CHOOSING HOW CODING AGENTS CONTINUE: A STUDY ON BIGCODEBENCH

Anonymous authors
Paper under double-blind review

ABSTRACT

After producing a program, a coding agent can stop, revise it, request a review, or start again. We ask whether information from the first attempt helps choose among these actions. On an eligible subset of BigCodeBench, we train a routing policy on exploratory executions and freeze it before evaluation on unseen tasks. Compared with a fixed strategy, regeneration by the stronger model, the observed gain was 2.5 percentage points (95% CI: -1.0 to 5.5). It was not statistically conclusive and fell below the preregistered practical target. A post-hoc analysis locates the net gain in stopping and stronger-model review. That review action performed poorly across the full evaluation set, but produced no execution errors where the policy selected it. All its execution errors coincided with output-limit hits; review and regeneration used different effective output allowances. These descriptive associations do not establish why the selections worked or whether they will recur. We report state-acquisition, continuation and learning costs separately. The task-level comparison exposes differences hidden by aggregate action rankings while leaving the generalisable benefit of adaptive continuation unresolved.

1 INTRODUCTION

A first attempt gives a coding agent both a program and information about that program. After running the examples in the prompt, the agent can return its solution, revise it, request a review, or generate a replacement. These actions use the first attempt differently. Revision and review retain its code and feedback; regeneration discards them. Choosing between these actions therefore involves more than selecting a model: the same model can be asked to repair an existing solution or solve the task again.

Work on model cascades, self-refinement and reflection offers several ways to continue after a first answer (Chen et al., 2023; Madaan et al., 2023; Shinn et al., 2023). Here we ask a narrower empirical question: *can the state of the first attempt help choose the next action on a new task?* A visible check evaluates the current program, whereas a continuation policy must predict the outcome of an action not yet taken. A failed check does not specify whether revision, review or regeneration will help. An action's average success rate also leaves open how it performs on the particular tasks where a policy selects it.

Evaluating that choice requires care. Running every action and then picking a successful one uses information that a deployed policy would not have. Learning a selector on those outcomes and assessing it on the same tasks has a similar problem. We therefore separate learning from evaluation. On an eligible subset of BigCodeBench (Zhuo et al., 2025), we execute all actions on exploration tasks, fit a small routing policy, and freeze both the policy and its fixed-action comparator before running a disjoint confirmation set. The policy observes the prompt, the first program, its generation cost and the visible-check result; it never observes hidden-test outcomes at decision time.

On held-out tasks, the policy's observed pass rate exceeded the fixed strategy's by 2.5 percentage points. The evidence did not establish a positive gain or meet the preregistered practical target. A post-hoc decomposition locates the net difference in stopping and expert-review decisions. Stronger-model review had many execution errors overall but none on the tasks where the policy selected it. Those errors coincided with output-limit hits, and review received a smaller effective output allowance than regeneration. This makes the observed action rankings specific to the executed budget protocol.

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

Our contribution is a task-level study of adaptive continuation under that protocol. The primary comparison estimates the benefit of the frozen policy; the descriptive analysis identifies which choices changed outcomes and documents the associated execution failures. We also separate the costs of acquiring the first attempt, taking the next action and collecting policy-training outcomes. The released records support re-analysis of these comparisons without new API calls.

2 PROBLEM DEFINITION

Each task has a first-pass program s_0 produced by model A and a visible-check result. A *continuation action* maps that state to a final program. The action set contains stopping, three forms of same-model revision, review by model B, and regeneration by B (Table 1). Both models remain fixed throughout the study.

Table 1: Continuation actions. Revision and review receive the task, the first-pass program and its visible-check feedback. B-solo receives the task but discards the program and feedback. The two B actions use the same model.

Action	Model	Output rule
T	none	Return the first-pass program unchanged.
A-self	A	Make one revision call, even if the visible checks already pass.
A-self$\times K$	A	Revise until a visible pass, the revision cap, or a budget refusal. Make no call if the first program already passes.
A-role	A	Make one call with a reviewer role.
B-expert	B	Review and correct the first-pass program in one call.
B-solo	B	Generate a replacement program from the task.

For A-self$\times K$, the cap is $K_{\text{revise}} = \min(8, \max(1, \lfloor b_{\text{cont}}/\bar{c}_A \rfloor))$, where $\bar{c}_A$ is the median first-pass generation cost in round 0. It equals 4 here. This cap is distinct from the number of independent realisations r of an action; the main comparison uses $r = 1$.

Let $y_{i,a} \in \{0, 1\}$ denote the hidden-test result of action a on task i. A policy $\pi(x_i)$ chooses an action using features available before any continuation call. Exploration outcomes supply labels for fitting the router; confirmation outcomes are used only for evaluation. The target is the policy's expected gain over the fixed action $\hat{a}$ selected during exploration:

$$\Delta(\pi, \hat{a}) = \mathbb{E}_i\big[y_{i,\pi(x_i)} - y_{i,\hat{a}}\big]. \tag{1}$$

The comparator is frozen along with the policy. It is not assumed to be the population's best fixed action. Nor is the task-wise maximum over the observed actions a deployable policy: it has already seen their results. Appendix F distinguishes this descriptive maximum from a probability oracle and from the policy gain tested here.

