Restart Economics for AI Agents

Cutoffs, Retries, and Cost per Resolved Task

James E. Murphy

ORCID 0009-0001-2217-4306

Working paper for discussion · v0.1 · 28 September 2026

jamesemurphy.com/ai-economics/restart-economics

Contents

Abstract 3

1. Introduction 3

2. Related Work 5

3. The Decision and the Estimand 6
- 3.1 The operator's problem . . . 6
- 3.2 Policy value . . . 6
- 3.3 The outside option . . . 7
- 3.4 The verifier . . . 7
- 3.5 What resolution means . . . 7

4. Data 8
- 4.1 The release and the configurations . . . 8
- 4.2 The attempt pool . . . 8
- 4.3 Prices . . . 9

5. Identification and Estimation 9
- 5.1 Replay . . . 9
- 5.2 Enumeration . . . 10
- 5.3 Cross-fitting and the full-pipeline bootstrap . . . 10
- 5.4 The transfer test . . . 11

6. Policies 11

7. Results 12
- 7.1 Retrying . . . 12
- 7.2 The cap given retry . . . 13
- 7.3 Schedules . . . 16
- 7.4 Execution state and the primary transfer . . . 16
- 7.5 Switching configurations . . . 19
- 7.6 Sensitivities . . . 20

8. Interpretation 21
- 8.1 Why a cap does not pay when the fallback is a person . . . 21
- 8.2 A rule of thumb in attempts . . . 22
- 8.3 Under review, whether to run the agent . . . 23
- 8.4 What state-aware stopping would need . . . 23
- 8.5 Choosing the configuration and choosing the policy . . . 24

9. Boundaries and Limitations 24

10. Conclusion 25

Appendix A. The Registered Sensitivities 26

Appendix B. Comparators and the Sample Oracle 31

Appendix C. Difficulty, the Distribution of Cost, and the Spread 35

Appendix D. Diagnostics 38

Appendix E. Verification 41

Appendix F. The Plan and the Choices It Left Open 42

Appendix G. The Ladder under Review at 0.1 and 0.3, and Every Crossing 44

Data, Code and Preregistration 47

References 47

Declarations 48

Declaration of generative AI and AI-assisted technologies in the writing process 49

Abstract

Repeated runs of the same AI coding agent on the same task differ widely in what they cost and whether they succeed, so an operator runs a policy rather than a single attempt: cap each attempt, retry, and hand the task to an engineer when the attempts are spent. Most work on stopping agents early asks how much compute a stopping rule saves for a given loss in success. This paper asks the operator's question instead: once every lost success has to be bought from a fallback that costs money, does stopping still save anything? It replays four logged OpenHands runs on each of the 500 SWE-bench Verified tasks for seven model configurations, 13,681 attempts in all, and scores every policy on expected cost per task, which charges agent spend, verification of what the agent submits, and a task-specific outside option priced from the benchmark's annotated engineer time at a swept wage. Because the outside option resolves every task, no policy can look cheap by abandoning hard ones. Policies are chosen on training folds and scored on held-out folds, under an analysis plan frozen and registered before any result existed.

Three things follow. When verification is automated and the fallback is an engineer, retrying is the lever: at $100 to $300 an hour it cuts expected cost by 13 to 21 percent for six configurations and 4 to 8 percent for the seventh, and raises the share of tasks the agent resolves itself from 58 to 69 percent to 69 to 78 percent. Choosing a cap on attempt length from these data adds nothing at any human wage, because attempts that have run long still succeed a third to a half of the time, and each success cut off costs tens of dollars against the few dollars of tokens a cutoff saves; a cap pays only while the fallback is worth fewer than two to four attempts, which is a few dollars an hour here, and in an unregistered check even the best cap chosen with hindsight saves at most three cents per task at $100 an hour. When each submitted patch must be reviewed at a large share of the fallback's cost, the economics reverse: retrying stops paying, the cap saves in point estimate at every wage, distinctly from zero below about $10 to $50 an hour, by keeping long attempts out of review, and, in an unregistered comparison, the best capped policy costs about what sending every task straight to the engineer costs, so the question becomes whether to run the agent at all. The registered low-complexity state rule adds little beyond a fixed cap, and switching configurations after a failure saves about 9 percent in point estimate, with an interval that includes zero.

Keywords: AI agents; unit economics; optimal stopping; retry policies; cost per resolved task; SWE-bench; preregistration.

JEL: C44, D24, L86, M15, O33.

1. Introduction

An AI coding agent given the same task twice does not do the same thing twice. Bai et al. (2026) report that runs of one agent on one task can differ by up to 30 times in total tokens, with the costliest run typically about twice the cheapest, and that some model configurations spend over 1.5 million more tokens per task than others on the same tasks. An attempt that is going badly may run to the harness's iteration limit and fail; a fresh attempt on the same task may finish quickly and succeed. Operators respond with policies. They set iteration limits, allow a second or third attempt, and route a task to a person when the agent has had its chances.

Which of these policies pay depends on a trade that pass rates and average token counts do not capture. A cap on an attempt saves the spend of attempts that would have run long, but it also cuts off some that would have succeeded, and each success lost must be bought from the fallback instead. If the fallback is an engineer, a lost success costs tens of dollars; the tokens a cap saves cost cents. The same arithmetic

runs through retries and through switching to another model after a failure. Whether a policy helps is therefore a question about expected cost per task, with the fallback priced in.

This paper answers that question on logged data. It uses the release accompanying Bai et al. (2026): four OpenHands runs on all 500 SWE-bench Verified tasks for each of seven model configurations. Any policy that stops attempts at or before the point where the harness stopped them can be evaluated exactly on these logs, by truncating the logged trajectories, with no model of what the agent would have done. The paper scores every policy on one number: agent spend, plus verification of what the agent submits, plus an outside option paid when the attempts are exhausted, priced from the benchmark's annotation of how long an engineer would take at a wage that is swept from $5 to $300 an hour. Because the outside option resolves every task, a policy cannot improve its score by giving up on hard tasks, and cost per incoming task is also cost per resolved task.

The policies form a ladder: one attempt; several attempts; several attempts each capped at a constant number of calls; several attempts under a schedule of caps that varies by attempt; a rule that reads the running attempt's state and decides when to restart; and schedules that switch configurations between attempts. Each is chosen on training folds and scored on held-out folds, and the intervals come from a bootstrap that repeats the choice inside every replicate. The analysis plan, including the primary comparison, the policy classes, the sensitivities and the expected direction of the primary result, was frozen and registered on OSF before any policy was evaluated.

The primary result is the value of the cap given retry: what an operator gains by choosing a cap on attempt length from about 400 tasks of history, over retrying without one. With an automated verifier, as in the benchmark's own test suite, retrying pays at every wage from $50 an hour up, and choosing a cap adds nothing measurable once the outside option exceeds two to four times what a median attempt costs. That threshold is a few dollars an hour here. In an observation that was not registered, it is also close to the point at which retrying without a cap first beats escalating every task without running the agent, which suggests that below it the cap's saving is mostly the saving of not running the agent. Above it, the cap's margin is within a few percent of what retrying costs, mostly on the cost side in point estimate, its interval straddles zero everywhere, and nine sensitivities leave that where it is. The gain available to any cap is small: in an unregistered check, even the best constant cap chosen with hindsight on all of a configuration's tasks saves at most three cents per task at $100 an hour. With human review priced at half the outside option, retrying stops paying, the cap saves in point estimate at every wage, and, in a comparison the plan did not register, the best capped policy costs about what escalating everything costs.

The paper also tests whether a low-complexity rule that reads an execution's state can do better than a fixed cap. The registered rule, a plug-in restart rule with eight coefficients fitted on the other six configurations, costs what retrying without a cap costs when the verifier is automated, because it almost never stops an attempt, and its measured gain over the best schedule is the schedule's own cost of being chosen from data. On synthetic attempts where the optimal rule can be computed exactly, the plug-in rule stops too many attempts too early. Switching configurations after a failure lowers cost in point estimate by about 9 percent with an automated verifier, but the interval includes zero, and outcomes across configurations are correlated at a median of 0.81, which limits what a portfolio can do.

The paper's first contribution is the question. Work on stopping agents early asks how much compute a stopping rule saves for a given loss in success, and reports tokens saved against resolution lost. Once a lost success has to be replaced by something that costs money, the trade has a price, and the question becomes whether stopping saves anything at all. On these logs it mostly does not, while retrying does, because another attempt costing about a dollar can avert a fallback costing tens of dollars. The practical problem

is then not how aggressively to stop bad runs but what another attempt, a verification and an escalation cost relative to one another. The paper's other contributions serve that question: a finite, priced estimand that needs no extrapolation beyond four attempts and does not reward abandonment; an exact evaluation of stopping and retry policies from logged runs, with selection handled by cross-fitting and a full-pipeline bootstrap; preregistered evidence on caps, retries, schedules, state-aware stopping and switching across seven configurations, with a break-even in units that transfer across configurations whose attempts differ fivefold in cost; and the evaluator, the results and the plan, released.

Section 2 places the paper in the literature. Section 3 defines the decision and the estimand, Section 4 describes the data, Section 5 the identification and estimation, and Section 6 the policies. Section 7 reports the results and Section 8 interprets them. Section 9 states the boundaries of the evidence and Section 10 concludes. The appendices report the registered sensitivities, comparators and diagnostics.

2. Related Work

Restarts and portfolios. Restarting a randomized procedure is an old idea in computer science. For a Las Vegas algorithm whose run time distribution is unknown, Luby, Sinclair and Zuckerman (1993) give a universal restart schedule within a logarithmic factor of the best fixed cutoff. Gomes, Selman, Crato and Kautz (2000) show that heavy-tailed run times in combinatorial search make restarts valuable, and Huberman, Lukose and Hogg (1997) and Gomes and Selman (2001) treat the choice among randomized solvers as a portfolio whose value depends on the correlation of their outcomes. Rice (1976) frames the choice of algorithm per instance. This paper borrows the portfolio view but not the guarantee: the classical results assume a procedure that eventually succeeds if restarted often enough and a cost measured in time alone, while an agent may never solve a task, each attempt is paid for, and the operator has an outside option.

Cost of language model inference. Work on cascades and routing chooses which model answers a query to trade cost against quality (Chen, Zaharia and Zou 2024). Repeated sampling raises the chance that some attempt succeeds (Chen et al. 2021; Brown et al. 2024), usually with a verifier assumed to pick the success. Kapoor et al. (2025) argue that agents should be evaluated jointly on accuracy and cost. Bai et al. (2026) measure how agents spend tokens on SWE-bench Verified, and release the trajectories used here. This paper takes the operator's side of the same data: it prices the outside option and the verifier, and asks which stopping and retry policies minimize expected cost per task.

Stopping software agents early. Several recent papers train monitors that predict an agent's failure from a partial trajectory and stop or restart it. Wang et al. (2026) predict failure from partial trajectories on SWE-bench Verified, report the tokens saved at a fixed false-positive rate, and propose a restart that does better than a cold restart. Guo et al. (2026) terminate attempts early using experience from earlier tasks, and Lin et al. (2026) and Liu et al. (2025) study agents' awareness of their own budgets. These papers change the agent or add a learned monitor, and report tokens saved against resolution lost. This paper holds the agent fixed, evaluates operator policies by replay, prices the resolution lost at the cost of the fallback, and restricts state-aware stopping to an eight-coefficient rule registered in advance. Its finding that the state adds little under this restriction is a statement about low-complexity rules on these logs, not about learned monitors.

What the benchmark measures. SWE-bench (Jimenez et al. 2024) and its human-validated subset, SWE-bench Verified (Chowdhury et al. 2024), score an attempt by whether the repository's tests pass. Test passing is not the same as a mergeable fix. Whitfill, Wu, Becker and Rush (2026) had maintainers

review agent pull requests that the grader scored as correct and found that roughly half would not be merged. He et al. (2026) find, on a different benchmark, that about a third of test-passing repairs violate constraints drawn from real code review, on synthesized tasks that state those constraints to the agent. OpenAI (2026) argues that contamination and test design now limit SWE-bench Verified as a measure of frontier capability. Kwa et al. (2025) find that the benchmark's annotated times understate how long engineers take on the shortest tasks. Each of these bears on the estimand here. The first two and the last enter as registered sensitivities, and the third as a boundary on the evidence (Section 9).

Estimation. Choosing a policy from data and scoring it on the same data overstates its value. The paper uses cross-fitting (Chernozhukov et al. 2018) to separate choice from scoring, and a low-complexity threshold rule chosen by empirical welfare maximization (Kitagawa and Tetenov 2018) as the comparator to the state-aware rule. Intervals come from a task-level bootstrap that repeats the choice, with a two-stage bootstrap (Davison and Hinkley 1997) as a sensitivity.

3. The Decision and the Estimand

3.1 The operator's problem

A task arrives. The operator may run the agent on it up to $K \leq 4$ times, each run a fresh attempt that starts from the task and not from the previous attempt's state. Each attempt runs under a configuration, a model under a harness, and may be stopped at a cutoff, counted in model calls, before the harness would stop it. An attempt that reaches its own end and produces a patch is verified. If the patch resolves the task the operator stops. If every allowed attempt fails, the operator pays for the task to be resolved another way: the outside option.

3.2 Policy value

For task i and a policy with K attempts, let C_{ij} be what attempt j spends up to its stop, D_{ij} indicate that it ends with a patch to verify, and S_{ij} indicate that it resolves. The cost of the task is

$$L_i = \sum_{j=1}^{K} \Big(\prod_{l<j} (1 - S_{il}) \Big) (C_{ij} + v_i D_{ij} + \phi H_i S_{ij}) + \Big(\prod_{j=1}^{K} (1 - S_{ij}) \Big) H_i,$$

where H_i is the outside option, v_i the cost of verifying a patch and ϕ the share of accepted patches that are false accepts, zero in the primary analysis. The value of the policy is the mean of L_i over incoming tasks, in expectation over which of the task's runs the attempts draw. Every task is resolved, either by an attempt or by the outside option, so the value is also the cost per resolved task, and the share of tasks the agent resolves without the outside option is reported beside it rather than traded against it.

Three properties are deliberate. The estimand is finite: with at most four attempts it needs no model of what a fifth would do, and a task that no attempt resolves costs its attempts plus the outside option rather than an undefined ratio of cost to success probability. Abandonment is never free, because a task given up on is paid for at the outside option. And the outside option is external and task-specific, set by the benchmark's own annotation rather than by the agent's behavior.

3.3 The outside option

SWE-bench Verified annotates each task with the time an engineer with a few hours of familiarity with the codebase would need to fix it, in four buckets: under 15 minutes, 15 minutes to an hour, 1 to 4 hours, and over 4 hours (Chowdhury et al. 2024). Following Kwa et al. (2025, Table 8), each bucket is converted to the geometric mean of its bounds, 3.9, 30, 120 and 480 minutes, with 16 hours as the upper bound of the longest, and H_i is that time at a fully loaded engineer wage. The wage is swept from \$5 to \$300 an hour. The lower end is far below any human wage on purpose. It is not a labor market: it is a low shadow price for guaranteed outside resolution, which might be a cheap automated fallback, and the sweep goes that low, and lower still on the second axis described below, to locate the break-even at which a cap starts to pay. Whatever the wage, the outside option resolves the task; a task deferred or dropped is not resolved and has no place in the estimand.

The tasks are mostly short. On the tasks scored here the mean annotated time is about 30 minutes, so at \$100 an hour the mean outside option is about \$50, against a median attempt cost of \$0.34 to \$1.66 depending on the configuration. Because attempt costs differ fivefold across configurations, the sweep is also run in units of attempt cost: at a multiple M, the wage is set so that the mean outside option equals M times the configuration's median attempt cost. A break-even in these units compares across configurations; the same break-even in dollars does not.

3.4 The verifier

Every result is computed under four verifier regimes, an automated verifier and review at 0.1, 0.3 and 0.5 of the outside option, and the text reads the two that bracket the answer, the automated verifier and review at 0.5, with review at 0.1 and 0.3 in Tables 4, 7, A1, A5, A8, A10 and A11 and in the results files. With an automated verifier, the benchmark's own tests, v_i is taken as zero. With a human reviewer, $v_i = fH_i$ for f of 0.1, 0.3 and 0.5: review costs a share of what it would cost the reviewer to fix the task, so it costs more on the tasks whose escalation costs more. Under review a cap earns a second credit, because an attempt cut off before it produces a patch saves a review as well as tokens.

3.5 What resolution means

An attempt resolves when the benchmark's tests accept its patch. That is benchmark resolution, not a correct and mergeable fix. Maintainers reviewing agent pull requests that SWE-bench Verified scores as correct would not merge roughly half of them (Whitfill et al. 2026), and on a different benchmark, whose synthesized tasks state constraints drawn from real code review, about a third of test-passing repairs violate them (He et al. 2026). The registered sensitivity treats a share ϕ of accepted patches as false accepts whose outside option is paid later, at $\phi = 0.34$ and $\phi = 0.50$, the two ends of the range these sources imply, and re-selects every policy with that charge in its objective. A single ϕ assumes the false-accept share does not depend on the policy, which may fail if patches produced after a cutoff-driven retry are more superficial than those from uninterrupted runs. The assumption is stated, not tested.

4. Data

4.1 The release and the configurations

The data are the trajectories released with Bai et al. (2026): four OpenHands runs on all 500 SWE-bench Verified tasks for each of eight models, in nine archives; one run of one configuration lacks one task. The paper treats each archive as a configuration rather than a model, since a run is a model under a harness version, at a date, through an endpoint, under a price schedule. An audit completed before the plan was frozen established what each archive was, and admitted seven. Two were excluded. In the Claude Sonnet 3.7 archive, the harness replaced every attempt that reached its 100-call limit with a fresh one and kept only the later try, so the recorded attempts are a sample from which long attempts were removed. The second Sonnet 4.5 archive runs a different agent under a different harness, with a critic that allows up to three attempts per task and keeps the one it accepts.

All seven included configurations run the same agent, CodeActAgent (Wang et al. 2024), under OpenHands (Wang et al. 2025). Table 1 describes them. Harness versions differ, and are reported with every result. GPT-5 stops at 100 iterations and the other six at 500, and the cap is treated as an attribute of the configuration: each is evaluated within its own cap, and a common 100-call horizon is a sensitivity. The audit found one model call per agent iteration and no history condensation in any configuration, so cutoffs are stated in calls.

Table 1. The seven configurations. Usable attempts are those in the attempt pool (Section 4.2); the resolved share is among them. Tasks with four usable draws are the tasks on which every comparison across attempt budgets is scored. The median attempt is the median cost of a usable attempt run to its own end, on those tasks, at list price. † Qwen3 Coder's price is imputed.

Configuration	Harness, run dates	Cap	Usable attempts	Resolved	Tasks with four draws	Median attempt
GPT-5	0.62.0, Dec 2025 to Jan 2026	100	1,995	59.8%	496	$0.34
GPT-5.2	0.62.0, Jan 2026	500	1,891	65.9%	405	$0.56
Claude Sonnet 4	0.40.0, Jun 2025	500	1,998	68.7%	498	$1.04
Claude Sonnet 4.5	0.62.0, Dec 2025	500	2,000	64.1%	500	$1.66
Gemini 3 Pro Preview	0.62.0, Jan 2026	500	1,893	61.3%	410	$0.82
Kimi K2	1.1.0, Jan 2026	500	1,904	56.7%	412	$0.70
Qwen3 Coder 480B†	0.48.0, Jul 2025	500	2,000	66.1%	500	$1.09

4.2 The attempt pool

How an attempt stopped and whether it resolved are recorded separately, because the harness evaluates whatever patch exists when it stops, including at its own limit. An attempt enters the pool when it is a draw from the configuration's behavior and its outcome was observed. Infrastructure failures (outages, rate limits, relayed provider errors and the harness's eight-hour timeout) and evaluations that returned no verdict are excluded. Finishes, empty patches, patches that did not apply, stops at the iteration limit

or by the harness's loop detector, provider refusals and errors caused by the model's own output are retained and scored by the benchmark's report. The pool holds 13,681 of the 13,999 executions. Three prespecified sensitivities change these rules: excluding refusals, which occur only for GPT-5; counting evaluations without a verdict as failures; and dropping executions the harness re-ran after a crash, which every configuration has, from 12 for GPT-5 to 108 for GPT-5.2.

