RA-MoWE: Workflow-Affinity Embeddings for Query Clustering and Agentic Workflow Generation

Anonymous authors
Paper under double-blind review

Abstract

Agentic workflows enable large language models (LLMs) to solve complex tasks by coordinating reasoning, tool use, and verification. However, a workflow optimized for an entire task collection can overlook differences in the reasoning strategies that individual queries need, while searching for a new workflow for every query repeats costly optimization. To address this tradeoff, we introduce RA-MoWE, a framework that uses workflow-affinity embeddings to cluster queries and guide the generation of reusable expert workflows. Each embedding records how well a fixed set of reference workflows solves a query, revealing similarities in which reasoning strategies are effective. RA-MoWE uses each cluster's queries and average embedding to initialize and refine a specialized workflow through execution feedback. An embedding encoder predicts these embeddings from query text, allowing new queries to select a generated expert without first executing the reference workflows. With predicted affinities, RA-MoWE achieves 42.90% on MixBench-H and 61.7% on MATH, using 1.98 language-model calls per MixBench-H query. With measured affinities, it reaches 62.1% on MATH and resolves 24.50% of SWE-bench Verified issues, indicating setting-dependent benefits from reusable expert workflows.

1 Introduction

Agentic workflows organize large language model (LLM) calls into procedures for reasoning, revising answers, and using tools, extending their capabilities on complex tasks (Yao et al., 2023b; Madaan et al., 2023). Their design determines which operations run and how they interact, affecting both solution quality and inference cost: repeated sampling, revision, and decomposition offer different ways to spend computation. Manually choosing a suitable design for a diverse collection of queries is therefore a substantial engineering effort (Hu et al., 2025; Zhang et al., 2025).

Recent methods automate workflow design through search and learned generation. ADAS and AFlow explore code-represented agents using execution feedback, while AgentSquare searches combinations of reusable modules (Hu et al., 2025; Zhang et al., 2025; Shang et al., 2025). In these works, the workflow is often optimized over a collection of tasks using a collection-level score and thus can overlook differences in individual query need. In contrast, some approaches, such as FlowReasoner (Gao et al., 2025), learn to generate a workflow for each query. These methods need to repeat costly optimization and also have less training samples for each workflow. Addressing such tradeoff requires identifying which queries should share a workflow and what computations each group needs. While a natural solution is to group queries based on their similarity. However, conventional semantic similarity does not necessarily capture whether queries benefit from the same reasoning strategies.

Our key idea is to introduce a *workflow-affinity embedding* and *group queries by how they respond to different reasoning workflows*. In particular, the *workflow-affinity embedding* is a vector whose coordinates estimate a query's task score under a fixed set of reference workflows, called probes. The resulting vector captures a query's relative strengths and weaknesses across different reasoning strategies, including which reasoning strategies succeed, partially succeed, or fail. Queries with

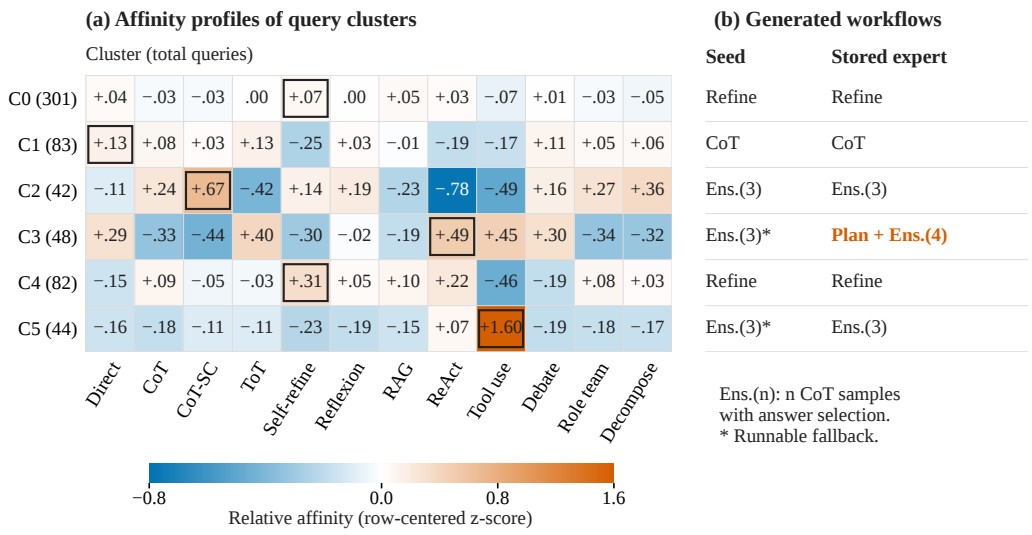

Figure 1: **Different workflow preferences motivate specialized experts.** (a) Average embeddings for six GPT-4o-mini training-query clusters show distinct responses to reference workflows. C2 favors sampling and answer selection (CoT-SC), whereas C4 favors self-refinement. (b) RA-MoWE uses these preferences to choose initial workflows. Values are standardized scores centered within each row; outlines mark row maxima, and parentheses give full cluster sizes. Refine revises a chain-of-thought (CoT) answer; Ens.(n) selects among n sampled CoT answers; Plan adds decomposition. Asterisks mark runnable substitutes for unavailable workflows. Measurement details are in Appendix C.3.

similar profiles can therefore be grouped together based on their computational needs rather than their semantic similarity.

We integrate this idea into RA-MoWE, a framework that uses the workflow-affinity embeddings to guide the generation of reusable expert workflows. Offline, we execute the reference workflows on training queries, compute their embeddings, and partition them into clusters. Clustering these embeddings groups queries with similar measured responses, giving expert construction both a focused set of examples and evidence about potentially useful reasoning strategies. For each cluster, we introduce an embedding-guided search mechanism, which uses the cluster prototype to choose an executable starting workflow and constrain the candidate operations. It then proposes refinements informed by execution feedback and probe outcomes on remaining failures, retaining edits that improve the cluster's validation score. The resulting expert is stored for reuse. Figure 1 connects the measured cluster affinities to the workflows produced by this process.

We then create an *embedding encoder*, which makes the generated experts accessible from query text alone. While workflow-affinity embeddings can be measured offline by executing the reference workflows, this process is too costly to repeat for every new query. We therefore train the encoder to predict a query's workflow-affinity embedding directly from its text, using the measured embeddings as supervision. At deployment, the predicted embedding selects a cluster's expert without executing the reference workflows.

We evaluate RA-MoWE across mathematics, science, programming, and repository repair against fixed prompting and four adapted workflow-search methods. On 300 test queries drawn from four benchmarks in mathematics, science, and programming, a supplementary guided-search experiment improves average task score by 4.04 percentage points over a system that selects from the reference workflows, while using 27.7% fewer LLM calls at inference. Both systems choose a workflow based on the query text. Further experiments compare query-cluster distributions, evaluate cluster experts across task families and within repository repair, and examine offline and inference costs.

