Team of Thoughts: Efficient Test-time Scaling of Agentic Systems through Orchestrated Tool Calling

Anonymous ACL submission

Abstract

Existing Multi-Agent Systems (MAS) typically rely on homogeneous model configurations, failing to exploit the diverse expertise inherent in different post-trained architectures. We propose Team-of-Thoughts, a heterogeneous MAS framework that treats diverse models as specialized tools within an orchestrator-driven paradigm. Team-of-Thoughts introduces two novel components: (1) Orchestrator Calibration, which identifies models with superior coordination and synthesis capabilities, and (2) Agent Self-Assessment, a protocol where tool agents profile their own domain-specific strengths to guide selection. During inference, the orchestrator dynamically activates the most compatible agents based on these profiles to maximize capability coverage. Across five mathematical reasoning and code generation benchmarks, Team-of-Thoughts consistently outperforms individual models and existing MAS baselines. Notably, on AIME24 and LiveCodeBench, Team-of-Thoughts achieves 96.00% and 77.91% accuracy, respectively, significantly improving over homogeneous role-play baselines (80.00% and 65.93%).

1 Introduction

Test-time scaling (TTS) has emerged as a transformative paradigm for extending the capabilities of large language models (LLMs) beyond the constraints of their training-time performance (Snell et al., 2024; Wu et al., 2025). By allocating additional computational budget during inference, models can unlock latent reasoning trajectories to solve increasingly non-trivial tasks by utilizing methods such as process reward model (PRM) scoring, beam search, or tree-based exploration (Wei et al., 2023; Yao et al., 2023; Besta et al., 2024). This shift recognizes that the strategic deployment of inference-time compute is as fundamental to general intelligence as the scale of pre-training.

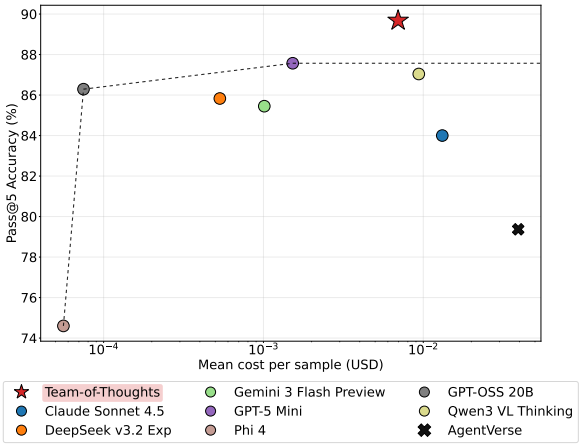

Figure 1: Pass@5 accuracy versus cost on MBPP+ for Team-of-Thoughts, single models, and AgentVerse. The dashed line denotes the Pareto front of single models.

Despite these gains, current TTS approaches remain fundamentally limited by their reliance on **model homogeneity**. Most frameworks are confined to single-model execution or static multi-agent workflows where a single model family is prompted with different personas (Qian et al., 2024; Hong et al., 2024; Wu et al., 2023). Such systems fail to exploit the "diversity dividend" inherent in the broader LLM ecosystem: different models, by virtue of distinct post-training procedures, architectural nuances, and dataset compositions, possess divergent and often complementary expertise. While recent industry developments such as Grok 4.2 (xAI, 2026) and Claude's agentic orchestration (Anthropic, 2026) signal a nascent shift toward coordinated teams, academic frameworks (Zhang et al., 2024; Chen et al., 2024) still lack the dynamic flexibility required to match specific model proficiencies to unique semantic attributes of a task.

We address these limitations with **Team-of-**

An anonymized code for this work can be accessed at: https://anonymous.4open.science/r/ToT-FA6A