Excluding attempts leaves some tasks with fewer than four usable draws, mostly in the three configurations that suffered outages: every configuration has at least one usable draw on 499 or 500 tasks and at least two on 498 to 500, but at least three on 486 to 500 and four on 405 to 500 (Table 1; the counts for every budget are in `results/ladder.csv`). Comparisons across attempt budgets are therefore scored on the tasks with four usable draws in every configuration involved, which is each configuration's own subset for the ladder (Table 1) and 275 tasks common to all seven for the cascades. Scoring every policy on every task that can fill it is a registered sensitivity.

4.3 Prices

Cost is reconstructed from token counts under each model developer's published list price on the run's first day, applied call by call over the token classes the developer bills. The logged cost is not the price. For GPT-5 and both Sonnet configurations the published schedule reproduces the logged cost of every call. For GPT-5.2 and Gemini the harness charged cached input at the full input price, and ignored Gemini's long-context tier; for Kimi the logged cost follows no fixed schedule, consistent with a router sending calls to providers at different prices. At the published schedules the six configurations that log cost spent $12,692, against $23,543 logged. Qwen3 Coder was self-hosted and logs no cost, so its schedule is imputed at the developer's hosted price at release, tiered by prompt length, and its dollar figures carry a dagger throughout. The lowest third-party price for the same weights, $0.22 per million input tokens and $1.80 per million output tokens, is a registered sensitivity.

5. Identification and Estimation

5.1 Replay

Every policy in the class stops an attempt at or before the point where the harness stopped it. The outcome of any such policy on a logged attempt is therefore a deterministic function of the log: the attempt's prefix up to the cutoff. An attempt cut off at call t costs what its first t calls cost, is discarded unverified, and resolves nothing. An attempt that reaches its own end costs what it cost, is verified if it produced a patch, and resolves if the tests accepted it. Nothing is imputed and nothing about the agent is modeled, so no ignorability assumption is needed. The harness's own stops are not censoring: continuing past them is outside the policy class, not unidentified within it.

One consequence is a direction, not a bias to be corrected. The harness submits whatever patch is in the workspace when it stops at its limit, and some such patches pass; a policy's cut-off attempt is discarded instead, because the workspace at an earlier call cannot be recovered. With an automated verifier, submitting would weakly dominate discarding, so the measured value of a cutoff is conservative. With a human verifier and a low-yield partial patch, discarding can be cheaper. The two are not ordered in general, and no bound is claimed.

5.2 Enumeration

A task's four runs are treated as exchangeable: any ordering of them is as likely as any other. A policy of K attempts is then evaluated exactly by averaging its cost over every ordered selection of K of the task's usable draws, 24 orderings for four attempts from one configuration. Exchangeability does not require the runs to be independent of one another. A configuration that sometimes repeats a trajectory, as Gemini does on four tasks, is priced as behaving that way, which is what a retry of it would meet. With the policy fixed out of sample, the mean of this estimator over tasks is unbiased for the policy's value under a retry process whose draws on a task have the same joint law as these, on a workload drawn like SWE-bench Verified. Nothing is claimed about any single task's conditional expected cost.

A cascade draws a task's attempts from several configurations, and is evaluated over the product of their orderings: every combination of one configuration's draws with another's, most of which were never jointly observed. That is exact only if, given the task, the draws of different configurations are independent. The assumption is plausible here, since the configurations were run months apart under different harness versions through different endpoints (Table 1), so nothing but the task links a draw of one to a draw of another, and the infrastructure failures that could have linked them are outside the pool. But it is an assumption, not something the logs establish, and the cascade's value rests on it where the single-configuration ladder does not.

5.3 Cross-fitting and the full-pipeline bootstrap

Every quantity chosen from data (the attempt budget, the cutoff, the schedule, the state rule, the cascade) is chosen on four of five task-grouped folds and scored once on the fifth. The split is fixed before any policy is scored. What a cross-fitted comparison estimates is therefore the value of the policy this procedure picks from about 400 training tasks, not the value of the best policy in the class: a class that contains a slightly better policy than the baseline can still lose out of sample, when the cost of picking from it exceeds what the best policy would save. Every margin below is that deployable value. Cross-fitting removes the optimism of scoring a choice on the tasks that made it but does not carry the variability of the choice into an interval, so intervals come from a bootstrap that resamples tasks and repeats the split, the choice and the scoring inside every replicate. Copies of a task drawn more than once in a replicate are kept in the same fold, so that a replicate never scores a choice on a task that made it. Intervals are 95 percent percentile intervals from 1,000 replicates for the primary result, and, for the fitted steps, whose models and choices are refitted inside each replicate at far greater cost, from 1,000 replicates with an automated verifier at $25, $100 and $300 an hour, where the plan reads them; elsewhere the fitted steps are point estimates, and the comparators of Appendix B, the cascade's switching margin and the two-stage transfer have intervals from 100 replicates. No significance test is run. A margin is called distinct from zero only when its 95 percent interval excludes zero; a saving or cost whose interval straddles zero is reported as a point estimate and read as such. Where an interval comes from 100 replicates, its ends rest on a few draws each, and an endpoint within a few cents of zero is not a sharp boundary.

A synthetic population with a known answer checks the interval before it is trusted. At four settings of the outside option and the verifier, the registered interval covers the population value of the policy its procedure chooses in 93 to 99 percent of 120 samples of 500 tasks (Appendix E). The same check shows the distinction above at work: at a large outside option the best cap in the population saves four cents per task, while the cap the procedure picks costs three, because picking it from 500 tasks costs seven cents, more than the best cap saves.

5.4 The transfer test

The primary transfer fits the state rule's models on the other six configurations and scores it on the target. For the target's fold f, the models are fitted on the other configurations with fold f's tasks excluded from them too, so that the rule never sees a held-out task through another configuration. The claim this licenses is configuration-level generality within SWE-bench Verified under this harness, with each configuration held out in turn, not generality across domains.

6. Policies

All policies are schedules of attempts over (configuration, cutoff) pairs, evaluated on the same terms with the same outside option and the same verifier. Table 2 lists the ladder.

Table 2. The policy ladder. Each step is chosen on training folds from its class, the attempt budget $K \leq 4$ included.

Step	Policy	What it measures
i	One attempt, no cutoff, then the outside option	Baseline
ii	K attempts, no cutoff	The value of retrying
iii	K attempts at a constant cutoff	The value of a cap, given retry
iii-b	K attempts under a schedule of cutoffs	The value of varying the cap by attempt
iv	A state-aware restart rule	The value of observing the execution
v	A schedule over (configuration, cutoff) pairs	The value of switching configurations

Decision points. Every policy acts at the same points: every fifth call from 5 to 100, then every twenty-fifth from 125 to 475, below the configuration's own cap. That is 19 points for GPT-5 and 35 for the others. A constant cutoff stops at one of them regardless of state; a schedule may use a different one for each attempt. Extra opportunities to stop are not allowed to pass for the value of richer state.

Schedules. Step iii-b is searched by coordinate ascent, one attempt at a time, starting from the best constant cutoff on the same training tasks, and a move is kept only when it lowers cost on those tasks. On the folds that choose it, the schedule is therefore never worse than the constant cutoff, and its whole margin out of sample is the value of letting the cap vary by attempt, net of the cost of choosing it. The search is exact arithmetic on the replayed pool, checked against the evaluator.

The state-aware rule. Step iv is a receding-horizon restart rule. At each decision point it compares the expected cost of committing the running attempt to its end against restarting now, and stops the attempt when

$$s + v > q\,(V_{j+1} - \phi\bar{H}),$$

where q is the fitted probability that the attempt resolves if run to its end, s its fitted remaining spend, v the verification cost at the mean outside option $\bar{H}$, and V_{j+1} the value of starting afresh with the attempts that remain. Two models supply q and s: a logistic regression and a log-link quasi-Poisson regression,

each on three state variables (the log of the call count, of one plus cumulative output tokens, and of one plus output tokens over the last five calls), pooled across decision points. That is eight coefficients per configuration. The restart values V_j are the rule's own replayed costs on the training folds, computed backward over attempts, and K is chosen as for every policy. Committing to termination ignores the option to stop later and so biases the rule toward restarting, while restart values that are averages over tasks, and that replay a rule which is not optimal, can pull either way. The net direction is measured on synthetic data where the optimum is computable (Appendix E), not asserted.

The same rule serves within and across configurations; the two differ only in where the two models are fitted. For the transfer, the other six configurations' rows are pooled with equal weight per configuration, and in both cases the restart values and K come from the target configuration's own training folds, so the transfer tests whether reading an execution's state carries across configurations, not whether a configuration's success rate does. Elapsed time was registered as a state variable and dropped because Qwen logs no call times.

Cascades. Step v lets each attempt name a configuration as well as a cutoff. Retrying the same configuration is one of the choices, so retry and reroute are one decision. For each training set the search starts from the best single configuration's schedule, the best one configuration an operator could have picked from the same data, and moves one (configuration, cutoff) pair at a time. Switching's margin is measured against that best single configuration, chosen on the same folds.

7. Results

All values are expected cost per incoming task in dollars, chosen on training folds and scored on held-out folds. Margins are written so that a positive number is what the richer policy saves. Intervals are 95 percent bootstrap intervals that repeat the choice inside every replicate, and a margin is distinct from zero when its interval excludes zero.

Figure 1 draws the whole ladder, from one attempt to the transferred state rule, at three wages in the two headline regimes; Table 5 gives the values behind it. With an automated verifier nearly all of the fall in cost is the step from one attempt to several, in the median 3 percent at \$25 an hour and 14 and 18 percent at \$100 and \$300; the cap and the schedule then add up to two points of cost, and the state rule, which stops almost nothing, returns to what retrying costs. Under review at 0.5 the fall is the step from no cap to a cap, 10 percent at \$25 an hour and about 3 percent at \$100 and \$300, and the state rule gives part of it back at \$25 and most or all of it at \$100 and \$300.

7.1 Retrying

With an automated verifier, retrying pays at every wage from \$50 an hour up: its saving is distinct from zero there in every configuration, and from \$25 an hour for GPT-5 (two of Figure 1's three wages; Table 3 gives the saving with its interval). At \$100 an hour, letting the folds choose up to four attempts lowers expected cost by 13.2 to 19.6 percent against a single attempt for six configurations, and at \$300 an hour by 16.4 to 21.0 percent. The folds choose four attempts at both wages. Beside the cost sits the agent's own contribution: one attempt resolves 58 to 69 percent of tasks without the outside option, and the chosen budget 69 to 78 percent, the rest being resolved by the engineer either way. GPT-5.2 gains least, 4.3 percent at \$100 and 8.0 percent at \$300, and at \$25 an hour retrying it does not pay at all. At \$25 the other six gain 1.5 to 13.8 percent, most of them with two attempts. Where the outside option is worth five median attempts or fewer, no configuration retries.

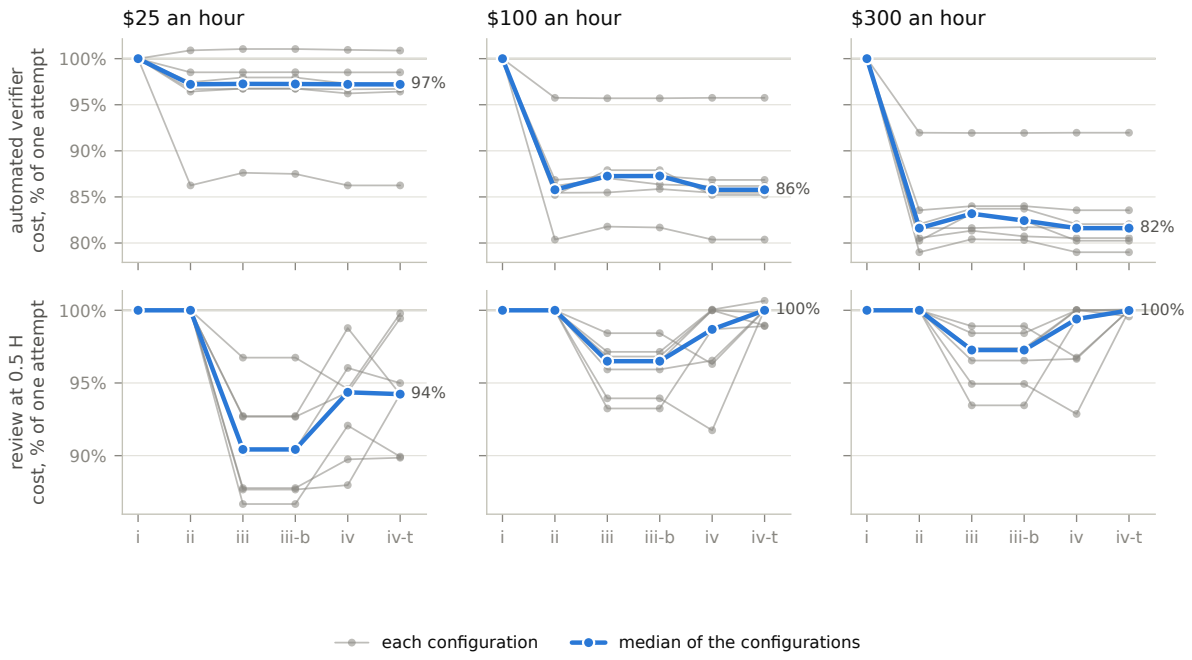

Figure 1. The ladder. Expected cost per task at each step as a share of the cost of one attempt followed by the outside option (step i), for each configuration (gray) and their median (blue), at $25, $100 and $300 an hour, with an automated verifier (top) and under review at 0.5 of the outside option (bottom). iv: the state rule fitted on the configuration itself; iv-t: fitted on the other six. Every step is chosen on training folds and scored on held-out folds, so a step can cost more than the one before it. † Imputed price.

Review changes this. When each patch submitted costs a review of 0.3 or 0.5 of the outside option, no configuration retries at any wage: every fold chooses one attempt. At 0.1 of the outside option, retrying pays in point estimate in 8 of the 63 configuration and wage cells, in none of them distinctly from zero. A second attempt is only worth its review when the first failed, and after a failure the chance that the next attempt succeeds is not high enough to cover a review priced at a large share of what the engineer would charge to fix the task.

7.2 The cap given retry

The primary result is the marginal value of a constant cutoff given retry, step ii minus step iii. The plan registered an expectation, set from the magnitudes the audit established and not from any policy result: that with an automated verifier the cap would not pay at any wage an enterprise would recognize, because the outside option exceeds attempt cost by two orders of magnitude, and that under human review it would pay, because a discarded attempt saves a review. Figure 2 shows the margin across the sweep in every regime, and Tables 4 and 5 give its break-evens and its values.

With an automated verifier, choosing a cap stops paying at two to four attempts' worth of outside option. For every configuration the margin is positive at the bottom of the sweep and crosses zero between 2.14 and 3.83 times the median attempt cost, which is between $1.74 and $7.08 an hour (Table 4). The saving is distinct from zero only below about one to two median attempt costs, between $1.16 and $5.00 an hour. Above the crossing the margin is small and mostly negative: at $100 an hour it runs from −0.74

Table 3. The value of retry. Step i minus step ii, dollars per task with its 95 percent interval from the same bootstrap as the cap's margin, automated verifier. The last three columns are at $100 an hour: the saving as a share of what one attempt and the outside option cost, the attempt budgets the folds chose, and the share of tasks resolved without the outside option by one attempt and by the chosen budget, in percent. Under review at 0.3 and 0.5 every fold chooses one attempt, so retry is worth nothing there. † Imputed price.

configuration	$25 [95%]	$100 [95%]	$300 [95%]	% of i	attempts	resolves i, ii
GPT-5	1.04 [0.41, 1.82]	5.67 [3.42, 8.80]	18.04 [11.27, 27.46]	19.6	4	60, 70
GPT-5.2	−0.06 [−0.12, 0.18]	1.01 [0.45, 2.16]	5.59 [2.76, 8.99]	4.3	2, 3, 4	67, 72
Sonnet 4	0.20 [−0.08, 0.35]	3.71 [2.13, 5.42]	13.58 [8.98, 18.76]	14.6	4	69, 78
Sonnet 4.5	0.12 [−0.23, 0.43]	3.83 [1.99, 5.73]	15.47 [10.52, 21.37]	13.8	4	64, 72
Gemini 3 Pro	0.25 [−0.03, 0.38]	3.64 [1.95, 5.44]	13.31 [8.67, 18.76]	13.2	4	61, 70
Kimi K2	0.29 [−0.05, 0.48]	4.19 [2.44, 5.91]	15.48 [10.71, 20.73]	14.2	4	58, 69
Qwen3 Coder†	0.21 [−0.14, 0.48]	4.09 [2.26, 5.88]	15.78 [10.85, 21.28]	14.8	4	66, 76

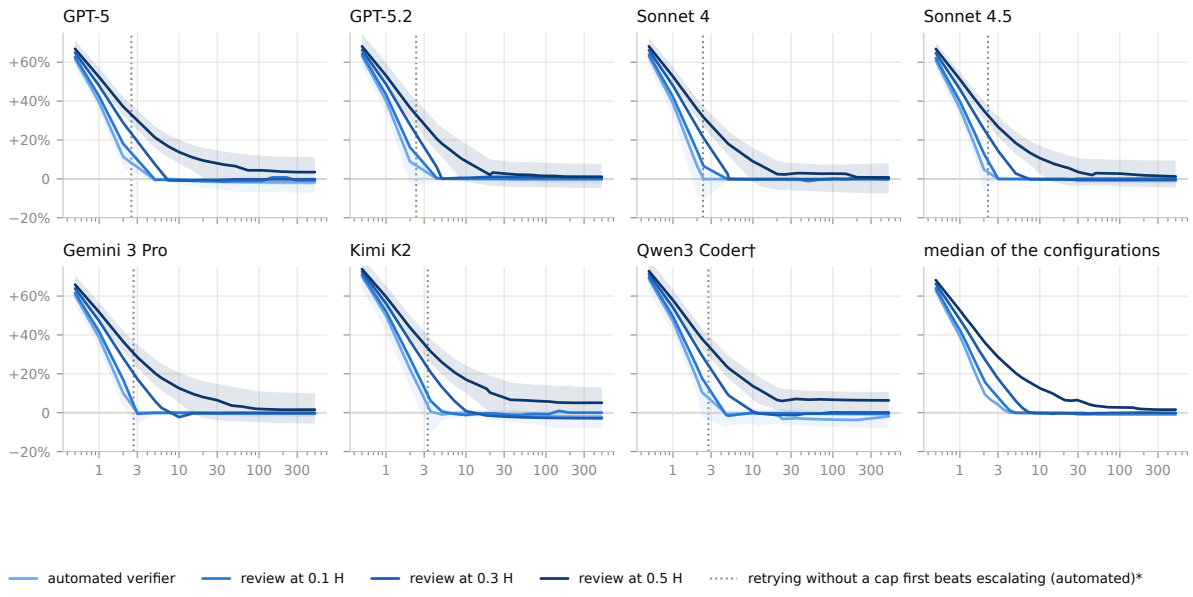

Figure 2. The cap's marginal value given retry. Step ii minus step iii across the sweep, for each configuration and as the median across them, as a share of what retrying without a cap costs. The horizontal axis is the outside option in multiples of the configuration's median attempt cost. Bands are 95 percent intervals from 1,000 bootstrap replicates for the automated verifier and for review at 0.5 of the outside option. The dotted line, an exploratory addition and starred in the legend, marks the outside option above which retrying without a cap first costs less than escalating every task. † Qwen3 Coder's price is imputed.

to +0.01 dollars per task, a median of −0.24 (Table 5), and its interval straddles zero at every point of the sweep above the crossings, in every configuration. It is never distinct from zero on the cost side either, in any of the sweep's 532 cells, though those cells are far from independent and the count is a description, not a test. Some configurations cross zero more than once, GPT-5.2 four times, and no crossing has the interval clear of zero on both sides.