Our contributions are:

1. **Workflow-affinity embeddings** that represent queries through measured workflow outcomes and organize training examples for specialist construction.
2. **An embedding encoder** that predicts this representation from text, enabling expert selection without online reference-workflow execution.
3. **Embedding-guided expert generation** that uses cluster affinities, query examples, and failure feedback to initialize and refine reusable workflows.

Our analysis identifies when measurement and prediction errors preserve useful decisions and when the available workflow operations support transfer from a strong probe to a generated expert.

2 RA-MoWE: EMBEDDING-GUIDED WORKFLOW GENERATION

Given training queries with reference answers or executable tests, our goal is to construct reusable workflows that serve different computational needs. RA-MoWE uses a shared representation to connect three decisions: which queries should be grouped together, how to build an expert for each group, and which expert should answer a new query. We first measure each query's response to reference workflows and cluster the resulting embeddings. These embeddings guide expert generation and supervise an encoder that selects an expert from query text. Figure 2 illustrates the framework; Algorithm 1 summarizes its construction and deployment.

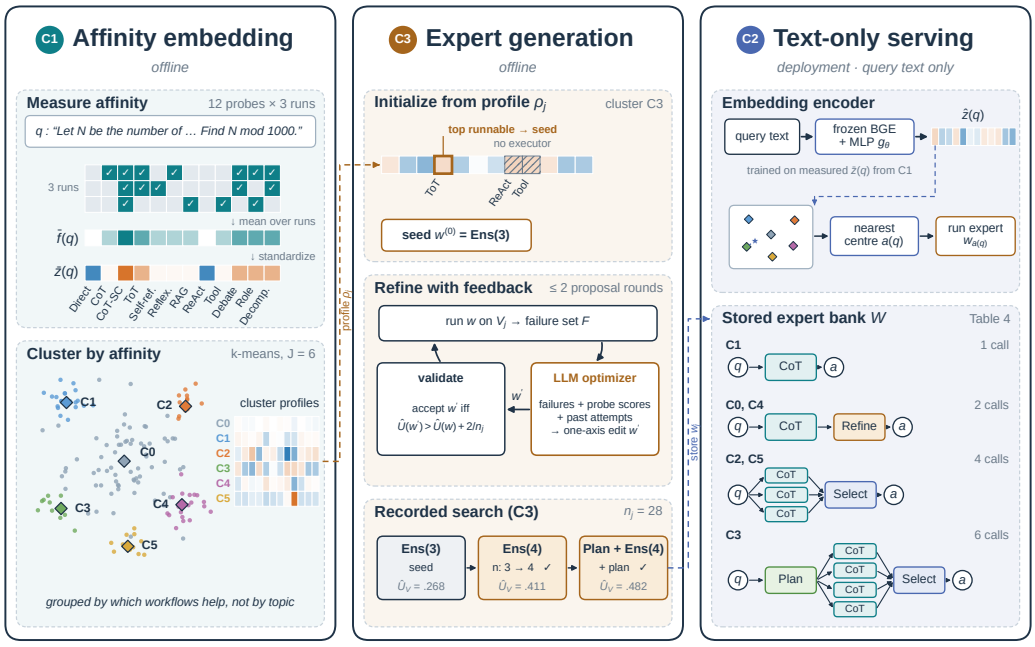

Figure 2: The RA-MoWE framework. Offline, reference-workflow scores define query embeddings for clustering, expert generation, and encoder supervision. At deployment, the encoder predicts an embedding from query text and selects a stored expert. C1–C3 mark the embedding, encoder, and generation contributions. Concrete cluster profiles, search edits, and expert costs illustrate the supplementary guided-search experiment (Appendix C.3).

2.1 WORKFLOW-AFFINITY EMBEDDINGS FOR QUERY CLUSTERING

A useful query group should reflect which reasoning procedures help its members. Questions about the same topic can require different computations, such as comparing several candidate answers or revising an initial solution. We therefore represent a query by its performance under a fixed panel of diverse reference workflows, called *probes*. These probes (Table 1) apply established strategies such as chain-of-thought, self-consistency, and self-refinement (Wei et al., 2022; Wang et al., 2023; Madaan et al., 2023).

Table 1: The twelve reference-workflow probes. The right column applies only to the supplementary guided-search interpreter (Appendix C.3), whose base candidates always use CoT; these mappings are approximations of the measured interventions.

Probe	Measured computational intervention	Guided interpreter mapping
Direct	One formatted response	Direct flag is inert; base candidate still uses CoT.
CoT	Step-by-step reasoning prompt (Wei et al., 2022)	CoT base candidate.
CoT-SC	Five inner samples plus answer aggregation (Wang et al., 2023)	Sampling and selection; seed uses three samples.
ToT	Bounded branching, evaluation and search (Yao et al., 2023a)	Sampling and selection; no tree search.
Self-refine	Critique and revision (Madaan et al., 2023)	One checking/rewrite call.
Reflexion	Outcome feedback, reflection and retry (Shinn et al., 2023)	Same revision flag; no separate feedback-memory loop.
RAG	Same-benchmark TF-IDF exemplar retrieval; query excluded	CoT flag; no retrieval in generated expert.
ReAct	Bounded action–observation interaction (Yao et al., 2023b)	Unavailable; cannot seed its behavior.
Tool use	Python execution or public-test feedback, domain dependent	Unavailable; cannot seed its behavior.
Debate	Independent debaters and a judge (Du et al., 2024)	Two fixed persona candidates plus selection.
Role team	Three complementary roles and a coordinator	Same two-persona branch as debate.
Decompose	Subproblem decomposition and synthesis (Zhou et al., 2023)	One plan prepended to subsequent candidate calls.

For a query q and probe p_k, let $Y_{k,s}(q) \in [0,1]$ be the task score from execution s under a fixed execution and scoring protocol. Averaging S repetitions gives

$$\bar{f}_k(q) = \frac{1}{S}\sum_{s=1}^{S} Y_{k,s}(q), \qquad \bar{f}(q) = \big(\bar{f}_1(q), \ldots, \bar{f}_K(q)\big)^\top, \tag{1}$$

where K is the number of probes. The vector $\bar{f}(q)$ retains the strengths and weaknesses of all reference workflows, including partial successes and multiple successful strategies. We standardize its coordinates using training statistics to obtain the *workflow-affinity embedding* $\bar{z}(q)$. This prevents differences in score variance from dominating the comparison between queries; the normalization and measurement settings are given in Appendix B.

We cluster the training queries in the embedding space using k-means, yielding J groups with centers $C = (c_1, \ldots, c_J)$. Each group supplies examples for a reusable expert. Grouping queries by their workflow responses therefore provides workflow construction with both relevant examples and evidence about which reasoning strategies are effective for those examples. Such evidence also has a precise interpretation within the probe panel: if two queries' expected probe scores differ by at most ϵ in every coordinate, transferring the best probe from one to the other loses at most 2ϵ in expected score. Proposition 1 and its proof relate this decision guarantee to measurement accuracy. The next step uses the resulting groups to construct workflows beyond the reference panel.

