Preprint · 15 October 2026 · pensievelabs.co/papers/recursive-priced-falsification

Recursive Priced Falsification: Asymmetric Trials with a Forecast-Scored Planner for Autonomous Discovery

Krish Ghosh

Pensieve Labs (krish@pensievelabs.co)

Code: github.com/pensievelab/rpf

Abstract

Autonomous discovery loops, in which language-model agents improve a design against an automatic evaluator, spend most of their effort on ideas that fail. We argue that such a loop should be built so that a failure is cheap and a success is not capped, and that the part that decides what to try should be scored like everything else. We introduce *recursive priced falsification*, an agentic topology that does both. It is an arrangement of agent roles, what each role may read, the gates between them and a record they share. Instruments measure the incumbent design, scouts write priced cards from that map, a pricer runs the cheapest test that could end a card, a builder builds what survives, and a breaker with a fresh context tries to break each claim. What moves between the roles is a *trial*: a forecast written before a run, the result that would end the idea, and the idea's price in the currency of the objective. Each trial is an asymmetric bet: a stake that is small and known before the run, against a gain that the stake does not cap. Trials are ranked by expected gain per unit of stake, not by their chance of success. The method is recursive: the loop applies it to itself. The loop arrived at its own arrangement of roles this way during the run: it began as a hub, priced candidate roles against its own record, and adopted the arrangement after running it once as a trial, with a forecast written first. The planner that ranks the trials is forecast-scored: it commits to a forecast in every task, every report lists what the task got wrong, a red team attacks its closed directions, and the loop revises its own procedure from its logged mistakes. We give the optimality conditions behind the price map and the ranking rule, an ordering rule for an evaluation cascade, and a rule for which comparison can decide that a change is exact. We implemented the topology in Pensieve, our research harness, and ran it for six days on a live public benchmark for quantum circuit optimisation, with other agent harnesses improving the same circuit under the same evaluator. The best score fell by 5.73% in 42 accepted entries. The loop produced 70.8% of that fall in 17 entries, took the top of the benchmark twelve times, and made the only two steps above 1% by any system (1.45% and 1.04%). The benchmark as a whole moved twelve times as fast during the run as in the five weeks before it (0.99% a day against 0.083%), and fell back to 0.063% a day when the loop stopped. Its trials paid in the way the design predicts. The median trial cost 50 minutes of one worker and kept nothing; the mean trial earned 0.03% of the score, the size of the median accepted entry of the other systems. Ten of 133 trials hold 79% of the loop's points, and the largest of them cost one builder 63 minutes. Trials beat the odds written for them (46% against 30%) and returned 1.6 times the size forecast for them. What paid was a capped loss against an open gain: the largest gains came from trials given about even odds, not from long shots. The loop revised its own procedure during the run, after each of 67 logged mistakes; after the full topology was in place, 1 of 12 builds kept nothing, against 10 of 35 before. We present it as an autonomous research loop and not as recursive self-improvement: the foundation model is frozen, and no revised loop was tested against its predecessor under a matched budget. The comparison with the other systems

is observational and has no ablation. All research was carried out by the agents; a human operator set the goal, approved actions with effects outside the loop and sent occasional directions on strategy.

1 Introduction

Language-model agents now improve designs, and their own scaffolding, with little human input. One agent edits a program and keeps what a fixed metric likes [22]. An evolutionary controller keeps a database of programs, mutates them with language models and scores them with an automatic evaluator [25, 36, 41]. A pipeline of roles writes ideas, runs experiments and reviews its own paper [31, 43, 67]. Multi-agent systems generate, debate and rank hypotheses [14], or run long cycles of analysis and literature search around a shared world model [34]. Search over machine-learning experiments has become a subject of its own [55, 63, 66].

A second line turns the search on the agent itself. Its formal ancestor is the Gödel machine, a self-referential program that rewrites its own code once it has proved that the rewrite raises its utility [44]. Its descendants replace the proof with empirical validation on a benchmark. A seed improver is run on itself [75]; a meta agent writes new agents as code [16]; archives and trees of coding agents that edit their own code are grown under a benchmark score [40, 58, 64, 76]. The newest systems make the meta agent itself editable [78], or let an outer loop rewrite the harness of the research agent that runs in the inner loop [53].

We call the common core of these systems a *discovery loop*: agents propose a change to a design, an evaluator tests it, and the loop keeps what passes. The design may be a program, a circuit, a proof, a model or an agent. What the systems differ in is who proposes, who evaluates and what is kept.

Most trials fail, and loops do not plan for it. In every one of these loops most of what is tried does not work. What a failed attempt costs, and what a successful one can return, are set by how the loop is built. Yet loops choose what to try by score, by novelty or by a learned preference, and most pay the full cost of an attempt before they learn that it failed. They do not ask the two questions an engineer asks first: what does one unit of improvement cost along each route, and what is the cheapest test that could end this idea?

The meta level is not under test. In nearly all of these loops the external feedback reaches only the object level: what is scored is the *candidate*. The meta level, the part that decides what to try, whether a controller, a meta agent or a planning agent, is fixed, or is judged only through the scores of its candidates. Its picture of the problem is never put on the table: when a task is badly posed, the experiment fails and the idea takes the blame. The systems that do edit the meta level keep parent selection and evaluation fixed [78], or accept a rewrite of the harness on a held-out score [53]. In the parts we read, the test in both is of what the meta level produced, not of what it believed.

Recursive priced falsification. We introduce recursive priced falsification (priced falsification, for short), an agentic topology: an arrangement of agent roles, the information each role may read, the gates between them and a record they share (Figure 1). It is aimed at both gaps. An idea gets a price before it gets a build. A task is issued as a trial, with a forecast from the party that set it and a second forecast from the party that will run it. Every report must say what the task

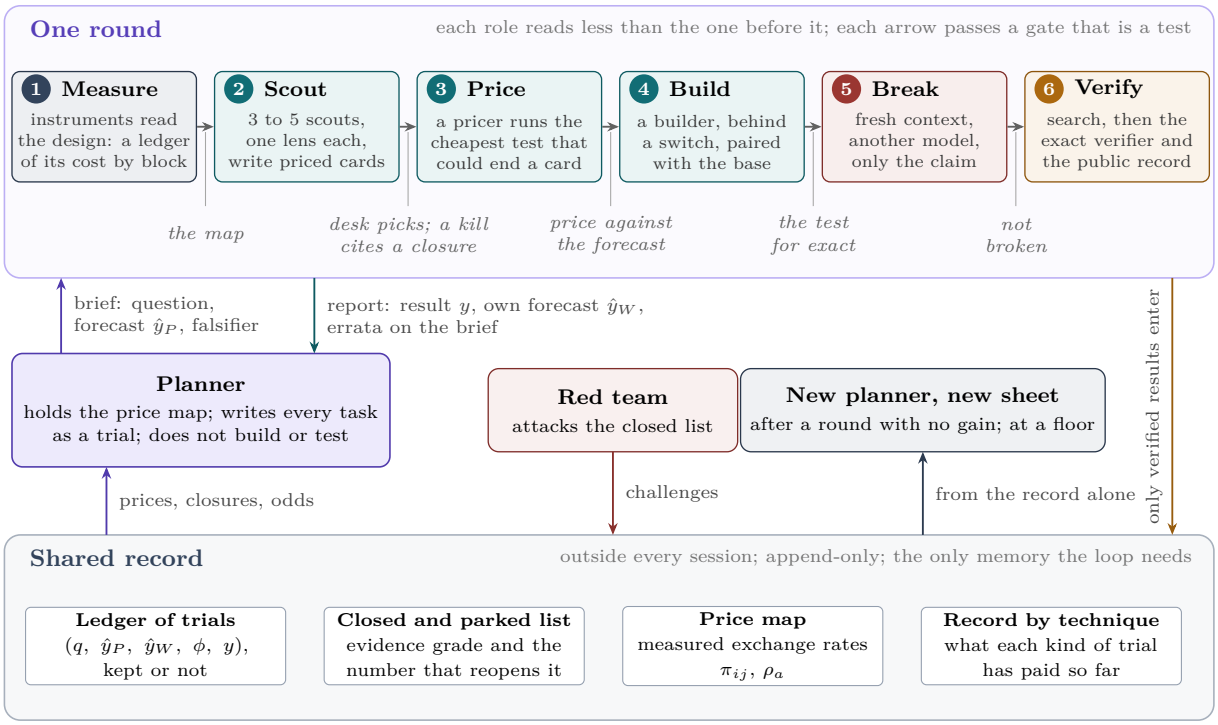


Figure 1: The topology of priced falsification (schematic). One round runs left to right. Instruments measure the accepted design; scouts turn that map into priced cards; a pricer runs the cheapest test that could end a card; a builder builds what survives; a breaker in a fresh context attacks the claim; only then is the exact verifier paid. The planner writes each task as a trial with its own forecast and a falsifier, and each report returns a second forecast and errata on the task. A red team attacks the planner's closed list, and after a round that keeps no gain the planner is rebuilt from the record alone. No agent's conversation holds the state of the research.

got wrong. An idea that is dropped keeps a number that would bring it back. And the party that plans is forecast-scored, corrected from below and attacked by a red team, like any other part of the loop. That is the sense of the word recursive in the name: the method is applied to the loop itself as well as to the design. The loop arrived at its own arrangement of roles in this way during the run, by pricing candidate roles against its record and running the arrangement once as a trial before adopting it (Section 3.2). These are the topology's design principles (Section 2.1). Each is enforced by a role, an edge or a gate, and none by an instruction to a single agent.

Small stakes, open gains. The topology is built so that being wrong is cheap. A trial costs the price of the cheapest test that could end it, and that price is known before the run. What it can gain is the idea's size at the objective's exchange rate, which the stake does not cap. Tests form an evaluation cascade ordered by cost per refutation, so most ideas end on a rung that costs minutes; an idea that is too dear today is parked with the number that would reopen it, and held at no cost. Trials are ranked by expected gain per unit of stake, not by their chance of success (Section 2.9). A loop built this way should lose most of its trials early and owe most of its progress to a few of them.

Why a topology and not a prompt. Each role reads less than the one before it and returns a claim that the next gate can test: a scout sees the whole map of the design's cost, a pricer one card, a builder one priced card, a breaker only the claim and the change. The gates between

roles are tests, not opinions. This is what makes the planner testable. A second forecast is worth something only if it comes from a context the planner did not write, and an attack on a claim is worth something only if the attacker did not inherit the claimant's reasons. One agent cannot give itself either. The literature on self-correction reaches the same point from the other side: a model revises its own reasoning poorly without a signal from outside [17], and the gain from a loop of generating, verifying and updating is bounded by the gap between generating and verifying [52].

Results. We implemented the topology in Pensieve, our research harness, and ran it for six days on a live public benchmark, with other agent harnesses improving the same circuit under the same evaluator. The best score fell by 5.73% in 138.7 hours, and the loop produced 70.8% of that fall in 17 of the 42 accepted entries, including the only two steps above 1%. The benchmark moved twelve times as fast during the run as in the five weeks before, and slowed when the loop stopped (Section 3.4). Eleven other entries used the loop's foundation model inside a general coding agent, the nearest the run has to a baseline without the topology; they hold 21.1% of the fall.

The trials paid in the way the design asks for (Section 3.5). The median trial cost 50 minutes of one worker and kept nothing, as did 85 of the 133, while ten trials hold 79% of the loop's points. Trials beat their odds and returned 1.6 times their forecast size, and the largest gains came from trials given about even odds, not from long shots. The largest step was an idea the loop had parked with a number, which a worker answered in forty minutes against the planner's forecast (Section 3.6). Later rounds show the full topology (Section 3.7): an empty list of cards to an accepted entry in under three hours, and a gate that held back an entry whose rise in failure rate no paired test could see.

What improves, and what we do not claim. The foundation model is frozen. What the loop changes is its scaffold: the price map, the instruments, the record, its own procedure, revised after each of 67 logged mistakes, and its arrangement of roles (Section 3.2). We call the method recursive in that narrow sense: its tests apply to the loop as well as to the design. We do not claim recursive self-improvement, an open-ended process or a controlled ablation, and no revised loop was tested against its predecessor under a matched budget (Section 5). Many parts are already in the literature: a forecast with a falsifier before a run [15, 19, 26, 50, 79], an append-only evidence ledger [15], agents rebuilt from records [9, 56, 80], failure memories with conditions for reopening [59, 86], staged evaluation [36] and agents whose job is a counterexample [18, 23, 68]. What is new is the combination, turned on the planner (Section 4 and Appendix A).

Contributions.

1. **An agentic topology whose gates are tests:** roles with narrowing views around a planner and a shared record, and a rule for which comparison decides that a change is exact (Sections 2.3 and 2.10).
2. **Asymmetric trials with a price:** a measured price map of the objective, ideas parked with the price that would reopen them, and trials ranked by expected gain per unit of stake, with the conditions under which these rules are optimal (Sections 2.4 and 2.9).
3. **Recursion:** two forecasts on every experiment, scored by role; errata in every report; a red team on the planner's closed directions; and an arrangement of roles that the loop found by the same tests (Sections 2.5 to 2.7, and 3.2).

4. **Evidence** from six days on a live public benchmark against concurrent agent harnesses, with four cases followed from first price to accepted entry and what did not work (Sections 3 and 3.9).
5. **An open implementation:** a skill and tools that install into a coding agent with one command (Appendix B.1).

How the paper is laid out. Section 2 sets out the topology, with proofs; Section 3 is the experiment; Section 4 the related work; Section 5 the discussion and limits. The appendix adds what the topology asks of a harness and the released package (Appendix B). For the result first, see Figure 3 and Section 3.6.

2 Recursive Priced Falsification

Recursive priced falsification is an agentic topology: a set of agent roles, the information each may read, the gates a claim must pass between them, and a shared record that holds the state of the research. Six design principles give it its shape. This section states the principles, then the topology, then its parts in turn: what the planner holds (Section 2.4), what moves along the edges (Sections 2.5 and 2.6), the record (Section 2.7), the gates and the stake (Sections 2.8 and 2.9), and four supporting mechanisms in brief (Section 2.10, with Appendix C). It is not tied to a language model or to an agent framework.

2.1 Design Principles

The exact verifier of a research problem is usually expensive. If it were free there would be nothing to plan: every idea could simply be built and judged. Everything in the topology follows from that one fact, in six principles (Table 1). Each names the part of the topology that carries it.

The six follow in order from the cost of the verifier. If most ideas must be judged before they are built, the judgement needs numbers, so an idea is priced first. A judgement that cannot be wrong teaches nothing, so it is written down as a forecast before the run. It rests on the planner's picture of the problem, so every report corrects that picture. A judgement against an idea may itself be wrong, so a closure keeps its grade and the number that would undo it. Judging must cost less than building, so the cheapest test that can end an idea runs first. And since the planner makes most of the judgements, the planner is tested too.

The first, second and fifth principles apply the method to the design. The third, fourth and sixth apply it to the loop itself: to the tasks the planner writes, the directions it closes and the conclusions it holds. This is the sense in which the method is recursive, and it is why a run leaves two things behind, a better design and a corrected procedure.

2.2 Setting

A discovery loop works on a *design* x: the thing being improved. A design consumes *resources*, the quantities the objective counts. For a circuit these are gates and qubits; for a program they might be time and memory. We write the resources of x as $r(x) = (r_1, \ldots, r_k) \in \mathbb{R}^k_{>0}$.

The *objective* is $J(x) = f(r(x))$, to be made as small as possible. It is the loop's utility function with the sign turned, so that lower is better. We assume f is differentiable and increasing in each resource, so that using less of anything is never worse.

Table 1: The six design principles. The last column says whether a principle applies the method to the design or to the loop itself.

Principle	Rule	Guards against	Carried by	Applies to
Price before build	An idea gets its cost per unit of gain, read from a map of measured exchange rates, before anything is built	Building ideas that cannot pay, however clever	The instruments and the pricer	Design
Forecast before run	The planner's forecast, the worker's forecast and a falsifier are written before the run	Explaining any result after the fact	The edge from planner to worker	Design
Correct the task	Every report carries errata on the task it was given	A wrong picture of the problem that outlives the experiment	The edge back	Loop
Close with a grade and a number	A dropped idea keeps the grade of the evidence against it and the number that would reopen it	Dead ends that rest on an argument; good ideas lost to one wrong price	The shared record	Loop
Cheapest refutation first	Instruments are ordered by cost per refutation, and each is admitted on a known answer	Paying the exact verifier for claims a cheap test would have ended	The order of the gates	Design
Test the planner	A red team challenges the closed list, a breaker attacks every claim before it is kept, and the planner is rebuilt from the record after a stall	A planner anchored on its own earlier conclusions	The red team, the breaker and the rebuilt planner	Loop

Some resources do not enter the objective but are capped. We call these *budgeted*: $r_i(x) \le C_i$. An error rate that must stay under a limit is the usual case.

An *exact verifier* computes J and checks the budgets. It is the ground-truth evaluator; every cheaper test in the loop is a proxy for it. We assume it is expensive enough that most ideas must be judged before they are built. The best design it has accepted so far is the *accepted design*, or incumbent.

The loop has a frozen foundation model, tools for code and computation, and a human *operator* who sets the goal and approves actions with effects outside the loop. Nothing in the loop changes the model's weights. What the loop can change is its own scaffolding. Appendix D.1 collects the symbols used below.

2.3 The Topology of Roles

By a topology we mean five things that are fixed before a round starts and that no agent in the round can change: the *roles*; the *view* of each role, that is what it is given to read; the *edges*, each of which carries one kind of message; the *gates*, a test on each edge that a message must pass; and

the *record*, the only state that outlives a session. What is left to the agents is the content: which ideas, which tests, which designs.

Priced falsification has a planner, a set of worker roles, a verifier ladder and a shared record (Figure 1).

The *planner* holds the price map, writes tasks and re-checks results. It does not build and it does not test. *Workers* are fresh sessions. Each has one question, an isolated copy of the work and a reading list. A worker knows nothing the planner did not put in its task, which is what lets it disagree. The *verifier ladder* is a sequence of instruments from cheap to exact, most of them built by the loop itself. It is an evaluation cascade [36] whose rungs the loop builds and admits. The *shared record* holds the state of the research: the trials, the closed directions, the numbered findings and a log of techniques. No agent's conversation holds any of it. An agent that is replaced loses nothing the loop needs. The record does the work that an archive does in open-ended systems [16, 76], with one difference: it keeps directions and the evidence for and against them, the parked ones included, and not a population of candidates.

The research moves in *rounds*. A round sends work through the worker roles in a fixed order (Table 2).

Two terms in the table recur below. A *card* is a scout's proposal in a fixed form: the place in the design it would change, its size as a low, middle and high estimate, its odds, and the cheapest test that could end it. The *desk* is the worker that chooses among cards. It ranks them by odds times size over stake (Proposition 6), keeps a quota for each lens and one long shot, and may drop a card only by citing the closure it repeats.

Four properties make this a topology and not a list of helpers.

Information narrows along the round. A scout sees the whole cost map and the closed list. A pricer sees one card, a builder one priced card, a breaker only the change and the claim. The breaker's blindness is the point: it cannot inherit the builder's reasons, so what it confirms it has found again.

A scout has one lens. Each scout is told one way to look: extend what has just paid; measure where the cost sits; read outside sources and rival entries; reopen a closure. A scout with every lens writes the cards of the lens it likes best. In the run, the two ideas of size in the first round of this kind both came from the lens that measures (Section 3.7).

The planner sets no prices. A price is a pricer's test, not the planner's arithmetic. The rule came from the record: three ideas for a large step had gone into a round on the planner's own paper prices, and all three lost on a part the price had not named.

Rounds overlap. While one design waits for the exact verifier, the instruments and scouts of the next round already run on it. If a better design clears its gates in that time, it takes the verifier's queue (Appendix C.3).

Hand-overs between roles use fixed fields, a failed gate returns to the worker that owns the claim, and each round is scored by what it kept. These are ordinary practice in multi-agent systems, and we claim none of them.

Table 2: The roles of one round. Each role reads less than the one above it and returns a claim that the gate after it can test.

Role	Reads	Returns	Gate after it
Instrument	The accepted design	A ledger of its cost by block; every setting whose price has changed, refitted; each resource limit priced again	None: it is the map the scouts read
Scout (three to five, one lens each)	The task, the map, the closed list	Cards: a place, a size as low, middle and high, odds, and the cheapest test that could end the card	A desk that picks; a card it kills must cite the closure
Pricer (one a card)	One card	That cheapest test, run; a price with a range; alive or dead	The price against the card's forecast
Builder (one a priced card)	One priced card and its own copy of the design	A build behind a switch, its count, the comparison with the base, planted faults	The test for "exact" (Appendix C.1)
Breaker (fresh context)	The change and the claim, nothing else	Broken or not, by tests of its own design	Not broken
Search and exact verifier	The cleared design	An accepted entry, or nothing	The benchmark's judge

Two roles sit outside the round. A *red-team worker* attacks the closed list (Section 2.7). And when the instruments show that the accepted design is at the floor of its present form, a *clean-sheet round* sends several designers to start somewhere else (Appendix C.4).

2.4 The Price Map

An objective is usually made of a few resources, and an idea usually spends one of them to save another. Whether the idea is worth building is then a question of price: how much of the first does it spend for each unit of the second, and is that more or less than the objective itself would pay?

Take a product objective $J = T \cdot Q$ with $T = 800{,}000$ gates and $Q = 1{,}250$ qubits. Removing one qubit lowers J by T; removing one gate lowers it by Q. One qubit is therefore worth $T/Q = 640$ gates. An idea that frees a qubit for 500 gates pays. An idea that frees one for 800 gates does not, and no further thought about its merits is needed. The definitions below make this general.

Definition 1 (Objective rate)**.** The objective rate between resources i and j at a design x is

$$\pi_{ij}(x) = \left.\frac{\partial f/\partial r_i}{\partial f/\partial r_j}\right|_{r(x)}, \tag{1}$$

the amount of r_j that is worth one unit of r_i to the objective.

The objective rate comes from the objective alone. It says what a trade is *worth*. What a trade *costs* depends on the design, and has to be measured.

Definition 2 (Knob and design rate)**.** A knob a is a one-parameter family of designs $x_a(s)$ through x. If turning it spends r_i and saves r_j, its design rate is

$$\rho_a = -\frac{\mathrm{d}r_j/\mathrm{d}s}{\mathrm{d}r_i/\mathrm{d}s}, \tag{2}$$

the units of r_j saved per unit of r_i spent. Design rates are measured, by building both settings and running them on identical inputs (Section 2.8).

A knob is anything that can be turned by degrees: the width of an approximation, the size of a table, the number of spare wires. The *price map* is the table of all objective rates π_{ij} and all measured design rates ρ_a. Three rules are read off it.

Rule 1: break-even.

Proposition 1 (Break-even)**.** *Suppose a small turn of knob a changes only r_i and r_j, with $\mathrm{d}r_i > 0$ and $\mathrm{d}r_j = -\rho_a\,\mathrm{d}r_i$. Then $\mathrm{d}J < 0$ if and only if $\rho_a > \pi_{ij}$.*

Proof. Substituting $\mathrm{d}r_j = -\rho_a\,\mathrm{d}r_i$ and $\partial_i f = \pi_{ij}\,\partial_j f$,

$$\mathrm{d}J = \partial_i f\,\mathrm{d}r_i + \partial_j f\,\mathrm{d}r_j = \partial_j f\,(\pi_{ij} - \rho_a)\,\mathrm{d}r_i.$$

Since $\partial_j f > 0$ and $\mathrm{d}r_i > 0$, the sign of $\mathrm{d}J$ is the sign of $\pi_{ij} - \rho_a$. □

In words. A trade improves the objective exactly when the design gives more than the objective asks. An idea can therefore be priced before it is built, and an idea whose rate is on the wrong side of π_{ij} is parked with that price written next to it.