The gain available to any cap is small, whichever way it is measured. In a check the plan did not register, chosen with hindsight on all of a configuration's tasks, where the capped family contains the uncapped policy and cannot lose, the best constant cap saves at most three cents per task at $100 an hour, 0.12 percent of what retrying costs, and nothing at all for Sonnet 4 (`cap_margin_in_sample` in `results/ladder.csv`). Out of sample, what the folds' choices lose is the cost of picking from that class on about 400 tasks.

What the folds choose explains the size of the margin. With an automated verifier at $100 an hour, step iii keeps four attempts, except in some folds for GPT-5.2, and caps each at 80 to 275 calls, or in at least one of GPT-5's folds not at all. The cap therefore stops only attempts that have already run long, some of which would have succeeded. Each success cut off costs the outside option, about $50 on average at this wage, against the few dollars of tokens the cap saves on the attempts it stops. The cap is chosen on the training folds because it helps there by a little, and out of sample that little does not survive.

Table 4. The break-even of the cap's marginal value. For each configuration and regime, the first rate at which the margin changes sign, in multiples of the median attempt cost with dollars an hour in parentheses, and the largest multiple up to which the cap's saving is distinct from zero. None: the margin does not change sign in the sweep; under review at 0.5 the cap saves across the whole sweep in point estimate. Every crossing found is in `results/breakeven.csv`; at none is the interval clear of zero on both sides. † Imputed price.

Configuration	Automated	Review 0.1	Review 0.3	Review 0.5
GPT-5	2.54 ($1.74); distinct to 2.0	3.18 ($2.18); 2.0	6.86 ($4.71); 2.0	none; 20.0
GPT-5.2	2.89 ($3.36); 1.0	5.79 ($6.73); 2.0	none; 2.0	none; 8.6
Sonnet 4	2.18 ($4.51); 1.0	2.68 ($5.54); 2.0	4.98 ($10.29); 2.4	none; 5.0
Sonnet 4.5	2.14 ($7.08); 1.5	2.68 ($8.87); 2.0	5.47 ($18.11); 3.0	none; 10.0
Gemini 3 Pro	2.35 ($3.90); 2.0	2.95 ($4.88); 2.0	6.13 ($10.16); 3.0	none; 10.0
Kimi K2	3.83 ($5.30); 2.0	5.90 ($8.16); 2.0	10.76 ($14.88); 5.0	none; 20.0
Qwen3 Coder†	2.73 ($5.91); 2.0	3.42 ($7.40); 2.3	10.56 ($22.82); 4.6	none; 23.1

Under review at half the outside option the cap saves, in point estimate at every wage, but the saving is distinct from zero only at the lower wages. With review at half the outside option, the margin is positive across the whole sweep for all seven configurations, from $0.75 to $3.56 per task at $100 an hour, but its interval excludes zero only up to between 5 and 23 median attempt costs, between $10 and $50 an hour. At review of 0.3, six configurations cross from saving to costing between 5.0 and 10.8 median attempt costs, and at 0.1 the crossings sit between 2.7 and 5.9, close to the automated ones. As with the automated verifier, on the cost side of every crossing the interval straddles zero. The registered expectation holds in point estimate at a review share of 0.5: opposite answers in the two regimes from one policy class, with the review side distinct from zero only at the lowest wages, up to $10 to $50 an hour. At review shares of 0.1 and 0.3 the answer is the automated one with a later break-even.

What the cap does under review is mostly not to run the agent. At $100 an hour under review at 0.5 the capped policy resolves only 7 to 51 percent of tasks itself, a median of 20, against 58 to 69 percent for one uncapped attempt (Table 5). At $5 and $10 an hour, every fold under review at 0.5 chooses one

attempt cut at 5 calls, which spends a few cents, submits nothing for review, and hands the task to the outside option. At higher wages most folds cut a single attempt at 20 to 90 calls, so that the long attempts, which are the least likely to succeed, are never reviewed; a few still cut at 5 calls, and at the top of the sweep some folds keep two to four capped attempts, for GPT-5 and Kimi K2 from $100 an hour and for Sonnet 4 and Sonnet 4.5 from 200 median attempt costs. A comparison the plan did not register makes the point sharper. Across the wage sweep under review at 0.5, the best capped policy costs between 3.7 percent less and 3.5 percent more than sending every task to the outside option without running the agent, and a single attempt without a cap costs a median of 6 percent more than escalating everything. At a review price of half the outside option, the agent on this benchmark is worth about nothing, and the cap's registered saving is the saving of submitting less for review.

7.3 Schedules

Letting the cap vary by attempt adds nothing measurable. With an automated verifier the schedule's margin over the constant cutoff at $100 an hour runs from −0.10 to +0.19 dollars per task, a median of zero, and at $300 from −0.08 to +0.60; its interval includes zero for every configuration at every rate at which it is computed. Under review at 0.5 the margin is zero at all three wages for every configuration, since the folds choose few attempts and cut them alike; at 0.1, where the folds retry, it runs from −0.64 to +0.31 at $300 in point estimate. The search starts from the best constant cutoff and moves only when a move pays on the training folds, so on those folds the schedule is never worse; the moves it finds do not carry to held-out tasks.

7.4 Execution state and the primary transfer

The primary transfer is the state rule's margin over the best schedule, step iii-b minus the rule, with its two models fitted on the other six configurations. Table 6 and Figure 3 give it with an automated verifier.

At the three wages with intervals, the transferred rule never loses to the schedule by more than two cents, and its margin is distinct from zero in three cells: GPT-5 at $100 and $300 an hour, and Sonnet 4 at $300, whose lower endpoint is a cent. The same three cells were distinct from zero at 100 replicates, and Sonnet 4 at $100, whose lower endpoint was three tenths of a cent below zero at 100, sits four tenths below at 1,000. The median margin is $0.01, $0.12 and $0.36 per task at the three wages. The last column says why. At $100 and $300 the transferred rule costs exactly what retrying without a cap costs, to three decimals, for every configuration: it does not stop attempts. Its margin over the schedule is therefore step ii minus step iii-b, the cost the schedule pays out of sample for having been chosen from data. The rule fitted on the configuration itself behaves the same way with an automated verifier. With the outside option worth tens of dollars and an attempt costing one, the fitted chance of success would have to fall to a few percent before stopping paid, and the models rarely predict that.

Under review the rule does stop attempts, and there the two fits part (Figure A1). Fitted on the configuration itself, the rule beats the best schedule at $100 an hour for GPT-5.2, by $1.01, and Kimi K2, by $1.20, and loses for the other five, by $0.33 to $2.87. Fitted on the other six configurations, it loses for all seven, by $0.75 to $3.33. What an execution's state says about its chances carries across configurations only as far as saying that it will probably finish, which with an automated verifier is all the rule needs to know. When the decision turns on which attempts to submit for an expensive review, the models fitted elsewhere do not carry what is needed.

Synthetic attempts whose optimal restart policy can be computed exactly show which way the rule errs (Appendix E). Where the rule's own state determines the attempt's prospects, the optimum stops 29 to

Table 5. Policy value across the sweep. Dollars per task, each step chosen on training folds and scored on held-out folds. Cap's saving: step ii minus step iii, with its 95 percent interval. Step iii chose: the folds' distinct choices, attempts × cutoff in calls, cutoffs sharing an attempt count separated by a slash; none is no cutoff. Resolves: the share of tasks steps i, ii, iii and iii-b resolve without the outside option, in percent. Under review at 0.5 step ii is step i, every fold there choosing one attempt. * Exploratory, not registered: every task to the outside option; it differs across configurations only because each is scored on its own tasks with four draws (Table 1). † Imputed price.

Configuration	$/h	i	ii	iii	iii-b	Cap's saving [95%]	Step iii chose	Resolves %	Escalate all*
Automated verifier									
GPT-5	25	7.54	6.50	6.60	6.59	−0.10 [−0.38, 0.01]	4×80/85/none	60/70/70/70	12.56
	100	28.91	23.23	23.64	23.61	−0.41 [−1.62, 0.01]	4×80/85/none	60/70/70/70	50.22
	300	85.90	67.86	69.07	68.99	−1.21 [−4.84, 0.01]	4×80/85/none	60/70/70/70	150.67
GPT-5.2	25	6.46	6.52	6.53	6.53	−0.01 [−0.08, 0.02]	1×175, 2×125/175	67/70/69/69	11.96
	100	23.66	22.65	22.64	22.64	0.01 [−0.43, 0.09]	2×175, 3×175, 4×125	67/72/72/72	47.84
	300	69.51	63.92	63.91	63.91	0.02 [−1.50, 0.06]	4×125	67/72/72/72	143.53
Sonnet 4	25	7.33	7.13	7.13	7.13	−0.00 [−0.06, 0.00]	2×200/275	69/74/74/74	12.58
	100	25.47	21.76	21.77	21.87	−0.01 [−0.32, 0.00]	4×200/275	69/78/78/78	50.31
	300	73.84	60.26	60.27	60.35	−0.01 [−0.73, 0.00]	4×200/275	69/78/78/78	150.93
Sonnet 4.5	25	8.31	8.19	8.19	8.19	−0.00 [−0.03, 0.01]	2×175/225	64/68/68/68	12.56
	100	27.72	23.89	24.12	23.94	−0.24 [−0.36, 0.02]	4×150/175	64/72/71/72	50.22
	300	79.48	64.01	64.64	64.15	−0.64 [−1.27, 0.01]	4×150/175	64/72/71/72	150.67
Gemini 3 Pro	25	7.68	7.43	7.43	7.43	−0.00 [−0.10, 0.01]	2×300/400	61/67/67/67	12.29
	100	27.67	24.03	24.14	24.14	−0.11 [−0.48, 0.05]	4×175/225	61/70/70/70	49.17
	300	80.95	67.64	68.00	68.00	−0.36 [−1.45, 0.05]	4×175/225	61/70/70/70	147.51
Kimi K2	25	8.14	7.85	7.87	7.87	−0.03 [−0.23, 0.13]	2×175/250/275	58/65/64/64	12.62
	100	29.44	25.26	25.69	25.71	−0.44 [−1.50, 0.21]	4×175/250	58/69/68/68	50.48
	300	86.26	70.78	72.22	72.22	−1.44 [−5.47, 0.15]	4×175/250	58/69/68/68	151.43
Qwen3 Coder†	25	8.05	7.85	7.89	7.89	−0.04 [−0.52, 0.12]	2×175/225/300	66/73/72/72	12.56
	100	27.64	23.55	24.30	24.30	−0.74 [−1.58, 0.37]	4×125/225	66/76/76/76	50.22
	300	79.88	64.10	66.44	65.84	−2.34 [−4.76, 0.25]	4×125/225	66/76/76/76	150.67
Review at 0.5 H									
GPT-5	25	13.59	13.59	12.60	12.60	1.00 [−0.24, 1.91]	1×5	60/60/0/0	12.56
	100	53.14	53.14	50.97	50.97	2.16 [−1.53, 6.20]	1×20/35, 2×20	60/60/10/10	50.22
	300	158.58	158.58	153.10	153.10	5.48 [−4.88, 17.85]	1×20/35, 2×20, 4×20	60/60/12/12	150.67
GPT-5.2	25	12.41	12.41	12.01	12.01	0.40 [−0.44, 1.22]	1×45/70	67/67/48/48	11.96
	100	47.45	47.45	46.70	46.70	0.75 [−2.02, 4.03]	1×45/70/75	67/67/51/51	47.84
	300	140.89	140.89	139.35	139.35	1.54 [−6.58, 11.09]	1×45/50/70/75	67/67/53/53	143.53
Sonnet 4	25	13.62	13.62	12.63	12.63	0.99 [−0.50, 1.94]	1×5	69/69/0/0	12.58
	100	50.62	50.62	49.18	49.18	1.45 [−3.13, 3.92]	1×60/65/70	69/69/45/45	50.31
	300	149.31	149.31	145.41	145.41	3.90 [−10.56, 10.94]	1×60/70	69/69/46/46	150.93
Sonnet 4.5	25	14.55	14.55	12.61	12.61	1.94 [1.06, 2.84]	1×5	64/64/0/0	12.56
	100	52.69	52.69	50.85	50.85	1.85 [−1.86, 5.59]	1×5/70/75	64/64/16/16	50.22
	300	154.40	154.40	150.17	150.17	4.22 [−5.93, 15.12]	1×70/75/90	64/64/27/27	150.67
Gemini 3 Pro	25	13.69	13.69	12.38	12.38	1.31 [−0.12, 2.44]	1×5	61/61/0/0	12.29
	100	51.69	51.69	50.05	50.05	1.64 [−2.36, 6.41]	1×5/60	61/61/7/7	49.17
	300	153.04	153.04	150.63	150.63	2.41 [−8.18, 15.89]	1×5/70	61/61/15/15	147.51
Kimi K2	25	14.42	14.42	12.64	12.64	1.78 [0.24, 2.84]	1×5	58/58/0/0	12.62
	100	54.56	54.56	51.25	51.25	3.30 [−1.48, 7.84]	1×40/55/60/65	58/58/20/20	50.48
	300	161.61	161.61	153.42	153.42	8.19 [−5.77, 21.69]	1×40/55/65, 3×40	58/58/18/18	151.43
Qwen3 Coder†	25	14.33	14.33	12.58	12.58	1.75 [0.66, 2.70]	1×5	66/66/0/0	12.56
	100	52.75	52.75	49.19	49.19	3.56 [−0.36, 6.03]	1×60/65	66/66/42/42	50.22
	300	155.21	155.21	145.06	145.06	10.15 [−2.16, 16.88]	1×65	66/66/44/44	150.67

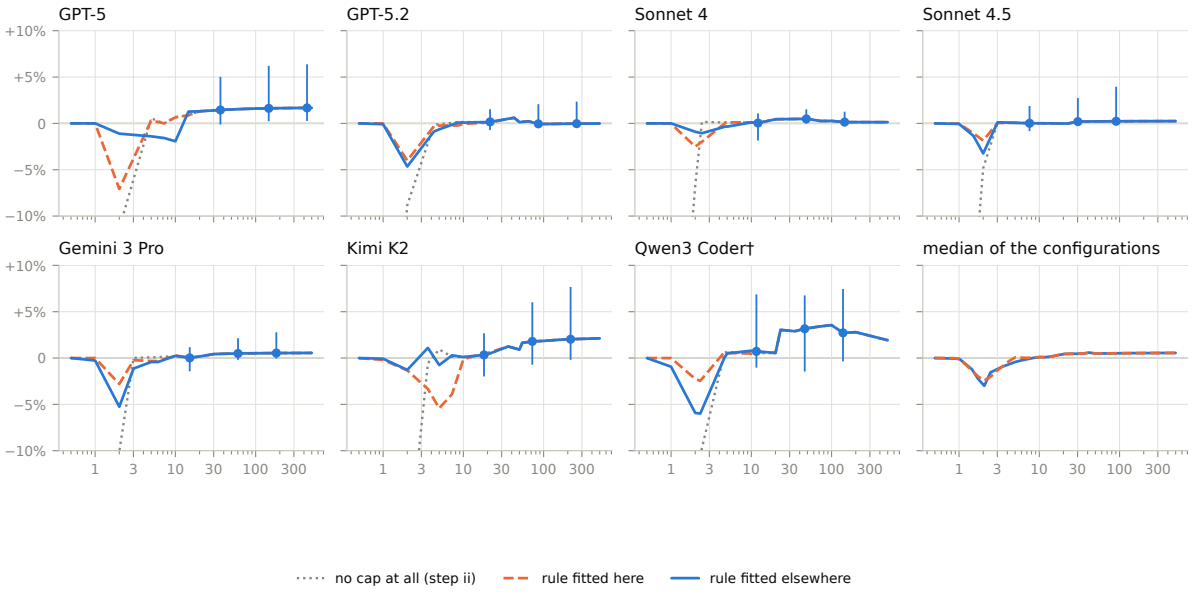

Figure 3. The state rule against the best schedule, automated verifier. Step iii-b minus the state rule, as a share of what retrying without a cap costs, for the rule fitted on the other six configurations (solid) and on the configuration itself (dashed), and for not capping at all (dotted). Points are 95 percent intervals for the transferred rule at $25, $100 and $300 an hour, from 1,000 replicates that refit every model. Where the lines coincide the rule is not stopping attempts. † Imputed price.

Table 6. The primary transfer. Step iii-b minus the state rule fitted on the other six configurations, automated verifier, dollars per task, with 95 percent intervals from 1,000 replicates that refit the rule's models and the schedule inside each. The last two columns, at $100 an hour, give the same margin for the rule fitted on the configuration itself, and how far the transferred rule sits from retrying without a cap. † Imputed price.

Configuration	$25	$100	$300	Within, $100	Transferred minus ii
GPT-5	0.09 [−0.01, 0.33]	0.38 [0.05, 1.44]	1.13 [0.18, 4.32]	0.38	0.000
GPT-5.2	0.01 [−0.05, 0.10]	−0.01 [−0.10, 0.47]	−0.02 [−0.06, 1.50]	−0.01	0.000
Sonnet 4	0.00 [−0.13, 0.08]	0.11 [−0.004, 0.33]	0.08 [0.01, 0.76]	0.11	0.000
Sonnet 4.5	0.00 [−0.07, 0.15]	0.05 [−0.04, 0.65]	0.15 [−0.03, 2.54]	0.05	0.000
Gemini 3 Pro	0.00 [−0.11, 0.09]	0.12 [−0.05, 0.51]	0.36 [−0.04, 1.88]	0.12	0.000
Kimi K2	0.03 [−0.16, 0.21]	0.45 [−0.18, 1.52]	1.44 [−0.14, 5.43]	0.45	0.000
Qwen3 Coder†	0.06 [−0.08, 0.54]	0.74 [−0.34, 1.59]	1.74 [−0.23, 4.78]	0.74	0.000

65 percent of first attempts with an automated verifier, at 8 to 42 calls on average, while the fitted rule stops 42 to 78 percent at 5 to 10 calls. The rule costs 7 to 32 percent more than the optimum with an automated verifier, and under review at 0.5 it costs 7 to 14 percent more at the two higher outside options and the same at the lowest, where both escalate almost at once; and in three of the six automated settings it costs more than the best constant cutoff. The bias toward restarting that the commit-to-termination approximation introduces outweighs the opposite pull of its restart values.

7.5 Switching configurations

After a failed attempt an operator can retry the same configuration or switch to another. Figure 4 and Table 7 compare the best cascade with the best single configuration chosen from the same training folds, on the 275 tasks with four usable draws in all seven configurations.