2.2 Embedding-guided expert generation

Each cluster supplies two complementary inputs for expert generation: (1) examples to optimize and (2) evidence about useful reasoning operations. For cluster j, let V_j be the subset of its training queries used to evaluate candidate workflows. We summarize their measured embeddings by

$$\rho_j = \frac{1}{|V_j|}\sum_{q \in V_j} \bar{z}(q), \tag{2}$$

Algorithm 1 RA-MoWE: constructing and using a bank of workflow experts

Require: Training queries and task scorers; reference workflows; cluster count J; search resources
1: Measure and standardize the workflow-affinity embedding $\bar{z}(q)$ of each training query
2: Train the embedding encoder to predict $\bar{z}(q)$ from query text
3: Cluster the measured embeddings into J groups with centers C
4: **for** each cluster j **do**
5: Choose construction queries V_j and compute their average embedding ρ_j
6: Initialize w_j from a high-affinity runnable probe; restrict candidate operations using ρ_j
7: **for** each allowed search round **do**
8: Use probe scores and remaining failures to guide an LLM-proposed edit
9: Evaluate on V_j; accept only if the score improvement exceeds the required margin
10: **end for**
11: Store the final expert w_j
12: **end for**
13: **return** Trained encoder, cluster centers C, and expert bank $\mathcal{W}$
Deployment: predict a new query's embedding, select the nearest cluster, and run its expert.

where $|V_j|$ is the number of queries in this subset. The prototype vector ρ_j describes the cluster's relative strengths across the reference workflows. It provides search with a starting strategy and evidence for choosing subsequent operations.

Search begins by mapping the highest-ranked runnable probe in ρ_j to an initial workflow. We construct workflows from executable operations such as planning, sampling, answer selection, and revision, following the modular-search principle of Zhang et al. (2025); Shang et al. (2025). The same embedding restricts candidate operations using the lowest-ranked probes, while retaining the operation chosen for initialization. In this way, the measured response guides both where search starts and which changes it considers. The value of the initialization depends on how well the executable workflow preserves the probe's useful behavior; Proposition 2 characterizes this connection.

Refinement focuses on the queries the current workflow has not solved. An LLM optimizer receives failure examples, summaries of probe scores on failed and solved queries, and previous search attempts. These summaries identify strategies that may address the remaining failures. The optimizer proposes an edit, which is executed on V_j and retained only if its average score improves by a prescribed margin. Repeating this process produces expert w_j; the collection $\mathcal{W} = \{w_1, \ldots, w_J\}$ forms the reusable expert bank. The operator mapping, proposal rules, and acceptance margin are specified in Appendix B. Search uses construction data, while the separate test pool evaluates the resulting bank.

Algorithm 1 presents the shared-clustering framework; the recorded alignment between construction and deployment clusters is detailed in Appendix C.3.

2.3 Learning the embedding encoder

The embeddings used for construction require scored probe executions. To select an expert for a new query without repeating that measurement, we train an encoder to predict the same coordinates from text. Let $\psi(q)$ be frozen text features and g_θ a trainable prediction head with parameters θ. The predicted embedding is

$$\hat{z}(q) = g_\theta(\psi(q)) \in \mathbb{R}^K. \tag{3}$$

Training pairs each query with its measured embedding $\bar{z}(q)$. We use frozen BGE features (Xiao et al., 2024) as inputs to a small prediction head, which is trained to predict the measured embedding from query text. The loss encourages agreement in the relative pattern of probe responses and penalizes errors in the coordinates used for clustering. Predicting the full embedding also allows the same learned target to support different cluster partitions. The loss, architecture, and training protocol are specified in Appendix B.

At deployment, the predicted embedding selects the closest cluster center:

$$a(q) = \arg\min_{1 \le j \le J} \|\hat{z}(q) - c_j\|_2^2. \tag{4}$$

Table 2: Evaluation data. Development counts describe the pool retained for the reported method; the protocol paragraph explains the baseline pool. MixBench-H contains the four datasets above its total row.

Dataset	Development	Test	Task score
AIME	200	100	Answer correctness
GPQA	200	100	Answer correctness
CodeContests	94	50	Public-test pass fraction
LiveCodeBench	100	50	Public-test pass fraction
MixBench-H (total)	594	300	Mean query score
MATH	119	486	Answer correctness
SWE-bench Verified	300	200	Harness resolution

Here c_j is the center of cluster j, $\|\cdot\|_2$ is the Euclidean norm, and $a(q)$ is the selected expert's index. We then execute the stored workflow $w_{a(q)}$ on q. Both probe measurement and workflow construction are therefore performed offline and reused across future queries.

The prediction need not recover every coordinate exactly to preserve an expert choice. If the prediction error is smaller than the measured embedding's distance to the nearest cluster boundary, the two embeddings select the same expert. Proposition 3 and Appendix A.3 formalize this condition and its counterpart for expected probe scores. This motivates evaluating the encoder through both prediction error and the resulting expert assignments.

3 Empirical evaluation

We examine whether workflow-affinity embeddings group queries differently from domains and semantic embeddings, whether the resulting cluster experts solve queries across and within task families, and how much computation construction and deployment require. We first describe the experimental setup, then compare query-cluster distributions. We next report task scores on MixBench-H, MATH, and SWE-bench Verified, use MATH ablations to examine search, affinity guidance, and expert selection, and compare offline and inference costs.

3.1 Experimental setup

Datasets and metrics. All reported experiments use GPT-4o-mini. MixBench-H combines AIME, GPQA, CodeContests, and LiveCodeBench (Mathematical Association of America, 2026; Rein et al., 2024; Li et al., 2022; Jain et al., 2025); Table 2 summarizes the training and test queries. Its task score averages answer correctness or the fraction of public coding tests passed, weighting queries equally. SWE-bench Verified (Jimenez et al., 2024) uses a repository-stratified training/test split and measures harness resolution. Its cluster experts select policy instructions within an Agentless-lite localize–repair pipeline (Xia et al., 2025). MATH (Hendrycks et al., 2021) uses the released AFlow validation/test split (Zhang et al., 2025) and measures answer accuracy. MixBench-H and SWE-bench use project evaluation splits rather than official leaderboard protocols; failed executions remain in their denominators.

Methods. **RA-MoWE (measured)** selects a cluster expert using a measured workflow-affinity embedding. **RA-MoWE (predicted)** uses the embedding encoder to predict that embedding from query text, avoiding reference-workflow executions at deployment. Measured assignment therefore requires scored probe executions on the test query. We compare both variants with IO, a single-prompt baseline, and adapted implementations of AFlow, ADAS, ScoreFlow, and FlowReasoner (Zhang et al., 2025; Hu et al., 2025; Wang et al., 2025; Gao et al., 2025). In the MATH ablation, RA-MoWE (full) denotes the predicted variant with embedding-guided search and query-dependent expert selection (routing). Their search procedures and budgets are specified in Appendix C.1.