Resources often come in whole units: one qubit, one iteration. Then the condition is exact, not a first-order one.

Corollary 1 (A discrete optimum at a kink)**.** *Let $J = TQ$. Let c^- be the amount of T added when one unit of Q is removed, and g^+ the amount of T saved when one is added. Then Q is a strict local minimum of J along this knob if and only if*

$$g^+ < \frac{T}{Q+1} \qquad \text{and} \qquad c^- > \frac{T}{Q-1}. \tag{3}$$

Proof. Adding a unit of Q and removing one change the objective by

$$J(Q+1) - J(Q) = (T - g^+)(Q+1) - TQ = T - g^+(Q+1),$$
$$J(Q-1) - J(Q) = (T + c^-)(Q-1) - TQ = c^-(Q-1) - T.$$

The first is positive exactly when $g^+ < T/(Q+1)$, and the second exactly when $c^- > T/(Q-1)$. □

In words. The right number of qubits is the one where adding a qubit saves too few gates and removing one costs too many. Two measurements, g^+ and c^-, settle whether a cap is in the right place. Appendix E.1 reports them for the benchmark.

Rule 2: spend a budget where it buys most. A budgeted resource has no objective rate, because the objective does not count it. The question there is not whether to spend the budget but where.

Proposition 2 (Equimarginal rule)**.** *Let a budget Λ of resource λ be spent across knobs $a = 1, \ldots, m$ in amounts $s_a \geq 0$, $\sum_a s_a \leq \Lambda$, and let knob a return $G_a(s_a)$ units of an objective resource, with each G_a concave, increasing and differentiable. An allocation with $\sum_a s_a = \Lambda$ maximises $\sum_a G_a(s_a)$ if and only if there is $\mu \geq 0$ with $G_a'(s_a) = \mu$ for every knob with $s_a > 0$ and $G_a'(0) \leq \mu$ for every knob with $s_a = 0$.*

Proof. The problem is the maximisation of a concave function under linear constraints, so the Karush–Kuhn–Tucker conditions are necessary and sufficient [4]; μ is the multiplier of the budget, its shadow price. □

In words. At the best allocation, the last unit of budget buys the same on every knob in use. If it buys more on one knob than on another, the allocation is not the best, and the next corollary says by how much.

Corollary 2 (Gain from reallocation)**.** *If two knobs have marginal rates $\rho_a > \rho_b$ at the current allocation, moving δ of the budget from b to a changes the return by $(\rho_a - \rho_b)\,\delta + O(\delta^2)$ at an unchanged total λ.*

In words. Suppose an error budget can be sold for gates on two settings, and the last error sold on the first bought 1,200 gates while the last on the second bought 500. Buy one error back on the second and sell it on the first: about 700 gates gained, and the error rate has not moved. Such gaps are not rare. In the run, ten settings sold the same failure budget at prices from 133 to 994 gates a failing shot (Appendix E.1).

Rule 3: read a rival's design as a list of trades. Corollary 2 is also how another party's design is read. It is taken apart into the trades it makes. Each trade is re-priced on one's own measured scale. Any trade made at a worse rate than one already available is a gain waiting to be taken. In the run, the settings that other entrants had turned sold a failing shot for about half the gates that the loop's own refit of its safety table obtained; entry E5 reversed two of their trades and made the same trade at the better rate (Section 3.3).

What a price is not. A price is local in two ways. It is a first-order quantity, so it does not see how two knobs interact. And it belongs to the place where it was measured: the same wire can cost a hundred gates in one part of a design and half a gate in another. Both limits showed up in the run (Section 3.6 and Appendix E.1). The technique's answer is the same in both cases: prices choose what to try, and the thing chosen is then measured whole.

Prices of this kind are standard in optimisation and in hardware design [2, 4, 11]. We did not find them as the planning object of a research agent.

2.5 Two-Sided Forecasts

A planner that writes a task, chooses its test and reads its result has three chances to protect its own picture of the problem. A forecast written before the run removes one of them: the result can

no longer be explained afterwards as what was expected. A second forecast, written by someone else, removes another.

A task is issued as a trial. The planner writes a question q, its forecast $\hat{y}_P$ of the result, and a falsifier ϕ: the outcome that ends the idea. Before its first run the worker writes its own forecast $\hat{y}_W$. After the run the ledger receives one line, whether the experiment succeeded or failed:

$$b = (q,\ \hat{y}_P,\ \hat{y}_W,\ \phi,\ y,\ \text{kept}). \tag{4}$$

The ledger is read by role. Forecasts in a research run span many orders of magnitude, from a fraction of an error to thousands of gates, so an error is measured as a ratio. For a positive quantity the log error of a forecast is $e = \ln(y/\hat{y})$: zero for a perfect forecast, negative when the forecast was too high, and ± 0.69 when it was off by a factor of two. The mean $\bar{e}$ over a round estimates a role's bias. The planner reviews its misses after each round and is meant to apply that bias to its next forecasts.

A fired falsifier ends a line of work when it is measured, without argument. That is its whole purpose: the condition for stopping was agreed before anyone had a result to defend. In the first fifty hours of the run four falsifiers fired, and each ended its line within the hour.

A forecast with a falsifier before a run is established practice [9, 15, 50]. The second forecast is what the technique adds. It carries information before the experiment is run.

Proposition 3 (The gap bounds the error)**.** *Let $e_P = \ln(y/\hat{y}_P)$ and $e_W = \ln(y/\hat{y}_W)$. Then*

$$\max(|e_P|, |e_W|) \ \geq\ \tfrac{1}{2}\,|\ln \hat{y}_P - \ln \hat{y}_W|. \tag{5}$$

If moreover the errors have biases β_P, β_W, variances σ_P^2, σ_W^2 and correlation r, then

$$\mathbb{E}[(\ln \hat{y}_P - \ln \hat{y}_W)^2] = (\beta_P - \beta_W)^2 + \sigma_P^2 + \sigma_W^2 - 2r\sigma_P\sigma_W. \tag{6}$$

Proof. The outcome cancels in the difference of the two errors: $\ln \hat{y}_P - \ln \hat{y}_W = e_W - e_P$. Hence

$$|\ln \hat{y}_P - \ln \hat{y}_W| \ \leq\ |e_P| + |e_W| \ \leq\ 2\max(|e_P|, |e_W|).$$

The second statement is the expectation of $(e_W - e_P)^2$, written out in biases, variances and the covariance $r\sigma_P\sigma_W$. □

In words. The first inequality needs no assumption. When two forecasts of the same number differ by a factor k, at least one party's model of the problem is wrong by a factor $\sqrt{k}$ or more, whatever the outcome turns out to be. A wide gap marks an experiment that is certain to teach the loop something, and the gap is known before any compute is spent.

The second identity says what the gap cannot see. If the two roles tend to err together ($r > 0$), the gap shrinks while both errors stay large. Two forecasts that agree are not thereby right. We return to this in Appendix E.4, where the run's ledger shows exactly that.

2.6 The Brief as a Hypothesis

A brief is the document a worker receives: the question, the facts about the current design that the planner believes, a target, a test and the bounds of the task. It encodes the planner's model of the problem, and that model is often wrong.

A report returns four things: the result, *what was wrong in the brief*, what was not checked, and technique cards for the record. The second is required, not invited. A worker that finds its brief wrong can always answer the question as posed, so "nothing was wrong" must be written as a claim. The planner corrects the brief and its own notes from the errata before it plans again. Errata are of three kinds:

- **Wrong facts about the design.** A count, a cost or a location that the planner carried from an earlier round and that no longer holds.
- **A wrong target.** A proxy was briefed where the real quantity was something else, so a worker could hit the target and gain nothing.
- **A wrong falsifier.** The stated stopping condition could not have detected the outcome that occurred.

The rule costs a few lines a report. Its effect is that a mistaken picture of the problem lasts one round, not the whole run. In the run every build report of the rounds behind E4 and E6 corrected its brief, with three to ten items each. The report that reopened the idea behind E6 corrected six points and questioned the price of a wire on which the idea had been parked; the next worker found that the price had been measured in another part of the design and did not hold where the new wires were needed (Section 3.6).

2.7 Closures: Grades, Parking and Reopening

Most ideas in a research run end as dead ends. How a loop records them decides how much it later loses to its own mistakes, because a wrongly closed direction is never looked at again.

Grades. A direction is closed only by evidence of a named grade. Table 3 lists the grades in increasing order of strength. The line between parking and closing is whether anything was run: an argument can fail in a way that nothing in it shows, while a number can be checked. The grade also says how far a closure can be trusted. In the run four closures were later overturned by a build; all four rested on a paper price or a model, and none that rested on a proof or an exhaustive search has been overturned.

Parking. A direction dropped by argument alone is *parked*, not closed, until a short pilot backs the argument. An idea that is sound but too dear at today's prices is parked as well. Parked ideas stay on the planner's list. They are the loop's stepping stones: kept though they do not pay today, because a later price may make them pay.

Reopening conditions. Every closed or parked row carries a *reopening condition* written as a number, for example "reopens if step s can be done for under c units" (Figure 2). The number names the weakest link in the case against the idea. A reopening condition turns a dead end into a question that one worker can answer in one sitting. The idea behind E6, the largest step of the run, was parked with "reopens if the three leftover wires can be erased for under 20 gates"; one worker answered in forty minutes with an erase that costs nothing (Section 3.6).

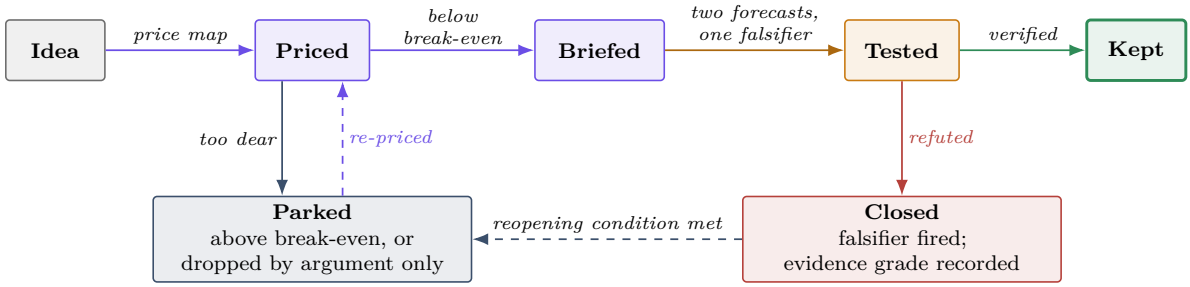


Figure 2: The life of a claim. An idea is priced before it is briefed. A refuted idea is closed with the grade of the evidence against it. An idea that is too dear, or that was dropped by argument alone, is parked. Each closed or parked idea carries a numeric condition that reopens it.

Table 3: Evidence grades for a closure, weakest first. A direction dropped on an argument alone is parked, not closed.

Grade	What the closure rests on	Effect
Argument	Reasoning alone, with nothing run	Parks, until a pilot backs it
Paper price	Arithmetic on unit costs that were measured elsewhere	Closes
Simulation	A model of the design, run on many inputs	Closes
Measurement	The real design, built and run	Closes
Exhaustive search	Every case of a bounded space, checked	Closes
Proof	A derivation that covers all cases	Closes

The red team. A *red-team worker* receives the closed list and has one task: to break it. Counterexample agents in earlier systems are aimed at new positive claims [15, 18, 23, 68]. This one is aimed at the planner's negative conclusions, which are where an anchored planner loses the most. In the run the red team attacked eleven closures. Every conclusion held, and two of the planner's reasons for them were wrong.

The breaker. The red team attacks what the loop has given up. The *breaker* attacks what the loop is about to keep. It is a fresh session, on a different model where one is available, and it is given only the change and the claim made for it. It writes its own tests. A claim enters the record after the breaker has failed to break it, not after the builder has finished testing it. In the run a breaker found an error in the incumbent that no entrant had seen (E11), and another showed that an accepted entry was not exact (E14; Sections 3.7 and 3.9).

2.8 The Verifier Ladder

Because the exact verifier is expensive, the loop builds its own instruments and orders them by cost. A claim is tested on a rung and climbs only if it survives. The question is which rung should come first.

Proposition 4 (Order of the rungs)**.** *Let rung i cost c_i and refute a claim that has reached it with probability p_i, independently of the other rungs. Testing in the order $1, \dots, k$ until the first refutation has expected cost*

$$C = \sum_{i=1}^{k} c_i \prod_{j<i} (1 - p_j), \tag{7}$$

and C is least when the rungs are sorted by increasing c_i/p_i.

Proof. Exchange two adjacent rungs u, v. The terms before them are unchanged, and those after them share the factor $(1-p_u)(1-p_v)$. The order u, v costs $c_u + (1-p_u)c_v$ at that position and the order v, u costs $c_v + (1-p_v)c_u$. The first is smaller if and only if $c_u p_v < c_v p_u$, that is $c_u/p_u < c_v/p_v$. Any order can be sorted by such exchanges without raising the cost. □

In words. Rungs are ordered by cost per refutation, not by strength. Suppose a model takes one minute and catches a third of wrong claims, and a build takes ten minutes and catches nine in ten. The ratios are 3 and 11, so the model goes first, although it is the weaker test. A cheap instrument with even a modest chance of ending a claim belongs in front of an exact one. This is the classical rule for ordering a sequence of tests [35, 49]; what the topology adds is a way to supply its c_i and p_i.

The run's ladder had seven rungs, most of them built by the loop (Table 24): an integer identity checked on millions of inputs and a gate-level model of one block, minutes each; the builder's gate count, under a minute; a paired run of two circuits on identical inputs, about ten minutes; a breaker's own tests; a search for a passing packet, hours long and $0.19 to $1.77 of rented compute; and the official judge. Prices on paper and pilots on the cheap rungs ended 55 of the 64 ideas they tested, before any build (Section 3.5).

Two rules keep the cheap rungs honest.

Admission. A new instrument I is used only after it reproduces a number that is already known on the incumbent design x_0:

$$\text{admit } I \iff |I(x_0) - J(x_0)| \le \epsilon. \tag{8}$$

An instrument that has never been run on a case with a known answer has not been tested; its first disagreement with reality will be blamed on the idea.

Pairing. Two designs are compared on identical inputs, never across independent draws. The next proposition says how much this buys.

Proposition 5 (Paired estimator)**.** *Let designs A and B be run on the same n random test packets, with failure counts $F_{A,t}$, $F_{B,t}$ of mean about λ, and let $\bar{D}$ be the mean of $D_t = F_{A,t} - F_{B,t}$. Write $F_{A,t} = S_t + U_{A,t}$ and $F_{B,t} = S_t + U_{B,t}$, where S_t counts the failures the designs share and U the failures of one design only. If the counts are Poisson and U_A, U_B are independent, then*

$$\operatorname{Var}(\bar{D}) = \frac{\kappa}{n}, \qquad \kappa = \mathbb{E}[U_A] + \mathbb{E}[U_B], \tag{9}$$

against $2\lambda/n$ for two independent samples. Pairing multiplies the effective sample size by $2\lambda/\kappa$.

Proof. $D_t = U_{A,t} - U_{B,t}$; the shared part cancels exactly. The variance of a difference of independent Poisson counts is the sum of their means. For independent samples $\mathrm{Var}(F_A - F_B) = 2\lambda$. □

In words. Two versions of one design fail on mostly the same inputs. Run on the same inputs, those shared failures cancel, and only the failures that differ add noise. When two designs differ by one small change, almost nothing differs, and a comparison that would need tens of thousands of independent packets needs a few hundred shared ones.

Remark. A repair that fixes a failures a packet may break b others. If it costs g, its true price is $g/(a-b)$ per failure removed. Counting only the failures it fixes gives g/a, too cheap by the factor $a/(a-b)$. The paired run measures a and b separately.

2.9 The Stake and the Gain of a Trial

The parts above fix what a trial costs and what it can return. Let a card have size G, the fall in the objective if it is kept, and let it face rungs $1, \dots, k$ with costs c_i and refutation probabilities p_i as in Proposition 4. Its *stake* is the expected cost C of that proposition. Its *odds* are $P = \prod_i (1 - p_i)$, the chance that it passes every rung.

A trial is an asymmetric bet in a precise sense. The loss is bounded and known before the run: it is at most $\sum_i c_i$, and in expectation C, most of it spent on the cheap rungs. The gain is not bounded by the stake, because G is set by the design and not by what the test cost. A parked idea is the limiting case: it is held at no stake until its reopening number is met.

Proposition 6 (Ranking trials)**.** *Let cards $a = 1, \dots, m$ have stakes C_a, odds P_a and sizes G_a, and let a round have a budget B of test cost, with every C_a small against B so that a card may be funded in part. The expected gain $\sum_a x_a P_a G_a$ over $x \in [0,1]^m$ with $\sum_a x_a C_a \le B$ is greatest when the cards are funded in decreasing order of $P_a G_a / C_a$.*

Proof. Suppose a card a is funded while a card b with a higher ratio is not fully funded. Moving a stake ϵ from a to b changes the expected gain by $\epsilon\,(P_b G_b / C_b - P_a G_a / C_a) > 0$. No such pair is left exactly when the cards are funded in decreasing order of the ratio. □

In words. The proposition is the greedy solution of a fractional knapsack, and when only the best result found is kept it gives way to Weitzman's reservation index [61]. A card is ranked by expected gain per unit of stake, not by its chance of success. A card with odds of one in ten is the better use of a worker than a near-certain one if it is thirty times larger at the same stake. Three parts of the topology supply the ratio. The price map gives G before anything is built. The ladder makes C small (Proposition 4). And the odds P come half from the scout that wrote the card and half from the record of what that kind of trial has paid before (Appendix C.2). The two forecasts of Section 2.5 are how the loop learns whether its odds were right.

The consequence for a plan is that most trials should lose, and should lose early. A loop whose trials mostly succeed is staking too little on large ideas; a loop whose trials lose on the dear rungs has its ladder in the wrong order. Section 3.5 reads the run's 133 trials against this shape.

2.10 Supporting Mechanisms

Four more parts of the topology follow from the principle that a gate must be a test. Appendix C gives each in full, with its proposition.

Which comparison decides that a change is exact. A verifier that draws random inputs indexes its random numbers by position, so a change that adds or moves one draw shows the two designs different randomness on the same input, and pairing is undone (Proposition 7). The comparison is therefore chosen by what the change does to the indices: an aligned pair, a control pair, or a list of approximate elements by structure. A change that is exact only by an argument goes to a breaker, who derives the inputs on which the argument would fail and draws inputs there (Appendix C.1).

Instruments before ideas. After every accepted entry, and before any scout runs, three instruments are read on the new design: a ledger of its cost by block, a refit of every setting whose price has changed, and each resource limit priced again. Cards are written against that map. A card from the ledger arrives with its price; a card from a list of ideas must be priced from nothing. Every trial is filed under a technique family, and that family's pay rate so far is the prior for the next trial of its kind (Appendix C.2).

Search with no memory. Where the exact verifier is a search for a passing instance, tries are independent, so the tries already spent say nothing about those to come. A search is moved to a better design the moment one clears its gates (Proposition 8), and where entries are public each cleared improvement is sent at once (Appendix C.3).

Fresh contexts. A worker reads only what its brief lists, and one worker a round may be told to read no earlier report. After a round that keeps no gain, the next round is planned by a new session built from a digest of the record. When the ledger shows every large block at its floor, a clean-sheet round sends designers who each start from a different place (Appendix C.4).

2.11 One Round

Algorithm 1 One round of priced falsification

1: Read the instruments on the accepted design x_0: the ledger by block, the refit, the limits. Read one outside source
2: Update the price map $\{\pi_{ij}, \rho_a\}$ and the record by technique
3: Scouts, one lens each, write priced cards; the desk ranks them by PG/C, picks, and cites a closure for each card it kills
4: **for** each picked card q **do**
5: Planner writes the brief $(q, \hat{y}_P, \phi, \text{reading list}, \text{bounds})$ and starts a fresh pricer
6: Pricer writes $\hat{y}_W$, runs the cheapest admitted test that can refute, and returns $(y,\ \text{errata on the brief},\ \text{what was not checked},\ \text{technique cards})$
7: If the card lives, a builder builds it behind a switch and runs the comparison of Appendix C.1
8: A breaker, given only the change and the claim, tries to break it
9: Planner re-runs the claimed result; appends the trial b to the ledger; applies the errata
10: **end for**
11: Close or park each direction with its evidence grade and reopening condition
12: Send a red-team worker at the closed list
13: Combine the kept changes into one package and refit it once
14: Pay the exact verifier; move the search to a better package the moment one clears
15: **if** the round kept no gain **then** rebuild the planner from a digest of the record
16: **end if**
17: **if** the ledger shows every large block at its floor **then** run a clean-sheet round
18: **end if**

Algorithm 1 puts the parts in order. Three lines need a word.

Line 1 keeps the loop from feeding only on itself: every round starts from one thing read outside and one thing measured inside.

Line 10 is the planner's re-check. A worker's claim is not a result until the planner has re-run the test behind it, on fresh inputs or from a second copy of the work. Only then does the trial go to the ledger.

Line 14 sets the bar for building on the *package*, not on each idea. Several changes that share one refit are judged together, because a change too small to pay for a refit alone may pay inside one that is happening anyway.

The rules are themselves part of the record and are revised between rounds. This is the loop's self-modification, and it is confined to the scaffold: the loop can rewrite its rules, its briefs and its instruments, and it cannot change the foundation model, the exact verifier or the gate on outside effects. Early rules came from the loop's own mistakes. After a round that kept nothing, a worker surveyed recent work on research loops, and three rules were adopted the same day: a forecast with a falsifier before each run [15, 26, 50]; evidence grades on closures, with a pilot before an idea is dropped by argument, on evidence that choices between directions made by reasoning alone are weak and that short pilots improve them [12, 21, 38]; and a rebuilt planner after a stall [9, 80].

The arrangement of roles was revised in the same way: the loop priced candidate roles against its record and ran the new arrangement once as a trial before keeping it (Section 3.2).

What the topology asks of the software. The topology needs seven things of the software that runs the agents, which we call the *harness*: separate contexts, isolated copies of the work, a shared record outside every session, scoped reading, parallel bounded execution, shared instruments, and a gate on effects outside the loop (Appendix B). Together they describe a distributed harness. The distribution is not there for speed. It is what makes the planner testable.

3 Experiments

We ran the technique for six days on a live public benchmark, with other agent harnesses at work on the same problem. We first give the outcome against those systems, which are our baselines. We then follow four cases from first price to accepted entry, analyse each part of the technique against the run's own records, and report the negative results.

3.1 Experimental Setup

The benchmark. ECDSA.fail is a public benchmark for one task: the cheapest quantum circuit for 256-bit elliptic-curve point addition, the core of Shor's algorithm against elliptic-curve signatures. Its organisers describe it as open autoresearch: entrants, most of them running coding agents, improve one shared circuit, and every accepted entry becomes public with a note on how it was made [30]. Any strictly better score is accepted. In practice each entry starts from the best one before it.

Four terms are enough to follow the results (Table 4).

The loop as run. The loop ran the topology of Section 2 on Pensieve, our research harness, which supplies the capabilities of Appendix B. The planner and every scout, pricer and builder ran on one foundation model, Claude Opus 5.5, with no fine-tuning; breakers ran on a second model. The run covers 138.7 hours and 42 rounds. At its end the record held 174 numbered findings, 267 ledger lines, 333 technique cards and 67 logged mistakes of the planner, each with the rule that changed. The topology was built up during the run. Until hour 75 it was a hub: the planner, fresh workers with one question each, the red team and the shared record, under the principles of Section 2.1. From hour 75 the workers of every round were arranged as the roles of Table 2, an arrangement the loop drew from its own record (Section 3.2). The two largest steps came from the hub; the cases of Section 3.7 came from the full topology. Appendix D maps each part of the technique onto the benchmark.