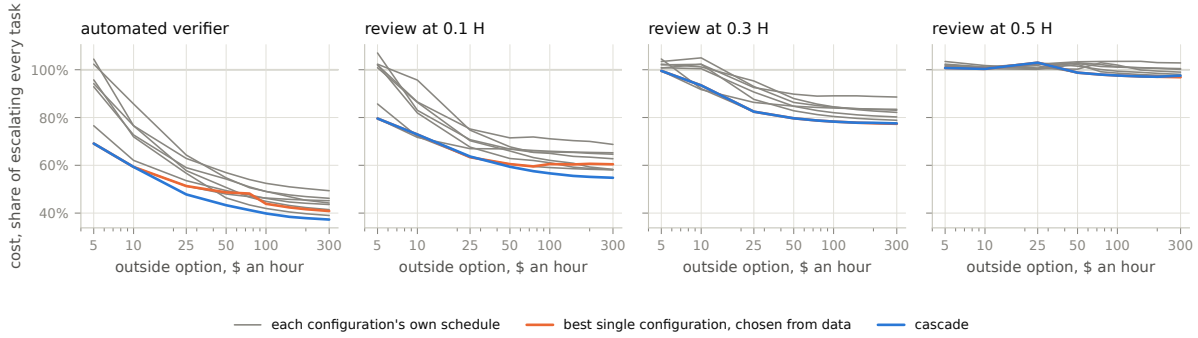

Figure 4. Switching configurations. The best cascade (blue), the best single configuration chosen from the same training folds (orange) and each configuration's own schedule (gray), as a share of the cost of escalating every task, on the 275 tasks with four usable draws in all seven configurations. Escalating every task is an exploratory reference. † Imputed price.

Table 7. Switching configurations. Dollars per task on the 275 common tasks. Switching saves: the best single configuration chosen on the training folds minus the cascade, with a 95 percent interval from 100 replicates where computed. Best own in hindsight: the cheapest configuration's own schedule, cross-fitted, and which configuration that is. Escalate all: every task sent to the outside option without running the agent, an exploratory reference not in the registration. † Imputed price.

Regime	$/h	Best single	Cascade	Switching saves	Best own in hindsight	Escalate all
Automated	25	6.42	5.98	0.44 [−0.27, 1.36]	6.42 (GPT-5)	12.51
Automated	100	21.97	19.94	2.03 [−2.22, 5.59]	20.99 (Sonnet 4)	50.05
Automated	300	61.36	56.01	5.35 [−8.30, 17.95]	58.56 (Sonnet 4)	150.16
Review 0.1	100	30.29	28.35	1.95	29.56 (GPT-5.2)	50.05
Review 0.3	100	39.19	39.19	0.00	39.19 (GPT-5.2)	50.05
Review 0.5	100	48.84	48.84	0.00	48.84 (GPT-5.2)	50.05

With an automated verifier, switching lowers cost by 6.8 percent at $25 an hour, 9.2 percent at $100 and

8.7 percent at $300 in point estimate, but every interval includes zero. Against the configuration that proves cheapest in hindsight, Sonnet 4 at $100 and $300, the cascade is 5 percent cheaper at $100 and 4 percent at $300; against the best configuration an operator could have picked from the same data, it is 9 percent cheaper. The cascades the folds build rely mainly on two configurations: removing GPT-5.2 raises the cascade's cost at $100 by $1.48 and removing Sonnet 4 by $0.96, while removing any of the other five changes it by less than $0.16. Under review at 0.3 and 0.5 at $100 an hour the best policy is a single configuration, GPT-5.2, with one capped attempt, and switching adds nothing, while at the highest wages under review the cascade chosen from data costs slightly more than the best single configuration; at 0.1 switching saves 6 to 9 percent in point estimate at $100 to $300.

Switching pays only where failures do not coincide, and here they mostly do. Across the 275 common tasks, the share of a configuration's draws that resolve is correlated with another configuration's at 0.62 to 0.90, a median of 0.81 (Figure A2). The two configurations that stand apart are Sonnet 4 and Qwen3 Coder, the two run under the oldest harness versions, which correlate with the other five at 0.62 to 0.74 and with each other at 0.85.

7.6 Sensitivities

Nine registered sensitivities change one input each and rerun the ladder, the break-even and the cascade with every choice redone (Appendix A); the cap stated in dollars rather than calls, which the plan also registers as a sensitivity, is reported with the comparators (Appendix B). None changes the pattern of the primary result, although individual cells move. With an automated verifier the first break-even stays between 1.8 and 8.8 median attempt costs under every sensitivity, and its median across configurations between 2.1 and 5.0. False accepts raise it most, to a median of 3.8 at $\phi = 0.34$ and 5.0 at $\phi = 0.50$, because a success cut off by the cap then costs less: part of its value was never real. Under no sensitivity is the cap's margin distinct from zero on the cost side anywhere in the sweep. The only crossings with the interval clear of zero on both sides come under the common 100-call horizon, two with an automated verifier and two under review at 0.1, and each runs from saving to exactly zero, where no fold chooses a cap that binds.

Under review at 0.5 the cap still saves across the whole sweep for every configuration under the false-accept and price sensitivities, and when refusals are excluded or evaluations without a verdict are counted as failures. False accepts raise that saving from a median of $1.85 per task at $100 an hour to $10.20 at $\phi = 0.34$ and $13.69 at $\phi = 0.50$, since every patch the cap keeps from review now also carries part of an outside option that will be paid later. Under the other four, the saving turns into a cost high in the sweep for one to three configurations: between 11.6 and 49 median attempt costs under the bucket-specific correction to the annotated times, which raises the shortest tasks' outside option about eightfold; at 41 and 58 under the common horizon; at 46 and 136 when every task that can fill a policy is scored; and at 272 when re-runs are dropped. As everywhere else, on the costing side the interval straddles zero.

Switching keeps its sign: with an automated verifier at $100 an hour it saves between $1.00 and $5.48 per task under every sensitivity, most under the annotation correction and the common horizon. The two-stage bootstrap, which resamples each task's attempts as well as the tasks, gives intervals for the cap's margin of about the same width as the task-level bootstrap, a median ratio of 1.02, and centered in the same place at the median, though individual cells shift. GPT-5's transfer margin stays distinct from zero at $100 and $300 an hour under it; Sonnet 4's at $300 does not.

The medians across configurations quoted throughout summarize seven estimates, each on its own configuration's tasks with four usable draws, 405 to 500 of them; they are not estimates on one common task

set. Appendix A adds a check, not registered, that scores every configuration on the 275 tasks common to all seven. On those tasks the medians move a little and the pattern does not: with an automated verifier retrying saves a median of 13 percent at $100 an hour against 14 on each configuration's own tasks, the cap's median margin is −$0.41 against −$0.24, and the first break-even runs from 2.0 to 5.2 median attempts, a median of 2.3.

8. Interpretation

8.1 Why a cap does not pay when the fallback is a person

A cap trades two things of very different size. It saves the spend of the attempts it stops, which is at most the cost of the calls they would have made after the cutoff, and it loses the successes among them, each of which must then be bought from the next attempt or from the outside option. With an automated verifier and a human fallback, an attempt costs cents to a dollar or two and the outside option tens of dollars, so a cap pays only if the attempts it stops almost never succeed. They do succeed: in the six configurations capped at 500 calls, an attempt still running at 100 calls goes on to resolve 33 to 55 percent of the time (Appendix D). No cutoff meets that test. In an unregistered check, even the best constant cap chosen with hindsight on all of a configuration's tasks saves at most three cents per task at $100 an hour, and the folds, choosing on about 400 tasks, cannot find that much: the caps they choose at $100 and $300 an hour, at 80 to 275 calls, stop only attempts that have already run long, and the margin that remains is the cost of having chosen them. Two statements are therefore distinct. The gain available to a cap here is a few cents at most; and the deployable value of choosing a cap from this much history, which is what the estimand measures, is nothing or a little less, by up to a few percent of what retrying costs. A far larger history could in principle find the few cents. It could not find more, on these logs.

The break-even says the same thing in the operator's units. Choosing a cap, given retry, pays only while the outside option is worth fewer than two to four median attempts. The multiple, unlike the dollar figure, does not depend on the price of tokens: a uniform change in prices rescales every attempt and every outside option measured in attempts by the same factor, so the break-even in multiples stays where it is and only its dollar value moves. For the cap to pay against an engineer at $100 an hour, with the agents behaving as they did here, token prices would have to be at least 14 to 57 times the list prices these runs were priced at. An iteration limit on these configurations is therefore a harness setting, a guard against runaway attempts, and not a lever on cost. Setting it well below where the harness sets it, as the common-horizon sensitivity does by stopping the 500-call configurations at 100, removes between 1 and 21 percent of their successes, and at $100 an hour raises what retrying costs them by $0.31 to $6.94 per task.

Retrying is the lever. It pays because a failed attempt does not doom the next one: the runs of one configuration on one task differ widely, and a second or third draw resolves tasks the first did not. Only 11 to 25 percent of tasks have mixed outcomes across their four runs, and all of retrying's value comes from them; on the 22 to 31 percent where every run fails, a retry adds an attempt that cannot succeed, which costs little against the outside option (Appendix D). The same size difference that makes a cap useless makes retrying valuable, since each success a retry buys saves an outside option for the price of an attempt.

8.2 A rule of thumb in attempts

Two regularities that the plan did not register make the break-even easier to use. The first is that it sits close to the outside option at which retrying without a cap first costs less than escalating every task without running the agent: between 2.24 and 3.33 median attempts with an automated verifier, against a break-even of 2.14 to 3.83, and within about half a median attempt of each other for every configuration. Below that point the cheapest thing to do is not to run the agent, and a cap at five calls is the nearest thing to it that the class contains, so the cap's saving at the bottom of the sweep is mostly the saving of not running the agent.

The second is that the break-even is stable across verifier regimes once review is counted as part of an attempt's cost (Figure 5). An attempt that produces a patch costs its tokens plus a review, so at a review share f and a break-even of M median attempts, the outside option is worth $M/(1+fM)$ full attempts. For five of the seven configurations, the first break-even lies between 2.0 and 2.7 full attempts with an automated verifier and under review at 0.1 and 0.3, and moves by 3 to 13 percent across the three. Kimi K2 and GPT-5.2 break even later, at up to 3.8 full attempts with an automated verifier or review at 0.1.

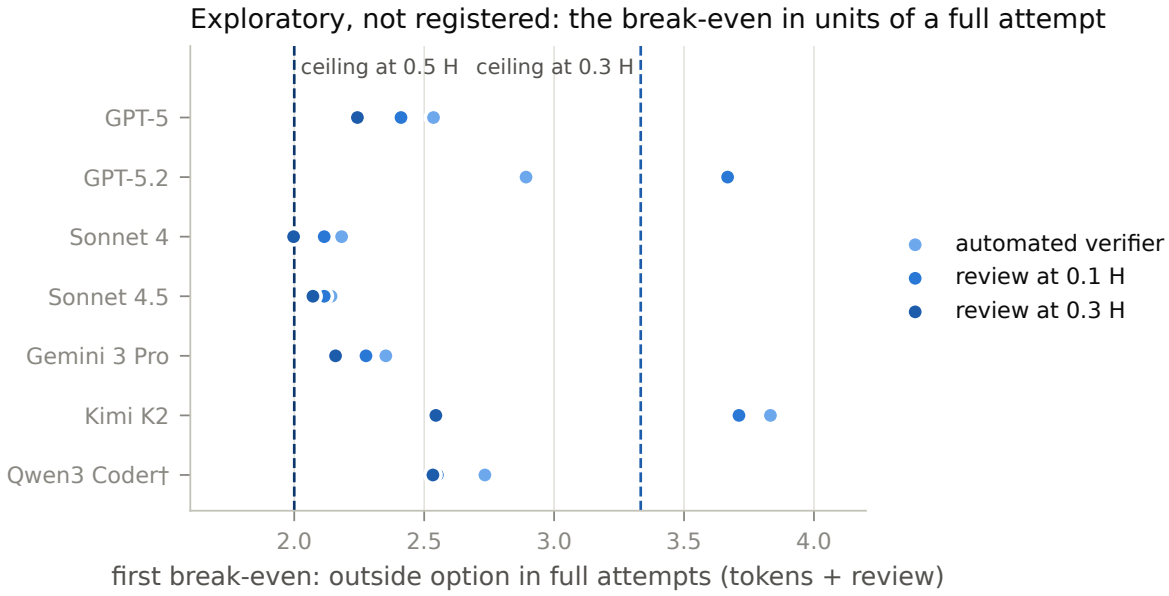

Figure 5. The first break-even in full attempts (exploratory). The outside option at the cap's first break-even, divided by the full cost of an attempt, its median token cost plus the review of its patch. The dashed lines are the most the outside option can be worth in full attempts under review at 0.5 and 0.3 of the outside option. Under review at 0.5 no configuration crosses, and GPT-5.2 does not cross at 0.3. Not registered; computed after the registered results. † Imputed price.

Read together, the two give an operator a rule that needs only two numbers: choosing a cap on attempt length pays when the outside option is worth fewer than about two to four full attempts, which is also about where running the agent at all stops paying. Above that, cap generously and retry. The rule is exploratory, drawn from one benchmark and one harness, and should be checked before it is used elsewhere.

8.3 Under review, whether to run the agent

With review priced at half the outside option, the question the policy class answers is not how long to let an attempt run but whether to run it. The comparisons with escalating every task in this section are exploratory. A single uncapped attempt followed by escalation saves the outside option when the attempt resolves, and costs its tokens and, when it produces a patch, a review. It pays only if the chance of resolving, weighted by the outside option, exceeds the review share plus the token cost as a share of the outside option. At $100 an hour a single attempt costs more than escalating every task for six of the seven configurations (Table 5), so for those six the resolve rate weighted by each task's outside option cannot be much above one half, although the unweighted rate is 58 to 69 percent. The agent resolves least often the tasks that are most expensive to escalate. The difficulty display shows where: with the annotation admitted to every policy, under review at 0.5 the cap saves $39 to $63 per task on the tasks annotated at an hour or more (Appendix C). The cap then saves by not submitting the long attempts, the least likely to succeed, for review, and the best capped policy ends within 4 percent of not running the agent at all.

What would change this is on the verifier's side. At a review of a tenth of the outside option the best single configuration costs 40 percent less than escalating every task at $100 an hour, retrying pays in point estimate in some cells, and the break-even returns to about where it is with an automated verifier. The value of an agent in a reviewed workflow depends on the price of review at least as much as on the agent's pass rate, and a benchmark's resolve rate, unweighted by what the tasks would cost to do another way, overstates it.

8.4 What state-aware stopping would need

The restart rule stops an attempt when its remaining spend and review exceed the fitted chance of success times what a fresh start would cost. With an automated verifier and an outside option of tens of dollars, a fresh start is worth tens of dollars and an attempt's remaining spend about one, so the rule stops only when the fitted chance of success falls to a few percent. Three summaries of an execution's progress seldom predict that, and the rule behaves as retrying without a cap. Its measured margin over the best schedule is then the cost the schedule pays for having been chosen from data, not a value of observing the state.

This is a statement about the registered rule, eight coefficients on three state variables, not about what an execution's state could tell a better reader, and the synthetic check shows the rule errs even where its state is informative: it stops too many attempts, too early. A richer model of the execution, or a learned monitor of the kind Wang et al. (2026) and Guo et al. (2026) train, could find attempts with a near-zero chance of success that these regressions cannot. The estimand here says what such a monitor has to achieve. A false alarm on an attempt that would have succeeded costs the outside option, less what the retry recovers, while a correct stop saves the rest of an attempt's spend; tokens saved at a fixed rate of false alarms is not enough to know whether a monitor pays. Under review the calculation shifts toward stopping, and there the rule fitted on the configuration itself gains in point estimate for two configurations, but the version fitted elsewhere loses for all seven.

8.5 Choosing the configuration and choosing the policy

On the 275 tasks that all seven configurations cover, with an automated verifier at $100 an hour, the configurations' best policies span $4.94 per task. Within a configuration, the policies from one attempt to the best fixed schedule span a median of $3.83 and at most $4.44; on each configuration's own tasks almost all of that range is the value of retrying, and the cap and the schedule move cost by tens of cents (Section 7). Under review at 0.1 and 0.3, at $25, $100 and $300 an hour, the range across configurations is nine or more times the median range across policies within one (Appendix C). Choosing the configuration and deciding to retry are the first-order decisions, and switching configurations after a failure, at about 9 percent in point estimate, is of the same order as the difference between configurations, though its interval includes zero. The sample oracle, which picks the cheapest schedule for each task with hindsight, costs 29 percent less than the best cascade at $100 an hour, an upper estimate of what knowing each task in advance could be worth (Appendix B). The configurations also differ in harness version and date, so their spread is not a comparison of models alone.

9. Boundaries and Limitations

Clean restarts. Every attempt starts fresh from the task, and in a cascade the next configuration starts fresh rather than inheriting a partial patch, context or tool state. A retry that carries forward what the failed attempt learned could do better or worse, and the gap is not measured. A cut-off attempt is discarded here, while the harness would have submitted its patch; with an automated verifier that makes the measured value of a cap conservative (Section 5.1).

The verifier. The automated verifier is the benchmark's test suite, deterministic and free. Production verification has a cost, priced here as a share of the outside option, and an error rate, priced as a single false-accept share taken from outside sources and assumed not to depend on the policy. Review that costs a fixed amount per patch rather than a share of the task's outside option would favor retrying on long tasks and penalize it on short ones.

One benchmark, one harness, four draws. The evidence comes from SWE-bench Verified under OpenHands, with four runs per task. The attempt budget is at most four, and nothing is said about a fifth attempt. The transfer is across configurations within this setting, and harness versions, run dates and endpoints differ across the configurations, so a configuration is not a model. Three configurations ran at temperature zero, and their variation across runs comes from nondeterminism in the providers and the harness; exchangeability of the four runs is assumed, not tested, although Gemini's occasional repeated trajectories are priced as the behavior a retry would meet. The cascades assume more: that, given the task, the draws of different configurations are independent, so that combinations of draws never jointly observed can be priced (Section 5.2). Task boundaries in production work are less crisp than a benchmark instance, and cascades here are sequential. An operator who cares about latency would run attempts in parallel, which is a different objective.

The task population. The tasks come from public repositories that predate the models' training. A memorized fix appears as a cheap, consistent success that a production workload would not contain, and OpenAI (2026) argues that contamination and test design now limit SWE-bench Verified as a measure of frontier capability. The direction of the resulting bias on the value of restarts is not known. It bears more on comparisons across configurations than on the comparisons within a configuration that the primary result concerns, but it bears on both.

The outside option. The outside option is the benchmark's annotated time at a swept wage. Annotators'

estimates understate how long engineers take on the shortest tasks (Kwa et al. 2025), and the bucket-specific correction, a registered sensitivity, widens the range of the break-even with an automated verifier to 1.8 to 8.8 median attempts without changing which way the margin runs or whether its interval clears zero. An engineer's time is also not the whole cost of an escalated task: coordination, delay and review of the engineer's own fix are left out, all of which make the outside option larger and, with an automated verifier, the cap less attractive still.

Prices. Costs are list prices on each run's first day. Prices change often, and the logged costs of three configurations departed from the published schedules. A uniform change in prices leaves every break-even in multiples of the median attempt where it is and moves only its dollar value, but a change in relative prices, such as a cheaper cached input, would move it. Qwen3 Coder's price is imputed. A lower third-party price, about a third of the imputed one, moves Qwen's margins on the dollar axis, since a cheaper attempt makes each wage a larger multiple of it, and leaves its break-evens in multiples unchanged.

