

Searching hypothesis spaces

Nodes, actions, values and terminal conditions in research search.

Reasoning agents, feedback, and scientific evidence

The first plausible explanation can be wrong

Imagine two explanations fit the available evidence. Committing to the first forces later reasoning to repair an assumption that may never have been justified.

Keeping both possibilities costs resources, but discarding one can prevent a useful line of investigation from ever being tested.

This motivates a question about control: what should remain available, and which partial idea deserves the next unit of computation?

Keeping alternative branches available

Represent a branch by its evidence, candidate explanation and remaining actions. An imagined outcome belongs to a hypothetical branch, not the observation record.

An edit can create a competing explanation. Retaining the parent makes it possible to return when the child fails a check.

Branches with shared prerequisites may form a graph. The representation and evaluator determine whether extending, revisiting or recombining a candidate is meaningful.

Choosing what the search retains

Candidate	Reason to choose it	Cost or limitation
Greedy refinement	One reliable local improvement is cheap	An early commitment can block another route
Tree search	Partial states admit useful evaluation	Branching and repeated evaluation cost compute
Population search	Whole candidates can be tested and recombined	Needs an archive and trustworthy fitness

Choose the retained object from the feedback available: partial reasoning states, executable candidates, or a simpler local edit.

Pedagogical reconstruction

[Yao et al. \(2023\)](#)

[Kocsis and Szepesvári \(2006\)](#)

Proposal, evaluation and execution are distinct stages

A proposal model suggests an expansion. An evaluator estimates its value. An executor produces an observation or a verified computational result.

These roles may use related models; statistical independence does not follow from distinct prompts. External tests and calibration can reduce shared errors.

Log proposal counts, evaluation calls, failed executions, and wall-clock cost so a comparison reflects the entire search policy.

Breadth-first and best-first search make tradeoffs

Breadth-first search explores shallow alternatives evenly; best-first search prioritises a heuristic. Beam search caps the retained frontier, potentially discarding a branch needed later.

Research nodes may have unequal costs and delayed observations, so depth alone is a poor proxy for expenditure.

Use toy tasks to expose the consequence of pruning before trusting a heuristic on a large and expensive scientific campaign.

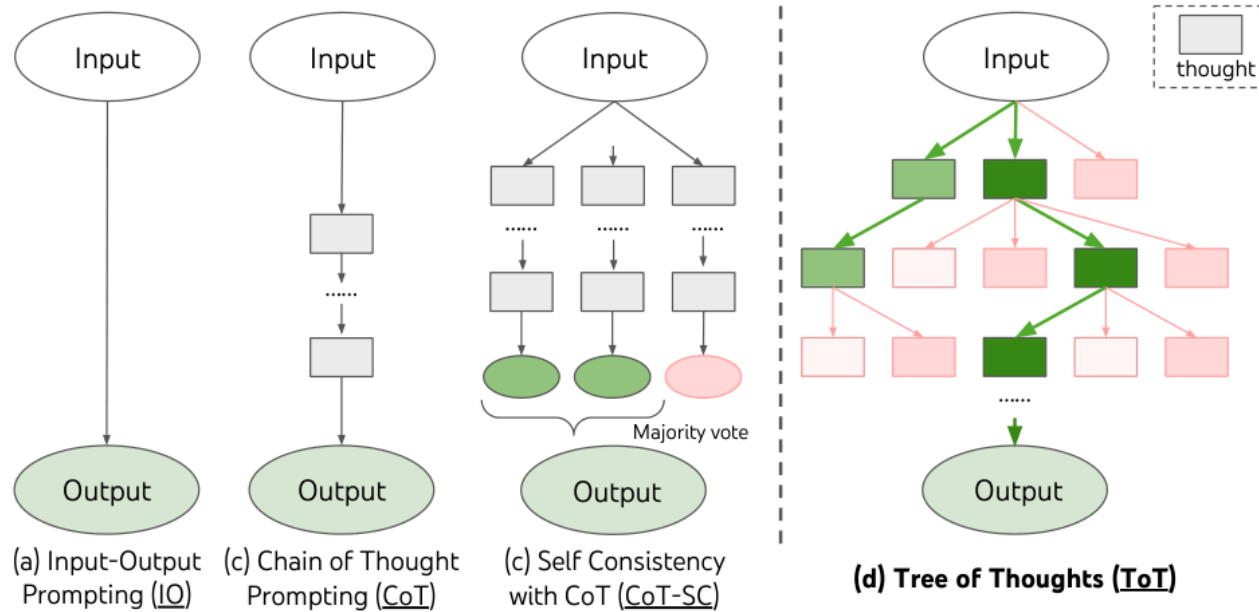
Tree of Thoughts: the method

Tree of Thoughts organises intermediate reasoning into candidate states that can be generated, evaluated, and searched.