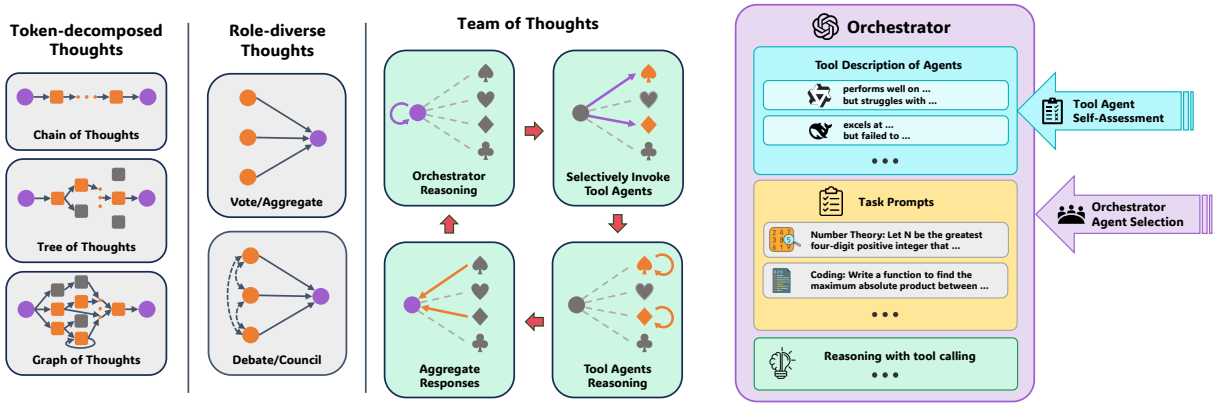

Figure 2: **Overview of Team-of-Thoughts. (Left)** Unlike single-model reasoning or homogeneous MAS (role-play), Team-of-Thoughts leverages heterogeneous model priors to maximize solution space coverage. **(Right)** Our architecture utilizes a pre-inference pipeline—comprising orchestrator calibration and agent self-profiling—to enable dynamic agent selection. During inference, the orchestrator selectively invokes optimal tool agents and synthesizes their outputs into a high-confidence final response.

Thoughts, a novel Multi-Agent System (MAS) that redefines test-time scaling through an orchestrated tool-calling paradigm (Figure 1). Rather than treating models as monolithic reasoners or assigning them rigid, instruction-based personas, our framework conceptualizes **diverse LLMs as specialized, callable tools**. By leveraging the robust, native tool-calling interfaces of modern LLMs, we construct a hierarchical architecture where a central orchestrator dynamically activates the most capable tool agents for a given query.

This design enables a "best-of-breed" approach to reasoning, facilitating massive parallelism and superior token efficiency. By replacing sequential, token-heavy reasoning with a coordinated, high-precision team effort, Team-of-Thoughts ensures that the most proficient "specialists" are invoked only when their specific expertise is required. Our key contributions are:

- **The Team-of-Thoughts Framework:** A novel MAS architecture that enables heterogeneous LLMs to collaborate through a dynamic, tool-calling hierarchy, maximizing capability coverage across the solution space.
- **Dual-Calibration Mechanisms:** We introduce an *orchestration calibration protocol* to identify models with superior synthesis capabilities and a *self-assessment protocol* that allows tool agents to autonomously profile their own domain expertise.
- **State-of-the-Art Efficiency:** Extensive evaluations demonstrate that Team-of-Thoughts consistently outperforms both standalone models and homogeneous MAS baselines. Notably, our framework achieves **96.67%** on AIME24 and **77.91%** on LiveCodeBench, while maintaining a computational cost significantly lower than traditional parallel sampling methods.

2 Background

2.1 Test-Time Scaling and Reasoning

Test-time scaling (TTS) enhances performance by allocating additional computational budget during inference. Reasoning frameworks, such as Chain-of-Thought (CoT) (Wei et al., 2023), Tree-of-Thoughts (Yao et al., 2023), and Graph-of-Thoughts (Besta et al., 2024), leverage this by decomposing tasks into intermediate steps, enabling structured search over the solution space.

However, a single agent's expressive power is bounded by its fixed parameterization θ. While search-based scaling laws (Snell et al., 2024; Wu et al., 2025; Brown et al., 2024) show performance gains, these remain constrained by the model's inherent biases, which may preclude reaching solutions in remote regions of the solution space. As training larger models is often prohibitively expensive, there is a clear need for architectures that scale capability at test-time without re-training.

2.2 Multi-Agent Systems (MAS)

Multi-Agent Systems (MAS) address single-model limitations by employing an ensemble $\{\theta_1, \theta_2, \ldots, \theta_n\}$. In these systems, agents oper-

ate through independent reasoning or collaborative interaction—exchanging intermediate states and debating hypotheses—to synthesize a final output.

Despite their potential, many current MAS frameworks are **internally homogeneous**, generating ensembles by prompting a single model with varying "personas" (Chen et al., 2024; Yang et al., 2025b; Li et al., 2025). Because these agents share identical parameterization ($\theta_1 = \theta_2 = \cdots = \theta_n$), they lack the distributional diversity necessary to explore complementary regions of the solution space. Furthermore, these frameworks are often computationally inefficient, requiring multiple rounds of exhaustive reasoning from every agent regardless of their task-specific relevance.