The state rule. The state-aware class is an eight-coefficient rule registered in advance, and its commit-to-termination approximation biases it toward restarting. A rule showing no margin over the best schedule cannot be distinguished from an approximation that lost the margin. The exact dynamic program is computed only for synthetic attempts whose state is the rule's own, where it shows the direction of the bias, not for the logged attempts.

Inference. Intervals are percentile intervals from a task-level bootstrap, 1,000 replicates for the primary result and for the fitted steps at the three wages where the plan reads them, and 100 for the comparators, the cascade's switching margin and the two-stage transfer, whose endpoints are correspondingly coarse. A margin is called distinct from zero when its interval excludes zero, cell by cell, without adjustment for the number of cells, and the cells of a sweep are highly dependent, so a count of cells on one side of zero describes the sweep and tests nothing. What the primary result rests on is not that count but the size of the point estimates, the intervals that straddle zero at every human wage, the nine perturbations that leave the pattern in place, and the share of long-running attempts that go on to succeed. A synthetic population checks that the primary interval covers at close to its nominal rate (Appendix E).

10. Conclusion

Work on stopping agents early asks how much compute a stopping rule saves for a given loss in success. This paper priced the loss. Once a success that a cutoff throws away has to be bought back from a fallback that costs money, the economics of restarting turn on the ratio of the fallback's cost to an attempt's cost, and on these logs that ratio settles the question. When a failed task goes to an engineer and the tests verify a patch for free, the ratio is in the tens to hundreds, retrying pays at every wage from $50 an hour up, and choosing a cap on attempt length adds nothing that can be told from zero, because the attempts a cap would stop still succeed a third to a half of the time and, in an unregistered check, the best cap that hindsight can find saves a few cents. A cap pays only when the fallback is worth fewer than two to four attempts, which is about where running the agent at all stops paying. When each patch must be reviewed at a large share of the fallback's cost, the economics reverse: the cap saves in point estimate at every wage, distinctly from zero only at the lower wages, mostly by keeping long attempts out of review, and the best such policy costs about what escalating every task costs. Both comparisons with escalating every task are exploratory. The registered low-complexity state rule adds little, and switching configurations after a failure helps in point estimate but is limited by how often the configurations fail together.

For an operator, the practical question is not how aggressively to stop bad runs but what another attempt,

a verification and an escalation cost relative to one another. The ordering that follows is to choose the configuration, retry, and set the iteration limit as a guard rather than a lever, then to price review before deciding whether to run the agent at all. For research on agent efficiency, the estimand suggests reporting cost per task with the fallback priced, rather than tokens saved or resolution lost on their own, since the trade between them is set by the fallback.

Appendix A. The Registered Sensitivities

Each sensitivity changes one input and reruns the ladder, the break-even and the cascade with every choice redone. The cap's margin keeps its 1,000-replicate interval under every sensitivity; the fitted steps are scored as point estimates. The nine are:

- **False accepts at $\phi = 0.34$ and $\phi = 0.50$.** That share of accepted patches is charged the outside option later, and every policy is re-selected with the charge in its objective (Section 3.5).
- **A common 100-call horizon.** Every configuration is cut at 100 calls, with cutoffs up to 95, and an attempt still running at 100 is scored as a failure.
- **All tasks.** Every task with a usable draw is scored, each policy on the tasks that can fill it, rather than only the tasks with four usable draws. The two sides of a margin can then rest on different tasks.
- **Refusals excluded, evaluations without a verdict counted as failures, and re-runs dropped.** The three changes to the attempt pool of Section 4.2.
- **The lowest third-party price for Qwen3 Coder**, $0.22 per million input tokens and $1.80 per million output tokens. Every configuration is rescored, since Qwen's attempts are among those the others' transfer rules are fitted on.
- **The bucket-specific correction to the annotated times.** Kwa et al. (2025, Table 8) timed seven baseline runs on six tasks: four tasks from the shortest bucket took a geometric mean of 32.9 minutes rather than 3.9, and two from the 1 to 4 hour bucket averaged 131.6 minutes rather than 120. The correction factor is interpolated in the logarithm of annotated minutes for the 15 minute to 1 hour bucket, giving 75.1 minutes, and the factor measured at 1 to 4 hours is held above 4 hours, giving 526.4.

Table A1 gives the break-even under each, Table A2 the primary margins, and Table A3 the two-stage bootstrap. Section 7.6 summarizes them.

One check that the plan did not register runs beside the nine and is marked as such in the tables: every configuration scored on the 275 tasks with four usable draws in all seven, the cascade's tasks, so that a median across configurations summarizes one task set rather than seven. The primary ladder scores each configuration on its own tasks with four usable draws, 405 to 500 of them. On the common tasks the automated break-evens run from 2.0 to 5.2 median attempts, a median of 2.3 against the primary's 2.5, and the cap's margin at $100 an hour is −$0.73 to −$0.00, a median of −$0.41 against −$0.24, with every interval straddling zero. Under review at 0.5 the cap still saves for six configurations at every wage, from $0.50 to $3.98 per task at $100; for GPT-5 it turns to a cost of $0.09 to $0.61 per task from $75 an hour up, with an interval that straddles zero by several dollars either side. The transfer's median margin is $0.18 at $100 against $0.12. The common tasks are the ones every configuration completed four times, which leaves out the tasks on which the three configurations that suffered outages lost draws, so they are not a random subset; the check is that the medians quoted in the text do not depend on which task set they summarize.

Table A1. The break-even under each sensitivity. The median and, in brackets, the range across configurations of the first rate at which the cap's margin changes sign, among the configurations whose margin changes sign in the sweep, in dollars an hour and in multiples of the median attempt cost. A crossing is supported when the interval excludes zero on both sides of it. Every crossing is in `results/sensitivity_breakeven.csv`. * Not registered: a check added after the results were in.

sensitivity	regime	cross in sweep	with a supported crossing	first break-even, $/h	x median attempt
primary	automated verifier	7 of 7	0	4.51 [1.74, 7.08]	2.54 [2.14, 3.83]
	review at 0.1 H	7 of 7	0	6.73 [2.18, 8.87]	3.18 [2.68, 5.90]
	review at 0.3 H	6 of 7	0	12.59 [4.71, 22.82]	6.49 [4.98, 10.76]
	review at 0.5 H	0 of 7	0		
phi 0.34	automated verifier	7 of 7	0	6.81 [2.63, 10.71]	3.83 [3.24, 5.79]
	review at 0.1 H	6 of 7	0	11.27 [3.83, 17.29]	5.38 [4.62, 12.50]
	review at 0.3 H	2 of 7	0	97.09 [96.10, 98.08]	38.09 [29.65, 46.53]
	review at 0.5 H	0 of 7	0		
phi 0.50	automated verifier	7 of 7	0	9.03 [3.42, 14.19]	4.98 [4.29, 7.68]
	review at 0.1 H	7 of 7	0	14.43 [6.01, 24.00]	7.97 [6.99, 12.50]
	review at 0.3 H	0 of 7	0		
	review at 0.5 H	0 of 7	0		
common horizon 100	automated verifier	7 of 7	2	4.56 [1.74, 9.31]	2.73 [2.54, 3.30]
	review at 0.1 H	7 of 7	2	5.71 [2.18, 11.44]	3.30 [2.68, 4.13]
	review at 0.3 H	7 of 7	0	11.14 [4.71, 73.31]	6.73 [4.62, 53.06]
	review at 0.5 H	2 of 7	0	100.76 [67.76, 133.76]	49.54 [40.80, 58.28]
all tasks	automated verifier	7 of 7	0	4.52 [1.74, 7.08]	2.49 [2.14, 4.29]
	review at 0.1 H	7 of 7	0	5.55 [2.22, 10.11]	3.06 [2.68, 7.12]
	review at 0.3 H	7 of 7	0	10.51 [4.80, 25.37]	6.13 [5.08, 17.86]
	review at 0.5 H	2 of 7	0	138.64 [53.10, 224.17]	90.65 [45.66, 135.64]
refusals excluded	automated verifier	7 of 7	0	4.51 [1.68, 7.08]	2.44 [2.14, 3.83]
	review at 0.1 H	7 of 7	0	6.73 [2.14, 8.87]	3.12 [2.68, 5.90]
	review at 0.3 H	6 of 7	0	12.59 [4.63, 22.82]	6.43 [4.98, 10.76]
	review at 0.5 H	0 of 7	0		
no verdict unresolved	automated verifier	7 of 7	0	5.30 [1.74, 7.53]	2.54 [2.14, 6.48]
	review at 0.1 H	7 of 7	0	7.40 [2.18, 8.87]	3.18 [2.68, 6.61]
	review at 0.3 H	6 of 7	0	12.60 [4.71, 22.82]	6.49 [4.98, 10.76]
	review at 0.5 H	0 of 7	0		

sensitivity	regime	cross in sweep	with a supported crossing	first break-even, $/h	x median attempt
re-runs dropped	automated verifier	7 of 7	0	4.35 [1.74, 11.35]	2.31 [2.14, 7.82]
	review at 0.1 H	6 of 7	0	7.21 [2.19, 14.49]	3.03 [2.63, 9.98]
	review at 0.3 H	7 of 7	0	18.79 [4.72, 127.87]	6.86 [4.80, 88.08]
	review at 0.5 H	1 of 7	0	551.72	271.66
Qwen lowest price	automated verifier	7 of 7	0	3.90 [1.74, 7.08]	2.54 [2.14, 3.83]
	review at 0.1 H	7 of 7	0	5.54 [2.18, 8.87]	3.18 [2.68, 5.90]
	review at 0.3 H	6 of 7	0	10.23 [4.71, 18.11]	6.49 [4.98, 10.76]
	review at 0.5 H	0 of 7	0		
METR minutes	automated verifier	7 of 7	0	1.77 [0.66, 5.52]	2.09 [1.84, 8.75]
	review at 0.1 H	7 of 7	0	2.13 [0.80, 4.41]	2.55 [2.22, 6.99]
	review at 0.3 H	7 of 7	0	3.66 [1.42, 10.59]	4.53 [3.62, 12.74]
	review at 0.5 H	3 of 7	0	26.27 [17.47, 46.34]	35.11 [11.60, 49.22]
common tasks*	automated verifier	7 of 7	0	4.26 [1.56, 6.84]	2.31 [1.99, 5.18]
	review at 0.1 H	7 of 7	0	6.15 [1.95, 15.91]	2.95 [2.49, 12.04]
	review at 0.3 H	7 of 7	0	9.73 [3.98, 107.98]	5.79 [4.71, 81.71]
	review at 0.5 H	1 of 7	0	70.37	106.27

The two-stage bootstrap resamples each task's usable draws as well as the tasks, on the same task resamples as the primary interval, so the two differ only by the second stage. For a mean over tasks the second stage would count within-task variation twice and widen the interval. Policy value is not such a mean: a retry draws a task's attempts without replacement, so a replicate that holds one draw twice prices a retry that can meet the same attempt again. The two bootstraps therefore need not be centered alike, and Table A3 gives each one's mean beside its interval. For the cap's margin the two intervals have about the same width, a median ratio of 1.02 over the 532 cells of the sweep, and the same center at the median, though in individual cells the centers differ by as much as $2. For the transfer the two-stage intervals, from 100 replicates, are narrower than the task-level ones from 1,000 in 20 of the 21 cells; GPT-5's margin stays distinct from zero at $100 and $300 an hour and Sonnet 4's at $300 does not.

Table A2. The primary margins under each sensitivity. Medians across configurations, dollars per task; positive is what the richer policy saves. Cap: step ii minus step iii, with an automated verifier (A) and under review at 0.5 of the outside option (R), at $25, $100 and $300 an hour. Transfer: step iii-b minus the state rule fitted on the other configurations, automated verifier. Switching: the best single configuration minus the cascade, automated verifier. * Not registered: a check added after the results were in.

sensitivity	cap A $25	cap A $100	cap A $300	cap R $25	cap R $100	cap R $300	transfer $25	transfer $100	transfer $300	switching $100
primary	−0.01	−0.24	−0.64	1.31	1.85	4.22	0.01	0.12	0.36	2.03
phi 0.34	−0.01	−0.07	−0.43	3.23	10.20	28.50	0.00	0.08	0.24	1.66
phi 0.50	−0.00	−0.05	−0.34	4.26	13.69	40.19	0.01	0.07	0.18	1.34
common horizon 100	0.00	0.00	0.00	1.04	1.07	1.98	0.00	0.00	0.00	4.42
all tasks	−0.00	−0.24	−0.64	1.26	1.85	4.22	0.00	0.12	0.39	4.40
refusals excluded	−0.01	−0.24	−0.64	1.31	1.85	4.22	0.01	0.12	0.36	4.10
no verdict unresolved	−0.01	−0.24	−0.64	1.32	1.85	4.22	0.01	0.12	0.36	2.01
re-runs dropped	−0.01	−0.25	−0.67	1.07	1.87	4.53	0.00	0.16	0.45	1.00
Qwen lowest price	−0.01	−0.24	−0.64	1.00	1.85	4.22	0.01	0.12	0.36	1.93
METR minutes	−0.06	−0.39	−1.27	0.31	0.37	0.88	0.07	0.30	0.91	5.48
common tasks*	−0.02	−0.41	−1.14	1.13	1.10	3.09	0.02	0.18	0.54	2.03

Table A3. The two-stage bootstrap. For the cap's margin in both regimes and the primary transfer at the three wages with fitted intervals: the point estimate, the task-level interval and the two-stage interval, each with the mean of its replicates for the cap; for the transfer the task-level mean is not recorded. The transfer's task-level interval is the one in Table 6, from 1,000 replicates; its two-stage interval is from 100. † Imputed price.

margin	configuration	$/h	estimate	one-stage [95%]	one-stage mean	two-stage [95%]	two-stage mean
cap, automated verifier	GPT-5	25	−0.10	[−0.38, 0.01]	−0.11	[−0.28, 0.01]	−0.06
		100	−0.41	[−1.62, 0.01]	−0.50	[−1.22, 0.01]	−0.32
		300	−1.21	[−4.84, 0.01]	−1.49	[−3.65, 0.01]	−0.95
	GPT-5.2	25	−0.01	[−0.08, 0.02]	−0.02	[−0.07, 0.02]	−0.01
		100	0.01	[−0.43, 0.09]	−0.06	[−0.41, 0.06]	−0.09
		300	0.02	[−1.50, 0.06]	−0.24	[−1.47, 0.07]	−0.32
	Sonnet 4	25	−0.00	[−0.06, 0.00]	−0.01	[−0.07, 0.00]	−0.01
		100	−0.01	[−0.32, 0.00]	−0.04	[−0.23, 0.01]	−0.04
		300	−0.01	[−0.73, 0.00]	−0.09	[−0.63, 0.01]	−0.08
	Sonnet 4.5	25	−0.00	[−0.03, 0.01]	−0.00	[−0.04, 0.01]	−0.01

margin	configuration	$/h	estimate	one-stage [95%]	one-stage mean	two-stage [95%]	two-stage mean
		100	−0.24	[−0.36, 0.02]	−0.05	[−0.25, 0.02]	−0.02
		300	−0.64	[−1.27, 0.01]	−0.19	[−1.19, 0.02]	−0.12
	Gemini 3 Pro	25	−0.00	[−0.10, 0.01]	−0.02	[−0.10, 0.00]	−0.02
		100	−0.11	[−0.48, 0.05]	−0.12	[−0.49, 0.04]	−0.11
		300	−0.36	[−1.45, 0.05]	−0.44	[−1.46, 0.04]	−0.36
	Kimi K2	25	−0.03	[−0.23, 0.13]	−0.04	[−0.20, 0.16]	−0.02
		100	−0.44	[−1.50, 0.21]	−0.44	[−1.38, 0.24]	−0.28
		300	−1.44	[−5.47, 0.15]	−1.52	[−4.25, 0.18]	−0.94
	Qwen3 Coder†	25	−0.04	[−0.52, 0.12]	−0.10	[−0.42, 0.05]	−0.08
		100	−0.74	[−1.58, 0.37]	−0.47	[−1.70, 0.25]	−0.34
		300	−2.34	[−4.76, 0.25]	−1.59	[−4.79, 0.17]	−0.95
cap, review at 0.5 H	GPT-5	25	1.00	[−0.24, 1.91]	0.82	[−0.26, 1.99]	0.80
		100	2.16	[−1.53, 6.20]	1.99	[−1.50, 6.30]	1.96
		300	5.48	[−4.88, 17.85]	5.68	[−4.80, 18.28]	5.42
	GPT-5.2	25	0.40	[−0.44, 1.22]	0.27	[−0.45, 1.26]	0.28
		100	0.75	[−2.02, 4.03]	0.70	[−1.92, 3.87]	0.76
		300	1.54	[−6.58, 11.09]	1.94	[−5.28, 11.01]	2.24
	Sonnet 4	25	0.99	[−0.50, 1.94]	0.77	[−0.53, 1.94]	0.75
		100	1.45	[−3.13, 3.92]	0.57	[−2.98, 4.10]	0.66
		300	3.90	[−10.56, 10.94]	0.45	[−9.70, 11.34]	0.99
	Sonnet 4.5	25	1.94	[1.06, 2.84]	1.94	[1.02, 2.81]	1.93
		100	1.85	[−1.86, 5.59]	1.38	[−2.16, 5.71]	1.38
		300	4.22	[−5.93, 15.12]	2.78	[−6.76, 15.85]	2.88
	Gemini 3 Pro	25	1.31	[−0.12, 2.44]	1.21	[−0.20, 2.42]	1.19
		100	1.64	[−2.36, 6.41]	1.32	[−2.39, 6.53]	1.32
		300	2.41	[−8.18, 15.89]	2.34	[−8.45, 15.81]	2.35
	Kimi K2	25	1.78	[0.24, 2.84]	1.61	[0.30, 2.84]	1.59
		100	3.30	[−1.48, 7.84]	3.63	[−1.51, 8.20]	3.68
		300	8.19	[−5.77, 21.69]	9.36	[−5.92, 22.60]	9.79
	Qwen3 Coder†	25	1.75	[0.66, 2.70]	1.66	[0.63, 2.77]	1.65
		100	3.56	[−0.36, 6.03]	3.05	[−0.75, 6.21]	3.00

margin	configuration	$/h	estimate	one-stage [95%]	one-stage mean	two-stage [95%]	two-stage mean
		300	10.15	[−2.16, 16.88]	7.88	[−3.38, 17.72]	7.76
transfer, automated verifier	GPT-5	25	0.09	[−0.01, 0.33]		[−0.00, 0.21]	0.06
		100	0.38	[0.05, 1.44]		[0.01, 0.91]	0.30
		300	1.13	[0.18, 4.32]		[0.04, 2.72]	0.90
	GPT-5.2	25	0.01	[−0.05, 0.10]		[−0.02, 0.07]	0.02
		100	−0.01	[−0.10, 0.47]		[−0.05, 0.37]	0.10
		300	−0.02	[−0.06, 1.50]		[−0.05, 1.30]	0.35
	Sonnet 4	25	0.00	[−0.13, 0.08]		[−0.01, 0.08]	0.02
		100	0.11	[−0.00, 0.33]		[−0.02, 0.26]	0.09
		300	0.08	[0.01, 0.76]		[−0.00, 0.58]	0.20
	Sonnet 4.5	25	0.00	[−0.07, 0.15]		[−0.00, 0.06]	0.01
		100	0.05	[−0.04, 0.65]		[−0.02, 0.58]	0.09
		300	0.15	[−0.03, 2.54]		[−0.02, 1.27]	0.32
	Gemini 3 Pro	25	0.00	[−0.11, 0.09]		[−0.02, 0.10]	0.02
		100	0.12	[−0.05, 0.51]		[−0.04, 0.49]	0.12
		300	0.36	[−0.04, 1.88]		[−0.04, 1.47]	0.41
	Kimi K2	25	0.03	[−0.16, 0.21]		[−0.18, 0.25]	0.03
		100	0.45	[−0.18, 1.52]		[−0.24, 0.99]	0.29
		300	1.44	[−0.14, 5.43]		[−0.13, 2.96]	0.97
	Qwen3 Coder†	25	0.06	[−0.08, 0.54]		[−0.08, 0.46]	0.08
		100	0.74	[−0.34, 1.59]		[−0.18, 1.65]	0.41
		300	1.74	[−0.23, 4.78]		[−0.15, 4.85]	1.11

Appendix B. Comparators and the Sample Oracle

The plan registers four comparators beside the ladder, each a family chosen on the training folds and scored on the held-out folds, and reports them in full without arguing from them.

