hold up to nine targets on their references: the plasma boundary’s radial extents, the position of its magnetic axis, its elongation κ and triangularity δ , the lower X-point position, and the plasma current I_p . Any hard physics violation ends the episode at once: the plasma drifting out of position, losing its current, exceeding a coil limit, touching the wall, or the solver diverging. The plant is genuinely unstable: left uncontrolled the plasma column runs away vertically, at open-loop growth rates matching those measured in real MAST-U vertical-displacement experiments. The scenario’s nominal voltage schedule nonetheless carries an unactuated plasma through part of the horizon, which is why the zero-feedback floor in Table 7 sits well above zero rather than near it.³

The benchmark protocol is 5 scenarios $\times$ 3 random initializations $\times$ a 100-step (50-ms) horizon, so a controller that never fails survives 1,500 steps in total. Four scenarios build the observation from

³The simulator derives growth rates of 417–470 rad/s for the four perturbed scenarios—a vertical-displacement doubling time of ≈ 1.5 ms, about three control periods—comfortably inside the 380–500 rad/s range measured on the MAST-U device itself⁵¹.

what magnetic diagnostics would see—32 flux loops and 32 pickup coils, corrupted by Gaussian noise, together with the 12 coil currents, the previous 12 voltage commands, and the target references, stacked over two frames and delayed by one step, for a 182–194-dimensional vector—and add plasma and actuator perturbations; one certification scenario gives the full simulator state, unperturbed. There are two task-native **baselines**. The first is a zero-feedback controller, the task’s performance floor: the plasma simply coasts on the scenario’s nominal voltages. The second—the demanding one—is a PCS-style controller based on the plasma-control architecture deployed on MAST-U in its 2022–2023 experimental campaign⁵²: damped pseudo-inverse virtual circuits for slow shape control, an independent vertical-stabilization PD loop, and an ohmic plasma-current controller. The benchmark implementation uses FreeGSNKE’s one-step voltage sensitivities and a virtual-circuit gain of $g_{vc} = 0.3$ selected by an offline stability scan. Two protocol boundaries qualify every number below. First, both closed-loop controllers are privileged-state controllers: the evaluation harness passes the simulator’s true target values to the controller on every step, and both read their tracking errors from that channel rather than from the noisy probe observation the scenarios construct, which neither controller consumes—we accordingly refer to those four scenarios below by their perturbations rather than their observation model. The comparison between them is therefore internally matched, but it does not establish that either would work from probe measurements alone; a probe-only re-evaluation behind a state estimator is outstanding work. Second, the PCS-style baseline is an architectural reconstruction rather than a port of the deployed controller: its slow loop is proportional instead of PI, it uses one-step voltage sensitivities in place of a steady-state current Jacobian, it stabilizes vertically through the fast PX coil instead of the antisymmetric P6 pair, and its vertical gains, I_p block, shaping-coil counter-drain, and waypoint warm-up are inherited from the PRAXIST controller. It is a strong in-simulator baseline, not an independent reproduction of MAST-U’s plasma control system. Appendix B.5 records the evaluator’s runner and reset lifecycle, to which every score below is bound, and the campaign configuration.

Discovered artifact. The selected artifact, HybridJacobianPDV1, is a sparse Jacobian-sign PD controller: a superposition of four functionally distinct channels—a full-error vertical PD on the dedicated fast coil with a fixed -0.30 counter-drain onto three shaping coils, a waypoint-ramped radial-position proportional loop, an ohmic I_p loop on the central solenoid, and six offline system-identified unit-norm Jacobian sign vectors that route each shape axis to the three coils identified as dominating it—under a global action clip and slew limit. The channels are decoupled by design intent, not by actuator: the vertical, radial, and shaping channels all act on P4–P6, and their contributions are summed into one voltage vector before the shared clip and slew limit. Architecturally, the artifact belongs to the same inverse-Jacobian PD family as the deployed PCS; it differs in replacing the dense damped pseudo-inverse feedback matrix with sparse sign vectors, and in closing the loop on seven geometric target axes rather than four (explicitly adding κ , δ , and the magnetic-axis radial position, which the deployed PCS does not feed back). Counting the dedicated vertical and I_p loops as well, the two controllers close nine and six axes respectively.

Results. Table 7 reports the benchmark comparison on both precision metrics, and the verdict is split rather than uniform. PRAXIST attains the highest aggregate survival (1,264 of 1,500 steps, versus 1,222 for the PCS-style controller and 1,039 for zero feedback) and the lowest common-horizon tracking error (WNRMSSE p_{95} of 2.86 versus 2.99 and 4.89), and it has the lowest common-horizon error in all four perturbed scenarios. The PCS-style controller completes more episodes (11 of 15 against 10) and holds the lower error on the benchmark’s original full-horizon metric (aggregate p_{95} of 4.42 versus 4.65), leading there on three of the four perturbed scenarios as well. Because a

controller that does nothing already survives 1,039 steps, survival margins are best read against the 461 steps of headroom that remain: PRAXIST recovers 48.8% of that headroom and the PCS-style controller 39.7%, a gap of 9.1 points that the raw ratios compress into a 22% improvement over the open-loop floor and a 3.4% improvement over the baseline.

Which precision metric to read is itself a finding of this case study. The full-horizon percentile is *not* monotone in controller quality, because longer-surviving controllers contribute more near-failure, high-error steps to the sample pool, and its aggregate values here show the distortion directly: it separates the two closed-loop controllers from zero feedback by only 10% and 5% even though they survive 18% and 22% longer, so on that metric feedback registers as nearly worthless on a plant that is unstable without it. Recomputing precision on a per-seed common horizon capped at the zero-feedback baseline’s survival length removes the confound and restores the physically expected ordering, at the price of discarding the late-horizon steps where the two closed-loop controllers differ most—which is why we report both rather than substituting one for the other. Neither is the benchmark’s verdict: its own classification thresholds are stricter— ≤ 1.0 for *acceptable* and ≤ 0.5 for *solved*, and both additionally require a completed horizon—so none of the three controllers is classified as passing under the official rule, and both precision columns are descriptive comparisons reported separately from that classification.

On the common horizon both closed-loop controllers cut per-step tracking error by roughly 40% relative to zero feedback; relative to the PCS-style controller, PRAXIST improves aggregate survival by 3.4% and common-horizon precision by 4.4%, margins that Appendix B.5 places against the reproduction spread of a single controller under the same protocol. The clearest qualitative difference appears on the certification scenario under the full horizon: there, explicit feedback on κ , δ , and the axis position keeps late-horizon tracking bounded (full-horizon p_{95} of 3.76 versus 13.46 for the PCS-style controller), the one place where the two horizons disagree by an order of magnitude rather than by a few percent.

Table 7: Fusion case study: MAST-U magnetic-control benchmark (5 scenarios $\times$ 3 seeds $\times$ 100-step horizon; 1,500 aggregate steps maximum). Survival counts steps before a hard physics termination; Compl. is the fraction of the 15 episodes completing the full horizon; WNRMSE p_{95} is the 95th percentile per-step weighted normalized tracking error (lower is better). The six common-horizon columns cap each episode at the zero-feedback baseline’s survival length for that scenario and seed; the last column is the benchmark’s original full-horizon aggregate, and Table 35 gives that metric per scenario. Each *all* entry is pooled over every scored step rather than averaged across the scenario columns. Scenario columns abbreviate *main_shape*, *axis_ip*, *ip_shape*, *xpoint*, and *cert_shape*. Best value per column in bold. Both closed-loop controllers read their target errors from the harness rather than from the probe observation, so the table is an internally matched comparison inside a privileged-state harness, as stated in the text.

Controller	Survival (/1500)	Compl.	Common-horizon WNRMSE p_{95} ($\downarrow$)						Full-hor.
			main	axis	ip	xpt	cert	all	all
Zero-feedback baseline	1039	0.267	4.77	6.10	5.12	3.85	4.74	4.89	4.89
MAST-U PCS-style	1222	0.733	2.63	3.54	2.63	1.38	2.76	2.99	4.42
PRAXIST (ours)	1264	0.667	2.55	3.48	2.43	1.21	2.81	2.86	4.65

Qualitative analysis. The value of this case study lies less in the margin than in the discovery

dynamics. The champion was synthesized by *composing* mechanisms discovered independently: a full-error vertical PD with coil counter-drain, its high-derivative-gain variant, simulator-identified Jacobian sign vectors, and a shared waypoint warm-up for the slow channels. The run’s own probes are directional rather than single-factor. Replacing the counter-drain collapses the plasma at step 82, and extending the delayed waypoint ramp to the vertical channel collapses it at step 85; both variants change the coil allocation at the same time as the mechanism under test, and both are certification-only, single-seed probes, so they order the variants without isolating one cause. The sign vectors are the clearest case of the run correcting itself. An early comparison of a hand-tuned controller against a system-identified one showed 57 steps against a full horizon, which reads as evidence that identification matters; but those two controllers also differed in gains, coil maps, deadbands, and smoothing. A later matched control that held the trajectory and gains fixed and randomly flipped half of the identified signs scored 297 against 298 aggregate steps, with identical results on three of the probe scenarios and certification error of 3.49 against 3.48. The run therefore recorded the identified signs as non-essential, and we report that as the standing conclusion. Subsequent experiments in the reported lineage tested observation-quality gating, integral loops, coil specialization, mechanism substitutions, per-coil quadratic programs, and spectral decoupling without finding a stronger controller, and the survival-versus-certification-precision Pareto frontier stayed around this artifact. That statement is bounded to the seven generations behind the reported result: a later continuation of the campaign, reported alongside it, produced controllers with substantially higher aggregate survival (up to 1,481 of 1,500) at worse full-horizon precision, so the frontier has since acquired new survival-heavy trade-off points rather than remaining a single point.

4 Conclusion and Discussion

This work presents PRAXIST, an autonomous research system built around a simple premise: long-horizon R&D improves when a campaign inherits evidence, not just scores. PRAXIST realizes this as a generational artifact-to-lineage cycle—evaluated artifacts become typed findings, findings populate a lane-structured frontier, PI/Chair synthesis emits the next agenda, durable lessons are compressed into Gems, and the accumulated lineage both directs later attempts and documents why the final artifact is credible. Our two-part evaluation targets complementary questions. On MLE-bench, PRAXIST obtains 60 medals (80.0%) across all 75 tasks, including 49 gold, compared with 55 medals (73.3%) and 34 gold for Claude Code on Claude Opus 4.8. The case studies show that the system can carry open-ended R&D problems to artifacts that beat their task-native baselines: it discovered a trading policy with a 53% walk-forward CAGR, 2.3 times the 23% of its paired all-eligible equal-weight baseline, turned the always-on visual update of a state-of-the-art SLAM system into a resource-aware schedule that cut the evaluator-captured visual-path processing time by 72.4% without costing trajectory accuracy, synthesized a tokamak controller with 3.4% higher aggregate survival and 4.4% lower common-horizon tracking error than a task-native MAST-U PCS-style controller inside the same privileged-state harness, though not on that benchmark’s original full-horizon precision metric, and produced a deterministic rocket-landing controller that lands 12,288/12,288 trajectories under a frozen first-contact protocol; Weco, an autonomous code optimizer given the same task, objective, and 4.03% starting artifact, reports 17.12%. The rocket controller was not found in one attempt: its lineage accumulates a fuel-commit governor, attitude-authority and phase-transition guards, and finally a closed-form constrained control allocator, each inherited as typed evidence rather than as a score.

Beyond the headline scores, two properties distinguish PRAXIST as a research collaborator. First, 49 of its 60 medals are gold (81.7%), so the majority of its successful MLE-bench outcomes reach the highest medal tier. Second, results can arrive with their solution lineage—an inspectable account of the mechanisms, controls, and failures behind a result—supporting reuse and extension by human scientists; the four case studies report those lineages in detail, whereas for the MLE-bench sweep we report only the graded outcome per task, and the coverage the ledgers achieve varies with how long a run persisted before it was stopped. These properties suggest immediate applications wherever a task admits an executable evaluator: model and algorithm development, controller synthesis for physical systems, simulation-driven engineering design, and quantitative strategy research. Looking forward, we see three directions: inheritance across campaigns, so that Gems and lineages from one problem seed the next; extension to slower or noisier evaluators, bringing the cycle closer to laboratory science; and deeper human–AI collaboration through lineages, positioning PRAXIST not as a replacement for scientists but as an instrument that compounds their evidence.

Code and Released Runs

The PRAXIST implementation, the project page, and the runs behind the results reported in this paper are publicly available. The released archive holds the generational record of each run—artifacts, evaluation records, findings, frontier lanes, agendas, and lineage traces—so that the trajectories summarized in Section 3 and in the appendices can be inspected generation by generation, rather than only through the aggregate scores reported here.

-  Code: github.com/sapientinc/praxist
-  Project page: praxist.sapient.inc
-  Released runs: [Google Drive archive](#)

Acknowledgements

We thank Yaning Han, Rongzu Zhang, Daohai Yu and Yangzhou Liu for their insightful discussions and contributions to this work.

References

1. Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery, 2024.
2. Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search, 2025.
3. Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants, 2025.
4. Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, et al. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024.
5. Alexander Novikov, Ngan Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery, 2025.
6. Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-driven exploration in the space of code, 2025.
7. Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAGentBench: Evaluating language agents on machine learning experimentation. In *International Conference on Machine Learning*, 2024.
8. Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, 2025.
9. Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. MLGym: A new framework and benchmark for advancing AI research agents, 2025.
10. Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-bench: Evaluating frontier AI R&D capabilities of language model agents against human experts, 2024.
11. Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. MLR-bench: Evaluating AI agents on open-ended machine learning research, 2025.
12. Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. DS Bench: How far are data science agents from becoming data science experts?, 2024.

13. Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL <https://arxiv.org/abs/2310.06770>.
14. John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024.
15. Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, et al. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025. URL <https://arxiv.org/abs/2407.16741>.
16. Zachary S. Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan. CORE-bench: Fostering the credibility of published research through a computational reproducibility agent benchmark, 2024.
17. Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI’s ability to replicate AI research, 2025.
18. Abhishek Sharma, Dániel Czégel, Michael Lachmann, Christopher P. Kempes, Sara I. Walker, and Leroy Cronin. Assembly theory explains and quantifies selection and evolution. *Nature*, 622:321–328, 2023. doi: 10.1038/s41586-023-06600-9.
19. Stuart M. Marshall, Cole Mathis, Emma Carrick, Graham Keenan, Geoffrey J. T. Cooper, Heather Graham, Matthew Craven, Piotr S. Gromski, Douglas G. Moore, Sara I. Walker, and Leroy Cronin. Identifying molecules as biosignatures with assembly theory and mass spectrometry. *Nature Communications*, 12(1):3033, 2021. doi: 10.1038/s41467-021-23258-x.
20. Christopher P. Kempes, Michael Lachmann, Andrew Iannaccone, G. Matthew Fricke, M. Redwan Chowdhury, Sara I. Walker, and Leroy Cronin. Assembly theory and its relationship with computational complexity. *npj Complexity*, 2:27, 2025. doi: 10.1038/s44260-025-00049-9.
21. Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.
22. Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.
23. Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

24. Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023.
25. Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for “mind” exploration of large language model society. In *Advances in Neural Information Processing Systems*, 2023.
26. Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation, 2023.
27. Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024.
28. Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *International Conference on Machine Learning (ICML)*, 2024.
29. Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *AAAI Conference on Artificial Intelligence*, 2024.
30. Yixin Liu, Guibin Zhang, Kun Wang, Shiyuan Li, and Shirui Pan. Graph-augmented large language model agents: Current progress and future prospects. *arXiv preprint arXiv:2507.21407*, 2025.
31. Chang Yang, Chuang Zhou, Yilin Xiao, Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang, Zijin Hong, Zheng Yuan, Zhishang Xiang, Shengyuan Chen, Huachi Zhou, Qinggang Zhang, Ninghao Liu, Jinsong Su, Xinrun Wang, Yi Chang, and Xiao Huang. Graph-based agent memory: Taxonomy, techniques, and applications. *arXiv preprint arXiv:2602.05665*, 2026.
32. Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.
33. Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites, 2015.
34. Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.
35. Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2018.
36. Paul Groth and Luc Moreau. PROV-overview: An overview of the PROV family of documents. W3c working group note, World Wide Web Consortium, 2013.
37. Stian Soiland-Reyes, Peter Sefton, Mercè Crosas, Leyla Jael Castro, Frederik Coppens, José María Fernández, Daniel Garijo, Björn A. Grüning, Marco La Rosa, Simone Leo, et al. Packaging research artefacts with RO-Crate. *Data Science*, 5(2):97–138, 2022. doi: 10.3233/DS-210053.

38. Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, and Corey Zumar. Accelerating the machine learning lifecycle with mlflow. In *IEEE Data Engineering Bulletin*, 2018.
39. Klaus Greff, Aaron Klein, Martin Chovanec, Frank Hutter, and Jürgen Schmidhuber. The sacred infrastructure for computational research. In *Proceedings of the Python in Science Conference*, 2017.
40. Mingqi Liu, Zhonghua Han, Tao Du, Chenzhou Xu, Han Zeng, Keshi Zhang, and Wenping Song. Optimal control efficiency characteristics and wide-speed-range aerodynamic design optimization method for grid fins of launch vehicle. *Acta Aeronautica et Astronautica Sinica*, 45(20):129887, 2024. doi: 10.7527/S1000-6893.2024.29887.
41. Weco AI. Rocket landing success: a Weco code-optimization run on the rocket landing task. Shared run trace on the Weco AI dashboard, 2026. URL https://dashboard.weco.ai/share/xZJAbpYNN7aJCvyiZuuZFsp_hzQdPqTy. Shared run trace; aggregate figures as reported by the run dashboard. Accessed 2026-08-24.
42. Yanning Guo, Matt Hawkins, and Bong Wie. Applications of generalized zero-effort-miss/zero-effort-velocity feedback guidance algorithm. *Journal of Guidance, Control, and Dynamics*, 36(3):810–820, 2013. doi: 10.2514/1.58099.
43. Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425, 2010. doi: 10.1109/CDC.2010.5717652.
44. Marc Bodson. Evaluation of optimization methods for control allocation. *Journal of Guidance, Control, and Dynamics*, 25(4):703–711, 2002. doi: 10.2514/2.4937.
45. Tor A. Johansen and Thor I. Fossen. Control allocation—a survey. *Automatica*, 49(5):1087–1103, 2013. doi: 10.1016/j.automatica.2013.01.035.
46. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
47. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.
48. Chunran Zheng, Wei Xu, Zuhao Zou, Tong Hua, Chongjian Yuan, Dongjiao He, Bingyang Zhou, Zheng Liu, Jiarong Lin, Fangcheng Zhu, Yunfan Ren, Rong Wang, Fanle Meng, and Fu Zhang. FAST-LIVO2: Fast, direct LiDAR-inertial-visual odometry. *IEEE Transactions on Robotics*, 41:326–346, 2025.
49. Thien-Minh Nguyen, Shenghai Yuan, Muqing Cao, Yang Lyu, Thien H. Nguyen, and Lihua Xie. NTU VIRAL: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint. *The International Journal of Robotics Research*, 41(3):270–280, 2022.
50. N. C. Amorisco, A. Agnello, G. Holt, M. Mars, J. Buchanan, and S. Pamela. FreeGSNKE: A python-based dynamic free-boundary toroidal plasma equilibrium solver. *Physics of Plasmas*, 31(4):042517, 2024.
51. A. Lvovskiy, H. Anand, A. S. Welander, et al. Framework for assessment of magnetic equilibrium controller performance on the MAST Upgrade spherical tokamak. *Plasma Physics and Controlled Fusion*, 67:075003, 2025.

52. H. Anand, W. Wehner, D. Eldon, et al. Real-time plasma equilibrium reconstruction and shape control for the MAST Upgrade tokamak. *Nuclear Fusion*, 64:086051, 2024.
53. Juncheng Gu, Mosharaf Chowdhury, Kang G. Shin, Yibo Zhu, Myeongjae Jeon, Junjie Qian, Hongqiang Liu, and Chuanxiong Guo. Tiresias: A GPU cluster manager for distributed deep learning. In *16th USENIX Symposium on Networked Systems Design and Implementation*, pages 485–500, 2019.
54. Deepak Narayanan, Keshav Santhanam, Fiodar Kazhamiaka, Amar Phanishayee, and Matei Zaharia. Heterogeneity-aware cluster scheduling policies for deep learning workloads. In *14th USENIX Symposium on Operating Systems Design and Implementation*, pages 481–498, 2020.
55. Aurick Qiao, Sang Keun Choe, Suhas Jayaram Subramanya, Willie Neiswanger, Qirong Ho, Hao Zhang, Gregory R. Ganger, and Eric P. Xing. Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning. In *15th USENIX Symposium on Operating Systems Design and Implementation*, pages 1–18, 2021.
56. Leonard Kleinrock. Time-shared systems: A theoretical treatment. *Journal of the ACM*, 14(2): 242–261, 1967.
57. Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.
58. Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. In *Proceedings of Machine Learning and Systems*, volume 2, 2020.
59. Jeffrey Dean and Luiz André Barroso. The tail at scale. *Communications of the ACM*, 56(2): 74–80, 2013.
60. David Q. Mayne, James B. Rawlings, Christopher V. Rao, and Pierre O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.
61. Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks, 2013.
62. Ahuva W. Mu’alem and Dror G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543, 2001.
63. Behçet Açıkmeşe and Scott R. Ploen. Convex programming approach to powered descent guidance for Mars landing. *Journal of Guidance, Control, and Dynamics*, 30(5):1353–1366, 2007. doi: 10.2514/1.27553.

Appendix

A	Method details	33
A.1	Task-Family Realizations	33
A.2	Artifacts, Evaluation Records, and Findings	33
A.3	Design Contracts and Diversity Allocation	34
A.4	PI/Chair Synthesis	35
A.5	Frontier Status and Agenda Decisions	36
A.6	Gems and Selective Memory	38
A.7	Lineage Trace	39
A.8	Resource Scheduling and Mature-Evidence Debt	41
A.9	Configuration Defaults and MLE-bench Budgets	45
A.10	Representative Prompt Contracts	45
A.11	MLE-Style Task Example	50
A.12	Worked MLE-bench Research Trajectory	50
B	Experimental Setup Details	51
B.1	MLE-bench	51
B.2	Rocket	52
B.3	Quant	61
B.4	SLAM	62
B.5	Fusion	63
C	Full MLE-bench Per-Task Results	65

Table 8: Task-family realization components. MLE-style tasks instantiate these components with competition context, submission artifacts, metric semantics, and grader outcomes; other task families instantiate them with their own artifact forms and evaluators.

Component	Method-level role
Allowed task context	Defines the public information, constraints, resources, and task description available to the run.
Artifact form	Defines the reproducible experimental unit a that grounds evaluator outcomes and later research claims.
Evaluator semantics	Defines how artifacts produce task-grounded outcomes $e = \text{EVAL}(a)$ and how validity, metric direction, failure states, or structured reports are interpreted.
Role-local instructions	Specialize peers, PI roles, and Chair synthesis to the task family while preserving the common PRAXIST cycle.
Evidence conventions	Normalize outcomes into findings ϕ , frontier status, agenda items, memory updates, and lineage records.

A Method details

This appendix gives the field-level contracts behind the method in Section 2. The main text defines PRAXIST as the generational state transition $\mathcal{S}_g = (\mathcal{F}_g, \mathcal{A}_g, \mathcal{G}_g, \mathcal{L}_g)$: artifacts ground attempts, findings interpret evaluated evidence, frontiers determine inheritance status, agendas direct later generations, Gems preserve durable lessons, and lineage records the path that produced the final artifact. Here we expand each object using the same symbols and vocabulary as the main text.

The emphasis is on methodological interfaces rather than software mechanics. We describe what information is represented, which role consumes it, and what each role emits. Concrete command invocations, storage layouts, and release-specific engineering details belong to implementation documentation or the experimental protocol. MLE-style tasks are used as a running example of a task-family realization; they are not the definition of PRAXIST.

A.1 Task-Family Realizations

PRAXIST uses a task-agnostic research cycle. A task-family realization supplies the domain-specific information needed to instantiate that cycle: the allowed task context, artifact form, evaluator semantics, role constraints, and evidence conventions. This separation allows the same research-state objects from Eq. 1 to operate across different evaluator-grounded tasks. Table 8 lists the components a realization must supply.

For example, an MLE-style realization provides a public competition description, available data context, submission format, metric name and direction, and evaluator semantics. A proof-search task could instead define a proof artifact and checker outcome; a software-engineering task could define a patch artifact and test report; a simulation task could define a simulation bundle and score report. The common requirement is that evaluated artifacts yield traceable evidence.

A.2 Artifacts, Evaluation Records, and Findings

The artifact layer provides provenance, while the finding layer provides reusable interpretation. Each evaluated attempt is represented by an artifact a_i , an evaluation record e_i , and one or more

Table 9: Evidence records used by PRAXIST. These are conceptual records; task families may instantiate them with different concrete fields.

Record	Contents and role
Artifact a	Minimal reproducible file set or object required to reproduce, inspect, or evaluate one attempt. In MLE-style tasks, this is typically a submission-centered bundle containing the submission, code, and supporting files needed for reproduction.
Evaluation record e	Evaluator outcome attached to an artifact, including validity state, score or report, metric direction, evidence stage, outcome interpretation, parent provenance, and generation/peer provenance.
Finding ϕ	Artifact-grounded claim containing the attempted intervention, observed outcome, supporting evidence, evidence maturity m , limitations, and recommended inheritance action α .

findings $\Phi_i = \text{INTERPRET}(a_i, e_i, d_i)$. The artifact is the reproducible experimental unit. The evaluation record attaches a task-grounded outcome to that artifact. Findings compress the outcome into research claims with maturity and inheritance recommendations. Table 9 defines these three records; Table 10 gives the evidence-stage ladder behind maturity m , and Tables 11 and 12 give the finding fields and the inheritance actions.

Artifact statuses. Artifacts are assigned a status before they are used as evidence: *committed* means the artifact is a complete canonical state that may serve as a fact source; *partial* means the attempt preserved useful intermediate evidence but did not reach a full artifact state; *failed* means construction or evaluation failed in a way that may still yield diagnostic evidence; and *superseded* means a later artifact or frontier entry replaces the artifact as the current inheritable state. Only committed artifacts are treated as primary sources for positive inheritance; partial and failed artifacts can still produce diagnostic or uncertain findings when the failure mode is informative.

Evaluator interface. The external evaluator is task supplied and returns $e_i = \text{EVAL}(a_i)$. An evaluation record should preserve the artifact identifier, evaluator version or protocol, metric name, score or structured report, metric direction, validity state, evidence stage, generation and peer identifiers, parent lineage, and any limitations needed to interpret the result. PRAXIST keeps validity separate from score: an invalid high score is a diagnostic signal, not a confirmed improvement.

Findings are typed by research function. Positive findings identify improvements or reusable mechanisms. Negative findings weaken an assumption or intervention. Diagnostic findings explain constraints, failure modes, invalidity conditions, or bottlenecks. Uncertain findings preserve evidence that may be useful after further checking. Procedural findings record constraints on how future experiments must run. Evidence maturity is distinct from outcome quality: a high-scoring artifact may still need validation, while a failed artifact can yield a mature diagnostic finding.

A.3 Design Contracts and Diversity Allocation

PRAXIST assigns a Deep Innovation Gate (DIG) design contract d_i before peer p_i constructs an artifact. The gate’s generation scope is a task setting: it defaults to the opening generation, where the fields below are produced by the systematic gate itself, and a task may set it to run in every generation. Where the default is used—as in the rocket campaign—later generations receive the

Table 10: Evidence-stage ladder. Maturity m is inherited from the artifact’s evaluation stage. Preliminary stages can inform memory and diagnostics but are not durable frontier evidence. Ranks 0–2 are the generic ordering the core enforces for every task; the replication rungs above them are task-owned labels, shown here with the names used by the trading campaign, and other campaigns supply their own (the rocket evaluator, for instance, reports canary, development, and complete). The core treats any task-declared replication stage as mature evidence above scored_complete.

Stage	Rank	Meaning	Mature?
smoke	0	Cheap sanity check, parser check, tiny run, or other minimal validity probe.	No
scout	1	Partial probe or early score used to decide whether a direction deserves more compute.	No
scored_complete	2	Complete task-grounded score or report for the artifact under the standard evaluator path.	Yes
full_T1	3	Full Tier-1 evaluation with stronger completion or reproducibility evidence than a single scored run.	Yes
T2	4	Replicated, ablated, or otherwise strengthened evidence beyond Tier 1.	Yes
T3 / forced_T3	5	Highest-maturity evidence, including replication or forced validation used for final claims.	Yes

same contract fields from the Chair’s agenda instead. DIG is a read-only innovation gate: it makes the intended experiment explicit by specifying what mechanism is being tested, where the intervention occurs, which prior evidence it extends, what evidence would support or weaken it, and which changes would invalidate the test. The contract is used later to judge whether the artifact actually tested the intended idea. Table 13 lists the contract fields, and Table 14 the cohort-level allocation caps that keep the contracts diverse.

At the generation level, PRAXIST allocates contracts across diverse design cells. ALLOCATE (Section 2.2) first scores candidate contracts by quality, lane fit, novelty, risk, and diagnostic need, then greedily selects a constrained set of C contracts. This is Quantified Diversity (QD): diversity is carried by explicit design-cell coordinates and controlled by caps, so the cohort covers distinct mechanism families, intervention surfaces, and intents in one generation. The caps bind differently across the run. In the opening allocation the greedy selector enforces them as hard constraints. In later generations the same coordinates and caps are given to the Chair as soft allocation targets when it writes the peer contracts, and the agenda validator treats missing or unrecognized planning dimensions as a warning rather than a rejection, so coverage there is planned rather than guaranteed.

A.4 PI/Chair Synthesis

PI/Chair synthesis is the generation-level decision stage. Peers publish local artifacts and findings. PI roles independently interpret the evidence from complementary perspectives. The Chair merges these views into an agenda with claim boundaries, validation targets, peer contracts, and archive decisions. Table 15 defines the synthesis roles, and Table 16 the panel modes and synthesis rounds.

The Chair’s agenda contains generation metadata, at least several cross-peer hypotheses with minimal tests and kill/promote conditions, exactly C peer contracts, panel summary, consensus actions,

Table 11: Finding tuple fields for $\phi = (\text{intervention}, \text{outcome}, \text{evidence}, \tau, m, \alpha)$. τ and α name the method-level roles that inheritance is defined over. The released implementation files a finding under a task-facing record enum (`result`, `hypothesis`, `insight`, `challenge`, `error`) and carries the inheritance recommendation in next-step-intent and parent-usage metadata rather than in a single α column, so an implementer should read this table as the semantics to preserve, not as the literal field contract of the `share_finding` tool.

Field	Meaning
Intervention	What was changed or tested, expressed in the vocabulary of the design contract d_i .
Outcome	What happened under the evaluator or diagnostic protocol, including raw task-grounded evidence when available.
Evidence	Artifact reference, evaluation record, limitations, validity state, and links to parent evidence.
τ	Finding type: positive, negative, diagnostic, uncertain, or procedural.
m	Evidence maturity, inherited from the evidence-stage ladder in Table 10.
α	Recommended inheritance action: reuse, validate, avoid, diagnose, preserve, or archive.

Table 12: Inheritance actions used by finding extraction and agenda formation. As with τ , these are the method-level roles; the emitted agenda expresses them through a finer next-step-intent vocabulary that distinguishes, for example, repairing a failure mode from ablating or falsifying a mechanism.