Breadth-first or depth-first procedures use model-proposed thoughts and heuristic evaluations to explore alternatives and backtrack. The method operates at inference time rather than defining a new pretrained architecture.

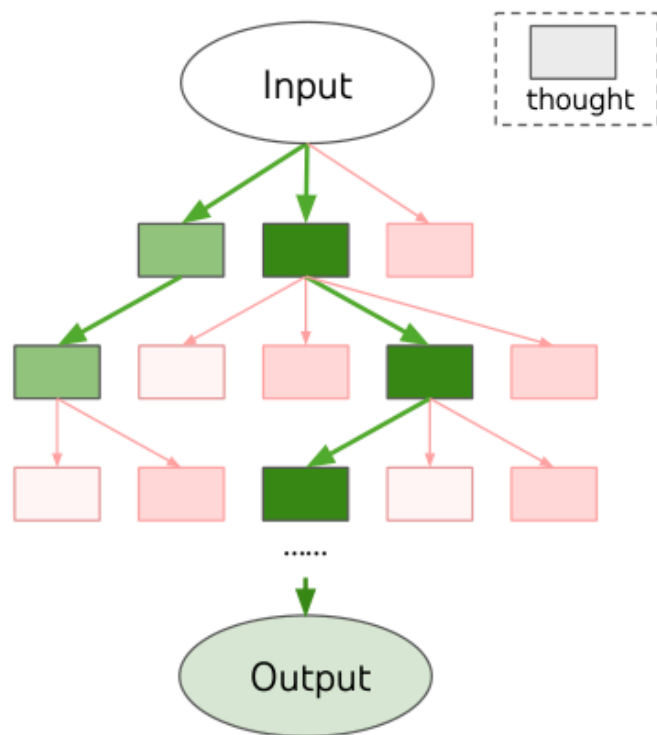
Its usefulness depends on thought granularity, evaluator quality, pruning, and the cost of exploring branches.

Tree of Thoughts: alternative search structures



Input–output prompting, chain-of-thought, self-consistency and tree search retain alternatives in different ways. A linear output can describe backtracking; an explicit tree additionally represents the search policy and retained frontier.

Tree of Thoughts: mechanism detail



(d) Tree of Thoughts (ToT)

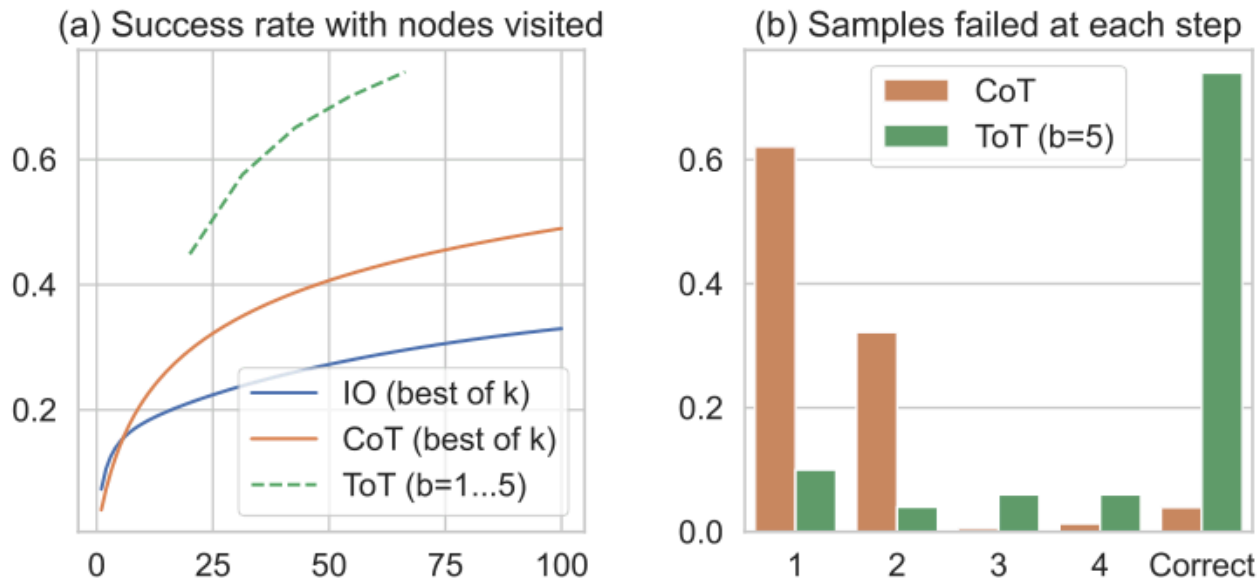
Pruning depends on the estimated value of a branch. Preserving its evidence matters when a later correction changes the grounds for rejection.