3 METHOD: AN OFFLINE ROUTING POLICY

Decision-time features. Prompt-only features include prompt length, docstring examples, argument count, a Returns-clause indicator and library requirements. Post-generation features add the visible-check outcome, failure count, error category, program size, AST statistics, generation tokens and dollar cost. We use 28 raw features: 26 numeric and two categorical. Library requirements use fixed indicators for the 12 most frequent libraries in the eligible pool plus an other indicator; error categories use eight predefined exception classes plus other/none. The post-generation representation thus contains both properties of the program and the result of running it. The comparison with prompt-only features tests this combination, not the effect of executing checks alone.

Action scoring. P2 fits one L_2-regularised logistic model $\hat{p}_a(x)$ per action and selects $\arg\max_a \hat{p}_a(x)$. P2-post uses prompt and post-generation features. P2-pre uses prompt-only features and selects its own feature subset and regularisation. A visible-outcome lookup table, P1, provides a simpler diagnostic.

Feature selection limits the encoded design matrix to 20 columns, including categorical columns. We select the feature subset and $\lambda \in \{0.3, 1, 3\}$ by repeated five-fold grouped cross-validation

on exploration tasks. Feature selection, scaling and the encoding of observed categorical levels are fitted within each training fold. The regularisation value with the highest mean policy gain is selected, after which the policy is refitted on all exploration tasks. Appendix E gives the ranking, tie-breaking and refitting rules. The repeated CV score is for model selection; the confirmation set is the independent test of the fitted policy.

Policy fixed before evaluation. Before confirmation, we record the feature subsets, regularisation values, coefficients, P1 table, fixed action $\hat{a}$, continuation budget and seed. Hidden-test results never enter the decision-time features or the prompts given to the models. The same fitting procedure is used in the real experiment and the synthetic study.

4 EXPERIMENTAL DESIGN

Tasks and split. We use a fixed revision of BigCodeBench Complete, retaining 403 tasks with an executable visible example and stable canonical hidden-test results. Static exclusions, repeated canonical evaluation and development-task exclusions are recorded in Appendix A. The exploration set contains 100 tasks: 59 re-eligible tasks from an earlier probe and 41 drawn with a fixed seed. We reuse the probe's original first-pass programs without selecting or regenerating them by outcome. The confirmation set contains 200 previously unused tasks; 103 further tasks remain sealed. All continuation actions are evaluated on the same first-pass state for a task.

Models and execution budget. Model A is claude-haiku-4-5 and B is claude-sonnet-4-6. Both use temperature 0.2 and an initial output ceiling of 4096 tokens. Each action is realised once per task; A-self and B-expert are additionally repeated on 30 exploration tasks to describe run-to-run flips. In round 0, B-expert runs on the first 10 exploration tasks without the per-action dollar cap, retaining the 4096-token ceiling. The median call cost defines b_{cont}. The main pass then reruns those actions under the measured budget.

On each subsequent call, input-token cost is deducted before the remaining budget determines the maximum output length. All continuation actions receive b_{cont} *except* B-solo: its implemented call budget also includes the task's actual first-pass generation cost. Thus this is not an equal-additional-budget comparison. B-solo ignores the first-pass program, but its budget depends on the cost of producing that program. This asymmetry is retained in the analysis and matters for interpreting the action rankings.

Registration and pool gate. Before confirmation, we froze the policy, comparator, task IDs and analysis rule. The earlier probe had failed an original ceiling of 70% for B-solo. That operating criterion was changed after the probe, but before the exploration set was assembled, to require a best exploration-action success rate of at most 80%. The original rule and failed probe remain in the record (Appendix A). The revised gate is an operational choice, not a guarantee of population headroom.

Primary comparison. For each confirmation task, $d_i = y_{i,\hat{\pi}_{\mathrm{post}}(i)} - y_{i,\hat{a}}$. The preregistered rule requires a one-sided exact McNemar $p < 0.05$ and an observed mean gain of at least 10 percentage points. Otherwise the recorded verdict is *unconfirmed*. The interval uses 1,000 paired task-bootstrap samples with the policy held fixed. It captures variation across sampled tasks, not uncertainty from refitting the policy or repeating model calls. The rule combines evidence of a positive difference with a practical threshold on its point estimate; it is not a test that the true gain exceeds that threshold.

Secondary analyses and cost. The registered secondary comparisons include P2-post versus P2-pre, P1, per-action pass and refusal rates, policy choices, and API expenditure. We distinguish the cost of obtaining the first-pass state from continuation calls and the executions used for policy learning. Section 5 reports both continuation-only costs and costs including the first pass for both strategies. Appendix B preserves the initial, asymmetric accounting and explains the post-confirmation correction. These are cost descriptions, not a cost-effectiveness test.

Synthetic checks. We apply the full exploration-and-confirmation procedure to a generator with both informative and uninformative features. It uses the same fitting code, feature selection, pool

gate and decision rule. Appendix C reports the observed passing and gate-hold rates separately. The simulations expose substantial learning loss and frequent holds under some conditions, so a non-passing real run cannot by itself distinguish a small available gain from a policy that failed to learn it.

