SHELDON: REASONING THROUGH DUAL SEARCH OVER HYPOTHESES AND OBSERVATION SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time scaling has enabled large language models to solve increasingly complex reasoning, discovery, and optimization tasks, but existing approaches often suffer from fixed search strategies, or hypothesis-centric reasoning that under-explores informative evidence. Inspired by the hypothetico-deductive method (Hempel, 1945) and dual-space models of scientific discovery (Klahr & Dunbar, 1988), we introduce SHELDON, a scientific reasoning harness that jointly models hypothesis and observation spaces. By treating observations as first-class objects, SHELDON can identify gaps in its current understanding, uncover new hypotheses, and design investigations that better distinguish among alternatives. Across diagnosis, abstract reasoning, and optimization benchmarks, SHELDON outperforms evaluated baselines by upto 26.7 points. Moreover, on system root cause analysis tasks, SHELDON discovers 29.8% points more ground-truth causes and investigates 34.4% points more discovered causes than other baselines, demonstrating that joint hypothesis-observation search leads to broader and more effective scientific reasoning at test time.

1 INTRODUCTION

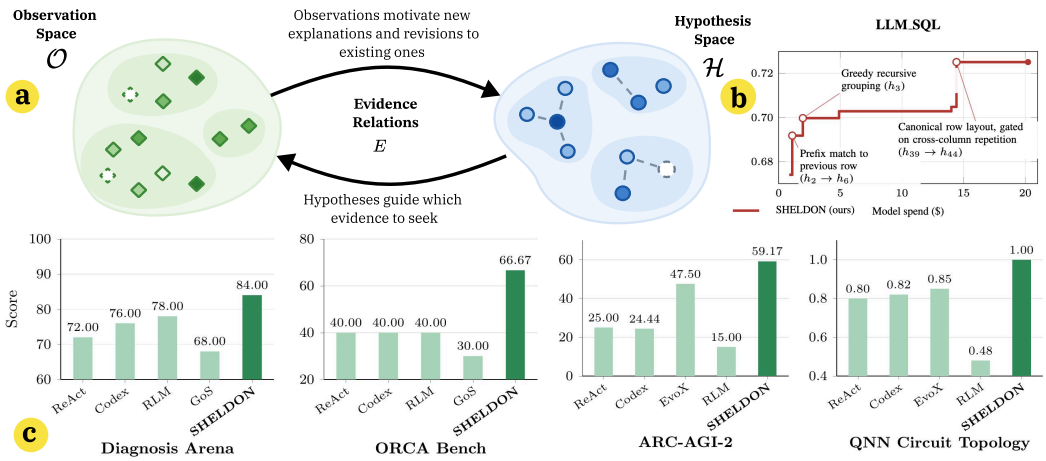


Figure 1: SHELDON uses scientific reasoning process to jointly modeling hypothesis and observation spaces, and outperforms strong baselines across tasks with and without explicit scoring functions. **(a)** conceptual illustration of dual-space reasoning. Edges in hypothesis space represent refinements. **(b)** shows improvements on system optimization task with new hypotheses. **(c)** Performance over diverse investigation tasks.

With advances in test-time scaling, LLMs have become increasingly capable of tackling complex tasks traditionally associated with general intelligence (Snell et al., 2025; Venkatraman et al., 2025; Brown et al., 2024), including algorithm discovery (Novikov et al., 2025; Mitchener et al., 2025), challenging mathematical reasoning (OpenAI, 2026b), and system optimization (Cheng et al., 2025; Cemri et al., 2026). For example, by using a swarm of 10000 concurrent agents, OpenAI resolves the Navier–Stokes Millennium Prize problem (OpenAI, 2026a).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

However, existing test-time scaling approaches exhibit different limitations depending on how they organize search. General coding agents such as Codex (OpenAI, 2025) and Claude Code (Anthropic, 2025) append output of tool calls to their context and take next steps. This narrows the exploration as long-running tasks accumulate context. Evolutionary methods such as AlphaEvolve (Novikov et al., 2025) and AdaEvolve (Cemri et al., 2026) broaden exploration by maintaining multiple candidate solutions. However, they rely on a fixed evolutionary strategy whose design choices are specified in advance. EvoX (Liu et al., 2026a) relaxes this by searching over evolutionary strategies themselves rather than committing to a single hand-designed strategy. Yet these methods still fail to utilize information uncovered during task investigation and execution.

Recent specialized harnesses address this limitation by explicitly retaining information through structured hypotheses, representing them as trees (Arbor) (Jin et al., 2026), graphs (GoS) (Luo et al., 2026), or sets of hypotheses maintained alongside their evidence (Scientist One) (Meng et al., 2026). However, these representations can quickly start to exploit existing known hypotheses rather than introduce qualitatively new alternatives. As a result, exploration often centers on refining existing hypotheses using previously collected evidence. Additionally, proposed hypotheses are not always challenged rigorously. They typically evaluate hypotheses in isolation, rather than considering all together. As a result, they may miss opportunities to identify which observations most effectively separate plausible hypotheses or expose shared failure modes.

To address this under-exploration of hypotheses and lenient investigation, we draw on the hypothetico-deductive method (Hempel, 1945; 1966), Simon & Lea (1974)'s work on rule induction, and its extension to scientific discovery through dual-space search (Klahr & Dunbar, 1988). They argue that scientific discovery is an iterative process in which hypotheses guide empirical investigation, while the resulting observations reshape subsequent hypotheses. We adapt these ideas by explicitly modeling both **hypothesis** and **observation** spaces. Maintaining a hypothesis space naturally supports *exploitation* by exposing the model to previously proposed hypotheses and their outcomes. However this can affect exploration. Therefore, by treating observations as first-class objects rather than merely as evidence attached to existing hypotheses, one can seek evidence in previously unexplored regions of the environment, and use these observations to generate qualitatively new hypotheses, leading to better *exploration*. Hence, the two spaces co-evolve balancing exploration and exploitation.

We instantiate this scientific reasoning process in our harness, **SHELDON** (Systematic Hypothesis Exploration and Learning through Directed Observation and Navigation). SHELDON guides language models through an iterative dual-space search over hypotheses and observations and maintaining how observations relate to hypotheses, through evidence relations. Hypotheses space directs what evidence to seek, while observation space investigates competing hypotheses and reveals new directions for exploration. Through its exploration, candidate-level investigation, comparative analysis, and exploitation stages, SHELDON continually updates both spaces and avoids premature collapse in a narrow set of hypotheses. As shown in Figure 1, SHELDON outperforms evaluated baselines across diagnosis, abstract reasoning, and optimization tasks by upto 26.7 points. Notably, it discovers a Quantum Neural Network (Liu et al., 2026b) topology achieving a perfect score of 1.00, surpassing both the strongest agent and human baseline. Ablations further show that our introduction of the observation space is critical. Removing it reduces performance by 14.0 and 13.34 points on DiagnosisArena (Zhu et al., 2026) and ORCA-Bench (Gong et al., 2026), respectively. We find that SHELDON discovers 29.8% more ground-truth causes and investigates 34.4% more of the discovered causes than GoS (Luo et al., 2026).

2 Related Works

Evolutionary Algorithms Recent work combines evolutionary search with LLMs to generate and refine populations of candidate solutions. FunSearch (Romera-Paredes et al., 2024) applies this paradigm to mathematical and algorithmic discovery, while AlphaEvolve (Novikov et al., 2025) scales it to larger programs. Following works explored scientific equation discovery, code evolution, program optimization, and prompt optimization (Algorithmic Super Intelligence, 2025; Shojaee et al., 2025; Hemberg et al., 2024; Agrawal et al., 2026). More recent methods adapt the search process itself. ShinkaEvolve (Lange et al., 2026) improves sampling and model selection, AdaEvolve (Cemri et al., 2026) adjusts exploration and resource allocation, DeltaEvolve (Jiang et al.,

2026) tracks semantic deltas across candidate solutions, and EvoX (Liu et al., 2026a) jointly evolves solutions and search strategies. These methods primarily search around and evolve candidate solutions. SHELDON instead also models observation spaces, allowing observations to motivate new hypotheses and discriminate among competing ones.

Research Agents LLM-based research agents increasingly automate scientific workflows spanning idea generation, literature review, experimentation, and synthesis (Lu et al., 2024; Yamada et al., 2025; Schmidgall et al., 2025; Kon et al., 2025; Swanson et al., 2025). CORAL (Qu et al., 2026) supports long-running exploration with periodic interventions when progress stalls. Other work explicitly structure the research state: GoS (Luo et al., 2026) represents hypotheses and evidence in a causal graph, however, only focus investigation on the highest probability hypothesis. On the other hand, SHELDON performs comparative analysis of hypotheses. ScientistOne (Meng et al., 2026) links scientific claims to supporting sources and execution artifacts, and Arbor (Jin et al., 2026) maintains hypotheses, artifacts, and evidence in a hypothesis tree. These systems largely organize evidence around existing hypotheses or artifacts. SHELDON has an explicit modelling of the observatin space, coupled to observation-driven hypothesis generation and targeted cross-hypothesis investigation. Several evolutionary and research-agent methods, including AdaEvolve, EvoX, Arbor, and CORAL, are designed for settings with an explicit quantitative scoring function. Instead, SHELDON can operate without such a scoring function, using observations as an implicit scoring function as shown with medical diagnosis and system root-cause analysis benchmarks.

3 Problem Formulation

Definition 1 (Investigative Problem) *An **investigative problem** p is defined by a statement x_p, case material $\mathcal{M}_p$, a set of available tools $\mathcal{T}_p$, and a solution quality function Q_p, which may or may not be made available to the agent solving the problem.*

The case material $\mathcal{M}_p$ comprises the problem-specific information provided for the investigation, such as system logs, seed programs, execution traces, experimental measurements. Tools $\mathcal{T}_p$ specifies the capabilities available to the agent, such as code execution, database access, or simulators. Q_p may measure agreement with the ground truth, or performance under an external evaluation metric.