Evidence: a success rate needs its search budget

Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	74%
IO + Refine (k=10)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

On Game of 24, the table reports 74% for ToT with breadth five, 4% for CoT prompting, and 49% for oracle best-of-100 CoT. Keep breadth, sample count, and the selection rule attached to each number. The oracle row measures candidate coverage; it is not a practical answer-selection policy.

Tree of Thoughts: interpretation and limits



Prompted value estimates can be wrong, and pruning can discard the eventual solution. A research task with costly or irreversible actions cannot inherit a cheap puzzle-search budget without modification.

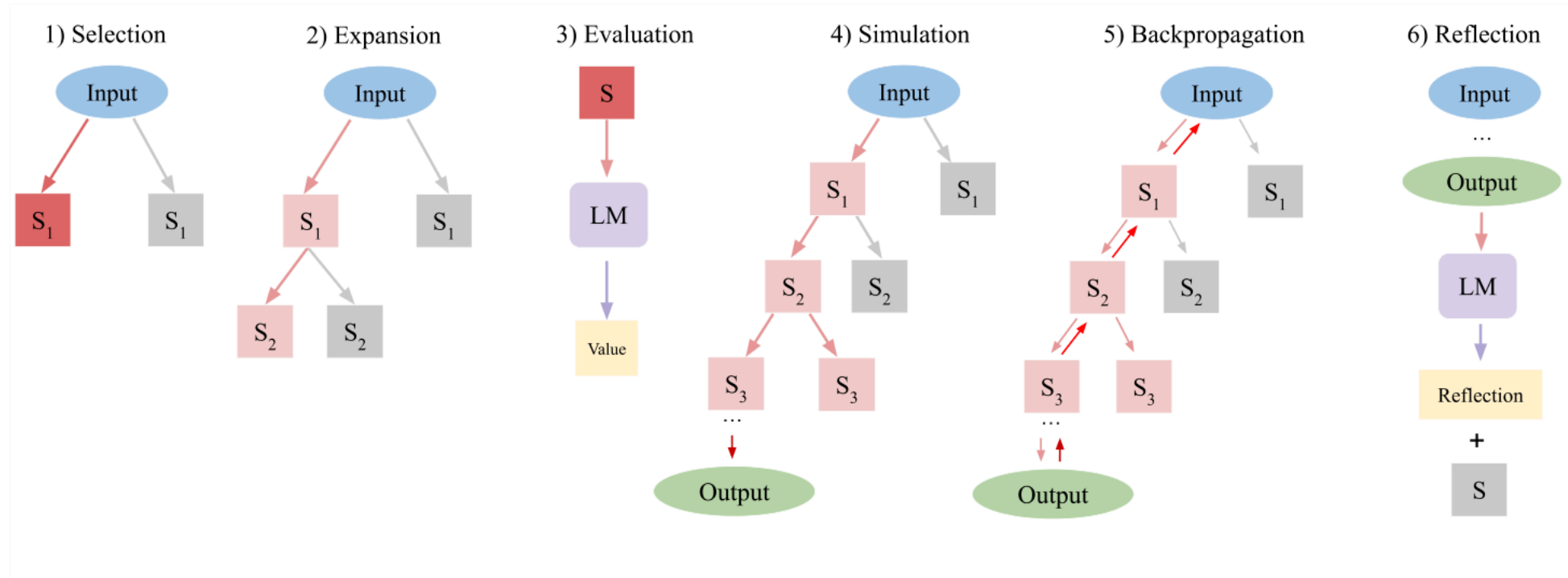
Language Agent Tree Search: the method

LATS combines language-model reasoning and action selection with Monte Carlo tree-search components and reflection.

Selection, expansion, evaluation, simulation, backup, and reflection organise candidate trajectories and their feedback. The environment can provide information that changes which branches are useful.