Baselines: concurrent systems. The benchmark gives a comparison that a private experiment cannot: other entrants were improving the same circuit, under the same judge, in the same hours. Four had entries accepted during the run (Table 5). We name them by the header of their public notes, which states a model and a harness. Every entry is public, so an entry can build on another within the hour; we report those cases separately. These systems are the baselines of this paper. They are concurrent and not controlled: operators, compute and hours differ (Section 5).

Autonomy and human oversight. All research reported here was carried out by the agents: reading the literature and the public entries, forming and pricing hypotheses, writing every brief, tool, proof, simulation and circuit change, running the searches, judging the results, writing the

Table 4: The benchmark in four terms.

Term	Meaning
Gates and qubits	The score is $J = T \cdot Q$: the mean number T of executed Toffoli gates times the peak number Q of qubits. These are the two objective resources. We say "gates" for Toffoli gates throughout.
Failing shots	The circuit is approximate by design. An entry is judged on a *packet* of 9,024 random test shots and must pass every one. The expected number λ of failing shots a packet is therefore a budgeted resource: a circuit can be made cheaper by letting it fail more often.
The search	The shots of a packet are derived from the entry itself, which includes one free number. An entrant must find a value of that number for which no shot fails. The cost of that search grows as e^{λ} (Appendix E.3).
Iterations and reductions	The circuit's main loop runs about 140 iterations in each direction. In most of them it adds three values to a 256-bit register and reduces the register modulo the curve's prime after each addition. Additions and shifts are where most of the gates are; the reductions are about 7% of them.

Table 5: The systems compared. Harness and model are as stated in the header of each entrant's public notes; entries are those accepted during the run.

System	Harness	Model	Entries
This work	Pensieve, running priced falsification	Claude Opus 5.5 for every role; breakers on a second model	17
Entrant A	General coding agents	The loop's base model (5 entries), another frontier model (11), both together (3)	19
Entrant B	A general coding agent	The loop's base model	4
Entrant C	Several research lanes inside a general coding agent	The loop's base model	1
Entrant D	A general coding agent	The loop's base model; its entry re-drew the test packet of the entry before it	1

public notes and the records, and revising the loop's own rules. No person wrote or edited code, proposed an idea about the circuit, designed or ran an experiment, or checked a result. The loop did not pause to ask how to proceed with the research.

The human operator did three things. They set the goal. They decided on actions with an outside effect: for the first 86 hours they gave a go or no-go before each submission and each rental of compute, and at hour 86.5 they replaced the go before a submission with a standing rule, that every improvement which passes the loop's own tests is submitted at once. The last five entries went out under that rule. And from time to time they sent a short direction in plain language. The run's records hold twenty-eight: constraints, approvals and directions on research strategy. None concerned the content of the circuit. We report this so that the claim of autonomy can be checked against the record.

3.2 How the Topology Was Found

The arrangement of roles in Table 2 was not fixed before the run. The loop arrived at it on the fourth day by the means it uses on the circuit: it priced candidate parts against its own record, ran the arrangement once as a trial with a forecast written first, and cut the part that earned nothing (Table 6).

A hub first. For its first 75 hours the loop was a hub: a planner, fresh workers with one question each, a red team and the shared record. Its program went through ten versions in the first day, each tied to a decision or a finding in the record (Section 2). On the fourth day three rounds of the hub, run as lanes by the planner, kept nothing.

Which parts earned a seat. Before it changed its form, the loop tested candidate parts on rows of its record whose outcomes were known. An outside decision model, tried as a forecaster, gave worse odds on the 90 scored trials than the record's own pay rate by stage, and was given no seat as a forecaster. On three further jobs it was good at one, finding the closed direction that a reworded proposal repeats, and poor at naming a trial's technique family or stage; it was given the one seat, as a gate. Two facts in the record set the rest. A closed direction is not dead: one closed in round 14 paid 6,303 gates in round 15, so a closure points to its reason and does not forbid. And odds alone must not choose: the least likely third of the scored trials held the pilots behind the two largest steps, so the desk ranks by odds times size over price, with a quota for each lens and one long shot. From these the loop drew the arrangement: scouts with one lens each, a desk, pricers that run the cheapest test first, builders, a breaker in a fresh context on another model, and scripts as the only judge.

The arrangement as a trial. At hour 75 the loop ran the arrangement once on the live benchmark, as it runs any trial: the forecast and the falsifier were written before any agent ran, and the baseline was the hub of the same day. The round went from an empty list of cards to an accepted entry, E11, in under three hours (Section 3.7). The gate earned nothing. It named a closed direction on five of the 22 cards, and each of those cards' scouts had already cited it, so the desk's picks did not change. With the closed list in the scouts' brief the gate had no work left, and it was removed. The roles, one lens to a scout, the cheapest test first and the breaker on another model were kept, and every round after hour 75 ran on them.

What this shows, and what it does not. The method was applied to its own arrangement. The parts were priced against the record, the whole was run as a trial with a forecast written first, and the part that earned nothing was cut. This is the sense of *recursive* that the paper claims. It does not show that the arrangement is the best one for the task. It was compared with one alternative, the hub of the same day, over one round and not under a matched budget, and one round is one sample. The record's one later sign is a step in failed builds: 1 of 12 after the change kept nothing, against 10 of 35 before (Section 5).

3.3 Main Results

The best score fell from 997,532,360 to 940,337,645, by 5.73% (Figure 3). The loop's seventeen entries account for 70.8% of the fall (Table 7). Each was the best on the benchmark when it was accepted, and the last held the top for eight hours after the run ended. The loop did not hold the top throughout: in a public race every accepted entry is a starting point for everyone. The loop

Table 6: How the loop arrived at its arrangement of roles. Hours are from the start of the run. Each test was run on rows of the record whose outcomes were already known, except the last, which was a round on the live benchmark.

Hour	What the loop tested	Result	What changed
0 to 23	Its own program, against the mistakes and findings of each round	Ten versions of the program in the first day	Rules such as tests on the same random inputs, a forecast and a falsifier before each run, a fresh planner after a stall
72	An outside decision model as the loop's forecaster, on the 90 scored trials	Brier score 0.206 at best, against 0.149 for the record's own pay rate by stage	No seat as a forecaster
73	The same model on three jobs with known answers	The closed direction a reworded proposal repeats: first in 10 of 12; technique family: 62% of 227; stage: 42% of 90	A seat as a gate on closed directions, and no other
73	The record's own lessons	A closed idea paid 6,303 gates one round later; the least likely third of trials held the pilots behind the two largest steps	Closures are pointers, not bans; the desk ranks by odds times size over price, with a quota for each lens and one long shot
75 to 78	The arrangement as one trial, forecast and falsifier written first; the baseline, the hub of the same day, whose three rounds had kept nothing	Entry E11, 797.8 gates (0.104%), above every forecast; the gate caught nothing on 22 cards	The gate removed; the roles kept for every later round

held the top for 49 of the 139 hours, lost it eleven times and took it twelve times. Appendix D lists all seventeen entries with the part of the technique behind each.

E4: the round after a round that kept nothing. A wide round of seven workers looked for a step of 1% inside the incumbent design and kept nothing. Under the rotation rule the next round was planned by a rebuilt planner. It agreed with the old planner's reading of the evidence and changed one rule: the bar for building should apply to a package of changes that share one refit, not to each idea. Two changes that were below the bar alone (0.136% and under 0.1%) were built for that reason. Refitted together with a new decision rule (0.675% alone), they gave 1.027%, more than the sum of the parts. In the same round three falsifiers fired, and each ended the line it was written for within the hour. The worker told to read no earlier report found an exact relation in the circuit that nine rounds had not looked for.

E5: prices against settings. After E4, five entries by Entrants A and B each took E4 and changed settings that trade a few tenths of a failing shot for gates. The loop measured every such knob on identical inputs (Appendix E.1) and found that the knobs the others had turned sold a failing shot at about half the price that its own refit of the safety table obtained. E5 reverses two of their trades and makes the same trade at the better rate. It reached the benchmark within an hour of the entry it replaced.

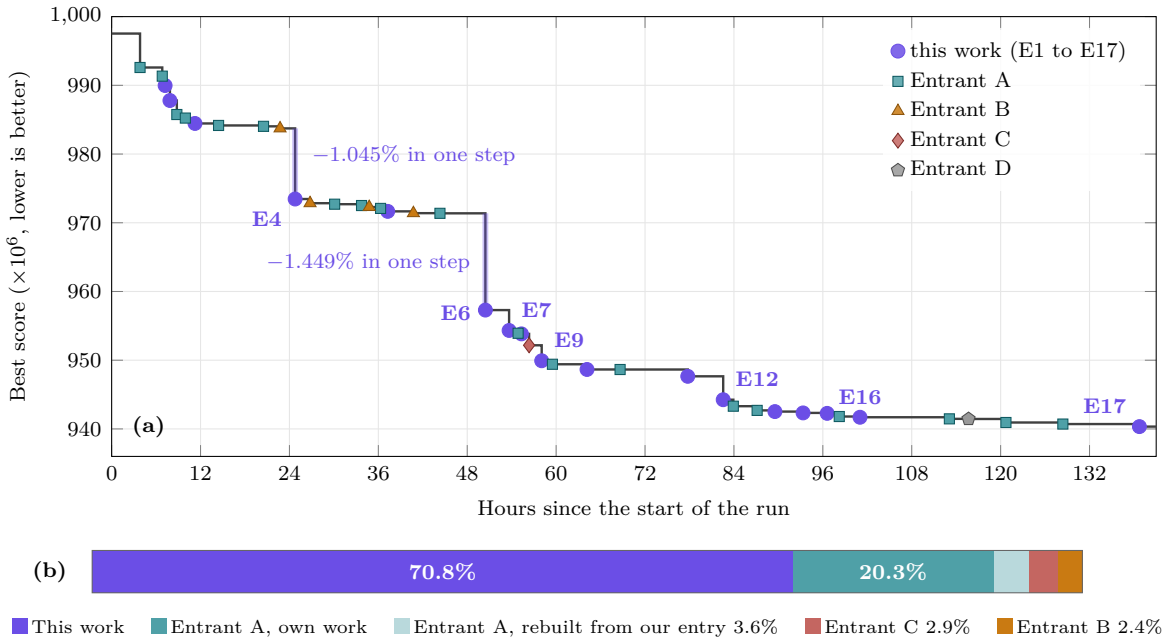


Figure 3: The benchmark over the run. (a) The best score over 138.7 hours. Forty-two entries were accepted: seventeen from the loop run with priced falsification (E1 to E17), nineteen by Entrant A, four by Entrant B and one each by Entrants C and D. The shaded drops are entries E4 and E6, the two largest steps of the period. (b) Each system's share of the total fall. The light teal segment is one entry that Entrant A rebuilt from our public entry. Entrant D's share, 0.03%, is too small to draw.

Table 7: Who moved the benchmark. Shares are of the total fall of 57,194,715 points over 138.7 hours.

System	Entries	Share	Mean step	Largest step
Priced falsification (this work)	**17**	**70.8%**	**0.245%**	**1.449%**
Entrant A, own work	18	20.3%	0.066%	0.495%
Entrant A, an entry rebuilt from our public entry	1	3.6%	0.207%	0.207%
Entrant C	1	2.9%	0.174%	0.174%
Entrant B	4	2.4%	0.036%	0.063%
Entrant D	1	0.03%	0.002%	0.002%
By note header: the loop's base model in a general coding agent	11	21.1%	0.112%	0.495%
By note header: another frontier model in a general coding agent	11	6.4%	0.034%	0.127%
By note header: both models, in two general coding agents	3	1.7%	0.034%	0.055%

E6: the largest step came from the parked list. E6 is not a new idea of its round. It had been priced two rounds earlier, found too dear and parked with a number attached. Section 3.6 follows it from that number to the benchmark.

E7 to E17: smaller steps, a similar share. In the 88 hours after E6 the benchmark fell by a further 1.70% of its starting score (1.77% of the score after E6) in twenty-two entries, eleven of them the loop's, with 67.1% of that fall. The steps were smaller for every system: the design

that all entrants shared was two days older and each of its large blocks was closer to its floor (Appendix E.6). Four of the eleven show a part of the technique that the first fifty hours did not. E11 is the first round run through the whole topology. E12 is a published technique placed by the loop's own trace, and the setting it unlocked. E15 moves the other factor of the score, a qubit, for the first time since E4, and was held at a gate before it was sent (Section 3.7). E17 lends a wire that only held a copy, on a design whose owner had bound its gate list to a hash, and its verdict at the breaker's gate was read twice (Section 3.7). Three entries were ports: E9 merged the loop's latest rounds onto Entrant C's entry 102 minutes after it appeared, E13 put four exact changes onto Entrant A's entry 2 hours 26 minutes after it took the top, and E16 put the two largest changes of Entrant A's next entry onto the loop's design 2 hours 52 minutes after that entry took the top.

Are the loop's steps larger? The other systems were not idle: they had 25 entries accepted to the loop's 17. The loop's share comes from the size of its steps and not from their number (Table 8, Figure 4). The two largest steps of the period are both the loop's, and so are four of the five steps of 0.3% or more; the third largest, 0.495%, is Entrant A's. Under the null hypothesis that step sizes do not depend on who made the entry, every choice of 17 of the 42 steps is equally likely to be the loop's. The rank sum of its seventeen steps is 429 of a possible 578, and the exact probability of a rank sum that large is $p = 0.053$ (one-sided). The share of the fall is dominated by two steps: a random seventeen of the forty-two steps reach 70.8% with probability $p = 0.020$ (permutation test, 400,000 draws).

The rank test is weaker over six days than over two. The loop's last five entries are among the smaller steps of the run (0.018%, 0.020%, 0.007%, 0.011% and 0.040%). They were sent because the loop's rule in a public race is to send each tested improvement at once (Appendix C.3); a rule that held small steps back would have a better rank sum and a worse score.

The same model in other harnesses. By their note headers, eleven of the twenty-five other entries were made with the loop's base model inside a general coding agent. Together they produced 21.1% of the fall with a mean step of 0.112%; the loop produced 70.8% with a mean step of 0.245%. Four of those eleven are one of the loop's entries with settings changed or rebuilt, and three more start from one and add work of their own. This is an observation, not a controlled experiment: the systems differ in operator, compute and hours (Section 5).

3.4 The Pace of the Benchmark

The benchmark is public and has run since May. By the morning of 15 October its history held 625 accepted steps, so the six days of the run can be set against the benchmark's own pace before and after them (Table 9). We split the history at two events. At 08:47 on 7 October an entrant posted a new circuit design that cut the score by 9.7%; the run began seven hours later on that design, and the step itself is left out. At 10:44 on 13 October the loop's last entry, E17, was accepted; the loop's work on the benchmark ended three hours later, and the other entrants went on without it on the same design.

Twelve times the pace. In the five weeks before the run the best score fell by 0.083% a day, and each tenth of a per cent took 29 hours. During the run it fell by 0.99% a day, and each tenth took 2.4 hours: twelve times the pace. Entries were accepted four times as often, 7.3 a day against 1.8. An average over five weeks can hide a fast week, so we also slid a window of the run's length, 138.7 hours, across those five weeks, one start each hour. Of the 735 windows, the median saw a fall of

Table 8: Step sizes: the loop against the other systems. A step is the fall in the score, in per cent of the score before it. The tests are one-sided and treat every choice of the loop's steps among all steps as equally likely.

	The loop	The other systems
Entries accepted	17	25
Share of the fall	70.8%	29.2%
Mean step	0.245%	0.069%
Median step	0.080%	0.030%
Largest step	1.449%	0.495%
Steps of 0.3% or more	4	1
Test		*Result*
Rank sum of the loop's steps, all 138.7 hours		429 of a possible 578; $p = 0.053$
the first 50 hours (six of twenty steps)		$p = 0.004$
up to hour 96.6		$p = 0.040$
Share of the fall, permutation (400,000 draws)		70.8%; $p = 0.020$

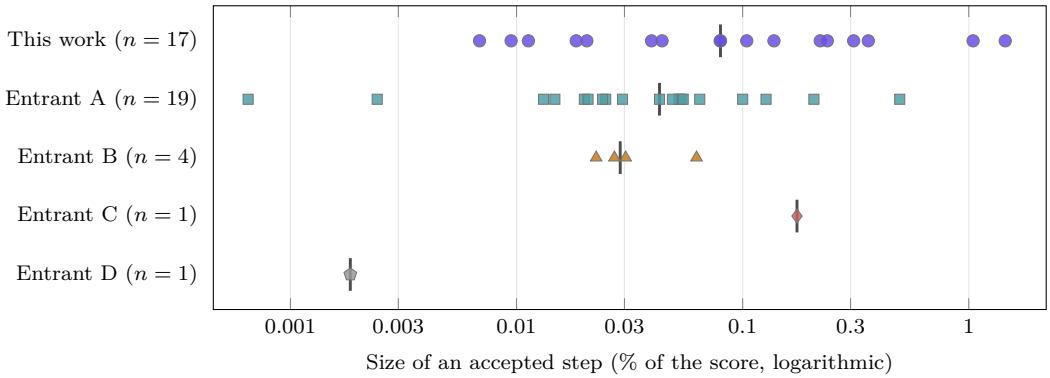


Figure 4: Every accepted step of the run, by size. Each mark is one entry; the bar is the median of its row. Sizes are in per cent of the score at the time, on a logarithmic axis.

0.17% and the best, from 03:00 on 25 September, a fall of 2.40%. The run's fall of 5.73% is 2.4 times that of the best window (Figure 5a).

Where the extra pace came from. Most of it is the loop's own entries. They took 4.06% off the score, 0.70% a day, which alone is more than eight times the whole benchmark's pace in the five weeks before. The other entrants' entries took 1.68%, 0.29% a day: 3.5 times their pace before the run, and less than a third of the run's (Figure 5b). The run was fast for two reasons, then. The loop added a pace of its own, and the others moved faster too. The record shows three ways in which the loop's work and the others' fed each other.

Large steps that became everyone's base. On this benchmark every accepted entry is a public commit with a public note, and the next entrant starts from the best one. A large step therefore moves everyone's starting point at once. The loop made the two largest steps of the run, E4 (1.045%) and E6 (1.449%), the only steps above 1% by any system; together they are 42.6% of the run's fall. After E4, five entries by Entrants A and B each took E4 and changed its settings (Section 3.3). One of Entrant A's entries was rebuilt from a public entry of the loop and its note,

Table 9: The pace of the benchmark before, during and after the run. The first period ends just before the step that brought the new design (9.7%, left out). The run starts seven hours after that step and ends with the loop's last entry, E17. The last period runs from E17 to the last entry accepted before the board was read, at 00:45 on 15 October. A fall is in per cent of the best score at the start of its period. Hours per 0.1% is the length of a period over its fall in tenths of a per cent. The two rows under the run split its fall by who made the entries.

Period	Hours	Entries	Entrants	Fall	Fall a day	Hours per 0.1%	Entries a day
1 Sep. to 7 Oct., the old design	872.8	65	10	3.01%	0.083%	29.0	1.8
The run, 7 to 13 Oct.	**138.7**	**42**	**5**	**5.73%**	**0.99%**	**2.4**	**7.3**
the loop's entries		17	1	4.06%	0.70%	3.4	2.9
the other entrants' entries		25	4	1.68%	0.29%	8.3	4.3
After E17, 13 to 15 Oct.	38.0	5	2	0.10%	0.063%	38.0	3.2

and holds 3.6% of the fall (Table 7). Of the eleven other entries made with the loop's base model, seven started from one of the loop's entries. Part of what the others added in the run was added to the loop's designs.

Quick answers kept the race close. The loop sent each tested improvement on its own, without holding it back to stack with others: after a go at the gate on outside effects in the first 86 hours, and at once after that, under a standing rule (Section 3.1). It also ported the others' best changes onto its design within hours. E9 merged the loop's latest rounds onto Entrant C's entry 102 minutes after it appeared, E13 came 2 hours 26 minutes after Entrant A took the top, and E16 2 hours 52 minutes after Entrant A's next entry; E5 reached the benchmark within an hour of the entry it replaced. The others answered as quickly. After each of the loop's seventeen entries, the next entry by another system came a median of 3.2 hours later, and within six hours after thirteen of the seventeen. The top changed hands 23 times in 139 hours, and the loop took it twelve of those times. The median time between accepted entries fell to 2.8 hours, from 4.1 hours before the run (Figure 5c). Each entry gave the others both a new base and a new score to beat.

When the loop stopped, the pace fell. In the 38 hours after E17, two entrants made five entries. The best score fell by 0.10%, or 0.063% a day: below the pace of the five weeks before the run, and a sixteenth of the run's. Each tenth of a per cent took 38 hours, and the median time between entries rose to 7.9 hours. Entrant A, which made nineteen of the other systems' twenty-five entries in the run, made three in these 38 hours, and its own pace fell from 0.24% a day to 0.03%. E17 held the top for eight hours, until Entrant A's next entry.

What this shows, and what it does not. The run was the fastest stretch of the benchmark in the five weeks before it, and the pace fell back when the loop stopped. That fits the account above: the loop's own large steps, the others' work on top of them, and a race of quick answers. It is not a controlled test, and three things besides the loop differ between the periods. The design: the run began on a design seven hours old, and a young design is easier to improve, so the others' faster pace in the run may owe as much to the design as to the loop. The field: ten entrants had entries in the five weeks before, four besides the loop in the run, and two after it. And the time after E17: the design was then six days older, with its large blocks closer to their floor (Appendix E.6), and 38 hours is a short window. No arm of the run kept the design and the field and took the loop

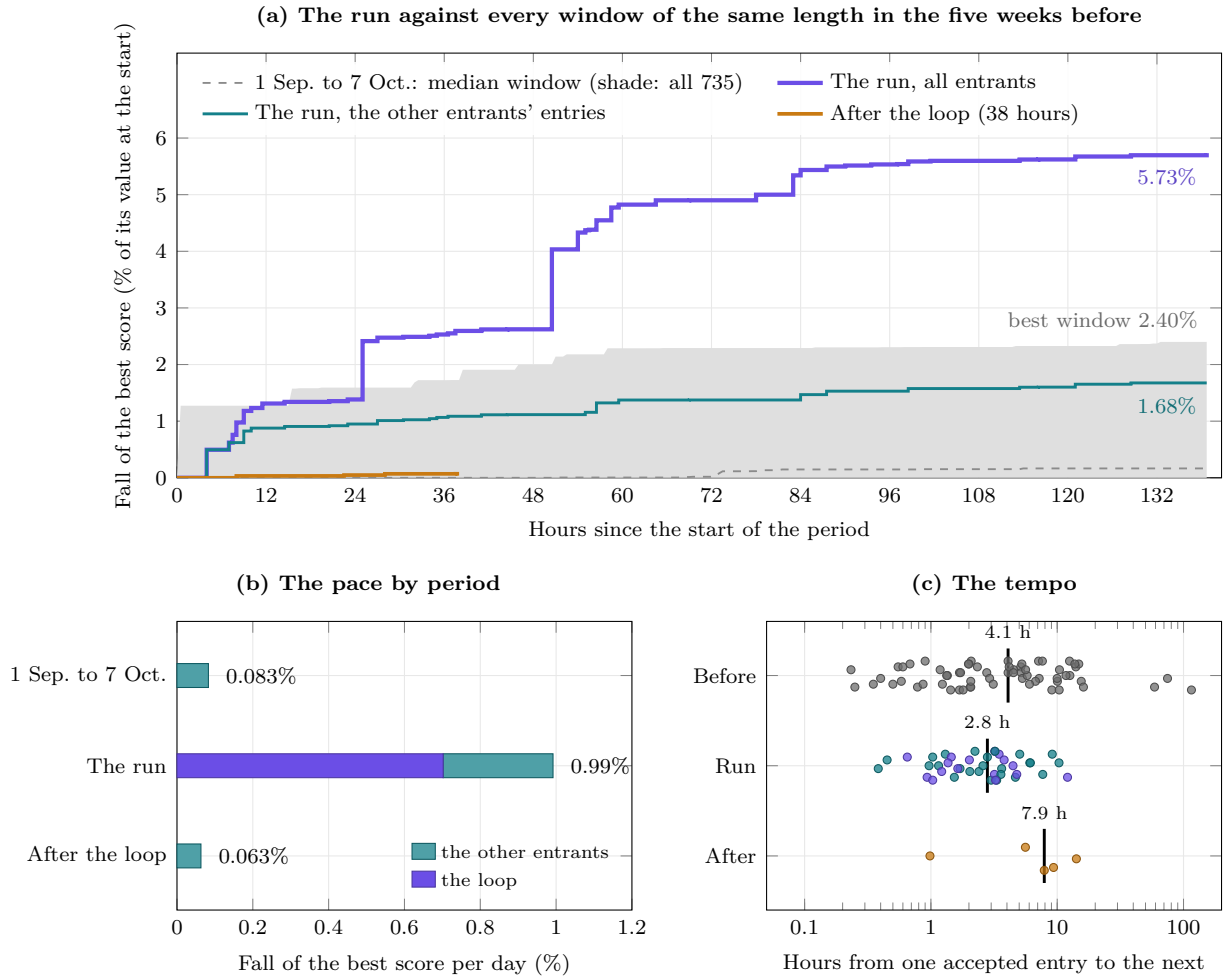


Figure 5: The benchmark ran twelve times as fast while the loop was in it. (a) The fall of the best score through the run, counting every entry (violet) and counting only the other entrants' entries (teal), against every window of the same length in the five weeks before, one starting each hour: the grey line is the median window, the shade the range of all 735. The amber line is the 38 hours after the loop's last entry, on the same design. (b) The fall per day in each period, split by who made the entries. (c) The time from one accepted entry to the next, one mark for each gap, with each period's median.

away. We read Table 9 as the pace of the benchmark with the loop in it and without it, not as a measured effect of the loop.

