Select-then-Solve: Paradigm Routing as Inference-Time Optimization for LLM Agents

Anonymous Authors[1]

Abstract

When an LLM-based agent improves on a task, is the gain from the model or from the reasoning paradigm wrapped around it? We study this question by comparing six inference-time paradigms, namely Direct, CoT, ReAct, Plan-Execute, Reflection, and ReCode, across four frontier LLMs and ten benchmarks, yielding roughly 18k runs. We find that reasoning structure helps dramatically on some tasks but hurts on others: ReAct improves over Direct by 44pp on GAIA, while CoT degrades performance by 15pp on HumanEval. No single paradigm dominates, and oracle per-task selection beats the best fixed paradigm by 17.1pp on average. Motivated by this complementarity, we propose a *select-then-solve* approach: before answering each task, a lightweight embedding-based router selects the most suitable paradigm. Across four models, the router improves average accuracy from 47.6% to 53.1%, outperforming the best fixed paradigm at 50.3% by 2.8pp and recovering up to 37% of the oracle gap. In contrast, zero-shot self-routing only works for GPT-5 at 67.1% and fails for weaker models, all trailing the learned router. Our results argue that reasoning paradigm selection should be a per-task decision made by a learned router, not a fixed architectural choice.

1. Introduction

Reasoning paradigms have become a central design axis for LLM-based agents. A single large language model can be wrapped with different inference-time strategies: *direct prompting* lets the model answer freely without imposing any reasoning scaffold, relying entirely on the model's own capabilities; *chain-of-thought* explicitly instructs the model to reason step by step before answering (Wei et al., 2022); *ReAct* interleaves reasoning with tool calls such as web search (Yao et al., 2023); *plan-then-execute* decomposes a task into a plan before acting (Wang et al., 2023a); *reflection* generates an initial answer, critiques it, and revises (Shinn et al., 2023; Madaan et al., 2023); and *ReCode* solves problems through recursive code generation and execution (Yu et al., 2026; Chen et al., 2023b). These paradigms differ in how many LLM calls they make, whether they invoke tools, and how they allocate test-time computation (Snell et al., 2024), yet they can all wrap the same base model.

Despite the growing variety of paradigms, controlled comparisons remain scarce. Prior work typically introduces a new paradigm and evaluates it on tasks tailored to its strengths, while changing the model, prompt format, tool stack, and benchmark simultaneously. As a result, the field has accumulated many positive case studies for individual paradigms, but much less evidence about a more basic question: *when does additional reasoning structure actually help, and when does it hurt?*

We address this question by building PARADIGM, a unified evaluation framework in which all six paradigms share the same model interface, evaluation code, and core tools, differing only in how they organize inference. We evaluate GPT-5 (OpenAI, 2025), Gemini-3-Flash (Google DeepMind, 2025), Qwen3-Max, and Qwen3-30B (Qwen Team, 2025) on ten benchmarks covering code generation (Chen et al., 2021), mathematics (Hendrycks et al., 2021b), question answering (Yang et al., 2018; Kwiatkowski et al., 2019), knowledge (Hendrycks et al., 2021a), and tool-use-heavy tasks (Mialon et al., 2024; Yao et al., 2024), yielding roughly 18k completed runs.

Our controlled comparison reveals that reasoning structure helps dramatically on some tasks but hurts on others. ReAct improves over Direct by 44pp on GAIA where web search is essential, while CoT degrades performance by 15pp on HumanEval where step-by-step reasoning disrupts code generation. The best paradigm changes systematically by dataset and by model: an oracle that selects the best paradigm per task outperforms the best fixed paradigm by 17.1pp on average.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

This complementarity raises a natural question: if different tasks benefit from different paradigms, can we select the right paradigm before answering each task? We propose a *select-then-solve* approach: a lightweight router analyzes the incoming task and dispatches it to the most suitable paradigm, including Direct itself when no structure is needed. Across four models, the router improves average accuracy from 47.6% (Direct) to 53.1%, outperforming the best fixed paradigm at 50.3% by 2.8pp and recovering up to 37% of the oracle gap. In contrast, zero-shot self-routing, where the model selects its own paradigm, only works for GPT-5 at 67.1% while weaker models drop below their Direct baselines, revealing that reliable paradigm selection remains a challenging meta-reasoning capability.

We make the following contributions:

◇ A controlled large-scale comparison of six inference-time paradigms across four LLMs and ten benchmarks.

◇ Evidence that reasoning structure is helpful, neutral, or harmful depending on the task, with gains up to 44pp on information-seeking tasks and losses of 15pp on code generation.

◇ A select-then-solve framework with an embedding-based paradigm router that improves average accuracy by 5.5pp over Direct and 2.8pp over the best fixed paradigm.

◇ Evidence that self-routing is unreliable: only GPT-5 benefits while weaker models fail, all trailing the learned router.

2. Related Work

2.1. Inference-Time Reasoning Paradigms

Chain-of-Thought prompting (Wei et al., 2022) established that allocating generation budget to intermediate reasoning can improve performance on complex tasks, with later work exploring zero-shot CoT (Kojima et al., 2022), self-consistency (Wang et al., 2023b), and automatic rationale construction (Zhang et al., 2023). Tool-augmented paradigms such as ReAct (Yao et al., 2023) extend this idea by interleaving reasoning with external actions, while planning-based methods separate decomposition from execution (Wang et al., 2023a; Liu et al., 2023). Self-refinement methods (Madaan et al., 2023; Shinn et al., 2023) spend extra inference-time budget on critique and revision, and code-centric approaches such as PAL, Program-of-Thoughts, and ReCode (Gao et al., 2023; Chen et al., 2023b; Yu et al., 2026) use executable programs as a reasoning substrate. Recent work has also explored reinforcement learning and search-based strategies for inference-time reasoning (Zhang et al., 2025a; Fan* et al., 2025), as well as comprehensive surveys of agent capabilities including memory, tool learning, and planning (Yang* et al., 2026). Most of these works validate a single paradigm family against some baselines on tasks tailored to the method's strengths. Our focus is complementary: we ask how these paradigm families compare under a common implementation and evaluation pipeline, and whether their differences can be understood as task-dependent reasoning paradigms rather than isolated method wins.

2.2. Agent Evaluation and Routing

Benchmark suites such as AgentBench (Liu et al., 2024), WebArena (Zhou et al., 2024), and MINT (Wang et al., 2024) evaluate complete agent systems in realistic environments. These benchmarks are valuable, but they confound paradigm choice with many other system decisions, including the underlying model, prompting strategy, tool interfaces, and environment design. Our work isolates one design axis, the inference-time paradigm, within a shared framework.

Our routing analysis builds on a growing line of work on LLM routing and adaptive inference. RouteLLM (Ong et al., 2024) trains routers to dispatch queries between strong and weak models based on query difficulty. FrugalGPT (Chen et al., 2023a) cascades through models of increasing cost until a satisfactory answer is found. Shnitzer et al. (2023) and Lu et al. (2024) route among multiple LLMs using benchmark-derived performance profiles. More recently, Zhang et al. (2025c) unite smaller LLMs via routing to challenge proprietary models, Zhang et al. (2025b) optimize routing for cost-performance trade-offs, and Yue et al. (2025) learn to route LLMs within multi-agent systems. These approaches select among *models*; we select among *reasoning paradigms* for a fixed model, which is a complementary and largely unexplored axis of test-time optimization.

