UNDERSTANDING AND SELECTING REASONING STRATEGIES IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study how different reasoning paradigms affect language model performance and when each is useful. Comparing direct answering, chain-of-thought, planning, reflection, and tool use reveals that strategy rankings vary across models and tasks, while multiple strategies can solve the same question at different computational costs. These observations motivate select-then-solve (STS), which learns strategy values from execution outcomes and token costs, then chooses a strategy before solving a new question. STS retains both successful alternatives and failed attempts as supervision. On the original in-distribution split, its cost-aware variant improves score by 2.57 percentage points over a training-selected fixed strategy while using 7.2% fewer solver tokens. Comparisons with dataset lookup and held-out task families indicate that task-family priors account for much of the observed benefit. Controlled experiments reveal instruction–tool interactions and compare repeated sampling within and across configurations. Together, our findings inform when and how to select reasoning strategies.

1 INTRODUCTION

An LLM-based agent can answer directly, reason through intermediate steps, invoke tools, plan, or revise an attempt. These strategies change both the chance of success and the computation used. Choosing among them before execution is therefore a practical inference decision. Prior work studies method selection, reasoning-structure composition, joint model–strategy routing, and confidence-aware selection (Zhao et al., 2023; Zhou et al., 2024; Pan et al., 2026; Qi et al., 2025). We ask how recorded outcomes across strategies can inform that decision.

Our starting point is PARADIGM, our project's archive of reasoning trajectories across models and tasks. Examining these trajectories reveals model- and task-dependent preferences: for Qwen3-30B, ReCode outperforms CoT on HumanEval, whereas CoT performs better on MATH500. Multiple strategies often solve the same question, yet their recorded token costs can differ substantially. Alongside questions where every strategy fails, these successful alternatives suggest learning a value for each candidate strategy instead of reducing each question to a single winning label.

We develop this idea as *select-then-solve* (STS; Figure 1). Offline, strategy-specific estimators learn from question features and the full set of observed outcomes. Value-Ridge predicts score; Cost-Ridge incorporates token cost into the target. Online, the router predicts these values from question text and executes the highest-valued strategy. The selected strategy can itself make several model calls. Linear estimators provide an inspectable implementation of this decision rule.

We evaluate STS on mathematics, code generation, and multiple-choice questions with complete strategy records. On the original split, Cost-Ridge improves over a training-selected fixed strategy by 2.57 points with 7.2% fewer solver tokens. Dataset lookup, repeated partitions, and whole-dataset holdouts assess the contribution of question features and the stability of these gains (Lee, 2026; Shihab et al., 2026; Yang et al., 2026). A factorial study then examines instruction–tool interactions and repeated sampling under explicit resource caps.

The paper follows this progression: characterize the strategy archive, formulate STS with full-outcome score and utility supervision, and evaluate its benefits and scope. We provide the versioned records and executable offline analyses alongside the selectors, so that routing decisions can be traced to their supervision, task information, and execution conditions.

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.

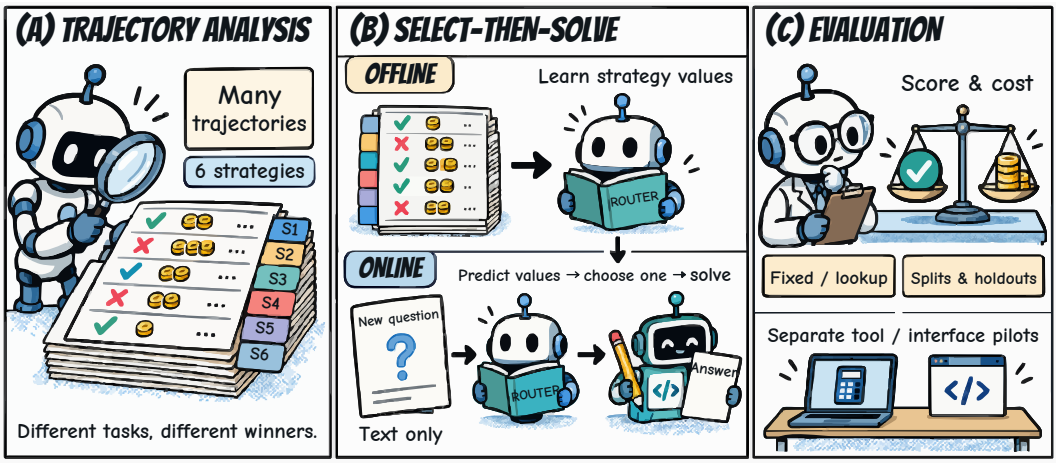


Figure 1: From trajectory analysis to select-then-solve. (a) Model–task variation, alternative outcomes, and resource use motivate strategy-value learning. (b) STS fits values offline and selects one strategy from question text before execution. (c) Routing comparisons assess score and cost; controlled experiments examine execution choices. These experiments are separate from the archival routing supervision.