Definition 2 (Investigation Task) *Given an investigative problem p and a budget B, an LLM agent comprising of a harness π using language model m interacts sequentially with case material $\mathcal{M}_p$ and tools $\mathcal{T}_p$. At step t, the agent generates an action a_t based on the interaction history $\tau_{t-1} = (a_1, o_1, \ldots, a_{t-1}, o_{t-1})$, and receives an observation o_t. Each step incurs cost c_t. Let the solution quality function $\hat{Q}$ be Q_p if it is available, and a generated estimate otherwise. After T steps, the agent computes the final solution $\hat{y}_{\pi,m} = \textsc{Answer}(\tau_T)$ where the objective of* ANSWER *is to maximize $\hat{Q}(\hat{y}_{\pi,m})$ subject to $\sum_{t=1}^{T} c_t \leq B$.*

4 SHELDON: A Harness for Scientific Reasoning

SHELDON organizes scientific reasoning as a coupled search over candidate hypotheses and observations acquired from the case material. Hypotheses guide which evidence to seek, while observations motivate new explanations and revisions to existing ones. Evolving maps of both spaces organize the alternatives under consideration and expose further directions for investigation.

4.1 Dual-Space Representation

Hypothesis and observation spaces. The dual-space account of Klahr & Dunbar (1988) describes scientific reasoning as coordinated search over a hypothesis space and an experiment space, with experiments providing evidence that distinguishes among hypotheses. We generalize the second space to an **observation space**, since evidence in our setting can arise from searches, inspections, program executions, explicit experiments, and other interactions with case material and tools.

We define the **hypothesis space** $H \in \mathcal{H}$ as the space of candidate explanations, solutions, or contributing ideas for a task. Depending on the task, a hypothesis may represent a proposed root cause,

Algorithm 1: SHELDON Dual-Space Reasoning Process

Require: Investigative problem p, interaction budget B
Ensure: Final solution $\hat{y}$

Let the exploration space at time t be $S_t = \langle H_t, O_t, wH_t, wO_t, E_t \rangle$
$\hat{Q} \leftarrow Q_p$ if one is available, and an estimated function otherwise.
$t \leftarrow 0$, $S_0 \leftarrow \textsc{Explore}(p, \emptyset)$
while budget remains and investigation has not stabilized **do**
  for all $h \in wH_t$ **do**
    $wO_t, E_t \leftarrow \textsc{UpdateO}(wO_t, \textsc{CandidateAnalysis}(p, \hat{Q}, S_t))$
  end for
  $wO_t, E_t \leftarrow \textsc{UpdateO}(wO_t, \textsc{ComparativeAnalysis}(p, \hat{Q}, S_t))$
  $(wH_t, wO_t) \leftarrow \textsc{Exploit}(p, \hat{Q}, S_t)$
  $S_{t+1} \leftarrow \textsc{Explore}(p, S_t)$
  $t \leftarrow t + 1$
end while
$\hat{y} \leftarrow \textsc{Answer}(p, \hat{Q}, S_t)$
return $\hat{y}$

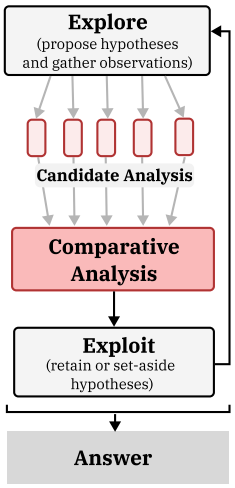


Figure 2: **Overview of SHELDON's reasoning process.** SHELDON alternates between exploration, parallel candidate analysis, comparative analysis across hypotheses.

a diagnosis, a transformation rule, or a candidate design. To facilitate exploration of the hypotheses, we explicitly characterize the space with *axes* and *regions*, inspired by the literature on hypothesis scopes (Thompson & Skau, 2023) and conceptual spaces Gärdenfors (2000). An axis identifies a dimension along which candidates may differ; its regions describe alternatives or portions of that dimension. Examples of axes include method families, underlying assumptions, or ways of determining free parameters. They provide an organization of the search space and a means of identifying qualitatively different investigative directions. At each time step, we track a set of *working hypothesis set* $wH_t = \{h_1, \ldots, h_n\} \subset H_t$, where H_t is the hypothesis space at time t.

Similarly, we define the **observation space** $O \in \mathcal{O}$ as the space of possible observations containing the evidence and evidence-grounded insights obtainable through interaction with the task environment. The observation space is also characterized by axes and regions, where possible axes include data sources, time ranges, entities or subsets, behavioral properties, scales, or tools. As observations also evolve iteratively; at each time step, we track a working set of observations $wO_t = \{o_1, \ldots, o_n\} \subset O_t$, where O_t is the observation space at time t. It contains the cumulative observations currently used to guide investigation. These may describe individual findings or higher-level patterns composed from several findings.

4.2 Dual-Space Reasoning Process

Algorithm 1 presents the high-level overview of the algorithm. Let $\hat{Q}$ be the solution quality solution Q_p if provided and an estimate otherwise. The *state* of exploration in step t, S_t captures the cumulative result of the actions and observations up to time t. $S_t = \langle H_t, O_t, wH_t, wO_t, E_t \rangle$, the current hypothesis and observation spaces (H_t, O_t), the working hypothesis and observation sets (wH_t, wO_t), and evidence relations

$$E_t : H \times O \to \{\textsc{Support}, \textsc{Refute}, \textsc{Neutral}, \textsc{Unresolved}\}$$

indicating the evidence status of a observation with respect to a hypothesis. Similar to Natural Language Inference (NLI) (Condoravdi et al., 2003; Bowman et al., 2015), these categories describe whether an observation provides evidence for a hypothesis, provides evidence against it, is neutral with respect to it, or has a relationship that has not yet been assessed. Here, the relation captures a task-dependent evidential judgment rather than strict textual entailment or contradiction. For example, an observation may support a diagnostic explanation, expose a failure mode of a candidate design, or reveal the conditions under which an implementation succeeds. This provides the agent with a quantitative way of analyzing the working hypotheses given the working observation set.

To manage the working sets of hypotheses and observations, we define the status of h_i, $\textsc{status}(h_i)$ as one of RETAINED or SETASIDE, indicating whether it is actively considered or temporarily set aside base on the strengths of its evidence. A set-aside hypothesis cannot be reconsidered until a cooldown period of a fixed number rounds of investigation has elapsed.

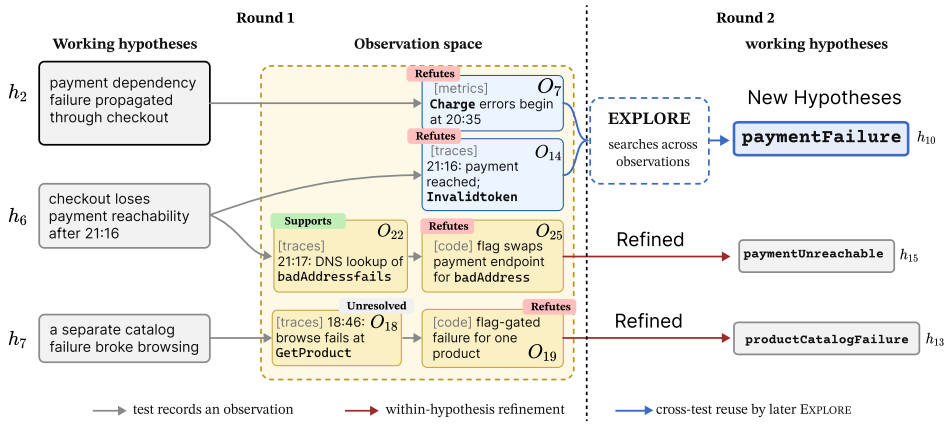


Figure 3: This ORCA-Bench incident contains concurrent catalog and payment failures. In round 1, investigating h_2 hypothesis leads to observation (o_7) **`Charge`** errors begin at 20:35. Whereas, h_6 leads to observation (o_{14}) **`Invalid token`** . In round 2, EXPLORE combines these observations to form a new hypothesis (h_{10}) that talks about `paymentFailure`, showing how observation space leads to newer hypotheses. On the other hand, hypothesis (h_{15}) `paymentUnreachable` and (h_{13}) `productCatalogFailure` are obtained by refining hypotheses from round 1.

We also introduce the boolean attribute PROVISIONAL to each observation because LLM's may occasionally make mistakes in their judgment (Grand et al., 2026) such as making wrong assumptions, buggy implementation, or being ambiguous. When a new observation is encountered, it is given a PROVISIONAL status until it is encountered again. A PROVISIONAL observation can be used to guide exploration and follow-up hypothesis testing, but it cannot be used to refute a hypothesis.

We define ANSWER$(p, \hat{Q}, S_t)$ as an agent-created function that computes a solution based on a set of hypotheses wH with respect to problem p that maximizes the solution quality function $\hat{Q}$; UPDATEO(wO, wO') is a function that adds wO' to wO.

The investigation is an iterative process consisting of these steps, as shown in Figure 2:

1. EXPLORE performs actions to examine the case material, tools, and the existing exploration state. It defines new hypothesis and observation spaces, and proposes a new set of working hypotheses and observations.
2. CANDIDATEANALYSIS gathers observations of individual hypotheses.
3. COMPARATIVEANALYSIS gathers observations of the full working hypothesis as a whole, seeking evidences that help compare the hypotheses.
4. EXPLOIT analyzes the gathered observations to refine the hypothesis, possibly dropping and adding hypothesis, and deciding on their (RETAINED, SETASIDE) statuses.