3. Experimental Framework

3.1. Paradigms as Inference-Time Policies

We treat a reasoning paradigm as a structured inference strategy $\mathcal{P}$ that maps a task q to a sequence of language-model calls, optional tool invocations, intermediate state updates, and a final answer $\hat{y}$. Under this view, paradigm choice determines how much test-time computation is spent on planning, acting, revising, or executing code. For a task $\tau = (q, y^*)$, each paradigm induces

$$\hat{y} = \mathcal{P}(\text{LLM}, q, \mathcal{T}), \tag{1}$$

where $\mathcal{T}$ denotes the available tool repertoire. We then evaluate correctness with a dataset-specific scoring function $\text{eval}_d(\hat{y}, y^*)$.

This framing lets us compare paradigms as algorithms for allocating inference-time computation. Importantly, we do not claim that every paradigm differs only in verbal reasoning style. Some paradigms invoke tools and others do

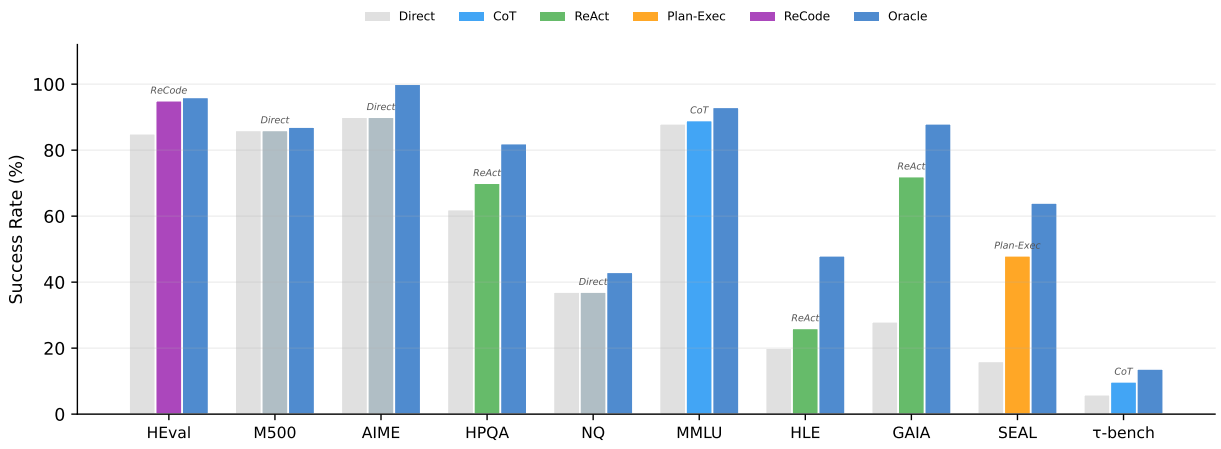

Figure 1. Direct prompting (gray), best single paradigm per dataset (colored), and oracle per-task selection (blue) for GPT-5. The best paradigm differs across tasks, and the oracle substantially exceeds any fixed choice, motivating our select-then-solve approach.

not. We treat this as part of the deployed paradigm family being studied: ReAct without tools is not the same reasoning paradigm as the ReAct systems commonly used in practice.

3.2. Studied Paradigms

We evaluate six representative paradigms that differ along two key dimensions: the degree of external control imposed on the model's reasoning process, and whether the paradigm grants access to external tools. Table 1 summarizes the paradigms studied.

Reasoning control. The paradigms impose varying levels of external structure on the model's reasoning. **Direct** imposes no control: the model receives the task and answers freely, using whatever internal reasoning it deems appropriate. This is not equivalent to "no reasoning"; a capable model may internally perform multi-step derivation, but the scaffolding does not prescribe how. **CoT** adds instruction-level control by explicitly prompting "think step by step," which can help weaker models organize their reasoning but may constrain stronger models that already reason effectively. **ReAct**, **Plan-Execute**, and **Reflection** impose orchestration-level control by structuring inference into multi-turn loops (thought-action, plan-then-execute, or generate-critique-revise). **ReCode** replaces the reasoning substrate entirely, using code generation and execution rather than natural language as the medium for problem-solving.

Tool access. Orthogonally, paradigms differ in whether they grant the model access to external tools. Direct and CoT operate without tools, relying entirely on parametric knowledge. The remaining four paradigms provide a web-search interface and a Python code-execution tool. This distinction is critical: tool access enables the model to retrieve information beyond its training data, which explains why tool-using paradigms dominate on information-seeking tasks (e.g., GAIA, SEAL) while Direct often suffices for knowledge-centric tasks (e.g., MMLU, NQ).

The repository also contains exploratory strategies outside this paper's scope; all quantitative results in the main text use only the six paradigms above.

3.3. Framework, Benchmarks, and Models

All paradigms share a common `BaseAgent` implementation for model access, tracing, metric collection, and result serialization. Tool-using paradigms access the same two tools: a web-search interface and a Python code-execution tool. Our unit of comparison is the paradigm family, not a fully orthogonalized factorial design over reasoning and tool access; this matches how paradigms are typically deployed.

We evaluate on ten benchmarks: HumanEval (Chen et al., 2021), MATH500 (Hendrycks et al., 2021b), AIME, HotpotQA (Yang et al., 2018), Natural Questions (Kwiatkowski et al., 2019), MMLU (Hendrycks et al., 2021a), HLE (Phan et al., 2025), GAIA (Mialon et al., 2024), τ-bench (Yao et al., 2024), and SEAL, covering code generation, mathematics, QA, knowledge, and tool-use tasks. Table 5 in the Appendix lists details. For large benchmarks we evaluate a fixed random subset (seed 42); for smaller ones we use the full set, yielding 761 tasks per model-paradigm pair and roughly 18k completed runs.

We evaluate four frontier LLMs: GPT-5, Qwen3-30B-A3B, Qwen3-Max, and Gemini-3-Flash. For every task we record correctness, total tokens, LLM call count, tool call count, and wall-clock time. Correctness is dataset-specific: code tasks use execution-based evaluation, math tasks use numeric matching, multiple-choice tasks use option extraction,

Table 1. Reasoning paradigms studied in the main paper, organized by the degree of reasoning control and tool access.

Paradigm	Key Idea	Control	Tools	LM Calls	Budget
Direct	Free-form answer	None	No	1	—
CoT	Step-by-step reasoning	Instructed	No	1	—
ReAct	Thought–action loop	Orchestrated	Yes	1–15	$T_{\max} = 15$
Plan-Execute	Plan, then act	Orchestrated	Yes	2–16	no replanning
Reflection	Critique and revise	Orchestrated	Yes	3–45	$M_{\max} = 3$
ReCode	Code as reasoning	Substrate	Yes	2–10	hierarchical

and QA tasks use normalized text overlap and LLM as judge.

4. Experiments

Our experiments address four questions in sequence. (1) How much does reasoning structure add beyond Direct prompting on each task type? (2) What is the computational cost of that structure? (3) Does the value of structure change across models of different capability? (4) How complementary are the paradigms at the individual task level? The answers to these questions motivate the routing approach in Section 5.