2 Related Work

Reasoning and inference computation. Chain-of-thought and zero-shot reasoning prompts elicit intermediate solution steps (Wei et al., 2022; Kojima et al., 2022); Plan-and-Solve separates planning from execution (Wang et al., 2023a). ReAct interleaves reasoning with environment actions (Yao et al., 2023b), while Self-Refine and Reflexion use feedback to revise attempts (Madaan et al., 2023; Shinn et al., 2023). PAL and Program of Thoughts delegate computation to executable programs (Gao et al., 2023; Chen et al., 2023b); ReCode expresses plans and actions through recursive code expansion (Yu et al., 2025). Tree of Thoughts explores branching solution paths (Yao et al., 2023a). These mechanisms act on different parts of the inference policy. Our six historical implementations instantiate these control-flow families with the local prompts and interfaces specified in Appendix A. Self-consistency and test-time scaling also show why additional attempts and computation are substantive baselines (Wang et al., 2023b; Snell et al., 2024).

Selecting models and strategies. FrugalGPT studies cost-sensitive model cascades (Chen et al., 2023a); benchmark-based routing and RouteLLM learn model-selection policies (Shnitzer et al., 2023; Ong et al., 2024). Within a model, Zhao et al. (2023) select between chain-of-thought and program-aided solving, while SELF-DISCOVER composes reasoning modules (Zhou et al., 2024). Route to Reason jointly selects a model and reasoning strategy (Pan et al., 2026); PRISM predicts strategy suitability and uses confidence to choose an execution pattern (Qi et al., 2025). Our study evaluates the routing information available to linear classifiers and regressors through task-prior baselines and dataset holdouts.

Evaluating routing gains. LLMRouterBench finds that several routers are difficult to distinguish under unified evaluation (Li et al., 2026). Shihab et al. (2026) separate outcome-oracle opportunity from realizable gains and develop selection-valid inference. Lee (2026) analyze task-type structure and repeated-run variability, explicitly without a holdout. Yang et al. (2026) study prediction of protocol-specific value, including source-held-out evaluation. Building on these controls, we combine an audited six-strategy archive with repeated-split and held-out-dataset evaluations, an instruction–calculator factorial experiment, and a paired code-extraction pilot.

Evaluation and task interfaces. HumanEval, MATH, and MMLU require distinct verification procedures (Chen et al., 2021; Hendrycks et al., 2021b;a). HotpotQA and Natural Questions also distinguish answer evaluation from the retrieval setting (Yang et al., 2018; Kwiatkowski et al., 2019). AgentBench, WebArena, and MINT evaluate interaction with environments, tools, or feedback (Liu

et al., 2024; Zhou et al., 2023; Wang et al., 2024); GAIA and τ-bench target assistant capabilities and stateful tool-agent-user interaction (Mialon et al., 2023; Yao et al., 2024). Our supplementary archive scores static action lists and exported Humanity's Last Exam answer surfaces (Phan et al., 2025). These adaptations measure agreement with exported references rather than official interactive or judge-based success; the primary routing evaluation uses mathematics, code, and multiple-choice tasks.

3 PATTERNS IN STRATEGY EXECUTION

PARADIGM contains 17,950 of 18,264 intended outputs for 761 questions. Six strategies—Direct, CoT, ReAct, Plan-Execute, Reflection, and ReCode—span single responses, tool use, planning, revision, and code execution. Historical model identifiers are GPT-5, Gemini-3-Flash-Preview, Qwen3-Max, and Qwen3-30B-A3B (OpenAI, 2025; Doshi, 2025; Yang et al., 2025). Prompts, tools, and stopping rules differ across these complete inference policies (Appendix A).

The primary subset comprises 100 HumanEval, 100 MATH500, 60 AIME, and 100 MMLU questions, with records for all 8,640 model–strategy assignments. We retain 111 top-level exceptions and 15 other empty answers, scored zero. The remaining 401 questions belong to five supplementary collections in Table 1 and the static τ adaptation in Appendix B.1. Table 1 also incorporates supplementary Gemini rerun scores; routing analyses retain the frozen archive. Coverage and sampling provenance appear in Appendix C.

Versioned scoring uses mathematical equivalence, explicit multiple-choice labels, and program execution. HumanEval re-execution under Python 3.13.9 matches all archived labels. Score changes, parser checks, and incomplete cost telemetry are documented in Appendices C and C.1.

Strategy preference depends on both model and task. The strategy matrix reveals different preferences across models and tasks (Table 1; primary-set detail in Appendix B). On Qwen3-30B, ReCode scores 96% on HumanEval but 44% on MATH500; CoT scores 64% and 90%, respectively. GPT-5 also favors ReCode over CoT on the archived code tasks (95% versus 70%), whereas Gemini's CoT and Reflection both reach 99%. Within each primary model–dataset column, all six strategies use the same questions. These paired aggregate differences motivate a task-family prior. Section 5 tests how much question features add to this prior on held-out questions.