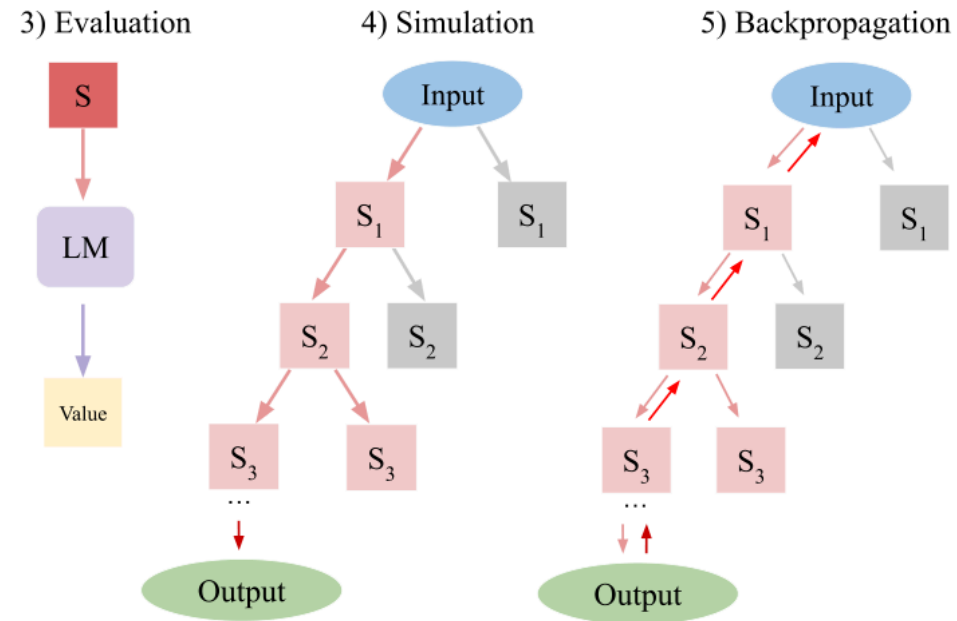
A search-value update is an estimate of future return; it should not be confused with a posterior update over a scientific mechanism.

LATS: search and feedback operations



The six operations separate model proposals from environment or evaluator feedback. Feedback changes search statistics, while reflection changes the context supplied to later proposals.

Language Agent Tree Search: mechanism detail



Evaluation, simulation and backup connect estimated values to observed returns. Their distinction matters for reproducible computation: a prediction about a branch and a validated execution result carry different evidential weight.

Evaluation scope

The study evaluates the framework on reasoning, acting, and programming tasks with task-specific success measures.

A table or architectural example must be interpreted within its model, prompt, and resource conditions.

The results motivate integrating search with environmental feedback. They do not establish that every component is necessary or that the same evaluator transfers to open scientific discovery.

Language Agent Tree Search: interpretation and limits

Prompt Method	Model	Pass@1 ↑
CoT (Wei et al., 2022)	GPT-3.5	46.9
ReAct (Yao et al., 2023b)	GPT-3.5	56.9
Reflexion (Shinn et al., 2023)	GPT-3.5	68.1
ToT (Yao et al., 2023a)	GPT-3.5	54.4
RAP (Hao et al., 2023)	GPT-3.5	63.1
LATS (ReAct)	GPT-3.5	83.8
Base LM	GPT-4	80.1
Reflexion	GPT-4	91.0
LATS (ReAct)	GPT-4	92.7

Environment feedback and prompted evaluation can have different reliability. Comparisons must account for the search budget, tool access, and reflection overhead, especially when actions have unequal cost.

UCB adds an exploration bonus

$$\text{UCB}_i = Q_i + c\sqrt{\log N/n_i}$$

An upper confidence bound (UCB) combines a visited child's mean reward Q_i with an exploration bonus. Upper Confidence bounds applied to Trees (UCT) uses this rule to select branches.

Initialise unvisited children explicitly. The coefficient c determines the reward scale of exploration.

Classical bandit guarantees require assumptions that adaptive language-agent search may violate. Here the formula is a practical selection rule, not an automatic optimality certificate.

A value backup does not itself revise a mechanism

A completed rollout updates estimates along a search path. For an average-return backup, include the new return and increment the visit count.

This changes the estimated usefulness of a branch. Revising a scientific hypothesis requires a separate interpretation of the evidence and its likelihood.

Conflating the two updates can make a search algorithm appear epistemically stronger than its actual observation model supports.

Progressive widening controls costly branching

An action generator can propose far more candidates than the budget allows the system to evaluate. Progressive widening expands the number of admitted actions as visits accumulate.

This keeps early effort focused while permitting new branches later. The widening schedule trades off exploration and evaluation quality.

Under delayed execution, track pending jobs separately so the system does not repeatedly schedule the same unobserved experiment.