The process continues until the interaction budget B is exhausted or the investigation stabilizes, meaning that a round produces no new hypotheses, observations, or investigative directions. At termination, SHELDON performs a final evidence assessment over the current working state and returns the candidate solution best supported by the accumulated evidence and available evaluations.

Explore. The EXPLORE method performs actions to inspect the case material using available tools as well as the current exploration state to produce a collection of findings. From the findings, it constructs new hypothesis and observation spaces by defining (1) axes to help diversify hypothesis generation and (2) regions to track what has been exploited and what remains to be explored. It then creates a set of working hypotheses and observations. In the case where there are SETASIDE hypotheses, EXPLORE may also revisit hypotheses from the *set-aside* pool once their cooldown period has elapsed. Although these hypotheses were previously deprioritized, their underlying ideas may still contribute useful components, assumptions, or directions for constructing new candidates. Reconsidering them therefore helps diversify the working set and reduces the risk of concentrating too narrowly on the currently dominant regions of the hypothesis space.

Analyze Hypotheses. SHELDON tests the hypotheses by first looking for evidence to support or refute individual candidates, then conducting a further analysis of the entire set of hypotheses.

CANDIDATEANALYSIS examines each working hypothesis individually in depth. For each hypothesis, the agent investigates its assumptions and components, seeks supporting and refuting evidence, examines conditions under which it succeeds or fails, and follows up on findings that warrant further examination. Where a quality function Q_p is available, it uses an LLM to implement and execute the function. This process is repeated N times. After each round, the investigation inspects the resulting artifact, scores, or failures on particular cases. It updates the evidence relations between the hypothesis and the observations and returns all the new observations. CANDIDATEANALYSIS thus develops an evidence-based account of a hypothesis's strengths, weaknesses, and failure modes.

COMPARATIVEANALYSIS considers the current working hypotheses jointly and seeks observations that distinguish among them. The agent identifies differences in candidates, assumptions, or predicted behaviors and chooses tests whose outcomes can discriminate among them. Such tests may favor some hypotheses while weakening others, expose failure modes shared across candidates, or reveal distinctions that were previously absent from the space maps. When scores are available, differences in candidate performance can motivate investigation into the source. It updates the evidence relations between the hypothesis and the observations and returns all the new observations. Together, CANDIDATEANALYSIS and COMPARATIVEANALYSIS enrich the working observation set and establish the plausibility of individual hypotheses and discriminate between them.

Exploit the Hypotheses. The EXPLOIT function then analyze the hypotheses based on the obserations. It can *refine* the hypothesis by improving it while preserving its core idea or *repair* a hypothesis if its implementated artifact is defective. In the end, it classifies hypotheses as either RETAINED or SETASIDE to guide future exploration.

After revising the current hypotheses, the agent re-enters the exploration stage using the updated working hypothesis set, the working observation set, and the current hypothesis- and observation-space maps as well as the evidence relations, as shown in Figure 3. Exploration is conditioned on the structure already uncovered in both spaces, allowing the agent to balance exploitation of promising regions with exploration of underrepresented ones. This enables the next generation of hypotheses to build on accumulated evidence without collapsing prematurely onto a narrow set of ideas.

5 EXPERIMENTS

Benchmarks. We evaluate SHELDON across 10 tasks with and without explicit quality scoring functions. Tasks without a scoring function are particularly challenging because the agent must infer whether a candidate solution is plausible from the available evidence. For this setting, we use DiagnosisArena (Zhu et al., 2026), a differential-diagnosis benchmark, ORCA-Bench (Gong et al., 2026), a system-level root-cause analysis benchmark, and ARC-AGI-2 (Chollet et al., 2025), where the agent must infer the transformation rule from input-output pairs. .We also evaluate SHELDON on tasks with an explicit scoring function, including molecular optimization (García-Ortegón et al., 2022), quantum neural network topology optimization, and five additional ADRS system-optimization tasks (Liu et al., 2026b). For all benchmarks, we use the official evaluation metrics and scoring scripts. We provide more details on benchmark tasks and evaluation metrics in Appendix A.

Baselines. We evaluate several baselines and use GPT-5.6-terra with high reasoning effort for all. *Iterative coding agent:* Codex (OpenAI, 2025) using `codex-cli` v0.128.0. ReAct (Yao et al., 2022) an agent that interleaves reasoning with tool-based actions. *Evolutionary:* EvoX (Liu et al., 2026a) requires an initial solution set and a quality scoring function. *Research agents:* ScientistOne (Meng et al., 2026) and GoS (Luo et al., 2026). *Autonomous agents:* Arbor (Jin et al., 2026) and CORAL (Qu et al., 2026), both require an initial artifact and a quality scoring function. RLM (Zhang et al., 2025), a recursive reasoning agent. ScientistOne framework is not publicly available, we evaluate its released ADRS programs. Since system optimization tasks can vary from hardware to hardware, we evaluate all methods on the same hardware. SHELDON uses GPT-5.6-terra for all reasoning stages and delegates code implementation to GPT-5.6-luna.

5.1 PERFORMANCE ON DIVERSE BENCHMARKS

SHELDON performs across diverse reasoning and discovery tasks. It achieves the highest reported score on all four tasks, outperforming baselines across reasoning agents, coding agents, recursive agents, and meta-evolution, as shown in Figure 1. The strongest baseline varies across domains,

RLM leads on Diagnosis Arena, while EvoX leads on ARC-AGI-2 and QNN Circuit Topology, but SHELDON maintains its advantage throughout, showing its generalization capability. **SHELDON achieves substantial gains on tasks that require distinguishing competing explanations from evidence.** DiagnosisArena evaluates clinical diagnostic reasoning, while ORCA Bench requires identifying incident root causes from system logs, metrics, traces, and source code. SHELDON achieves 84.00 on DiagnosisArena, exceeding RLM's 78.00 by 6 percentage points and Codex's 76.00 by 8 points. On ORCA Bench, it reaches 66.67, improving over the strongest reported baseline score of 40.00 by 26.67 points. These gains are consistent with SHELDON's being rigorous by explicitly modeling observation spaces and comparative analysis of hypotheses.

SHELDON also advances abstract reasoning and circuit discovery through structured exploration of candidate solutions. ARC-AGI-2 tests the discovery and application of transformation rules, including interacting and context-dependent rules. QNN Circuit Topology evaluates circuit structures for a classification task. On ARC-AGI-2 under $2.5 budget per task, SHELDON achieves 59.17, exceeding EvoX's 47.50 by 11.67 percentage points. On QNN Circuit Topology, it reaches perfect score of 1.00. SHELDON discovers a ten-gate, two-qubit classifier that combines parameterized rotations and CNOT gates, followed by a Hadamard gate and measurement on the second qubit, to classify angle-encoded inputs. These results support the effectiveness of maintaining alternative hypotheses and refining them through investigation across both rule induction and executable design, extending SHELDON's advantages beyond explanatory diagnosis.

SHELDON achieves the best performance on four of six molecular targets and exceeds the library-best reference on three. For the molecular optimization benchmark, SHELDON outperforms Codex, ReAct, and EvoX on all six targets and is competitive with CORAL, achieving the best score on ADRB1, CDK2, PPARA, and F2. On PPARG and PGR, CORAL obtains the highest scores of 9.69 and 10.65, respectively, while SHELDON reaches 9.47 and 9.99. Coral extensively uses search to retrieve existing molecules from the database and their properties, and retain the information in context. However, we observe that it exhausts the token budget during longer runs. Notably, SHELDON exceeds the library-best reference scores on CDK2, F2, and PGR, demonstrating that its evidence-guided exploration and candidate revision generalize effectively to molecular optimization.

Table 1: Performance comparison for molecular optimization task. Yellow cells indicate results that exceed the best reference scores.

System	ADRB1	CDK2	PPARA	PPARG	F2	PGR
Codex	5.97	6.92	7.46	6.38	9.31	7.40
ReAct	8.49	8.32	8.98	8.40	7.79	7.67
EvoX	8.63	7.19	8.62	7.41	8.36	9.17
Coral	11.01	9.32	9.64	**9.69**	9.10	**10.65**
SHELDON	**11.26**	**10.26**	**9.85**	9.47	**9.85**	9.99

5.2 System Optimization Tasks

SHELDON delivers its clearest gains where further investigation has room to improve solution quality. Table 2 reports results over three runs per method, with SHELDON achieving the best or tied-best mean and best-run performance across all five tasks. Its strongest advantages emerge on transaction scheduling and `LLM SQL`, where inspection of the discovered programs reveals distinct strategies for improving solution quality. For transaction scheduling, SHELDON evaluates transaction-position pairs during construction, then refines the schedule through deterministic swap and relocation searches. This construction-intensive strategy achieves best mean and "best" score, exceeding all baselines. For `LLM SQL`, SHELDON measures repetition patterns and selects between recursive partitioning and per-row reordering. Its implementation runs in 4.3s compared with EvoX's 7.1s, achieving a better combined cache-reuse and runtime objective. This illustrates that SHELDON's general hypothetico-deductive scientific reasoning is capable of deriving domain-specific strategies to result in improvements in both average and peak performance.

SHELDON exhibits robust performance across benchmarks, including tasks with limited observed headroom. Its best or tied-best means across all five tasks are consistent with its emphasis on maintaining alternatives and repeatedly reassessing them against evidence. Cloudcast illustrates the distinction between finding a strong solution and doing so consistently: Codex achieves a best transfer cost of 618.10 but a mean of 677.10, whereas SHELDON achieves a best of 618.07 and a mean of 618.09. This consistency remains informative where best-run scores show apparent saturation. The best performance for EPLB, PRISM, Cloudcast is mostly the same across all methods, suggesting that they are relatively simple problems that can be handled with weaker techniques.