5 RESULTS

5.1 FROZEN CHOICES FROM EXPLORATION

B-solo was the best fixed action on exploration, with a 79.0% pass rate, and satisfied the revised pool gate. Round 0 set $b_{\text{cont}} = \$0.012834$. Model selection chose $\lambda = 0.3$ and 18 raw features (20 encoded columns) for P2-post; P2-pre used $\lambda = 3$ and 17 features. The comparator $\hat{a}$ = B-solo and both policies were then frozen. The selected P2-post model had a repeated-CV gain of -1.6 percentage points relative to the fixed action selected within each training fold (Table 12). This was a model-selection score, not an independent confirmation result. Appendix E reports all exploration-action rates, selection scores, repeated-call flip rates and expenditure.

5.2 THE HELD-OUT GAIN REMAINS UNCERTAIN

On the 200 confirmation tasks, P2-post passed 148 (74.0%) and the frozen B-solo comparator passed 143 (71.5%). The paired difference was 2.5 percentage points (95% CI [-1.0, 5.5]); the policy won on 8 discordant tasks and lost on 3 (one-sided exact McNemar $p = 0.1133$). The registered verdict is **unconfirmed**. The estimate is positive, but the interval includes no gain and the result does not establish the preregistered practical improvement. This evaluates the policy fitted in exploration, not every possible policy over these actions.

Table 2: Confirmation set ($n = 200$ in every row): paired differences d_i; b = tasks solved only by the first policy, c = tasks solved only by its comparator; exact one-sided McNemar p. Only the first row is the pre-registered primary comparison and carries the verdict ($p < 0.05$ and mean $d \geq 10$ pp $\Rightarrow$ pass, else unconfirmed). The remaining rows are secondary analyses reported with the same statistics for description; they are not tested against the rule and do not substitute for the primary verdict.

Comparison	Role	Mean d (pp)	b	c	p	95% CI	Verdict
P2-post vs $\hat{a}$	primary	2.5	8	3	0.1133	[-1.0, 5.5]	**unconfirmed**
P2-pre vs $\hat{a}$	secondary	0.0	7	7	0.6047	–	n/a
P1 vs $\hat{a}$	secondary	0.0	0	0	1.0000	–	n/a
P2-post vs P2-pre	secondary	2.5	10	5	0.1509	–	n/a

5.3 POST-GENERATION FEATURES DID NOT ESTABLISH AN ADVANTAGE OVER PROMPT-ONLY FEATURES

P2-post exceeded P2-pre by 2.5 percentage points on the same tasks ($b = 10$, $c = 5$, $p = 0.1509$). That secondary result is also inconclusive. Table 2 includes P2-pre and P1 against the fixed comparator; only the P2-post–fixed comparison carries the preregistered verdict. Per-action results appear in Table 3.

5.4 CONTINUATION COST AND STATE-ACQUISITION COST

At the decision point the first-pass state already exists. Counting only the next action gives mean costs of \$0.0048 for the policy and \$0.0055 for B-solo. Charging both for acquiring that same state gives \$0.0077 and \$0.0085. B-solo's implemented budget depended on first-pass cost, so a deployment that skips the first pass under otherwise identical settings was not tested. Table 4 describes the costs of the executed choices; it is not an equal-budget comparison or a cost-effectiveness test. Policy-learning costs are not amortised over future use. Appendix B gives the experimental expenditure and the accounting revision made after confirmation.

Table 3: Confirmation set ($N_c = 200$): per-arm hidden-test pass rate, refusal rate and own cost.

Arm	n	Hidden pass (%)	Refused (%)	Own cost / task (USD)	Calls / task
T	200	66.5	0.0	0.0000	0.00
A-self	200	68.5	0.0	0.0036	1.00
A-self$\times K$	200	68.0	11.5	0.0022	0.58
A-role	200	69.5	0.0	0.0041	1.00
B-expert	200	48.5	0.5	0.0103	0.99
B-solo	200	71.5	0.0	0.0055	1.00

Table 4: Mean API cost per confirmation task (USD). Both columns charge the same cost components, but the actions did not receive equal additional budgets: B-solo's call allowance includes the first-pass generation cost. Policy-learning and verification-compute costs are not included. Full accounting appears in Appendix B.

Cost components	Policy	Fixed B-solo
Continuation only; first-pass state already available	0.0048	0.0055
Continuation plus the same first-pass acquisition	0.0077	0.0085

5.5 Stopping and review account for the observed net gain

The following analysis is post-hoc and descriptive. It uses the stored confirmation records without refitting the policy or adding a hypothesis test. Table 5 groups tasks by the frozen policy's chosen action and compares that action with B-solo on the same tasks. Stopping contributed +2 net successes and expert review +3; the other selected actions made no net contribution.

The review action illustrates the difference between an overall action rate and a policy's selected cases. It had 49 execution errors when run across all 200 tasks. On the 39 tasks where the policy selected it, there were none, and review passed 31 tasks against B-solo's 28. Those selected tasks are not a random sample, and this contrast is not evidence that a particular feature caused good choices. It locates the observed difference; it does not resolve the uncertainty in the primary comparison.