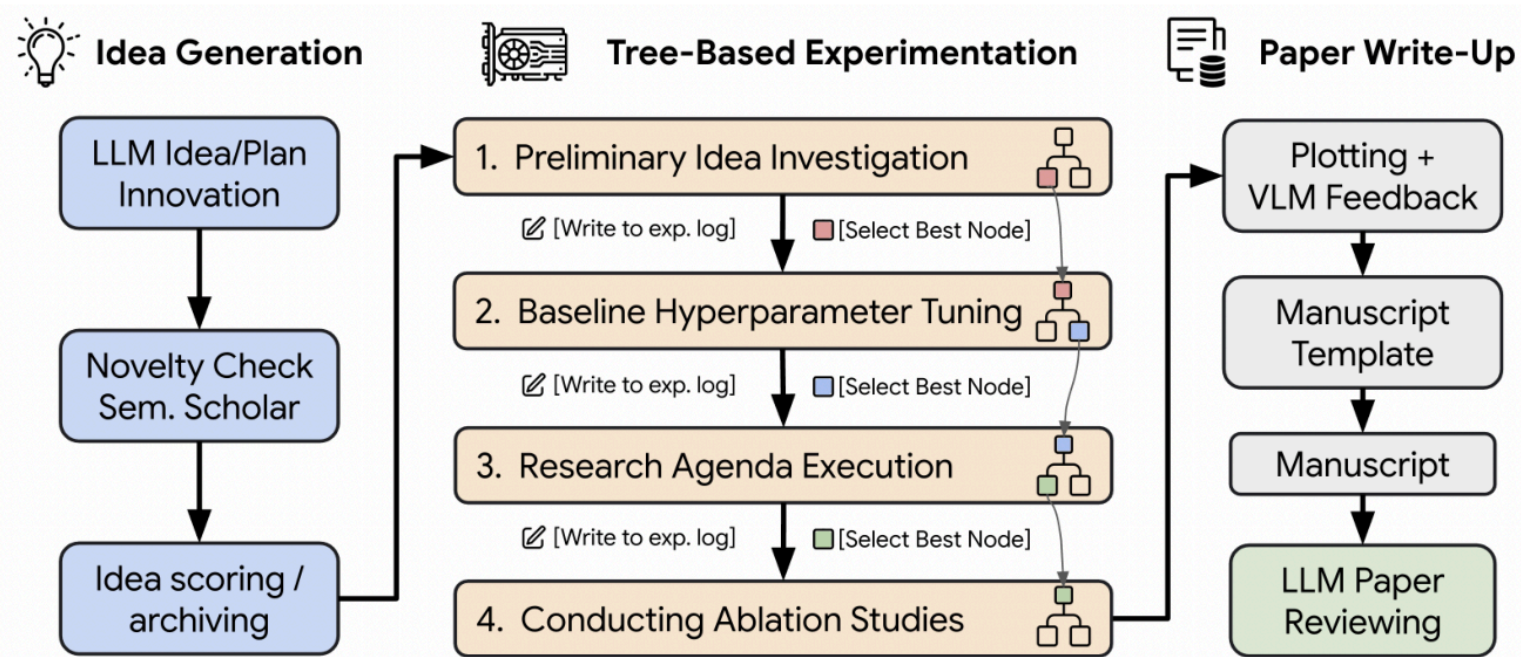
The AI Scientist-v2: the method

The workflow uses agentic tree search to organise experimental research rather than relying exclusively on a fixed experiment template.

Research stages include preliminary investigation, tuning, agenda execution, and ablation, followed by figure creation and writing. Candidate nodes preserve alternative computational paths.

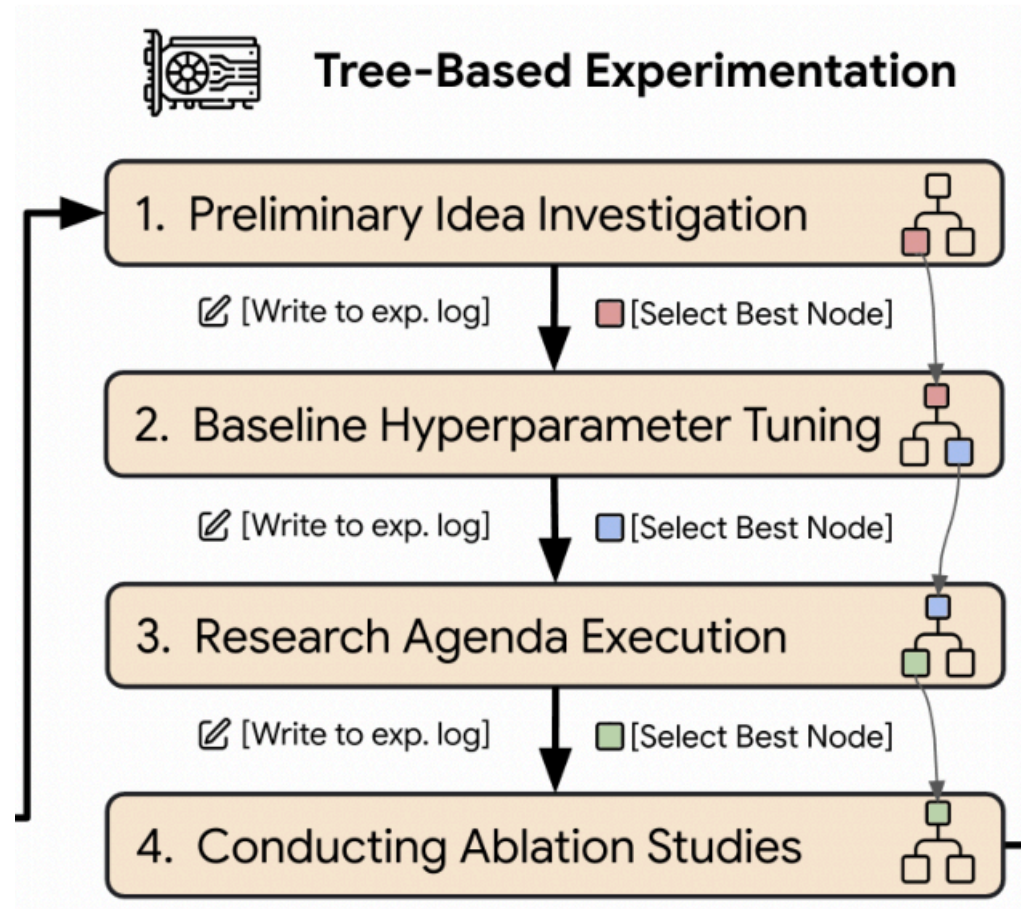
An evaluator helps choose promising branches, but the scientific interpretation of a branch still depends on its implementation and empirical evidence.