Evaluation protocol. The MixBench-H baselines use the earlier implementation (v1), while the reported RA-MoWE variants use the updated implementation (v2). Both evaluate exactly the same 300 test queries. The earlier training pool contains 600 queries; the updated pool removes six Code-

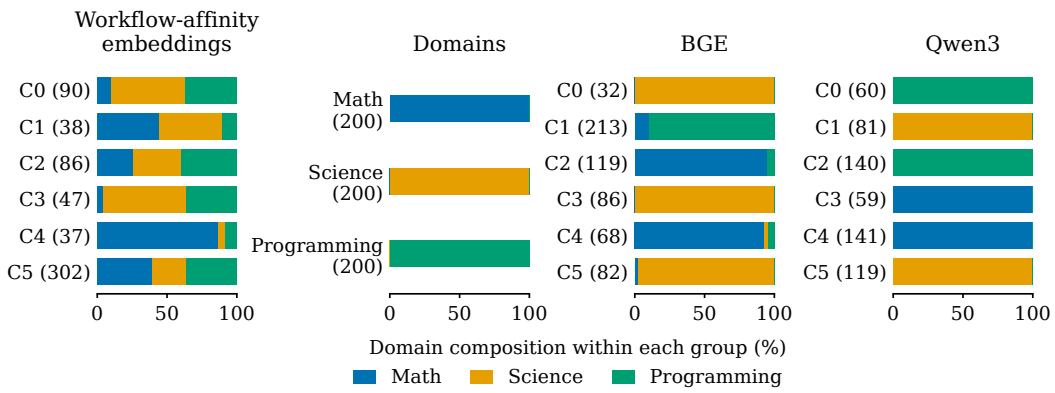

Figure 3: Domain composition of query groups. Natural domains define three groups; workflow-affinity, BGE, and Qwen3 embeddings each define six. Bars show within-group percentages, and parentheses give group sizes. Cluster indices are local to each partition. All panels use the same 600 training queries (Appendix D).

Contests records and assigns the remaining 594 to construction, selection, and calibration subsets of 414, 90, and 90 queries. The update also changes prompts, execution, and scoring, including support for functional coding tests skipped by the earlier scorer. Consequently, the combined table compares complete configurations and does not isolate the effect of the framework changes. The SWE-bench results retain the earlier Agentless-lite evaluation for every method, including both RA-MoWE variants; no updated SWE-bench run is implied. Cluster-distribution plots use the saved 600-query training pool. Additional execution settings and uncertainty estimates appear in Appendix C.

MATH protocol. The 119 development problems are divided into 71 construction and 48 held-out selection problems. We measure twelve probes with three repetitions, form three construction clusters of sizes 61, 6, and 4, and allow up to sixteen search proposals per supported cluster; one cluster retains a support fallback. Expert selection uses the held-out problems with a pooled empirical-Bayes gate, and the embedding encoder uses BGE-large text features. Each RA-MoWE configuration has one test execution on 486 problems. Baselines use AFlow's evaluator; RA-MoWE uses the project harness. Both obtain 54.1% for IO, which checks that control but does not establish evaluator equivalence. The stored MATH searched-arm executions were affected by a sandbox timeout defect; their component comparisons remain conditional on that execution setting. FlowReasoner additionally uses answer labels during search, averages three final executions per completed problem, and counts two failed problems as zero. Further measurement details appear in Appendix C.2.

3.2 Query clustering with workflow-affinity embeddings

Workflow-affinity embeddings produce groups that cross domain boundaries. Figure 3 compares partitions of the same 600 training queries. Every workflow-affinity cluster contains mathematics, science, and programming queries; BGE clusters largely follow these domains, and Qwen3 clusters are domain-pure. Thus, measured workflow responses induce a different grouping from topical similarity. This distribution comparison describes membership, while the following experiments assess task performance.

3.3 Main results across and within task families

Across task families. RA-MoWE (predicted) reaches a 42.90% average task score on MixBench-H, compared with 42.81% for RA-MoWE (measured). The predicted variant scores 29.50% on AIME, 35.50% on GPQA, 42.58% on CodeContests, and 84.83% on LiveCodeBench (Table 6). These scores describe performance across different computational needs. In this completed search, every selected expert retains a reference workflow; the result evaluates their construction and selection without establishing an additional benefit from newly generated workflow designs.

Table 3: Task performance (%, ↑): MixBench-H task score, MATH accuracy, and SWE-bench Verified resolution. The experimental setup specifies the evaluation protocols.

Method	MixBench-H	MATH	SWE-bench Verified
IO	28.72	54.1	20.00
AFlow	27.22	53.9	19.00
ADAS	20.42	42.8	20.00
ScoreFlow	27.56	49.0	21.00
FlowReasoner	35.65	53.2	20.50
RA-MoWE (measured)	42.81	62.1	24.50
RA-MoWE (predicted)	42.90	61.7	20.00

Table 4: MATH ablations on 486 problems. Accuracy intervals use 10,000 query-bootstrap draws; Δ is relative to the full configuration, in percentage points, computed before rounding. Tokens are per test query.

Variant	Acc. (%) ↑	95% CI	Δ full	Tokens/query ↓
RA-MoWE (full)	**61.7**	57.4–65.8	0.0	4,082
RA-MoWE - routing	62.6	58.2–66.7	+0.8	4,240
RA-MoWE - affinity guidance	59.3	54.9–63.6	-2.5	5,527
RA-MoWE - affinity guidance - routing	60.1	55.8–64.4	-1.6	5,790
RA-MoWE - search	58.4	54.1–62.8	-3.3	2,782

Within mathematics. On MATH, RA-MoWE (measured) scores 62.1% and the full predicted-embedding configuration scores 61.7%, compared with 54.1% for IO and 53.9% for AFlow. The two affinity configurations therefore have similar point estimates on this task. Section 3.4 examines which components contribute to the observed performance.

Within repository repair. RA-MoWE (measured) resolves 49 of 200 SWE-bench issues (24.50%); RA-MoWE (predicted) resolves 40 (20.00%), selecting the IO policy throughout. The saved probe outcomes nevertheless show variation within this task family: the union of twelve policies resolves 66 issues, compared with 42 for the test-selected best fixed policy. Even within Django, the corresponding counts are 34 and 26 of 92 issues; IO resolves five issues missed by retrieval, and retrieval resolves four missed by IO. These single-execution, outcome-selected counts illustrate complementary workflow outcomes rather than an achievable prediction rule (Table 7).