Table 2: Performance across ADRS system-level optimization task benchmarks. "Mean" reports the average score over 3 runs. "Cost to best" reports the cost in USD required to obtain the best-performing solution; lower is better. The darkest cell marks the best result.

	EPLB			PRISM			Cloudcast			Txn Scheduling			LLM_SQL		
System	**Mean**	**Best**	**Cost to best ↓**	**Mean**	**Best**	**Cost to best ↓**	**Mean ↓**	**Best ↓**	**Cost to best ↓**	**Mean**	**Best**	**Cost to best ↓**	**Mean**	**Best**	**Cost to best ↓**
Codex	**0.1441**	**0.1441**	0.45	25.60	25.65	0.23	677.10	618.10	0.24	3069	3558	0.19	0.6877	0.7109	0.22
ReAct	0.1440	0.1440	0.28	26.24	**26.26**	0.54	621.11	618.10	0.49	3932	4000	0.30	0.7001	0.7069	0.64
EvoX	0.1318	0.1433	9.70	**26.26**	**26.26**	2.00	618.10	618.08	0.15	4009	4184	9.51	0.7149	0.7196	12.58
ScientistOne	0.1319	0.1440	–	**26.26**	**26.26**	–	624.10	618.10	–	2533	3962	–	0.7034	0.7137	–
Arbor	0.1433	0.1435	6.86	26.25	**26.26**	2.31	618.80	618.10	2.52	3668	3759	4.79	0.7130	0.7218	1.23
CORAL	0.1436	0.1436	12.01	**26.26**	**26.26**	0.30	619.00	619.00	0.24	4077	4201	10.89	0.7115	0.7163	1.23
SHELDON	**0.1441**	**0.1441**	0.80	**26.26**	**26.26**	1.28	**618.09**	**618.07**	6.84	**4198**	**4274**	9.84	**0.7204**	**0.7245**	14.42

Table 3: SHELDON ablation. **Bold** denotes the change relative to the full SHELDON configuration.

Variant	**DiagnosisArena ↑**	**ORCA-Bench ↑**	**ARC-AGI 2 ↑**	**QNN Topology ↑**
SHELDON (full)	**84.00**	**66.67**	**59.17**	**1.00**
w/o comparative analysis	78.00 (**-6.00**)	56.67 (**-10.00**)	50.00 (**-9.17**)	0.8833 (**-0.117**)
w/o observation space	70.00 (**-14.00**)	53.33 (**-13.34**)	50.00 (**-9.17**)	0.8667 (**-0.133**)
w/o continued exploration	82.00 (**-2.00**)	53.33 (**-13.34**)	57.70 (**-1.47**)	0.7333 (**-0.267**)

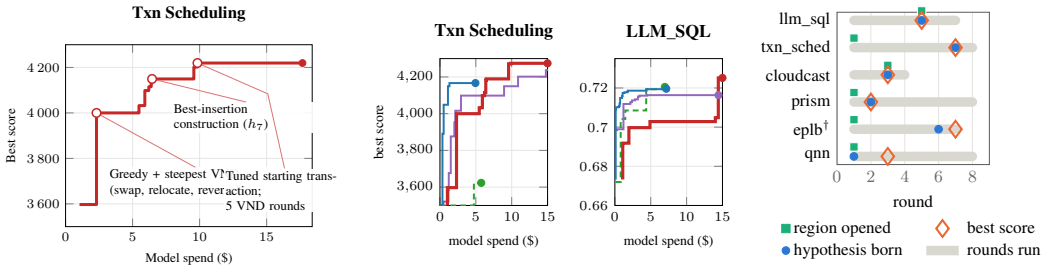
6 ANALYSIS

Each component of SHELDON plays a distinct role in sustaining effective investigation. We ablate comparative analysis, the observation space, and exploration from the second round onward, with the latter allowing the system to continue refining existing hypotheses without introducing new directions (Table 3). All three components contribute to performance, with the observation space and comparative analysis producing the most consistent degradation across all benchmarks when removed. Continued exploration is also important, particularly on ORCA-Bench and QNN topology task, where removing it causes a substantial drop. Comparative analysis further improves results by explicitly contrasting competing hypotheses and actively seeking evidence that can distinguish.

SHELDON exhibits distinct cost-performance trajectories across benchmarks, reaching competitive solutions with little model spend on simpler tasks while continuing to improve when additional exploration remains valuable, as shown in Figure 4b and 6. On EPLB and PRISM, it reaches near-final performance within roughly $1-$1.3, whereas on transaction scheduling it continues making substantial gains through roughly $10, ultimately reaching 4274.41. The contrast is strongest on LLM SQL: performance improves only gradually through about $14 of spend before a late jump from 0.7048 to 0.7245, consistent with the paper's observation that newly explored mechanisms can produce large improvements after an extended plateau. Overall, the trajectories suggest that SHELDON additional spend is not uniformly required, but can support continued exploration that yields higher-quality solutions on tasks with greater remaining headroom.

Coupling exploration with holistic hypothesis investigation. SHELDON pairs hypothesis exploration with holistic candidate and comparative analysis, so that relevant alternatives are not merely proposed but also investigated. On OrcaBench across 30 tasks, SHELDON generates hypotheses that cover 59 of 77 ground-truth causes (76.6%), compared to 36 (46.8%) for GoS. This broader coverage is accompanied by more comprehensive investigations. SHELDON tests 58 of its 59 generated hypothesis that identified the correct cause (98.3%), whereas only 23 of the 36 causes identified by GoS (63.9%) were investigated. The difference therefore extends beyond proposing plausible hypotheses to rigorously investigating them. EXPLORE diversifies the working hypothesis set, while the candidate and comparative analyses examine them against observed evidence.

Exploration vs. Exploitation. The ground truth causes found by SHELDON on ORCA-Bench arise through both exploration and exploitation. Of the 59 ground-truth causes found, 40 are identified through EXPLORE, while the remaining 19 emerge through refining hypotheses already under consideration post testing. Thus, exploration accounts for most discoveries, but exploitation also



(a) Search trajectory evolution (b) Performance against model spend. (c) When does the winning idea appear?

Figure 4: Hypotheses exploration, score vs cost, and the emergence of winning ideas.

contributes substantially: relevant explanations can emerge not only by opening new directions, but also by developing existing hypotheses as additional evidence exposes their limitations.

A similar pattern appears in the optimization tasks in Figure 4c, where the timing of the winning ideas varies substantially across tasks. In several cases, such as PRISM and transaction scheduling, the winning region is opened early, but the winning hypothesis appears only later, **suggesting that continued exploitation within an already-discovered region is important for realizing its potential**. In contrast, for LLM SQL and Cloudcast, the winning region and winning hypothesis emerge at nearly the same time, **indicating that further exploration is what uncovers the decisive direction.** QNN provides another exploitation-heavy case: the relevant region is available from the first round, while the best score is reached only after several rounds of refinement.

How strategies evolve. On transaction scheduling, SHELDON first explored several greedy constructions (h_1–h_4), then improved them with variable-neighbourhood descent (h_5), reaching 4000, as shown in Figure 4a. When this line plateaued, EXPLORE introduced a new NEH-style best-insertion family (h_7), which was combined with the earlier local search in h_8–h_{11} to reach 4274. Follow-up tests showed that reversal was unnecessary (h_{12}) while the starting transaction mattered (h_{13}), narrowing the final strategy to the components that actually improved performance. On `LLM SQL`, early progress came from prefix matching (h_2, refined to h_6) and greedy recursive grouping (h_3), with further refinement to a bounded conditional grouping tree (h_{15}) reaching 0.703, as shown in Figure 1. Although performance then plateaued, EXPLORE continued opening new regions and eventually discovered a multiset-canonical layout strategy (h_{39}). Combining this new mechanism with the previously refined grouping strategy in h_{44} produced the largest improvement to 0.725.

Minimal Reliance on Seed Programs. While seed programs are often provided to jumpstart optimizations, **SHELDON is not heavily dependent on them.** As shown in Figure 5, it recovers 99.9% of the seeded improvement on PRISM, 97.6% on Cloudcast, 96.6% on EPLB, and 95.5% on transaction scheduling, approaching the gains obtained with an initial implementation. Recovery remains substantial on LLM_SQL and QNN Circuit Topology, at 79.3% and 73.7%, respectively. These results demonstrate that SHELDON can generate and refine effective solutions without a supplied starting program, while the larger gaps on LLM_SQL and QNN indicate that informative initialization remains valuable on some tasks.

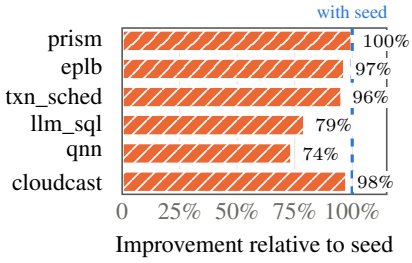


Figure 5: Performance of SHELDON without seed program relative to with seed program.

7 CONCLUSION

We introduced SHELDON, a scientific reasoning harness that jointly searches over hypothesis and observation spaces. By treating observations as first-class objects, SHELDON enables more systematic exploration, comparison, falsification, and refinement of competing explanations. Across diagnosis, abstract reasoning, and optimization tasks, SHELDON consistently outperforms strong agentic baselines, while ablations show that explicitly modeling the observation space is critical to its performance. These results suggest coupling hypothesis and observation search is a promising direction for building more scientific and general-purpose reasoning agents.