3 Team of Thoughts: An Efficient Heterogeneous MAS

We introduce **Team-of-Thoughts**, a multi-agent system (MAS) designed to maximize capability coverage through the coordination of heterogeneous models. The architecture comprises a central **orchestrator agent** and a diverse ensemble of **tool agents**, where each agent is characterized by a distinct, model-specific prediction distribution. The orchestrator governs the reasoning process by dynamically dispatching queries to tool agents based on the specific requirements of the input D. Its operation is defined by three core functions: (1) **Selection**, identifying the tool agents most qualified for the task; (2) **Evaluation**, assessing the quality and relevance of agent-generated outputs; and (3) **Aggregation**, synthesizing these insights to refine the global context. By utilizing this specialized tool-calling paradigm, the framework optimizes both distributional diversity and inference efficiency.

The remainder of this section provides a probabilistic motivation for the framework before detailing its formal implementation.

3.1 Probabilistic Foundations of Team-of-Thoughts

To formalize the advantages of the Team-of-Thoughts framework, we model the problem of task-solving through the lens of density estimation and stochastic processes.

Mathematical Problem Formulation

A task-solving problem consists of an input query D and an unknown target answer X within a high-dimensional solution space $\mathcal{S}$. We represent the prediction distribution of a language model with parameters θ as $p_\theta(\cdot \mid D)$. While LLMs fundamentally generate categorical distributions over discrete tokens, their pre-softmax continuous latent representations (logits and embeddings) can be effectively modeled as a multivariate Gaussian distribution (Gao et al., 2023):

$$p_\theta(X \mid D) \approx \mathcal{N}(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ denote the mean and covariance matrix, respectively. During inference, the goal is to iteratively refine this distribution to maximize the likelihood of the target X.

The Constraints of Single-Agent Trajectories

Standard agentic frameworks, such as Chain-of-Thought (CoT) (Wei et al., 2023), attempt to reach the target X by generating a sequence of intermediate reasoning steps $[X_1, X_2, \ldots, X_T]$. This can be viewed as a **Markov process** where each step X_t is sampled from a distribution conditioned on the prompt and the preceding trajectory: $X_t \sim p_\theta(\cdot \mid D, X_1, \ldots, X_{t-1})$. Mathematically, these steps shift the mean and variance of the prediction:

$$\begin{gathered}
X_1 \sim p_\theta(\cdot \mid D) \approx \mathcal{N}\left(\boldsymbol{\mu}_\theta^1, \boldsymbol{\Sigma}_\theta^1\right), \\
X_2 \sim p_\theta(\cdot \mid D, X_1) \approx \mathcal{N}\left(\boldsymbol{\mu}_\theta^2, \boldsymbol{\Sigma}_\theta^2\right), \\
X_3 \sim p_\theta(\cdot \mid D, X_1, X_2) \approx \mathcal{N}\left(\boldsymbol{\mu}_\theta^3, \boldsymbol{\Sigma}_\theta^3\right), \\
\vdots \\
\hat{X} \sim p_\theta(\cdot \mid D, X_1, X_2, \ldots, X_T) \approx \mathcal{N}\left(\boldsymbol{\mu}_\theta', \boldsymbol{\Sigma}_\theta'\right)
\end{gathered}$$

As illustrated in Figure 3 (left), these reasoning steps heuristically "drift" the initial distribution toward the target. However, the ability to transform $\mathcal{N}(\boldsymbol{\mu}_\theta', \boldsymbol{\Sigma}_\theta')$ is fundamentally constrained by the initial parameterization θ. In complex tasks where the target X is distant from the initial mean $\boldsymbol{\mu}_\theta$, a single reasoning chain often requires an impractical number of steps to converge or risks becoming trapped in local optima defined by the model's inherent biases.

Team-of-Thoughts as a Gaussian Mixture Model

Team-of-Thoughts overcomes these topological constraints by employing a Multi-Agent System (MAS) of heterogeneous agents. By utilizing an ensemble of n distinct tool agents $E = \{p_{\theta_i}\}_{i=1}^n$, we initialize the system with multiple **exploratory heads** distributed across the solution space.