Successful strategies overlap, but their costs differ. Figure 2 examines outcomes within each question, complementing the aggregate rankings. Across the four models, 75.8–97.5% of primary questions have at least two successful strategies. Thus, a single winner label frequently discards a working alternative. Success also leaves substantial variation in cost: among questions with multiple successes, the median ratio of the most to the least expensive successful strategy ranges from 4.7 to 12.2 across models. These are recorded solver tokens, including input and output across calls. The overlap and cost spread summarize one archived run per strategy. They motivate learning separate strategy values, with predictive benefit assessed on held-out outcomes.

From outcome patterns to a selection rule. Three observations shape STS. Model–task variation motivates conditioning the choice on the question and backend. Overlapping successes motivate retaining a separate outcome for every strategy, including questions where all six fail. Cost differences motivate learning utility as well as score. The resulting router predicts strategy values before execution; the next sections define this rule and test which observed differences are predictable from question text.

4 SELECT-THEN-SOLVE FROM FULL OUTCOMES

4.1 THE PRE-EXECUTION DECISION

For a fixed model m, question x from dataset d, and strategy $p \in \mathcal{P}$, let $Y_{mp}(x, \omega) \in \{0, 1\}$ denote the score of the resulting answer and $C_{mp}(x, \omega)$ its token cost. The random variable ω includes sampling and service variability. A question-only router $\pi(x)$ chooses a strategy before observing

Table 1: Strategy scores (%) across primary and supplementary collections, including supplementary Gemini reruns (Appendix B). Bold values mark the highest nonzero score within each model–dataset column. Held-out routing performance is evaluated separately.

	Primary benchmark				Supplementary collections				
Strategy	**HEval**	**M500**	**AIME**	**MMLU**	**HPQA**	**NQ**	**GAIA**	**HLE**	**SEAL**
GPT-5									
Direct	85.0	**100.0**	90.0	88.0	**47.0**	15.0	18.0	10.0	16.0
CoT	70.0	**100.0**	88.3	**89.0**	38.0	8.0	22.0	10.0	34.0
ReAct	86.0	96.0	90.0	**89.0**	28.0	3.0	**42.0**	8.0	28.0
Plan-Execute	35.0	97.0	95.0	81.0	24.0	0.0	30.0	8.0	**48.0**
Reflection	85.0	90.0	**96.7**	85.0	28.0	1.0	22.0	12.0	46.0
ReCode	**95.0**	99.0	88.3	83.0	**47.0**	**17.0**	30.0	**16.0**	10.0
Gemini-3-Flash									
Direct	73.0	97.0	**96.7**	**90.0**	47.0	18.0	**22.0**	8.0	22.0
CoT	**99.0**	97.0	**96.7**	88.0	27.0	6.0	14.0	**12.0**	**62.0**
ReAct	98.0	85.0	91.7	86.0	0.0	3.0	6.0	2.0	**62.0**
Plan-Execute	98.0	87.0	91.7	87.0	2.0	3.0	10.0	6.0	48.0
Reflection	**99.0**	84.0	93.3	86.0	1.0	4.0	6.0	6.0	34.0
ReCode	81.0	**99.0**	95.0	89.0	**50.0**	**22.0**	6.0	4.0	40.0
Qwen3-Max									
Direct	85.0	**100.0**	81.7	**85.0**	24.0	4.0	8.0	8.0	10.0
CoT	94.0	**100.0**	83.3	83.0	27.0	4.0	10.0	**14.0**	**40.0**
ReAct	**96.0**	95.0	**88.3**	**85.0**	6.0	0.0	14.0	12.0	30.0
Plan-Execute	71.0	92.0	75.0	79.0	17.0	1.0	12.0	6.0	38.0
Reflection	92.0	97.0	86.7	10.0	0.0	0.0	**16.0**	8.0	30.0
ReCode	**96.0**	48.0	1.7	80.0	**35.0**	**17.0**	4.0	12.0	2.0
Qwen3-30B									
Direct	18.0	79.0	16.7	70.0	**13.0**	**5.0**	**6.0**	**4.0**	6.0
CoT	64.0	**90.0**	21.7	**75.0**	7.0	2.0	4.0	2.0	**28.0**
ReAct	52.0	70.0	13.3	67.0	0.0	0.0	2.0	0.0	12.0
Plan-Execute	7.0	61.0	**23.3**	70.0	**13.0**	2.0	**6.0**	**4.0**	20.0
Reflection	50.0	76.0	10.0	69.0	1.0	0.0	0.0	2.0	20.0
ReCode	**96.0**	44.0	6.7	69.0	11.0	**5.0**	2.0	**4.0**	0.0

HEval: HumanEval; M500: MATH500; HPQA: HotpotQA; NQ: Natural Questions. Metric definitions and collection details: Appendix B.

its answer. The target is an expected utility such as

$$J(\pi) = \mathbb{E}_{d,x,\omega}\left[Y_{m,\pi(x)}(x,\omega) - \lambda \widetilde{C}_{m,\pi(x)}(x,\omega)\right], \tag{1}$$

where our empirical cost is $\widetilde{C} = \log(1 + C)/s$, with s fitted on training outputs (Section 4.2). The logarithm expresses diminishing marginal token penalties; raw monetary cost would define a different objective. We macro-average over datasets and then the fixed model roster, and report raw solver tokens separately. A router that also observes d has a different information set and is labeled accordingly.