Table 5: Post-confirmation, post-hoc descriptive analysis: the frozen policy's choices on the 200 confirmation tasks, grouped by chosen arm. For each group, successes of the chosen arm and of the fixed arm B-solo *on the same tasks*, discordant wins/losses, and the chosen arm's execution errors within the group. The totals reproduce the primary counts (148 vs. 143, $b = 8$, $c = 3$). This table explains where the net difference arose; it is not a new test, and differences between groups are not evidence of selection superiority or of a causal mechanism.

Chosen arm	n	Chosen pass	Fixed pass	Win	Loss	Net	Errors
T	27	22	20	3	1	+2	0
A-self	6	4	4	0	0	+0	0
A-self$\times K$	16	10	10	1	1	+0	0
A-role	21	17	17	1	1	+0	0
B-expert	39	31	28	3	0	+3	0
B-solo	91	64	64	0	0	+0	0
Total	200	148	143	8	3	+5	–

The execution records also show why the review action needs to be read together with its call settings. B-expert made 199 calls and had 1 pre-call budget refusal. Its median output ceiling was 644 tokens, compared with 965 for B-solo. It reached the ceiling in 71 calls, including every one of its 49 execution errors. B-solo had no such errors or limit hits. No larger-budget review condition was run, so these records do not identify how much of the difference a larger output budget would remove.

Visible checks were also imperfect proxies for the final outcome: 40 of 157 visible passes failed hidden tests, while 16 of 43 visible failures passed them. Appendix D contains the cross-table and error breakdown. We did not investigate individual causes of those disagreements.

6 Discussion and conclusion

The frozen policy did not establish an advantage over regeneration by the stronger model. Its small observed gain nevertheless has an identifiable composition: stopping retained some successful first attempts, and selected review calls solved tasks that regeneration missed. The comparison concerns these choices under one protocol, rather than a general ranking of agent collaboration methods.

Review under the implemented budget. Review and regeneration used the same stronger model, but received different inputs and effective output allowances. The error records therefore cannot distinguish difficulties with reviewing another program from difficulties with completing that review within the allowed output length. Varying the review allowance while holding its inputs fixed would help distinguish these explanations. Until then, the review action's low overall success rate should not be interpreted as a general failure of the model to help another agent.

Performance on policy-selected tasks. The review action's aggregate error rate also misses a feature of this particular policy: none of its selected review calls produced an execution error. That observation describes what happened on the selected tasks, not why the policy selected them. The groups were not randomised, and the primary comparison remains inconclusive. Testing the same frozen selector on new tasks would assess whether this pattern recurs without selecting a new policy to fit it.

The information supplied by visible checks. Visible checks disagreed with hidden tests in both directions, so their verdict alone was not a reliable substitute for the benchmark outcome. The routing problem asks more of that signal: it must help predict a future action's success. Our representation mixes executed-check feedback with program properties available before the check runs. A prompt-plus-program policy that omits execution feedback is needed to isolate the check's contribution. That comparison should include the cost of executing the check, not only the cost of the selected continuation.

The paired records support evaluating a continuation policy separately from its available actions' overall rankings. They do not yet establish that learning such a policy improves outcomes on this task pool. The next comparisons should test the frozen choices on independent tasks and vary the review budget while holding its inputs fixed.

7 Related work

Model cascades already use information from a generated answer to decide whether to continue. FrugalGPT (Chen et al., 2023) learns a quality scorer and queries the next API when the current answer does not clear a threshold. Our router is also a learned scorer. The difference studied here is the choice among keeping a program, revising it, reviewing it and regenerating it, using both program features and visible-check outcomes.

Self-Refine (Madaan et al., 2023) alternates feedback and revision from the same model. Reflexion (Shinn et al., 2023) adds evaluation and stored verbal reflections; its programming setup uses generated unit tests. Our revision actions use the provided visible checks and do not include those methods' feedback-generation or memory mechanisms. They are not replications of either method, and the experiment does not compare against them.

REDEREF (Hosseini et al., 2026) updates beliefs from judge feedback and uses reflection for recursive rerouting and termination. KABB (Zhang et al., 2025) combines knowledge-distance priors with outcome-based bandit updates to select experts. BiRouter (Yang et al., 2026) learns next-hop decisions, including termination. Pandora-style routing (Fisch et al., 2026) treats inspecting an expert as a costly decision. These works address related selection problems; our prompt-only policy is an internal ablation, not a proxy for them. Direct comparisons would require matching models, feedback and budgets.

At the system level, Kim et al. (2026) study coordination across 260 configurations and report single-agent baseline strength as a strong predictor of whether collaboration helps. Here the comparison is within tasks: given a particular first program, can a frozen policy improve on a fixed continuation?

8 LIMITATIONS

This is one eligible subset of one benchmark, one model pair and one main realisation per task–action pair. The router was fitted on 100 tasks with a restricted model and feature menu. The synthetic study shows that this fitting procedure can lose much of the gain available to an oracle; it does not measure how much gain was learnable on the real tasks. Neither failure to meet the practical threshold nor a nonsignificant test establishes equivalence to the fixed strategy.