4.1. Experimental Setup

The experimental code uses a unified `BaseAgent` with swappable strategy modules and a common result format. Each task result is cached as a JSON file keyed by (model, paradigm, dataset, task_id), which makes interrupted runs resumable and lets us derive all summary tables from the raw per-task outputs. The repository also contains exploratory strategies beyond the six paradigms studied here; all numbers in this paper are produced from a paper-specific configuration released with the code, which defines an evaluation budget of 761 tasks per model-paradigm pair (18,264 runs in total, with roughly 18k completed runs in the current matrix).

Evaluation is dataset-specific. For HumanEval we extract Python code from model responses and execute against the benchmark tests. For MATH500 and AIME we extract numeric answers, including boxed and fractional LaTeX forms, and compare with tolerance $\epsilon = 10^{-6}$. For HotpotQA, NQ, GAIA, and SEAL we apply normalized text matching with token-overlap fallback. For MMLU we extract the selected option letter. For τ-bench we compare the predicted action sequence against the reference workflow. All prompts instruct the model to place its final answer inside `\boxed{}`, ensuring consistent answer extraction across paradigms. Prompt templates are provided in the Appendix.

4.2. The Marginal Value of Reasoning Structure

Table 2 presents the complete results across all four models, six paradigms, and ten datasets. We first analyze GPT-5, our strongest model, before examining cross-model patterns. The key question is whether reasoning structure provides consistent gains or whether its value depends on the task.

The marginal value of reasoning structure varies dramatically across tasks. Where tasks create information gaps that tools can fill, structure provides transformative gains: ReAct outperforms Direct by 44pp on GAIA and Plan-Execute improves 32pp on SEAL. In contrast, on knowledge-centric tasks where parametric knowledge suffices, Direct nearly matches the best paradigm: 88% vs 89% on MMLU, and best or tied-best on NQ and MATH500. Adding structure to these tasks increases cost without proportionate accuracy gain.

Structure can also actively hurt. CoT underperforms Direct by 15pp on HumanEval because step-by-step reasoning degrades code quality. At the other extreme, some tasks remain hard for all paradigms: the best paradigm achieves only 26% on HLE and below 10% on τ-bench, demonstrating capability ceilings that no reasoning structure can overcome.

A striking pattern emerges: the **best** paradigm is different for nearly every dataset. CoT/ReAct leads on MMLU, ReAct on GAIA and HotpotQA, Plan-Execute on SEAL, ReCode on MATH500, and Direct on NQ and AIME. This raises a natural question: if different tasks favor different paradigms, what would happen if we could select the right paradigm for each task?

4.3. The Cost of Reasoning Structure

Accuracy gains must be weighed against computational cost (Figure 2). Token usage forms clear tiers: Direct and CoT at 1.0–1.1×, ReCode at 2.8×, ReAct at 4.0×, Plan-Execute at 6.9×, and Reflection at 9.4×. Reflection costs 9.4× more than Direct on HLE for only 2pp gain, while ReAct's overhead buys 44pp on GAIA. The Pareto frontier in Figure 2 makes clear that no single paradigm dominates on both axes; the best operating point depends on the marginal value of accuracy in the deployment context.

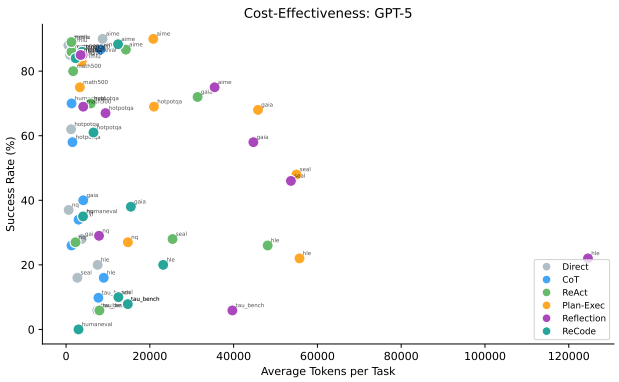

Figure 2. Cost-effectiveness scatter plot: success rate vs. average tokens per task. Points closer to the top-left corner represent better cost-efficiency. Direct and CoT cluster at the cheap end, Reflection at the expensive end, and ReAct/Plan-Execute occupy intermediate cost tiers with sharply task-dependent returns.

4.4. Structure as Capability Compensation

If reasoning structure provides universally additive value, its contribution $\Delta_{\mathcal{P}}$ should be roughly constant across models. If it compensates for capability gaps, $\Delta_{\mathcal{P}}$ should be larger for weaker models.

Reasoning structure compensates for capability gaps. The data strongly support the compensation hypothesis. On HumanEval, CoT improves Qwen3-30B from 18% to 64%, a 46pp gain, while the same paradigm *hurts* GPT-5 by 15pp. On GAIA, ReAct lifts Qwen3-30B from 12% to 28% and Gemini from 44% to 70%, but GPT-5 sees a similarly large jump from 28% to 72%. The marginal value of reasoning structure *decreases* as model capability *increases* on tasks within the model's reach, but remains large on tasks requiring external information regardless of model strength. On MATH500, the gap between Direct and the best paradigm narrows from 23pp for Qwen3-30B to 0pp for GPT-5, confirming that structure primarily compensates for what the model cannot do alone.

Task-type interactions are model-invariant. Despite substantial variation in absolute performance, GPT-5 reaches 90.0% on AIME while Qwen3-30B reaches only 11.7%, the relative paradigm rankings remain remarkably stable across models. ReAct or Plan-Execute consistently tops GAIA and HotpotQA across all four models, ReCode leads on HumanEval for all models after our format fix, and Direct or ReCode dominates MATH500 universally. HLE remains hard at 14–26% and τ-bench near-zero for all. This stability suggests that task structure, not model capability, is the primary driver of which paradigm works best.

4.5. Paradigm Complementarity

The preceding analysis repeatedly highlights one observation: the best paradigm differs across tasks. GAIA favors ReAct, MATH500 favors ReCode, SEAL favors Plan-Execute, and NQ favors Direct. But this is at the dataset level. We now ask: does this complementarity hold at the individual task level? If so, how large is the potential gain from task-level paradigm selection?

Oracle analysis. For each task, we identify whether any of the five paradigms (excluding the Direct baseline) solves it correctly, and compute the oracle accuracy: the success rate achieved by always selecting the best paradigm per task.

Table 3 reveals that oracle selection outperforms the best single paradigm by 17.1pp on average. The gap is largest for Qwen3-30B at 24.1pp and smallest for GPT-5 at 12.4pp, yet even for GPT-5 it represents a 20% relative improvement.

Paradigms solve genuinely different problems. This oracle gap could arise trivially if one paradigm solved almost all solvable tasks. Figure 8 tests this by computing the Jaccard similarity between paradigm success sets. The relatively low overlap between ReCode and other paradigms confirms genuine complementarity. CoT and Reflection show the highest overlap at 0.61, yet even they disagree on 39% of their combined success sets.

Summary. Across four models and ten benchmarks, we find that reasoning structure is task-conditional, not universally beneficial. Structure helps most when it closes information gaps via tools, hurts when it constrains an already-capable model, and has no effect at capability ceilings. The 17.1pp oracle gap and low inter-paradigm overlap confirm that this variation is not noise but genuine complementarity. This motivates the routing approach we develop next.