4.2 Learning strategy values

We encode all 761 questions with full-question Qwen3-Embedding-8B features (Zhang et al., 2025), without client-side text truncation. **Value-Ridge** uses every observed binary outcome as a target $u_{ip} = Y_{mp}(x_i)$, preserving both alternative successful strategies and all-failure questions. **Cost-Ridge** changes the target to $u_{ip} = Y_{mp}(x_i) - 0.05 \log(1 + C_{mp}(x_i))/s$, with s the 95th percentile of $\log(1 + C)$ pooled over training questions and all six strategies. For unit-normalized features

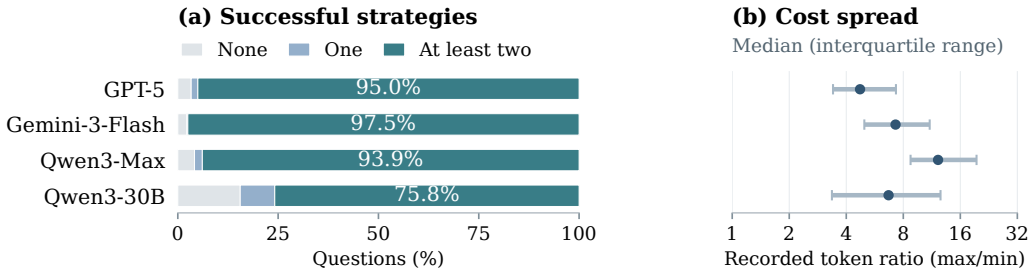


Figure 2: Within-question outcomes in the primary archive. (a) Number of successful strategies, with equal weight per question within each model. (b) Ratio of maximum to minimum recorded solver tokens among successful strategies on the same question, conditional on at least two successes. Dots and bars show the median and interquartile range on a logarithmic axis. Each strategy contributes one archived run per question.

$z(x)$, each strategy has a linear value model:

$$\hat{f}_p = \arg \min_{f_p(z)=w_p^\top z+b_p} \sum_{i\in\mathcal{T}} a_i \big(f_p(z(x_i)) - u_{ip}\big)^2 + \alpha\|w_p\|_2^2,$$
$$\pi(x) = \arg\max_{p\in\mathcal{P}} \hat{f}_p(z(x)). \tag{2}$$

$\mathcal{T}$ indexes training questions; a_i is normalized inverse dataset size (mean one). At inference time, each router uses question text alone and executes only its selected strategy. Ridge α is fixed at 1 for all fits.

Oracle scores and identifiable gains. For evaluation datasets $\mathcal{D}$, with n_d questions in each, the retrospective ceiling for model m is

$$\widehat{O}_m = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \frac{1}{n_d} \sum_{i:d_i=d} \max_{p\in\mathcal{P}} Y_{mp}(x_i, \omega_{ip}). \tag{3}$$

We report the model mean; each $\widehat{O}_m$ differs from $\mathbb{E}_x \max_p \mathbb{E}_\omega Y_{mp}(x,\omega)$. As a simple illustration, if all P strategies have identical independent Bernoulli success probability q on every question, then the observed union succeeds with probability $1-(1-q)^P$, although selecting a single strategy cannot improve on q. Correlated outputs change this selection effect; no independence assumption is imposed on the empirical analyses. Repeated calls are needed to assess whether apparent task–strategy preferences persist beyond a single observation.

5 Evaluating Strategy Selection

5.1 Comparisons and Evaluation Protocol

The historical task split has 532 training and 229 test questions; its primary-subset intersection has 252 and 108. A question has the same split under every model. For leave-one-dataset-out (LODO) evaluation, each target dataset is removed completely from fitting; all its questions are evaluated. Policies are retrained independently for each model, split, and cohort. All-failure questions stay in the evaluation denominator.

Fixed chooses the strategy with the highest training macro score. **DS-Lookup** chooses by training score within each dataset; it falls back to Fixed for an unseen dataset. **Label-LR** predicts the cheapest successful strategy with logistic regression on embeddings, omitting all-failure questions only from this classifier's training labels. This baseline compares value learning with winner-label supervision. **Value+DS** adds dataset one-hot features to Value-Ridge. Logistic-regression C is fixed at 1; Label-LR standardization uses its training subset.