The AI Scientist-v2: staged search workflow



The four experimental stages retain selected nodes and pass their records to writing. The tree organizes alternatives and attempts. Scientific likelihood and novelty require evidence beyond the search structure.

The AI Scientist-v2: mechanism detail



The transition from investigation to tuning selects a retained node. Its evidential basis and the treatment of discarded branches determine whether the resulting search trace supports the selection.

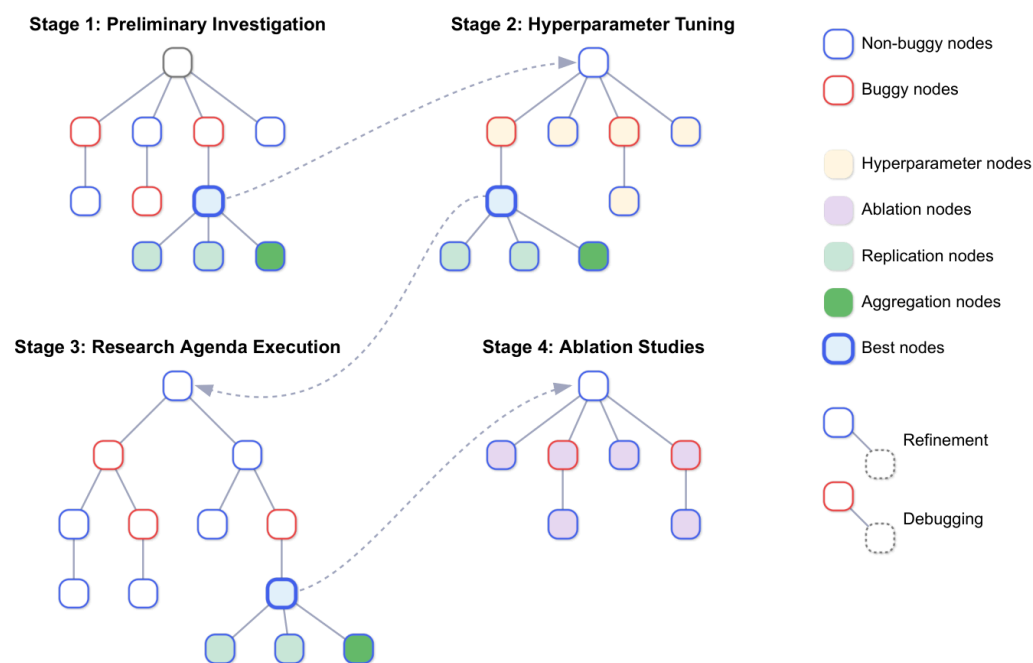
Evaluation scope

The source presents workshop-level generated research and a tree-structured experimental procedure. The extracted search-stage figure shows how node types and selected paths differ.

It is an architectural illustration, not a plot of causal benefit from tree search.

To estimate that benefit, compare a matched sequential workflow using the same tools, evaluation procedure, and total experimental budget.

The AI Scientist-v2: interpretation and limits



Buggy and non-buggy branches lead to a selected checkpoint for the next stage. Debugging success and favorable model evaluation concern different targets. The workflow describes search organization rather than establishing novelty.

AlphaEvolve searches over programs

AlphaEvolve uses language models to propose code changes and evaluators to measure candidate programs. A population of retained programs supplies material for later proposals.

The executable representation makes some properties directly testable. Its scientific reach depends on the problem encoding and evaluator.