Action	Agenda implication
Reuse	Treat the evidence as a parent, baseline, mechanism, or constraint when mature enough.
Validate	Schedule reproduction, ablation, or stronger evaluation before confirming the claim.
Avoid	Prevent later peers from repeating an invalid assumption, intervention, or procedure.
Diagnose	Allocate a focused diagnostic contract to explain a failure mode or uncertainty.
Preserve	Keep the lesson as durable memory even if it is not a current frontier parent.
Archive	Retain provenance while removing the item from active inheritance.

dissent-to-experiment conversions, minority high-upside ideas when justified, claim-boundary updates, and validation status. Peer-contract roles are execution roles rather than PI review roles; typical roles include exploit, falsifier, bridge, anti-mainline, and theorist.

A.5 Frontier Status and Agenda Decisions

The frontier assigns evidence an operational status for future generations. We use four method-level statuses throughout the paper; a task-family realization declares its own lane names and maps them to these inheritance roles, and a task that declares none falls back to a single primary-metric frontier. The trading campaign, for instance, declares *confirmed alpha*, *alpha incubator*, *benchmark floor*, and *diagnostic control*, while the SLAM and fusion campaigns declare no lanes and run on the

Table 13: Deep Innovation Gate (DIG) design-contract fields for structured exploration.

Field	Function
Selected candidate and variant	Identifies the candidate design and names the artifact variant that peer p_i should construct.
Design cell c_i	Assigns the contract to a cell $c_i =$ (mechanism family, intervention surface, intent) in the behavior space $\mathcal{C}$.
Intent	Purpose of the attempt within its design cell: exploit, repair, bridge, ablate, falsify, explore, diagnose, control, or audit—the same vocabulary balanced by the intent cap in Table 14.
Mechanism hypothesis	States the causal or functional mechanism expected to produce the outcome.
Why selected	Explains why the direction is worth one peer of the cohort budget under the current frontier, agenda, and Gems.
Rejected alternatives	Records at least several nearby alternatives so the lineage can distinguish intentional coverage from accidental omission.
Files or components to modify	Specifies the artifact surfaces that may be changed.
Allowed changes	Defines what edits, parameters, data transformations, proof steps, or experiment settings are in scope.
Forbidden changes	Defines changes that would invalidate the test, such as altering the evaluator, changing a data split, or modifying the metric calculation.
Implementation plan	Gives the peer a concrete construction path without overriding evaluator semantics.
Expected evidence signature	States what outcome pattern would support, weaken, or clarify the design intent.
Ablation and validation hooks	Specifies checks that make the result interpretable, including minimal reproductions, controls, or comparisons.
Fail-fast checks	Gives cheap checks to run before expensive artifact construction.
Semantic family and parent lineage	Records approach family and inherited evidence, including frontier entries, Gems, or independent starting points.
Novelty axis	Identifies what dimension of the design is intended to be new relative to the current cohort and frontier.

fallback. Table 17 lists the four statuses and their agenda effects, and Table 18 the agenda schema and lane-to-disposition mapping they feed.

Promotion is deliberately stricter than finding extraction. Finding extraction preserves useful evidence; promotion decides which evidence is allowed to shape future work. The promotion operator PROMOTE (Section 2.4) admits the top k findings by the primary metric, with $k = 2$ by default, and may admit one extra finding per task-defined anchor metric. Within each lane, candidates are selected under lane-specific axes rather than only global score. Hard gates reject findings that are marked non-promotable, lack required tier or maturity metadata, or remain at preliminary evidence stages such as smoke or scout. Preliminary evidence is preserved in research memory or diagnostic records but is not allowed to displace mature frontier evidence.

Table 14: Quantified Diversity (QD) cohort allocation constraints used to keep the C peers diverse. Fractions are applied to the cohort size and then interpreted as integer peer caps.

Constraint	Default cap	Purpose
Same formal design cell	1 peer	Prevent exact duplicate contracts unless a reproduction is explicitly requested.
Same mechanism family	0.34 of cohort	Avoid spending the generation on one approach family.
Same intervention surface	0.50 of cohort	Preserve coverage across where artifacts are changed.
Same intent	0.60 of cohort	Balance exploit, repair, bridge, ablate, falsify, explore, diagnose, control, and audit intents.
Same semantic family	0.34 of cohort	Reduce near-duplicate concepts even when their formal cells differ.
Same parent lineage	0.50 of cohort	Prevent one parent from monopolizing the generation.
Diagnostic contracts	≤ 0.20 of cohort and at most 2 peers	Preserve diagnostic work without allowing failure analysis to dominate the cohort.

Table 15: PI/Chair synthesis roles. Role names denote review functions; task families specialize the domain vocabulary used by each role.

Role	Synthesis function
Builder PI	Constructs the strongest evidence-backed mainline and proposes concrete next-generation tests.
Skeptic PI	Audits unsupported claims, fragile interpretations, invalid evidence, and risks that require checks.
Portfolio PI	Allocates effort across approach families, exploitation, exploration, validation, and diagnosis.
External-validity PI	Checks evidence boundaries, reproducibility, task constraints, and claims that may overfit to available feedback.
Chair	Merges PI memos into an agenda, resolves conflicts, narrows claims, and assigns peer contracts within the generation budget.

A.6 Gems and Selective Memory

Gems are compact lessons retained across longer horizons. Compression is an opt-in facility that is disabled unless a campaign turns it on; among the case studies reported here only the trading campaign runs it, and its released state records three resets with four active Gems. Gems are created at reset boundaries or other compression points from frontier-balanced evidence, recurring findings, validated mechanisms, rejected assumptions, diagnostic patterns, and procedural constraints. Gems complement the frontier: frontier entries assign inheritance status to current evidence, while Gems preserve durable knowledge that stays visible after local details are archived. Table 19 lists common Gem categories, and Table 20 the selective-memory routing that delivers it to each role.

Where enabled, compression is considered every $\rho = 6$ generations by default, with at most three

Table 16: Panel modes and synthesis rounds. The full panel is the default; high-stakes mode adds the external-validity role and an explicit confidence round.

Item	Contract
Mini panel	Builder PI and Skeptic PI. Used when the run needs lightweight mainline construction plus fragility audit.
Full panel	Builder PI, Skeptic PI, and Portfolio PI. Used as the default mode for balancing score, reliability, and coverage.
High-stakes panel	Full panel plus External-validity PI. Used when final claims, evidence boundaries, or reproducibility risks require extra scrutiny.
Round 0: evidence freeze	Builds the shared evidence core and role-private packs from Φ_g , $\mathcal{F}_{g+1}$, recent agendas, Gems, and lineage.
Round 1: independent memos	Each PI writes an independent interpretation before seeing other PI conclusions.
Round 2: anonymized cross-review	PI memos are cross-reviewed against fixed questions about support, fragility, validation, missing controls, and portfolio coverage.
Round 2.5: confidence boundary	High-stakes mode records confidence and claim boundaries before Chair arbitration.
Chair synthesis	The Chair merges memos and reviews into $\mathcal{A}_{g+1}$, validates agenda shape, and commits a single executable generation policy.

Table 17: Frontier statuses and their agenda effects.

Status	Meaning	Typical agenda effect
Confirmed ($\mathcal{F}^{\text{cf}}$)	Reliable inheritable evidence, mechanism, artifact, or constraint.	Continue: use as parent, baseline, comparison point, or constraint.
Candidate ($\mathcal{F}^{\text{cd}}$)	Promising evidence with incomplete maturity or incomplete coverage.	Explore or validate: reproduce, ablate, or combine with caution.
Diagnostic ($\mathcal{F}^{\text{dg}}$)	Failure mode, invalidated assumption, control, or process observation.	Stop, constrain, or diagnose: prevent repeated errors and schedule focused tests.
Validation ($\mathcal{F}^{\text{vl}}$)	Evidence requiring reproduction, ablation, or checking before promotion.	Validate: assign validation peers or define a minimal test.

reset events and a small active Gem set. The selection policy is frontier-lane balanced: confirmed evidence can yield reusable mechanisms, candidate or validation evidence can yield boundaries and follow-up lessons, and diagnostic evidence can yield failure-mode or procedural Gems. Compression archives ordinary findings after preserving the lessons needed for future inheritance, so later peers do not need to carry the full transcript of earlier generations.

A.7 Lineage Trace

PRAXIST records lineage as a typed trace over artifacts, evaluation records, findings, design contracts, PI/Chair decisions, frontier updates, agendas, and Gems. The trace is accumulated during the research cycle rather than reconstructed only after final selection. Each artifact can inherit from parent artifacts, findings, frontier entries, agenda assignments, design contracts, or Gems. Each finding points back to its supporting artifact and evaluation record. Each frontier update records

Table 18: Agenda schema and lane-to-disposition mapping.

Agenda field or disposition	Meaning
Generation metadata	Identifies which generation the agenda controls and which evidence freeze produced it.
Cross-peer hypotheses	Shared claims with minimal tests, kill conditions, and promote conditions.
Peer contracts	Exactly C executable contracts for the next generation, each with role, parent or independent target, success signal, and forbidden actions.
Panel summary and consensus actions	Chair-level synthesis of what the PI panel believes should continue, stop, validate, diagnose, or explore.
Claim-boundary updates	Narrowing statements that prevent later peers from overstating what the evidence supports.
Continue	Usually emitted for confirmed evidence that should be reused as a parent, baseline, or constraint.
Stop	Usually emitted for negative or diagnostic evidence that invalidates an assumption or weakens a direction.
Validate	Usually emitted for validation-lane evidence or high-upside candidates with incomplete maturity.
Explore	Usually emitted for under-covered design cells, minority high-upside ideas, or portfolio gaps.

Table 19: Common Gem categories. The categories are descriptive roles rather than a stored field: a Gem record carries its source finding, task lane, metric, evidence stage, and selection provenance, and its category is read from that content.

Category	Role
Validated mechanism	Preserves a reusable strategy, transformation, or design pattern across generations.
Rejected assumption	Records an assumption weakened by evidence so that later peers avoid relying on it.
Recurring failure mode	Summarizes a diagnostic pattern observed across attempts or generations.
Procedural constraint	Preserves task-specific or task-family-specific constraints needed for valid artifacts.
Evidence boundary	Records the scope within which a conclusion is supported.

how evidence became inheritable. Each agenda records how synthesis directed later work. Table 21 lists the recorded components, and Table 22 the typed edge vocabulary connecting them. Physically, the trace is written as several typed ledgers that share identifiers rather than as one serialized graph: the explicit typed edge list covers finding-to-finding relations, while artifacts are indexed in an artifact ledger, synthesis and promotion events in the run’s event trajectory, and agendas, frontier state, and Gems in per-generation state files. Reconstructing a full cross-object graph therefore means joining those ledgers on their shared identifiers.

The final output is a reproducible artifact together with the solution lineage that produced it. The run itself terminates by writing out the frontier, the lineage ledgers, and a run summary; Eq. 5 is the selection rule applied over those outputs at reporting time rather than a call the loop makes

Table 20: Selective-memory routing. Each role receives the subset of state it needs to act without treating every prior observation as equally inheritable.

Recipient	Routed context
Peer p_i	Agenda-relevant frontier entries, relevant Gems, parent lineage, the assigned design contract d_i , and task-family constraints.
PI role	Evidence pack for Φ_g , current frontier, relevant Gems, recent agenda history, and role-specific review questions.
Chair	PI memos, anonymized cross-review signals, agenda candidates, frontier state, and validation or dissent items requiring arbitration.
Future run or human reader	Final artifact a^* , confirmed frontier evidence, Gems, and lineage $\mathcal{L}_G$ explaining the solution path.

Table 21: Lineage trace components.

Component	Recorded relation
Artifact	Reproducible experimental unit produced or modified by a peer.
Evaluation record	Task-grounded outcome attached to an artifact.
Finding	Research claim supported by an artifact and evaluation record.
Design contract	Pre-artifact intent that motivated a peer’s attempt.
PI/Chair decision	Synthesis decision derived from generation-level evidence.
Frontier update	Inheritance-status decision for a finding, artifact, or evidence item.
Agenda	Generation-level policy scheduling future directions, validation, or constraints.
Gem	Durable compressed lesson retained across generations.

on its own, so a campaign stopped early still leaves a complete frontier and lineage from which the artifact is chosen. The lineage exposes which evidence supported the selected artifact, which failures constrained it, which candidates required validation, which Gems shaped later work, and which agendas directed its construction, so a reader can audit the reported result end to end.

A.8 Resource Scheduling and Mature-Evidence Debt

PRAXIST executes peer experiments through a thin resource-scheduling layer that operates within the generation’s wall-clock envelope (Section 2.3). The layer modeled here performs three functions: admission of experiments onto shared GPUs, lease accounting for acquired devices, and backfill of idle capacity. The deployed scheduler adds a coarse host-pressure guard on top of these, described under *Boundary and caveats* below. Launches are paced by a single scalar demand signal derived from evidence maturity, while all scientific interpretation stays outside the layer: maturity criteria, Pareto value, lineage, and plan selection belong to the research state of Section 2. This subsection specifies the resource model, the maturity signal, and the launch controller, then reports the paired event simulation used to validate the design. All quantitative claims below are simulation results under synthetic workloads, not measurements from the live runs of Section 3; symbols introduced here are local to this subsection.

GPU admission. Each experiment j declares, for each of the g_j GPUs it requires, an average utilization estimate $u_j \in [0, 1]$ (an nvidia-smi-style busy fraction, treated as a linear compute

Table 22: Lineage edge vocabulary. Edges are typed so that later peers and readers can distinguish inheritance, support, conflict, and update relations.

Edge type	Meaning
derived_from	The child artifact, finding, agenda, or Gem was constructed from the parent evidence.
supports	The source provides evidence in favor of the target claim, promotion, agenda item, or Gem.
challenges	The source weakens, falsifies, or limits the target claim or direction.
updates	The source revises the target, such as a later frontier entry superseding an earlier candidate.
related_to	The source is relevant to the target but does not imply support, challenge, derivation, or update.

capacity) and a peak-memory estimate v_j in GB. For every physical device d with memory capacity V_d , admission of the set $\mathcal{D}_j$ of experiments co-located on d requires

$$\sum_{j:d \in \mathcal{D}_j} u_j \leq 1, \quad \sum_{j:d \in \mathcal{D}_j} v_j \leq 0.95 V_d. \quad (6)$$

Multi-GPU experiments acquire their g_j devices as a gang, and placement favors devices that minimize the larger of the compute load and the fractional memory commitment, limiting both compute-load and memory fragmentation. Two low-utilization, low-memory experiments may share a GPU, while either dimension can independently block admission. This utilization-and-memory view follows goodput-oriented deep-learning cluster schedulers^{53,54,55}, reduced to the two quantities an experiment harness can cheaply declare.

CPU contention without core allocation. CPU allocation is delegated entirely to the operating system: the scheduler performs no per-experiment core packing and never rejects a GPU experiment because aggregate CPU demand exceeds the core count. Each experiment declares a runnable CPU demand w_j and a CPU sensitivity $a_j \in [0, 1]$. With host core count W , the host pressure at time t is $h(t) = \sum_{j \text{ running}} w_j / W$, and each running experiment’s remaining work is served at the processor-sharing slowdown

$$S_j(t) = 1 + a_j \max(0, h(t) - 1), \quad (7)$$

following the classical time-shared-systems abstraction⁵⁶: at $h = 2$ a fully CPU-sensitive experiment ($a_j = 1$) takes roughly twice its solo wall time, while a GPU-dominated one ($a_j = 0$) is unaffected. In this model CPU load is therefore a dynamic factor inflating critical time rather than an admission dimension; the deployed scheduler does consult observed host load, as noted below.