All policies share evaluation questions. Intervals use 3,000 paired bootstrap resamples of question identities within each dataset, keeping all models and policies together. They condition on the fixed

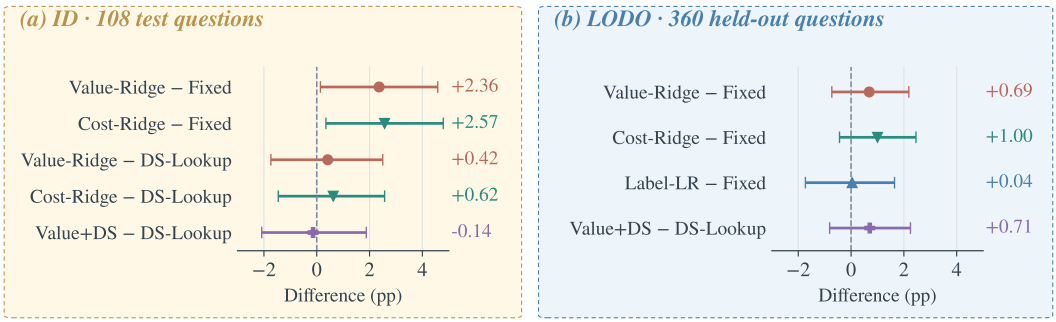


Figure 3: Paired routing gains and their scope. Value-Ridge and Cost-Ridge improve over Fixed on the original ID split; changing the comparator to DS-Lookup reduces these gains. LODO holds out each dataset, where lookup falls back to Fixed. Intervals preserve task dependence across policies and models; they are exploratory and unadjusted for multiple comparisons.

model roster and fitted policies. Token counts sum reported solver input and output; zero stored tokens in exception records denote unavailable usage. Embedding usage is reported separately (Appendix C.1). Oracle selected-answer cost differs from the cost of acquiring all candidate answers.

5.2 Routing performance and cost

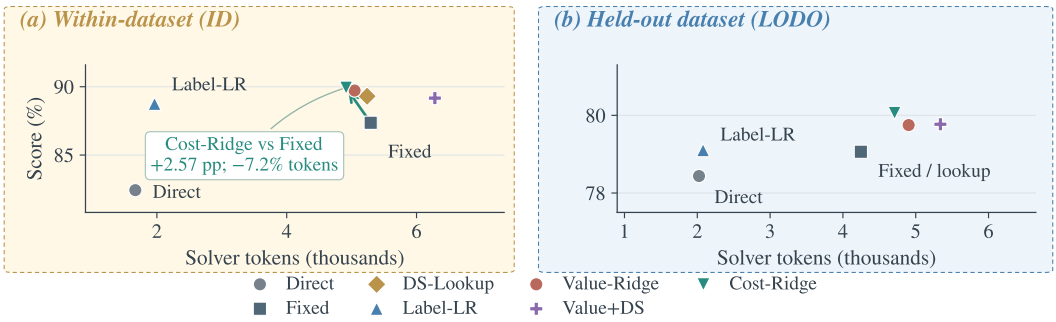
Original-split accuracy and solver cost. The complete routing table (Appendix D, Table 12) reports an ID score of 82.4 for Direct and 87.4 for the training-selected Fixed policy. Value-Ridge reaches 89.7, gaining 2.36 percentage points over Fixed (95% paired interval: [0.14, 4.58]). Cost-Ridge reaches 89.9, gaining 2.57 points [0.35, 4.79] while reducing mean solver tokens by 7.2% (4,915 versus 5,294). Both choose from question text before observing an answer. This operating point combines higher recorded score with lower solver usage on the original ID split.

Dataset lookup as a task-family reference. DS-Lookup reaches 89.3. Relative to this stronger prior, Value-Ridge gains +0.42 points (interval: [-1.74, +2.50]); Cost-Ridge gains +0.62 [-1.46, +2.57]. Adding question embeddings to dataset features gives Value+DS a difference of -0.14 points from lookup, with an interval of [-2.08, 1.88]. The paired intervals include zero for all three comparisons (Figure 3). These estimates characterize the fitted policies over the tested features.

Cross-dataset evaluation. In LODO, lookup falls back to Fixed. Value-Ridge achieves 79.8 versus 79.1, a 0.69-point gain [-0.73, 2.19]; Label-LR gains 0.04 [-1.73, 1.65]. Cost-Ridge's +1.00 points [-0.44, +2.46] accompanies 10.9% more recorded tokens than Fixed. The paired score intervals include zero. Value-Ridge's positive macro difference is driven by Qwen3-30B; the other three models have negative point differences, partly entangled with operational failures. ID and LODO use different questions and training splits; comparisons are against each regime's own baseline.

Cost and predictability favor different policies. Label-LR uses 1,964 recorded tokens at score 88.8, versus 5,239 and 89.3 for lookup. Relative to Value-Ridge, Cost-Ridge gains 0.21 points and saves 2.6% tokens. Training-only penalty selection and same-utility Fixed/lookup sweeps provide additional operating points (Appendix D.1). Tokens exclude embedding inference, tools, latency, and unrecorded usage after exceptions. Embeddings were generated once and reused across models and policies; Appendix C.1 reports their usage.

Execution profiles across strategies. Qwen3-Max Reflection's 10% MMLU score includes 88 connection failures; only 12 answers returned, ten scored correct. ReCode tool returns contain `SyntaxError` in all 60 AIME and 100 MATH500 runs for Qwen3-Max. Some final answers remain correct, distinguishing intermediate execution errors from final-answer outcomes. No fitted ID router selects a top-level exception, but LODO Value-Ridge and Cost-Ridge select nine and six



Macro means of input + output solver tokens; embedding and supervision costs are reported separately.