Three searches retain different objects

Search design	Retained state	What evaluation controls
Tree of Thoughts	Partial reasoning trajectories	Expansion or pruning of branches
Monte Carlo tree search	Nodes, visit counts and value estimates	Exploration versus exploitation
Evolutionary program search	Executable candidates and an archive	Admission, parent selection and mutation

The unit being searched matters: a promising partial thought, a well-explored node and a useful program are not interchangeable.

An evaluator has several jobs

Separate validity checks from the objective being optimized. Reject programs that violate the declared contract before comparing efficiency or quality.

A staged evaluator can use cheap checks before expensive measurements. Preserve failed attempts and the exact tests because the search policy adapts to the feedback.

A small program-search counterexample

Candidate	Public inputs 0, 1, 2	Held-out input 3
$f(n) = n(n+1)(2n+1)/6$	0, 1, 5	14
$g(n) = f(n) + n(n-1)(n-2)$	0, 1, 5	20

Synthetic task: compute the sum of the first n squares. Both programs pass the three public cases. Only f gives the stated held-out answer.

An archive changes what can be proposed next

Round	Candidate and feedback	Archive decision
1	Faster code fails a required test	Reject: speed cannot compensate for invalidity
2	Valid code improves the target metric	Admit with code, score and evaluation context
3	A different valid strategy covers another case	Retain if the archive policy values that niche
4	A later mutation combines useful components	Re-evaluate the new program from scratch

Illustrative evolution trace, not an AlphaEvolve run. A stepping stone is useful only under an explicitly chosen archive policy.

Illustrative search: two competing branches

Child	Mean reward Q	Completed visits n	Before calculating
A	0.6	10	Larger exploitation value
B	0.4	2	Larger exploration bonus

For a parent with $N = 20$ visits, eight visits went to children outside this two-child comparison. With $c = 0.3$ and natural logarithms, the UCB score combines mean value and exploration bonus. The child with the larger mean need not have the larger score.

Calculate the exploration bonuses

$$\text{UCB}(A) \approx 0.764, \quad \text{UCB}(B) \approx 0.767$$

For A, the bonus is 0.3 times the square root of $\ln(20)/10$, approximately 0.164. For B, it is approximately 0.367.

The UCB scores are therefore about 0.764 and 0.767. B narrowly wins because it has been explored less.

This is a toy calculation. The near tie means small changes in the exploration coefficient or reward estimate can reverse the decision.

Back up a completed rollout

$$Q'(B) = \frac{nQ + r}{n + 1} = \frac{1.7}{3}$$

Suppose a rollout from B returns reward 0.9. Its previous total reward was $2 \times 0.4 = 0.8$. The updated mean is $1.7/3$, approximately 0.567.

The meaning of a terminal reward of 0.9 depends on its metric and validation status. A promising but failed execution provides different evidence from a validated result.

This numerical backup updates search value; it is not a posterior probability that B's mechanism is true.

An ablation that isolates the source of gain

A fixed candidate set separates random selection, prompted evaluation and execution-based evaluation. With the evaluator fixed, a second comparison isolates differences in candidate generation.

Total execution and evaluation budgets define the resource comparison. Success across tasks and seeds, including unfinished jobs, provides a broader account than the number of language-model calls.

This separates gains from search control, candidate coverage, evaluator quality, and additional resources.

Action selection and sample-mean backup

$$Q_{i, \text{new}} = Q_i + \frac{r - Q_i}{n_i + 1}$$

This incremental average follows by dividing the old total return $n_i Q_i$ plus the new return r by $n_i + 1$. It requires storing only the mean and visit count.

The return can encode a task-specific utility, including execution cost, provided the convention is consistent across branches.

This statistical update estimates branch value. A scientific posterior update instead needs a likelihood for the observed evidence.

When an expensive branch looks unexplored

A branch with several pending external jobs can still have a small completed visit count. Naively applying an exploration bonus may launch redundant work.

Track completed evaluations and pending actions separately; specify a scheduling rule for unavailable outcomes.

Immediate and delayed feedback can differ in duplicate work and final utility. Matching total execution cost separates feedback timing from the contribution of a tree-selection rule.