Task selection is another restriction. The eligible pool requires executable visible examples and stable canonical results, and the 80% exploration gate was adopted after an earlier probe. The untouched confirmation set protects that comparison from fitting on its outcomes, but does not undo benchmark or protocol selection. BigCodeBench's hidden tests are publicly available, so model contamination cannot be excluded.

The post-generation features combine program properties, generation costs and executed-check feedback. The prompt-only ablation cannot isolate these sources. Similarly, review and regeneration differ in both their inputs and effective budgets. The observed association between errors and output limits is not a causal estimate of the effect of increasing the budget.

The expenditure figures cover API calls rather than full system cost. Visible verification took a mean of 3.716605 seconds per confirmation task, measured as wall-clock time in the evaluation container. This is neither CPU time nor end-to-end latency. Compute was not priced, and learning costs were not amortised over a deployment horizon. These omissions prevent a net-benefit or cost-effectiveness conclusion.

REPRODUCIBILITY STATEMENT

The supplement contains the registered protocol, frozen policy, task IDs, stage reports, task-level analysis records, budget ledger, implementation and synthetic-study code. It supports independent re-analysis of the confirmation result and regeneration of the tables. It does not contain candidate programs or hidden-test contents and does not support rerunning the API experiment. `REPRODUCE.md` lists the dependencies and supported commands. A manifest distinguishes historical source hashes from those of the anonymised distribution copies. The frozen policy and task-level analysis records retain their original bytes; the distribution uses neutral package and environment-variable names without changing numerical settings or results. All numerical tables and result macros are generated from stored records.

AI USE STATEMENT

The author used Claude (Anthropic) and Codex (OpenAI) to develop the research question and experimental design, write and review code and tests, implement the synthetic-study generator, check statistical statements, search and verify references, analyse saved results, and draft and revise the manuscript. Reviews by another assistant were part of this process, not independent replications. The author directed the work, approved expenditure and takes responsibility for the submitted claims. Language models are also the objects of study.

REFERENCES

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Adam Fisch, Shubhendu Trivedi, Fantine Huot, William W. Cohen, Michael Kaisers, Mirella Lapata, Kate Larson, and Jacob Eisenstein. Pandora's AI model routing box: Efficient allocation with costly value estimation. *arXiv preprint arXiv:2608.20316*, 2026.

Mohammad Parsa Hosseini, Ankit Shah, Saiyra Qureshi, Alex Huang, Connie Miao, and Wei Wei. Training-free agentic ai: Probabilistic control and coordination in multi-agent llm systems. *arXiv preprint arXiv:2603.13256*, 2026.

Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall, A. Ali Heydari, Yao Yan, Zhihan Zhang, Yuchen Zhuang, Yun Liu, Mark Malhotra, Paul Pu Liang, Hae Won Park, Yuzhe Yang, Xuhai Xu, Yilun Du, Shwetak Patel, Tim Althoff, Daniel McDuff, and Xin Liu. Capable language models can outgrow the benefits of collaboration. *Nature Machine Intelligence*, 8:1157–1172, 2026. doi: 10.1038/s42256-026-01268-y.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

Qingwen Yang, Feiyu Qu, Tiezheng Guo, Yanyi Liu, and Yingyou Wen. Augmented runtime collaboration for self-organizing multi-agent systems: A hybrid bi-criteria routing approach. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, 2026. arXiv:2512.00740.

Jusheng Zhang, Zimeng Huang, Yijia Fan, Ningyuan Liu, Mingyan Li, Zhuojie Yang, Jiawei Yao, Jian Wang, and Keze Wang. KABB: Knowledge-aware bayesian bandits for dynamic expert coordination in multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. arXiv:2502.07350.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro von Werra. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.15877.

A RECORDED PROTOCOL AND TASK ELIGIBILITY

Table 6: Protocol recorded before confirmation, reproduced from the saved configuration and task-set hashes (identifiers truncated to 12 hex characters). Hashes identify the saved files; they do not denote external registration.

Item	Registered value
Design version / date	v4 / 2026-09-15
Pilot config sha256	`ffdc7bd1de7b`
Eligible pool	BigCodeBench Complete, 403 tasks
Exploration / confirmation / reserve	100 (59 probe-reused s_0 + 41 new) / 200 / 103 (sealed); seed 20260915
ID-set sha256 (explore / confirm / reserve)	`75f232befaa1` / `7cf767296fb7` / `68a6b3d76a2e`
Arms, repetitions	T, A-self, A-self$\times K$, A-role, B-expert, B-solo; $r = 1$; flip subset 30 tasks $\times$ A-self, B-expert at $r = 2$ (config fields `k` / `flip_k` denote realisations, not the inner revision cap)
Models	A = `claude-haiku-4-5-20251001`, B = `claude-sonnet-4-6`, $T = 0.2$, max tokens 4096
Round 0	b_{cont} = median cost of one B-expert call on 10 tasks, run without the per-arm dollar budget (output limit max_tokens=4096 kept)
Pool gate	best exploration arm $\leq 80\%$ (original registered ceiling 70%, revised after the probe; both kept)
Feature menu / selection	28 raw features, $\leq$ 20 design columns after encoding; $\lambda \in \{0.3, 1, 3\}$; nested GroupKFold $K = 5$, $R = 20$
Frozen items	feature_subset, lambda, P2-post coef, P2-pre coef, P1 table, a_hat (best exploration arm), b_cont, seed
Primary rule	exact one-sided McNemar $p < 0.05$ and mean $d \geq 10$ pp $\Rightarrow$ pass, else unconfirmed; CI = task bootstrap of d_i
Budget	ledger cap \$16.00 (estimates: exploration \$5.00, confirmation \$9.00)