4.6. Qualitative Case Studies

To ground the aggregate statistics, we present two representative cases from GPT-5 that illustrate the dominant failure and success modes. Two additional cases (tool-essential GAIA query and capability-ceiling HLE problem) appear in Appendix H.

Case 1: Structure hurts on a knowledge-centric task (MMLU). The question asks which type of statistical error (Type I or Type II) is more serious for a dialysis center vs. a towel factory, with the correct answer C (dialysis: Type II; towels: Type I).

Table 2. Complete success rates (%) across all models, paradigms, and datasets. **Best** and second best paradigm per (model, dataset) are highlighted. Avg is the unweighted mean across all ten datasets.

Model	Paradigm	HEval	M500	AIME	HPQA	NQ	MMLU	HLE	GAIA	SEAL	τ-b	Avg
GPT-5	Direct	85.0	**86.0**	90.0	62.0	**37.0**	88.0	20.0	28.0	16.0	5.9	51.8
	CoT	70.0	84.0	86.7	58.0	26.0	**89.0**	16.0	40.0	34.0	**9.8**	51.3
	ReAct	86.0	80.0	86.7	**70.0**	27.0	**89.0**	**26.0**	**72.0**	28.0	5.9	57.1
	Plan-Exec	35.0	75.0	90.0	69.0	27.0	83.0	22.0	68.0	**48.0**	7.8	52.5
	Reflect.	85.0	69.0	75.0	67.0	29.0	85.0	22.0	58.0	46.0	5.9	54.2
	ReCode	**95.0**	**86.0**	88.3	61.0	35.0	84.0	20.0	38.0	10.0	7.8	52.5
Gemini-Flash	Direct	73.0	**82.0**	40.0	64.0	**43.0**	**90.0**	14.0	44.0	22.0	7.8	48.0
	CoT	**99.0**	76.0	**53.3**	62.0	30.0	60.0	16.0	54.0	**62.0**	3.9	51.6
	ReAct	98.0	12.0	3.3	68.0	31.0	88.0	10.0	70.0	**62.0**	5.9	44.8
	Plan-Exec	98.0	25.0	10.0	**72.0**	29.0	56.0	12.8	**71.4**	35.0	**15.0**	42.4
	Reflect.	**99.0**	15.0	3.3	67.0	31.0	86.0	10.0	60.0	50.0	10.0	43.1
	ReCode	81.0	80.0	26.7	64.0	39.0	**90.0**	**25.0**	45.0	25.0	**15.0**	49.1
Qwen3-Max	Direct	85.0	56.0	58.3	52.0	27.0	**83.0**	14.0	18.0	10.0	**9.8**	41.3
	CoT	94.0	41.0	38.3	59.0	26.0	53.0	**16.0**	44.0	**40.0**	5.9	41.7
	ReAct	**96.0**	50.0	61.7	59.0	28.0	66.0	**16.0**	**56.0**	30.0	2.0	46.5
	Plan-Exec	71.0	47.0	55.0	**65.0**	29.0	68.0	14.0	48.0	38.0	0.0	43.5
	Reflect.	92.0	46.0	61.7	54.0	27.0	10.0	14.0	48.0	30.0	2.0	38.5
	ReCode	**96.0**	**45.0**	1.7	50.0	**33.0**	80.0	12.0	8.0	2.0	5.9	33.4
Qwen3-30B	Direct	18.0	42.0	10.0	27.0	20.0	70.0	**14.0**	12.0	6.0	0.0	21.9
	CoT	64.0	15.0	3.3	34.0	19.0	48.0	2.0	**28.0**	**28.0**	**3.9**	24.5
	ReAct	52.0	31.0	8.3	45.0	18.0	65.0	10.0	**28.0**	12.0	0.0	26.9
	Plan-Exec	7.0	34.0	11.7	43.0	**29.0**	61.0	12.0	24.0	20.0	0.0	24.2
	Reflect.	50.0	29.0	1.7	**48.0**	25.0	65.0	10.0	14.0	20.0	0.0	26.3
	ReCode	**96.0**	**33.0**	5.0	17.0	17.0	**70.0**	8.0	4.0	0.0	**3.9**	25.4

Table 3. Oracle analysis: per-task best paradigm selection achieves substantially higher accuracy than any single paradigm. Oracle Gap = Oracle − Best-single.

Model	Direct	Best-Single	Oracle	Gap
GPT-5	60.3%	61.8% (ReAct)	74.1%	+12.4pp
Gemini-Flash	48.0%	56.1% (CoT)	75.2%	+19.1pp
Qwen3-Max	41.3%	46.5% (ReAct)	72.5%	+21.6pp
Qwen3-30B	21.9%	31.7% (ReAct)	55.7%	+24.1pp
Average	42.9%	49.0%	69.4%	+17.1pp

Paradigm	Success	Answer	Tokens
Direct	✓	C	900
CoT	✓	C	1,872
ReAct	✗	D	1,350
Plan-Execute	✗	D	1,878
Reflection	✗	D	3,064
ReCode	✓	C	1,536

ReAct, Plan-Execute, and Reflection all answered D, arguing that Type II error is more serious in *both* cases. Their extended reasoning about the towel manufacturer led them to over-weight the consequences of missed defects, failing to recognize that for towels (a low-stakes product) an unnecessary production halt (Type I) is the more costly concern. Reflection even validated its own wrong answer as "SATISFACTORY" during self-critique, illustrating how additional reasoning can reinforce rather than correct an initial error. Direct, with minimal reasoning overhead, correctly identified C.

Case 2: Paradigm complementarity (SEAL). The question asks which nation ranked as the world's second-largest rice producer in 2023 per the FAO; the correct answer is China.

Paradigm	Success	Answer	Tokens	Tools
Direct	✗	India	648	0
CoT	✗	India	966	0
ReAct	✓	China	11,174	5
Plan-Execute	✓	China	25,600	9
Reflection	✗	India	32,650	10
ReCode	✗	India	9,168	1

This question tests whether parametric knowledge can be overridden by evidence. The model's default belief strongly associates India as the #2 rice producer, but the 2023 FAO data places China second. Most strikingly, Reflection made 10 tool calls and *still* answered "India": it found conflicting evidence but its self-reflection loop reinforced the initial parametric belief. Only ReAct and Plan-Execute, which are structured to act on retrieved evidence rather than validate prior beliefs, answered correctly. This illustrates that tool access alone is insufficient; the paradigm must also be structured to *trust* retrieved evidence over prior beliefs.

5. Select-then-Solve: Paradigm Routing

The 17.1pp oracle gap establishes that paradigm complementarity is substantial. A natural idea follows: before answering a task, first decide which reasoning paradigm to use. If the selection is good, the agent solves more tasks than any fixed paradigm choice.

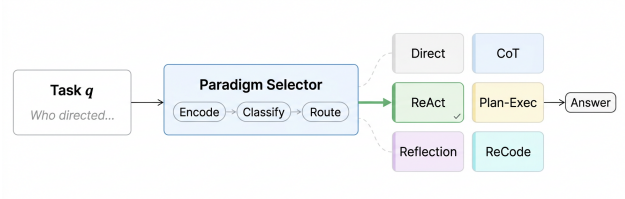

Figure 3. The select-then-solve pipeline. A Paradigm Selector encodes, classifies, and routes each task to one of six paradigms. Only the selected paradigm is executed.