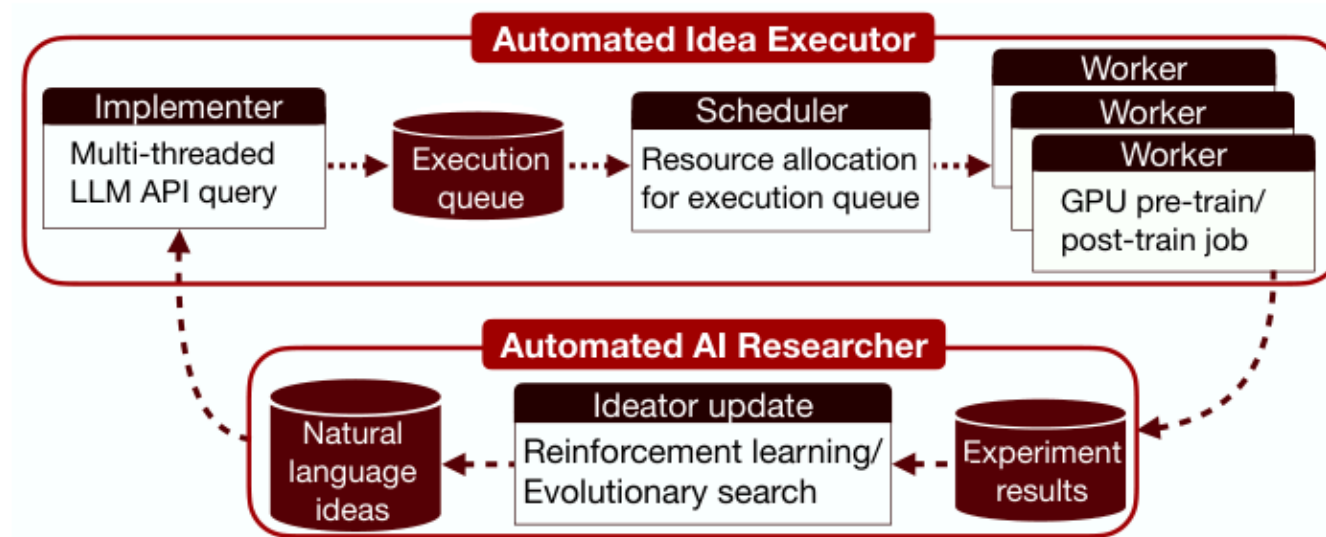
Recent research: Execution-grounded AI research

The study separates an ideator from an automated executor containing implementation, scheduling, and worker components.

Proposed ideas are implemented and tested in defined machine-learning environments. Execution feedback can then guide evolutionary search or reinforcement learning of the ideator.

The literature snapshot is dated 11 September 2026. Findings remain conditional on the source version and evaluation conditions. Recent preprints contribute reported results with distinct validation requirements.

Execution-grounded research: feedback architecture



The architecture links a natural-language proposal to implementation, an execution queue and worker results, then returns feedback to the researcher. It explicitly separates proposing an idea from obtaining an executed result.

Recent methods in context

AlphaProof (Nature 2025/26). Uses reinforcement learning in a formal proof environment, including problem-specific test-time training. Adaptation cost and human formalization affect the result; a verified proposition alone does not establish broader mathematical novelty.

Model Discovery Agent (preprint; revised 2026). Combines Bayesian model/parameter inference, LLM proposals of model structures and value-of-information experiment choice. Version 4 reports fixing prompt leakage and rerunning benchmarks; comparisons must use the corrected version.

AutoDiscovery (NeurIPS 2025). AutoDiscovery combines hypothesis exploration with Bayesian surprise. It links an observed belief change to further search, while surprise and the prospective value of a future experiment remain different quantities.

References (1/2)

[Shunyu Yao, Dian Yu, Jeffrey Zhao et al. \(2023\). Tree of Thoughts: Deliberate Problem Solving with Large Language Models](#)

[Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman et al. \(2023\). Language Agent Tree Search Unifies Reasoning Acting and Planning in Language Models](#)

[Yutaro Yamada, Robert Tjarko Lange, Cong Lu et al. \(2025\). The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search](#)

[Dhruv Agarwal, Bodhisattwa Prasad Majumder, Reece Adamson et al. \(2025\). AutoDiscovery: Open-ended Scientific Discovery via Bayesian Surprise](#)

[Levente Kocsis, Csaba Szepesvári \(2006\). Bandit based Monte-Carlo planning](#)

References (2/2)

[Chenglei Si, Zitong Yang, Yejin Choi et al. \(2026\). Towards Execution-Grounded Automated AI Research](#)

[Alexander Novikov, Ngân Vũ, Marvin Eisenberger et al. \(2025\). AlphaEvolve: A coding agent for scientific and algorithmic discovery](#)

[Thomas Hubert, Rishi Mehta, Laurent Sartran et al. \(2025\). Olympiad-level formal mathematical reasoning with reinforcement learning](#)

[Kevin Murphy \(2026\). Model Discovery Agent: LLM-assisted Bayesian experiment design for data-efficient discovery of mechanistic world models](#)

[Adrien Couëtoux, Jean-Baptiste Hooch, Nataliya Sokolovska et al. \(2011\). Continuous Upper Confidence Trees](#)