3.4 Ablation study

We use MATH to examine expert search, affinity guidance, and query-dependent expert selection (Table 4). RA-MoWE (full) combines all three. Removing routing applies one workflow from the same bank of probes and searched experts to every query, selected on all development problems. Removing affinity guidance keeps the clusters, search, and routing but hides probe-score summaries and preferences from the proposer. Removing search uses one development-selected probe; this control also removes query-dependent selection, so it does not isolate search alone.

Affinity guidance is associated with a 2.5-point higher accuracy than unguided search and lower token use ($4{,}082$ versus $5{,}527$). The full configuration also exceeds the probe-only control by 3.3 points, at additional inference cost. However, routing does not improve accuracy here: the selected fixed workflow scores 62.6%, versus 61.7% for the full configuration, while using slightly more tokens. The paired intervals for full versus unguided search and versus the probe-only control include zero. These comparisons identify component tradeoffs; they do not establish an accuracy improvement from every component.

3.5 Offline and inference costs

Table 5 reports inference usage on MixBench-H, separating cluster-expert execution from the probe executions needed to measure an embedding. The IO, chain-of-thought, and fixed-probe controls

Table 5: MixBench-H inference usage per query. Expert calls and tokens exclude embedding measurement; probe tokens are additional. Cluster-expert usage averages two executions per query.

Configuration	Expert calls	Expert tokens	Probe tokens
IO	1.000	834	0
Chain-of-thought	1.000	1,064	0
Fixed probe	1.850	2,715	0
RA-MoWE (predicted)	1.977	2,038	0
RA-MoWE (measured)	1.398	1,567	114,103

use the same executor as the reported RA-MoWE configurations. Local embedding-encoder computation is additional.

The embedding encoder avoids online reference-workflow execution. The predicted variant uses about 1.98 LLM calls and $2{,}038$ tokens per query. The measured variant's selected expert is cheaper, but obtaining its embedding adds 93.02 calls and $114{,}103$ tokens, bringing its total to about 94.42 calls and $115{,}670$ tokens. These measurements distinguish the cost of an expert from the cost of selecting it.

Retained offline LLM work costs an estimated USD 58.55 for the predicted configuration and USD 45.76 for the measured configuration, using the experiment's configured input/output token rates. These standalone estimates each include shared measurement and calibration, so they should not be added as total expenditure. They exclude local fitting, harness compute, and discarded failed attempts. Appendix C.4 gives the accounting; no complete amortization break-even is inferred.

4 Discussion and conclusion

Workflow-affinity embeddings connect query clustering, embedding-encoder training, and expert generation through measured workflow outcomes. The representation groups queries by their responses to computational strategies and supplies the same evidence to expert construction. Its clusters cross domain boundaries, while the experiments show complementary workflow outcomes across task families and within repository repair. The predicted variant makes those experts accessible from query text, avoiding reference-workflow execution at deployment.

The benefit of query-dependent expert selection varies across settings. On MixBench-H, the predicted variant reaches 42.90% with about two LLM calls per query; on MATH, its 61.7% accuracy is close to the measured variant's 62.1%. However, a single development-selected workflow performs slightly better than routing on MATH, and the SWE-bench encoder selects one policy throughout. These outcomes show that heterogeneous workflow responses alone do not guarantee a benefit from predicted assignment. A useful encoder must preserve the distinctions that change which available expert should execute, and those experts must offer complementary advantages on new queries.

A priority is to separate the contributions of representation, search, and selection more precisely. The MixBench-H bank retains reference workflows, and the MATH probe-only control removes both search and routing. Future experiments should compare searched and unsearched banks under the same assignments, use matched guided and unguided proposal budgets, and repeat construction with independent seeds. These controls would test whether affinity guidance improves expert construction beyond identifying a strong initial workflow. Larger construction pools would also permit more reliable selection in small clusters without relying on a fallback.

Another direction is to make measurement and prediction more selective. A fixed probe panel can miss useful computational strategies, while measuring every probe repeatedly is expensive. Adaptive probe allocation could concentrate executions where workflow preferences are uncertain, and an uncertainty-aware embedding encoder could identify queries near cluster boundaries before committing to an expert. Such extensions require evaluation of both task performance and total computation, including local fitting and tool execution. Testing different backbones and shifted query distributions would further establish when learned workflow preferences transfer and when the probe panel or expert bank needs to be rebuilt.

AI USE STATEMENT

We used generative AI tools for create or modify scientific figures or images, and edit the research paper to improve readability. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Beijing Academy of Artificial Intelligence. BAAI/bge-large-en-v1.5: Model card, 2023. URL https://huggingface.co/BAAI/bge-large-en-v1.5. Accessed September 16, 2026.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Hongcheng Gao, Yue Liu, Yufei He, Longxu Dou, Chao Du, Zhijie Deng, Bryan Hooi, Min Lin, and Tianyu Pang. FlowReasoner: Reinforcing query-level meta-agents. *arXiv preprint arXiv:2504.15257*, 2025. URL https://arxiv.org/abs/2504.15257.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi: 10.1080/01621459.1963.10500830.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/36b7acf6f6010652b3f2a433774a66fe-Abstract-Conference.html.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/94074dd5a072d28ff75a76dabed43767-Abstract-Conference.html.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022. doi: 10.1126/science.abq1158. URL https://arxiv.org/abs/2203.07814.

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. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Mathematical Association of America. American invitational mathematics examination, 2026. URL https://maa.org/maa-invitational-competitions/. Official competition resource; accessed September 16, 2026.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. AgentSquare: Automatic LLM agent search in modular design space. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/0ae94013da7cd459402fd77874e09ee3-Abstract-Conference.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yinjie Wang, Ling Yang, Guohao Li, Mengdi Wang, and Bryon Aragam. ScoreFlow: Mastering LLM agent workflows via score-based preference optimization. *arXiv preprint arXiv:2502.04306*, 2025. URL https://arxiv.org/abs/2502.04306.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying LLM-based software engineering agents. *Proceedings of the ACM on Software Engineering*, 2(FSE), 2025. doi: 10.1145/3715754. URL https://doi.org/10.1145/3715754.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 641–649, 2024. doi: 10.1145/3626772.3657878. URL https://sigir-2024.github.io/proceedings.html.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://proceedings.neurips.cc/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

A PROOFS OF THE METHOD PROPOSITIONS

These propositions formalize the three local claims used in Sections 2.1–2.3. They concern a fixed execution and scoring protocol; they do not certify the performance of the complete learned system. Let $Q \sim \mathcal{D}$, $f_k(q) = \mathbb{E}[Y_{k,s}(q) \mid q]$, and $\bar{f}_k(q)$ be the measured mean. With fixed training-fitted means μ and positive coordinate scales $\Lambda = \mathrm{diag}(\sigma_1, \ldots, \sigma_K)$, define $z(q) = \Lambda^{-1}(f(q) - \mu)$ and $\bar{z}(q) = \Lambda^{-1}(\bar{f}(q) - \mu)$. The encoder predicts $\hat{z}(q) = g_\theta(\psi(q))$. Routing statements condition on the fitted encoder and distinct centroids; r_C selects the nearest centroid using a fixed tie rule.