3.5 The Trials Paid: Stakes and Returns

The topology rests on one claim about research: most of what is tried fails, so a loop should make failing cheap and leave succeeding open (Section 2.9). This section asks whether the run had that shape. We read every row of the run's ledger and kept the 133 trials that sought a gain: 23 prices on paper, 41 pilots, 47 builds, 8 extensions of a change that had just paid and 14 ports. Checks, surveys, searches and tools are left out. Each trial carries its stage, the odds and the size written before it ran, the cost of its workers where the logs hold one, the gain it kept and the points it earned on the benchmark (Figure 6).

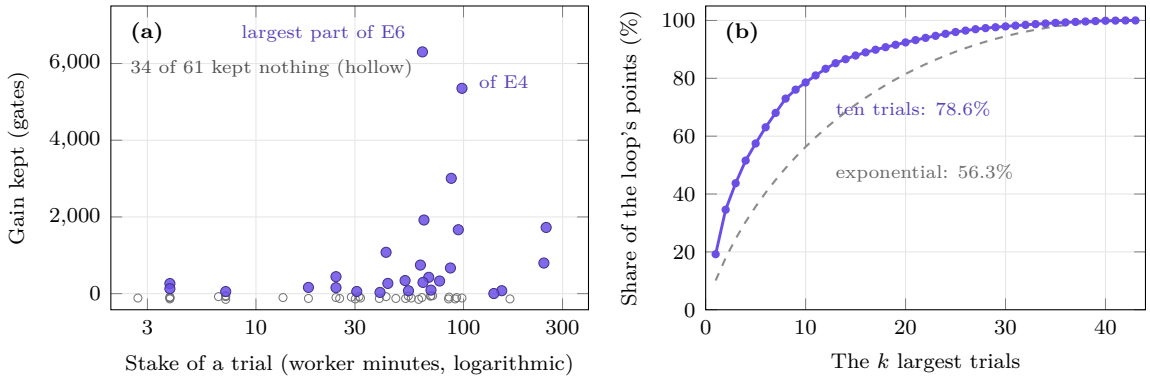


Figure 6: The stake was capped; the gain was not. (a) Each trial's stake in worker minutes against the gain it kept, for the 61 trials whose logs hold a stake. Hollow marks kept nothing. The two highest marks are the largest parts of entries E6 and E4. (b) The share of the loop's points on the benchmark held by its k largest trials, over the 43 trials that earned points. The dashed line is what an exponential spread of returns would give.

Table 10: The 133 trials by stage. A trial was sent on when its price or pilot led to a build. A port carries gains made earlier in the run onto another design, so only gain that was new is counted here. Points are the loop's points on the benchmark (40.5 million in all). The stake is in worker minutes, over the trials whose logs hold all of it.

Stage	Trials	Sent on	Kept a new gain	Earned points	Share of points	Median stake (n)
Paper price	23	3	0	0	0%	25 (8)
Pilot	41	6	0	0	0%	53 (19)
Build	47	–	36	32	71.5%	58 (20)
Extension	8	–	8	7	21.7%	77 (5)
Port	14	–	4	4	6.8%	147 (2)
All	133	9	48	43	100%	50 (54)

The account. The 133 trials earned 40.5 million points: 4.06% of the score and 70.8% of the benchmark's fall. Builds earned 71.5% of them, extensions 21.7% and ports the rest; prices and pilots earned none directly (Table 10, Figure 7a). By technique, three families hold 84% of the points: the first builds of a new design, identities found by hand, and extensions of a change that had just paid (Figure 7b). The median trial earned nothing. The mean trial earned 0.03% of the score, which is the size of the median accepted entry of the other systems (0.030%). Of the worker time that the logs hold for trials, 43% went to trials that kept nothing, and the other 57% produced the whole gain. With the losers counted in, each worker-hour staked on a trial returned 459 gates, about 0.06% of the circuit's gate count.

The stake was small and bounded. A trial stakes the time of its workers: one question, one context, a bound. For the 54 trials whose whole stake the logs hold (61 have a recorded stake in minutes, some of it partial), the median is 50 minutes and 283,000 tokens. Nine in ten cost under 94 minutes, and the largest cost 250, five times the median. No worker used more than 468,000 tokens, 1.6 times the median worker. The stake counts worker time only; the planner's time and the compute of tests are not in it. Each rented search for a passing packet, the dearest rung of the ladder, cost between $0.19 and $1.77.

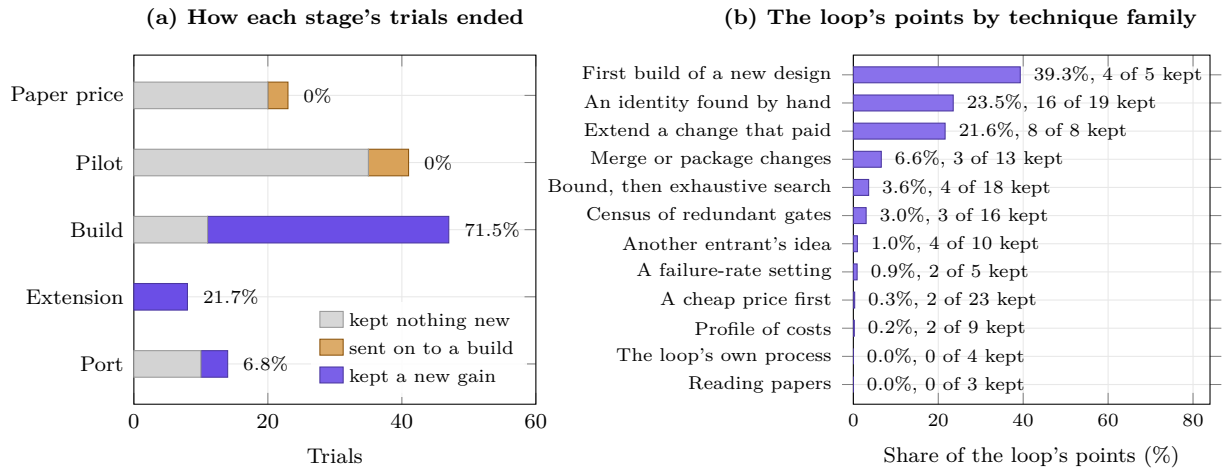


Figure 7: Most trials ended early; a few families earned the points. (a) The 133 trials by stage: kept nothing new, sent on to a build, or kept a new gain. The label is the stage's share of the loop's points on the benchmark. Prices and pilots ended 55 of 64 ideas and earned nothing directly; builds and extensions earned 93% of the points. A port counts only gain that was new. (b) The loop's points by the technique family of each trial, with how many of the family's trials kept a gain.

Most trials kept nothing. 85 of the 133 trials kept no new gain. Of the 64 prices and pilots, 55 ended an idea where it stood and nine sent one on to a build. Of the 47 builds, 36 paid, and so did all eight extensions (Table 10).

Returns were few and large. Ten trials hold 78.6% of the points the loop earned on the benchmark, and the largest holds 19.2%. If returns were spread as an exponential distribution spreads them, the ten largest of 43 would hold 56.3%. Across trials the Gini coefficient is 0.90 for points and 0.44 for stake minutes; most of the first comes from the 90 trials that earned no points, which alone set a floor of 0.68. Entries show the same thing. E4 and E6 are the only steps above 1% that any system made in the run, and the largest step by another system is 0.495%. The two hold 60.2% of the loop's fall and 42.6% of the whole benchmark's. Forty-three values cannot tell one heavy-tailed law from another, so we claim concentration and no law of the tail.

The stake predicted little of the return. The rank correlation between a trial's minutes and the gain it kept is 0.33 on 61 trials ($p = 0.01$), and between its tokens and its gain 0.05: dearer trials kept somewhat more, but a stake explains a ninth of the spread in rank. The largest trial of the run, 6,303 gates or 0.80% of the score, cost one builder 63 minutes, and the idea behind it had been reopened by one worker in forty (Section 3.6).

Trials beat their odds and their forecast sizes. 49 trials carried stated odds, from 0.03 to 0.6. (We write *odds* for a stated probability; a trial *paid* when it kept a gain on the benchmark's measure; the event its odds were written for, often that its cheapest test would not end it, is not always the same thing.) Of the 48 whose event is known, the event came true in 46%, against mean odds of 0.30, and more often than its odds in both bands of Table 11 (Figure 8a). Sizes ran low as well. Of 26 trials with a forecast of size and a result, 19 beat the forecast, and the results sum to 1.6 times the forecasts (Figure 8b). The planner's error fell on the side that the stake does not cap.

Where the gain came from. The gain came from trials that the planner gave about even odds. Trials given 0.35 to 0.6 paid 20 times in 23 and kept 921 gates for each hour of recorded worker time; trials given under 0.35 paid 4 times in 26 and kept 20 (Table 11). Counting each pilot with

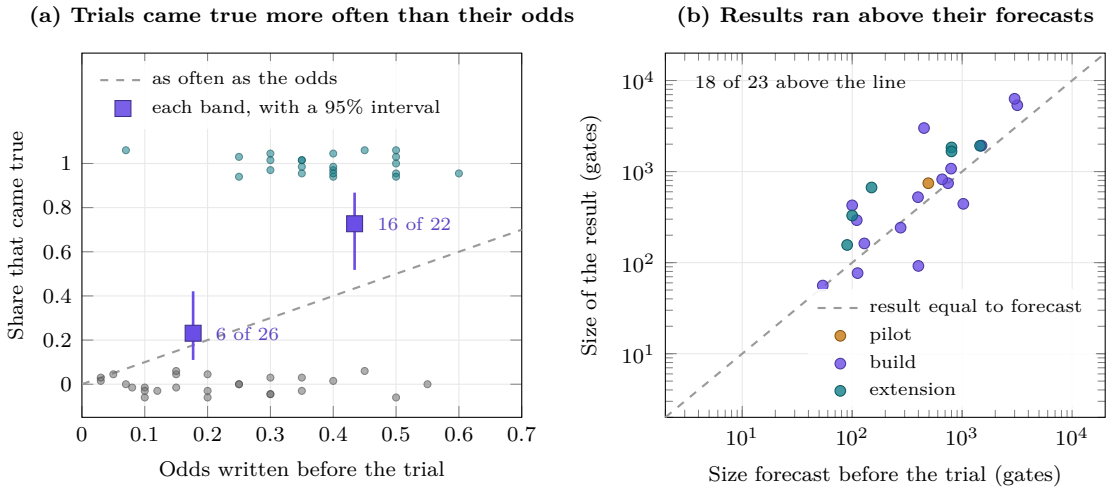


Figure 8: Trials came true more often, and came out larger, than forecast. (a) Each of the 48 trials with stated odds and a known event, at the odds written before it ran: teal marks came true, grey did not (spread a little in height so that they can be seen). The squares are the two bands of Table 11, at their mean odds, with 95% Wilson intervals. Both lie above the dashed line, on which events come true as often as their odds say. (b) The size forecast for a trial against the size of its result, in gates, for the 23 of the 26 trials whose forecast and result are both above zero, as a logarithmic axis needs. Left out are two prices whose result was zero and a build forecast to lose 333 gates that saved 74. Of the 23, 18 lie above the line of a result equal to its forecast.

Table 11: The 49 trials with stated odds, by band. The event is the one the odds were written for. Gates kept a worker-hour are over the trials whose logs hold a stake; in the last column each pilot is counted together with the build it led to. No trial was given odds above 0.6.

Stated odds	Trials	Mean odds	Event came true	Paid	Gates kept a worker-hour	
					Trials	Chains
Under 0.35	26	0.18	6 of 26 (23%)	4	20	363
0.35 to 0.6	23	0.43	16 of 22 (73%)	20	921	682
All	49	0.30	22 of 48 (46%)	24	548	

the build it led to narrows the gap to 682 against 363 but does not close it. None of the ten largest trials had been given less than 0.4. So long shots did not earn more than even chances. The asymmetry that paid in this run is in the payoff, a capped loss against an open gain, and not in a taste for unlikely ideas. Low odds mattered at the head of a chain. The two pilots that reopened the idea behind E6 were given 0.30. And at hour 72 the record by technique rated the pilots behind the largest parts of E4 and E6 among its 30 least likely trials of 90, of which the other 27 did not pay (Appendix E.5). A rule that cut on odds alone would have cut both.

What this shows, and what it does not. The record has the shape the design asks for: a stake that is bounded and alike across trials, a majority of trials that keep nothing, a return concentrated in a few, and no way to tell the large ones by their cost. It does not show that ranking by expected gain per unit of stake caused the result. That needs a second arm of the same loop that ranks by odds, on the same task and budget, and we have not run one (Section 5). Two limits lie in the data. The logs hold a whole stake for 54 of the 133 trials, because they begin at hour 6 and a trial

run by the planner has no worker. And the table of trials was read from the ledger by hand, by an agent, with its conventions written beside it.

3.6 Case Study: One Idea, From Parked to the Largest Step

E6 is the largest step of the period and used nearly every rule of Section 2. We follow it in order.

The idea. In most iterations the main loop adds three values to a register and reduces the register after each addition. The idea is to reduce once, after all three.

Parked on a price. A census of the circuit's cost first raised the idea and priced it on paper. The gain looked like 0.2% to 0.4%. Against it stood about 45 more wires inside the reduction, at a measured price of 155 gates a wire, and an assumed 60 gates to erase three leftover wires. At those prices the idea did not pay, and it was parked. Under the closure rule it kept a reopening condition: *reopens if the three leftover wires can be erased for under* 20 *gates.*

Reopened by its number. In the next round the planner sent that one question to a worker, with its own forecast that no such erase existed. The worker forecast that one did exist if six intermediate values were kept, checked the identity on 4.4 million inputs, and found an exact erase for 0 gates. The idea was open again. The worker's report also corrected six points of its brief and questioned the wire price.

The wrong price. A second worker counted the new reduction on a gate-level model. It needs 17 and 26 more wires in the two directions, not 45 (24 and 36 at its peak, all told). More important, the price of 155 gates a wire turned out to belong to one place in the existing design: two cells where a reduction sits inside an addition. The new operation has no such cell. In the place where the new wires are needed, a wire costs 0.5 gates a pass. The paper gain rose to between 0.79% and 0.96%. Both clauses of the planner's falsifier came out exactly on their limits (36 wires against "more than 36", 5 against "more than 5") and did not fire. The loop did not debate the old falsifier; it wrote a new one in the next rung's own unit.

Built one rung at a time. Table 31 in Appendix F lists every stage with both forecasts and the outcome. One iteration was built first, with a falsifier of 25 gates saved a pass; it saved 49.0. The whole first build, in both directions of the loop, saved 6,303 gates (0.80%) at the same number of qubits and the same failure rate ($+0.12 \pm 0.15$ failing shots on 320 shared packets).

Follow-ups chosen from the new design's prices. As soon as the first build held, the planner re-measured the price map on it and sent two workers in parallel, each on an isolated copy. One looked for exact savings in the fixed logic of the new reduction. It cut the core from 29 logical ANDs to 22 and 23 by algebra and removed a recomputation: 3,007 gates. The other fitted the new operation to iterations with less working room: 1,841 gates. A third worker merged the two. Before anything was switched on, the merged source had to rebuild three known answers byte for byte: each parent and the base. The merge fell 77 gates short of the plain sum, because the two studies had used alternative means on some passes.

Safety bought back. In the package one setting had been moved a step to the safe side, at a cost of 268 gates. Compared with the incumbent on 240 of the incumbent's own packets, the package failed less often, by 0.59 ± 0.16 failing shots. The room freed by the smaller core then let twelve more passes take the new operation (611 gates). Figure 9 shows the whole path.

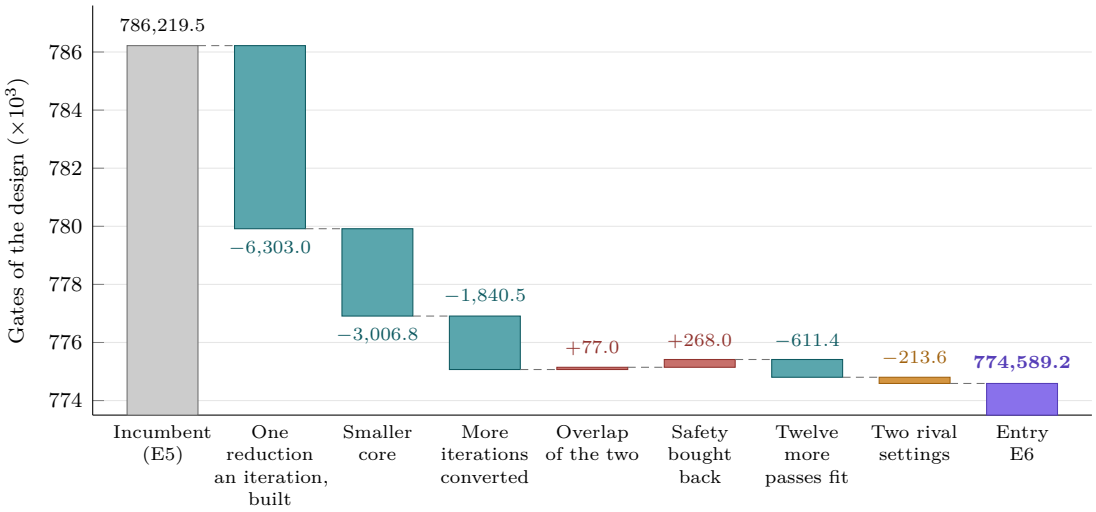


Figure 9: The path of E6. The gate count of the design through its stages, at an unchanged number of qubits. Teal bars are gates saved by a stage; red bars are gates given back; the amber bar is two settings taken from rival entries that were accepted while E6 was being built. The vertical axis starts at 773,500.

Search and acceptance. While the package was being built and while it waited for a search, two rival entries were accepted. Each was a setting on top of E5, and each was taken into the candidate in one line (214 gates together). After the second, the candidate was shown to give the same outputs as before on 1.8 million shots. The search was planned from e^{λ} at 158 million tries and found a passing packet after 53 million. The official judge accepted the entry at 957,294,360, a step of 1.449%, with 0.34 ± 0.16 fewer failing shots than the entry it replaced. From the reopening question to acceptance took under thirteen hours.

What the case shows.

- **The registry paid.** The largest step of the run came from the list of parked ideas, not from a new one. Without a registry the idea would have been a line in an old report.
- **A reopening number is a small question.** "Under 20 gates" could be answered by one worker in forty minutes. "Is this idea any good?" could not.
- **The planner was wrong, cheaply.** It forecast that no erase existed, that the gain was unlikely to reach 0.5% (a chance of 30%), and that the two follow-ups would save 450 and 800 gates where they saved 3,007 and 1,841. Each error was found on a rung that cost minutes.
- **The workers were wrong on the same side.** In four of the five build stages both forecasts were below the outcome. Two forecasts that agree are not thereby right (Proposition 3, second part).
- **A price belongs to a place.** The idea had been parked on a real, measured price that was applied to a design it had not been measured on. The planner had made that mistake twice. It was found when a worker traced what the price was made of.
- **No rung was skipped.** Identity, model, one iteration, all iterations, paired test, search, judge. The first build worked at the first try; the two rungs before it had already removed the errors a build would have found slowly.

3.7 Three Cases From the Full Topology

Three entries after hour 75 show the roles of Table 2 at work. Table 12 sets each forecast written before a step of these cases against what happened.

E11: an empty list of cards to an accepted entry in 2 hours 41 minutes. The first round run through every role of Table 2 was set up as a test of the topology against a day of rounds run by the planner's own lanes, which had kept nothing. The forecast and the falsifier were written before any agent ran: a verified gain of 18 gates or more with a chance of 35%, and of 150 or more with 12%.

Four scouts wrote 22 cards. Five pricers ended nine of them for the price of 44 builds of the circuit; a whole lens, the eight cards that proposed new settings, was worth 3.5 gates in all. Both cards of size came from the lens that measures where the cost sits. The larger form was found while the smaller was being priced: a pricer who was modelling a saving of 305 gates in one block noted that the block copies a constant into a register only because its three additions are written out, and a second pricer modelled the block with nothing written. The build came in at 797.8 gates, 0.104% of the score, and above every forecast made for it.

The breaker rebuilt the change from its description to the same bytes and then, instead of repeating the builder's test, built a small circuit that holds the new block beside the old one and drove it with 360,960 inputs aimed at the block's edges. On 330 packets of the incumbent's own inputs six outputs differed. In all six the incumbent was wrong and the new block right: the incumbent's block had an error on one to three shots in 100,000 that no entrant had seen. A claim of "exact" would have been false in the loop's favour, and the note of the entry says what the breaker found.

One part of the round was dropped. An external yes-or-no decision model had been placed after the scouts as a gate on closed directions. It flagged five cards, and the scouts had already cited the closure on all five. With the closed list in every scout's reading, a gate of that kind has no work left.

E15: an entry held at a gate, and the price of keeping a promise. The score is a product, and from E5 to E14 every entry had moved the gate count only. The limit on qubits had been closed twice, each time on a plain build. After E13 the instruments priced it again with the settings that depend on working room refitted (Appendix C.2). The first qubit, forecast at 700 to 1,200 gates with a chance of 7% that any other limit would win, cost 543 against the 617 that breaks even. The forecast had priced the wire; most of the plain cost was two settings sitting on cliffs that had been fitted to the old limit.

A new limit moves the plan of almost every stage, so it shifts the indices of the verifier's random numbers everywhere and no aligned pair exists (Appendix C.1). On the same inputs the candidate and the incumbent gave the same outputs on 2.17 million shots, and the paired difference read -0.14 ± 0.13 failing shots a packet against a control at -0.10 ± 0.11. By the test, nothing had changed.

The builder's list by structure said otherwise. Twelve classes of approximate element were equal in the two designs. In one more the candidate had 22 more approximate comparisons than the incumbent, 1.97% more summed miss rate, about $+0.064$ failing shots a packet. The control's standard error of 0.11 on 240 packets gives $\kappa = 2.9$ and $\theta = 0.08$ in Proposition 7; to resolve 0.064 at two standard errors the run would have needed about 2,800 packets. The candidate was not sent.