Figure 4: Score and observed solver cost when executing one selected strategy. The arrow highlights Cost-Ridge's ID improvement over Fixed: 2.57 points with 7.2% fewer solver tokens. Comparisons are within panels because ID and LODO use different questions. Score uncertainty appears in Figure 3; penalty sweeps appear in Appendix D.1.

of the connection failures. These statuses contribute to both training targets and evaluation scores. Appendix C.1 reports frequencies and sensitivity bounds.

Oracle coverage and single-run variability. The observed six-strategy oracle scores 95.6 in ID and 92.6 in LODO. The difference from the fitted routers combines predictable task variation, prediction error, limited training data, and single-run outcome variability. Separating these contributions requires repeated outcomes beyond the single archived output per strategy and question. Section 6 adds repeated observations under explicit resource caps. Appendix D reports model-specific results, all paired intervals, and the broader ten-collection sensitivity analysis. The latter uses the collection-specific QA and adapted-task metrics defined in the appendix.

Repeated splits and alternative estimators. Across ten additional stratified splits, Value-Ridge and Cost-Ridge average +1.04 and +1.08 points over Fixed, positive in seven splits, but -1.44 and -1.41 versus lookup. Cost-Ridge averages 18.2% more tokens than Fixed. Keeping all-failure training rows does not improve on an otherwise matched Ridge ablation. Full-outcome logistic, a random forest and predicted-dataset lookup strengthen the controls (Appendix D.1). These repeated-data diagnostics quantify sensitivity to the training and test partition.

Supervision budgets and cross-model transfer. The primary training matrix uses 6,048 outputs and 30.7 million recorded solver tokens. Across 12 source–target model transfers, Value-Ridge averages 82.6 versus 87.4 for target-calibrated Fixed, which has additional target supervision. Reduced-label fits and transfer details appear in Appendix D.2.

6 Instructions, Tools, and Repeated Sampling

The archival strategies combine several inference choices. We use a factorial study to vary instructions and tool access under explicit resource caps, then examine how repeating or mixing these configurations changes answer coverage.

Design. We conduct a 2×2 factorial pilot on 24 MATH500 and eight AIME questions selected by a seeded hash of task identity. The interventions are a brief versus structured reasoning instruction and calculator access versus no tool. Each condition is evaluated with GPT-5 and Qwen3-30B-A3B-Instruct-2507 at cumulative output-token caps of 1,024 and 4,096, with two repetitions: 1,024 assigned trajectories. All conditions share the same final-answer format and maximum of three model calls. Tool-enabled conditions share one calculator and a maximum of four tool executions. Tokens reported for internal reasoning, when present, count against the output cap. Actual input tokens, output tokens, tool calls, truncations, and operational failures are recorded separately. Appendix E specifies the exact protocol and network-recovery handling.

Table 2: Factorial pilot: cell scores (%) and marginal effects (percentage points; paired 95% intervals). Each cell contains 32 questions and two repeats. Cap is the cumulative output-token allowance. All assigned runs remain in the denominator; unavailable answers score zero. Intervals describe recorded success, with operational-missingness bounds reported separately.

	No calculator		Calculator		Marginal effect [95% CI]	
Cap	Brief	Structured	Brief	Structured	Instruction	Calculator
GPT-5						
1,024	65.6	76.6	34.4	76.6	+26.6 [+17.2, +36.7]	-15.6 [-23.4, -7.8]
4,096	68.8	81.2	51.6	76.6	+18.8 [+9.4, +29.7]	-10.9 [-19.6, -1.6]
Qwen3-30B-2507						
1,024	57.8	64.1	62.5	64.1	+3.9 [-3.1, +12.5]	+2.3 [-5.5, +9.4]
4,096	64.1	73.4	67.2	75.0	+8.6 [+0.8, +18.0]	+2.3 [-4.7, +10.9]

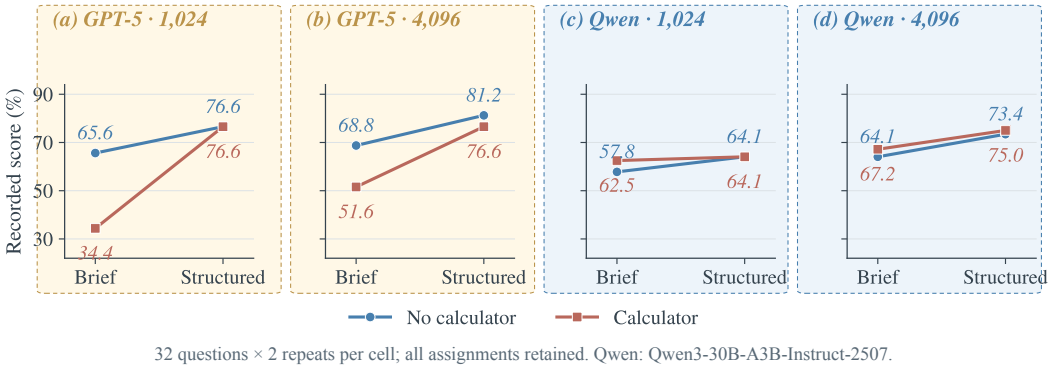


Figure 5: Factorial cell means. The difference between GPT-5's brief and structured tool-enabled conditions shows an instruction–calculator interaction at the smaller cap. Scores include all assigned runs. Caps bound output tokens; realized token use varies.