5.1. Routing Strategies

We evaluate three routing strategies of increasing sophistication. The simplest uses 22 handcrafted features capturing dataset identity, text statistics, and content detectors, trained with Logistic Regression and a 2-layer MLP. The second encodes task text via `text-embedding-3-small` into 1536 dimensions, optionally concatenated with the handcrafted features. The third requires no training: each model reads its own task and selects a paradigm via zero-shot prompting. All routers select from six paradigms including Direct, so they can learn when no structure is needed. We train per-model classifiers on 70% of tasks and evaluate on the held-out 30%, measuring downstream task accuracy. Labels are the most token-efficient successful paradigm per task; tasks where all paradigms fail are excluded from training but counted as incorrect during evaluation.

5.2. Routing Results

Figure 4 and Table 7 in the Appendix show a clear progression from simple to effective routing. The Combined LR router, which concatenates embeddings with handcrafted features, achieves 53.1% average accuracy, improving 5.5pp over Direct at 47.6% and 2.8pp over the best fixed paradigm at 50.3%. Every embedding-based router exceeds Direct on all four models, with gains largest on Gemini at +6.7pp and GPT-5 at +3.9pp.

The upgrade from handcrafted to embedding features is meaningful. Handcrafted predictors rely on dataset identity, amounting to per-dataset majority voting. Embedding-based routers use the task text directly, enabling within-dataset discrimination. On GPT-5, the best handcrafted router merely ties the best fixed paradigm at 62.4%, while the embedding router exceeds it at 64.2%. On Gemini, the embedding LR at 61.9% outperforms the handcrafted LR at 58.5% by 3.4pp. Combined LR recovers 26% of the oracle gap on average, reaching 31% on GPT-5 and 37% on Gemini, from a single embedding API call per task. Oracle gap recovery is computed as (Router − Direct) / (Oracle − Direct); for GPT-5: $(64.2 - 60.3)/(72.9 - 60.3) \approx 31\%$; for Gemini: $(62.2 - 55.5)/(73.4 - 55.5) \approx 37\%$. Because the router often selects Direct over expensive paradigms, the average token cost under routing is approximately 7.0k per task for GPT-5, compared to 3.5k for always-Direct and 14.0k for always-ReAct. The router achieves higher accuracy than ReAct at half its token cost.

5.3. Self-Routing

Zero-shot self-routing, where each model selects its own paradigm without training, averages 48.4%, slightly above Direct at 47.6%. GPT-5 achieves 67.1%, surpassing its Direct baseline, but weaker models fail: Qwen3-Max drops to 42.4% and Qwen3-30B to 27.5%, well below their baselines. Weaker models overwhelmingly select tool-using paradigms regardless of task type, even for knowledge-centric questions where Direct suffices.

Table 4 reveals distribution asymmetry in detail. The learned router produces model-adapted distributions: it assigns 47–73% of tasks to Direct depending on model capability, with the remainder spread across CoT, ReAct, and other paradigms. In contrast, self-routing collapses to a single dominant choice per model. GPT-5 over-selects Direct at 65%, Qwen3-Max and Qwen3-30B over-select ReAct at 42–48%, and no model ever selects Reflection. This pattern suggests that self-routing fails not because models lack knowledge of the paradigms, but because they cannot calibrate which paradigm matches each task's requirements. The ability to execute a paradigm when instructed does not imply the ability to choose when to use it, highlighting paradigm selection as a distinct meta-reasoning capability.

6. Discussion

What the router learns. Table 4 shows the paradigm distribution predicted by the Combined LR router. The router assigns Direct to 47–73% of tasks depending on the model, confirming that it learns to avoid unnecessary reasoning structure. The fraction routed to Direct increases with model capability, from 47% for Qwen3-30B to 66% for GPT-5, consistent with our finding that stronger models need less structure. Weaker models receive more diverse routing across all six paradigms.

The contrast with self-routing is revealing. GPT-5 over-selects Direct at 65%, achieving 67.1% accuracy that exceeds Direct but still trails the learned router. Qwen3-Max and Qwen3-30B exhibit the opposite bias, over-selecting ReAct at 42–48%, which hurts on knowledge-centric tasks. No model ever selects Reflection. While GPT-5's self-routing partially works, weaker models cannot calibrate their own paradigm selection.

When does Direct outperform structured paradigms? Direct prompting sometimes matches or exceeds more elaborate paradigms. This is not a confound but a central finding.

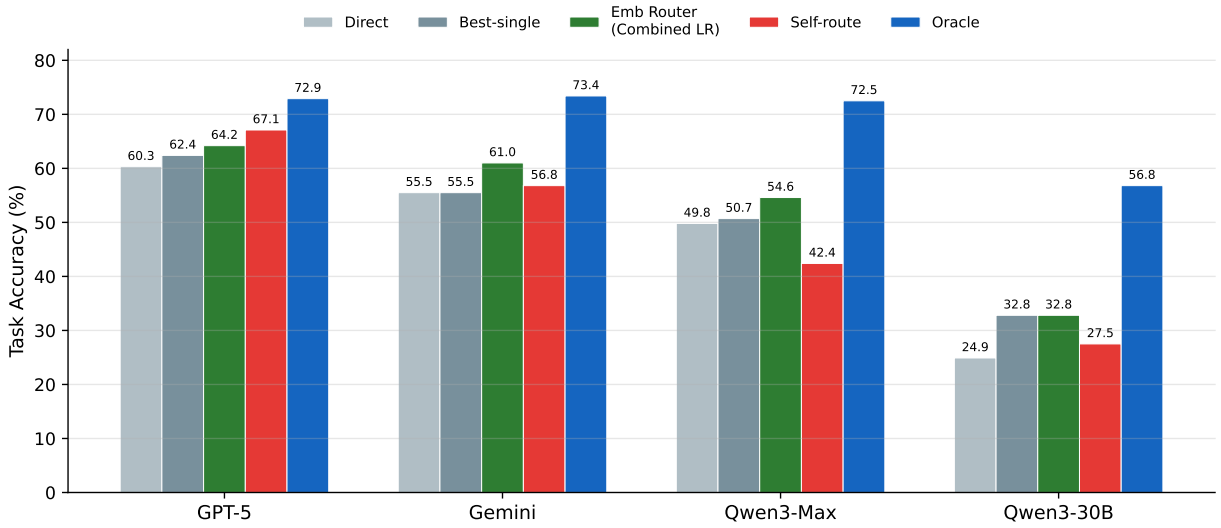

Figure 4. Router comparison across four models. The embedding router (green) consistently outperforms Direct and Best-single baselines. Self-routing (red) shows mixed results. The oracle (blue) shows substantial remaining headroom.

Table 4. Paradigm distribution (%) predicted by the learned router vs. zero-shot self-routing on the test set. Self-routing shows degenerate biases; the learned router produces diverse, model-adapted distributions.