Maturity interface. The scheduler-facing notion of a *mature* result is a thin summary of the evidence-stage ladder of Appendix A.2: an experiment counts as mature once it has completed full-stage execution with sufficient independent evaluation coverage (in the simulator, a completed-work ratio of at least 0.75 and an evaluation-coverage ratio of at least 0.80), so that a nearly complete experiment is not discarded solely because of a categorical label. Scout-stage runs are preliminary probes that can be *continued* into mature evidence rather than restarted, in the spirit of multi-fidelity search^{57,58}, and a scientific negative result at full stage counts as mature evidence. The thresholds are simulation defaults, not domain-independent constants.

Mature-evidence debt controller. With cohort size C , the controller tracks a mature quota $Q = \max(1, \lceil C/4 \rceil)$, the number M_t of mature results completed by time t , and the outstanding debt

$$D_t = \max(0, Q - M_t), \quad (8)$$

(a task may raise Q above $\lceil C/4 \rceil$ through its own mature-evidence quorum), and targets $K_t = \min(C, 3D_t)$ mature-directed experiments in flight; actual launches still require GPU admission under Eq. 6. The factor of three gives bounded redundancy against failures and heavy-tailed durations: if one attempt succeeds before the deadline with probability p , then n independent attempts succeed with probability $1 - (1-p)^n$ (97.3% for $p = 0.7$, $n = 3$), consistent with tail-tolerant execution⁵⁹. The controller recomputes D_t at every completion, failure, and resource-release event, making it an event-driven receding-horizon feedback rule⁶⁰ rather than an online probabilistic program. The launch priority stack below is the simulated policy; the deployed scheduler implements its lease, first-wave, retry, deadline, and exploration-reserve components, while the ranking in step (ii) is performed by the research layer when it chooses which peer-owned plan to fill a lease with, not by a scheduler scoring function. Launch priorities are: (i) while mature-directed work in flight is below $3D_t$, continue an existing scout or launch a direct mature experiment; (ii) once the guarantee is supplied, rank follow-ups by expected marginal Pareto gain per unit of predicted time; (iii) launch a new scout when no follow-up is available; (iv) retry an infrastructure failure once; and (v) after the quota is met, admit only emergency mature work passing a 25% completion guard. The first wave mixes direct mature experiments and scouts, preserving an explorer whenever capacity permits, so exploration competes with time and hardware budgets without being eliminated⁶¹.

Reduced designs. To identify which parts of the controller are load-bearing, the simulation study compares it against a *minimal idle backfill* design: a work-conserving mechanism⁶² that, whenever a resource goes idle, issues a single-experiment lease from the fixed inventory of plans already owned by peers, with asynchronous refill similar in structure to massively parallel successive-halving systems⁵⁸. Backfill variants differ only in maturity feedback: *no feedback* (an idle lease selects a mature follow-up with fixed probability 25%, 50%, or 75%, for sensitivity analysis), a *Boolean maturity signal* (while $M_t < Q$, every idle waiter prioritizes a mature plan, so concurrent waiters can create implicit, accidental redundancy), and a *nonredundant thin token* (maintain at most $Q - M_t$ mature experiments in flight; quantitative but with no tail redundancy).

Simulation protocol. The comparison covers 512 randomly drawn scenarios spanning 2–16 peers, 1–16 GPUs with 16–80 GB memory, 4–192 host cores, log-normal heavy-tailed experiment durations (median minutes to three days), cold-start runtime bias, transient and persistent failures, varying scout cost and scout/true-quality correlation, and generation horizons from 3 hours to 4 days. Each of 8 policies is run 1,000 times per scenario with paired seeds, totaling 4,096,000 policy runs; 348 of the 512 scenarios satisfy a physical feasibility criterion (the 90th percentile of quota-completion time under a maximally aggressive full-only reference policy fits the horizon), and results are reported over these. Policy deltas use 5,000 scenario-level bootstrap samples. This study is a closed synthetic experiment: the scenario generator, its seed ledger, and the per-run outputs behind Tables 23 and 24 are not part of the released evidence package, so the two tables should be read as a design study of the admission and backfill layer under the stated model rather than as an independently replayable benchmark.

Results. Table 23 summarizes the comparison. Idle backfill alone raises mean GPU utilization from 10.00% to roughly 23.6% and CPU utilization from 11.71% to roughly 24.3%, directly fixing sustained hardware idleness; yet without maturity feedback, quota success still varies from 94.37%

Table 23: Simulated scheduling policies over 348 physically feasible scenarios. Quota success is the probability that a generation meets its mature-evidence quota Q before the horizon; P10 is the 10th percentile of per-scenario success.

Policy	Quota success (%)	P10 (%)	Mature mean	GPU util. (%)	CPU util. (%)
Passive peers (no backfill)	27.03	0.0	1.72	10.00	11.71
Idle backfill, 25% mature follow-up	94.37	79.2	16.99	23.62	24.30
Idle backfill, 50% mature follow-up	96.54	90.0	17.53	23.55	24.26
Idle backfill, 75% mature follow-up	97.64	94.5	17.93	23.50	24.24
Boolean maturity signal	98.65	97.6	17.78	23.55	24.30
Nonredundant thin token	98.36	96.6	17.67	23.55	24.29
Mature-evidence debt controller	99.85	99.8	18.42	23.56	24.37
Full-only (feasibility reference)	99.53	98.6	18.20	26.49	27.26

Table 24: Simulated quota success under operational resource pressure. Strata are defined by the observed state of the full-only reference within the same scenario.

Condition	Scenarios	Boolean (%)	Thin token (%)	Debt controller (%)
Low CPU pressure ($h < 0.35$)	232	99.21	99.10	99.94
High CPU pressure ($h \geq 1.20$)	52	95.74	94.45	99.49
CPU saturated over half the generation	59	96.20	95.05	99.55
Observed GPU utilization $\geq 50\%$	64	96.87	96.42	99.76
Observed GPU-memory utilization $\geq 50\%$	67	95.33	94.63	99.76

to 97.64% as the mature-follow-up probability changes. All feedback policies and the debt controller occupy statistically indistinguishable hardware (the Boolean policy differs from the debt controller by -0.012 GPU-utilization points, with a bootstrap interval crossing zero), but their evidence outcomes differ: the Boolean signal is -1.20 quota-success points below the debt controller (95% CI $[-1.82, -0.69]$) and the thin token is -1.49 points below (95% CI $[-2.17, -0.92]$). Equal utilization therefore does not imply equal evidence supply: backfill and maturity feedback close two separate loops, one preventing hardware idleness and one controlling whether high-grade results arrive before the deadline. Table 24 stratifies by operational pressure: the gap between the reduced designs and the debt controller grows from under one point at low CPU pressure to 3.75 points under high CPU pressure and 4.43 points under high GPU-memory pressure, where an all-scout first wave is more costly and the lack of tail redundancy in the thin token is exposed. An earlier version of the simulation that modeled each GPU as an exclusive integer slot understated these gaps (the Boolean gap was 0.22 points rather than 1.20), indicating that co-location, memory fragmentation, and CPU contention materially affect critical time. The debt controller’s own result was nearly unchanged across the two resource models.

Boundary and caveats. The production boundary of the layer follows the simulated design with one documented departure. As in the simulation, the harness supplies GPU count, average utilization, and peak-memory estimates (replaceable by runtime observations); GPU admission is decided by Eq. 6; no per-experiment CPU cores are allocated or packed, and no experiment is rejected merely because declared CPU demand exceeds the core count. The departure is that the deployed allocator also watches observed host pressure: it reduces its concurrency ceiling above roughly 92% CPU or 95% memory utilization and reports no supply headroom under sustained CPU, mem-

ory, or I/O pressure, which withholds new launches until the pressure clears. Observed host load is therefore a coarse gate on launch timing in production, even though it is not a declarative per-experiment admission dimension. The evidence layer publishes mature-evidence demand via Eq. 8 while the research layer selects which peer-owned plan fills an idle lease; exploration is preserved when capacity permits; and Pareto value, lineage, and maturity criteria stay outside the scheduler. The study’s limits are inherited by these conclusions: GPU utilization is treated as a linear capacity (no SM, Tensor-Core, memory-bandwidth, or interconnect modeling); CPU demand and sensitivity follow broad synthetic distributions under a processor-sharing approximation; the fixed plan inventory does not model how the quality distribution of newly generated directions evolves; and success rates are conditioned on physically feasible scenarios rather than unconditional.

A.9 Configuration Defaults and MLE-bench Budgets

Exact values are task configurable. The defaults in Table 25 are included to make the method concrete and to disambiguate the symbols used in the main text. They are starting points, not tuned settings: a campaign adjusts cohort size, generation count, and compression period to the task’s evaluation cost, and the frontier and Gem caps to the evidence each generation yields.

The finalized MLE-bench ledger draws all 75 retained task results from a single campaign, whose settings are listed in Table 26. Every task in that campaign runs `deepseek-v4-pro [1m]` as its research-agent model, with a cohort of 12 peers, promotes the top 4 peers at each generation boundary, and admits at most 20 generations. Experiments are scheduled one job per GPU, so each experiment occupies a single accelerator, and a task keeps up to eight experiments in flight at any moment. The generation ceiling, the per-experiment GPU budget, and the task wall cap are configured ceilings rather than measurements of realized compute.

A.10 Representative Prompt Contracts

PRAXIST uses role-specific prompt contracts to maintain the research-state structure described above. The templates below are representative method-level contracts, summarized in Table 27. They use placeholders for task-family content and omit task-specific commands or storage details. Their expected outputs are stated in the method vocabulary of this appendix; a deployment renders them against the task-facing field names noted in Tables 11 and 17, so these templates should be read as the contract’s semantics rather than as a directly executable field specification.

Example task prompt template. The following prompt sketch is adapted from the task-family templates used in our runs. It is an abridged method-level example: concrete commands, workspace paths, helper names, and storage conventions are replaced by placeholders. The structure is the important part: a task-specific source-of-truth block, inherited research state, a generation agenda, a peer-specific contract, and an evidence-reporting discipline.

Autonomous research peer. You are Peer <peer_id> in Generation <g> of a PRAXIST run for one evaluator-grounded research task. Coordinate with sibling peers through shared findings and build on previous generations through the current frontier, agenda, Gems, and lineage state. The task-specific block is the source of truth for the artifact definition, evaluator semantics, allowed evidence, and reporting requirements.

Task-specific source-of-truth block.

- **Task:** *task name*.

Table 25: Representative default configuration values.

Quantity	Default	Role
Generations G	8	Maximum number of generation boundaries in a run.
Cohort size C	5	Number of parallel peers per generation.
Promotions per generation	2 plus anchor extras	Baseline count of primary metric promotions before task-specific anchor additions.
Per-peer safety cap	5 hours	Hard upper bound for any single peer; generation closing is event driven.
Gem compression	Disabled	When enabled, resets every 6 generations, with at most 3 resets and 4 active Gems.
DIG candidate count	8	Number of candidate designs considered before cohort allocation.
DIG coverage targets	4 mechanism families, 3 intervention surfaces	Minimum diversity pressure for candidate generation.
Multi-PI panel	Disabled	Full mode is the default when enabled, using Builder, Skeptic, Portfolio, and Chair roles; high-stakes mode adds the External-validity role.
PI rounds	2 when enabled	Evidence freeze, independent memos, cross-review, and Chair synthesis.
Synthesis trigger	30 findings, 120 minutes, 3 peers; 240-minute cap	Typical information-density trigger with a maximum-time safety cap.
Evaluator seeds	42–46	Representative deterministic seeds for task evaluations that expose seeded protocols.

- **Objective:** *task objective and allowed scope.*
- **Artifact:** *minimal reproducible file set for one attempt.*
- **External evaluator:** *metric, success criterion, validity states, and score direction.*
- **Allowed context:** *public task context, allowed data, and allowed prior knowledge.*
- **Constraints:** *validity, resource, safety, or benchmark restrictions.*

Generation context. PRAXIST may provide selected frontier entries, Gems, peer-local memory, and a design contract. Use these channels to preserve diversity, reuse high-value evidence, and avoid repeated failures. Treat confirmed, candidate, diagnostic, and validation evidence according to its stated inheritance role. Do not edit frontier, Gem, agenda, or lineage state directly; publish evidence for PRAXIST to synthesize.

Panel agenda. If a PI/Chair agenda is available, use it to identify the current panel read: visible approach families, main risks, key tradeoffs, cross-peer hypotheses, validation targets, and directions to continue, stop, diagnose, or explore.

Your peer contract.

- **Role:** *explore, exploit, validate, or diagnose.*

Table 26: PRAXIST campaign configuration behind the finalized MLE-bench ledger.

Setting	Value
Research-agent model	deepseek-v4-pro [1m]
Cohort size	12 peers
Promotions per generation	Top 4
Maximum generations	20
GPU budget per experiment	6 GPU-h
Experiment scheduling	Single-GPU job, one job per GPU
Experiments in flight per task	Up to 8
Task wall cap	24 h (70 tasks); 36 h (5 tasks)

Table 27: Prompt families used by PRAXIST.

Prompt family	Primary input	Expected output
Peer research	Task context, inherited state, agenda, design contract, artifact constraints.	Reproducible artifact and artifact-grounded evidence.
Design allocation	Frontier, agenda, Gems, coverage state, prior design cells.	Peer design contracts covering diverse research directions.
Finding extraction	Artifact, evaluation record, peer notes, diagnostics.	Positive, negative, diagnostic, uncertain, or procedural findings.
Frontier update	Findings, evaluation records, prior frontier, validation needs.	Inheritance status and promotion or validation decisions.
PI memo	Evidence pack, frontier state, Gems, task constraints.	Role-specific interpretation, objections, proposed experiments, and peer contracts.
Chair agenda	PI memos, cross-reviews, evidence pack, frontier state.	Next-generation agenda with peer contracts and claim boundaries.
Gem compression	Durable findings, frontier entries, recurring patterns, archived evidence.	Compact Gems with scope, evidence, and future-use guidance.

- **Target hypothesis or objective:** *claim, candidate, or gap to test.*
- **Success signal:** *evaluator outcome, validation result, or diagnostic evidence.*
- **Forbidden assumptions:** *assumptions or shortcuts to avoid.*
- **Evidence stage:** *smoke, scout, full evaluation, or diagnostic.*
- **Next-step intent:** *reuse, validate, avoid, diagnose, or preserve.*

Core workflow.

1. Inspect the task block, inherited research state, and peer contract.
2. Choose a concrete work item aligned with the agenda and design intent.
3. Construct or revise a reproducible artifact.
4. Evaluate the artifact when it reaches the required evidence stage.
5. Convert the outcome into evidence with limitations and provenance.
6. Publish positive, negative, diagnostic, or uncertain findings.

7. Preserve artifact references so the lineage trace can connect this attempt to later frontier and agenda decisions.

Evidence reporting reminder. Report evaluator outcomes in their original task semantics. Mark fragile or partial evidence as candidate or validation-needed. Preserve useful failures as diagnostic evidence. Include the artifact reference, evaluator outcome, evidence stage, limitations, and recommended inheritance action when publishing findings.

For an MLE-style task, the placeholders are instantiated with the public competition description, sample submission structure, metric name and direction, submission-centered artifact definition, and official grader semantics. Other task families replace these fields with their own artifact and evaluator semantics while preserving the same prompt-level research contract.

Peer research contract.

Role. Construct a reproducible artifact for one evaluator-grounded task and publish evidence useful to later peers, PI/Chair synthesis, and the frontier.

Inputs. Public task context; artifact definition; evaluator semantics; current agenda; selected frontier entries; relevant Gems; parent lineage; optional design contract.

Required outputs. Artifact or artifact plan according to the task protocol; raw evaluator outcome when available; notes on attempted intervention, observed evidence, limitations, and useful failures.

Discipline. Use allowed task context and task-grounded outcomes. Report unsupported measurements as unavailable. Preserve failures when they clarify constraints.

Design allocation contract.

Objective. Allocate <k> peers for generation <g> using the current frontier, agenda, Gems, recent findings, and coverage state.