Table 12: Forecasts against outcomes in the three cases. Every forecast was written before its step ran.

Entry	Quantity	Forecast	Outcome
E11	Verified gain of the round	18 gates or more at 35%; 150 or more at 12%	797.8 gates (0.104%)
E15	Gates paid to remove the first qubit	700 to 1,200	543, against 617 at break-even
	Tries to a passing packet	120 million	124 million
	Score	942,266,715	942,266,715
E17	Gates returned by the freed wire	60 to 365, floor 61; the builder's median 110, with 28% to 40% for 100 or more	292.7
	An aimed class leaning three standard errors against the change	12% (planner)	Found in 24 minutes

An offset was built: other comparisons of the same kind were made two bits wider at one gate each. The first offset matched the summed miss rate and, on an aligned pair of 120 packets, bought back one failing shot where the arithmetic owed nine. Its comparisons sat late in the computation, where they almost never meet the input on which they would miss. Miss rate is comparable only inside a class. The second offset widened, for each added comparison, two of the same strength in the same place; it bought back two failing shots where three were owed, and the candidate was then at or below the incumbent in every class.

The offsets cost 53 of the 106 gates that the new limit had gained once its settings were refitted (the plain build had gained 74). Half the gain was spent to keep a promise that no test of practical size could have checked: that an entry never raises the failure rate above the entry it replaces. One entry earlier a change had been sent as exact that is not, and no paired test had shown it (Section 3.9). The search forecast for E15 was written before the search and held on every line (120 million tries forecast, 124 million taken; a score of 942,266,715 forecast and obtained).

E17: a wire that only held a copy, a price five times too low, and a verdict read twice. For 26 hours before E17 the top was held by Entrants A and D, over four entries. A desk worker listed every measured, unsent change in the loop's record against the newest design: apart from one, they summed to about four gates. The one came from an instrument put to the finished gate list and not to its source: which wires, through the operation that sets the peak, hold nothing but a copy of another wire? The top wire of one register did, in 127 iterations of two stages. Undoing the copy before the operation and making it again after costs at most two gates of a kind the objective does not count, and changes no output and no flag (1.08 million shots).

The freed wire was priced at 60 to 365 gates with a floor of 61; the builder's own median was 110, with a chance of 28% to 40% for 100 or more. It returned 292.7. Both forecasts had priced the wire from its cheapest known use. But the plan of every iteration is fitted against the qubit limit on that iteration, so every one of the 127 iterations became cheaper, by 0.5 to 10 gates each. The stake was one builder for about an hour and three quarters.

Entrant A's design bound 334 gates of edits to a hash of its whole gate list, so any change upstream stopped its build; the builder replayed the edits on the base until they reproduced its gate list exactly, and carried 27,533 of the 27,537 through as one table.

The spent iterations have another plan, so measurements move and no aligned pair exists (Appendix C.1). On 2.17 million shared random shots no output differed and failing shots read 18.550 against 18.558 a packet. The task to the breaker said, before it ran, that a class of aimed inputs leaning three standard errors against the candidate would count as broken; the planner gave that 12%. The breaker found one in 24 minutes: with a run of 22 or 24 equal bits ending where the new plans cut their additions, shots that pass in the base fail in the candidate far more often than the other way. It also found the mirror, at the places where only the base's plans cut (Table 13). By the written criterion each design is broken against the other. The planner then asked for the measurement the criterion should have named, a run of equal bits at every position with nothing picked; on it the candidate came out ahead by 232 ± 72 shots. The candidate is the weaker design at 10 positions and the base at 16; a count of approximate comparisons agrees (1,919 against 1,934). No output differed on any aimed shot. The entry was sent with the class, its mirror and the scan in its public note. The criterion in Appendix C.1 now reads: for a change that re-fits a plan, the net over inputs nobody picked decides, and one-sided classes are reported beside it.

3.8 Evidence for the Mechanisms

The run has no ablation: no part of the topology was removed. In its place each part is tested against the record, and every number is computed by a script from the ledger and the journal of the run and from the benchmark's public history. Table 14 lists the tests, and Appendix E gives each in full.

What held. The price map found a discrete optimum on the qubit cap and a price list on which the best knob paid about twice the next, and it predicted the gates of a package to within 4.2%. Pairing cut the variance of a comparison by factors of 3.7 to 270, and its standard error by 1.9 to 16. The record by technique was a better prior than a coin, and at least as good as an external decision model that was shown the same record. On the one day when every source of cards ran at once on one design, 300.7 of the gates that reached the benchmark came from instruments and ports and 56.0 from 35 written ideas. Every build report of the rounds behind E4 and E6 corrected its brief, with three to ten items each. Four falsifiers fired in the first fifty hours, and each ended its line within the hour. The red team attacked eleven closures; every conclusion held and two of the planner's reasons were wrong.

What did not. Neither role forecast better than the other: the worker was closer in 7 of 13 matched pairs. An earlier version of this analysis, on seven pairs, found the worker closer in six, and we withdrew the claim. The two roles erred on the same side in 10 of 13, so the gap between two forecasts flags a disagreement and not a blind spot that both share. The planner priced costs high, with outcomes at 0.77 of its forecasts, and its stated chances scored worse than a coin because it expected too little; the idea behind E6 was parked on such a cost. Failing shots, unlike gates, did not add up across a package, so prices choose a package and the package is then measured whole. Four closures were later overturned by a build. All four had rested on a paper price or a model, and none that rested on a proof or an exhaustive search has been overturned. And trials given low odds did not earn more for each worker-hour than trials given even odds (Section 3.5).

Table 13: What random and aimed inputs saw. Counts are of shots on which the two designs disagree about passing. Aimed inputs are derived by a breaker from the structure of the change; random inputs are the verifier's own.

Change	Inputs	Shots	Base passes, change fails	Change passes, base fails
E14	Random, aligned pair of 240 packets	2.17 million	0	–
	Random, drawn by the breaker	1.08 million	0	0
	Aimed at its structure	81,216	10,718	4,082
A second change of E14's form	Aimed at the edge of a 56-bit window	–	965	1
E17	Random, shared	2.17 million	no output differed; 18.550 and 18.558 failing shots a packet	
	Aimed: a run of 22 or 24 equal bits ending at the new plans' cuts	–	383	8
	Aimed, the mirror: at the base's cuts	–	33	210
	A run of equal bits at every position, nothing picked	162,120	2,503	2,735

3.9 Negative Results

Six things did not work, or worked less well than claimed. Table 15 lists them and the paragraphs below give each.

A hunt for one large step. After E9 the aim became a third step of 1%. Three lanes looked for it. The closed list was searched for ideas that would be worth 0.5% if their dearest untested part came cheap: three could be tested and all three failed on a part their paper price had not named. Thirty-eight concepts from eight other fields were mapped onto the cost map and six were tested: none survived at 0.5%. And the design's floor was derived again on six axes: three prizes of the wanted size exist on paper and none has a way to collect it. The aim was then changed to steady gains by the method, and the next round produced E12, the largest step of the second half.

Reading papers for whole blocks. Six rows of the record read published work for a block to replace, and none paid. When each literature scout was instead given one technique that had already paid in the run and asked where else it applies, one of about sixteen cards was a placement the loop had not made, worth 745 gates. The paper gave a sentence; the loop's own trace gave the place [13].

Two agents in dialogue. Two agents on two models, each with the benchmark's full history, wrote ideas alone and then in seven alternating turns: 35 ideas, one built, and by their own last check none above 0.4% left open.

Designs from a blank sheet. When the ledger showed the large blocks at their floor, four designers started elsewhere: a division with no record of its steps; fewer divisions; tables and large jumps; ideas from other fields. All four designs lost, at 1.49 to 4.04 times the accepted score. Before

Table 14: The tests of Section 3 at a glance. Appendix E gives each test in full.

Claim	Test	Result
The qubit cap is a discrete optimum (Cor. 1)	$g^{+} < T/(Q{+}1)$ and $c^{-} > T/(Q{-}1)$	$419.2 < 635.7$ and $748.5 > 636.7$
Knobs sell one resource at different prices (Prop. 2)	ten settings, each paired with a reference on 160 packets	133 to 994 gates per failing shot; the best pays twice the next
Prices predict a package (first order)	sum of four measured trades against the package measured whole	gates -371 against -356 (4.2%); failing shots -0.75 against -0.16 ± 0.18 (not additive)
Pairing shrinks the variance (Prop. 5)	paired standard error against $\sqrt{2\lambda/n}$	0.18 against 0.35 at $n = 320$ (3.7×); 0.16 against 0.40 at $n = 240$ (5.9×); down to 0.03 against 0.49 at $n = 160$ (270×)
A one-sided count misprices a repair	fixed and newly broken failures counted apart	$a = 0.830$, $b = 0.284$: 529 gates per shot, not 328
Failing shots are Poisson	variance over mean	0.997 ± 0.010 on 21,062 packets
The planner's forecasts of costs are biased	mean log error, $n = 24$, bootstrap	-0.26, interval $[-0.42, -0.11]$
Workers' forecasts are biased	mean log error, $n = 18$, bootstrap	$+0.29$, interval $[+0.14, +0.48]$
One role forecasts better than the other	matched pairs, sign test	worker closer in 7 of 13, $p = 0.50$: not supported
The two roles err together (Prop. 3)	matched pairs on the same side of the outcome	10 of 13
Misalignment hides a real difference (Prop. 7)	control pair, then the list by structure	$\kappa = 2.9$ at $n = 240$; a rise of 0.064 by structure needs about 2,800 packets to resolve
The record by technique is a usable prior	Brier score on 90 scored trials	0.149 by stage and 0.201 by family, against 0.250 for a coin
Instruments pay more than idea lists	gates accepted by source of the card, one day	162.5, 85.0 and a net 53.2 from instruments and ports; 56.0 from 35 ideas
The loop's steps are larger than the concurrent systems'	exact rank test, 17 of 42 steps	rank sum 429, $p = 0.053$ ($p = 0.004$ on the first 50 hours, 0.040 at hour 96.6)
The loop's share of the fall	permutation of 17 of 42 steps	70.8%, $p = 0.020$
Returns are concentrated in few trials	share of the loop's points in its ten largest trials of 133	78.6%, against 56.3% for an exponential spread; two entries hold 60.2% of the loop's fall
The mean trial pays though the median does not	points earned per trial, 133 trials	median 0; mean 0.03% of the score, the other systems' median entry
Trials beat their stated odds	outcome rate against mean stated odds, 48 trials	46% against 30%; 73% against 43% for odds of 0.35 to 0.6
The stake does not predict the return	rank correlation with the gain kept, 61 trials	0.33 for minutes, 0.05 for tokens
Trials given low odds earn more per stake	gates kept per worker-hour, odds under 0.35 against 0.35 or more	20 against 921; 363 against 682 with pilots counted in their builds: not supported
Builds fail less often under the full topology	builds that kept nothing, before and after hour 75; the earlier period split in halves	10 of 35 before, 1 of 12 after ($p = 0.24$, two-sided); 5 of 17 and 5 of 18 inside the earlier period: a step, not a trend
A rule on odds alone would cut the pilots of the large steps	the 30 of 90 trials the record rated least likely at hour 72	27 did not pay; the other 3 are the pilots behind the largest parts of E4 and E6

counting, three of the four designers had given odds of 10% to 12% that their design would come within 1.3 times; after counting, 3%. The round kept three things that are not designs: a proof that two divisions are needed in a model of field operations and stepwise division; the reason a design without a step record gains nothing, which is that the record and the working registers share one

Table 15: What did not work.

Attempt	What was tried	What came of it
A hunt for one large step	After E9, three lanes: closed ideas reopened on their dearest part, 38 concepts from eight other fields, the design's floor derived again on six axes	Nothing at 0.5%; the aim was changed to steady gains, and the next round produced E12
Reading papers for whole blocks	Six rows of the record read published work for a block to replace	None paid; a scout holding one paid technique found one placement worth 745 gates
Two agents in dialogue	Two agents on two models, seven turns	35 ideas, one built (56 gates); none above 0.4% left open
Designs from a blank sheet	Four designers, each starting from a different place	All four lost, at 1.49 to 4.04 times the accepted score; a proof, a cause and a floor kept
A gate by opinion	An external decision model in three more seats	A useful pointer to closed directions (10 of 12); as a judge, no better than the record
A claim of exactness	E14, cleared by an aligned pair of 240 packets	Not exact on inputs aimed at its structure; a fourth comparison added

pool of wires whose safety margins add; and a floor of two gates a column for the addition that costs three today.

A gate by opinion. Tried in three more seats, the external decision model of Appendix E.5 found the closed direction that a reworded proposal touches in ten of twelve cases, which is useful as a pointer; as a judge of whether a trial will pay it was no better than the record alone.

A claim of exactness that was false. E14 wrote the first iteration of one stage as one shift and one addition where three additions stood. It was cleared by an aligned pair of 240 packets, 2.17 million shots: no output differed, and no shot passed in the base and failed in the new design. Six hours later, after the cut-off of this paper's counts, a second change of the same form went to a breaker with the instruction to aim at the structure and not to repeat the builder's test. The breaker derived the inputs on which the shorter form drops a carry at the edge of a 56-bit window and drew inputs there: 965 shots passed in the base and failed in the new design, against one the other way. That change was not sent. A breaker was then aimed at E14 itself, with the result in Table 13: thousands of aimed shots separate it from its base, and a million random shots do not. By the breaker's model, inputs on which only E14 errs come up about 2.4 times in 10^9 random shots and inputs on which only the older form errs about 46 times; the model is not a measurement. So E14 is not exact, no paired run of practical size could have shown it, and whether it raised the failure rate is not known. Appendix C.1 now carries a fourth comparison. The entry's public note cannot be amended.

4 Related Work

Appendix A sets each mechanism of the topology against the closest published work, row by row. Here we place the work in six lines of research.

Self-referential and self-improving agents. The Gödel machine rewrites itself only after a proof that the rewrite raises its utility [44]. Current systems keep the self-reference and replace the proof with empirical validation. STOP runs a seed improver on itself under a frozen model [75]; Gödel Agent patches its own running code [71]; Promptbreeder evolves the prompts that mutate its task prompts [10]; in ADAS a meta agent programs new agents into an archive [16]. The Darwin Gödel Machine, SICA, the Huxley-Gödel Machine and Group-Evolving Agents grow archives or trees of coding agents that edit their own code, and differ in how a parent is selected and in what is shared [40, 58, 64, 76]. Hyperagents makes the meta agent editable as well [78], and AIDE^2 nests a research agent inside an outer loop that rewrites its harness and keeps a rewrite on a private held-out score [53]. A recent survey grades such loops by which decision has passed from the human to the system [7]. In all of these the thing improved is an agent and the feedback is a benchmark score. Priced falsification improves a design and not an agent, its self-modification is limited to rules and instruments, and it adds a kind of feedback we did not find in these systems: a score on the meta level's own forecasts.

Cost-aware search. Choosing what to try by expected gain per unit of cost is classical. Weitzman's index settles the search among options that each cost something to open [61], and the order of tests that least costs to reach a verdict goes back to Mitten [35] and Simon and Kadane [49]. Bayesian optimisation weighs expected improvement per unit of cost [51], and Hyperband spreads a budget across configurations by stopping the poor ones early [27]. For language-model agents, CostAda values a discovery loop's progress relative to the realised cost of each action [83], and Calibrate-Then-Act gives an agent what it needs to weigh the cost of exploring against what exploring would tell it [6]. Propositions 4 and 6 are instances of these results. What this paper adds is where their inputs come from: a price map measured on the design, and a planner whose forecasts are scored.

Discovery systems with an automatic evaluator. FunSearch and AlphaEvolve pair mutation by language models with an evaluator and a database of programs [36, 41]; AlphaEvolve adds an evaluation cascade that prunes with cheap tests first. Eureka evolves reward functions with reflection on training statistics [33], and the AI Scientist chains ideas, experiments, a write-up and an automated reviewer [31]. Their condition is ours: candidates that can be scored automatically. Our verifier ladder is a cascade that the loop builds itself and admits on a known answer (Section 2.8), and our record keeps closed and parked directions where these systems keep candidates.

Self-improvement through weights and context. A third line improves the model or its context instead of a scaffold. The model is fine-tuned on its own rationales that reached the right answer [74], on answers it has judged itself [73], on tasks it proposes and an executor checks [84], or on training data it writes for itself [87]. Without a change of weights, an agent keeps verbal reflections [47], a library of verified skills [57] or an itemised playbook [77]. We use no weight method. The technique cards of our record are nearest to the last three.

Limits and risks of self-improvement. Models correct their own reasoning poorly without external feedback [17]. The gain from a loop of generating, verifying and updating is bounded by the gap between generating and verifying [52]. Models trained on their predecessors' output lose

the tails of the distribution [48]. Self-evolving agents can drift into harm along each path by which they evolve [45], agents on research benchmarks have been observed to game the score [39, 53], and RE-Bench sets agents against human experts under equal conditions [65]. These results shaped the design. The planner's errors are found by another context that holds a measurement, a claim is attacked before it is kept, and no proxy stands in for the exact verifier at the last gate. They also describe what went wrong here: one accepted entry passed the evaluator's random inputs and is not exact (Section 3.9).

Evidence-governed research loops. The closest systems are loops that govern evidence around a proposer: a prediction and a falsifier before a run [15, 50, 79], an append-only ledger [15], agents rebuilt from a record and not from their own history [9, 80], and records of failed routes with conditions for reopening [59, 86]. We adopt all of these. What we did not find in them, or in the lines above, is the combination this paper is about: two forecasts on one experiment scored by role, required errata on the task, a red team on the closed list, and a price map as the object the planner plans from.

5 Discussion and Limitations

What the six days suggest. The first two days and the last four are different experiments. In the first, a young design gave two steps above 1%, and both came from the rules about the planner: a rebuilt planner that changed one rule, and a parked idea that a worker reopened against the planner's forecast. In the last four, the design was older and its large blocks were near their floor; the loop kept its share of the progress, but through many small exact steps found by instruments, and its statistical lead over the other systems shrank. The technique found the large steps while they were there to find and did not invent one when they were not. Its later days are better read as a measurement of the floor (Appendix E.6) than as a search that failed.

What the roles add. The topology was adopted at hour 75 and we have no controlled comparison of the loop with and without it. The record gives three observations. Its first round reached an accepted entry in under three hours from an empty list. Its gates stopped four changes that would have raised the failure rate, three of which an ordinary test had passed or could not see. They also passed one change as exact that is not (E14, Section 3.9); the comparison that caught it was added afterwards. And a breaker that saw only the claim found an error in the incumbent that its own authors and three other entrants had not.

Is this self-improvement? In a limited sense, and not a recursive one. The loop revised its own procedure during the run: after each of 67 logged mistakes of its planner, with instruments it built for its own ladder, with three rules taken from the literature after a stall, and with an arrangement of roles found by its own tests (Section 3.2). Whether the revisions made it better at its method the record cannot say. Builds that kept nothing, the dear failure that pricing exists to prevent, fell from 10 of 35 before the full topology to 1 of 12 after it (Fisher's exact test, $p = 0.24$), with no trend inside the earlier period (5 of 17, then 5 of 18): a step that came with one change, not a loop that improved as its record grew. Yield did not rise either: the last third of the trials earned 6.1 million points against 29.6 million in the second, and a curve cannot tell a better loop from an easier stretch of the problem. The test that would settle the question is the one the literature on recursive self-improvement asks for [7, 53]: successive versions of the loop on a fixed task under

matched budgets, scored on a held-out signal. We have not run it. The system is an autonomous research loop whose procedure was revised during the run; the model is frozen.

Safety and oversight. The loop modifies only its own scaffold. It cannot reach the foundation model's weights or edit the benchmark's evaluator, and candidate circuits run only in the benchmark's sandboxed harness. Actions with an effect outside the loop waited for the human operator for the first 86 hours; after that, entries that passed the loop's tests went out under a standing rule, and rentals of compute stayed under the operator's approval and bounds (Section 3.1). One failure of the kind this literature warns about occurred, in a mild form: the evaluator's random inputs are a proxy for correctness, and the loop sent one entry that satisfies the proxy and is not exact; a breaker of the same loop found the gap afterwards (Section 3.9). One decision under the standing rule deserves a line. A breaker separated E17 from its base on aimed inputs in both directions and wrote that the choice belonged to the operator. The planner read the net over unpicked inputs as deciding, said it would send the entry unless told otherwise, had no answer in three and a half hours, and sent it with the breaker's findings in the note. A stricter rule would have waited. We did not audit the run for attempts to alter a scorer or to act outside a worker's copy of the work. A stronger loop of this kind will need the evaluator, the approval gate and the record kept out of its reach by construction, not by convention.

Limitations.

- **No controlled comparison.** There is no run of the same model without the technique on the same starting point, and no ablation of any rule or role. In particular there is no baseline with the loop's revision of its own rules switched off. The tables attribute outcomes from the record; they do not measure each rule's effect. The comparison with the other entrants is observational: the systems differ in operator, compute and hours.
- **No evidence of recursive improvement.** The loop's revisions of its own rules were not run against their predecessors under a matched budget, and some followed the operator's directions. The one change the record shows, fewer failed builds after hour 75, is a single step at the change of topology, not a trend. The run shows a loop that revises its scaffold, not one that gets better at improving.
- **One harness, one benchmark, one base model, six days.** There is no held-out task and no transfer to a second problem, model or harness; that another harness with the capabilities of Appendix B can run the technique is a claim about design. The task has an exact public verifier and accepts any strict improvement. The technique is untested where the verifier is slow, noisy or a human reviewer.
- **A young design, and a field that changed.** The run began seven hours after a new circuit design was posted. Improvement was easier for every system in that window than in the weeks before, and it grew harder for every system over the six days. The benchmark's twelvefold pace during the run, and its fall when the loop stopped, are set against periods with a different design or a different set of entrants (Section 3.4); they are not a measured effect of the loop.
- **Every entry stands on the last.** Each entry on this benchmark starts from the best public one, the loop's included. E6 carries two rival settings worth 214 gates, E9 and E13 are the loop's changes ported onto another entrant's design, and several entries by others are one of the loop's entries with settings changed or with work added. Shares of the fall are shares of steps, not of independent work.

- **Small samples, and a weaker test over the longer run.** Forty-two entries, 57 numeric forecasts and thirteen matched pairs. The rank test gives $p = 0.004$ on the first 50 hours, $p = 0.040$ at hour 96.6 and $p = 0.053$ on all 138.7. The forecast set covers the first fifty hours only and was extracted from free-text ledger lines by the loop itself; the split of forecasts by kind was made after the data were seen. One conclusion of an earlier version, that the worker is the better forecaster, did not survive six more pairs.
- **First-order prices.** The price map is local. It missed a cross term between two settings, the effects of settings on the failure rate did not add (Table 32), one price was applied outside the place it was measured (Section 3.6), and a resource limit was closed twice on settings fitted to the old limit (Section 3.7).
- **Exactness rests on samples and a model.** Until hour 99 a change counted as exact when an aligned pair on the verifier's random inputs showed no difference; where no aligned pair exists, on a count of approximate elements in classes the loop chose, and its first choice of classes was too coarse. One accepted entry, E14, is inexact on inputs aimed at its structure (Section 3.9). Most other entries were cleared the same way and have not been attacked with aimed inputs; E17, which was, is separated from its base in both directions (Section 3.7).
- **Parts are enforced by convention.** The price map, the registry and the record by technique are kept as structured text that the planner maintains, not yet as data structures that code checks.
- **Supervision of workers is incomplete.** Some workers exceeded their resource limits, two stalled for hours before the planner noticed, and one long job started beside a running search took a rented machine down for five minutes.
- **Novelty is a search result.** The two nearest works were less than a week old when the run began; others may exist.