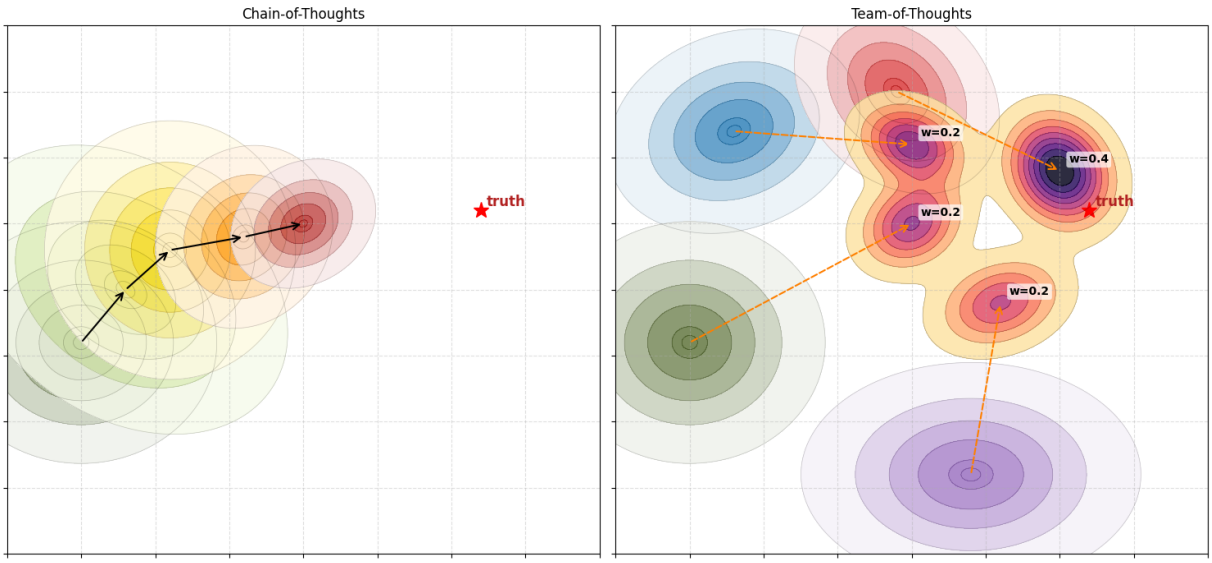

Figure 3: **Schematic comparison of Chain-of-Thought (CoT) and Team-of-Thoughts. (Left)** Standard agentic reasoning via CoT generates sequential intermediate steps to refine the prediction distribution heuristically. However, on complex tasks, CoT may fail to converge on distant targets. **(Right)** Team-of-Thoughts utilizes heterogeneous tool agents to explore a broader solution space. During inference, these agents refine their individual distributions through local reasoning, which an orchestrator then prioritizes and aggregates into the final output.

As shown in Figure 3 (right), the orchestrator agent p_{orch} selectively invokes these agents and integrates their reasoning trajectories $\mathbf{X}^{(i)}$ into its global context. From a probabilistic perspective, the resulting distribution formed by the orchestrator can be modeled as a **Gaussian Mixture Model (GMM)**:

$$p_{\text{orch}}\left(\cdot \mid D, \left\{\mathbf{X}^{(i)}\right\}_{i=1}^{n}\right) \approx \sum_{i=1}^{n} \pi_i p_{\theta_i}\left(\cdot \mid D, \mathbf{X}^{(i)}\right)$$

where π_i represents the mixture weights. In our framework, these weights are not explicitly defined scalars but are implicitly computed via the orchestrator's cross-attention mechanism. By attending to the tool agents' latent representations, the orchestrator dynamically assigns importance to each agent's contribution, effectively producing a weighted mixture of individual distributions.

The Efficiency of Distributional Jumps

The GMM serves as a conceptual framework to explain why Team-of-Thoughts is more efficient than single-model "linear drift." Rather than relying on a single model to slowly navigate toward the target, the orchestrator can effectively "jump" to high-probability regions already localized by specialized tool agents.

This formulation contains two practical nuances:

- The mixture weights π_i are typically sparse, as the orchestrator identifies and invokes only a high-confidence subset of $k \leq n$ agents based on the self-assessment profiles described in Section 3.3.
- The orchestrator conditions its generation on the discrete responses $\mathbf{X}^{(i)}$ of the tools, using them as anchors to refine the synthesized distribution into a definitive, high-confidence final answer.

This synthesis allows Team-of-Thoughts to maximize capability coverage while maintaining significantly higher token efficiency than exhaustive sampling or consensus-based search.

3.2 Dynamic Orchestration Calibration

A model's proficiency as a standalone solver does not inherently correlate with its efficacy as a coordinator. Effective orchestration requires a distinct set of capabilities, including tool-schema comprehension, multi-trajectory synthesis, and conflict resolution. To identify the optimal "manager" for a given domain, we introduce an **Orchestration Calibration** procedure.