For each peer, specify. Mechanism family; intervention surface; intent; parent lineage or independent starting point; expected evidence signature; validation hook or fail-fast check; forbidden assumptions; success signal; recommended finding type.

Allocation rule. Cover multiple design cells, preserve validation and diagnostic needs, and use near-duplicate contracts only for explicit reproduction.

Finding extraction contract.

Objective. Convert an evaluated artifact into one or more artifact-grounded findings.

For each finding, report. Grounding artifact; attempted intervention; observed outcome; supporting evidence; finding type; maturity; limitations; inheritance action: reuse, validate, avoid, diagnose, preserve, or archive.

Discipline. Keep evaluator outcomes raw and task-grounded. Mark fragile claims as uncertain or validation-needed. Convert informative failures into diagnostic findings.

Frontier update contract.

Objective. Assign inheritance status to evidence using findings, evaluation records, prior frontier state, validation needs, and agenda priorities.

Decision fields. Frontier inclusion; status: confirmed, candidate, diagnostic, or validation; supporting evidence; promotion or validation condition; expected agenda effect.

Discipline. Use reliability and research utility to assign status. Preserve diagnostic evidence when it constrains future work. Schedule validation for promising but incomplete evidence.

PI memo contract.

Role. Write an independent memo from the assigned PI perspective using the shared evidence core, frontier state, recent findings, Gems, evaluator outcomes, and agenda history.

Report. Evidence-grounded claims; objections and claim boundaries; proposed experiments or validation checks; proposed peer contracts; confidence and highest-risk claims; evidence to promote, narrow, validate, or archive.

Perspectives. Builder emphasizes the strongest mainline; Skeptic audits unsupported or fragile claims; Portfolio balances effort across approaches; External-validity checks evidence boundaries, reproducibility, and task constraints.

Chair agenda contract.

Role. Merge PI memos and cross-review evidence into the next-generation agenda.

Produce. Panel summary; dominant mechanisms and main risks; cross-peer hypotheses with minimal tests; validation or falsification contracts; directions to continue, stop, validate, diagnose, or explore; peer contracts; claim-boundary updates and archive decisions.

Discipline. Use supplied evidence. Convert blocking objections into peer contracts, validation targets, or narrowed claim boundaries. Preserve auditable high-upside minority ideas when compatible with task constraints.

Gem compression contract.

Objective. Compress durable cross-generation knowledge from frontier-balanced evidence, recurring findings, validated mechanisms, rejected assumptions, diagnostic patterns, and procedural constraints.

For each Gem, report. Concise lesson; supporting evidence; scope and limitations; category; guidance for future peers; revisit or retirement condition.

Discipline. Prefer durable lessons over local details. Keep the active Gem set compact. Preserve provenance to artifacts, findings, and frontier decisions.

A.11 MLE-Style Task Example

MLE-style tasks provide one concrete instantiation of the method. The task context contains a public competition description, available data, sample submission format, metric name, and metric direction. A peer constructs an artifact centered on a submission file and the supporting files required to reproduce it. The external evaluator returns a task-grounded outcome. Findings preserve raw evaluator evidence, limitations, and inheritance recommendations. The frontier assigns evidence an inheritance status, PI/Chair synthesis emits the next agenda, Gems preserve durable lessons, and lineage records the path from earlier evidence to the final artifact.

This example illustrates the task-family realization boundary. Other task families can define different artifact forms and evaluator semantics while preserving the same PRAXIST cycle. The method requires traceable evaluated artifacts, artifact-grounded findings, explicit inheritance status, agenda-controlled future work, selective memory, and lineage accumulation.

A.12 Worked MLE-bench Research Trajectory

The Jigsaw Toxic Comment Classification task provides a compact trace from evaluated parents to the artifact that generation 1 selected; its primary metric is column-wise ROC AUC (higher is better). Table 28 reports that progression across generations 0 and 1.

Table 28: Research trajectory for the Jigsaw Toxic Comment Classification submission selected in generation 1. Scores are recorded column-wise ROC AUC values.

Stage	Evaluated artifact or intervention	Score	Role in the trajectory
Generation 0	Class-balanced focal-loss BERT	0.98620	Initial hypothesis
Generation 0	Matched BCE ablation, BERT seed 42	0.98657	Confirmed parent
Generation 1, C07	Metadata-only null model	0.76106	Diagnostic finding
Generation 1	Equal average of the two BERT seeds	0.98679	Two-model finding
Generation 1	Add DistilRoBERTa to the two-seed ensemble	0.98723	Three-model parent finding
Generation 1	Add BiLSTM to the three-model ensemble	0.98711	Negative finding
Generation 1	Add RoBERTa-CLS to the three-model ensemble	0.98749	Confirmed final selection

Generation 0 tested class-balanced focal loss, then retained the matched BCE ablation after its 0.00037 improvement. In generation 1, pod 6 completed the C07 null-model diagnostic, then reused preserved prediction artifacts to build the ensemble branch. Averaging the BCE parent with a second BERT seed reached 0.98679, and adding DistilRoBERTa raised the score to 0.98723. Adding the weaker BiLSTM component reduced the score by 0.00012; adding the independent RoBERTa-CLS parent instead raised it by 0.00026 to 0.98749. The final finding records an updates link to the three-model result and a derived_from link to the two-seed BERT ensemble, closing the lineage from a matched ablation through a diagnostic and a negative branch. The trace shows structure, not the task’s finalized entry: the release ledger reports the campaign’s highest-scoring integrity-clean attempt, which for Jigsaw is a later 0.9880 (Table 36).

B Experimental Setup Details

This appendix records the execution layer beneath Section 3. For each study, the main text defines the task, evaluator, and baselines; what follows states how many repetitions each reported number rests on, which hardware and time envelope produced it, and how the PRAXIST campaign that searched for the selected artifact was configured. The intent is that the replication depth and the cost of each headline result can be read off directly rather than inferred from the narrative; where a released ledger does not close—the SLAM timing field below is the clearest case, and the reported-tier note in the Quant subsection is another—we say so at that point rather than implying uniform coverage. The synthetic scheduling study of Appendix A.8 is a discrete-event simulation of the admission and backfill layer rather than a research campaign, and its setup is stated in full there.

Reporting conventions. Three distinctions recur below. A *configured* budget—GPU-hours per experiment, generation duration, or maximum generations—is a ceiling admitted by the scheduler, not measured consumption, and a campaign that terminates early consumes far less than its configuration allows. An *artifact wall span* is the interval between the first and last archived timestamp of a campaign; it includes evaluator time, synthesis time, idle capacity, interruptions, and resume gaps, so it bounds elapsed time rather than measuring compute. And “iteration” is ambiguous across these studies: PRAXIST generations, peer sessions, optimizer updates or epochs, simulator steps, and evaluation rollouts are separate quantities, so each is reported under its own name.

Resource-accounting scope. The descriptions distinguish configured scheduling ceilings, archived wall spans, task-level utilization, and dollar-equivalent model spend. These quantities measure different parts of an execution and are reported separately, not added into a synthetic total. Hardware attributions derived from run logs describe the host visible to the process, since pods can share physical devices. Table 29 summarizes the five studies; the subsections give the per-study detail.

B.1 MLE-bench

Execution environment. The PRAXIST task runs behind Section 3.2 executed on a shared pool of NVIDIA H100 80GB GPUs: each task run declares an 80 GB GPU-memory requirement, every experiment is a one-GPU job, and up to eight experiments may be in flight per task at once. Seventy tasks ran under a 24-hour wall cap and five under a 36-hour cap, each followed by a 3,600-second finalization grace period in which a task may still write out its submission. These are admitted ceilings rather than measurements of realized compute.

Campaign configuration. All 75 tasks ran under one campaign configuration with a cohort of 12 peers per task run; its generation and budget settings appear in Table 26. The ledger records 272 completed generations, with individual tasks completing between one and nine generations: 54 tasks ended on an external stop and 21 on the task wall limit. Summing the task-level elapsed fields gives roughly 1,157.5 task-run hours, but the tasks ran concurrently on the shared pool, so this figure is neither a campaign wall time nor a GPU-hour total.

Execution accounting. Each of the two arms in Table 2 received one locally run 75-task sweep on the shared H100 pool. The reported PRAXIST scores are read from the anti-cheat audited release ledger, in which every task retains its single best integrity-clean official submission, verified by SHA-256 against the run journal; the audit correction of 26 August 2026 settled that ledger at 49 gold. The PRAXIST and Claude Code + Opus 4.8 arms have finalized dollar-equivalent model-spend ledgers, itemized in the paragraphs below and reported only as resource context rather than as an input to medal assignment.

Table 29: Execution summary of the five studies in Section 3. Generation counts are committed generation boundaries, except for Fusion, whose legacy trajectory records completed generation result sets instead.

Study	Repetition behind the reported result	Execution record
MLE-bench	75 tasks; one finalized result per task; one benchmark sweep, so no repeated seeds and no medal-rate variance estimate.	Shared pool of H100 80GB GPUs; one GPU per experiment, up to eight experiments in flight per task; cohort of 12 peers; 24-hour task wall cap, 36 hours on 5 of the 75 tasks.
Rocket	Complete evaluation: 4,096 trajectories from each of the nominal, near-ODD, and hard-ODD banks = 12,288 landing trajectories, plus 1,024 frozen roll-disturbance cases. Separate post-run audit: all 40,960 rows of each bank = 122,880 landing trajectories.	16 peers, promote 4, 2-hour generation window, 0.0075 configured GPU-h per experiment; up to eight one-GPU evaluations on 8 H100 80GB GPUs; 12 committed generation boundaries, terminated by SIGTERM during generation 12 after 14.35 hours.
Quant	28 quarterly walk-forward windows, each retrained from scratch on the trailing 36 months; the five-seed tier evaluates 145 window-seed cells, but the reported artifact stopped at the one-seed, 29-cell tier.	12 peers, promote 4, 18-hour generation window, 18 configured GPU-h per experiment; 37 generation boundaries over a 176-hour artifact wall span.
SLAM	Fourteen NTU-VIRAL sequences, one accepted run per method and sequence, selected from a campaign containing replays and re-runs; no training seeds and no variance estimate. Candidates compile the pipeline and replay bags.	8 peers, promote 2, 5-hour generation window; sampled pods ran the CPU FAST-LIVO2 path on a 168-logical-core host with no NVIDIA device exposed.
Fusion	5 scenarios $\times$ 3 seeds $\times$ a 100-step horizon = 15 episodes and at most 1,500 survived steps per controller; no optimizer iterations.	5 peers, promote 2, 6-hour generation window; the reported comparison is three sidecar runs of 28.05, 30.32, and 30.64 minutes.

PRAXIST model spend. Token totals were 110,910,266,304 cache-hit input, 4,602,586,778 cache-miss input, and 1,108,180,075 output. Per-million-token prices for these categories were CNY 0.025/3/6 with deepseek-v4-pro and CNY 0.02/1/2 with deepseek-v4-flash, yielding category costs of CNY 2,770.38/11,766.49/6,157.97 and CNY 20,694.84 (approximately US\$3,054) in total. Both schedules appear because part of the token volume was served by the cheaper deepseek-v4-flash; deepseek-v4-pro [1m] is the research-agent model on every task.

B.2 Rocket

Plant, initial state, and actuation contract. The Rocket task of Section 3.3 freezes everything except the controller. Candidates are scored on the Swordfish C05 six-degree-of-freedom rigid body reached through the task’s frozen adapter, integrated with RK4 at one substep and a 0.1 s step for at most 900 steps, or 90 s of simulated flight; here C05 names that plant configuration and

nothing else. The vehicle carries 22,200 kg of dry mass and 7,000 kg of initial main propellant, so its initial total mass is 29,200 kg. Every episode starts at 2,000 m altitude with zero initial angular velocity and a fixed source quaternion, the one corresponding to the source bank’s $[0, 0, -20^\circ]$ Euler annotation, and the controller receives the exact simulated state, with no navigation estimator interposed between plant and controller. Table 30 collects these locks. The controller returns the normalized nine-channel action

$$a = \left[\frac{\delta_{g,y}}{0.0873}, \frac{\delta_{g,z}}{0.0873}, 2\eta - 1, \frac{\tau_{\text{RCS},x}}{\tau_{\text{auth}}(r)}, 0, 0, \frac{\delta_{f,y}}{0.35}, \frac{\delta_{f,z}}{0.35}, 0 \right],$$

clipped componentwise to $[-1, 1]$, in which δ_g are the two gimbal deflections, η is throttle, δ_f are the two grid-fin deflections, and $\tau_{\text{auth}}(r)$ is the roll-torque authority. Three channels are hard-locked to zero by the task contract: the two lateral reaction-control commands RCS_y and RCS_z , and the grid-fin roll command. The reaction-control system is therefore restricted to roll. Pitch and yaw remain fully actuated throughout, by engine gimbaling and by the grid fins.

Table 30: The frozen Rocket protocol. Every entry is fixed for the whole campaign: a candidate may change the controller, and nothing else.

Item	Frozen value
Plant	Swordfish C05 six-degree-of-freedom rigid body, through the task’s frozen adapter
Integration	RK4, one substep, 0.1 s step, at most 900 steps (90 s)
Dry mass	22,200 kg
Initial main propellant	7,000 kg
Initial total mass	29,200 kg
Initial altitude	2,000 m
Initial attitude	Fixed source quaternion, the source’s $[0, 0, -20^\circ]$ Euler annotation
Initial angular velocity	Zero
State feedback	Exact simulated state; no navigation estimator
Locked channels	$\text{RCS}_y = \text{RCS}_z = 0$ and grid-fin roll = 0
Scoring endpoint	Interpolated first landing-leg contact; zero post-contact scored steps

Landing banks. Initial states are drawn from three fixed source banks that share the base velocity $(-75, 20, -6) \text{ m s}^{-1}$ in the plant’s (x, y, z) convention, in which x is the vertical axis. The nominal bank places the vehicle area-uniformly on a disk of radius $r \leq 1,500 \text{ m}$ and leaves the base velocity unchanged. The near-OOD bank reuses the nominal positions, paired row by row, and multiplies all three velocity components by one scale $s \sim U[0.92, 0.98]$. The hard-OOD bank draws positions area-uniformly from the annulus $1,500 \leq r < 1,650 \text{ m}$ and scales all three velocity components by $s \sim U[1.02, 1.08]$. Complete evaluation uses a fixed 4,096-row subset of each bank. Two structural facts bound what the banks can show. Nominal and near-OOD are paired rather than independent, and hard-OOD moves radius and velocity jointly, isolating neither factor. The banks are also fixed and were adaptively reused across the campaign, visible to the research loop through aggregate development and complete metrics; the source label `nominal_unseen` is historical and does not indicate blind evaluation. Finally, the source rows store an initial mass of 29,000 kg, and both the

complete evaluator and the full-bank audit override only that field to the frozen 29,200 kg protocol value, preserving all other state components, the row order, and the mapping back to source rows.

Success predicate. Scoring stops at the interpolated state where the first landing-leg tip crosses the ground. A trajectory succeeds only if every one of the following holds jointly at that state: first contact is detected; all endpoint values are finite; horizontal error is at most 5.0 m; vertical center-of-mass velocity lies in $[-1.0, 0.0] \text{ m s}^{-1}$; contacting-leg downward speed is at most 1.0 m s^{-1} ; horizontal speed is at most 0.3 m s^{-1} ; tilt is at most 1.5° ; the roll rate satisfies $|\omega_x| \leq 0.02 \text{ rad s}^{-1}$; the pitch/yaw rate satisfies $\|\omega_{yz}\| \leq 0.03 \text{ rad s}^{-1}$; and the remaining-fuel fraction satisfies $(m - 22,200)/7,000 > 0.02$. A result counts as protocol-valid only if the forbidden-channel contract also passes. Four readings of this conjunction matter for interpreting the numbers we report. The fuel inequality is strict, so more than 140 kg of modeled main propellant must remain. Upward motion does not pass the vertical gate, since the admissible interval is closed at zero from above. Center-of-mass sink and contacting-leg sink are distinct quantities, because the leg-tip velocity includes rotational motion about the center of mass. And scoring genuinely ends at contact: `post_contact_scored_steps` is 0 and `gear_damping_credit_rate` is 0, so no post-contact spring-damper response can improve or degrade the scored state, and nothing here speaks to what the vehicle does after the leg touches. This task carries exactly one success predicate. The per-component gate rates reported alongside it decompose that conjunction as diagnostics; they are not alternative definitions of success, and no “standard” or “strict” variant gate exists in this task.