AI USE STATEMENT

In this work, we used generative AI tools to help with implementing methods (coding tasks). We have not used generative AI tools to formulate mathematical claims, design or provide feedback on research methodology or experiments, propose or refine hypotheses, interpret results, or support qualitative or thematic data analysis. The remaining required-disclosure tasks, including generating synthetic datasets, developing theoretical models or conceptual frameworks, providing critical ingredients for mathematical proofs, assisting in the writing of proofs, cleaning or reformatting datasets, and translation, are not applicable to this work.

Additionally, we used generative AI tools to create or edit software code and to polish manuscript writing for clarity and readability. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by authors, and all AI-assisted edits to the manuscript were also reviewed by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our evaluation includes medical diagnosis benchmarks, but these experiments are intended only to study the reasoning behavior of the proposed system in controlled benchmark settings. We do not make claims about SHELDON's performance in real-world clinical diagnosis, where cases may be substantially more complex and involve additional sources of uncertainty, incomplete information, workflow constraints, and patient-specific considerations. The system is not evaluated as a clinical decision-making tool, and the results should not be interpreted as evidence of suitability for medical use.

REPRODUCIBILITY STATEMENT

We will publicly release all code, data, and artifacts produced for and accompanying this paper. We provide the prompts used in Appendix C. We provide other benchmark and baseline related information in Section 5. We provide the code as supplementary material.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Algorithmic Super Intelligence, 2025. URL `https://github.com/algorithmicsuperintelligence/openevolve`. Accessed: 2026-09-11.

Anthropic. Anthropic's claude code, May 2025. URL `https://code.claude.com/docs/en/overview`. Accessed: 2026-09-11.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 632–642, 2015.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Mert Cemri, Shubham Agrawal, Akshat Gupta, Shu Liu, Audrey Cheng, Qiuyang Mang, Ashwin Naren, Lutfi Eren Erdogan, Koushik Sen, Matei Zaharia, et al. Adaevolve: Adaptive llm driven zeroth-order optimization. *arXiv preprint arXiv:2602.20133*, 2026.

Audrey Cheng, Shu Liu, Melissa Pan, Zhifei Li, Bowen Wang, Alex Krentsel, Tian Xia, Mert Cemri, Jongseok Park, Shuo Yang, et al. Barbarians at the gate: How ai is upending systems research. *arXiv preprint arXiv:2510.06189*, 2025.

Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. Arc-agi-2: A new challenge for frontier ai reasoning systems. *arXiv preprint arXiv:2505.11831*, 2025.

Cleo Condoravdi, Dick Crouch, Valeria De Paiva, Reinhard Stolle, and Daniel Bobrow. Entailment, intensionality and text understanding. In *Proceedings of the HLT-NAACL 2003 workshop on Text meaning*, pp. 38–45, 2003.

Miguel García-Ortegón, Gregor NC Simm, Austin J Tripp, José Miguel Hernández-Lobato, Andreas Bender, and Sergio Bacallado. Dockstring: easy molecular docking yields better benchmarks for ligand design. *Journal of chemical information and modeling*, 62(15):3486, 2022.

Peter Gärdenfors. *Conceptual spaces*, volume 3. MIT press Cambridge, MA, 2000.

Albert Gong, Kyuseong Choi, Abhineet Agarwal, Jason Schechner, Ryan Huang, Raj Agrawal, Anish Agarwal, and Raaz Dwivedi. Orca-bench: How ready are language model agents for oncall? *arXiv preprint arXiv:2607.28545*, 2026.

Gabriel Grand, Valerio Pepe, Joshua B Tenenbaum, and Jacob Andreas. Shoot first, ask questions later? building rational agents that explore and act like people. In *International Conference on Learning Representations*, volume 2026, pp. 22899–22944, 2026.

Erik Hemberg, Stephen Moskal, and Una-May O'Reilly. Evolving code with a large language model. *Genetic Programming and Evolvable Machines*, 25(2):21, 2024.

Carl G Hempel. Studies in the logic of confirmation (i.). *Mind*, 54(213):1–26, 1945.

Carl G. Hempel. *Philosophy of Natural Science*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

Jiachen Jiang, Tianyu Ding, and Zhihui Zhu. Deltaevolve: Accelerating scientific discovery through momentum-driven evolution. *arXiv preprint arXiv:2602.02919*, 2026.

Jiajie Jin, Yuyang Hu, Kai Qiu, Qi Dai, Chong Luo, Guanting Dong, Xiaoxi Li, Tong Zhao, Xiaolong Ma, Gongrui Zhang, et al. Toward generalist autonomous research via hypothesis-tree refinement. *arXiv preprint arXiv:2606.11926*, 2026.

David Klahr and Kevin Dunbar. Dual space search during scientific reasoning. *Cognitive science*, 12(1):1–48, 1988.

Patrick Tser Jern Kon, Jiachen Liu, Qiuyi Ding, Yiming Qiu, Zhenning Yang, Yibo Huang, Jayanth Srinivasa, Myungjin Lee, Mosharaf Chowdhury, and Ang Chen. Curie: Toward rigorous and automated scientific experimentation with ai agents. *arXiv preprint arXiv:2502.16069*, 2025.

Robert Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. In *International Conference on Learning Representations*, volume 2026, pp. 74026–74078, 2026.

Yoonho Lee, Joseph Boen, and Chelsea Finn. Feedback descent: Open-ended text optimization via pairwise comparison. *arXiv preprint arXiv:2511.07919*, 2025.

Shu Liu, Shubham Agarwal, Monishwaran Maheswaran, Mert Cemri, Zhifei Li, Qiuyang Mang, Ashwin Naren, Ethan Boneh, Audrey Cheng, Melissa Z Pan, et al. Evox: Meta-evolution for automated discovery. *arXiv preprint arXiv:2602.23413*, 2026a.

Shu Liu, Mert Cemri, Shubham Agarwal, Alexander Krentsel, Ashwin Naren, Qiuyang Mang, Zhifei Li, Akshat Gupta, Monishwaran Maheswaran, Audrey Cheng, Melissa Pan, Ethan Boneh, Kannan Ramchandran, Koushik Sen, Matei Zaharia, Alexandros G. Dimakis, and Ion Stoica. Skydiscover: A flexible, adaptive framework for ai-driven scientific and algorithmic discovery. In *Proceedings of the ACM Conference on AI and Agentic Systems*, CAIS '26, pp. 1223–1227. Association for Computing Machinery, 2026b. doi: 10.1145/3786335.3813221. URL https://doi.org/10.1145/3786335.3813221.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Yu Luo, Rongchen Gao, Lu Teng, Xidao Wen, Jiamin Jiang, Qingliang Zhang, Yongqian Sun, Shenglin Zhang, Jiasong Feng, Tong Liu, et al. Graph of states: Solving abductive tasks with large language models. *arXiv preprint arXiv:2603.21250*, 2026.

Rui Meng, Bhavana Dalvi Mishra, Jiefeng Chen, Chun-Liang Li, Palash Goyal, Mihir Parmar, Yiwen Song, Yale Song, Rajarishi Sinha, Parthasarathy Ranganathan, et al. Scientistone: Towards human-level autonomous research via chain-of-evidence. *arXiv preprint arXiv:2605.26340*, 2026.

Ludovico Mitchener, Angela Yiu, Benjamin Chang, Mathieu Bourdenx, Tyler Nadolski, Arvis Sulovari, Eric C Landsness, Daniel L Barabasi, Siddharth Narayanan, Nicky Evans, et al. Kosmos: An ai scientist for autonomous discovery. *arXiv preprint arXiv:2511.02824*, 2025.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

OpenAI. Introducing codex, May 2025. URL https://openai.com/index/introducing-codex/. Accessed: 2026-09-11.

OpenAI. On the navier–stokes millennium prize problem, September 2026a. URL https://openai.com/index/navier-stokes-solution/. Accessed: 2026-09-11.

OpenAI. An openai model has disproved a central conjecture in discrete geometry, May 2026b. URL https://openai.com/index/model-disproves-discrete-geometry-conjecture/. Accessed: 2026-09-11.

Ao Qu, Han Zheng, Zijian Zhou, Yihao Yan, Yihong Tang, Shao Yong Ong, Fenglu Hong, Kaichen Zhou, Chonghe Jiang, Minwei Kong, et al. Coral: Towards autonomous multi-agent evolution for open-ended discovery. *arXiv preprint arXiv:2604.01658*, 2026.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using llm agents as research assistants. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, 2025.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan Reddy. Llm-sr: Scientific equation discovery via programming with large language models. In *International Conference on Learning Representations*, volume 2025, pp. 16054–16085, 2025.

Herbert A Simon and Glenn Lea. Problem solving and rule induction: A unified view. *Knowledge and cognition*, pp. 105–127, 1974.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab of ai agents designs new sars-cov-2 nanobodies. *Nature*, 646(8085):716–723, 2025.

William Hedley Thompson and Simon Skau. On the scope of scientific hypotheses. *Royal Society Open Science*, 10(8):230607, 2023.

Siddarth Venkatraman, Vineet Jain, Sarthak Mittal, Vedant Shah, Johan Obando-Ceron, Yoshua Bengio, Brian R Bartoldson, Bhavya Kailkhura, Guillaume Lajoie, Glen Berseth, et al. Recursive self-aggregation unlocks deep thinking in large language models. *arXiv preprint arXiv:2509.26626*, 2025.

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. *arXiv preprint arXiv:2504.08066*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Lihang Zeng, Shaoting Zhang, and Xiaofan Zhang. Evidx: Evidence-aware active diagnosis with scaffolded llm agents. *arXiv preprint arXiv:2608.24570*, 2026.

Alex L Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025.