6 Conclusion

Recursive priced falsification is a topology of agents, not a prompt and not a pipeline of fixed steps: roles with narrowing views, gates that are tests, a planner that can be proved wrong, and a record that holds the state. In the vocabulary of self-improving agents, it puts the meta level under the external feedback that such loops keep for their candidates. It differs from published research loops less in its parts than in whom the parts apply to. Forecasts, falsifiers, ledgers, fresh contexts and failure memories are in the literature. The topology turns them on the planner, plans from measured prices whose optimality conditions are elementary and whose predictions can be checked, and makes every trial an asymmetric bet: a small stake that is known before the run, against a gain the stake does not cap.

Run autonomously on a live public benchmark for six days, with a frozen foundation model and concurrent agent harnesses as baselines, it produced most of the progress made there, including the two largest steps, and with the loop in the race the benchmark moved twelve times as fast as in the five weeks before. Its trials paid in the way its design asks for: a stake of about an hour, no gain kept on two trials in three, and ten trials that hold 79% of its points. Its largest step was an idea it had first turned down, brought back by a closure with a number and a forty-minute question. Its ledger shows where each role errs, a planner that prices costs too high and workers who expect

too little, and it shows the method applied to the loop itself: sixty-seven revisions of its procedure, and an arrangement of roles that the loop tested before it kept it.

The next experiments are the ones Section 5 asks for: the loop with rules and roles removed, on a second problem and a second harness, with the registry and the price map as code; and successive versions of the loop under a matched budget, which is the test of whether its revision of itself compounds.

AI Usage Statement

Language-model agents are the object of study in this work and also carried it out. Every research step reported here was taken by agents, as Section 3.1 sets out with what the human operator did. The text, the figures and the analysis scripts of this paper were written by the same agents from the run's records, and read and directed by the operator.

Acknowledgements

The benchmark's organisers built the public judge and the open record that made the comparison possible. Every entrant's entries and notes are public, and the loop read and built on them, as they did on its entries.

References

[1] L. A. Agrawal et al. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. *arXiv:2507.19457, 2025.* https://arxiv.org/abs/2507.19457

[2] O. Azizi et al. Energy-performance tradeoffs in processor architecture and circuit design: a marginal cost analysis. *ISCA 2010.* https://doi.org/10.1145/1815961.1815967

[3] F. Bianchi et al. Harnessing the Collective Intelligence of AI Agents in the Wild for New Discoveries. *arXiv:2606.10402, 2026.* https://arxiv.org/abs/2606.10402

[4] S. Boyd and L. Vandenberghe. Convex Optimization. *Cambridge University Press, 2004.* https://web.stanford.edu/~boyd/cvxbook/

[5] S. Chung et al. Autonomous Mathematical Discovery in an Open-World Multi-Agent Environment. *arXiv:2608.23691, 2026.* https://arxiv.org/abs/2608.23691

[6] W. Ding, N. Tomlin and G. Durrett. Calibrate-Then-Act: Cost-Aware Exploration in LLM Agents. *arXiv:2602.16699, 2026.* https://arxiv.org/abs/2602.16699

[7] Y. Duan et al. The Last AI Built by Humans: Toward Genuine Recursive Self-Improvement. *arXiv:2609.11873, 2026.* https://arxiv.org/abs/2609.11873

[8] R. Fang et al. Memp: Exploring Agent Procedural Memory. *arXiv:2508.06433, 2025.* https://arxiv.org/abs/2508.06433

[9] M. Farahbakhsh et al. AgentDiscover: Autonomous Discovery with Minimal Search Scaffolding. *arXiv:2610.05334, 2026.* https://arxiv.org/abs/2610.05334

[10] C. Fernando et al. Promptbreeder: Self-Referential Self-Improvement Via Prompt Evolution. *arXiv:2309.16797, 2023.* https://arxiv.org/abs/2309.16797

[11] T. Forster et al. Improving Hardware Requirements for Fault-Tolerant Quantum Computing by Optimizing Error Budget Distributions. *arXiv:2509.02683, 2025.* https://arxiv.org/abs/2509.02683

[12] T. S. Foster et al. AI Research Preference Models. *arXiv:2608.13940, 2026.* https://arxiv.org/abs/2608.13940

[13] C. Gidney. How to factor 2048 bit RSA integers with less than a million noisy qubits. *arXiv:2505.15917, 2025.* https://arxiv.org/abs/2505.15917

[14] J. Gottweis et al. Accelerating scientific discovery with Co-Scientist. *arXiv:2502.18864, 2025.* https://arxiv.org/abs/2502.18864

[15] B. Guo et al. EPOCH: Reliable Discovery through Evidence-Governed Search. *arXiv:2610.06986, 2026.* https://arxiv.org/abs/2610.06986

[16] S. Hu, C. Lu and J. Clune. Automated Design of Agentic Systems. *arXiv:2408.08435, 2024.* https://arxiv.org/abs/2408.08435

[17] J. Huang et al. Large Language Models Cannot Self-Correct Reasoning Yet. *arXiv:2310.01798, 2023.* https://arxiv.org/abs/2310.01798

[18] K. Huang et al. Automated Hypothesis Validation with Agentic Sequential Falsifications. *arXiv:2502.09858, 2025.* https://arxiv.org/abs/2502.09858

[19] A. Jain et al. Agentic AutoResearch for Space Autonomy: An Auditable, LLM-Driven Research Agent for Aerospace Control Problems. *arXiv:2606.20394, 2026.* https://arxiv.org/abs/2606.20394

[20] J. Ji et al. Rehearse: Stepping Back from the Confidence Cliff in Self-Improving Autoresearch. *arXiv:2607.27687, 2026.* https://arxiv.org/abs/2607.27687

[21] G. Juneja et al. Learning the ARTS of Search for Automated Discovery. *arXiv:2606.21891, 2026.* https://arxiv.org/abs/2606.21891

[22] A. Karpathy. autoresearch (software repository). *GitHub, 2026.* https://github.com/karpathy/autoresearch

[23] S. Kim and M. Pilanci. AI-Assisted Discovery of Convex Relaxations via Dual Agents. *arXiv:2606.31182, 2026.* https://arxiv.org/abs/2606.31182

[24] T. Kim et al. Agent Memory Distillation: Empowering Small LLM Agents with Hierarchical Teacher Memory. *arXiv:2608.07169, 2026.* https://arxiv.org/abs/2608.07169

[25] R. T. Lange et al. ShinkaEvolve: Towards Open-Ended And Sample-Efficient Program Evolution. *arXiv:2509.19349, 2025.* https://arxiv.org/abs/2509.19349

[26] J. Lao. Cutting 14 Million Gates from a Quantum Circuit. *rekursiv blog, 11 August 2026.* https://rekursiv.ai/blog/three-ai-teams-quantum-circuit/

[27] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh and A. Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research* 18(185):1–52, 2018. https://arxiv.org/abs/1603.06560

[28] G. Liu et al. Falsifiable Commitment Planning for Self-Correcting Web Agents. *arXiv:2607.24167, 2026.* https://arxiv.org/abs/2607.24167

[29] X. Liu et al. Janus: An Algorithm-Evaluator Co-Evolution Framework for LLM-Driven Discovery under Expensive Evaluation Budgets. *arXiv:2608.08189, 2026.* https://arxiv.org/abs/2608.08189

[30] J. Long et al. ECDSA.Fail: Open Autoresearch for Optimizing Elliptic-Curve Point Addition in Shor's Algorithm. *arXiv:2609.09582, 2026.* https://arxiv.org/abs/2609.09582

[31] C. Lu et al. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery. *arXiv:2408.06292, 2024.* https://arxiv.org/abs/2408.06292

[32] Z. Luo et al. The More You Automate, the Less You See: Hidden Pitfalls of AI Scientist Systems. *arXiv:2509.08713, 2025.* https://arxiv.org/abs/2509.08713

[33] Y. J. Ma et al. Eureka: Human-Level Reward Design via Coding Large Language Models. *arXiv:2310.12931, 2023.* https://arxiv.org/abs/2310.12931

[34] L. Mitchener et al. Kosmos: An AI Scientist for Autonomous Discovery. *arXiv:2511.02824, 2025.* https://arxiv.org/abs/2511.02824

[35] L. G. Mitten. An Analytic Solution to the Least Cost Testing Sequence Problem. *Journal of Industrial Engineering* 11(1):17, 1960.

[36] A. Novikov et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv:2506.13131, 2025*. https://arxiv.org/abs/2506.13131

[37] S. Ouyang et al. ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory. *arXiv:2509.25140, 2025*. https://arxiv.org/abs/2509.25140

[38] W. Pan et al. The Tasteful Agent: Measuring and Improving Taste in Long-Horizon Tasks. *arXiv:2609.25804, 2026*. https://arxiv.org/abs/2609.25804

[39] B. Rank et al. PostTrainBench: Can LLM Agents Automate LLM Post-Training?. *arXiv:2603.08640, 2026*. https://arxiv.org/abs/2603.08640

[40] M. Robeyns, M. Szummer and L. Aitchison. A Self-Improving Coding Agent. *arXiv:2504.15228, 2025*. https://arxiv.org/abs/2504.15228

[41] B. Romera-Paredes et al. Mathematical discoveries from program search with large language models. *Nature* 625:468–475, 2023. https://doi.org/10.1038/s41586-023-06924-6

[42] A. Safdar et al. Long-Horizon Autonomous Architecture Research with a Language-Model Agent: A Behavioural Case Study. *arXiv:2608.01995, 2026*. https://arxiv.org/abs/2608.01995

[43] S. Schmidgall et al. Agent Laboratory: Using LLM Agents as Research Assistants. *arXiv:2501.04227, 2025*. https://arxiv.org/abs/2501.04227

[44] J. Schmidhuber. Goedel Machines: Self-Referential Universal Problem Solvers Making Provably Optimal Self-Improvements. *arXiv:cs/0309048, 2003*. https://arxiv.org/abs/cs/0309048

[45] S. Shao et al. Your Agent May Misevolve: Emergent Risks in Self-evolving LLM Agents. *arXiv:2509.26354, 2025*. https://arxiv.org/abs/2509.26354

[46] M. Sharma et al. AutoQResearch: LLM-Guided Closed-Loop Policy Search for Adaptive Variational Quantum Optimization. *arXiv:2604.24283, 2026*. https://arxiv.org/abs/2604.24283

[47] N. Shinn et al. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv:2303.11366, 2023*. https://arxiv.org/abs/2303.11366

[48] I. Shumailov et al. The Curse of Recursion: Training on Generated Data Makes Models Forget. *arXiv:2305.17493, 2023*. https://arxiv.org/abs/2305.17493

[49] H. A. Simon and J. B. Kadane. Optimal Problem-Solving Search: All-or-None Solutions. *Artificial Intelligence* 6(3):235–247, 1975. https://doi.org/10.1016/0004-3702(75)90002-8

[50] T. Smith. The Little Scientist: LLM Agent-Driven Discovery via the Scientific Method. *arXiv:2608.16951, 2026*. https://arxiv.org/abs/2608.16951

[51] J. Snoek, H. Larochelle and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *NeurIPS 2012*. https://arxiv.org/abs/1206.2944

[52] Y. Song et al. Mind the Gap: Examining the Self-Improvement Capabilities of Large Language Models. *arXiv:2412.02674, 2024*. https://arxiv.org/abs/2412.02674

[53] D. Srikanth et al. Recursive self-improvement of AI research agents. *arXiv:2609.26457, 2026*. https://arxiv.org/abs/2609.26457

[54] I. Takahara et al. Toward Auditable AI Scientists: A Hypothesis Evolution Protocol for LLM Agents. *arXiv:2607.09195, 2026*. https://arxiv.org/abs/2607.09195

[55] E. Toledo et al. AI Research Agents for Machine Learning: Search, Exploration, and Generalization in MLE-bench. *arXiv:2507.02554, 2025*. https://arxiv.org/abs/2507.02554

[56] Y. Virk et al. SwarmResearch: Orchestrating Coding Agents for Open-Ended Discovery. *arXiv:2607.02807, 2026*. https://arxiv.org/abs/2607.02807

[57] G. Wang et al. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv:2305.16291, 2023*. https://arxiv.org/abs/2305.16291

[58] W. Wang et al. Huxley-Gödel Machine: Human-Level Coding Agent Development by an Approximation of the Optimal Self-Improving Machine. *arXiv:2510.21614, 2025*. https://arxiv.org/abs/2510.21614

[59] H. Wang et al. Negative Knowledge as Failure-aware Shared Memory for AutoResearch. *arXiv:2606.21024, 2026*. https://arxiv.org/abs/2606.21024

[60] C. Wang et al. Fail-Fast, Restart-Smart: Early Failure Prediction and Restart for SWE Agentic Tasks. *arXiv:2608.03222, 2026*. https://arxiv.org/abs/2608.03222

[61] M. L. Weitzman. Optimal Search for the Best Alternative. *Econometrica* 47(3):641–654, 1979. https://doi.org/10.2307/1910412

[62] J. Wen et al. Predicting Empirical AI Research Outcomes with Language Models. *arXiv:2506.00794, 2025*. https://arxiv.org/abs/2506.00794

[63] Y. Weng et al. DeepScientist: Advancing Frontier-Pushing Scientific Findings Progressively. *arXiv:2509.26603, 2025*. https://arxiv.org/abs/2509.26603

[64] Z. Weng et al. Group-Evolving Agents: Open-Ended Self-Improvement via Experience Sharing. *arXiv:2602.04837, 2026*. https://arxiv.org/abs/2602.04837

[65] H. Wijk et al. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts. *arXiv:2411.15114, 2024*. https://arxiv.org/abs/2411.15114

[66] W. Xu, Y. Liu, Y. Nan et al. Neural Architecture Discovery via Autonomous Evolution. *COLM 2026; arXiv:2507.18074, 2025 (version 1 titled "AlphaGo Moment for Model Architecture Discovery")*. https://arxiv.org/abs/2507.18074

[67] Y. Yamada et al. The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search. *arXiv:2504.08066, 2025*. https://arxiv.org/abs/2504.08066

[68] R. Yang et al. ARIS: Autonomous Research via Adversarial Multi-Agent Collaboration. *arXiv:2605.03042, 2026*. https://arxiv.org/abs/2605.03042

[69] Y.-T. Yang et al. Toward Reliable Design of LLM-Enabled Agentic Workflows: Optimizing Latency-Reliability-Cost Tradeoffs. *arXiv:2605.23929, 2026*. https://arxiv.org/abs/2605.23929

[70] S. Yang et al. Advancing Model Research in AgentX: Long-Horizon Autonomy for Industrial Recommender Systems. *arXiv:2609.30001, 2026*. https://arxiv.org/abs/2609.30001

[71] X. Yin et al. Gödel Agent: A Self-Referential Agent Framework for Recursive Self-Improvement. *arXiv:2410.04444, 2024*. https://arxiv.org/abs/2410.04444

[72] X. Yu et al. PrimeScientist: Strategic Allocation of Research Effort in Autonomous Research. *arXiv:2609.17846, 2026*. https://arxiv.org/abs/2609.17846

[73] W. Yuan et al. Self-Rewarding Language Models. *arXiv:2401.10020, 2024*. https://arxiv.org/abs/2401.10020

[74] E. Zelikman et al. STaR: Bootstrapping Reasoning With Reasoning. *arXiv:2203.14465, 2022*. https://arxiv.org/abs/2203.14465

[75] E. Zelikman et al. Self-Taught Optimizer (STOP): Recursively Self-Improving Code Generation. *arXiv:2310.02304, 2023*. https://arxiv.org/abs/2310.02304

[76] J. Zhang et al. Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. *arXiv:2505.22954, 2025*. https://arxiv.org/abs/2505.22954

[77] Q. Zhang et al. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. *arXiv:2510.04618, 2025*. https://arxiv.org/abs/2510.04618

[78] J. Zhang et al. Hyperagents. *arXiv:2603.19461, 2026*. https://arxiv.org/abs/2603.19461

[79] Y. Zhang et al. An AI Scientist that Doesn't Drift: Taste, Structure, and Falsifiable Findings in a Quadruped Navigation Research Loop. *arXiv:2608.07542, 2026*. https://arxiv.org/abs/2608.07542

[80] Q. Zhang et al. Stateless Language Agents: Scaling Long-Horizon Automated Research. *arXiv:2610.07625, 2026*. https://arxiv.org/abs/2610.07625

[81] Z. Zhang et al. Evolving in the Agent Jungle via History-Informed Opponent Awareness. *arXiv:2608.02005, 2026*. https://arxiv.org/abs/2608.02005

[82] Y. Zhang et al. EvalResearchBench: Can AI Agents Design Their Own Evaluations?. *arXiv:2610.04184, 2026*. https://arxiv.org/abs/2610.04184

[83] Y. Zhang et al. Budget-Aware LLM Discovery via Cost-Calibrated Frontier Utility. *arXiv:2607.26828, 2026*. https://arxiv.org/abs/2607.26828

[84] A. Zhao et al. Absolute Zero: Reinforced Self-play Reasoning with Zero Data. *arXiv:2505.03335, 2025*. https://arxiv.org/abs/2505.03335

[85] J. Zheng et al. Can We Predict Before Executing Machine Learning Agents?. *arXiv:2601.05930, 2026*. https://arxiv.org/abs/2601.05930

[86] R. Zhenzhuo et al. EULER: Exploring Underused Links with Evidence-Checked Return for Multi-Agent Mathematical Discovery. *arXiv:2609.00032, 2026*. https://arxiv.org/abs/2609.00032

[87] A. Zweiger et al. Self-Adapting Language Models. *arXiv:2506.10943, 2025*. https://arxiv.org/abs/2506.10943

Appendix

Contents

	Contents	Supports	Page
A	**Related Work, Mechanism by Mechanism**	Sec. 4	48
A.1	How the Citations Were Checked		51
B	**What the Technique Asks of a Harness**	Sec. 2.3	51
C	**Further Parts of the Method**	Sec. 2.10	54
C.1	Which Comparison Decides That a Change Is Exact		55
C.2	Instruments Before Ideas, and the Record by Technique		57
C.3	Search With No Memory		57
C.4	Context Isolation, Planner Rotation and the Clean Sheet		57
D	**Experimental Setup Details**	Sec. 3.1	58
D.1	Notation	Sec. 2	58
D.2	The Technique on the Benchmark	Sec. 3.1	59
D.3	The Loop's Seventeen Entries	Sec. 3.3	60
E	**Evidence for the Mechanisms, in Full**	Sec. 3.8	61
F	**Detailed Results**	Sec. 3	67
F.1	The Stages of the Case Study	Sec. 3.6	67
F.2	The Price Map's Prediction for One Package	App. E.1	67
F.3	Forecast Accuracy	App. E.4	68
F.4	The Record by Technique as a Forecaster	App. E.5	68
F.5	Where One Day's Accepted Gates Came From	App. E.6	68
F.6	What the Remaining Mechanisms Did, by Count	App. E	69
G	**The Forecast Set**	App. E.4	70

A Related Work, Mechanism by Mechanism

We searched for each mechanism separately, then for the combination, and read a public collection of 31 papers on self-improving agents one by one. In the parts we read, none describes two forecasts on one experiment, required errata on the task, a red team on closed directions, or a price map. "Published" below means we do not claim it. "Not found" reports a search, not a proof of absence.

Table 16: Each mechanism against the closest published work.

Mechanism	Closest work, and what it has	What we did not find there	Status
A forecast and a falsifier before each run	A seven-field plan with a numeric prediction and a falsification criterion [50]; a proposal with predicted effect and intended falsifier [15]; a fitness prediction stored before scoring [9]; immutable experiment cards [79]; auditable loops [19, 54]; pre-registered experiments on the same benchmark [26]	–	Published; adopted
A second forecast by the planner, scored by role	One forecaster per run in all of the above; forecasting research outcomes as a task [62, 85]; models that choose which candidate to run [12]; confidence in self-improving loops [20]	Two forecasts on one experiment; the planner's bias tracked and fed back	Not found
Required errata on the task	A shared record that corrected a supervising session [26]; commitments that a web agent checks and repairs itself [28]; telling a bad idea from a bad build [21]	Correction of the plan as a duty of every report	Not found as a rule
Closures with an evidence grade, a parked state and a reopening number	Failed-route records with conditions for reopening [86]; negative knowledge as shared memory [59]; a three-valued admission gate and claims scoped by evidence type [15]; measured non-improvements kept apart from inconclusive failures [80]	Argument-only closures held as parked; overturns counted by grade	Published in part
A red team aimed at the planner's closed list	An agent that searches for counterexamples to each proposed constraint [23]; sequential falsification of hypotheses [18]; adversarial review [68]; challenges aimed at a candidate's mechanism [15]	All aim at new positive claims, none at the planner's negative conclusions	Not found
A rebuilt planner after a stalled round	Context resets after a stall [9]; every invocation rebuilt from harness state [80]; explorers with fresh contexts [56]; restart on predicted failure [60]; a second agent at a plateau [50]	–	Published; adopted, in a weaker form than Zhang et al. [80]

Table 16, continued

Mechanism	Closest work, and what it has	What we did not find there	Status
A price map as the planning object	Outside research agents: shadow prices [4], marginal-cost analysis in processor design [2], error-budget distribution in fault-tolerant computing [11], shadow prices to split a workflow's budget [69]. Inside: allocation of research effort [63, 72]; progress per realised action cost [83]; cost-aware exploration [6]	Measured exchange rates between the objective's resources, used to close ideas and to build packages	Cost per action is used; exchange rates between the objective's own resources were not found
A verifier ladder built by the system, with an admission rule	A cascade of evaluators set by the user [36]; proxy evaluators calibrated on real outcomes [29]; independent replay [15]; closed-loop policy search [46]; paired comparison under identical conditions [81]; agents designing evaluations as a benchmark [82]	Reproducing the incumbent's known number as the gate for a new instrument	Published in part
One refit for a package of changes	Several parts of one system optimised together [1]; long-horizon model research [42, 70]	The bar applied to a package that shares one refit	Published in part
Records as the state; technique cards	Harness-owned state [80]; a knowledge graph of attempts [9]; an evidence ledger [15]; reasoning and procedural memories [8, 24, 37, 77]	–	Published
The combination	Evidence governance around one proposer [15]; a stateless advisor with workers [80]; a research loop with immutable experiment cards [79]	A planner that is itself forecast-scored, corrected from below and red-teamed, planning from a price map	Not found

What is distributed. A research system can spread four things: who may start work, where the memory lives, who judges, and who may prove whom wrong. On the first, an open environment with no central coordinator [5], an arena of independent agents [3] and the open benchmark itself [30] are more decentralised than our method, which has a planner at its hub. On the second, Stateless Language Agents is stricter: no agent keeps its conversation [80]. Priced falsification spreads the fourth further than any system we found: the right to be proved wrong reaches the hub (Table 17). The capabilities of Section B are what that spreading needs from the software underneath.

Table 17: Where authority sits. Cells hold what we could check in each work's text; "not described" means we did not find it.