Pre-registration here refers to the version-controlled protocol and analysis rule recorded before confirmation, not registration with an external service. The supplementary snapshot retains the historical execution-approval record and file hashes. These document the executed configuration; they are not independent validation of the study.

Eligibility record. BigCodeBench Complete contains 1140 tasks at the fixed revision used here. Of these, 1021 have a doctest with an expected output; 630 pass the static screen for randomness, seed dependence and related exclusions. This screen already removes 2 of the 6 development tasks. There are then 411 tasks with at least one executable visible check, 408 whose canonical solutions pass the official hidden tests in 3 independent runs, and 407 consistent with the stored evaluation history. Excluding the remaining 4 development tasks leaves 403 tasks ($407 - 4 = 403$). The full screen is recorded in `bcb_eligibility.json`.

Probe and gate revision. The earlier B-solo probe passed 45 of 60 tasks (75.0%, 95% CI [62.8, 84.2]), exceeding the original 70% ceiling. That failed gate was retained as *unconfirmed*. Before assembling the exploration set, the rule changed to a maximum of 80% for the best of the six exploration actions. The rationale, observed headroom at least twice the practical target, was an operating choice made after the probe, not a statistical guarantee. The 59 re-eligible probe tasks supplied their original first-pass programs; the 41 new exploration tasks and 200 confirmation tasks came from the 344 unused tasks, leaving 103 reserves. The confirmation tasks were not used to select or revise the policy.

Failures and cost attribution. Every API attempt reserves a cost before execution. An HTTP error response is recorded as unbilled; when no response is observed, the reserved amount is retained as a conservative attribution and kept separate from measured spend. Incomplete task–action episodes remain in the ledger and on disk rather than being silently regenerated. The feature-definition version (`features-i1`) and execution settings are included in cache keys.

B COST ACCOUNTING AND EXPERIMENTAL EXPENDITURE

Table 7 preserves all three accounting views. The initial stage report charged first-pass generation to the policy but omitted it for the fixed strategy (view C). That calculation remains in the record, but is not an equivalent deployment comparison: the implemented B-solo budget depended on the first-pass generation cost. Views A and B distinguish the continuation decision from acquiring its state and charge the same components to both strategies. This accounting clarification was made after confirmation; no call costs, action outcomes or hypothesis tests were changed.

Table 7: Cost (USD). Top: per-task cost of the frozen policy and of the fixed arm B-solo under three viewpoints. The executed budget rule gives B-solo a call budget of b_{cont} plus the task's actual s_0 generation cost, so (A) and (B) are not a comparison at an equal additional budget; (C) preserves the initial asymmetric accounting as a retrospective calculation that omits the fixed arm's s_0 cost. A deployment of B-solo that decides without s_0 under the same call settings was not run. Bottom: total spend from the ledger of this study, and two items deliberately kept outside it: the historical generation cost of the 59 first-pass candidates reused from an earlier probe, and visible-check wall-clock time. CPU time was not measured; compute cost was not priced.

Item	Policy	Fixed
Per-task means (three viewpoints; no cost test)		
(A) additional call at decision time, s_0 already present	0.0048	0.0055
(B) same s_0 acquisition charged to both	0.0077	0.0085
(C) retrospective: policy includes s_0, fixed omits it	0.0077	0.0055
s_0 acquisition (mean per task)		0.0030
Experiment ledger: all executed arms and tasks		
Exploration stage		3.3160
Confirmation stage (incremental)		5.7366
Total / cap		9.0526 / 16.00
Confirmation arms: measured / unknown-attributed		5.1395 / 0.0000
Incomplete episodes (USD, count)		0.0000 (0)
Outside this ledger (not summed above)		
Historical generation cost of the 59 reused probe s_0		0.1811
Visible-check wall-clock time		reported in seconds; compute unpriced

Total experimental API expenditure was $9.0526 across 1671 recorded attempts, including all executed actions, calibration and repeated calls. No cost was attributed to attempts with unobserved billing, and no incomplete episodes remained. These totals describe this experiment, not the cost of deploying only the fitted policy. The historical cost of the reused first-pass programs is reported separately. Verification compute is unpriced, and policy-learning costs are not amortised over a deployment horizon.