- **The dollar cutoff.** A sensitivity family in the plan, fitted separately from the cap in calls: K attempts, each stopped at the first decision point at which its spend exceeds a cutoff in dollars, over 40 cutoffs spaced evenly in logarithm between the 5th percentile of spend at call 5 and the largest attempt spend on the training folds. It asks whether the primary result depends on stating the cap in calls.
- **The threshold comparator.** An empirical welfare maximization family (Kitagawa and Tetenov

2018): stop at decision point t when spend per call over the last five calls exceeds $a + bt$, with a at the 5th to 95th percentiles of that rate over running attempts on the training folds and b moving the threshold at the last decision point from a quarter of a to four times a. If the state rule cannot beat it, the regressions add nothing.

- **The universal schedule.** Cutoffs in the sequence 1, 1, 2, 1 of a unit (Luby, Sinclair and Zuckerman 1993), the first four terms of the universal sequence. The plan does not fix the unit, and a unit of one call would stop every attempt at once, so the unit is chosen on the training folds among the decision points whose double is also a decision point or reaches the configuration's cap. Its classical guarantee assumes unbounded restarts of a procedure that eventually succeeds and is not claimed to carry over.
- **The first look.** The state rule allowed to act only at the first decision point, which if nearly as good as the full rule would recommend one early look and a fixed rule thereafter.

With an automated verifier at $100 an hour, the cap stated in dollars behaves as the cap in calls does: its margin runs from −0.58 to +0.18 per task and every interval includes zero. Under review at 0.5, at the same wage, it saves $1.40 to $4.88, more than the cap in calls for six of the seven configurations. At $100 an hour with an automated verifier the state rule beats the threshold comparator for all seven, by $0.25 to $1.17, distinct from zero for Gemini alone (at $300 it is distinct from zero for two, and loses for Qwen3 Coder); since the rule does not stop attempts there, this says that the threshold family, chosen from data, costs more than not capping at all. Under review at 0.5 at $100 an hour the threshold comparator does better than the rule for six of the seven. At $100 an hour the universal schedule costs less than the searched schedule in point estimate for five of the seven with an automated verifier, by up to $0.39, and every interval includes zero, which is the searched schedule's cost of being chosen from data again. With an automated verifier at $100 and $300 an hour the first-look rule is the full rule, since neither stops attempts. Under review at 0.5 at $100 an hour, restricting the rule to the first decision point costs $0.69 to $4.50 more for four configurations and changes the other three by at most three cents, so one early look does not stand in for the full rule where the rule acts at all.

Figure A1 draws the state rule against the best schedule across the sweep under review at 0.5, the regime in which the rule stops attempts: fitted on the configuration itself it gains for three configurations at some points of the sweep, and fitted on the other six it gains in 3 of the sweep's 133 cells and loses by more than a cent in 97.

Leave-one-out essentialness. Removing one configuration from the cascade's choices and rechoosing shows which the cascade relies on. With an automated verifier, removing GPT-5.2 raises the cascade's cost by $1.48 per task at $100 an hour and $2.19 at $300, and removing Sonnet 4 by $0.96 and $2.05; at $25 GPT-5 and Sonnet 4 matter most, at $0.41 and $0.31. Removing Sonnet 4.5 or Gemini lowers the cascade's cost at $100, by $0.06 and $0.15, because the cascade chosen with them does worse out of sample. Under review at 0.5, only removing GPT-5.2 raises the cost by more than a cent, by $1.12 per task at $100 and $2.70 at $300.

The sample oracle. For each of the 275 common tasks, the oracle is the cheapest schedule of one to four (configuration, cutoff) attempts on that task's own draws, and its value is the average over tasks. No policy in the class the cascades are drawn from can beat it on these draws, but as an estimate of any population quantity it is biased downward, since it takes a minimum over seven configurations on four draws per task, so its gap to the best cross-fitted cascade is an upper estimate of what knowing the task in advance would be worth, not a value of information. The minimum is found exactly by a recursion that the tests hold to a brute-force search over every schedule.

Table A4. The registered comparators, automated verifier. Dollars per task; positive is what the richer or registered policy saves. Cap in calls: step ii minus step iii. Cap in dollars: step ii minus the dollar cutoff. Rule over threshold: the threshold comparator minus the state rule. Searched over universal: the universal schedule minus step iii-b. First look: step iii-b minus the first-look rule, beside the same margin for the unrestricted rule. Intervals from 100 replicates with every family refitted inside each. † Imputed price.

configuration	$/h	cap in calls	cap in dollars [95%]	rule over threshold [95%]	searched over universal [95%]	first look [95%]	rule, any point
GPT-5	25	−0.10	0.01 [−0.30, 0.05]	0.18 [−0.02, 0.89]	−0.02 [−0.17, 0.11]	0.09 [−0.00, 0.25]	0.09
	100	−0.41	0.01 [−1.21, 0.02]	0.78 [−0.02, 2.60]	−0.08 [−0.84, 0.06]	0.38 [0.07, 1.61]	0.38
	300	−1.21	0.01 [−3.63, 0.02]	2.40 [−0.02, 7.86]	−0.23 [−2.53, 0.16]	1.13 [0.22, 4.82]	1.13
GPT-5.2	25	−0.01	−0.01 [−0.07, 0.03]	0.15 [−0.07, 0.59]	0.00 [−0.07, 0.07]	0.01 [−0.05, 0.10]	0.01
	100	0.01	0.18 [−0.43, 0.27]	0.48 [−0.05, 2.12]	0.00 [−0.27, 0.04]	−0.01 [−0.05, 0.45]	−0.01
	300	0.02	0.04 [−1.49, 0.21]	2.59 [0.03, 7.11]	0.01 [−0.70, 0.02]	−0.02 [−0.10, 1.52]	−0.02
Sonnet 4	25	−0.00	−0.00 [−0.07, 0.00]	0.08 [−0.12, 0.32]	0.00 [−0.06, 0.11]	0.00 [−0.11, 0.07]	0.00
	100	−0.01	−0.01 [−0.37, 0.00]	0.42 [−0.03, 1.69]	−0.10 [−0.24, 0.03]	0.11 [−0.00, 0.32]	0.11
	300	−0.01	−0.01 [−0.89, 0.00]	1.25 [−0.02, 5.00]	−0.08 [−0.53, 0.01]	0.08 [0.01, 0.73]	0.08
Sonnet 4.5	25	−0.00	−0.06 [−0.10, 0.02]	0.03 [−0.15, 0.19]	−0.00 [−0.14, 0.12]	0.00 [−0.12, 0.15]	0.00
	100	−0.24	−0.41 [−1.22, 0.03]	0.47 [−0.30, 1.17]	0.03 [−0.65, 0.07]	0.05 [−0.02, 0.66]	0.05
	300	−0.64	−1.21 [−3.62, 0.03]	1.47 [−0.08, 3.57]	0.03 [−2.48, 0.05]	0.15 [−0.02, 2.49]	0.15
Gemini 3 Pro	25	−0.00	−0.02 [−0.22, 0.00]	0.37 [0.06, 0.62]	0.00 [−0.03, 0.13]	0.00 [−0.10, 0.08]	0.00
	100	−0.11	−0.12 [−1.76, 0.00]	1.17 [0.08, 2.69]	−0.03 [−0.37, 0.02]	0.12 [−0.04, 0.60]	0.12
	300	−0.36	−0.37 [−5.84, 0.00]	3.91 [1.19, 8.51]	−0.09 [−1.12, 0.02]	0.36 [−0.03, 2.68]	0.36
Kimi K2	25	−0.03	−0.05 [−0.28, 0.06]	0.05 [−0.04, 0.44]	0.00 [−0.08, 0.11]	0.03 [−0.10, 0.14]	0.04
	100	−0.44	−0.58 [−1.40, 0.30]	0.59 [−0.32, 2.72]	−0.13 [−0.36, 0.12]	0.45 [−0.27, 1.39]	0.45
	300	−1.44	−1.83 [−4.25, 0.10]	2.05 [−0.01, 6.91]	−0.36 [−1.07, 0.15]	1.44 [−0.11, 4.32]	1.44
Qwen3 Coder†	25	−0.04	−0.02 [−0.33, 0.13]	0.02 [−0.19, 0.28]	0.00 [−0.25, 0.14]	0.04 [−0.07, 0.53]	0.05
	100	−0.74	−0.31 [−1.32, 0.43]	0.25 [−0.28, 1.36]	−0.39 [−0.91, 0.11]	0.74 [−0.26, 1.74]	0.74
	300	−2.34	−1.11 [−4.17, 0.23]	−0.12 [−0.23, 3.43]	−0.59 [−2.56, 0.06]	1.74 [−0.23, 4.76]	1.74
median	25	−0.01	−0.02	0.08	0.00	0.01	0.01
	100	−0.24	−0.12	0.48	−0.08	0.12	0.12
	300	−0.64	−0.37	2.05	−0.09	0.36	0.36

With an automated verifier at $100 an hour the oracle costs $14.07 per task against $19.94 for the best cross-fitted cascade, a gap of $5.87, or 29 percent of the cascade's cost; across the three wages the gap is 27 to 33 percent, and under review at 0.5 at $100 it is 31 percent. Across the regimes and the three wages the gap is 27 to 37 percent of what the best cascade costs. Knowing each task in advance could be worth that much, far more than the cap, the schedule or the state rule is worth on these logs, although the gap is an upper estimate.

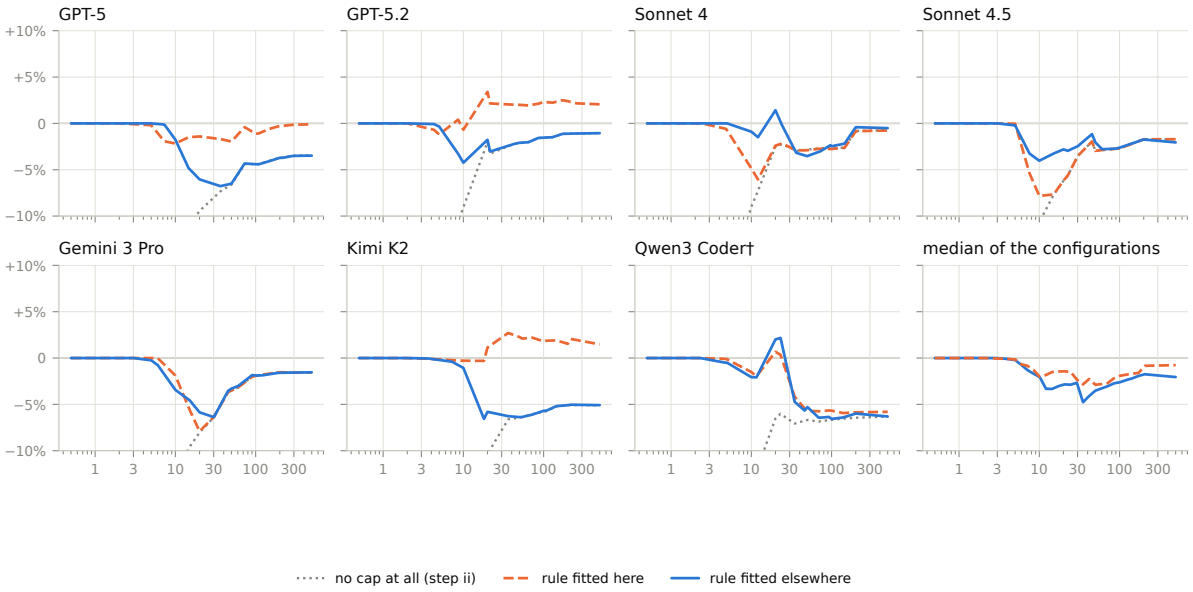

Figure A1. The state rule against the best schedule under review at 0.5 of the outside option. As Figure 3, not the registered regime for the transfer and without intervals: the rule fitted on the other six configurations (solid), on the configuration itself (dashed), and not capping at all (dotted), as a share of what retrying without a cap costs. Where the dotted line leaves the frame, not capping costs far more than the schedule. † Imputed price.

Table A5. The sample oracle. On the 275 common tasks, dollars per task: the oracle, the best cross-fitted cascade and single configuration on the same tasks, and the gap as a share of the cascade's cost. * Exploratory, not registered: every task sent to the outside option. † Imputed price.

regime	$/h	sample oracle	cascade	best single	escalate all*	cascade minus oracle	share of the cascade
automated verifier	25	3.99	5.98	6.42	12.51	1.99	33%
	100	14.07	19.94	21.97	50.05	5.87	29%
	300	40.90	56.01	61.36	150.16	15.11	27%
review at 0.1 H	25	5.02	7.97	7.93	12.51	2.95	37%
	100	18.13	28.35	30.29	50.05	10.22	36%
	300	53.01	82.26	90.73	150.16	29.25	36%
review at 0.3 H	25	7.01	10.32	10.32	12.51	3.30	32%
	100	26.18	39.19	39.19	50.05	13.01	33%
	300	77.17	116.43	116.17	150.16	39.26	34%
review at 0.5 H	25	8.85	12.88	12.88	12.51	4.03	31%
	100	33.78	48.84	48.84	50.05	15.06	31%
	300	100.16	146.44	145.44	150.16	46.29	32%

Appendix C. Difficulty, the Distribution of Cost, and the Spread

Difficulty. In the primary ladder no policy observes a task's annotation. Here the annotation is admitted to every policy at once: each is chosen within a difficulty bucket, and the cap's margin is reported per bucket with its interval. The longest bucket holds a handful of tasks, too few for a training fold to choose on, so it is shown by its count and also pooled with the 1 to 4 hour bucket. Pooling the choices made within the buckets and comparing them with the primary's blind choice gives what admitting the annotation is worth.

Table A6. The cap's margin by difficulty. Dollars per task at $100 an hour, every policy chosen within the bucket. * The 95 percent interval from 1,000 replicates excludes zero; every interval is in `results/difficulty.csv`. Over 4 hours: too few tasks for a training fold to choose on, with their count in parentheses; the two longest buckets are also shown together. All, within: the choices made under 15 minutes, from 15 minutes to 1 hour and at 1 hour or more, pooled. All, blind: the primary. Tasks per bucket: 158 to 194, 210 to 261, 30 to 42, 2 to 3 and 32 to 45. † Imputed price.

regime	configuration	under 15 min	15 min to 1 h	1 to 4 h	over 4 h	1 h or more	all, within	all, blind
automated verifier	GPT-5	−0.03	−0.20	−0.01*	(3)	−0.01	−0.12	−0.41
	GPT-5.2	−0.02	−0.00	−6.80	(2)	−6.38	−0.51	0.01
	Sonnet 4	−0.01	−0.00	0.00	(3)	0.00	−0.00	−0.01
	Sonnet 4.5	−0.01	−0.46	−9.68	(3)	−9.03	−1.06	−0.24
	Gemini 3 Pro	0.00	−0.23	−0.01	(2)	−0.01	−0.11	−0.11
	Kimi K2	−0.02	0.12	−5.21	(3)	−4.78	−0.37	−0.44
	Qwen3 Coder†	0.04	0.31	−8.98	(3)	−8.68	−0.60	−0.74
review at 0.5 H	GPT-5	−0.09	0.03	43.47*	(3)	62.83*	5.68	2.16
	GPT-5.2	−0.00	−0.45	26.01	(2)	38.62	2.81	0.75
	Sonnet 4	−0.01	−0.49	33.56	(3)	40.22	3.38	1.45
	Sonnet 4.5	−0.13	−0.13	43.39*	(3)	53.97	4.74	1.85
	Gemini 3 Pro	−0.01	−0.04	36.30	(2)	56.00*	5.03	1.64
	Kimi K2	0.16	0.68	33.41*	(3)	53.42	5.09	3.30
	Qwen3 Coder†	0.13	0.50	44.09	(3)	54.62	5.23	3.56

With an automated verifier, admitting the annotation changes little. At $100 an hour the cap's margin within the two short buckets stays within about half a dollar of zero, and pooled over the buckets the cheaper of steps ii and iii costs at most $0.41 per task less than when chosen blind, and for Gemini $0.03 more. Within the tasks annotated at an hour or more, the cap chosen within the bucket costs $4.78 to $9.03 per task for four configurations, the cost of choosing among caps on 32 to 45 tasks, though no interval excludes zero. The only cells in the paper where the interval puts the cap wholly on the cost side are here: in GPT-5's 1 to 4 hour bucket, 42 tasks, the cap costs about a cent per task with an automated verifier at every wage from $25 an hour, and more at three wages under review at 0.1.

Under review at 0.5 the long tasks are where the cap pays. At $100 an hour it saves $38.6 to $62.8 per task within the tasks annotated at an hour or more, distinct from zero for two configurations, and $26.0 to $44.1 within the 1 to 4 hour bucket, distinct from zero for three. Pooled over the buckets its saving is $2.81 to $5.68 per task against $0.75 to $3.56 when chosen blind, and admitting the annotation lowers the cost of the cheaper of steps ii and iii by $1.67 to $3.52 per task, 3 to 7 percent. Under review, what the cap saves is concentrated on the tasks that are most expensive to escalate.

The distribution of cost. A cap is partly insurance, and the mean alone undervalues it. The distribution of cost per incoming task is taken over tasks and, within a task, over the orderings of its draws the policy can use, each task weighing the same, and its quantiles are lower quantiles, so each is a cost some task and ordering actually incurs. The median-cost version of the primary comparison chooses steps ii and iii on the training folds by median cost and scores them by the median of the held-out costs.