	Model	Direct	CoT	ReAct	Plan-Exec	ReCode	Reflect.
Router	GPT-5	65.9	9.6	14.4	7.9	1.3	0.9
	Gemini	73.4	13.5	6.5	0.4	6.1	0.0
	Qwen3-Max	58.9	19.7	2.6	7.0	10.9	0.9
	Qwen3-30B	47.2	26.2	15.7	3.1	6.6	1.3
Self	GPT-5	65.1	18.8	15.3	0.4	0.4	0.0
	Gemini	72.1	10.9	16.6	0.0	0.4	0.0
	Qwen3-Max	21.4	10.9	42.4	3.5	21.8	0.0
	Qwen3-30B	16.6	29.3	48.0	0.9	5.2	0.0

We identify three scenarios: (1) *Knowledge-centric tasks*: on MMLU the answer exists in parametric knowledge and additional steps add latency without improving accuracy. (2) *Strong models on moderate tasks*: on MATH500, GPT-5 achieves 86% with Direct, matching ReCode. (3) *Tasks where structure introduces errors*: on HumanEval, CoT underperforms Direct by 15pp because step-by-step reasoning degrades code quality. The router correctly learns these patterns: it routes 66% of GPT-5 tasks to Direct, reserving structured paradigms for tasks where they provide genuine value.

7. Conclusion

We presented a controlled study of six inference-time reasoning paradigms across four LLMs and ten benchmarks. The value of reasoning structure is sharply task-dependent, creating complementarity that an embedding-based router exploits to outperform any fixed paradigm, recovering up to 37% of the oracle gap at half the token cost of always using ReAct. Self-routing only works for the strongest model, revealing paradigm selection as a distinct meta-reasoning capability. The most effective agents may not be those with the most elaborate scaffolds, but those that know when to use them and when to step aside. Limitations and future work are discussed in Appendix A.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. We study how to make LLM agents more reliable and cost-efficient by selecting the appropriate inference-time strategy per task. By reducing reliance on unnecessary tool calls and reasoning rollouts, the proposed router can lower the energy and monetary cost of agent deployment. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Chen, L., Zaharia, M., and Zou, J. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023a.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023b.

Fan*, Y., Zhang*, K., Zhou*, H., Zuo, Y., Chen, Y., Fu, Y., et al. Ssrl: Self-search reinforcement learning. *arXiv preprint*, 2025.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models. *International Conference on Machine Learning*, pp. 10764–10799, 2023.

Google DeepMind. Gemini 3 flash: Best for frontier intelligence at speed. *Google AI Blog*, 2025.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *International Conference on Learning Representations*, 2021a.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. *Advances in Neural Information Processing Systems*, 34, 2021b.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:22199–22213, 2022.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., and Stone, P. Llm+p: Empowering large language models with optimal planning proficiency, 2023. URL https://arxiv.org/abs/2304.11477.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., et al. Agentbench: Evaluating llms as agents. *International Conference on Learning Representations*, 2024.

Lu, K. et al. Routing to the expert: Efficient reward-guided ensemble of large language models. *Annual Meeting of the Association for Computational Linguistics*, 2024.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2023.

Mialon, G., Fourrier, C., Swift, C., Wolf, T., LeCun, Y., and Scialom, T. GAIA: A benchmark for general AI assistants. *International Conference on Learning Representations*, 2024.

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

OpenAI. Introducing GPT-5. *OpenAI Blog*, 2025.

Phan, L. et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023.

Shnitzer, T., Ou, A., Silva, M., Soule, K., Sun, Y., Solomon, J., Thompson, N., and Yurochkin, M. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*, 2023.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., and Lim, E.-P. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models, 2023a. URL https://arxiv.org/abs/2305.04091.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models, 2023b. URL https://arxiv.org/abs/2203.11171.

Wang, X., Wang, Z., Liu, J., Chen, Y., Yuan, L., Peng, H., and Ji, H. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *International Conference on Learning Representations*, 2024.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought

prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022.

Yang*, X., Li*, L., Zhou*, H., Zhu*, T., Qu, X., Fan, Y., Wei, Q., Ye, R., Kang, L., Qin, Y., et al. Toward efficient agents: Memory, tool learning, and planning. *arXiv preprint arXiv:2601.14192*, 2026.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *Conference on Empirical Methods in Natural Language Processing*, 2018.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *International Conference on Learning Representations*, 2023.

Yao, S. et al. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yu, Z., Zhang, J., Su, H., Zhao, Y., Wu, Y., Deng, M., Xiang, J., Lin, Y., Tang, L., Luo, Y., Liu, B., and Wu, C. Recode: Unify plan and action for universal granularity control, 2026. URL https://arxiv.org/abs/2510.23564.

Yue, Y., Zhang, G., Liu, B., Wan, G., Wang, K., Cheng, D., and Qi, Y. Masrouter: Learning to route LLMs for multi-agent systems, 2025.

Zhang, G., Geng, H., Yu, X., Yin, Z., Zhang, Z., Tan, Z., Zhou, H., Li, Z., Xue, X., Li, Y., et al. The landscape of agentic reinforcement learning for llms: A survey. *Transactions on Machine Learning Research*, 2025a.

Zhang, Y., Li, H., Chen, J., Zhang, H., Ye, P., Bai, L., and Hu, S. Beyond GPT-5: Making LLMs cheaper and better via performance-efficiency optimized routing. In *Proceedings of the Seventh International Conference on Distributed Artificial Intelligence*, pp. 122–129. ACM, 2025b.

Zhang, Y., Li, H., Wang, C., Chen, L., Zhang, Q., Ye, P., Feng, S., Wang, D., Wang, Z., Wang, X., Xu, J., Bai, L., Ouyang, W., and Hu, S. The avengers: A simple recipe for uniting smaller language models to challenge proprietary giants, 2025c.

Zhang, Z., Zhang, A., Li, M., and Smola, A. Automatic chain of thought prompting in large language models. *International Conference on Learning Representations*, 2023.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, Y. B. T., Fried, D., Alon, U., et al. Webarena: A realistic web environment for building autonomous agents. *International Conference on Learning Representations*, 2024.

A. Limitations and Future Work

Limitations. We compare one implementation per paradigm family, not the full space of prompts, search budgets, or tool interfaces. We evaluate a fixed benchmark sample rather than re-sampling across seeds. Tool access is treated as part of the paradigm family rather than fully orthogonalized. Our router uses a modest training set of approximately 400 tasks per model; larger sets would likely improve accuracy.

Future work. Immediate extensions include fine-tuning small language models as paradigm routers, multi-objective routing trading accuracy against cost, and transferring routing across models. More broadly, our framework supports adaptive strategies that mix paradigms within a single task, escalating from Direct to tool-using paradigms only when needed.

B. Evaluation Benchmarks

Table 5. Evaluation sets used in the paper. "Eval instances" refers to the number of tasks actually run per model-paradigm pair.

Benchmark	Domain	Eval Instances	Metric	Notes
HumanEval	Code	100	Execution	function completion
MATH500	Math	100	Numeric	multi-step derivation
AIME	Math	60	Numeric	olympiad-style problems
HotpotQA	QA	100	Text match	multi-hop QA
NQ	QA	100	Text match	factoid QA
MMLU	Knowledge	100	Letter match	broad subject coverage
HLE	Hard reasoning	50	LLM-as-judge	expert-level questions
GAIA	Agent QA	50	Text match	web/tool use required
τ-bench	Tool planning	51	Action match	service workflows
SEAL	Current knowledge	50	Text match	recent-event QA

C. Paradigm Prompt Templates

All six paradigms share the same `BaseAgent` framework and differ only in the system prompts and control flow. We list the core prompts below. All prompts include the `\boxed{}` instruction to ensure consistent answer extraction.