C SYNTHETIC OPERATING CHARACTERISTICS

Table 8: Synthetic operating characteristics of the full exploration procedure (feature-subset selection $\leq$20, $\lambda \in \{0.3, 1, 3\}$ by nested CV, pool gate at 80%), $N_e = 100$, $N_c = 200$, 20 seeds per alternative cell and 60 per null. (a) passes with the gate, (b) passes ignoring the gate, (c) passes among gate-passed replicates; brackets are exact 95% binomial intervals in %. Null (a): features independent of outcomes; null (b): difficulty-only signal with an invariant best arm. Truth = mean realised gain of the learned policy on the large held-out set. These are observed rates under this generator, not power statements.

Condition	#feat	(a) pass	(b) pass no gate	(c) pass given gate	Gate hold	Truth (pp)
gain 10 pp	6	0/20 [0,17]	3/20 [3,38]	0/1 [0,98]	19	8.0
gain 10 pp	12	0/20 [0,17]	3/20 [3,38]	0/1 [0,98]	19	5.3
gain 10 pp	20	0/20 [0,17]	1/20 [0,25]	0/1 [0,98]	19	3.3
gain 10 pp	28	0/20 [0,17]	1/20 [0,25]	0/1 [0,98]	19	2.6
gain 15 pp	6	6/20 [12,54]	16/20 [56,94]	6/7 [42,100]	13	12.8
gain 15 pp	12	3/20 [3,38]	9/20 [23,68]	3/7 [10,82]	13	9.7
gain 15 pp	20	2/20 [1,32]	5/20 [9,49]	2/7 [4,71]	13	7.8
gain 15 pp	28	1/20 [0,25]	3/20 [3,38]	1/7 [0,58]	13	6.4
gain 5 pp	6	1/20 [0,25]	1/20 [0,25]	1/5 [1,72]	15	1.9
gain 5 pp	12	1/20 [0,25]	1/20 [0,25]	1/5 [1,72]	15	0.3
gain 5 pp	20	0/20 [0,17]	0/20 [0,17]	0/5 [0,52]	15	-0.8
gain 5 pp	28	0/20 [0,17]	0/20 [0,17]	0/5 [0,52]	15	-1.0
null (a)	6	0/60 [0,6]	0/60 [0,6]	0/29 [0,12]	31	-1.7
null (a)	12	0/60 [0,6]	0/60 [0,6]	0/29 [0,12]	31	-2.4
null (a)	20	0/60 [0,6]	0/60 [0,6]	0/29 [0,12]	31	-2.7
null (a)	28	0/60 [0,6]	0/60 [0,6]	0/29 [0,12]	31	-2.8
null (b)	6	0/60 [0,6]	0/60 [0,6]	0/56 [0,6]	4	-0.5
null (b)	12	0/60 [0,6]	0/60 [0,6]	0/56 [0,6]	4	-1.0
null (b)	20	0/60 [0,6]	0/60 [0,6]	0/56 [0,6]	4	-1.2
null (b)	28	0/60 [0,6]	0/60 [0,6]	0/56 [0,6]	4	-1.4

Realised truth bins: pooled cells, non-independent replicates

Truth bin	n	(a)	(b)	(c)
$<$ 0 pp	513	0/513 (0%)	0/513 (0%)	0/336 (0%)
[0, 5) pp	90	2/90 (2%)	2/90 (2%)	2/24 (8%)
[5, 10) pp	82	2/82 (2%)	14/82 (17%)	2/19 (11%)
[10, 15) pp	33	10/33 (30%)	25/33 (76%)	10/13 (77%)
$\geq$ 15 pp	2	0/2 (0%)	2/2 (100%)	–

The three quantities (a)/(b)/(c) are reported separately and never summed; the truth-bin rows pool replicates across cells and are descriptive only (no interval).

The generator uses 100 exploration and 200 confirmation tasks, with 20 seeds per alternative cell and 60 per null cell. Both null conditions—features independent of outcomes, and difficulty-only features with an invariant best action—passed in 0/60 replicates in each cell. At a calibrated Bayes-optimal gain of 15 percentage points, the rule passed without the gate in 16/20 replicates with six features and 3/20 with 28 features. With the gate included those counts were 6/20 and 1/20. At the 10-point condition no replicate passed with the gate and at most 3/20 without it. The fitted policy's mean realised gain in the 15-point condition fell from 12.8 to 6.4 points as the menu grew from six to 28 features. These are observed frequencies under the stated generator, not universal power guarantees.

D POST-HOC ACTION DIAGNOSTICS

All tables in this appendix were generated after the confirmation stage from the stored per-task records, the frozen policy and the per-cell traces by a read-only script; no model was called, no candidate was re-executed or re-graded, no parameter was refit, and no test or p-value was added. They describe one realisation of the pre-registered experiment and do not modify its verdict.

Table 9: Post-hoc descriptive analysis: visible-check verdict of the first-pass candidate s_0 against its official hidden-test result (arm T) on the 200 confirmation tasks. A hidden pass is the observed result of the official evaluation tests, not a guarantee of semantic correctness; the causes of the disagreements were not investigated (no code or test was re-run).