Table A7. The distribution of cost per task at $100 an hour. Under steps ii and iii as chosen on mean cost: the mean, median and 95th percentile of cost per incoming task, and the cap's margin at the 95th percentile; and the median-cost version of the primary comparison. † Imputed price.

regime	configuration	ii mean	ii median	ii p95	iii mean	iii median	iii p95	cap's saving at p95	median version
automated verifier	GPT-5	23.23	0.50	201.30	23.64	0.50	201.44	−0.14	−0.00
	GPT-5.2	22.65	0.72	200.85	22.64	0.72	200.85	0.00	−0.00
	Sonnet 4	21.76	1.26	203.06	21.77	1.26	203.06	0.00	0.00
	Sonnet 4.5	23.89	2.06	66.87	24.12	2.06	64.81	2.06	0.00
	Gemini 3 Pro	24.03	1.23	202.47	24.14	1.23	202.47	0.00	0.00
	Kimi K2	25.26	1.21	202.85	25.69	1.21	203.00	−0.15	−0.00
	Qwen3 Coder†	23.55	1.39	203.31	24.30	1.39	204.82	−1.52	0.00
review at 0.5 H	GPT-5	53.14	25.29	300.41	50.97	50.15	200.29	100.11	0.00
	GPT-5.2	47.45	25.44	300.40	46.70	25.44	200.76	99.65	0.00
	Sonnet 4	50.62	25.85	301.08	49.18	25.85	201.08	100.00	−0.00
	Sonnet 4.5	52.69	26.48	301.77	50.85	50.06	201.29	100.48	−0.00
	Gemini 3 Pro	51.69	25.69	300.71	50.05	50.07	200.08	100.63	−0.01
	Kimi K2	54.56	25.56	300.83	51.25	50.25	200.46	100.37	0.00
	Qwen3 Coder†	52.75	25.84	301.49	49.19	25.84	201.12	100.37	−0.00

With an automated verifier at $100 an hour, the 95th percentile of cost per task is about $200 for six configurations, the outside option of a task annotated at 1 to 4 hours, and about $65 for Sonnet 4.5, and the cap moves it by between $2.06 down and $1.52 up. Under review at 0.5 the cap lowers the 95th percentile by about $100 for every configuration, from about $300 to about $200: in the tail, a long task that would have been attempted, reviewed at half its outside option and then escalated is escalated without the review. As insurance, then, the cap pays only where review is dear. It is not free insurance: for four configurations the cap chosen on mean cost also raises the median cost per task under review,

from about $26 to about $50, because it escalates the typical task, one annotated at 30 minutes whose outside option is $50, rather than attempting it and reviewing the patch. The median-cost version of the primary comparison is within about a cent of zero in every regime, since the median task is resolved cheaply whether or not the attempts are capped.

The spread. On the 275 common tasks, the range of policy value across the configurations, each at its best policy or at a single attempt, against the range across steps i to iii-b within a configuration: whether choosing the configuration or the policy moves cost more.

Table A8. Choosing the configuration against choosing the policy. On the 275 common tasks, dollars per task: the range across configurations, holding each at its best policy and at one attempt, and the median and largest range across steps i to iii-b within a configuration. † Imputed price.

regime	$/h	across configurations, best policy	across configurations, one attempt	within a configuration, median	within a configuration, largest	cheapest	dearest
automated verifier	25	1.76	1.47	0.24	0.78	GPT-5	Sonnet 4.5
	100	4.94	5.15	3.83	4.44	Sonnet 4	Kimi K2
	300	13.72	14.96	14.22	15.08	Sonnet 4	Kimi K2
review at 0.1 H	25	1.47	1.47	0.01	0.09	GPT-5.2	Kimi K2
	100	5.14	5.14	0.41	1.01	GPT-5.2	Kimi K2
	300	14.96	14.93	1.64	2.86	Sonnet 4	Kimi K2
review at 0.3 H	25	1.53	1.46	0.10	0.35	GPT-5.2	Sonnet 4.5
	100	5.40	5.12	0.15	0.31	GPT-5.2	Kimi K2
	300	15.76	14.88	0.58	1.07	GPT-5.2	Kimi K2
review at 0.5 H	25	0.34	1.46	1.13	1.77	Qwen3 Coder†	Sonnet 4
	100	2.69	5.10	1.10	3.98	GPT-5.2	GPT-5
	300	8.37	14.83	3.09	11.44	GPT-5.2	GPT-5

With an automated verifier at $100 an hour, the range across configurations at their best policies is $4.94 per task, against a median range across policies within a configuration of $3.83 and a largest of $4.44. At the three wages of Table A8 the choice of configuration mostly moves cost more than the choice of policy, by nine times or more the median range within a configuration under review at 0.1 and 0.3, where the configurations seldom retry. The exceptions are the top of the automated sweep, where retrying makes the two about equal ($13.72 across against a median of $14.22 within at $300), and review at 0.5 at $25 an hour, where whether to run the agent at all is the larger choice.

Appendix D. Diagnostics

The share of the variance in log spend that is within task, and the share of tasks with mixed outcomes across their four runs, motivate restarts but do not bound their value, which depends jointly on the cost and success distributions, their dependence, the outside option, K and the cutoff.

Table A9. Within-task variation. Per configuration, on its tasks with four usable draws: the share of the variance in log spend that is within task, the shares of tasks with mixed outcomes, with all four draws failing and with all four resolving, the median attempt cost, and the share of attempts whose spend passes their own task's outside option at $25 an hour. † Imputed price.

configuration	tasks	within-task share of log-spend variance	mixed outcomes	all four fail	all four resolve	median attempt, $	attempts past their outside option at $25/h
GPT-5	496	64%	24%	30%	46%	0.345	0.05%
GPT-5.2	405	30%	11%	28%	61%	0.556	0.68%
Sonnet 4	498	29%	19%	22%	58%	1.039	4.82%
Sonnet 4.5	500	24%	15%	28%	56%	1.661	14.50%
Gemini 3 Pro	410	43%	22%	30%	48%	0.815	3.54%
Kimi K2	412	33%	25%	31%	44%	0.698	3.82%
Qwen3 Coder†	500	26%	23%	24%	53%	1.085	5.75%

The four runs of a task differ widely in what they cost, and 24 to 64 percent of the variance of log spend is within task, the most for GPT-5. They differ much less in whether they succeed. Only 11 to 25 percent of tasks have mixed outcomes across their four runs, while 22 to 31 percent fail on every run and 44 to 61 percent resolve on every run. All of the value of retrying comes from the mixed tasks; on the tasks where every run fails, a retry adds an attempt that cannot succeed, which is cheap when an attempt costs a dollar and the outside option fifty. Long attempts are not hopeless. In the six configurations capped at 500 calls, an attempt still running at 100 calls goes on to resolve 33 to 55 percent of the time, against 58 to 69 percent at the start (Figure A3), which is why a cap cuts successes as well as spend. The share of attempts whose spend passes their own task's outside option at $25 an hour runs from 0.05 percent of GPT-5's to 14.5 percent of Sonnet 4.5's, whose attempts cost most.

Whether switching after a failure can pay depends on how often the configurations fail together. Their outcomes are highly correlated (Figure A2): across the 275 common tasks the share of a configuration's draws that resolve is correlated with another's at 0.62 to 0.90.

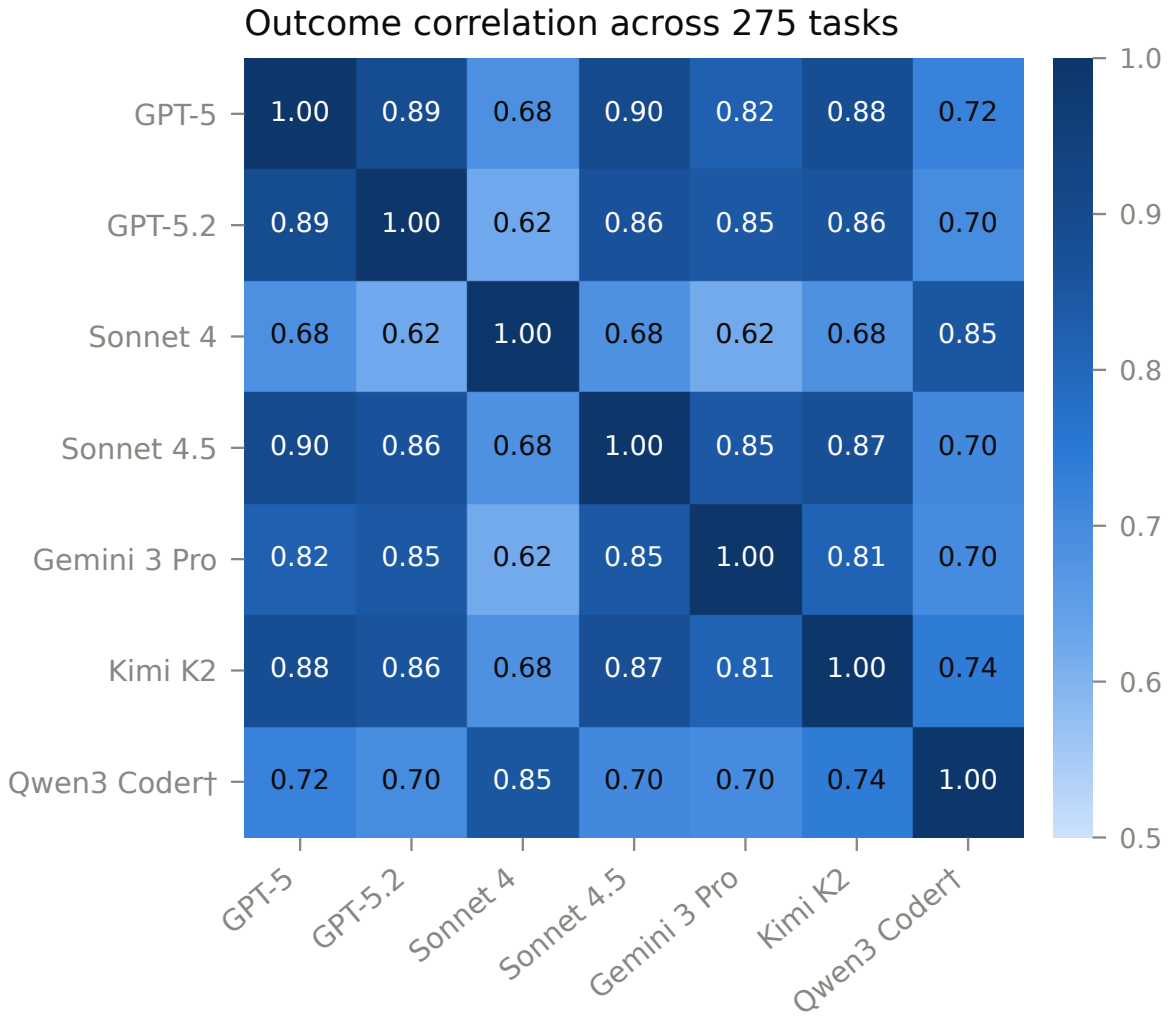

Figure A2. Outcome correlation across configurations. The correlation, across the 275 common tasks, of the share of each configuration's draws that resolve. It is the portfolio predictor of whether switching after a failure can pay. † Imputed price.

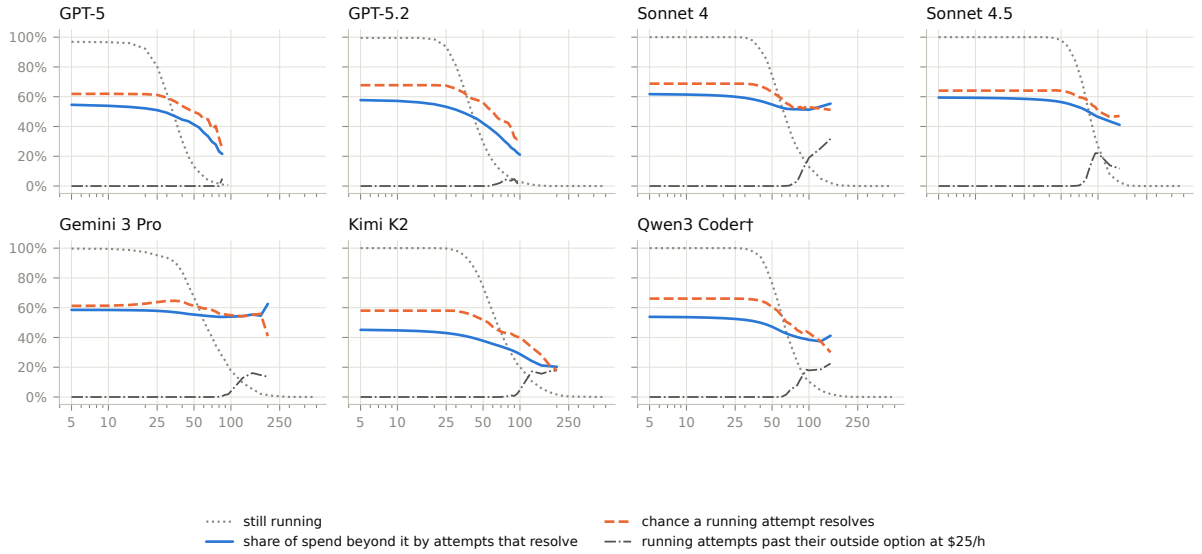

Figure A3. Tail composition. For each cutoff on the decision grid: the share of attempts still running (dotted); of the spend incurred beyond the cutoff, the share by attempts that go on to resolve (solid), which is what a cutoff there would cut from successes; the chance that an attempt still running there resolves (dashed); and the share of running attempts whose spend has already passed their own task's outside option at $25 an hour (dash-dot), which marks where on the grid a cutoff at the annotated engineer time would fall. The last three are drawn while at least 1 percent of attempts are still running. † Imputed price.

Appendix E. Verification

Coverage of the primary interval. A synthetic population of 12,000 tasks has what makes the real pools hard: tasks that differ in difficulty and share it across their draws, attempts whose chance of resolving falls with their length, calls that grow dearer as context accumulates, and an outside option drawn from the benchmark's four buckets in about their proportions. Its median attempt costs $0.45 and its attempts resolve 48 percent of the time. What the cross-fitted cap margin estimates is the population value of the policies the procedure chooses from a training set, the target, which is computed by choosing on many independent samples and scoring each choice on the whole population. Table E1 gives, at four settings, the target, the best policy in the population, the bias of the point estimate on samples of 500 tasks, and how often the registered interval covers the target over 120 samples.

Table E1. Coverage of the registered interval on a synthetic population. The cap's margin in dollars per task. Target: the population value of the procedure's choice. Best: the margin of the best policies in the population. Bias: the mean point estimate minus the target, with its standard error.

Outside option	Verifier	Target	Best	Bias (s.e.)	Mean width	Covers
1 median attempt	automated	+0.464	+0.464	+0.003 (0.003)	0.111	93%
5 median attempts	automated	+0.374	+0.386	−0.006 (0.008)	0.417	98%
50 median attempts	automated	−0.029	+0.043	+0.015 (0.022)	1.022	98%
20 median attempts	review at 0.5	+1.527	+1.612	−0.020 (0.031)	1.435	99%

The point estimate is no more than one standard error from the target at every setting, and the interval covers at 93 to 99 percent. At 50 median attempts the target is negative while the best policy's margin is positive: the cost of choosing a cap from data exceeds what the best cap would save, which is the pattern the real data show above the break-even.

The state rule against the exact optimum. Where an attempt's prospects are fully described by the rule's own state, the optimal restart policy can be computed by backward induction. The synthetic attempts are of two kinds that they do not reveal: a good attempt almost always resolves when it finishes, a bad one seldom does and runs longer and writes more. In each interval of five calls an attempt writes a high or low volume of output, pays for it and finishes with a chance that depends on its kind, so the rule's three state variables determine how many high-output intervals the attempt has had, a sufficient statistic for its kind. Tasks are alike, so the restart values' other weakness, being averages over tasks, does not arise. The rule is fitted on 2,000 simulated tasks and every policy is scored on another 8,000. Table E2 gives the result for signals of two strengths.

The optimum's own policy, replayed by the evaluator on the held-out tasks, reproduces its computed value to within 1 percent in every setting, which checks the evaluator against an independent calculation. The rule stops more first attempts than the optimum in every setting where the two differ, and at the same call or earlier: the commit-to-termination approximation's bias toward restarting outweighs the opposite pull of its restart values. With an automated verifier it costs 7 to 32 percent more than the optimum, and in three of the six automated settings more than the best constant cutoff. Under review it still beats the best constant cutoff at the two higher outside options, where stopping early is valuable enough that reading the state pays despite the bias.

Table E2. The state rule against the exact optimum on synthetic attempts. Dollars per task. Signals weak: a good attempt has a high-output interval with probability 0.3 and a bad one 0.6; strong: 0.2 and 0.8. Stopped: the share of first attempts each policy stops before their end, and the mean call at which it does.

Signals	Outside option	Verifier	Optimum	Rule	Rule above optimum	Best constant cutoff	No cap	Stopped: optimum	Stopped: rule
weak	2	automated	1.220	1.302	6.8%	1.511	1.771	65% at 12	78% at 10
weak	10	automated	2.150	2.842	32.2%	2.636	2.636	42% at 29	65% at 8
weak	50	automated	5.420	6.198	14.3%	5.642	5.642	29% at 42	42% at 5
weak	2	review at 0.5	1.579	1.579	0.0%	1.579	2.531	92% at 5	92% at 5
weak	10	review at 0.5	6.514	6.967	7.0%	7.458	8.984	67% at 14	81% at 10
weak	50	review at 0.5	28.954	32.692	12.9%	33.158	41.248	57% at 23	81% at 10
strong	2	automated	0.952	1.068	12.2%	1.483	1.819	57% at 8	72% at 8
strong	10	automated	1.751	2.236	27.7%	2.762	2.762	47% at 15	63% at 7
strong	50	automated	5.034	6.079	20.7%	5.689	5.689	42% at 23	52% at 7
strong	2	review at 0.5	1.543	1.544	0.1%	1.544	2.559	92% at 5	92% at 5
strong	10	review at 0.5	5.717	6.335	10.8%	7.301	8.842	60% at 12	76% at 10
strong	50	review at 0.5	25.694	29.337	14.2%	32.406	40.257	60% at 12	76% at 10

Other checks. Hand-worked cases check policy cost to the cent for a task whose four draws all fail, one where all resolve, one where one resolves, and one with an attempt stopped at the harness's limit; the schedule and cascade searches and the oracle's recursion are checked against brute-force searches on small cases; and the faster route by which the sensitivities' intervals are computed is checked against the ladder's estimate on every row. Against the release, the extraction reproduces what Bai et al. (2026) state in numbers: Sonnet 4.5 and Kimi K2 consume 3.14 and 2.19 million more tokens per execution than GPT-5, against their statement of over 1.5 million, and their GPT-5 price schedule is the one validated here. The code's tests run on every change.

Appendix F. The Plan and the Choices It Left Open

The analysis plan was frozen at the `plan-frozen` tag of the repository after the data audit and registered on OSF Registries (osf.io/pdmw9) on 20 September 2026 under the Secondary Data Preregistration template, both before any policy was evaluated. Everything the audit found that the plan had not anticipated was recorded in the plan before the freeze: the configurations admitted and excluded, the attempt pool, each configuration's own iteration cap, the unequal numbers of usable draws, the price schedules, the state rule's class, and the value of ϕ. The paper departs from the frozen plan in four places, each a matter of what is reported rather than of what is computed, and each is stated here so that the plan and the paper can be read against each other.

- **Intervals for the fitted steps.** The plan asks for a 95 percent interval on every quantity chosen from

data and allows a reduced replicate count where refitting inside every replicate is too expensive. Steps iii-b, iv and iv-t carry intervals only at $25, $100 and $300 an hour with an automated verifier, where the plan reads the primary transfer; elsewhere they are point estimates, which the plan did not provide for. The cap's margin and the value of retry carry their intervals at every point of the sweep in every regime.

- **The cascades' tasks.** The plan says that cascades enumerate all available draw combinations and are unbiased at any count, and also that comparisons across attempt budgets rest on the tasks with four usable draws in every configuration involved. The second rule is applied: step v, the leave-one-out, the sample oracle and the spread are scored on the 275 tasks that all seven configurations cover with four draws. Those tasks are not a random subset of the 500 (Appendix A).
- **Where the diagnostics and the flags sit.** The plan places the diagnostics beside the results they explain and the configuration's cap and harness version in every table; the diagnostics are in Appendix D, cross-referenced from Section 8.1, and the cap and harness version are in Table 1.
- **Two summaries given in prose or in the results files rather than as exhibits.** The rates at which the chosen attempt budget and cutoff change over the sweep are described in Sections 7.1 and 7.2 and recorded in the `choice_*` columns of `results/ladder.csv`; the 95th percentile of cost is given for steps ii and iii (Table A7) rather than for every step, with the rest in `results/distribution.csv`.