Two evidence scopes. The study keeps two evaluation scopes separate, and they answer different questions. Complete private validation v2 is the canonical scope: 12,288 landing trajectories, 4,096 from each bank, plus a frozen suite of 1,024 roll-disturbance cases, for 13,312 evaluation units in total. It is the scope in which the frozen task baseline and a candidate controller are compared at the same protocol version, and it is the only evidence used for promotion inside the run and for post-run controller selection. The full three-bank audit v1 is a separate, post-run scan of the frozen selected controller over all 40,960 rows of each bank, that is 122,880 landing trajectories, with no roll suite attached; it measures coverage of the selected artifact and is not promotion evidence, so no baseline delta may be computed inside it. One counting caution follows from the first scope: the 13,312 units are not 13,312 landings. Only 12,288 of them are landing trajectories, and every landing-success rate we quote at this scope has 12,288, not 13,312, in its denominator.

B.2.1 First-contact controller discovery, matched evaluation, and full-bank audit

Controller: guidance, phases, and the fuel-commit governor. The artifact is a deterministic hybrid powered-descent controller with no learned component anywhere in it. In phase P0 a rolling zero-effort-miss/zero-effort-velocity law⁴² scores a bank of time-to-go candidates on a cost combining a feasibility-violation term, a fuel proxy, a time term, and a jerk term; the implemented continuity term is switched off in this configuration by `tgo_continuity_weight` = 0, and the selected time-to-go is separately constrained to be monotone non-increasing across replans, which prevents the horizon from being pushed outward to cheapen the cost. Guidance runs at 10 Hz. The lateral horizon is floored at 20 s before capture and at 18 s inside the below-450 m capture layer. Phases P1 and P2 then follow a terminal descent corridor with a -15 m s^{-1} P1 floor and a height-scheduled lateral second-order PD law, and powered control continues all the way to the scored first leg contact rather than being released early into a ballistic settle. Two state-dependent guards carry much of the accumulated progress. A fuel-commit governor activates in P0 below 450 m once remaining main fuel falls under 80%, replacing an expensive near-hover target with a

two-band descent reference of about -22 m s^{-1} above 350 m and about -12 m s^{-1} below 250 m, linearly interpolated between the bands; while it is committed, the release gate tightens to a 3 m lateral radius, a 0.15 m s^{-1} lateral speed, and a 20-step dwell. A second branch keys on the *initial* horizontal radius: when $r_0 < 450 \text{ m}$ it requires $v_x > -15 \text{ m s}^{-1}$ before phase release and bypasses the generic axial acceleration slew while committed, with the 0.08-per-step throttle rate limit still active throughout. The corresponding code calls this field `far_bin`, but initializes it from the initial radius, so it is an $r_0 < 450 \text{ m}$ branch and not a far-radius branch; we name it by what it tests.

Controller: attitude authority and actuator budgets. Attitude tracking follows a geometric formulation on the rotation group⁴³, and the accumulated lineage spends most of its late gains on how much authority that loop is allowed. The P0 pitch/yaw natural frequency and damping are 1.0 and 0.7, the P0 attitude-reference rate limit is 6.5° s^{-1} , and a height schedule between 100 and 40 m raises the P0 pitch/yaw bandwidth by as much as $1.4\times$ as the vehicle enters the region where lateral residual motion has to be removed quickly. The non-emergency grid-fin budget is 0.24 rad, which sits deliberately inside the 0.25 rad threshold at which the audit would record grid saturation; the same strict-interior discipline governs the gimbal box described below. Because these budgets are set inside the audit thresholds by construction, the saturation rates we report should be read as evidence that the controller stays within its own declared envelope, not as a measurement of how much actuator authority the task leaves unused.

Closed-form pitch/yaw allocator. The newest method contribution in the lineage is a control allocator^{44,45} that splits each requested pitch or yaw torque between the gimbal and the grid fins. Unlike convex powered-descent guidance⁶³, which solves a trajectory-level program online, this module solves two independent two-variable box-constrained quadratic programs, one per axis, in closed form at every control step. Each program minimizes $(au_f + bu_g - d)^2 + wu_f^2$ subject to the boxes $u_f \in [u_f^{lo}, u_f^{hi}]$ and $u_g \in [u_g^{lo}, u_g^{hi}]$, where u_f is the grid-fin deflection, u_g the gimbal deflection, d the requested torque, and $w = 0.1$. The coefficients come from a frozen local torque approximation,

$$\tau_y = -G \delta_{f,z} + A \delta_{g,z}, \quad \tau_z = G \delta_{f,y} - A \delta_{g,y}, \quad G = x_f \bar{q} S C_L, \quad \bar{q} = \frac{1}{2} \rho \|v\|^2, \quad A = x_e T,$$

in which x_f and x_e are the grid-fin and engine moment arms, S is the reference area, C_L the fin lift coefficient, ρ the local air density, $\|v\|$ the vehicle speed, and T the current thrust; the bar in $\bar{q}$ distinguishes dynamic pressure from the attitude quaternion. The approximation is local and decouples the two axes, so it is an engineering linearization of the true coupled map rather than an exact inverse of it. Because only the grid deflection is regularized, the cost is gimbal-primary: the gimbal answers the demand first and the grid fins close the residual when the gimbal box binds. The implementation is not an iterative general-purpose solver. It enumerates five clipped KKT candidates—one interior point and four active-boundary points—and returns the feasible one of least cost, which makes the allocation deterministic and bounded in cost per step.

Actuator boxes, fallback, and roll. The two boxes carry both amplitude and per-step rate limits. The normal grid-fin amplitude is 0.24 rad; the generation-11 normal gimbal amplitude is $0.97 \times 0.075 = 0.07275 \text{ rad}$, and the audited generation-12 configuration widens it to $0.995 \times 0.075 = 0.074625 \text{ rad}$; the grid rate is capped at 0.05 rad and the gimbal rate at 0.015 rad per 0.1 s step; and the emergency absolute limits are 0.35 rad for the grid fins and 0.0873 rad for the gimbal. Nonfinite inputs or an empty feasible box trigger a byte-preserved heuristic allocator that predates the quadratic program, and setting `alloc_qp_enable = 0` selects that same fallback, which is what makes a direct one-key allocator ablation available on the identical frozen protocol. One

caveat must travel with the saturation numbers: both normal gimbal caps sit strictly below the 0.075 rad amplitude at which the audit records gimbal saturation, so a zero gimbal-saturation rate is in part constructed by that strict interior reserve and does not by itself demonstrate that authority went unused. Roll is handled outside this allocator entirely, by a damped roll-rate loop that drives only RCS_x with normal and emergency caps of 30 and 60 kN m, while the lateral reaction-control channels and the grid-fin roll command stay at exactly zero on both sides of the plant boundary.

Provenance: generation 11 versus generation 12. The method and the evaluated configuration are deliberately reported as two different objects. The method is the quadratic-program allocator committed at generation 11, inside the controller `gen11_peer5_satfree_h3g1_latauth_binlat_rate6p5_sinkguard_decouple_qpalloc_v1`, the last committed rank-1 frontier artifact of the run. The configuration we evaluate and audit is `gen12_peer15_qpalloc_cap0995_v1`, whose `controller.py` is byte-identical to that committed generation-11 controller and whose configuration differs in exactly one field, `alloc_gimbal_cap_frac`, from 0.97 to 0.995. The run committed twelve generation boundaries, generations 0 through 11, and was terminated during generation 12, so no generation-12 boundary exists; the cap-0.995 configuration is therefore the run’s current complete-evidence champion, selected post-run, and not a committed frontier artifact. Its selection carries little scientific weight in any case. The cap sweep holds 100% complete-protocol landing success and zero grid and gimbal saturation at 0.92, 0.97, 0.99, and 0.995 alike, and 0.995 lowers the sink-speed P95 by only about $1.09 \times 10^{-5} \text{ m s}^{-1}$ relative to 0.97: a deterministic lexicographic tie-break among equal-scoring configurations, not a method advance.

A strict generation-boundary alternative. Nothing in the headline depends on the uncommitted generation. Read under an evidence rule that admits only committed generation-boundary artifacts, the generation-11 configuration’s complete result is 12,288/12,288 landing successes, a descriptive Wilson 95% lower bound of 99.9687% on that fixed set, a sink-speed P95 of $0.326373 \text{ m s}^{-1}$, a lateral-speed P95 of $0.047199 \text{ m s}^{-1}$, a fuel P05 of 3.6701%, zero grid and gimbal saturation, a grid total variation of 0.046014 rad, and a gimbal total variation of 0.884548 rad. The full-bank audit remains an audit of the generation-12 configuration and stays labeled as post-run supporting evidence throughout. For reproducibility, the audited artifacts hash to b9a10ca54bc5, f4eb2d4d1429, 84f821c2f016, d6f82d5813b5, and 5b88d6e2e2b5 for, respectively, `controller.py`, its controller configuration, the variant manifest, the complete evaluation summary, and the matched baseline summary, each a SHA-256 digest truncated to its leading twelve hexadecimal characters.

Matched-protocol results. Table 31 reports the frozen task baseline and evaluated controller under the same protocol version, 13,312 scored units, and fixed rows of each bank. Landing success rises from 495/12,288 (4.0283%) to 12,288/12,288 (100%), a gain of 95.9717 percentage points, and the descriptive 95% Wilson lower bound on that fixed set rises from 3.6948% to 99.9687%. Both Wilson figures summarize a fixed, adaptively reused evaluation set; neither is a post-selection coverage guarantee for any population beyond it. The per-bank rows use the same 4,096 cases per bank throughout, and the worst-radius-bin row uses the same frozen radius strata, so no part of the comparison is purchased by resampling initial states.

Two features of the panel are easy to misread. First, the baseline already reached a 100% first-contact rate: every baseline trajectory brought a landing leg to the ground inside the horizon. What the controller changes is therefore the *quality* of that contact—the sink speed, lateral speed, lateral error, tilt, and remaining propellant at the interpolated first-contact state—and not whether contact occurs at all. The 88.6393% baseline fuel-depletion rate and the 66.3928 m s^{-1} baseline sink P95

Table 31: Panel A: matched complete-protocol comparison of the frozen task baseline against the evaluated controller. All rows use the same 12,288 landing trajectories, except the two roll rows and the coupling row, which use the frozen 1,024-case roll suite. Rows marked *adverse* are those on which the controller is worse than the baseline.

Metric	Frozen baseline	PRAXIST controller	Change
Landing successes	495/12,288	12,288/12,288	—
Landing success rate	4.0283%	100%	+95.9717 pp
Wilson 95% lower bound	3.6948%	99.9687%	descriptive
Nominal-bank success	5.6641%	100%	same 4,096 cases
Near-ODD-bank success	6.4209%	100%	same 4,096 cases
Hard-ODD-bank success	0%	100%	same 4,096 cases
Worst radius-bin success	0%	100%	same radius strata
First-contact rate	100%	100%	unchanged
Vertical joint-gate pass	11.4176%	100%	—
Fuel-gate pass	7.5846%	100%	—
Fuel depletion	88.6393%	0%	—
Fuel reserve, mean	0.4733%	9.7872%	—
Fuel reserve, P05	0%	3.6701%	1.6701 pp above gate
COM sink P95	66.3928 m/s	0.32636 m/s	99.51% lower
Contacting-leg sink P95	66.3987 m/s	0.33460 m/s	99.50% lower
Lateral speed P95	1.40592 m/s	0.04705 m/s	96.65% lower
Lateral error P95	7.09291 m	1.75437 m	75.27% lower
Tilt P95	4.56062°	0.35184°	92.29% lower
Grid saturation step rate	51.1584%	0	—
Gimbal saturation step rate	0.00950%	0	—
Grid total variation	7.94128 rad	0.04510 rad	99.43% lower
Gimbal total variation	0.49031 rad	0.88468 rad	80.43% higher (adverse)
Roll-stability rate	100%	100%	unchanged
Roll settling P95	2.5 s	2.5 s	unchanged
Roll-to-pitch/yaw coupling P95	0.29786 rad/s	0.55865 rad/s	87.55% higher (adverse)

make clear that most baseline contacts were uncontrolled arrivals rather than descents that narrowly missed a threshold. Second, two metrics move the wrong way. Gimbal total variation rises by 80.43% and the roll-to-pitch/yaw coupling P95 of the frozen roll suite rises by 87.55%, even though the roll-stability gate stays at 100% on all 1,024 of its cases. Both shifts are consistent with the controller moving pitch and yaw activity off the grid fins and onto the gimbal, which is exactly what the allocator is designed to do: grid total variation falls by 99.43% over the same set, and the grid saturation step rate falls from 51.1584% to zero. The trade is a quieter aerodynamic surface bought with a busier gimbal, and we report it as a cost rather than folding it into the headline.

Full-bank audit. The complete protocol scores a fixed 4,096-row subset of each bank. To measure how far the selected controller extends across the rest of those banks, we froze it after the run and evaluated all 40,960 rows of each bank, giving 122,880 landing trajectories. This is a post-run coverage scan, not PRAXIST promotion evidence: no controller was selected, ranked, or committed

on its output, and no baseline was audited at this scale, so Table 32 carries no baseline deltas. The three banks are separately defined—a nominal disk, a velocity-scaled near bank paired row-by-row with it, and a jointly faster and wider hard annulus—so they are not one identically distributed sample, and the pooled row is a descriptive sum only. Per-bank counts are the primary audit claim.

Table 32: Panel B: post-run all-row audit of the frozen selected controller over every row of each source bank. Per-bank counts are the primary claim; the pooled row is a descriptive sum.

Bank	Successes	Success rate	Wilson 95% LB	COM sink P95	Fuel P05	Min. successful fuel
Nominal	40,959/40,960	99.9975586%	99.9861709%	0.32283 m/s	5.7309%	4.6021%
Near-OOD	40,959/40,960	99.9975586%	99.9861709%	0.32778 m/s	5.9162%	4.8016%
Hard-OOD	40,960/40,960	100%	99.9906223%	0.32927 m/s	3.3543%	2.8908%
Pooled (descriptive only)	122,878/122,880	99.9983724%	99.9940652%	0.32636 m/s	3.6904%	2.8908%

All 122,880 trajectories reached first contact and passed the vertical and fuel component gates, and grid and gimbal saturation rates were zero throughout. Exactly two trajectories failed the joint predicate, both on the lateral-speed conjunct alone. Nominal row 29254 arrived with a lateral speed of $0.408226 \text{ m s}^{-1}$ against the 0.3 m s^{-1} bound, with COM and contacting-leg sink speeds of 0.157 and 0.178 m s^{-1} , a lateral error of 0.375 m , and 42.13% of its main propellant remaining. Near-OOD row 16173 arrived at $0.322299 \text{ m s}^{-1}$ lateral speed, with sink speeds of 0.629 and 0.633 m s^{-1} , a lateral error of 0.667 m , and 42.49% fuel remaining. Neither is an impact and neither is a fuel failure: both are residual lateral motion at an otherwise gentle, well-placed, propellant-rich first contact. Both also lie in the $[0, 450) \text{ m}$ initial-radius bin, so each nominal and near-OOD instance of that bin scores $3,686/3,687 = 99.972878\%$ while every other radius bin passes completely.