A.1 MEASUREMENT AND PROBE TRANSFER

Proposition 1 (Affinity estimation preserves probe decisions)**.** *Fix a query and protocol. If each probe's S repetitions are independent, identically distributed, and bounded in $[0, 1]$, then*

$$\Pr\big(\|\bar{f}(q) - f(q)\|_\infty > \epsilon\big) \le 2Ke^{-2S\epsilon^2}. \tag{5}$$

On the event $\|\bar{f} - f\|_\infty \le \epsilon$, selecting $\hat{k} \in \arg\max_k \bar{f}_k(q)$ incurs expected probe regret at most 2ϵ.

Proof of Proposition 1. For each fixed q and k, independent bounded executions give

$$\Pr\big(|\bar{f}_k(q) - f_k(q)| > \epsilon \mid q\big) \le 2\exp(-2S\epsilon^2) \tag{6}$$

by Hoeffding's inequality (Hoeffding, 1963). A union bound over K coordinates proves the stated $2K\exp(-2S\epsilon^2)$ bound. Independence between different probes is not required. Let $k^* \in \arg\max_k f_k(q)$ and $\hat{k} \in \arg\max_k \bar{f}_k(q)$. On the uniform accuracy event,

$$f_{k^*}(q) - f_{\hat{k}}(q) = f_{k^*}(q) - \bar{f}_{k^*}(q) + \bar{f}_{k^*}(q) - \bar{f}_{\hat{k}}(q) + \bar{f}_{\hat{k}}(q) - f_{\hat{k}}(q) \tag{7}$$

$$\le \epsilon + 0 + \epsilon. \tag{8}$$

Thus a unique best probe with population gap exceeding 2ϵ is recovered. For transfer between two queries, replace $\bar{f}(q)$ in this argument by $f(q')$. If $\|f(q) - f(q')\|_\infty \le \epsilon$, using a best probe for q' on q loses at most 2ϵ relative to the best probe for q. □

The bound concerns decisions among the measured probes, rather than unseen workflows. With $K = 12$, $S = 3$, and confidence 0.95, its sufficient radius $\sqrt{\log(2K/0.05)/(2S)} \approx 1.014$ is vacuous for scores in $[0, 1]$; three repetitions do not provide a numerical ranking certificate. Fixed standardization gives $\|f(q) - f(q')\|_\infty \le \sigma_{\max}\|z(q) - z(q')\|_2$, so the transfer bound in standardized space includes the scale factor $\sigma_{\max} = \max_k \sigma_k$.

A.2 AFFINITY-DERIVED INITIALIZATION

Proposition 2 (Utility of an affinity-derived initialization)**.** *Fix a cluster. Let r_k be probe k's expected raw score and s_k its estimate, with $\max_k |s_k - r_k| \le \epsilon$. Let $\mathcal{R} \subseteq \{1, \ldots, K\}$ be the nonempty set of runnable probe indices and $\hat{k} \in \mathcal{R}$ the chosen probe. Define $\alpha = \max_k r_k - \max_{k \in \mathcal{R}} r_k$ and $\beta = \max_{k \in \mathcal{R}} s_k - s_{\hat{k}}$. If its mapped seed $v_{\hat{k}}$ has expected cluster utility $U(v_{\hat{k}}) \ge r_{\hat{k}} - \tau$, then*

$$U(v_{\hat{k}}) \ge \max_k r_k - \alpha - \beta - 2\epsilon - \tau. \tag{9}$$

Proof of Proposition 2. Suppress the cluster index. Let $\hat{k} \in \mathcal{R}$ be the selected runnable probe and $v_{\hat{k}}$ its executable seed. On the event $\max_k |s_k - r_k| \le \epsilon$, use the definitions $\alpha = \max_k r_k - \max_{k \in \mathcal{R}} r_k$ and $\beta = \max_{k \in \mathcal{R}} s_k - s_{\hat{k}}$ to obtain

$$\max_k r_k = \alpha + \max_{k \in \mathcal{R}} r_k \tag{10}$$

$$\le \alpha + \max_{k \in \mathcal{R}} s_k + \epsilon = \alpha + s_{\hat{k}} + \beta + \epsilon \tag{11}$$

$$\le \alpha + r_{\hat{k}} + \beta + 2\epsilon \le \alpha + U(v_{\hat{k}}) + \beta + 2\epsilon + \tau. \tag{12}$$

Rearranging proves the claim. Only the one-sided transfer condition $U(v_{\hat{k}}) \ge r_{\hat{k}} - \tau$ for the chosen seed is needed. □

The raw-score deficit β allows initialization by standardized rather than raw cluster scores. The transfer term τ accounts for differences between a measured probe and its executable approximation, including the mappings in Table 1. Neither term is assumed to vanish. The statement bounds an initialization relative to the probe panel; it does not establish that later adaptive edits improve expected utility.

A.3 Stability of expert assignment

Here $X = \psi(Q)$ is the frozen text representation of a fresh query, and $\mathcal{E}$ is the encoder's expected squared error relative to the population embedding.

Proposition 3 (A margin condition for faithful assignment)**.** *Condition on fixed distinct centroids and a fixed encoder. For a uniquely assigned point z, let $a = r_C(z)$ and*

$$m_C(z) = \min_{b \neq a} \frac{\|z - c_b\|_2^2 - \|z - c_a\|_2^2}{2\|c_b - c_a\|_2}. \tag{13}$$

Set $m_C(z) = 0$ on ties. If $\|\hat{z} - z\|_2 < m_C(z)$, the assignments agree. With $\mathcal{E} = \mathbb{E}\|g_\theta(X) - z(Q)\|_2^2$, for every $t > 0$,

$$\Pr\{r_C(g_\theta(X)) \neq r_C(z(Q))\} \leq \Pr\{m_C(z(Q)) \leq t\} + \mathcal{E}/t^2. \tag{14}$$

Proof of Proposition 3. Set $a = r_C(z)$ and write $\tilde{z} = z + e$. For every competing centroid b,

$$\|\tilde{z} - c_b\|_2^2 - \|\tilde{z} - c_a\|_2^2 = \|z - c_b\|_2^2 - \|z - c_a\|_2^2 + 2e^\top(c_a - c_b) \tag{15}$$

$$\geq 2\|c_b - c_a\|_2\big(m_C(z) - \|e\|_2\big). \tag{16}$$

If $\|e\|_2 < m_C(z)$, all comparisons are strictly positive, so the nearest centroid is unchanged. With $Z = z(Q)$, $X = \psi(Q)$, and $\mathcal{E} = \mathbb{E}\|g_\theta(X) - Z\|_2^2$, a disagreement can occur only if $m_C(Z) \leq t$ or $\|g_\theta(X) - Z\|_2 \geq t$. The union bound and Markov's inequality give

$$\Pr\{r_C(g_\theta(X)) \neq r_C(Z)\} \leq \Pr\{m_C(Z) \leq t\} + \mathcal{E}/t^2. \tag{17}$$

□

The deterministic margin argument also applies with $z = \bar{z}(q)$: prediction error below $m_C(\bar{z}(q))$ preserves that query's measured assignment. Agreement with a measured embedding need not imply agreement with the population embedding, because the probe measurements are noisy. The proposition preserves an assignment, not its quality, and does not make measured assignment an upper bound on predicted assignment.