Where the plan left a detail open, the choice made is stated here.

- **Sign.** The plan defines the cap's marginal value as step iii minus step ii in cost. The paper reports the same quantity with the opposite sign, step ii minus step iii, so that a positive margin is a saving throughout.
- **The annotation correction above 4 hours.** The plan interpolates the correction for the buckets without measured baselines. Above 4 hours there is nothing to interpolate toward, and the factor measured at 1 to 4 hours is held (Appendix A).
- **The universal schedule's unit** is chosen on the training folds, since the plan does not fix it (Appendix B).
- **The difficulty display** shows all four buckets, with the smallest shown by its count only and pooled with the next, and the pooled choices are those made under 15 minutes, from 15 minutes to 1 hour and at 1 hour or more (Appendix C).
- **Quantiles and the median version** are defined as in Appendix C.
- **The two-stage bootstrap** uses the same task resamples as the primary and reports both centers without correcting either (Appendix A).
- **Break-evens** are located by a scan in the logarithm of the rate refined by bisection. Table 4 gives the first crossing per configuration and regime, and every crossing the scan found is in Table A11, as the plan requires.
- **The tail composition** marks the annotated outside option on the decision grid by the share of running attempts whose spend has already passed it at $25 an hour, since the outside option differs by task.
- **The first figure.** The plan asks that the abstract and the first figure be built from the primary result and the primary transfer. Figure 1 draws the whole ladder from the registered values of Table 5, each step as a share of step i, and the primary result and the primary transfer are two of its steps; Figures 2 and 3 then give each its own figure. It adds a display, not a quantity.
- **Replicates for the fitted steps.** The plan allows fewer than 1,000 replicates where refitting inside every replicate is too expensive, and asks that the count be reported. The fitted steps' intervals

at $25, $100 and $300 an hour with an automated verifier come from 1,000 replicates, each refitting every model and choice, computed first at 100 and then extended to 1,000 on the same task resamples, which changed no conclusion; the comparators' intervals (Appendix B), the cascade's switching margin and the two-stage transfer are from 100. The value of retry's interval comes from the same bootstrap and the same resamples as the cap's margin.

- **The common-tasks check** in Appendix A was added after the registered results were in, and is marked as unregistered where it appears.
- **The best cap chosen with hindsight** (Sections 7.2 and 8.1, `cap_margin_in_sample` in `results/ladder.csv`), the in-sample margin of the capped family over the uncapped one, is not in the plan, whose only hindsight quantity is the per-task sample oracle of Appendix B. It is reported as an unregistered check on the size of the gain any cap could find.

Several exhibits were added after the registered results were in, and each is marked as exploratory where it appears: escalating every task without running the agent as a reference line (Tables 5, 7, A5 and A10, Figure 4) and the comparisons of the best capped policy with it under review (the abstract, Sections 1, 7.2, 8.3 and 10); the outside option at which retrying first beats escalating (Figure 2 and Section 8.2); the best cap chosen with hindsight, above; and the break-even in full attempts (Figure 5). The argument that the break-even in multiples does not depend on the level of token prices (Section 8.1) is arithmetic on the estimand rather than a result.

Appendix G. The Ladder under Review at 0.1 and 0.3, and Every Crossing

The plan reports every result under an automated verifier and under review at 0.1, 0.3 and 0.5 of the outside option. The text and the main exhibits read the automated verifier and review at 0.5, the two regimes that bracket the answer; Tables 4, 7, A1, A5 and A8 carry all four. Table A10 gives the ladder of Table 5 under review at 0.1 and 0.3, and Table A11 lists every crossing of the cap's margin the scan found, where Table 4 gives only the first. Every number here is in `results/ladder.csv` and `results/breakeven.csv`.

Table A10. Policy value under review at 0.1 and 0.3 of the outside option. As Table 5, with each patch submitted charged a review of 0.1 (top) or 0.3 (bottom) of its task's outside option. Resolves: the share of tasks steps i, ii, iii and iii-b resolve without the outside option, in percent. * Exploratory, not registered: every task to the outside option. † Imputed price.

Configuration	$/h	i	ii	iii	iii-b	Cap's saving [95%]	Step iii chose	Resolves %	Escalate all*
Review at 0.1 H									
GPT-5	25	8.75	8.81	8.93	8.94	−0.13 [−0.32, 0.05]	1×85/none, 2×80/85	60/61/62/63	12.56
	100	33.75	33.81	33.55	33.45	0.25 [−1.46, 0.49]	2×80/85/none	60/65/66/66	50.22
	300	100.43	98.74	99.45	99.14	−0.72 [−4.26, 0.90]	2×80/85/none	60/66/66/66	150.67
GPT-5.2	25	7.65	7.65	7.64	7.64	0.01 [−0.09, 0.04]	1×125	67/67/67/67	11.96
	100	28.42	28.42	28.45	28.45	−0.03 [−0.65, 0.13]	1×125/150/175	67/67/67/67	47.84
	300	83.79	83.79	83.93	83.93	−0.14 [−2.27, 0.45]	1×125/150/175	67/67/67/67	143.53
Sonnet 4	25	8.59	8.59	8.60	8.60	−0.01 [−0.08, 0.01]	1×200/275	69/69/69/69	12.58
	100	30.50	30.50	30.85	30.84	−0.35 [−0.54, 0.29]	1×275, 2×150	69/69/71/71	50.31
	300	88.93	89.07	89.26	89.26	−0.19 [−2.19, 1.08]	1×275, 2×150/175	69/73/72/72	150.93
Sonnet 4.5	25	9.56	9.56	9.57	9.57	−0.01 [−0.03, 0.01]	1×175/200/225	64/64/64/64	12.56
	100	32.71	32.97	32.94	33.15	0.03 [−0.44, 0.24]	1×225, 2×175	64/67/67/67	50.22
	300	94.46	92.51	92.39	93.03	0.12 [−2.00, 1.25]	2×175	64/68/68/68	150.67
Gemini 3 Pro	25	8.89	8.89	8.89	8.89	−0.00 [−0.08, 0.00]	1×300/400	61/61/61/61	12.29
	100	32.47	32.93	32.95	32.95	−0.02 [−0.31, 0.04]	1×400, 2×300/400	61/63/63/63	49.17
	300	95.37	96.38	96.43	96.44	−0.06 [−0.93, 0.21]	1×400, 2×300/400	61/63/63/63	147.51
Kimi K2	25	9.39	9.39	9.42	9.42	−0.02 [−0.24, 0.07]	1×175/200/250/275	58/58/58/58	12.62
	100	34.47	34.91	35.14	35.14	−0.23 [−1.03, 0.92]	1×250/275, 2×125/175	58/62/61/61	50.48
	300	101.33	100.18	100.14	100.14	0.04 [−3.76, 2.87]	2×175/275	58/65/64/64	151.43
Qwen3 Coder†	25	9.31	9.31	9.33	9.33	−0.02 [−0.46, 0.04]	1×175/225/300	66/66/66/66	12.56
	100	32.66	32.69	32.85	32.85	−0.17 [−1.72, 0.16]	1×300, 2×175/300	66/71/71/71	50.22
	300	94.94	92.60	93.16	93.16	−0.56 [−5.71, 0.39]	2×175/300	66/73/72/72	150.67
Review at 0.3 H									
GPT-5	25	11.17	11.17	11.23	11.23	−0.06 [−0.31, 0.10]	1×70/85/none	60/60/59/59	12.56
	100	43.44	43.44	43.54	43.54	−0.09 [−1.22, 0.38]	1×85/none	60/60/60/60	50.22
	300	129.51	129.51	129.79	129.79	−0.28 [−3.62, 1.11]	1×85/none	60/60/60/60	150.67
GPT-5.2	25	10.03	10.03	9.93	9.93	0.10 [−0.28, 0.26]	1×90/95	67/67/66/66	11.96
	100	37.93	37.93	37.60	37.60	0.34 [−0.77, 0.93]	1×90/95	67/67/66/66	47.84
	300	112.34	112.34	111.37	111.37	0.96 [−2.61, 2.75]	1×90/95	67/67/66/66	143.53
Sonnet 4	25	11.10	11.10	11.13	11.13	−0.03 [−0.82, 0.04]	1×150/175/275	69/69/68/68	12.58
	100	40.56	40.56	40.65	40.65	−0.09 [−2.29, 0.06]	1×175/200/275	69/69/69/69	50.31
	300	119.12	119.12	119.23	119.23	−0.11 [−8.16, 0.16]	1×200/275	69/69/69/69	150.93
Sonnet 4.5	25	12.06	12.06	12.08	12.08	−0.03 [−0.79, 0.14]	1×150/175	64/64/63/63	12.56
	100	42.70	42.70	42.87	42.87	−0.17 [−0.81, 0.27]	1×150/175/225	64/64/63/63	50.22
	300	124.43	124.43	124.95	124.95	−0.52 [−2.44, 0.67]	1×150/175/225	64/64/63/63	150.67
Gemini 3 Pro	25	11.29	11.29	11.33	11.33	−0.04 [−0.64, 0.03]	1×200/275/400	61/61/61/61	12.29
	100	42.08	42.08	42.18	42.18	−0.09 [−1.73, 0.07]	1×200/400	61/61/61/61	49.17
	300	124.20	124.20	124.49	124.49	−0.29 [−4.44, 0.25]	1×200/400	61/61/61/61	147.51
Kimi K2	25	11.90	11.90	12.08	12.08	−0.17 [−0.99, 0.65]	1×80/150	58/58/53/53	12.62
	100	44.51	44.51	45.64	45.64	−1.13 [−3.80, 1.19]	1×80/150	58/58/53/53	50.48
	300	131.47	131.47	135.14	135.14	−3.67 [−11.40, 2.82]	1×80/150	58/58/53/53	151.43
Qwen3 Coder†	25	11.82	11.82	11.86	11.86	−0.04 [−0.63, 0.46]	1×80/90/175	66/66/58/58	12.56
	100	42.71	42.71	42.90	42.90	−0.20 [−2.52, 0.66]	1×150/175	66/66/66/66	50.22
	300	125.08	125.08	124.92	124.92	0.15 [−8.37, 1.15]	1×175	66/66/66/66	150.67

Table A11. Every crossing of the cap's marginal value. For each configuration and regime, each rate at which the margin changes sign over the sweep, in dollars an hour and in multiples of the median attempt cost, with the sign on either side and whether the bootstrap interval is clear of zero below it, above it, on both sides or on neither. Under review at 0.5 the margin does not change sign. † Imputed price.

regime	configuration	crossing	$/h	x median attempt	below, above	interval clear of zero
automated verifier	GPT-5	1 of 3	1.74	2.54	saves, costs	below
		2 of 3	5.08	7.39	costs, zero	neither
		3 of 3	6.48	9.44	zero, costs	neither
	GPT-5.2	1 of 4	3.36	2.89	saves, costs	below
		2 of 4	4.46	3.83	costs, saves	neither
		3 of 4	6.49	5.58	saves, costs	neither
		4 of 4	71.68	61.66	costs, saves	neither
	Sonnet 4	1 of 1	4.51	2.18	saves, costs	below
	Sonnet 4.5	1 of 3	7.08	2.14	saves, costs	below
		2 of 3	39.84	12.04	costs, saves	neither
		3 of 3	81.29	24.58	saves, costs	neither
	Gemini 3 Pro	1 of 1	3.90	2.35	saves, costs	below
	Kimi K2	1 of 1	5.30	3.83	saves, costs	below
	Qwen3 Coder†	1 of 1	5.91	2.73	saves, costs	below
review at 0.1 H	GPT-5	1 of 3	2.18	3.18	saves, costs	below
		2 of 3	88.02	128.21	costs, saves	neither
		3 of 3	154.58	225.17	saves, costs	neither
	GPT-5.2	1 of 3	6.73	5.79	saves, costs	below
		2 of 3	8.76	7.53	costs, saves	neither
		3 of 3	50.18	43.16	saves, costs	neither
	Sonnet 4	1 of 3	5.54	2.68	saves, costs	below
		2 of 3	364.36	176.41	costs, saves	neither
		3 of 3	473.87	229.43	saves, costs	neither
	Sonnet 4.5	1 of 2	8.87	2.68	saves, costs	below
		2 of 2	75.41	22.80	costs, saves	neither
	Gemini 3 Pro	1 of 1	4.88	2.95	saves, costs	below
	Kimi K2	1 of 2	8.16	5.90	saves, costs	below
		2 of 2	155.49	112.43	costs, saves	neither
	Qwen3 Coder†	1 of 3	7.40	3.42	saves, costs	below
		2 of 3	21.17	9.80	costs, saves	neither
		3 of 3	21.98	10.17	saves, costs	neither
review at 0.3 H	GPT-5	1 of 1	4.71	6.86	saves, costs	below
	Sonnet 4	1 of 1	10.29	4.98	saves, costs	below
	Sonnet 4.5	1 of 1	18.11	5.47	saves, costs	below
	Gemini 3 Pro	1 of 1	10.16	6.13	saves, costs	below
	Kimi K2	1 of 1	14.88	10.76	saves, costs	below
	Qwen3 Coder†	1 of 2	22.82	10.56	saves, costs	below
		2 of 2	190.34	88.08	costs, saves	neither

Data, Code and Preregistration

`loong0814/openhands_trajectories`, the Hugging Face release accompanying Bai et al. (2026), holds the trajectories. The release records no license, so the tables derived from it are not redistributed; the repository instead ships the code that acquires each archive, checks it against the checksum Hugging Face records and rebuilds the tables locally. The code, the results files, the exhibits and the sources of this manuscript are at github.com/james-e-murphy/restart-economics, where `results/README.md` names the command that writes each file every number here comes from. The code is under the MIT license, and the analysis plan and data audit under CC BY 4.0, the license they were registered under. The analysis plan, the data audit and the repository at the `plan-frozen` tag are registered at osf.io/pdmw9. The code, results and manuscript sources this version was built from are at the `v0.1` tag.

References

Bai, Longju, Zhemin Huang, Xingyao Wang, Jiao Sun, Rada Mihalcea, Erik Brynjolfsson, Alex Pentland, and Jiaxin Pei. 2026. “How Do AI Agents Spend Your Money? Analyzing and Predicting Token Consumption in Agentic Coding Tasks.” arXiv:2604.22750.

Brown, Bradley, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. “Large Language Monkeys: Scaling Inference Compute with Repeated Sampling.” arXiv:2407.21787.

Chen, Lingjiao, Matei Zaharia, and James Zou. 2024. “FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance.” *Transactions on Machine Learning Research*. arXiv:2305.05176.

Chen, Mark, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, et al. 2021. “Evaluating Large Language Models Trained on Code.” arXiv:2107.03374.

Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. 2018. “Double/Debiased Machine Learning for Treatment and Structural Parameters.” *The Econometrics Journal* 21(1): C1-C68.

Chowdhury, Neil, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, et al. 2024. “Introducing SWE-bench Verified.” OpenAI, August 13, 2024; updated February 24, 2025.

Davison, A. C., and D. V. Hinkley. 1997. *Bootstrap Methods and Their Application*. Cambridge: Cambridge University Press.

Gomes, Carla P., and Bart Selman. 2001. “Algorithm Portfolios.” *Artificial Intelligence* 126(1-2): 43-62.

Gomes, Carla P., Bart Selman, Nuno Crato, and Henry Kautz. 2000. “Heavy-Tailed Phenomena in Satisfiability and Constraint Satisfaction Problems.” *Journal of Automated Reasoning* 24(1-2): 67-100.

Guo, Yaoqi, Ying Xiao, Jie M. Zhang, Mark Harman, Yiling Lou, Yang Liu, and Zhenpeng Chen. 2026. “EET: Experience-Driven Early Termination for Cost-Efficient Software Engineering Agents.” *Findings of the Association for Computational Linguistics: ACL 2026*, 33008-33024. arXiv:2601.05777.

He, Xin, Yanlin Wang, Mingwei Liu, Jiachi Chen, Hongyu Zhang, and Guanbin Li. 2026. “SWE-Gate: Passing Functional Tests Is Not Enough for Software Engineering Agents.” arXiv:2609.04167.

Huberman, Bernardo A., Rajan M. Lukose, and Tad Hogg. 1997. “An Economics Approach to Hard Computational Problems.” *Science* 275(5296): 51-54.

Jimenez, Carlos E., John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. “SWE-bench: Can Language Models Resolve Real-World GitHub Issues?” *The Twelfth International Conference on Learning Representations* (ICLR 2024). arXiv:2310.06770.

Kapoor, Sayash, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. 2025. “AI Agents That Matter.” *Transactions on Machine Learning Research*. arXiv:2407.01502.

Kitagawa, Toru, and Aleksey Tetenov. 2018. “Who Should Be Treated? Empirical Welfare Maximization Methods for Treatment Choice.” *Econometrica* 86(2): 591-616.

Kwa, Thomas, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, et al. 2025. “Measuring AI Ability to Complete Long Software Tasks.” *Advances in Neural Information Processing Systems* 38: 92213-92266. arXiv:2503.14499v4.

Lin, Yuxiang, Zihan Wang, Mengyang Liu, Yuxuan Shan, Longju Bai, Junyao Zhang, Xing Jin, Boshan Chen, Jinyan Su, Xingyao Wang, Jiaxin Pei, and Manling Li. 2026. “BAGEN: Are LLM Agents Budget-Aware?” arXiv:2606.00198.

Liu, Tengxiao, Zifeng Wang, Jin Miao, et al. 2025. “Budget-Aware Tool-Use Enables Effective Agent Scaling.” arXiv:2511.17006.

Luby, Michael, Alistair Sinclair, and David Zuckerman. 1993. “Optimal Speedup of Las Vegas Algorithms.” *Information Processing Letters* 47(4): 173-180.

OpenAI. 2026. “Why SWE-bench Verified No Longer Measures Frontier Coding Capabilities.” OpenAI, February 23, 2026.

Rice, John R. 1976. “The Algorithm Selection Problem.” *Advances in Computers* 15: 65-118.

Wang, Chenyu, Yunbo Lyu, Junda He, Zhou Yang, Chenxing Zhong, Yaniv Harel, and David Lo. 2026. “Fail-Fast, Restart-Smart: Early Failure Prediction and Restart for SWE Agentic Tasks.” arXiv:2608.03222.

Wang, Xingyao, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. “Executable Code Actions Elicit Better LLM Agents.” *Proceedings of the 41st International Conference on Machine Learning*, PMLR 235: 50208-50232.

Wang, Xingyao, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, et al. 2025. “OpenHands: An Open Platform for AI Software Developers as Generalist Agents.” *The Thirteenth International Conference on Learning Representations* (ICLR 2025). arXiv:2407.16741.

Whitfill, Parker, Cheryl Wu, Joel Becker, and Nate Rush. 2026. “Many SWE-bench-Passing PRs Would Not Be Merged into Main.” METR Note, March 10, 2026.

Declarations

Funding. This work received no external funding.

Competing interests. The author declares no competing interests.

Acknowledgements. The paper rests on the trajectories that Bai et al. (2026) released; the author thanks them for making the logs public.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author used large language model assistants, including Anthropic's Claude and OpenAI's ChatGPT, for writing and testing the analysis code, drafting support, source-verification support, critical review, and editorial iteration under the author's direction. After using these tools, the author reviewed and edited all content, independently verified the sources and the claims they support, and takes full responsibility for the content of this publication.

Comments are invited; this is an explicitly provisional working draft.

Working paper for discussion. Not investment, legal, or tax advice.