The audit’s integrity record is as follows. All 120 expected 1,024-row blocks completed, the bank-hash checks passed, no trajectory produced a nonfinite value or a forbidden actuator command, and an independent recomputation of the predicate from the stored first-contact endpoints matched the recorded success array exactly. One margin figure should temper how the hard-bank row is read: its minimum successful fuel reserve is 2.8908% , only 0.8908 percentage points above the strict 2% gate, and its P05 reserve of 3.3543% is the lowest of the three banks. A 100% count on that bank therefore records that no row crossed the gate on this fixed set, and must not be read as broad margin against disturbances the simulator does not model.

Allocator ablation. Disabling the allocator with `alloc_qp_enable=0` selects the byte-preserved heuristic allocator that preceded it, which makes the comparison in Table 33 a one-key ablation: the QP-off configuration and the committed generation-11 controller share the gimbal cap of 0.97 , differ in that single field, and were scored on the same frozen complete protocol. The allocator lowers COM sink P95 from 0.337891 to $0.326373 \text{ m s}^{-1}$ and lateral-speed P95 from 0.065620 to $0.047199 \text{ m s}^{-1}$, removes gimbal saturation, and cuts grid total variation by 99.5781% , at the cost of 2.1277% more gimbal total variation and a fuel-reserve P05 0.101879 percentage points lower.

The attribution this ablation supports is narrow. Its immediate heuristic parent was *already* at 100% landing success on the frozen complete bank, so the ablation gives no evidence that the allocator produced the $4.03\% \rightarrow 100\%$ success gain; that gain belongs to the accumulated lineage that preceded it. What the ablation does support is attributing the actuator-load, sink-speed, and lateral-speed effects to the allocator, because those quantities move under a single-field change with every other element of the controller, the plant, and the evaluation held fixed. The generation-12 cap change that produced the evaluated configuration is smaller still: sweeping the cap over

Table 33: One-key allocator ablation on the frozen complete protocol. The two configurations share the same 0.97 gimbal cap, differ only in `alloc_qp_enable`, and are both already at 100% landing success.

Metric	Allocator off	Allocator on	Change
Landing / hard / worst-bin success	100%	100%	ceiling retained
COM sink P95	0.337891 m/s	0.326373 m/s	3.4088% lower
Lateral speed P95	0.065620 m/s	0.047199 m/s	28.0727% lower
Gimbal saturation step rate	0.12303%	0	removed
Grid total variation	10.906433 rad	0.046014 rad	99.5781% lower
Gimbal total variation	0.866120 rad	0.884548 rad	2.1277% higher (adv.)
Fuel reserve P05	3.772024%	3.670145%	0.101879 pp lower (adv.)

0.92, 0.97, 0.99, and 0.995 retains 100% complete-protocol success and zero saturation at every setting, and 0.995 lowers sink-speed P95 by roughly $1.09 \times 10^{-5} \text{ m s}^{-1}$ relative to 0.97. It is a deterministic lexicographic tie-break among configurations already indistinguishable on complete-protocol success, and we report it as such, not as an advance in the controller’s method.

Lineage. Table 34 condenses the parent chain from the task baseline to the evaluated configuration, giving the main change accumulated at each stage together with the complete-protocol landing success measured there. The chain crosses the interesting thresholds early: the fuel-commit governor and faster guidance period take the controller from 4.0283% to roughly 49%, the generation-2 attitude retuning takes it to 98.3561%, and the last percent is closed by a sequence of narrow guards before the allocator arrives at generation 11 with success already saturated.

This table records cumulative evidence along one realized parent chain, not an additive decomposition of the total gain into per-mechanism contributions. The generation-1 variants each bundle more than one parameter change, several stages alter guidance and attitude behavior simultaneously, and no full factorial over the mechanisms was run, so the success figure at a stage attributes to everything accumulated up to it, not to the change named on its row. The one mechanism with a clean single-field control is the allocator, whose isolated effect is reported in Table 33.

Autonomous-optimizer baseline. The comparison this study reports is against another autonomous research system rather than against the artifact the task ships with. The same task package was given to Weco AI’s weco optimizer, an LLM-driven tree search over code built on the AIDE engine⁶, with the same objective metric `landing_success_rate` and the same 4.03% starting artifact⁴¹. That run evaluated 793 candidate controllers in 2 hours 37 minutes and reached a best reported score of 17.12%, a $4.25\times$ relative gain over that starting point which nonetheless leaves the task unsolved; its own search statistics record 676 improvements against only 43 breakthroughs, which suggests that most of its accepted candidates were marginal gains rather than step changes. The 17.12% figure is the best aggregate reported by Weco’s own run dashboard on that task and objective; the PRAXIST value is measured by the frozen 12,288-trajectory complete protocol. The systems use different underlying language models and wall-clock budgets, 2.6 hours against 14.35 hours, and Weco’s search space also spanned the task-variant manifest, whereas the PRAXIST protocol freezes the plant, contact model, integrator, evaluator, and data so a candidate can only change the controller. We therefore read Weco as an autonomous-optimizer reference on the same task and metric, not as a controlled cross-model comparison.

Table 34: Condensed lineage along the actual parent chain, with the complete-protocol landing success recorded at each stage.

Stage	Main accumulated change	Complete landing success
Baseline	Original SF-CAC v2 first-contact controller	4.0283%
Gen 1	Fuel-commit trigger, two-band descent reference, settled-release gate, and P0 lateral-horizon floor 28 $\rightarrow$ 20 s; then guidance period 0.2 $\rightarrow$ 0.1 s	47.9411% $\rightarrow$ 49.2920%
Gen 2	P0 pitch/yaw ω_n : 0.55 $\rightarrow$ 1.0, ζ : 0.9 $\rightarrow$ 0.7, and reference-rate limit 6 $\rightarrow$ 10° s ⁻¹	98.3561%
Gen 3–4	Reference-rate limit restored 10 $\rightarrow$ 6° s ⁻¹ ; low-altitude P0 bandwidth schedule added; grid budget capped at 0.24 rad	about 98.99%
Gen 6–8	Initial-radius/slew branch, 6.5° s ⁻¹ reference rate, and sink guard	99.9674% $\rightarrow$ 99.9919%
Gen 9	Remove the $r_0 < 450$ m branch’s fuel-commit vertical-speed conjunct while retaining axial-slew decoupling and the -15 m s ⁻¹ release sink guard	100%
Gen 11	Closed-form box-QP pitch/yaw allocator, cap 0.97	100%, with lower sink and lateral speed and zero grid/gimbal saturation
Gen 12	Gimbal cap 0.97 $\rightarrow$ 0.995 only	100%, marginal lexicographic tie-break

Campaign configuration. The campaign was configured for 30 generations with 16 peers per generation, a 2-hour generation window, and four promotions per generation. It closed 12 contiguous committed generation boundaries, generations 0 through 11. The process received SIGTERM while working on generation 12 after about 14.35 hours, so `run_summary.json` records `status: failed` with exit code 143; the result files that had already completed remain valid evidence, but no generation-12 boundary exists and the configuration evaluated above is consequently post-run rather than committed. The scheduler ran up to eight concurrent one-GPU evaluations on a host with eight NVIDIA H100 80GB GPUs, and each task experiment declared a configured budget of 0.0075 GPU-hours, one GPU, and 2 GiB of GPU memory. Its ledger records 697 completed jobs, 18 failed terminal records spanning 16 distinct jobs, and one rejected job; these are execution-accounting facts about job scheduling, not scientific sample counts, and they should not be added to or compared with trajectory counts. The research-agent model was DeepSeek V4 Pro with a 1M-token context through the Claude SDK runtime, and it is not a component of the controller: the artifact it produced contains no neural network, no learned parameters, and no inference call. Gem compression was disabled for this task, and the run produced no Gems.

Simulator scope and limitations. Every rate reported above is a property of a frozen low-order simulator evaluated on fixed initial-state banks with exact state feedback and no navigation esti-

mator. The audit varies none of wind, atmospheric density, mass error, thrust scale, inertia, drag, sensor noise or latency, actuator failures, the integration step, the initial height, the initial attitude, or the initial angular velocity: the only quantities that differ between trajectories are the initial horizontal position and the velocity scale that defines each bank. The disturbances that a flight design would have to absorb are therefore charged against no configuration here, so a controller tuned in this environment may allocate less authority to disturbance rejection than a real descent would demand. The banks compound this: they are fixed and were adaptively reused across generations, nominal and near-OOD are paired row-by-row rather than independent, and the hard bank changes radius and velocity jointly, so it isolates neither factor. Repeated selection against fixed evaluation sets carries a selection-overfitting risk that the audit’s breadth does not remove, because breadth here means more rows of the same three distributions rather than new ones.

The claim is also bounded in time and in what it charges. Scoring stops at the interpolated first landing-leg contact, so nothing above speaks to post-contact dwell, bounce, leg loads, slip, overturn, or terrain response; a trajectory satisfying the predicate has arrived acceptably, not landed stably. Reported fuel is modeled main-engine propellant only, and the reaction-control system’s propellant is uncharged, so no result here speaks to total propellant use. The allocator itself uses a decoupled local pitch/yaw torque approximation rather than a full nonlinear coupled allocation, which is what makes its closed-form solution possible and what limits its validity away from the operating points it was fitted around. The measured tradeoffs point the same way: gimbal total variation and roll-to-pitch/yaw coupling both rise relative to the baseline, and the hard bank’s minimum successful fuel reserve sits only 0.8908 percentage points above the gate. Taken together, these limitations mean the measured rates characterize this harness and these fixed banks, and cannot be read as real-world landing reliability, evidence of flight readiness, or any form of hardware validation.

B.3 Quant

Evaluation tiers. The walk-forward protocol of Section 3.4 is instantiated at three cost tiers over a fixed seed list of {42, 77, 11, 100, 7}. Tier T1 evaluates one seed across 29 cells—the 28 quarterly windows plus the excluded 2026 validation window—at a configured expectation of about 60 minutes; T2 evaluates three seeds across 87 cells at about 150 minutes; and T3, the campaign’s top tier, evaluates all five seeds across 145 cells at about 240 minutes. The artifact reported in Section 3.4 stopped at T1—one seed over 29 cells—and carries three hard constraint violations, so it is not clean-promotion eligible, which is why the main text presents it as a post hoc case-study selection rather than as the campaign’s promoted artifact. The campaign’s strongest confirmed-lane result is a different, generation-19 cross-sectional attention policy with a complete 145-cell T3 evaluation, no hard violations, and a lower CAGR. Hard evaluator timeouts are set at 6 hours for T1 and 18 hours for T2 and T3. Because every cell retrains the policy from scratch on its own trailing 36-month window, a T3 evaluation is 145 independent training runs rather than 145 inference passes, which is what makes the tier structure necessary.

Reported-policy configuration. The reported `gen15_peer10_diversify_repair_strong` policy receives a 12,505-dimensional observation, applies running normalization and a 256-dimensional projection, and updates a 128-dimensional LSTM state. Separate actor and value heads give 3,412,809 parameters. Training uses eight vectorized environments and 64-trading-day episodes. PPO uses $\gamma = 0.995$, a 0.1 clipping radius, three update reuses per iteration, a 3×10^{-4} learning rate, and a 0.001 entropy coefficient. Bounded replay retains five rollout batches,

samples four recurrent chunks of length 32, and adds replay at ratio 0.30 under a 0.15 clip. The concentration repair sets an effective-number proxy target of 12 and a maximum weight of 0.15. The deterministic adapter selects the top eight scores at temperature 0.16, targets gross exposure in $[0.80, 0.98]$, holds cash in $[0.02, 0.20]$, caps one name at 0.28, and applies top-five, top-ten, effective-number, turnover, and L1 liquidity limits. For task-cost context, a separate archived recurrent candidate’s complete 145-cell T3 evaluation took 54,141.8 seconds (15.04 hours).

Campaign configuration. The campaign ran DeepSeek V4 Pro through the agent SDK as a locally resumed run, with 12 peers per generation, at most 200 generations, four promotions per generation, an 18-hour generation window, and no plateau-based early stop. Each experiment declared 18 GPU-hours on one GPU per pod; Gem compression was enabled on a six-generation period with at most three resets and four active Gems. Synthesis opened on a 150-to-480-minute window once at least six pods had reported and used the two-round PI panel, with a Chair budget of 12 peer contracts. The run closed 37 generation boundaries (generations 0 through 36) with a partial generation 37 open at termination, and its archived timestamps span 176 hours and 30 minutes, ending on an external signal rather than at the configured generation cap. Run logs identify eight 80 GB H100 devices used to parallelize some multi-seed evaluations, which again records those evaluations rather than sustained campaign-wide occupancy.

B.4 SLAM

Measurement protocol. The comparison in Section 3.5 reports one accepted run per (method, sequence) pair over the fourteen sequences. There are no training seeds and hence no run-to-run variance estimate, but the campaign is not a single pre-registered pass either: several sequences were replayed or re-run—for example `rtp_01` on the COVSCHED side and `eee_02` on the baseline side—and the table takes the accepted run rather than the first one, without a pre-declared acceptance rule. This reflects the artifact’s nature rather than an economy—the candidate is a compiled variant of a deterministic estimator, not a trained policy, so a peer’s inner loop is a build followed by a bag replay, and the evaluation units are 100-second windows during search and full-sequence playbacks at validation. Trajectory error is APE RMSE after rigid Umeyama alignment to the Lica ground truth with a 0.02-second association tolerance, visual-update counts are parsed from the runs’ launch logs, and the reported timing field is the evaluator’s last-captured cumulative visual-path (VIO) average, not a measured end-to-end wall time per LiDAR frame.

Mechanism parameters and their qualifications. The champion configuration sets `PVTR_MODE=8` (observability-gated visual scheduler) and `VMAP_DEDUP=1` (visual-map admission filter). Two implementation details qualify the description in Section 3.5. First, the dwell counter decimates frames that stay degenerate for a long time rather than protecting them, so it *raises* skip demand; what prevents permanent visual starvation is the saturation of the skip budget below one together with the hard cap of three consecutively skipped frames. The net schedule is still strongly protective of low-observability frames, because only a handful of frames ever exceed the dwell threshold. Second, the admission filter deduplicates by rejection rather than fusion: a candidate within 0.08 m and 15° in normal of an existing map point in the same voxel is discarded, which is why the run reports admitted and retrieved point counts instead of a merged-observation statistic.

Further caveats on the comparison. Beyond the timestamp confound treated in Section 3.5, three caveats bound the result. APE is computed over ground-truth-associated poses, and those sets are not identical between the arms: COVSCHED’s associated-sample ratio averages 1.03 with median

0.985 but ranges from $0.79\times$ on `eee_03` to $1.76\times$ on `nya_02`, so the full-sequence comparison should be read with coverage. The effect is also primarily a full-trajectory and resource-efficiency one rather than a uniform improvement of every local statistic: relative pose error over short 1–30 s windows is mixed across sequences. Finally, the component controls cover one to three sequences, not fourteen, and the rate-matched periodic-skip control disables the map-admission filter that COVSCHED keeps enabled, differing in two respects at once: they show that generic visual thinning is already a strong baseline, but do not separate the LiDAR-observability signal from the map-admission filter, which the paired re-run listed in Section 3.5 would supply.

Host provenance. Sampled pod logs from the campaign show the CPU FAST-LIVO2 build path: 168 logical cores visible, roughly 25 GiB of pod memory, ROS Noetic, no `nvidia-smi` present, and no NVIDIA device exposed to a direct probe. That characterizes the containers we sampled rather than every process in the campaign, and because pods share a physical machine the reported per-frame timings are indicative of the relative cost of the two visual-update policies rather than isolated-benchmark measurements of either. The timing field itself is weaker than a clean ratio: the evaluator scans the launch log for the last line named Average Total Time, which both the LiDAR and the visual thread emit, so it captures whichever thread wrote last; skipped frames enter the visual average as zero time; and the scheduler’s own eigenvalue computation falls outside the timed region. Separating the two threads over the fourteen accepted runs gives a 72.4% mean reduction on the visual path and a 1.0% mean *increase* on the LiDAR path, with the summed internally timed thread-wall work falling 22.1%. We therefore report the visual-path reduction and that aggregate proxy, and we do not treat the ratio as an end-to-end latency, throughput, or CPU-load measurement; the absolute milliseconds are host-dependent.