B Implementation details

B.1 Probe measurement and normalization

Table 1 lists the twelve executable probe families and their approximations in the supplementary guided-search interpreter. The names identify local implementations with fixed prompts, budgets, tools, and scoring rules. Each measured coordinate averages three outer executions. The CoT-SC probe uses five inner samples within each execution; these are not five independent embedding observations.

Let $\mathcal{T}_{\text{enc}}$ be the encoder-training subset. We fit probe-coordinate means μ_k and standard deviations σ_k on this subset and define

$$\bar{z}_k(q) = \frac{\bar{f}_k(q) - \mu_k}{\sigma_k}, \qquad \bar{z}(q) = \Lambda^{-1}\big(\bar{f}(q) - \mu\big), \quad \Lambda = \operatorname{diag}(\sigma_1, \ldots, \sigma_K). \tag{18}$$

A fitted standard deviation below 10^{-8} is replaced by one. The fitted transform is reused for remaining construction queries and measured test embeddings. Positive coordinates indicate scores above a probe's training mean; they are not success probabilities.

B.2 EMBEDDING ENCODER AND CLUSTERING

The recorded 600-query mixed-task encoder uses L2-normalized frozen BGE-large-en-v1.5 features, truncated to 512 tokens (Beijing Academy of Artificial Intelligence, 2023). Its prediction head is $1024 \to 512 \to 512 \to K$, with ReLU and dropout 0.1 after each hidden layer. It minimizes

$$\mathcal{L}(\theta) = \frac{1}{N_{\text{enc}}} \sum_{q \in \mathcal{T}_{\text{enc}}} \left[\lambda\big(1 - \cos(\hat{z}(q), \bar{z}(q))\big) + \frac{1-\lambda}{K} \|\hat{z}(q) - \bar{z}(q)\|_2^2 \right], \quad \lambda = 0.7, \tag{19}$$

where $N_{\text{enc}} = |\mathcal{T}_{\text{enc}}|$. The prediction is already in standardized coordinates and is not normalized again at deployment. Adam uses learning rate 0.001, weight decay 0.0001, and batch size 32.

A stratified outer split reserves 90 of the 600 queries for encoder validation. Within the remaining 510, an inner 15% split selects the epoch count using the combined loss, with a maximum of 200 epochs and patience 25. A reinitialized head is then fitted on all 510 outer-training queries for that count. Initialization and the outer split use seed 13; the inner split uses seed 14. Normalization is fitted on the outer-training fold, including its inner-selection rows, and excludes the outer-validation fold.

This configuration uses Euclidean k-means with six clusters, seed 13, and ten restarts. Clustering includes all construction queries, including those held out from the encoder fit, with a fixed tie rule. The supplementary guided experiment uses two saved clusterings aligned by a training-fitted label mapping (Appendix C.3). The reported MixBench-H, SWE-bench, and MATH configurations use the dataset-specific splits, cluster counts, and selection procedures stated in Appendix C; the 600-query settings above do not describe every reported run.

B.3 SUPPLEMENTARY GUIDED-SEARCH INTERPRETER

The supplementary experiment in Appendix C.3 implements an expert as an operator set, a sample count between one and five, and an instruction. Execution follows an optional plan, CoT candidate generation, optional answer selection, and optional revision. Either persona flag adds two candidates; listing both does not add another branch. With multiple candidates and an ensemble or persona flag, a separate call selects an answer. Without such a flag, the first surviving candidate is returned. Empty outputs and scoring failures receive zero; truncated outputs remain scored.

For cluster j, let V_j contain n_j search-validation queries and H_j the remaining construction members. The local validation cap is 60, with at least 20 queries reserved locally when possible. The profile ρ_j averages standardized embeddings on V_j. Search starts from the highest-ranked runnable probe. Its bottom-three rule excludes an operator only if all probes mapped to it are low-ranked, exempts the seed's operator, and retains a nonempty fallback.

Each proposal receives failure texts, mean raw probe scores for failed and solved queries, whole-cluster probe averages, the current specification, and prior attempts. The optimizer is prompted to edit the sample count, operator additions, operator removals, or instruction. Recognized edit types reset fields outside the selected axis; unknown types receive vocabulary and prior filtering without that axis constraint. Invalid or repeated specifications are skipped. This interpreter evaluates an initial workflow and at most two edits, subject to configured cost and time guards, with one execution per candidate and search-validation query. A candidate replaces the incumbent only when

$$\widehat{U}_{V_j}(w') > \widehat{U}_{V_j}(w) + \frac{2}{n_j}, \qquad \widehat{U}_{V_j}(w) = \frac{1}{n_j} \sum_{q \in V_j} \text{score}(w, q). \tag{20}$$

The margin is an implementation rule, not a confidence bound. Whole-cluster feedback includes H_j, so that remainder is not an independent selection holdout; the outer test pool is separate. This fixed interpreter is distinct from the tree-search procedure used for the MATH ablation.

C EXPERIMENTAL PROTOCOL AND SUPPORTING RESULTS

The MixBench-H test pool contains 100 AIME, 100 GPQA, 50 CodeContests, and 50 LiveCodeBench queries. Removing six duplicate CodeContests records from the 600-query development pool leaves 594 queries, split into 414 construction, 90 expert-selection, and 90 calibration queries.

Table 6: MixBench-H task score (%) averaged over two expert executions. The aggregate weights each test query equally.

Configuration	AIME	GPQA	CodeContests	LiveCodeBench	Overall
RA-MoWE (predicted)	29.50	35.50	42.58	84.83	42.90
RA-MoWE (measured)	31.00	37.00	43.33	77.50	42.81

Table 7: Observed SWE-bench probe complementarity. Entries count resolved test issues; the fixed probe and union are outcome-selected diagnostics.

Test subset	n	Observed best fixed probe	Fixed	Union of 12
All repositories	200	Role team	42	66
Django	92	CoT	26	34
Sympy	30	Tree-of-thought	6	9

The reported RA-MoWE configurations use four clusters, construction seed 13, three probe repetitions, and two expert executions per test query. All eight selected experts retain probe workflows. Table 6 gives the source-level scores.