Visible verdict of s_0	n	Hidden pass	Hidden fail
pass	157	117	40
fail	43	16	27

Table 10: Post-hoc descriptive analysis of the two B arms on the 200 confirmation tasks: calls actually made, calls refused by the per-arm budget before being sent (their recorded token value is a budget computation, not a request), the range of `max_tokens` of the calls made, calls whose output reached that limit (completion tokens = `max_tokens` in the trace), execution errors, and the error cause read from the visible-check record (syntax error / missing entry point / unknown), split by whether the output limit was reached.

Arm	Called	Refused	max_tokens (called)	At limit	Errors	Error causes
B-expert	199	1	432–757 (median 644)	71	49	entry point missing, output limit: 5; syntax error, output limit: 44
B-solo	200	0	887–1185 (median 965)	0	0	–

Exploration diagnostics. These counts were computed after the policy freeze and were not part of the primary test. On the 100 exploration tasks, B-solo passed 79 and the union of observed action successes covered 82. B-expert solved 0 tasks that B-solo missed, whereas B-solo solved 33 that B-expert missed. This is a descriptive union of realised outputs, not an estimate of the probability oracle in Appendix F. B-expert reached its output limit in 43 calls, including all 32 execution errors. Among those errors, 31 were syntax errors and one lacked the entry point; their output ceilings ranged from 381 to 697 tokens. A-self$\times K$ had a revision call refused by its remaining budget in 10 tasks. No larger-budget comparison was run.

E EXPLORATION MODEL SELECTION

Table 11: Exploration set ($N_e = 100$, $r = 1$): per-arm hidden-test pass rate, refusal rate and own cost.

Arm	n	Hidden pass (%)	Refused (%)	Own cost / task (USD)	Calls / task
T	100	67.0	0.0	0.0000	0.00
A-self	100	70.0	0.0	0.0039	1.00
A-self$\times K$	100	67.0	10.0	0.0018	0.45
A-role	100	69.0	0.0	0.0041	1.00
B-expert	100	46.0	0.0	0.0103	1.00
B-solo	100	79.0	0.0	0.0062	1.00

Table 12: Exploration-set model selection (nested GroupKFold CV, $K = 5$, 20 repeats): CV gain of each policy over the best fixed arm, per λ. * = selected. This is model selection, not the primary test.

Policy	λ	CV gain (pp)	95% CI	Raw features
P2-post	0.3*	-1.6	[-7.8, 3.2]	18
P2-post	1	-2.0	[-7.1, 3.4]	18
P2-post	3	-1.7	[-6.3, 3.5]	18
P2-pre	0.3	-3.8	[-10.6, 2.9]	17
P2-pre	1	-3.1	[-9.5, 3.1]	17
P2-pre	3*	-2.8	[-8.4, 3.5]	17
P1	–	0.0	–	–

Feature selection is repeated inside training folds; the repeated CV scores select λ. There is no outer CV assessment of that choice. The confirmation holdout provides the independent evaluation of the frozen policy.

Concretely, for each λ on the menu: GroupKFold ($K = 5$, tasks as groups, 20 repeats) in which, inside every training fold, the model is fit once on the full menu, numeric columns are ranked by the mean absolute standardised coefficient across arms and truncated so that at most 20 columns remain (categorical columns are kept and counted), standardisation and category vocabularies are fixed on the training fold, and the held-out fold is scored by the gain of the policy's pick over the best fixed arm. The λ with the largest CV gain is chosen (ties go to the smaller value); the frozen subset is then selected once on the whole exploration set with that λ, and the model is refit once. The same function implements this procedure in the executor and in the synthetic study, so the reported operating characteristics are those of the procedure that was actually run.

On the 30-task repeated subset, observed flip rates were 3.3% for A-self and 0.0% for B-expert (30 tasks each). Exploration spent $3.3160 across 556 ledger calls, with 0 incomplete episodes. This includes calibration and repeated calls, not just the main action cells.

F POLICY GAIN AND ORACLE QUANTITIES

Let q_{ia} be the success probability of action a conditional on task i and its first-pass state, and let $a^* = \arg\max_a \mathbb{E}_i[q_{ia}]$ be the population-optimal fixed action. The expectation averages over tasks and first-pass states. Three quantities must not be confused. (1) The *true-probability oracle gain over the frozen baseline*, $\mathbb{E}_i[\max_a q_{ia}] - \mathbb{E}_i[q_{i\hat{a}}]$, which bounds $\Delta(\pi, \hat{a})$ for every policy π and is observable only in simulation; the related quantity $\mathbb{E}_i[\max_a q_{ia}] - \max_a \mathbb{E}_i[q_{ia}]$ is the oracle gain over the population-optimal fixed arm and differs from the former by the selection loss $\mathbb{E}_i[q_{ia^*}] - \mathbb{E}_i[q_{i\hat{a}}] \geq 0$, so it is a bound on the gain over a^*, not on the gain over $\hat{a}$. (2) The *in-sample maximum of single realisations*, $\frac{1}{n}\sum_i \max_a y_{ia} - \frac{1}{n}\sum_i y_{i\hat{a}}$, which contains realisation luck and is reported only as a descriptive statistic, never as an estimate of (1). (3) The gain of a policy learned from observable features, which is what we test.