Campaign configuration. The campaign used DeepSeek V4 Pro via the cheap-pod command-line coding-agent profile, with 8 peers per generation, at most 200 generations, two promotions per generation, and a 5-hour generation window. Its effective specification declares no compute budget, no Gem configuration, and no evaluator seed list, so design allocation and Gem behavior fall back to the system defaults of Table 25. Per-experiment GPU budgeting has no entry in that table and none in this campaign’s specification; the only GPU figure in its ledger is a stage-level request of 3,200 GPU-hours, which is not a per-experiment realized budget. The deterministic evaluator uses no seeds. The archived active-run snapshot spans 43 generation boundaries (generations 0 through 42) over 145 hours 52 minutes.

B.5 Fusion

Evaluation protocol and cost. The benchmark of Section 3.6 runs 5 scenarios $\times$ 3 seeds (0, 1, and 2) $\times$ a 100-step horizon, so a controller is scored on 15 episodes and at most 1,500 survived simulator steps. The selected artifact is synthesized Torch control code rather than a trained policy, so there are no optimizer epochs or policy-gradient iterations anywhere in this study: the meaningful iteration counts are generations, peer sessions, episodes, and simulator steps. The three runs behind Table 7 took 1,682.7, 1,819.4, and 1,838.7 seconds of runner wall time for zero feed-back, the PCS-style controller, and the PRAXIST controller respectively—28.05, 30.32, and 30.64 minutes—which makes the whole reported comparison a sub-two-hour evaluation.

Hardware attribution. Representative official evaluations from the same archived campaign show four NVIDIA H100 80GB devices visible on the host. This is host context rather than workload

occupancy: FreeGSNKE evaluation is CPU-heavy and process-parallel, and the configured 8 GPU-hours per experiment is a scheduling reservation rather than a realized-compute measure.

Campaign configuration. The lineage that produced HybridJacobianPDV1 ran DeepSeek V4 Pro through OpenRouter under the agent SDK as a local run, configured for at most 12 generations with 5 peers, two promotions per generation, and a nominal 6-hour generation window; each experiment declared 8 GPU-hours, 20 GB of memory, a half-device utilization request, 16 CPU cores, and at most five parallel jobs per pod. The run did not reach its configured cap: it completed 7 generation result sets (generations 0 through 6) with all five pod slots filled in each, recording 114 peer sessions and seven synthesis sessions before termination. This is a legacy trajectory that predates the current boundary object, so those generations are attested by their result sets and syntheses rather than by `generation_boundary.json` files. Start to termination spans about 18 hours and 15 minutes, of which the summed generation durations account for about 17.15 hours. The selected controller first appears in generation 0, and the remaining six generations refined and tested it.

Full-horizon and common-horizon precision. Table 35 gives both precision metrics per scenario, so that the horizon choice discussed in Section 3.6 can be checked rather than taken on trust. For the zero-feedback baseline the two coincide by construction, since the common horizon is capped at that controller’s own survival length. The two metrics disagree about which closed-loop controller is more precise: on the common horizon PRAXIST leads everywhere except the certification scenario, while on the full horizon the PCS-style controller leads on three of the four perturbed scenarios and on the aggregate, and loses the certification scenario by an order of magnitude. Neither ordering is the benchmark’s pass rule, which no controller here satisfies.

Table 35: Fusion case study: full-horizon and common-horizon WNRMSSE p_{95} (lower is better) per benchmark scenario. The full horizon is the benchmark’s original definition, scoring every step a controller survives; the common horizon caps each episode at the zero-feedback baseline’s survival length for that scenario and seed. Aggregate rows are pooled over all scored steps rather than averaged across scenarios.

Scenario	Zero-feedback	MAST-U PCS-style		PRAXIST (ours)	
	full = common	full	common	full	common
main_shape	4.77	4.18	2.63	8.50	2.55
axis_ip	6.10	5.76	3.54	6.26	3.48
ip_shape	5.12	3.37	2.63	3.22	2.43
xpoint	3.85	2.87	1.38	2.98	1.21
cert_shape	4.74	13.46	2.76	3.76	2.81
Aggregate	4.89	4.42	2.99	4.65	2.86

Runner and reset sensitivity. The same controller scores substantially differently under different environment lifecycles, so every Fusion figure must be read together with the runner that produced it. Under the fresh-reset, action-clipping sidecar used for Table 7, the selected controller survives 1,264 steps with a 0.667 full-horizon completion rate; a second fresh validation gives 1,298 and 0.733; and the default in-run evaluator, which resets once and deep-copies the post-reset environment for each episode, gives 894 and 0.200. The reported protocol builds and resets a fresh environment for every episode and is applied identically to all three controllers in Table 7, so the comparison is internally consistent even though its absolute levels are lifecycle-dependent.

C Full MLE-bench Per-Task Results

This appendix reports the complete benchmark comparison across all 75 competitions, together with the integrity adjudication that stands behind the PRAXIST column. Tables 36 and 37 report the complete per-task comparison between the Claude Code + Opus 4.8 baseline and PRAXIST (base model deepseek-v4-pro) on all 75 MLE-bench competitions, grouped by task category. Conventions follow Table 3: cell shading encodes the medal earned by that submission (gold, silver, bronze; unshaded = no medal; not_scored = not scored), and the better score of each scored pair, in the direction of the task metric, is shown in bold. Tasks are sorted alphabetically within each category and numbered consecutively (1–75) in display order; the # column gives this index, which is used for cross-reference throughout the paper. Scores are shown to at most four digits; bold is assigned on the full-precision values, so two displayed-equal scores may still differ. Medal labels are assigned by the MLE-bench grading-report thresholds. The Claude Code + Opus 4.8 scores are taken from the finalized 75-task ledger after the benchmark’s post-run screen for severe-cheating violations: 70 tasks retain accepted scores and five are not scored. The associated finalized dollar-equivalent cost ledger records US\$38,370 for the full 75-task sweep; cost is reported only as resource context and is not used for per-task medal assignment.

The PRAXIST scores are taken from the corrected release ledger, whose recorded selection policy is the highest canonical official clean attempt by metric direction: for each task the highest-scoring official attempt that passes an integrity adjudication is reported, and the payload of the selected submission is verified by SHA-256 against the run journal. The adjudication assigns one of three statuses to every task. `live_best_verified` (60 tasks) retains the best official attempt with its payload verified against the journal. `fresh_lineage_reviewed_clean` (6 tasks: #28, #41, #51, #72, #74, #75) marks a lineage that was re-reviewed in this pass and cleared. `clean_fallback` (9 tasks) marks a task whose previously best attempt belonged to a contaminated lineage; that lineage is excluded in full and a clean actor’s submission is substituted in its place. Across the campaign 90,423 attempts were rejected, 85,960 of them under `integrity_policy` and 4,463 as `canonical_invalid`. Across the nine clean-fallback tasks the substituted submission still earns gold on five, falls to silver on two, and earns no medal on two. This adjudication governs the PRAXIST ledger only, and is separate from the benchmark’s own post-run screen for severe-cheating violations described above, which applies to the Claude Code + Opus 4.8 arm. Both arms used the same hardware: NVIDIA H100 80GB accelerators, with each experiment run as a single-GPU job (one job per GPU) and up to eight experiments in flight per task. The tier denominators used throughout are the 22, 38, and 15 tasks of the Low, Medium, and High complexity splits, which are the denominators behind every per-tier rate reported for this benchmark.

Table 36: Full MLE-bench per-task comparison (Part I of II): image classification, image-to-image, and text classification. Shading encodes the medal (gold, silver, bronze; unshaded = none); bold marks the better score in the metric’s direction, assigned on full-precision values.

Category	#	Task	Metric	Claude Code	PRAXIST (ours)
Image Classification	1	aerial-cactus-identification	AUROC $\uparrow$	1.000	1.000
	2	alaska2-image-steganalysis	weighted AUROC $\uparrow$	0.9260	0.9140
	3	aptos2019-blindness-detection	quadratic kappa $\uparrow$	0.9354	0.9320
	4	cassava-leaf-disease-classification	accuracy $\uparrow$	0.9010	0.9021
	5	cdiscout-image-classification-challenge	accuracy $\uparrow$	0.7661	0.6138
	6	dog-breed-identification	log loss $\downarrow$	0.1886	0.1162
	7	dogs-vs-cats-redux-kernels-edition	log loss $\downarrow$	0.0074	0.0010
	8	herbarium-2020-fgvc7	macro F1 $\uparrow$	0.4131	0.6396
	9	herbarium-2021-fgvc8	macro F1 $\uparrow$	0.5121	0.5444
	10	herbarium-2022-fgvc9	macro F1 $\uparrow$	0.8024	0.8079
	11	hms-harmful-brain-activity-classification	KL divergence $\downarrow$	0.5135	0.3854
	12	hotel-id-2021-fgvc8	mAP@5 $\uparrow$	0.7929	0.7368
	13	imet-2020-fgvc7	micro F1 $\uparrow$	0.6848	0.6882
	14	inaturalist-2019-fgvc6	top-1 error $\downarrow$	not_scored	0.1427
	15	iwildcam-2019-fgvc6	macro F1 $\uparrow$	0.6934	0.4092
	16	iwildcam-2020-fgvc7	accuracy $\uparrow$	0.8650	0.8036
	17	kuzushiji-recognition	F1 $\uparrow$	0.9714	0.9741
	18	leaf-classification	log loss $\downarrow$	0.0015	0.0000
	19	plant-pathology-2020-fgvc7	mean col. AUROC $\uparrow$	0.9979	0.9978
	20	plant-pathology-2021-fgvc8	micro F1 $\uparrow$	0.9409	0.9351
	21	ranzcr-clip-catheter-line-classification	AUROC $\uparrow$	0.9725	0.9737
	22	rsna-2022-cervical-spine-fracture-detection	weighted log loss $\downarrow$	0.5313	0.5458
	23	rsna-breast-cancer-detection	probabilistic F1 $\uparrow$	0.4719	0.2752
	24	siim-isic-melanoma-classification	AUROC $\uparrow$	0.9378	0.9461
	25	statoil-iceberg-classifier-challenge	log loss $\downarrow$	0.1451	0.1271
	26	whale-categorization-playground	mAP@5 $\uparrow$	0.5844	0.5983
Image to Image	27	denoising-dirty-documents	RMSE $\downarrow$	0.0065	0.0070
	28	vesuvius-challenge-ink-detection	F _{0.5} $\uparrow$	0.4830	0.4659
Text Classification	29	AI4Code	Kendall τ $\uparrow$	0.8606	0.8227
	30	detecting-insults-in-social-commentary	AUROC $\uparrow$	0.9582	0.9590
	31	facebook-recruiting-iii-keyword-extraction	micro F1 $\uparrow$	0.7851	0.7959
	32	jigsaw-toxic-comment-classification-challenge	mean col. AUROC $\uparrow$	0.9883	0.9880
	33	jigsaw-unintended-bias-in-toxicity-classification	bias-weighted AUC $\uparrow$	0.8635	0.8640
	34	learning-agency-lab-automated-essay-scoring-2	quadratic kappa $\uparrow$	0.8312	0.8384
	35	lmsys-chatbot-arena	log loss $\downarrow$	0.8718	0.9812
	36	random-acts-of-pizza	AUROC $\uparrow$	not_scored	0.8414
	37	spooky-author-identification	log loss $\downarrow$	0.2011	0.1633
	38	tweet-sentiment-extraction	Jaccard $\uparrow$	0.7225	0.7239

Table 37: Full MLE-bench per-task comparison (Part II of II): remaining task categories. Conventions follow Table 36.

Category	#	Task	Metric	Claude Code	PRAXIST (ours)
Image (Other)	39	histopathologic-cancer-detection	AUROC $\uparrow$	0.9987	0.9979
	40	petfinder-pawpularity-score	RMSE $\downarrow$	16.77	16.86
	41	rsna-miccai-brain-tumor-radiogenomic-classification	AUROC $\uparrow$	0.5341	0.6588
Audio Classification	42	freesound-audio-tagging-2019	LRAP $\uparrow$	0.7471	0.7445
	43	mlsp-2013-birds	AUROC $\uparrow$	0.9603	0.9555
	44	tensorflow-speech-recognition-challenge	accuracy $\uparrow$	0.9796	0.9736
	45	the-icml-2013-whale-challenge-right-whale-redux	AUROC $\uparrow$	0.9937	0.9920
Tabular	46	champs-scalar-coupling	log MAE $\downarrow$	-3.053	-1.7129
	47	h-and-m-personalized-fashion-recommendations	mAP@12 $\uparrow$	not_scored	0.0334
	48	icecube-neutrinos-in-deep-ice	angular error $\downarrow$	1.005	1.0563
	49	new-york-city-taxi-fare-prediction	RMSE $\downarrow$	3.830	2.7477
	50	nomad2018-predict-transparent-conductors	mean col. RMSLE $\downarrow$	0.0500	0.0497
	51	smartphone-decimeter-2022	haversine dist. $\downarrow$	6.284	4.4584
	52	stanford-covid-vaccine	log loss $\downarrow$	0.2013	0.2245
	53	tabular-playground-series-dec-2021	accuracy $\uparrow$	0.9635	0.9629
	54	tabular-playground-series-may-2022	AUROC $\uparrow$	0.9980	0.9968
Sequence to Sequence	55	text-normalization-challenge-english-language	accuracy $\uparrow$	0.9992	0.9984
	56	text-normalization-challenge-russian-language	accuracy $\uparrow$	0.9956	0.9909
Training LLMs	57	billion-word-imputation	Levenshtein $\downarrow$	2.506	5.3056
	58	chaii-hindi-and-tamil-question-answering	word Jaccard $\uparrow$	0.7585	0.8894
	59	google-quest-challenge	mean col. Spearman $\uparrow$	0.4537	0.4469
Forecasting	60	osic-pulmonary-fibrosis-progression	Laplace log-lik. $\uparrow$	-7.036	-5.6407
	61	ventilator-pressure-prediction	MAE $\downarrow$	not_scored	0.1515
Image Segmentation	62	3d-object-detection-for-autonomous-vehicles	mAP $\uparrow$	0.4322	0.1520
	63	google-research-identify-contrails-reduce-global-warming	global Dice $\uparrow$	0.6805	0.6839
	64	hubmap-kidney-segmentation	Dice $\uparrow$	0.9487	0.9488
	65	multi-modal-gesture-recognition	Levenshtein $\downarrow$	not_scored	0.0337
	66	tgs-salt-identification-challenge	mean IoU precision $\uparrow$	0.8596	0.8764
	67	uw-madison-gi-tract-image-segmentation	Dice-Hausdorff $\uparrow$	0.6274	0.8722
Signal Processing	68	predict-volcanic-eruptions-ingv-oe	MAE $\downarrow$	5.176×10^5	1.228×10^6
	69	seti-breakthrough-listen	AUROC $\uparrow$	0.8826	0.8164
Text (Other)	70	tensorflow2-question-answering	micro F1 $\uparrow$	0.6929	0.6536
	71	us-patent-phrase-to-phrase-matching	Pearson r $\uparrow$	0.8568	0.8739
Video Classification	72	nfl-player-contact-detection	MCC $\uparrow$	0.7128	0.7285
Image to Text	73	bms-molecular-translation	Levenshtein $\downarrow$	24.16	25.04
Object Detection	74	siim-covid19-detection	mAP $\uparrow$	0.5947	0.6245
	75	vinbigdata-chest-xray-abnormalities-detection	mAP@IoU>0.4 $\uparrow$	0.5044	0.4086