Yakun Zhu, Zhongzhen Huang, Linjie Mu, Yutong Huang, Wei Nie, Jiaji Liu, Shaoting Zhang, Pengfei Liu, and Xiaofan Zhang. Diagnosisarena: benchmarking diagnostic reasoning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3074–3098, 2026.

Table 4: Performance on diagnostic and reasoning benchmarks across different agent and optimization paradigms. Cell shading indicates the top three performing methods within each benchmark, with darker shading denoting higher rank.

Method	Method Family	Diagnosis Arena	ORCA Bench	ARC AGI-2	QNN Circuit Topology
ReAct	Reasoning agent	72.00	40.00	25.00	0.80
Codex	Coding agent	76.00	40.00	24.44	0.82
EvoX	Meta-evolution	×	×	47.50	0.85
RLM	Recursive agent	78.00	40.00	15.00	0.48
GoS	Reasoning agent	68.00	30.00	×	×
SHELDON (Ours)	Scientific Reasoning	**84.00**	**66.67**	**59.17**	**1.00**

A Benchmarks and Metrics

For ARC-AGI-2, we use a budget of $2.5 per task. For molecular optimization, we permit up to 1,000 evaluations subject to a $40 budget cap. For DiagnosisArena and ORCA-Bench, the system is allowed to produce its final answer without an explicit evaluation budget. For ADRS and QNN topology optimization, we impose a $15 budget and a maximum of 100 evaluations.

We evaluate all baselines on two Intel Xeon E5-2660 processors at 2.20 GHz, with eight cores and two threads per core per socket, yielding 16 physical cores, 32 logical CPUs, and two NUMA nodes.

We require the ReAct baseline to execute at least 20 iterations per run.

B Additional Results

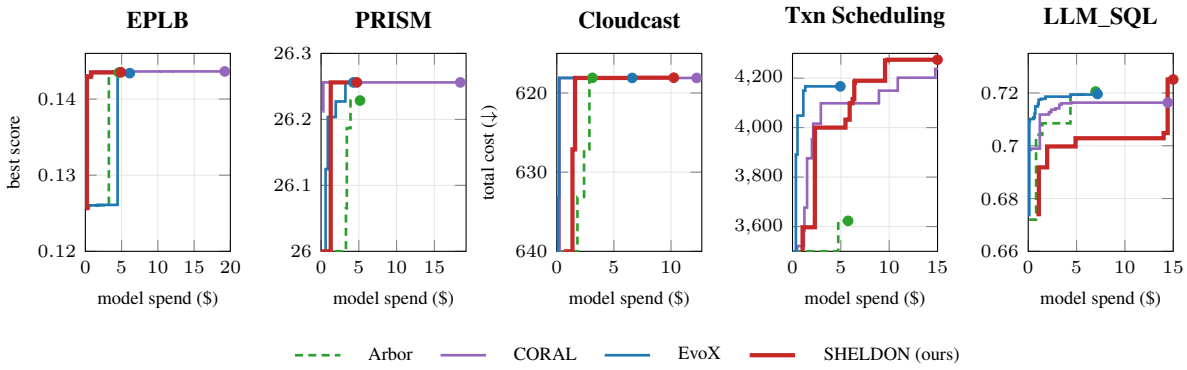


Figure 6: Performance of various methods against model spend on ADRS tasks.

C PROMPTS

This section reproduces every prompt SHELDON sends, in the configuration used in our experiments. Each prompt is a template; values filled in at run time are shown as ⟨*name*⟩, and all other text is verbatim. Every step returns its result by calling a submit tool whose arguments follow a fixed schema (for example, a hypothesis has a `statement`, a `rationale` and a `region`; an observation has a `region`, the retrieval ids it rests on, and a verdict per hypothesis). Tool calls interleave with each step until that tool is called.

C.1 PLAN

One call before the first round. It separates the problem from the specification of the answer, so only the final step is handed formats and output paths.

System

```
You read a task statement and separate it into parts. You are not solving it. Copy text; do
not paraphrase.

## Parts

- `situation`: the problem, the system, and what can be used to investigate it.
- `deliverable`: the required form of the final answer (sections, formats, where to write it)
. Empty if none is given.
- `requirements`: what a finished answer must contain, one short concrete line each, taken
from the deliverable.
- Every part of the statement goes in exactly one of `situation` and `deliverable`.
```

User

```
---
⟨task statement⟩
---
```

C.2 ROUND CONVERSATION (SYSTEM)

Each round is one conversation. This is its system turn; the EXPLORE, selection, COMPARATIVE-ANALYSIS and EXPLOIT prompts below are its user turns, and the model finishes each by calling that stage's submit tool. Tool calls happen inside the turns.

System

```
You are investigating a task in rounds. This conversation is one round: each message gives
you the next stage, and you finish a stage by calling its submit tool.

## How a round goes

- Explore: map where answers and evidence could lie, and propose hypotheses.
- Hypotheses are then built and tested one by one, outside this conversation; their results
come back to you.
- Global test: gather evidence that tells the tested hypotheses apart.
- Revise: decide, for each tested hypothesis, whether to keep, repair, refine or shelve it.
- The state carried over from earlier rounds is in the first message.
```

User

```
Round ⟨round number⟩.
```

C.3 EXPLORE

Opens each round's conversation, so it carries the whole state; its two parts are sent together as the conversation's first user turn. Shown for a later round; in round one the map, hypothesis, evidence and shelf sections are absent and the model opens the map from the task alone.

System

```
You open lines of investigation. You map where answers and evidence could lie, and propose
hypotheses to test. Others build and test them; you do not judge them.

## The map

- Hypothesis regions: kinds of answer, grouped under axes (the dimensions answers differ
along).
- Observation regions: kinds of evidence, grouped under axes (where and how one would look).
- Add a region only if it is distinct from every region already on the map. Add an axis only
if it is new, or subsumes an existing axis (then set `replaces_axis`).
- You can use the tools to look at the environment before you propose; a hypothesis grounded
in what exists beats one about what might.

## Hypotheses

- Propose ⟨n⟩ hypotheses in the least explored regions and axes (counts are shown; UNTRIED
regions and thin axes first). Each needs a testable `statement`, a `rationale`, and a `region
`.
- Make them compete: different regions, different mechanisms, not variants of one idea.
- Read the evidence on last round's hypotheses and every hypothesis's metrics per part of the
 task. Do not re-propose what is held.
- If hypotheses win on different parts, or the best score has stopped improving, at least one
 proposal must be a combination (name the hypotheses and what it takes from each) or a `
revive` of an eligible shelved hypothesis.
- Call submit_exploration once.
```

User

```
**Task**

⟨task⟩

⟨starting artifact, if any⟩

**Hypothesis regions**

⟨hypothesis regions, by axis, with counts⟩

**Observation regions**

⟨observation regions, by axis, with counts⟩

**Current hypotheses**

⟨hypotheses with status, scores and evidence counts⟩

**Evidence on last round's hypotheses**

⟨evidence on last round's hypotheses⟩

**Observations**

⟨observations⟩

**Shelved, that you should consider for reuse especially if the scores plateauing. It could
be useful to take different ideas from different hypothesis and create a hypothesis that
combines them.**

⟨shelved hypotheses past their cooldown⟩

**Shelved, but different from the leader on some part of the task** (not revivable yet; their
 artifacts can be ingredients of a new hypothesis that combines them)

⟨shelved hypotheses that win on some part of the task⟩
```

C.4 SELECT

From round two on, chooses which new, refined, repaired or revived hypotheses are tested. Later turns send only what changed since the model last saw it.

User

```
Next stage: select.

## Choose
```

```
- Candidates: ⟨pending hypothesis ids⟩. Choose at most ⟨k⟩ to build and test this round; the rest are
 shelved and can be revived later.
- Kept hypotheses (⟨kept hypothesis ids⟩) carry over untested and take no slot.
- Prefer tests that would settle what the evidence leaves open; spread across regions unless
the evidence clearly favours one.
- Call submit_selection once.

## New since your last message

**Hypotheses**

⟨new or changed hypotheses⟩

**Observations**

⟨new observations⟩
```

C.5 CANDIDATE ANALYSIS (LOCAL TEST)

One per selected hypothesis, run by a separate tool-using agent outside the round conversation; this is one iteration of its build–evaluate–learn loop. Shown with every optional section; the artifact and evaluator lines appear only on benchmarks that execute candidates.

User

```
You test one hypothesis. This is iteration ⟨i⟩ of at most ⟨max iterations⟩; each iteration builds
on the last.

## This iteration

- Build an artifact that expresses the hypothesis, write it to `⟨artifact path⟩`, and run it.
⟨artifact contract⟩
- Measure it with `⟨evaluator tool⟩`, once, on the file you wrote, and wait for the result:
evaluations take turns, so a call can wait many minutes before it starts. Improve on the
previous iteration where you can; a lower score is still a result.
- Look for evidence that supports and that refutes the hypothesis. The observation regions
below are kinds of evidence to look for, not tools.
- Revise the claim only if evidence you gathered contradicts it, or a small change of
approach -- not a code fix -- should help, and the revision predicts a better score (`
predicts_improvement`). A finding that something does not help is an observation refuting the
 claim, not a revision.

## What to submit

- `observations`: what you found, each with its region, the retrieval ids it rests on, and a
verdict on ⟨hypothesis id⟩.
- Separate a flaw in the artifact from a flaw in the claim; a crash is not a refutation.
- Set `done` when another iteration would not add anything. End with submit_local: your final
 answer in its shape.

## Context

**Task**

⟨task⟩

**Hypothesis under test**

⟨hypothesis, with its record⟩

⟨starting artifact⟩ Gains are measured against it.

**Start from this artifact**: `⟨base artifact path⟩` (⟨its origin⟩). Copy it to your artifact path and
edit the copy; where it marks an editable region (for example between EVOLVE-BLOCK markers),
change only what is inside that region.

**Repair requested** (keep the claim, fix how it is realized): ⟨what the revision asked to fix⟩

**Observation regions** (kinds of evidence)

⟨observation regions⟩

**Previous iterations**

- iteration ⟨i-1⟩: score ⟨score⟩; ⟨artifact path⟩; ⟨notes⟩
```

C.6 COMPARATIVE ANALYSIS (GLOBAL TEST)

After the local tests return, looks for evidence that tells the tested hypotheses apart.