System	Who plans	Where the state lives	Who is held to a forecast	Can a report correct the plan, by rule
autoresearch [22]	One agent	A results file and version history	Not described	No separate planner
AlphaEvolve [36]	A fixed evolutionary controller	A program database	Not described	Not described
SwarmResearch [56]	A coordinating agent over search agents	Selective sharing; fresh contexts for explorers	Not described	Not described
Stateless Language Agents [80]	A stateless advisor, rebuilt each epoch	The harness	Not described	Not described
AgentDiscover [9]	The search agent; a meta agent writes guidance	A database of ideas and candidates	The search agent, on its own candidate	Not described
EPOCH [15]	One proposal policy with phases	An append-only evidence ledger	The proposer, with an intended falsifier	No workers described
The Little Scientist [50]	One scientist agent; a second agent at a plateau	Versioned files	The scientist, with a falsification criterion	Not described
Zhang et al. [79]	One research agent; subagents in mechanical roles	Immutable experiment cards	The agent, on each card	Not described
EULER [86]	Several agents with an evidence check	Records of failed routes with reopening conditions	Not described	Not described
The Station [5]	No one; agents choose	A shared archive of reviewed papers	Not described	No plan to correct
A three-team run on the same benchmark [26]	Three teams, with supervising sessions	A shared record	The teams	It happened: the record corrected a supervising session. Not stated as a rule

Table 17, continued

System	Who plans	Where the state lives	Who is held to a forecast	Can a report correct the plan, by rule
Priced falsification	A planner, rebuilt after a round with no gain	Ledger, registry, findings, cards	Planner and worker, each with its own forecast, scored by role	Yes: required errata in every report, and a red team on the closed list

The two closest works, EPOCH [15] and Stateless Language Agents [80], appeared in the week before the run. EPOCH governs evidence around a single proposing policy; its text describes no workers and no party that corrects the proposer's picture of the task. Stateless Language Agents moves all state out of the agents and shows large gains from doing so; its advisor is not held to a forecast. Priced falsification is weaker than each on the point that work is built around, and adds what neither has: a planner under test, and prices.

Work on the limits of self-correction supports the design (Section 4). Models correct their own reasoning poorly without an outside signal [17], and self-improvement is bounded by the gap between generating and verifying [52]. Here the planner's errors are found by a different session with a measurement in hand, not by the planner thinking again. Reports of hidden failure modes in automated-science systems [32] are the reason the planner's mistakes are published.

A.1 How the Citations Were Checked

Table 18 says how each kind of source was checked. Two details that a search suggested about other papers could not be found in their text and are not used.

B What the Technique Asks of a Harness

Priced falsification is a topology of agent roles. It is not tied to a language model, and it is not tied to an agent framework. It does assume things about the software that runs the agents, which we call the *harness*. This appendix states them, so that the technique can be carried to another harness.

Together the seven describe a *distributed* harness: several agents with separate memories that coordinate through a record and can check one another. The distribution is not there for speed. It is what makes the planner testable. A second forecast is worth something only if it comes from a context the planner did not write (H1). A re-check is worth something only if it starts from a copy the claimant did not touch (H2). A forecast binds only if it was written where it cannot be edited (H3).

A single agent session with tools can keep a ledger and write forecasts. It cannot supply H1, H2 or H4 to itself, and those are the three that put the planner under test.

The requirements say nothing about the model. The technique needs no special model, no fine-tuning and no learned controller. In the run reported here the planner, scouts, pricers and builders

Table 18: How the sources were checked.

Sources	How they were checked
The 75 arXiv entries among the 80 that the list held before an independent review on 14 October 2026	Opened on arxiv.org; title and authors matched against the entry. All 79 arXiv entries were matched again against arXiv's records on 15 October 2026; one had since been retitled by its authors and is cited by its current title
The other five of those 80: a book, a conference paper, a journal article, a software repository and a blog post	Opened at the address given; the two with a DOI matched against Crossref on 15 October 2026
Two 2026 preprints added after the review [6, 83]	Opened on arxiv.org and matched
Five classical works added at the same time [27, 35, 49, 51, 61]	Matched on 15 October 2026 against Crossref or arXiv; Mitten [35], which neither indexes, against later works that cite it
What a paper contains	Taken from its abstract, except for Guo et al. [15], Zhang et al. [80], Farahbakhsh et al. [9], Smith [50], Chung et al. [5], Liu et al. [29], Zhenzhuo et al. [86], Virk et al. [56] and Lao [26], where the statement was checked in the full text
The works on self-improving agents cited in Sections 1 and 4	Each read in the main text by a reader agent, section by section, through a tool that returns summaries; not checked word for word, except Zelikman et al. [75], whose text we read directly. Romera-Paredes et al. [41] was read in its abstract only and Duan et al. [7] to its third section
Numbers about the run	From its ledger, findings and journal; scores and acceptance times from the public benchmark; each other entrant's model and harness from the header of its public notes

were one model with different tasks; breakers ran on a second model, which is useful and not required.

The harness used for the run provides all seven. We do not describe it further. The technique is the topology of Section 2, and a harness that supplies H1 to H7 can run it. The benchmark run was made on one harness only, and we say so among the limitations (Section 5).

B.1 The Released Package

We release the topology as a package that installs into a coding agent with one command, at github.com/pensievelab/rpf. It has two parts. The first is a skill: the planner's workflow, one round from measurement to revision, and a file for each role (scout, thinker for a clean-sheet round, pricer, builder, breaker and red team). The second is a small set of tools, one command `rpf`, that hold what should not be left to an agent's judgment (Table 20).

The package is agnostic to the harness. The planner is the host's own agent with the skill, and Table 21 shows how the package meets each capability of Table 19 on any host.

The package is not the software of the six-day run, which was made on Pensieve. It carries the same roles, report contracts, ranking rule and record to other harnesses, and we have not used it

Table 19: The seven capabilities the technique needs, and what breaks without each.

	Capability	Rules that use it	Without it
H1	**Separate contexts.** A planning session can start other sessions that share none of its conversation	The second forecast; the worker with no earlier reports; the red team; the rebuilt planner	The second forecast repeats the first, and no agent is placed to contradict the planner
H2	**Isolated copies of the work.** Each worker changes its own copy; changes meet only in an explicit merge	Parallel studies; the planner's re-check from a second copy; one refit for a package	Studies overwrite each other, and a claimed result cannot be rebuilt by anyone but its author
H3	**A shared record outside every session**, append-only, that all sessions read and write	Forecasts written before a run; closures with their numbers; the planner rebuilt from a digest	The state of the research lives in one conversation; forecasts can be revised after the fact; rotation is impossible
H4	**Scoped reading.** The harness decides what a session is given to read	Reading lists; the worker that reads no earlier report	Every agent inherits every earlier conclusion, including the wrong ones
H5	**Parallel, bounded execution.** Several workers at once, each under limits on time and compute, with the planner told when each ends	A round of several questions; follow-ups chosen while a build is still fresh	One line of work at a time, so prices are never compared across ideas within a round
H6	**Shared instruments.** A tool built by one session can be run by any other, against a scorer that no worker can change	The verifier ladder; the admission rule; the planner's re-check	Each claim is checked only by the agent that made it
H7	**A gate on outside effects.** Actions that leave the loop wait for the operator; nothing else does	Autonomy inside the loop, accountability at its edge	Either a person in every step, or no person at all

to repeat the benchmark run. It differs from the run in one rule: the odds of a card start at its scout's figure and move toward its family's record as that record grows, where the run took half of each; with no record, the run's rule pulled a card its scout had ruled out up to even odds. As a check that an agent can follow it unaided, two fresh agent sessions, on the version before `pair`, `frontier`, `numbers` and `digest` were added, were each given only the installed skill and a small task with an exact evaluator: a straight-line program scored by its operations times its registers. Each took the seed from 66 to 20 in one round with every worker in a fresh context; the second did so with four workers in twelve minutes. Each run also returned a list of faults in the skill's text and tools, which were corrected.

Table 20: The tools of the released package.

Command	What it holds
`rpf init`	The record: a ledger of trials, a closed list, the rounds, and the run's own copy of the workflow
`rpf brief`	A worker's task, with the planner's forecast recorded first and left out of the worker's sight. A builder gets its own copy of the files; a breaker gets the change and the claim, never the reasons
`rpf worker`	A worker in a fresh, headless session of the host, stopped at its stake
`rpf check`, `rpf recall`	Each report against its contract; for an incomplete one, the message that sends it back to its author
`rpf desk`	The ranking of cards by expected gain per unit of stake. Under a floor it picks nothing, which is the signal to stop
`rpf trial`	How a trial ended. It needs the brief's record, so a forecast always comes before its result, and a gain needs its paired comparison
`rpf pair`	The paired comparison of a base and a new design on the same inputs, case by case: outputs that differ, cases wrong in only one, and the change in cost with its paired error
`rpf close`, `reopen`, `closed`	The closed list: a direction cannot be closed without the grade of its evidence and the number that would reopen it
`rpf frontier`	The best score outside the run, how far a step there usually goes, and whether a margin beats it
`rpf numbers`	Every number in an outgoing text that no result file holds
`rpf round`, `rpf score`	Each round under the version of the workflow that ran it; the forecasts scored by role, and each version by its rounds
`rpf digest`	The record for a rebuilt planner: what paid, the record by family and stage, the closed list. Facts and prices, no plan

Table 21: How the package supplies the seven capabilities.

	Capability	In the package
H1	Separate contexts	Each worker is a fresh session of the same host
H2	Isolated copies of the work	Each worker gets its own copy of the files it may change
H3	A shared record	A folder of files outside every session
H4	Scoped reading	Each task lists what its worker may read
H5	Parallel, bounded execution	Each task carries a bound in minutes
H6	Shared instruments	The tools are one command that any session can run
H7	A gate on outside effects	Stays with the host and its operator

C Further Parts of the Method

This appendix gives in full the four parts of the topology that Section 2.10 states in brief.

C.1 Which Comparison Decides That a Change Is Exact

Many changes are meant to leave the design's behaviour untouched: the same outputs on every input, at a lower cost. Whether that is true is the most common question a builder must answer, and the pairing rule alone does not answer it.

A verifier that draws random inputs usually indexes its random numbers by position: the k-th random number goes to the k-th place in the design that asks for one. A change that adds, removes or moves one such place shifts every later index. The two designs then see different random numbers on the same input, and their failures decorrelate.

Proposition 7 (Misalignment undoes pairing)**.** *In the setting of Proposition 5, let the change be exact, so that both designs fail at rate λ, and suppose a fraction θ of those failures depend on random numbers whose index the change has shifted. Then the failures the two designs share have mean $(1-\theta)\lambda$, and*

$$\mathrm{Var}(\bar{D}) = \frac{2\theta\lambda}{n}, \tag{10}$$

where an aligned pair has $\mathrm{Var}(\bar{D}) = 0$. A real difference δ in the failure rate is resolved at two standard errors only if $\delta > 2\sqrt{2\theta\lambda/n}$.

Proof. An exact change has no failures of its own, so with aligned random numbers $U_A = U_B = 0$ and $\kappa = 0$. A failure that depends on a shifted random number recurs in the other design only by chance; counting those as unshared gives $\mathbb{E}[U_A] = \mathbb{E}[U_B] = \theta\lambda$, and Proposition 5 gives $\kappa = 2\theta\lambda$. □

In words. An exact change can look like a change in the failure rate, and a real change can hide, for the same reason: the two designs were not shown the same randomness. On equal inputs with shifted indices, "no output differs on a few packets" says little.

The rule is to pick the comparison by what the change does to the indices (Table 22).

Table 22: Which comparison decides that a change is exact. The first three run on the verifier's own random inputs and bound a change in the failure rate; the fourth is for claims that rest on an argument.

Comparison	Use when the change	How	The change passes when
Aligned pair	Adds, removes or moves a few random draws	A copy of the new design is padded with idle draws until both sides ask for the same numbers in the same order; outputs and failure flags are compared shot by shot	No shot passes in the base and fails in the new design
Control pair	Rewrites a whole stage, so that no padding aligns it	The base is compared with itself with only that stage's indices shifted, which measures θ directly	No shot is wrong only in the new design, there are no more changes of verdict than the control shows, and planted faults are seen
List by structure	Shifts the indices everywhere, as a new resource limit does	Every approximate element of both designs is listed by its place and its strength, and the lists are compared as multisets, class by class	No class holds more approximate elements than the base's
Aimed inputs	Replaces an exact element by one that is exact only by an argument	A breaker derives the inputs on which the argument would fail and draws inputs there	A new form: no aimed class leans against it. A re-fitted plan: the net over inputs nobody picked, with one-sided classes reported

Why the list by structure overrides a paired run. When the indices shift everywhere, neither pair exists. A design that has more approximate elements than the base in any class is not "no worse", whatever a paired run of practical size reads, because by Proposition 7 that run cannot see the difference.

A design that fails the third comparison can be repaired by an *offset*: other elements are strengthened until the lists balance. An offset is checked like any other claim, by the failures it buys back on an aligned pair, and not by its arithmetic. Section 3.7 gives a case where the arithmetic balanced and the offset bought back one failure where it owed nine.

Why aimed inputs are needed. The first three comparisons bound a change in the failure rate; they do not prove that a change is exact. A change that is wrong only on a class of inputs rarer than about one in nm shots, for n packets of m shots, passes every one of them. So a claim that rests on an argument, such as a carry that is said never to occur or a window that is said to be wide enough, goes to a breaker. Section 3.9 gives the case that put this comparison on the list: an aligned pair of 240 packets passed a change that aimed inputs broke (Table 13). What counts as broken depends on the kind of change. A change that adds a new form fails if any class of aimed inputs leans against it, because nothing fails the other way. A change that re-fits a plan moves its approximate elements to other places, so the old plan and the new each fail on inputs where the other passes; there the net over inputs that nobody picked decides, and the one-sided classes are reported beside it (the case of E17, Section 3.7).

C.2 Instruments Before Ideas, and the Record by Technique

A round can start from a list of ideas or from a measurement of the design. The technique starts from the measurement.

After every accepted entry, before any scout runs, three instruments are read on the new design. A *ledger* gives its cost block by block, and sets side by side any two blocks that do the same work. A *refit* sweeps every setting whose price the entry has changed. And each *resource limit* is priced again with those settings refitted, because by Corollary 1 a limit is optimal only for the prices it was fitted to. The cards of the round are then written against this map.

The reason is an asymmetry in what the two sources cost to check. A card that comes from the ledger arrives with its price: the block, its count, and the cheaper count next to it. A card that comes from a list of ideas has to be priced from nothing, and most are priced to zero. Section E.6 compares the two on one day of the run.

The record by technique. Every trial in the ledger is filed under a technique family (a new design; an extension of a design that has just paid; a setting; a port of a rival's idea; a paper price; a search; a check) and under the stage it reached. The pay rate of each family and stage so far is the planner's prior for the next trial of that kind, and each task names its family and that family's record. This turns the bias readings of Section 2.5 from a correction of single forecasts into a rule for where to look.

C.3 Search With No Memory

On problems where the exact verifier is a search for a passing instance, the search is the dearest rung, and two plain facts govern it.

Proposition 8 (Switching costs nothing)**.** *Let each try on design A pass independently with probability p_A, and each try on design B with p_B. After any number of failed tries on A, the expected number of further tries to a pass is $1/p_A$ on A and $1/p_B$ on B. A search on A is therefore moved to B whenever B has cleared its gates and $J(B) < J(A)$, whatever has been spent on A.*

Proof. The number of tries to the first pass is geometric, and the geometric distribution is memoryless: the tries already made carry no information about the tries to come. □

In words. Tries already spent are neither lost nor kept by a switch. The only cost of a switch is the difference in pass rates, and the gain is the better score. A design that clears its gates in the middle of a search takes the machines at once.

The second fact is about time. Where entries are public and any party may build on the last one, a cleared design loses value by the hour. The loop therefore watches the benchmark while it searches, stops a search when the best public score moves, and sends each cleared improvement as soon as the verifier accepts it instead of stacking several into one entry.

C.4 Context Isolation, Planner Rotation and the Clean Sheet

An agent that has argued for a conclusion tends to keep it. The technique does not try to argue an agent out of this. It changes who is asked.

A worker reads only what its brief lists. One worker per round may be told to read no earlier report at all; it sees the design and the question and nothing of what the loop already believes.

After a round that keeps no gain, the next round is planned by a new session. It receives a digest of the record and not the previous planner's conversation, and it is told that it may disagree. Rebuilding agents from records is the central idea of Stateless Language Agents [80] and of context resets in AgentDiscover [9]; we apply it to the planner on a stall.

The clean sheet. Rounds improve the design that is there. When the ledger shows that the large blocks are each at the floor of their present form, the question changes from what in this design is dear to whether there is another design. A clean-sheet round sends several *designers*, each made to start from a different place. Each is given the task, the number to beat and the present design in a few lines, so that it is not proposed again. Each returns one whole design counted on paper, with the assumption most likely to break it tested on a small model, and its own odds before and after counting. A design within a stated factor of the accepted one goes to a pricer. A round of this kind that finds no design is still worth its cost if each loss names its cause, because a cause closes a family of designs and an opinion closes nothing.

D Experimental Setup Details

D.1 Notation

Table 23: Notation.

Symbol	Meaning
x, $r_i(x)$	A design, and the amount of resource i it consumes
$J = f(r)$	The objective; lower is better
C_i	The cap on a budgeted resource
π_{ij}	Objective rate: the amount of resource j that is worth one unit of resource i to the objective
ρ_a	Design rate of knob a: units of resource j saved per unit of resource i spent
$\hat{y}_P$, $\hat{y}_W$, y	The planner's forecast, the worker's forecast, and the outcome
$e = \ln(y/\hat{y})$	The log error of a forecast
ϕ	A falsifier: the outcome that ends an idea
c_i, p_i	The cost of rung i of the verifier ladder, and its chance of refuting a claim
λ, κ	Mean failures a test packet; mean failures a packet that two designs do not share

D.2 The Technique on the Benchmark

Table 24: The technique as instantiated on the benchmark.

Part of the technique	On this benchmark
Objective and resources	$J = T \cdot Q$: gates T, qubits Q; failing shots λ budgeted
Objective rate	One qubit is worth $T/Q \approx 636$ gates at entry E4 and 617 at entry E14
Verifier ladder	An integer identity checked on millions of inputs (minutes); a gate-level model of one block, checked on every input of its fixed part (minutes); the circuit builder's gate count (45 s); a paired run of two circuits on identical inputs, aligned where the change allows (10 min); a breaker's own tests; a search for a passing packet (hours); the official judge (2 min)
Instruments built	An exact re-implementation of the judge; the paired test and the padding that aligns it; a list of every approximate element by place and strength; a ledger of the gate count by block; an integer model of the main loop; a gate-level model of the reduction; a three-stage search; a GPU port of the search's first stage; a watcher on the public record
Scale of the run	138.7 hours, 42 rounds, 146 worker agents
Record at the end	174 numbered findings, 267 ledger lines, 333 technique cards, 67 logged planner mistakes
Compute for the searches	One rented GPU at a time until hour 88, then two or three cheaper ones; under two dollars a search

D.3 The Loop's Seventeen Entries

Table 25: The loop's accepted entries. Steps are against the best entry before.

Entry	Hour	Step	Source of the gain	Part of the technique behind it
E1	7.2	−0.138%	Nine exact rewrites of control logic, with a wider safety table	Verifier ladder: gains of this size are below the noise of pass or fail
E2	7.9	−0.220%	The same rewrites on the original table	Verifier ladder; paired test
E3	11.2	−0.080%	A new invariant of the main loop frees one qubit at the peak	A census of the scarcer resource
E4	24.8	−1.045%	A new decision rule that removes one iteration, plus two structural changes, refitted as one package	Rebuilt planner; falsifiers; the worker with no earlier reports; package refit
E5	37.3	−0.044%	The failure budget re-traded at better prices	Price map (Corollary 2)
E6	50.5	−1.449%	One reduction an iteration in place of three, on about two thirds of the iterations, with a smaller reduction core	A parked idea and its reopening number; the verifier ladder, rung by rung (Section 3.6)
E7	53.6	−0.310%	The new operation on 47 more passes, with smaller cores	An extension of a design that had just paid, begun while E6 was still in its search
E8	55.3	−0.009%	The edges of the new operation	The same, on a rival's entry that had landed in between
E9	58.0	−0.237%	The loop's latest rounds merged onto Entrant C's entry	Reading a rival's design as a list of trades; a merge admitted on known answers
E10	64.2	−0.080%	Two additions cut at their top and three more passes	A search moved to a better design in mid-search (Proposition 8)
E11	77.8	−0.104%	A block's phase computed without writing its register	The first round through every role; the breaker (Section 3.7)
E12	82.5	−0.359%	A phase paid where the reverse stage rebuilds the value, and the setting this unlocked	A literature scout holding one paid technique; the aligned pair
E13	89.5	−0.018%	Four exact changes ported onto Entrant A's entry	Aligned pairs; one change from the two-agent dialogue
E14	93.3	−0.020%	One stage's first iteration as one shift and one addition (not exact: Section 3.9)	A ledger of two stages that do the same work
E15	96.6	−0.007%	One qubit less, with settings refitted and an offset	A resource limit priced again; the list by structure (Section 3.7)
E16	101.0	−0.011%	The two largest changes of Entrant A's entry, measured on the loop's design, and one gate built under a second bit	A merge admitted piece by piece; a breaker that withdrew the first form six minutes into its search
E17	138.7	−0.040%	A wire that only held a copy, lent to the main addition on 127 iterations; Entrant A's four passes over the gate list carried as one table	A census of the finished gate list; the breaker's verdict read on the net over inputs nobody picked (Section 3.7)

E Evidence for the Mechanisms, in Full

This appendix tests the parts of the technique against the records of the run. It stands where an ablation study would stand in a controlled experiment and is weaker than one: each part is tested against the record, and none is removed. Every number is computed by a script from the ledger and the journal of the run and from the benchmark's public history. Table 14 in Section 3.8 lists the tests, and the other tables of this appendix are in Appendix F; the paragraphs below give each test in turn. The forecast ledger analysed here covers the first fifty hours, where every numeric forecast was extracted and checked by hand; the later rounds are reported by count.

E.1 The price map

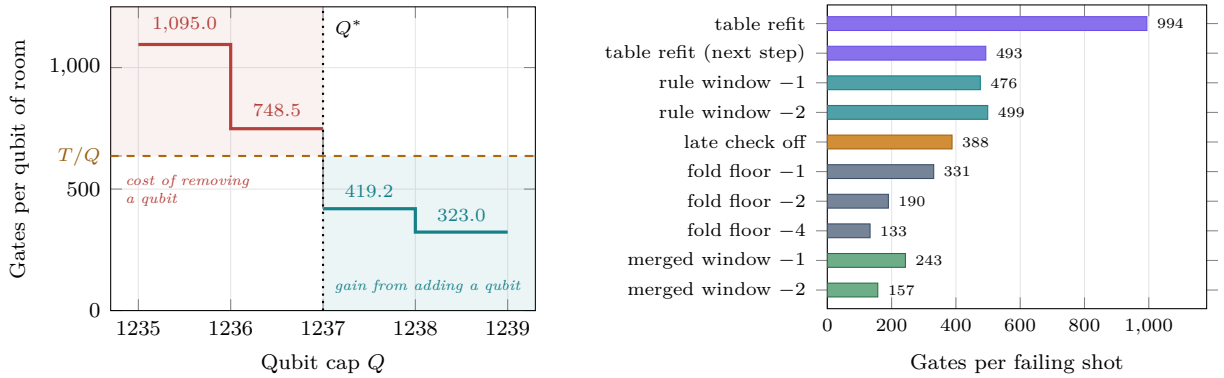


Figure 10: Two measured parts of the price map. Left: the marginal price of a qubit of working room on entry E4, against the objective rate $T/Q = 636$ (dashed). Left of Q^* (red) removing a qubit costs more gates than it is worth; right of it (teal) adding one gains fewer than it costs. The cap sits where the measured price crosses the rate (Corollary 1). Right: what one failing shot sells for on each knob, measured on identical inputs; colours group the knobs of one mechanism. The best knob pays about twice the next.

A discrete optimum. On E4, $T = 786{,}948$ and $Q = 1{,}237$, so the thresholds of Corollary 1 are $T/(Q+1) = 635.7$ and $T/(Q-1) = 636.7$. A sweep of the qubit cap measured $g^+ = 419.2$ and $c^- = 748.5$, with the next units at 323.0 and 1,095.0 (Figure 10, left). Both conditions hold, and the marginal prices are monotone, so the cap is optimal along this knob: $J(Q-1) = 973{,}592{,}874$ and $J(Q+1) = 973{,}722{,}654$, against $J(Q) = 973{,}454{,}676$.