Estimands. For a fixed model and cap, the instruction effect is the score difference between structured and brief conditions, averaging the two tool settings. The calculator effect averages the tool-access difference over the two instructions; the interaction is the difference between instruction effects with and without tools. Pilot means weight questions equally, retaining the 24:8 MATH500–AIME mixture. The contrasts measure API-level interventions under the specified prompt, tool, and resource settings. Bootstrap intervals retain the dataset mixture, repeats, and paired conditions; they are exploratory and unadjusted for multiple comparisons.

Instructions and tool access interact. Structured instructions improve GPT-5's recorded score at both caps (Table 2). At 1,024 tokens, brief prompting scores 65.6 without the calculator and 34.4 with it; structured prompting scores 76.6 in both settings (Figure 5). The interaction is +31.25 points (95% interval: [+14.1, +48.4]). The contrast links the calculator's recorded effect to the instruction format in this setting. Tool access provides the schema and an option to call the calculator.

Recorded success and completion. All 1,024 assigned trajectories remain in the scores: 52 operational failures, 81 truncations, one empty answer, and 890 completions. Qwen has 48 operational failures, 43 at the larger cap. Allowing each unavailable final answer to be incorrect or correct gives a Qwen instruction-effect range of [-2.3, +31.2] points at 4,096 tokens. Table 2 estimates recorded pipeline success; the separate missingness bounds include both positive and negative instruction effects. GPT-5's calculator-access effect remains negative at both caps under these bounds (Appendix E).

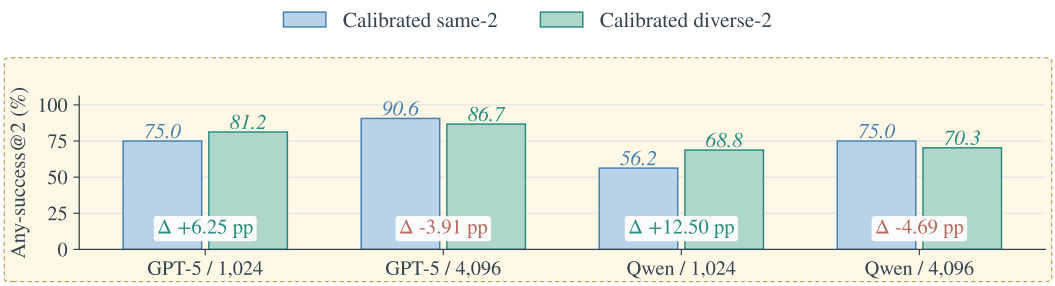


Figure 6: Calibration-matched repeated sampling. Both policies use the other 31 questions and optimize any-success@2. Mixing conditions improves the smaller-cap point estimates but falls below calibrated repetition at the larger cap. Estimates use overlapping calibration folds and two repeats per condition and are reported descriptively.

Calibration-matched repeated sampling. Mixing two conditions exceeds repeating a uniformly chosen condition in all four model–cap settings. To account for differences in average condition quality, we give both alternatives the same other-31-question calibration set and select each for the same any-success@2 objective. One alternative repeats a selected condition; the other selects a pair of distinct conditions. On each held-out question, the former uses its two recorded repeats and the latter averages over all four cross-condition repetition pairings. Any-success@2 measures verifier-assisted answer coverage at a fixed attempt count. Realized tokens vary, and deployment additionally requires choosing an answer without ground-truth labels.

Figure 6 shows that calibrated diversity gains +6.25 points for GPT-5 and +12.50 for Qwen at 1,024 tokens, but loses 3.91 and 4.69 points, respectively, at 4,096. The distinct-pair constraint changes the available quality mix when one condition is stronger; calibration and tie-breaking also affect the comparison. This exploratory analysis was added after inspecting the initial repeat results and reuses the pilot's questions and two repeats per condition. Appendices E.1–E.2 report selection rules, completion-conditional scores, realized tokens and tie-order sensitivity, including sign changes across selection rules.

Implications for the strategy menu. The factorial results help interpret the strategy values learned by STS. At the smaller cap, structured instructions improve GPT-5 more when the calculator is available than when it is absent. A tool's value therefore depends on the instruction accompanying it. For a selector, the relevant candidate is the complete instruction–tool configuration evaluated under a stated budget. The repeated-sampling comparison adds a second consideration: a condition with a high individual score and a pair with broad answer coverage solve different selection problems. Their calibration objectives should reflect whether deployment will execute one strategy or collect multiple attempts. Together, these observations motivate measuring candidate configurations jointly and calibrating selection for the intended resource setting. The pilot supplies controlled evidence about these design choices; its outcomes are separate from the archival supervision used to fit STS.