User

```
Next stage: global test. ⟨tested hypothesis ids⟩ are tested; the local results that came back are
below.

## Read the local results with care

- Scores and support/refute counts come from experiments that can themselves be flawed: a bug
, a bad measurement, a narrow check.
- Look at the artifacts and the evidence behind each result before trusting it.
- Where two hypotheses predict different things, find out which happened. Compare their
metrics part by part: a claim that loses overall can win somewhere.

## What to submit

- New observations, each with its observation region and the retrieval ids it rests on.
- For each, a verdict (supports, refutes, neutral) on every hypothesis it bears on, with one
line why.
- Call submit_evidence once.

## New since your last message

**Hypotheses**

⟨hypotheses changed by the local tests⟩

**Observations**

⟨observations filed by the local tests⟩
```

C.7 EXPLOIT (REVISE)

One decision per tested hypothesis: keep, repair, refine or shelve (set aside). `repair` is offered only on benchmarks that execute candidates.

User

```
Next stage: revise. Decide every one of: ⟨tested hypothesis ids⟩.

## Decisions

- `keep`: the claim stands; it is carried forward without retesting.
- `repair`: the claim is sound but its artifact expressed it badly; it is rebuilt and
retested.
- `refine`: the claim needs a changed clause, scope or mechanism; give the new `statement`.
- `shelve`: set it aside; it can be revived if the run stops making progress.

## How to decide

- A result that rests on one measurement of one artifact is weak evidence either way.
- Do not shelve a claim that wins on some part of the task, even if it loses overall: say so
in `rationale`.
- Weigh what supports each claim against what refutes it, and cite those observation ids in `
rationale`; your last submission was filed as the observations below.
- Call submit_revision once.

## New since your last message

⟨observations from the global test⟩

⟨changed hypotheses⟩

**Evidence on each hypothesis to decide**

⟨supporting and refuting observations per hypothesis⟩
```

C.8 FINAL ANSWER

After the last round, whatever the budget.

System

```
You write the final answer from an investigation that is now over.

## How

- Build the answer from the best-supported hypotheses and from what the observations ruled
out.
- Cite the evidence the answer rests on where the format asks for it; the retrieval ids and
their contents are below.
- Do not state as established what was only not refuted.
- If the answer is an artifact, give the path of the best one in `file_path`. ⟨artifact contract⟩
- Call submit_final once.

## Format

⟨the benchmark's answer format⟩
```

User

```
**Task**

⟨task statement⟩

**The answer must contain**

- ⟨requirement⟩

**Hypotheses, best first**

⟨hypotheses, ranked, with scores and evidence⟩

**Observations**

⟨observations⟩

**Retrievals**

⟨retrieval index and contents⟩
```

D BENCHMARK SETUP

ARC-AGI 2. We follow the official setup, defined here: `https://arcprize.org/arc-agi/2`.

ADRS and QNN We follow the official setup provided by Liu et al. (2026b) here: `https://github.com/skydiscover-ai/skydiscover/tree/main/benchmarks`

Molecular Optimization We use the dataset provided by García-Ortegón et al. (2022) and using the official Pypi package docstring 0.3.4, which bundles AutoDock Vina 1.1.2 and the prepared receptors. For task formulation we follow Lee et al. (2025). The paper supplies the objective and the six targets (ADRB1, PGR, PPARA, PPARG, CDK2, F2).

DiagnosisArena. We follow the interactive setup of EviDx (Zeng et al., 2026) for Diagnosis-Arena (Zhu et al., 2026): each case opens with the patient's presentation, and the rest of the record is available through queries to the environment. We report EviDx's open-ended diagnosis metric. We evaluate on a random subset of 50 cases.

ORCA-Bench. In ORCA-Bench (Gong et al., 2026), a task's ground truth can include faults that began after the incident was reported. The telemetry the agent investigates may contain them, but an operator responding at reporting time could not have known about them, so requiring them would be unfair and leaves the task underspecified. We therefore keep only the root causes that occurred before the reporting time. For the same reason, the task description states that the answer should include all plausible causes from the start of the day up to the reporting time. We evaluate on a subset of 30 incidents, stratified by difficulty, and report RCA accuracy as computed by ORCA-Bench's verifier: an incident counts as solved when the report names every root cause in its ground truth.

The task description, which is the same for every method, is shown below; values filled in per task are shown as ⟨*name*⟩.

```
# Root Cause Analysis

You are an expert site reliability engineer. The current time is ⟨current time⟩.

## Task Description

The following issue was reported at ⟨report time⟩:

⟨reported issue⟩

NOTE:
* The time that an issue was reported is not necessarily the same as the time the incident
    actually began. Your task is to pinpoint the root cause events and the times of their
    occurrence.
* There may be multiple root causes or no root cause at all.
* The user may have encountered the problem at any point that day before or during the
    reported time.

## Source Code

The source code for the system is available at `/app/opentelemetry-demo`.

## Telemetry Tools

Please use the following tools:
- Grafana HTTP API (available at $GRAFANA_URL with username `admin` and password `admin`; the
    URL is set in the `GRAFANA_URL` environment variable)

## Instructions

First determine whether an incident actually occurred. If no incident occurred, please write
    an empty file to `/app/report.md`. If an incident occurred, please write an incident RCA
    report to `/app/report.md` naming the root cause events.

---

## Context

The following is not a requirement of this task; it is how incident RCA reports are
    conventionally written.

Summary: Focus on what DID happen and the observable impact -- mention request failures if
    they occurred, but do NOT mention the absence of failures or errors as this can be
    misread and cause confusion.

Timeline: A chronological narrative of ONLY what went wrong, reconstructed from the provided
    telemetry. Each entry should be a single succinct line. Cover higher-level events -- when
     a change was introduced, when symptoms started, when a change was reverted, when
    symptoms subsided -- not individual error occurrences. The first entry should be the
    earliest anomaly visible in the provided telemetry; this establishes the incident start
    time.

For each entry:
- State the time (UTC), the service(s) involved, and what happened
- Quantify where possible (error rate, duration, number of affected requests)
- Do NOT include normal/healthy behavior, baseline noise, or things that worked correctly

Example format:
```
09:44 UTC -- analytics-job deployed with no memory limit set
14:17 UTC -- redis-cache-0 OOMKilled; node memory at 97%
14:18 UTC -- inventory-svc cache misses begin; all requests fall through to Postgres
14:18 UTC -- order-svc goroutine pool exhaustion begins; 503 error rate climbs to 14%
14:29 UTC -- redis-cache-0 rescheduled; error rate returns to baseline
```

Causal chain: Each step must be grounded in the provided telemetry. Cite only the telemetry
    types relevant to that step, using the format below for each.

   - Metrics: one sentence describing the baseline value (or typical range), what it changed
    to, and at what UTC timestamp. Where relevant, include rate of change or duration. Follow
     with the exact PromQL query that would surface it.
    Example -- "Error rate on order-svc was 0.1% at 14:17 UTC, jumping to 14% at 14:18 UTC.
    PromQL: rate(http_requests_total{service="order-svc",status=~"5.."}[1m])"

   - Traces: the cascade from the highest-level operation down to the deepest failing span,
    followed by representative trace IDs from the provided data.
```

```
     Example -- "checkout request -> order-svc goroutine exhaustion -> inventory-svc Redis GET
      timeout. Trace IDs: 4bf92f3577b34da6, a3ce929d0e0e4736"

   - Logs: one sentence with service, endpoint, error, and time range, followed by the exact
     Lucene query that would retrieve the relevant log lines.
     Example -- "Between 14:18:02 and 14:29:11 UTC, POST /api/checkout returned 503s. Lucene:
     service:order-svc AND status:503 AND endpoint:\"/api/checkout\""

Never accept "human error" or "operator mistake" as a terminal answer. Ask why the human was
     in a position to make that error. Stop when you reach a cause that is: (a) actionable --
     something a team can change, or (b) a process or system gap that allowed the failure to
     propagate. Only include steps that are directly supported by the provided telemetry. If
     the causal chain cannot be completed with the available evidence, stop and note what
     additional telemetry would be needed to continue. If the incident has multiple
     contributing causes that cannot fit a single chain, say so explicitly rather than forcing
      one.
```

Consistency across methods. These settings are the same for SHELDON and every baseline: all methods receive the same task descriptions and the same information about each task, and are graded against the same ground truth.

Task ids. DiagnosisArena (case numbers): 42, 64, 76, 94, 96, 102, 128, 144, 196, 210, 257, 266, 324, 328, 334, 367, 395, 415, 445, 454, 459, 489, 506, 517, 534, 546, 554, 561, 565, 566, 582, 617, 618, 626, 634, 656, 685, 721, 723, 724, 728, 737, 777, 819, 825, 845, 848, 865, 902, 912.