AIME uses a mathematical answer checker, GPQA uses answer-option correctness, and coding scores average the fraction of available public tests passed. The updated scorer supports stdin/stdout and functional interfaces; the baseline scorer skips unsupported functional interfaces. Missing tests, unsuccessful executions, and empty outputs receive zero. Measured embeddings use scored probe executions on the evaluated query; predicted embeddings use query text. These settings complement the comparison protocol in Section 3.1.

Within-repository complementarity. The SWE-bench evaluation uses a repository-stratified 300/200 development/test split and twelve policies in the fixed Agentless-lite localize–repair executor. RA-MoWE (predicted) assigns the direct policy throughout, resolving 40 test issues; measured assignment resolves 49. Table 7 records complementary outcomes within the same repository. Within Django, direct generation resolves five issues missed by retrieval, and retrieval resolves four missed by direct generation. The fixed probe and union are selected on the same single-execution test outcomes, including unresolved failures.

C.1 Baseline adaptations

All adaptations use GPT-4o-mini, with failed queries retained as zero scores. IO uses direct generation; on SWE-bench it uses the direct policy in the fixed Agentless-lite pipeline. AFlow (Zhang et al., 2025) searches Custom and ScEnsemble workflows on MixBench-H using five optimization rounds after initialization, parent sample size four, and one validation execution per query. The SWE adaptation searches policy text on 150 repository-stratified training instances for five rounds.

ADAS (Hu et al., 2025) retains archive-conditioned proposal, reflection/debugging, and bootstrap fitness selection. It searches agent code for mixed tasks and policy text for SWE, where five generations begin from direct, chain-of-thought, and ReAct-policy seeds. ScoreFlow (Wang et al., 2025) uses SF-0 generation and one in-context preference-optimization step without weight updates. Mixed-task inference generates four candidates without test-score feedback; SWE compares twelve initial policies and one refinement on the common 150-instance training subset.

FlowReasoner (Gao et al., 2025) uses a search-only adaptation without meta-reasoner training. On MixBench-H it searches separately for each test query using that query's scorer feedback, then executes the selected workflow three times. It completes 297 of 300 queries; the other three receive zero in the reported 35.65%. The SWE adaptation instead selects a single policy using the disjoint 150-instance training subset. On MATH, the reproduced AFlow selects round eight after eleven rounds, ADAS completes 26 generations, and ScoreFlow compares initial and in-context-refined candidates before selecting one fixed workflow on validation data. MATH FlowReasoner uses query-specific score feedback and three final executions, with two failed queries scored zero.

Table 8: Supplementary guided-search results on 300 queries and three executions per configuration. SD is across execution-level means; inference calls exclude offline work. USD totals cover 900 query executions.

Configuration	Task score (%)	SD	Calls/query	Total USD
IO	27.63	0.31	1.000	.2701
Single chain-of-thought	32.95	1.80	1.000	.4702
Predicted-embedding probe selection	29.55	1.33	2.924	1.0924
RA-MoWE (predicted)	33.58	0.97	2.113	.9919

C.2 MATH ABLATION PROTOCOL

The full predicted configuration and the controls without routing, without guidance, without either, and without search solve 300, 304, 288, 292, and 284 of 486 problems, respectively. Each has one test execution per problem. Table 4 retains the supplied 10,000-draw query-bootstrap intervals; differences use unrounded accuracy fractions. The measured-embedding accuracy of 62.1% was supplied separately and has no corresponding ablation interval in that record.

The paired full-minus-unguided interval is $[-1.23, 6.17]$ percentage points, and full-minus-probe-only is $[-0.21, 6.79]$. Thus the point estimates do not establish significant gains from these components. The archived searched arms were affected by a sandbox timeout defect with different tool and self-test exposure across policies; the supplied record reports a corrected rerun in progress. The ablation describes these recorded executions.

C.3 SUPPLEMENTARY GUIDED-SEARCH EXPERIMENT

The supplementary experiment supporting the abstract uses the same 300 MixBench-H test queries and three executions per configuration. Figure 1 averages train-standardized probe scores over 60, 60, 22, 28, 60, and 24 search-validation members of clusters C0–C5, then centers each displayed row. Parentheses show full training-cluster sizes. Each search evaluates an initial expert and at most two edits; only C3 accepts edits, producing planning followed by four sampled answers and selection. The remaining experts retain single chain-of-thought, refinement, or sampling-and-selection structures.

The executor uses a fixed plan–candidate–selection–refinement sequence, with runnable strategy substitutes shown in Figure 1. Search-validation outcomes guide both edits and their acceptance. A training-only Hungarian alignment maps the construction and assignment clusterings, agreeing on 517 of 600 training queries. All 300 test queries remain separate from construction. Experts and the single-call chain-of-thought control use a 3,000-token cap; IO and probe selection use a 4,000-token cap. The comparison therefore evaluates these complete configurations.

RA-MoWE (predicted) exceeds predicted-embedding probe selection by 4.04 percentage points (95% CI $[0.42, 7.73]$), using 27.7% fewer inference calls. Its difference from single-call chain-of-thought is 0.63 points ($[-1.58, 2.89]$). These exploratory intervals average each query over three executions and use the stored 10,000-draw unstratified query bootstrap.

C.4 COST ACCOUNTING

MixBench-H offline costs include development-probe measurement, unique construction and selection outcomes requested by saved search archives, calibration outcomes, and successful optimizer calls for seed 13. The predicted configuration uses 119,789 calls and 150,283,614 tokens (USD 58.54824); the measured configuration uses 94,538 calls and 117,269,118 tokens (USD 45.75810). Each is charged its shared preparation as a standalone configuration, so their totals must not be added as actual expenditure. Local embedding-encoder fitting, harness compute, and discarded failed attempts are excluded.

Predicted inference averages 1.977 calls and 2,038.47 tokens (USD 0.000882354) per query. Measured assignment adds 93.023 probe calls and 114,103.02 probe tokens to 1.398 expert calls and 1,566.78 expert tokens, totaling USD 0.045744842. Dollar equivalents use the configured

USD 0.15/0.60 per million input/output tokens. The supplementary guided search records 2,544 offline calls and USD 1.2038, excluding probe collection and encoder fitting; its predicted experts use 3,017.84 tokens per query, versus 3,903.03 for probe selection.

D CLUSTER DISTRIBUTION ANALYSIS

Figure 3 uses the same 600 training queries for every partition: 200 AIME mathematics queries, 200 GPQA science queries, and 200 programming queries split equally between CodeContests and LiveCodeBench. Natural domains define three groups; measured workflow-affinity embeddings, BGE, and Qwen3 each use their saved six-cluster partition. Each bar shows the fraction of its members from each domain; parentheses give group sizes. Cluster indices are local to their partition.

The figure compares membership distributions rather than predictive accuracy. Its workflow-affinity clusters use training probe outcomes, and its partition differs from the supplementary guided-search partition in Figure 1. The plotting script verifies source hashes and query identifiers against the saved evidence; every panel contains exactly 600 queries with the same domain totals.