The result is conditional on the other knobs. A later entry by another entrant moved the cap together with a second setting that draws on the same room and gained 0.015%. First-order prices do not see cross terms, and settings that share a resource must be swept jointly. Three days later the loop moved the cap itself, one qubit down, with two settings refitted to the new limit (Section 3.7): a limit is optimal for the prices it was fitted to.

A price list. Ten settings of five knobs each trade failing shots for gates. Each was built and run against a fixed reference on the same 160 packets (Proposition 5). The rates range from 133 to about 1,000 gates per failing shot (Figure 10, right). The other entries of that night had sold on knobs paying about 480 to 530. By Corollary 2, buying those sales back and selling on the best knob is a gain at an unchanged failure rate.

Does the price map predict? Table 32 compares the sum of the separately measured trades with the package measured as a whole. The gates are additive to within 4.2% (−371 predicted, −356 measured). The failing shots are not: the package has 0.59 more than the sum of the rows, against a standard error of 0.18 for the package and 0.03 to 0.25 for each row. Settings share failing shots, so their effects on λ interact.

The practical rule follows: prices choose the package, and the package is then measured as a package. The measured package went under the incumbent's score at no more failures, and was accepted as E5. The merge inside E6 behaved the same way: the plain sum of its two parts predicted 4,847 gates and the merged build gave 4,770, a shortfall of 1.6%.

E.2 The paired estimator

With $\lambda \approx 19.4$ failing shots a packet, two independent samples of n packets compare two circuits with a standard error of $\sqrt{2\lambda/n}$. Table 26 sets the measured paired standard errors against it; by Proposition 5 each gives the discordant shots κ a packet and the factor $2\lambda/\kappa$ by which pairing enlarges the sample.

Table 26: Paired against independent samples. Standard errors are in failing shots a packet.

Comparison	Packets n	Paired	Independent	κ	Factor
E5 against the incumbent	320	0.18	0.35	10.4	3.7×
E6 against the entry before it, on that entry's packets	240	0.16	0.40	6.5	5.9×
A single knob, at its lowest	160	0.03	0.49	0.14	about 270×

E6 changes far more of the circuit than E5 did, and pairing still helped; it is what resolved a difference of 0.34 failing shots. For single-knob comparisons two circuits differ in few shots. The differences the loop had to resolve when it priced the settings were 0.2 to 1.0 failing shots, against an unpaired noise of 0.49 at 160 packets.

The remark after Proposition 5 was also met in practice. A repair that costs $g = 289$ gates was first counted on the shots it fixes alone, read as 0.88 a packet, and priced at 328 gates per shot. The paired count gave $a = 0.830 \pm 0.069$ fixed and $b = 0.284 \pm 0.040$ newly broken: net 0.546, a true price of 529, higher by a factor 1.6.

E.3 The failure budget and the cost of search

An entry needs a packet on which no shot fails. Failing shots per packet were found to be Poisson (Table 27), so the chance of a clean packet is $e^{-\lambda}$. The number of tries to a clean packet is geometric with mean e^{λ}, so

$$\frac{\partial \ln \mathbb{E}[N]}{\partial \lambda} = 1, \tag{11}$$

that is, each failing shot sold multiplies the expected search by e.

Table 27: Checks of the failure model, and what it predicts for the search.

Check	Data	Result
Variance over mean of failing shots	21,062 packets	0.997 ± 0.010
the same, on E5	1,793 packets	0.99
Clean packets against $e^{-\lambda}$, safer variant ($\lambda = 6.489$)	252,733 packets	359 clean, 384 expected (ratio 0.93 ± 0.05)
Expected search for E5	$\lambda = 19.37 \pm 0.10$	$e^{\lambda} = 2.6 \times 10^8$ tries
Expected search for E6, before its search	$\lambda = 18.88 \pm 0.28$	$e^{\lambda} = 1.6 \times 10^8$ tries

This gives the price map a third column: a knob that sells a failing shot for g gates also costs a factor 2.7 in search time, and an entry is worth little if a rival's entry lands before the search ends.

Search lengths. The first eleven searches that the loop ran to a find ended early, which under the geometric model is unlikely; the pattern did not last (Table 28). Part of the early pattern is selection, since searches that did not finish early were abandoned when the benchmark moved. The direct counts of Table 27 support the Poisson model, and failures were found not to cluster. Searches are planned from e^{λ}, and the number of tries is not used as evidence about the failure rate.

Table 28: Search lengths, as a share of the tries forecast from e^{λ}. The p-values combine the searches of a row by Fisher's method under the geometric model.

Searches	Ended at, % of forecast	p
The first eleven run to a find	3% to 35%	0.002
The six after them, run to a find	91, 19, 30, 37, 118 and 104	0.51
Two stopped without a find	71 and 56	
The run's last two	6 and 320	

Switching. Proposition 8 was used three times (Table 29).

Table 29: The three searches moved to a better design.

Moved after	Because	Then
154 million tries without a find	A better design had cleared its gates	The better design was accepted as E10
37.5 million tries, twenty minutes in	A fourth change cleared its gates	The find came 32 minutes later, as E13
34 million tries	The design with the second offset was ready	It became E15

Twice in the same period a cleared design was overtaken because it had waited: once behind a search of 2.4 hours on one machine, and once because the watcher looked at the machine and not at the benchmark, so that a fired falsifier went unseen for 21 minutes. The rules of Section C.3 date from those two losses.

E.4 Forecasts

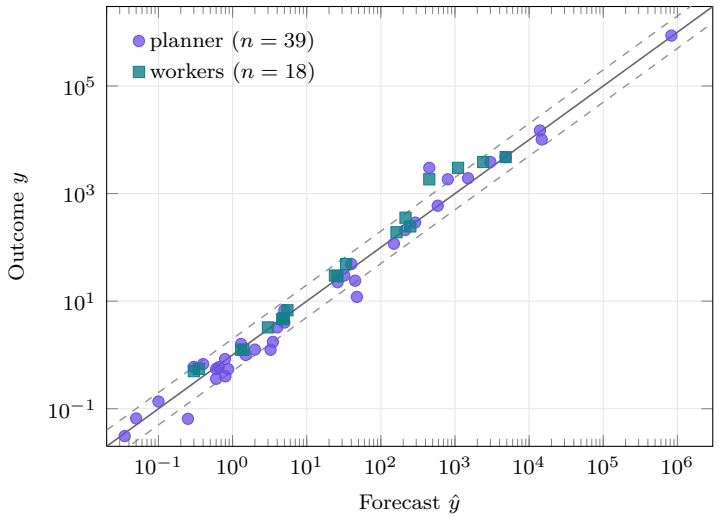


Figure 11: Forecasts written before the run, against outcomes. Both axes are logarithmic. The diagonal is a perfect forecast; the dashed lines are a factor of two. Points below the diagonal are outcomes smaller than forecast; points above it are outcomes larger than forecast.

The ledger of the six rounds up to hour 50 holds 37 experiments with a forecast and a falsifier written before the run. From them we extracted every numeric forecast with a positive numeric outcome: 39 by the planner and 18 by workers (Appendix G). Ranges are entered at their midpoint. Search lengths, whose outcome is a random draw, are treated above.

What the ledger shows about the planner. Over all 39 forecasts the planner has no detectable bias: a mean log error of -0.08, with an interval that covers zero, and a typical error of a factor $e^{0.39} = 1.5$ (Table 33, Figure 11). The average hides two opposite errors. On costs, sizes and rates the planner forecast high: outcomes were $e^{-0.26} = 0.77$ of its forecasts, and the interval excludes zero. On gains it tended to forecast low, though 15 forecasts do not settle that.

A planner that expects things to cost more than they do is a cautious planner, and a cautious planner under-builds. The run shows the consequence: the idea behind E6 was parked on a cost that was too high.

What it shows about the workers. Workers' outcomes exceeded their forecasts by a factor of $e^{0.29} = 1.3$ on average, and the interval excludes zero. On the six gains they forecast, outcomes were 1.9 times the forecast. Workers find more than they expect to.

Neither role is the better forecaster. On 13 quantities the planner and a worker both forecast the same number. The worker was closer in 7 (sign test, $p = 0.50$; Table 30).

Table 30: The two roles compared. Matched pairs are quantities that both the planner and a worker forecast; stated chances are scored against what happened.

	Planner	Worker	Both
13 *matched pairs*			
Closer to the outcome	6	7	
Mean absolute log error	0.41	0.34	0.36[a]
Both forecasts on the same side of the outcome			10
Stated chances			
Forecasts	9	6	
Brier score (a coin scores 0.25)	0.30	0.22	

[a] The geometric mean of the two forecasts.

An earlier version of this analysis, on the first seven pairs, found the worker closer in six ($p = 0.0625$) and read the second forecast as the better one. The six pairs added by the two rounds behind E6 went the other way: the planner was closer in five. We report both, and no longer make the claim.

The two roles err together. In 10 of the 13 pairs both forecasts fell on the same side of the outcome. This is the correlation $r > 0$ of Proposition 3, and it limits what the gap can show. For one follow-up of E6 the forecasts were 450 and 1,100 gates. The gap guarantees that one of them is off by at least $\sqrt{2.4} = 1.6$. The outcome was 3,007: they were off by factors of 6.7 and 2.7. In the one case with the widest gap, a planner's forecast of a speed-up of 3.25 against a worker's 1.3, the bound was closer to the truth: at least 1.6, and the outcome of 1.25 put the planner off by 2.6.

The gap flags a disagreement. It does not flag a blind spot that both roles share. What the second forecast gives for certain is the reading of Table 33: a bias for each role, by kind of quantity, that the planner can apply to its next forecast.

Stated chances. Nine forecasts by the planner and six by workers carried a probability. The planner's Brier score is worse than a coin's (Table 30): the four chances it stated in the two rounds behind E6, between 30% and 50%, all came true. The workers' is better. Fifteen forecasts cannot calibrate anyone, but they agree with the picture above: both roles expected too little.

After the first fifty hours. The later rounds agree with the readings above on the point that matters most for planning. The first build of a new form was forecast low each time: 797.8 gates where a gain of 150 or more had been given 12%; 1,919.5 where a model had said 1,455; 162.5 where the paper price was 129. Extensions of a design that had just paid were kept five times in five, the first four at 2 to 4 times their forecast and the fifth at 1.7 times. Of 17 ideas priced before a build by hour 61, 13 ended as a no-go.

E.5 The record by technique as a forecaster

At hour 72 the ledger held 90 trials that had been scored as paid or not; 44 had paid. Each was given odds in three ways and the odds were scored against the outcome, always using only the trials before it (Table 34).

The record's own running rates beat a coin by a wide margin (Brier scores of 0.149 by stage and 0.201 by family, against 0.250) and matched or beat an external decision model that was shown

the same record (0.206; by family the two are level). The model ranked trials best (0.839) and committed little: its odds ran from 0.17 to 0.62. Two cautions apply. The stage a trial reached is partly an outcome, and the text of each trial was written after it ended. The table supports the narrower claim that the loop uses: a family's record is a better prior than no prior, and it is cheap. It is not a rule for what to cut. Ranked by the record's odds, the 30 least likely of the 90 trials hold 27 that did not pay and three that the record counts as paid: the paper and model pilots behind the largest parts of E4 and E6.

E.6 Instruments against idea lists

On the fifth day of the run every source of cards was in use at once, on one design, so their yields can be set side by side (Table 35).

What reached the benchmark came from reading the design. Of the gates accepted that day, 162.5 came from a ledger of two stages that do the same work, 85.0 from exact cards ported from earlier rounds and a net 53.2 from pricing the qubit limit again. The one idea that was built, of 35 that two agents wrote in dialogue, gave 56.0; a census, a scout on other forms of the large blocks and four designers from a blank sheet gave none. The ledger and the limit are both measurements that end in a number for each block. The one idea from the dialogue that was built was also a reading, of wires that are written, read by nothing and reset; used again as a census, the same lens found only safety margins. A lens works once.

The ledger also answers why every step of that day was small. Additions are 56.3% of the design's gates and shifts 28.4%; the day's three finds all came from the other 15.3%. A scout priced seven other forms of the two large blocks on an exact integer model. Four were worse by 4.7% to 36.7% or did not fit, and one was level. Two looked better by 0.1% to 0.6% under an ideal rule for choosing each step, and the better of them came out 0.25% worse under a rule the circuit can afford. Each large block is at the floor of its present form, and the number of blocks is set by a design choice that sits at a flat optimum.

E.7 Errata, closures, the red team and the breaker

The remaining mechanisms act on single cases, so we report them by count; Appendix F.6 gives the full list. Every build report of the rounds behind E4 and E6 corrected its brief, with three to ten items each. Four falsifiers fired in the first fifty hours, and each ended its line within the hour it was measured. The red team attacked eleven closures: every conclusion held, two of the planner's reasons were wrong, and two parts of E4 began as its side ideas. Four closures were later overturned by a build, the qubit limit among them; all four had rested on a paper price or a model, and none that rested on a proof or an exhaustive search has been overturned. Up to the cut-off of this paper's counts, no package sent to the benchmark was broken by a breaker in its final form; a breaker withdrew the first form of E16's change six minutes into its search, and E17's verdict was read on the net over inputs that nobody picked (Table 25). After the cut-off a breaker broke a second change of E14's form and E14 itself (Section 3.9). Breakers also found the incumbent's own error behind E11, a comparison in which a builder had compared a new form with itself, and a change of failure pattern behind E12 that the builders' averages had hidden. An aligned pair showed that three changes proposed as exact trade failures; the first had passed "no output differs" on thirty packets. None was sent.

F Detailed Results

The tables of Section 3 are collected here, in the order in which the text uses them.

F.1 The Stages of the Case Study

Table 31: The stages of E6. Each row is one trial: the instrument, the number in question, the two forecasts written before the run, and the outcome. A dash means no forecast was written by that role.

Stage	Instrument	Number in question	Planner	Worker	Outcome
Reopening question	Integer identity, 4.4 million inputs	Gates to erase three leftover wires	35 to 60	0	0
Count	Gate-level model	Wires added at the peak	32	24	24 and 36
One iteration	Circuit builder	Gates saved a pass	40	34	49.0
First half	Circuit builder	Gates saved	3,000	2,400	3,859
Smaller core	Model checked on every input, then builder	Gates saved	450	1,100	3,007
More iterations	Circuit builder; stress inputs	Gates saved	800	450	1,841
Package	Three known answers, then builder	Gates saved by the merge	4,817	4,827	4,770
Safety	Paired test, 240 packets	Failing shots against the incumbent	−0.65	–	-0.59 ± 0.16
Search	Filter, then GPU and CPU	Tries to a passing packet	158 million	–	53 million
Exact verifier	Official judge	Score	pass	–	957,294,360

F.2 The Price Map's Prediction for One Package

Table 32: The E5 package. The price map's first-order prediction against the measured package.

Trade	Gates	Failing shots	Rate (gates per shot)
Buy back: decision window one bit wider	+438	−0.92	476
Buy back: merged window one unit wider	+325	−0.61	533
Sell: safety table refitted at a looser budget	−954	+0.78	1,223
Room re-allotted cell by cell	−180	0	–
Sum of the rows (prediction)	−371	−0.75	
Measured as one package, 320 packets	−356	-0.16 ± 0.18	

F.3 Forecast Accuracy

Table 33: Forecast accuracy by role and by kind of quantity. Log error is $e = \ln(y/\hat{y})$; the interval is a 95% bootstrap interval. The split by kind was made after the data were seen.

Role and kind	n	Mean e (bias)	95% interval	Mean $\lvert e \rvert$	Within ×1.25	Within ×2
Planner, all	39	−0.08	[−0.26, +0.10]	0.39	46%	87%
costs, sizes and rates	24	−0.26	[−0.42, −0.11]	0.31	58%	92%
gains	15	+0.19	[−0.15, +0.55]	0.51	27%	80%
Workers, all	18	+0.29	[+0.14, +0.48]	0.32	61%	89%
costs, sizes and rates	12	+0.13	[+0.03, +0.24]	0.16	83%	100%
gains	6	+0.62	[+0.28, +1.01]	0.63	17%	67%

F.4 The Record by Technique as a Forecaster

Table 34: Odds that a trial pays, scored on 90 trials. Lower Brier scores are better; the ranking column is the chance that a paid trial was given higher odds than an unpaid one.

Forecaster	Brier	Ranking
Always 50%	0.250	0.500
The record: pay rate of the trial's technique family so far	0.201	0.751
The record: pay rate of the trial's stage so far	0.149	0.828
An external decision model, given the trial's text	0.243	0.684
The same model, given the record so far	0.206	0.839

F.5 Where One Day's Accepted Gates Came From

Table 35: Where one day's accepted gates came from. "Accepted" is what reached the benchmark in an entry.

Source of the cards	Cards	Built	Gates accepted
Two agents in dialogue, seven turns	35 ideas	1	56.0
A census with the lens of that one idea	6,148 gates listed	0	0
A ledger of two stages that do the same work	2 lines of size	1	162.5
The qubit limit priced again, settings refitted	30 builds	1	53.2 net
Exact cards left from earlier rounds, ported	3	3	85.0
A scout on other forms of the two largest blocks	7 forms	0	0
Four designers from a blank sheet	4 designs, 17 ideas screened	0	0

F.6 What the Remaining Mechanisms Did, by Count

Table 36: What the remaining mechanisms did, by count. Rows marked "first 50 hours" were counted at that point and not again.

Mechanism	Observed in the run
Falsifiers (first 50 hours)	Three fired in one round alone. Each ended its line of work within the hour it was measured; none was disputed. One more fired in a later round and parked its line. In the case study two clauses came out exactly on their limits and did not fire
Errata on the brief (first 50 hours)	Every build report of the round that produced E4, every concept report two rounds later and all seven reports of the two rounds that produced E6 corrected their briefs, with three to ten items each. Four corrections were to the target or to the falsifier itself
Red team (first 50 hours)	Eleven closures attacked. Every conclusion held; two of the planner's reasons were wrong or incomplete and three of its constants were off. Two parts of E4 began as the red team's side ideas
Evidence grades	Four closures were later overturned by a build, the qubit limit among them. All four had rested on a paper price or a model. None resting on a proof or an exhaustive search was overturned
Reopening conditions	A parked idea's numeric condition was met by a worker, against the planner's forecast. The idea was reopened and became E6, the largest step of the run (Section 3.6)
Planner rotation	Used once, before E4. The rebuilt planner changed one rule, and the next round produced E4
Instrument admission	Introduced after a fitting tool was found to understate every candidate by 0.30% of J; it had never been run on the incumbent. In the case study a merge was admitted only after it rebuilt three known answers byte for byte
Published errors	67 planner mistakes logged, each with the rule that changed
Breaker	Five packages examined from hour 50 to the cut-off, none broken in its final form. The first form of E16's change was withdrawn six minutes into a breaker's search. Behind E17 one aimed class leaned against the change and its mirror class leaned for it, and the verdict was read on inputs that nobody picked (Section 3.7). After the cut-off a breaker broke E14 and a second change of its form (Section 3.9). The breaker also found an error of the incumbent behind E11, a comparison in which a builder had compared a new form with itself, and a change of failure pattern behind E12 that the builders' averages had hidden
Aligned and control pairs	Three changes proposed as exact were shown to trade failures and were not sent; the first had passed "no output differs" on thirty packets. A control pair set the bar for one rewrite of a whole stage (12 changes of verdict against the control's 13 on 300 packets)
List by structure	Used once, on E15: 22 added approximate comparisons found in a design that two paired tests had passed
Clean-sheet round	Used once: four designs, all lost; a proof, a cause and a floor kept

G The Forecast Set

Every numeric forecast with a positive numeric outcome in ledger rows m035 to m071 (the experiments with a forecast and a falsifier written before the run). P is the planner, W a worker. Ranges are entered at their midpoint, unless the forecaster named a point. An outcome measured in both directions of the circuit is entered at its mean. A forecast of the whole design's gate count is entered as the change against the study's base, so that a level of several hundred thousand gates does not hide the error. Quantities are named in general terms.

Table 37: The forecast set. 57 numeric forecasts: 39 by the planner (P) and 18 by workers (W).

Row	Role	Quantity	Forecast	Outcome	$\ln(y/\hat{y})$
m035	P	gates of a test, per iteration	45	24	−0.63
m035	P	iterations gained	1.5	1	−0.41
m035	P	gain on paper (%)	0.3	0.6	+0.69
m037	P	gates saved	1,500	1,928	+0.25
m037	P	failing shots added	1.3	1.6	+0.21
m037	P	net gain (%)	0.25	0.065	−1.35
m039	P	gates for a second qubit	150	117	−0.25
m039	P	its net gain (%)	0.05	0.066	+0.28
m039	P	qubits freed	5	4	−0.22
m039	P	its net gain (%)	0.1	0.136	+0.31
m040	P	gain of a decision rule (%)	0.4	0.675	+0.52
m041	P	gates of a wide shift	585	594.6	+0.02
m041	P	gain of an alternative (%)	0.6	0.36	−0.51
m042	P	gain of a package (%)	0.79	0.838	+0.06
m043	P	gain of a joint retune (%)	0.035	0.031	−0.12
m048	P	cost ratio of another design	2	1.255	−0.47
m050	P	bits per symbol	4.9	4.84	−0.01
m050	P	bits given the last symbol	4.7	4.64	−0.01
m050	P	bits readable from the state	0.8	0.4	−0.69
m052	P	compares, % of gates	4	3.25	−0.21
m052	P	folds, % of gates	5	6.78	+0.30
m053	P	failing shots bought	0.87	0.545	−0.47
m053	P	gates paid	290	289	−0.00
m053	P	spread of rates (times)	3.5	1.75	−0.69
m054	P	speed-up of a search step (times)	3.25	1.25	−0.96
m055	P	search rate on a GPU (per s)	14,800	10,150	−0.38
m061	P	gates of an uncomputation	47.5	12	−1.38
m062	P	iterations kept by a predicate	0.6	0.55	−0.09
m063	P	wires added to a block, at its peak	32	30	−0.06
m064	P	gates saved per pass of a new operation	40	49	+0.20
m064	P	gates saved by half of a build	3,000	3,859	+0.25
m065	P	ANDs of a core block	26	22.5	−0.14
m065	P	gates saved by exact rewrites	450	3,006.8	+1.90
m066	P	gates saved by an extension	800	1,840.5	+0.83
m067	P	gates saved by a merged package	4,816.5	4,770.3	−0.01
m068	P	gates from a rival setting	213	210	−0.01

Table 37, continued

Row	Role	Quantity	Forecast	Outcome	$\ln(y/\hat{y})$
m068	P	failing shots under the incumbent	0.65	0.592	−0.09
m071	P	search rate, GPU and CPU (per s)	14,000	14,874	+0.06
m071	P	tries per kept try of a filter	830,000	864,000	+0.04
m048	W	cost ratio of another design	1.4	1.255	−0.11
m050	W	bits per symbol	4.87	4.84	−0.01
m050	W	bits given the last symbol	4.68	4.64	−0.01
m050	W	iterations of a 4-wire rule	162.5	191	+0.16
m052	W	compares, % of gates	3	3.25	+0.08
m052	W	folds, % of gates	5.5	6.78	+0.21
m054	W	speed-up of a search step (times)	1.3	1.25	−0.04
m055	W	changed lines of a port	250	245	−0.02
m055	W	kept-but-failing shots	0.3	0.5	+0.51
m062	W	iterations kept by a predicate	0.35	0.55	+0.45
F104	W	gates gained by moving failures	215	356	+0.50
m063	W	wires added to a block, at its peak	24	30	+0.22
m063	W	ANDs of a core block	26	29	+0.11
m064	W	gates saved per pass of a new operation	34	49	+0.37
m064	W	gates saved by half of a build	2,400	3,859	+0.47
m065	W	gates saved by exact rewrites	1,100	3,006.8	+1.01
m066	W	gates saved by an extension	450	1,840.5	+1.41
m067	W	gates saved by a merged package	4,826.5	4,770.3	−0.01