ORCA-Bench:

- 38b393f4b9fee541
- d6-big-productcatalogfailure-on-05-easy_ttd1440m_day_same-day
- d1-i2-adfailure-on-03-easy_ttd480m_period_same-period
- d5-d4dep-cartfailure-on-03-easy_ttd480m_period_same-period
- d4-d3dep-productcatalogfailure-on-11-easy_ttd15m_at_off+10m
- d2-d1dep-cartfailure-on-05-easy_ttd60m_at_off+10m
- d2-cc1-admanualgc-on-03-easy_ttd480m_range30m_off+60m
- d5-ic2-adfailure-on-04-easy_ttd480m_range30m_off+60m
- d1-i2-adfailure-on-univ00-universal_ttd480m_period_same-period
- d5-ic2-imageslowload-on-univ00-universal_ttd480m_period_previous-period
- d2-d1dep-paymentfailure-50-univ00-universal_ttd480m_period_same-period
- d3-d2dep-productcatalogfailure-on-univ00-universal_ttd480m_at_off+10m
- d2-cc1-adfailure-on-univ00-universal_ttd480m_at_off+10m
- d3-d2dep-paymentunreachable-on-univ00-universal_ttd15m_at_off+10m
- d5-d4dep-paymentunreachable-on-univ00-universal_ttd15m_at_off+10m
- d5-d4dep-paymentfailure-10-univ00-universal_ttd480m_range30m_off+10m
- d1-n1-loadgeneratorfloodhomepage-on-univ00-universal_ttd60m_range30m_off+10m
- d6-big-cartfailure-on-univ00-universal_ttd480m_range30m_off+60m
- d4-d3dep-paymentfailure-50-univ00-universal_ttd60m_range30m_off+10m
- d4-d3dep-paymentfailure-10-01-hard_ttd1440m_day_same-day
- d2-cc1-adfailure-on-01-hard_ttd1440m_day_same-day
- d2-d1dep-cartfailure-on-00-hard_ttd480m_period_same-period
- d2-d1dep-paymentfailure-50-01-hard_ttd1440m_day_same-day
- d6-big-paymentfailure-50-01-hard_ttd480m_period_same-period
- d6-big-productcatalogfailure-on-01-hard_ttd15m_at_off+10m
- d1-c1-admanualgc-on-00-hard_ttd60m_at_off+10m
- 23c5cb27fee69d1e
- d4-d3dep-paymentfailure-50-02-hard_ttd60m_range30m_off+10m
- d3-d2dep-productcatalogfailure-on-00-hard_ttd480m_range30m_off+10m
- d5-d4dep-paymentunreachable-on-00-hard_ttd480m_range30m_off+60m

E STATE REPRESENTATION

SHELDON keeps its state S_t in three record types: regions, hypotheses and observations. Nothing in the state calls a model; stages read it through fixed renderings and change it only through the operations below, so every change is recorded in the trace.

```python
@dataclass
class Region:
    axis: str            # the dimension this region splits along
    name: str            # identity of the region within its space
    description: str

@dataclass
class Hypothesis:
    id: str              # h1, h2, ...
    statement: str       # the testable claim
    rationale: str
    region: str          # a hypothesis region
    status: str          # pending | tested | shelved
    parent: str          # set on a refinement: the hypothesis it refines
    born: int            # round proposed
    touched: int         # last round it was proposed, tested, shelved or revived
    reason: str          # why it was shelved
    repair: str          # what a requested repair should fix
    artifact: str        # best artifact its local test produced
    scores: list[float]  # every evaluator score of its artifacts
    metrics: dict        # per-part metrics of its best artifact
    notes: str           # the local test's own summary

@dataclass
class Observation:
    id: str              # o1, o2, ...
    content: str         # what was found, checkable by another reader
    region: str          # an observation region
    sources: list[str]   # ids of the tool results it rests on
    round: int           # round filed
    origin: str          # the step that filed it
    verdicts: dict       # hypothesis id -> {relation, why}; supports | refutes | neutral
```

Axes and regions. The hypothesis space H and the observation space O are each a map from region name to `Region`; each region belongs to one axis, so a space is a two-level partition: axes (the dimensions along which answers, or ways of looking, differ) and regions within them (kinds of answer, or kinds of evidence). EXPLORE declares both. A region whose name matches an existing one after normalisation is folded into it. A new axis that subsumes an existing one is declared with `replaces_axis`, and the old axis's regions move under it. Every hypothesis names one hypothesis region and every observation one observation region.

Hypotheses. The working hypothesis set wH is the hypotheses whose status is `pending` (awaiting CANDIDATEANALYSIS: new, refined, repaired or revived) or `tested` (tested and RETAINED, carried forward without retesting); a `shelved` hypothesis is SETASIDE. Nothing is deleted. A claim identical to one already held after normalisation is not added again. A refinement is a new hypothesis in the same region with `parent` set, so the trace shows every claim in the form it was tested.

Observations and evidence. Observations are filed by CANDIDATEANALYSIS and COMPARATIVEANALYSIS. An observation whose content repeats an existing one is folded into it: its sources are merged and its verdicts added. The evidence relation E, is conceptually separate, but at a data level is stored on the observations, as `verdicts`: for each hypothesis the observation bears on, a relation (SUPPORT, REFUTE or NEUTRAL) and a one-line reason; a pair with no verdict is unresolved. The support and refute counts shown for a hypothesis are counts over these verdicts.

What the stages see. Each space is rendered axis by axis, with counts per axis and per region: for hypothesis regions, how many hypotheses were proposed and tested there and the best score reached; for observation regions, how many observations were filed. A hypothesis region with nothing proposed in it is marked UNTRIED. These counts are what EXPLORE is asked to use when it proposes in the least explored regions and axes. A hypothesis is rendered with its status, region, parent, scores, evidence counts, per-part metrics and artifact; an observation with its region, sources and verdicts; and EXPLOIT additionally sees, for each hypothesis it decides, the observations that support and refute it.

F COST

Table 5 reports what each method spent, and how long it took, per task on DiagnosisArena and ORCA-Bench. All costs are computed from each run's token counts under one pricing rule, the list price of GPT-5.6 Terra ($2.50 per million input tokens, $0.25 per million cached input tokens, $15.00 per million output tokens)

All methods are run with a budget limit of 10 dollars per task. SHELDON spends more than the single-agent baselines because it is more meticulous. Diagnosing a patient and finding the root cause of an incident are tasks where a confident wrong answer is costly: rather than committing to the first plausible explanation, it keeps several hypotheses open, tests each one against the evidence, and compares them before answering. Each round therefore runs several investigations. The comparison with GoS, which also maintains multiple hypotheses and spends a similar amount, shows that the cost alone does not buy the accuracy: at a comparable budget, SHELDON scores higher on both benchmarks.

Table 5: Cost ($) and wall-clock time (minutes) per task, on the 50 DiagnosisArena cases and the 30 ORCA-Bench incidents.

	DiagnosisArena		ORCA-Bench	
Method	Cost	Time	Cost	Time
ReAct	0.02	0.3	0.54	4.2
Codex	0.06	0.8	0.73	4.5
RLM	0.09	1.7	1.56	2.7
GoS	2.79	29.1	7.54	55.4
SHELDON	2.91	14.0	6.90	22.0

G EXPERIMENT CONFIGURATION

A run is specified by one configuration file. We show the configuration of one ADRS task (Cloudcast) as a representative; runs on other benchmarks differ in the benchmark options, the budget, the round cap, and otherwise use the same method hyperparameters. `max_usd` is the budget per task in US dollars; `null` lifts a cap. Table 6 explains the method hyperparameters.

```
benchmark: skydiscover
split: adrs
method: hypodeducto
model: gpt-5.6-terra
effort: high
max_usd: 15
max_rounds: null
max_calls: null
params:
  mode: session
  plan: true
  rounds: 8
  n_initial: 5
  n_explore: 3
  select_k: 5
  local_max_evals: 3
  local_patience: 2
  local_min_gain_pct: 0.5
  shelf_cooldown: 3
  stagnation_rounds: 2
  round_reserve: 0.5
  explore_probe_calls: 20
  select_probe_calls: 2
  global_test_probe_calls: 10
  revise_probe_calls: 4
  finalize_probe_calls: 20
  local_concurrent: true
  local_probe_calls: 20
  local_engine: toolloop
  local_engine_params: {model: gpt-5.6-luna}
  probes: [run_shell]
  benchmark_opts: {tasks: ADRS.cloudcast, isolate: true, runtime: host, shell_nice: 10,
     state_timeout: true}
```

Table 6: Method hyperparameters.

Parameter	Value	Meaning
`mode`	session	a round's deciding steps run as turns of one conversation (`session`)
`plan`	on	separate the problem from the answer specification before round 1
`rounds`	8	round cap
`n_initial`	5	hypotheses proposed by EXPLORE in round 1
`n_explore`	3	hypotheses proposed by EXPLORE in each later round
`select_k`	5	hypotheses selected for CANDIDATEANALYSIS per round (round 2 on)
`local_max_evals`	3	iterations of one CANDIDATEANALYSIS loop (evaluator calls, where there is an evaluator)
`local_patience`	2	scored iterations without improvement that end a CANDIDATEANALYSIS loop
`local_min_gain_pct`	0.5	improvement, in % of the starting score, that counts as progress
`shelf_cooldown`	3	rounds before a set-aside hypothesis can be revived
`stagnation_rounds`	2	rounds with no new best and no new observation after which any set-aside hypothesis can be revived
`round_reserve`	0.5	no new round starts with less budget left than this fraction of the mean round cost
`explore_probe_calls`	20	tool calls allowed in EXPLORE
`select_probe_calls`	2	tool calls allowed in selection
`global_test_probe_calls`	10	tool calls allowed in COMPARATIVEANALYSIS
`revise_probe_calls`	4	tool calls allowed in EXPLOIT
`finalize_probe_calls`	20	tool calls allowed in the final answer