7 Discussion and Limitations

We connect observations of reasoning behavior, pre-execution selection, and controlled instruction–tool experiments. STS turns outcome and cost differences into a question-only decision rule. The paired archive and task-prior controls provide a reusable basis for comparing strategy selectors.

The results concern the tested estimators and implementations. Gains depend on the comparator and partition; solver-token totals exclude embedding and unreported usage, and execution reliability contributes to scores. These findings motivate calibration for the intended backend and task family. Versioned records, split assignments, and executable analyses accompany the separate supplementary PDF; Appendix G discusses further implications.

AI USE DISCLOSURE

AI tools supported language editing, literature search and verification, partial code generation, data processing and analysis, refinement of methods and experiments, and illustrative artwork. Checks included source-record and reference verification and executable tests. The authors take responsibility for the final content.

REFERENCES

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance, 2023a. URL https://arxiv.org/abs/2305.05176.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023b. URL https://arxiv.org/abs/2211.12588.

Tulsee Doshi. Gemini 3 flash: Frontier intelligence built for speed, 2025. URL https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/. Google model release announcement, December 17.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *International Conference on Machine Learning*, pp. 10764–10799, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *International Conference on Learning Representations*, 2021a.

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. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021b. URL https://arxiv.org/abs/2103.03874.

Hugging Face. Math-Verify, 2026. URL https://github.com/huggingface/Math-Verify. Software, version 0.9.0; accessed September 18, 2026.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2205.11916.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Janghoon Lee. Most of the LLM routing gap is task type, 2026. URL https://arxiv.org/abs/2608.23023.

Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, Zhen Wang, and Shuyue Hu. LLMRouterBench: A massive benchmark and unified framework for LLM routing, 2026. URL https://arxiv.org/abs/2601.07206.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.03688.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2023.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

OpenAI. Introducing GPT-5, 2025. URL https://openai.com/index/introducing-gpt-5/. Model release announcement, August 7.

Zhihong Pan, Kai Zhang, Yuze Zhao, and Yupeng Han. Route to reason: Adaptive routing for LLM and reasoning strategy selection. In *Proceedings of the ACM Web Conference 2026*, pp. 5568–5578, 2026. doi: 10.1145/3774904.3792556. URL https://arxiv.org/abs/2505.19435.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, et al. Humanity's last exam, 2025. URL https://arxiv.org/abs/2501.14249.

Python Software Foundation. What's new in python 3.14: Deferred evaluation of annotations, 2025. URL https://docs.python.org/3.14/whatsnew/3.14.html. Python 3.14 documentation; accessed September 21, 2026.

Shihao Qi, Jie Ma, Ziang Yin, Lingling Zhang, Jian Zhang, Jun Liu, Feng Tian, and Tongliang Liu. Plan before solving: Problem-aware strategy routing for mathematical reasoning with LLMs, 2025. URL https://arxiv.org/abs/2509.24377.

Ibne Farabi Shihab, Abu Sa-Adat Mohamed Moon-Im Al Ahsan, and Md Najmus Swaqeeb. Opportunity is not realizability: Selection-valid diagnostics for multi-LLM routing, 2026. URL https://arxiv.org/abs/2608.08265.

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

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets, 2023. URL https://arxiv.org/abs/2309.15789.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2609–2634, 2023a. doi: 10.18653/v1/2023.acl-long.147. URL https://aclanthology.org/2023.acl-long.147/.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.10691.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023b. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Chih-Hsuan Yang, Jingyan Jiang, Cheng-Hau Yang, Vikram Vasudevan, Huihuo Zheng, Venkatram Vishwanath, and Rajeev Thakur. LLMs can predict failure risk, but struggle to predict which collaboration protocol pays off: Cost-aware protocol routing across reasoning tasks, 2026. URL https://arxiv.org/abs/2608.14927.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *Conference on Empirical Methods in Natural Language Processing*, 2018.

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. *Advances in Neural Information Processing Systems*, 2023a. URL https://arxiv.org/abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *International Conference on Learning Representations*, 2023b.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Zhaoyang Yu, Jiayi Zhang, Huixue Su, Yufan Zhao, Yifan Wu, Mingyi Deng, Jinyu Xiang, Yizhang Lin, Lingxiao Tang, Yuyu Luo, Bang Liu, and Chenglin Wu. ReCode: Unify plan and action for universal granularity control, 2025. URL https://arxiv.org/abs/2510.23564.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

James Xu Zhao, Yuxi Xie, Kenji Kawaguchi, Junxian He, and Michael Qizhe Xie. Automatic model selection with large language models for reasoning. *arXiv preprint arXiv:2305.14333*, 2023.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V Le, Ed H Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. Self-discover: Large language models self-compose reasoning structures. *arXiv preprint arXiv:2402.03620*, 2024.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents, 2023. URL https://arxiv.org/abs/2307.13854.