Direct.

```
You are a helpful assistant.  Answer the question directly and concisely.  Put your
final answer inside \boxed{}, e.g., \boxed{42} or \boxed{Yes}.
```

Chain-of-Thought (CoT).

```
You are a helpful assistant.  Think through the problem step by step before
giving your final answer.  Show your reasoning, then put your final answer inside
\boxed{}.
```

ReAct.

```
You are a helpful assistant that can use tools to answer questions.  For each step,
you should:  1) Think about what you need to do next.  2) If needed, use a tool
to gather information.  3) Once you have enough information, provide your final
answer.  Put your final answer inside \boxed{}.
```

Plan-then-Execute. Uses two prompts: a planning prompt that produces a numbered step list, followed by an execution prompt:

```
[Planning] First, create a step-by-step plan to solve the problem.  Output ONLY
the plan as a numbered list.  Do not solve the problem yet.
```

```
[Execution] You are executing a plan step by step.  You have access to tools.
After completing all steps, put your final answer inside \boxed{}.
```

Reflection. Uses three phases: initial solve, critique, and revision.

```
[Solve] Think carefully and use tools when needed to find the answer.  Put your
final answer inside \boxed{}.
[Reflect] Examine the following answer and identify any errors.  If the answer is
correct, respond with ``SATISFACTORY''. If there are issues, explain what's wrong.
[Revise] Your previous answer had issues.  Revise based on the feedback.  Put your
final answer inside \boxed{}.
```

ReCode. Generates Python code with placeholder functions, then decomposes recursively:

```
[Generate] Write a Python function solve() that solves the problem.  Mark
placeholder functions with # PLACEHOLDER. Available primitives:  web_search(query),
code_exec(code).
[Decompose] Implement the placeholder function {func_name}.  You may create new
placeholders if needed.
[Extract] Given the execution output, extract the final answer.  Put it inside
\boxed{}.
```

D. Detailed Paradigm Comparison

Table 6 provides a detailed comparison of how the six paradigms differ in their inference-time behavior.

Table 6. Detailed comparison of the six paradigms. LM calls and tool calls are per-task averages on GPT-5.

Paradigm	Avg LM Calls	Avg Tool Calls	Tools?	Multi-turn?	Key Mechanism
Direct	1.0	0	No	No	Single-shot generation
CoT	1.0	0	No	No	Step-by-step verbal reasoning
ReAct	1–15	0–14	Yes	Yes	Thought-action-observation loop
Plan-Exec	2–16	0–15	Yes	Yes	Plan first, then execute with tools
Reflection	3–45	0–42	Yes	Yes	Solve, critique, revise (up to 3 rounds)
ReCode	2–10	0–8	Yes	Yes	Recursive code decomposition

The paradigms form a spectrum of increasing inference-time structure. Direct and CoT are single-call, zero-tool paradigms that differ only in whether step-by-step reasoning is elicited. ReAct and Plan-Execute both use tools but differ in control flow: ReAct interleaves reasoning and action in a flat loop, while Plan-Execute separates planning from execution. Reflection adds a self-critique phase after initial solving. ReCode is unique in using code generation as the primary reasoning substrate.

E. Router Training Details

Label construction. The router's training labels are derived from the experimental matrix without any additional LLM calls. For each (model, task) pair, we observe the outcomes of all six paradigms from the main experiment. If at least one paradigm succeeds, the label is the successful paradigm with the lowest token cost (favoring cheaper paradigms when multiple succeed). If all paradigms fail, the task is labeled `none` and excluded from training but counted as incorrect during evaluation. This yields approximately 300–400 labeled training examples per model.

Features. We compare three feature representations: (1) *Handcrafted*: 22 features including dataset one-hot encoding (10 dims), text statistics (length, word count, line count, average word length), content flags (has_code, has_math, has_choices), and question-type indicators (5 dims). (2) *Embedding*: the task text is encoded via the `text-embedding-3-small` API, producing a 1536-dimensional vector. (3) *Combined*: concatenation of the embedding and handcrafted features (1558 dims total).

Classifiers. For each feature set, we train Logistic Regression (LR; `class_weight=balanced`, `max_iter=2000`) and a 2-layer MLP (128→64→6, dropout 0.3, Adam optimizer with learning rate 5×10^{-4}, early stopping with patience 15). Features are standardized to zero mean and unit variance before training. All classifiers are trained independently per model.

Train/test split. Tasks are split 70/30 stratified by dataset, with the same split shared across all models (532 train / 229 test). The split is fixed with seed 42 for reproducibility.

Deployment cost. At inference time, the router requires one embedding API call per task (negligible latency and cost) followed by a linear classifier prediction. No LLM calls are needed for routing. The total overhead is orders of magnitude cheaper than running a single paradigm, let alone all six.

LLM configuration. All models are accessed through an OpenAI-compatible API with `temperature=0` for deterministic outputs. The maximum context length varies by model: GPT-5 supports 128k tokens, Gemini-3-Flash 1M tokens, and Qwen3 models 32k tokens. For tool-using paradigms, we set a maximum of 15 interaction turns for ReAct, 16 for Plan-Execute, and 3 revision rounds for Reflection. The code execution tool enforces a 30-second timeout per execution. All paradigms use the same system prompt format with `\boxed{}` answer extraction. The embedding model `text-embedding-3-small` produces 1536-dimensional vectors and is called with default parameters. Complete prompt templates are listed in Appendix C.

F. Detailed Router Results

Table 7. Downstream task accuracy (%) on the held-out test set. Each router selects a paradigm per task (including Direct); accuracy is the fraction of tasks solved correctly under the selected paradigm. Best router per model is **bolded**.

	Method	GPT-5	Gemini	Qwen3-Max	Qwen3-30B	Avg
Base	Direct	60.3	55.5	49.8	24.9	47.6
	Best-single	62.4	55.5	50.7	32.8	50.3
HC	Handcrafted (LR)	59.4	58.5	51.5	34.9	51.1
	Handcrafted (MLP)	62.4	55.5	53.3	37.1	52.1
Emb	Embedding (LR)	64.2	61.9	53.3	32.3	52.9
	Combined (LR)	**64.2**	61.0	**54.6**	32.8	**53.1**
	Combined (MLP)	62.4	**62.2**	53.3	**33.6**	52.9
ZS	Self-route	60.3	56.8	42.4	27.5	46.7
	Oracle	72.9	73.4	72.5	56.8	68.9

G. Additional Figures

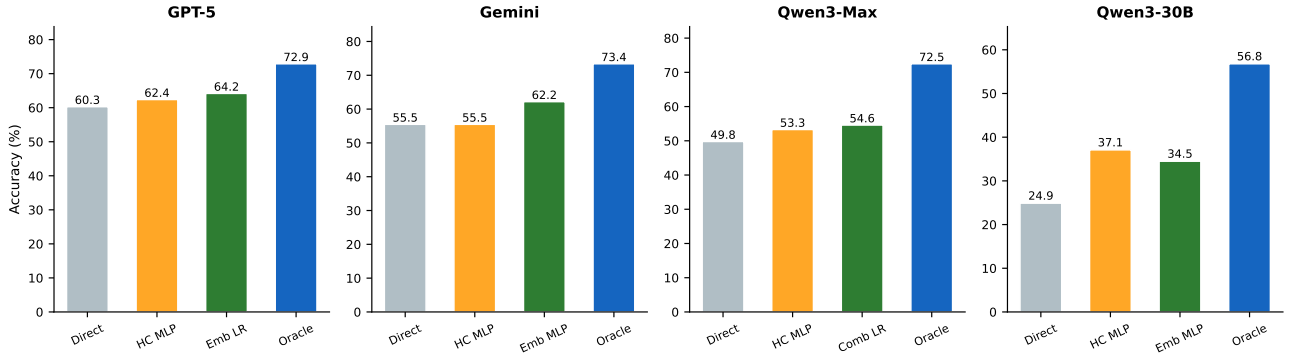

Figure 5. Oracle gap recovery across models. Each group shows the progression from Direct (gray) through handcrafted router (orange) and embedding router (green) to the Oracle upper bound (blue). The embedding router consistently narrows the gap, with the largest recovery on Gemini.

H. Additional Case Studies

In addition to the two cases in Section 4.6, we present two further cases from GPT-5 that illustrate (a) the necessity of tools for information-seeking tasks and (b) hard capability ceilings.

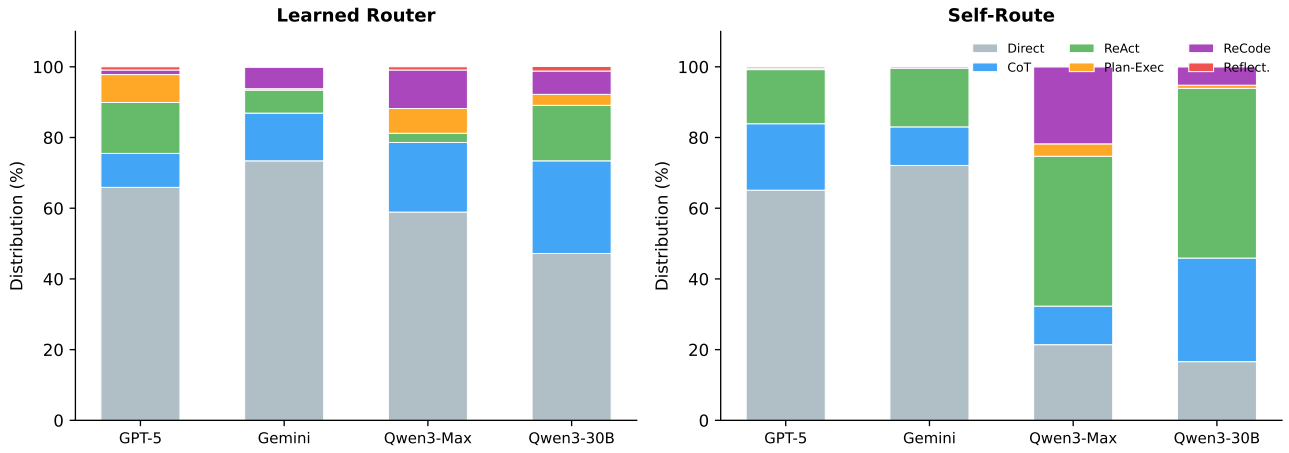

Figure 6. Paradigm distribution comparison: learned router (left) vs. zero-shot self-routing (right). The learned router produces diverse, model-adapted distributions, while self-routing shows degenerate biases (GPT-5 selects Direct 100%; Qwen3-30B over-selects ReAct at 48%).

Case 3: Tools are essential (GAIA).

Who did the actor who played Ray in the Polish-language version of Everybody Loves Raymond play in Magda M.? Give only the first name.
Answer: Wojciech

Paradigm	Success	Answer	Tokens	Tool Calls
Direct	✗	Andrzej	5,192	0
CoT	✗	Bartek	4,566	0
ReAct	✓	Wojciech	21,248	7
Plan-Execute	✓	Wojciech	15,417	5
Reflection	✓	Wojciech	13,177	3
ReCode	✗	Piotr	1,889	1

This multi-hop question requires: (1) identifying the Polish adaptation "Wszyscy Kochaja Romana", (2) finding the actor who played the Ray counterpart (Bartlomiej Kasprzykowski), and (3) looking up his role in Magda M. Without tools, every paradigm hallucinated a different Polish first name. The three paradigms with sufficient tool interaction (ReAct with 7 searches, Plan-Execute with 5, Reflection with 3) all converged on the correct answer. ReAct's trace shows it progressively refining its web queries from English to Polish-language searches before finding the correct actor and role.

Case 4: Capability ceiling (HLE).

The braid group B_n acts on the torus link $T(n,n) \subset S^3$ by permuting strands, inducing an action on $Kh(T(n,n);\mathbb{Q})$. Let d_n be the dimension of the B_n-fixed subspace. Find $\prod_{n=1}^{8} d_n$.
Answer: 2,490,840,000

Paradigm	Success	Answer	Tokens	Tool Calls
Direct	✗	362,880	5,300	0
CoT	✗	362,880	7,849	0
ReAct	✗	362,880	86,430	15
Plan-Execute	✗	362,880	103,027	15
Reflection	✗	362,880	79,097	13
ReCode	✗	362,880	13,132	1

All six paradigms produce the same wrong answer: 362,880 ($= 9!$). Each conjectures $d_n = n + 1$ via a plausible but incorrect argument citing Schur-Weyl duality and symmetric tensors. The actual computation requires detailed knowledge of Khovanov homology that exceeds the model's training data. ReAct spent 86,430 tokens and 15 tool calls searching for

Figure 7. Success rate heatmap for GPT-5 across paradigms (rows) and datasets (columns). Clear task-paradigm interactions are visible: no single row dominates all columns.

relevant references but could not locate the specific computations needed. The total cost across all paradigms for this single failed task was over 295,000 tokens. This demonstrates a hard capability ceiling where no amount of orchestration or tool use can compensate for missing domain knowledge.

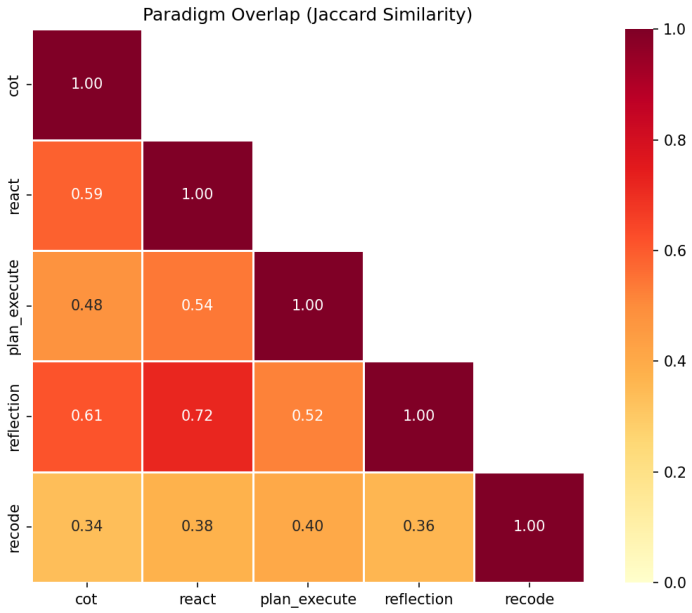

Figure 8. Jaccard similarity between paradigm success sets (aggregated across all models). Lower values indicate greater complementarity; paradigms solve different subsets of tasks.