We evaluate candidate orchestrators based on their empirical ability to aggregate tool-agent responses within a specific task category c. For each candi-

date model θ_{cand}, we compute a calibration score:

$$\text{Score}(\theta_{\text{cand}}, c) = \frac{\sum_{D \in \mathcal{D}_{\text{val}}^{(c)}} \mathbb{I}\left[\hat{X}_{\text{cali}}(D, \theta_{\text{cand}}) = X_{\text{cali}}(D)\right]}{\left|\mathcal{D}_{\text{val}}^{(c)}\right|} \quad (1)$$

where $\mathcal{D}_{\text{val}}^{(c)}$ is a category-specific validation set, $X_{\text{gt}}(D)$ is the ground-truth answer, and $\hat{X}_{\text{agg}}$ is the prediction generated by the orchestrator after processing the available tool-agent outputs.

By treating orchestrator selection as a distinct optimization problem, we ensure that the "central node" of our GMM is specialized for distributional synthesis rather than just raw generation. This allows the system to remain robust even when the underlying tool agents provide divergent or noisy reasoning paths.

3.3 Contextual Agent Profiling via Self-Assessment

To leverage the heterogeneous priors of diverse model families, the orchestrator must understand the specific expertise of its available tools. We formalize this through a **Self-Assessment Mechanism**, which translates raw model parameters into a "capability profile" through three stages:

1. **Empirical Trajectory Profiling:** Each tool agent θ_i generates reasoning traces and predictions on a representative validation set: $\hat{X}_i \sim p_{\theta_i}(\cdot \mid D, \mathbf{X}^{(i)})$.
2. **Performance Quantification:** The system computes an objective accuracy score $s_i^{(c)}$ for each category c, identifying where the agent is empirically reliable or prone to failure.
3. **Qualitative Characterization:** In a final "meta-reflection" step, the agent examines its own performance data to generate a summary of its strengths and weaknesses.

These profiles are injected into the orchestrator's context during inference. This mechanism transforms the system from a "dense" MAS (where all agents are invoked) into a **sparse, priority-based architecture**. By matching the semantic requirements of a query D against the agents' proficiency profiles, the orchestrator selectively activates only the most compatible tools. This strategic allocation of the reasoning budget ensures that the system maximizes capability coverage while minimizing the redundant token consumption typical of homogeneous ensembles.

4 Experiments

Models and Benchmarks We evaluate **Team-of-Thoughts** using an ensemble of seven heterogeneous model families: Claude 4.5 Sonnet, GPT-5 Mini, Gemini 3 Flash, DeepSeek-V3.2, GPT-OSS-20B, Qwen3-VL-235B, and Phi-4. Our evaluation spans mathematical reasoning (AIME 2024/2025) and code generation (HumanEval+, MBPP+, and LiveCodeBench v6).

Calibration and Constraints To profile agent expertise, we utilize a 10% random sample for most benchmarks. For the AIME benchmarks, we employ a **cross-calibration** strategy to mitigate the limited sample size: 50% of AIME 2025 serves as the calibration set for AIME 2024, and vice-versa. This ensures robust profiling while strictly preventing test-set contamination. Unless otherwise specified, we activate $k = 2$ tool agents per query.

4.1 Comparative Performance Evaluation

The Team-of-Thoughts framework integrates tool agents via descriptions derived from their self-assessment profiles. Guided by the calibration results in Section 4.2, we utilize DeepSeek-V3.2 as the orchestrator for math and GPT-5 Mini for code.

As detailed in Table 1, Team-of-Thoughts demonstrates robust performance across both domains. On mathematical benchmarks, it achieves superior stability compared to standalone models, consistently reaching higher **pass@1** accuracy while remaining competitive at higher sample counts (**pass@3/5**). In code generation, Team-of-Thoughts secures state-of-the-art results on MBPP+. On HumanEval+, while it trails Claude 4.5 Sonnet marginally at pass@1—a metric increasingly constrained by benchmark saturation—it demonstrates superior scaling, surpassing Claude as the number of allowed generations increases.

For LiveCodeBench, Team-of-Thoughts achieves peak pass@1 performance but slightly lags behind the strongest single-model baseline at pass@3. We attribute this to **calibration sensitivity** in high-diversity tasks; a taxonomic analysis (Appendix C.1) confirms that LiveCodeBench is significantly more diverse than HumanEval+, leading to sparse category coverage during limited-sample

Table 1: **Performance comparison of Team-of-Thoughts across five benchmark tasks.** Success rates (%) are reported using pass@1, 3, and 5. Team-of-Thoughts is evaluated against individual model baselines and state-of-the-art multi-agent methods. The *Theoretical Limit* represents the oracle upper bound achieved by selecting the best result among all individual models for each problem.

	AIME 2024			AIME 2025			Humaneval+			MBPP+			LiveCodeBench v6		
Model	pass@1	@3	@5	@1	@3	@5	@1	@3	@5	@1	@3	@5	@1	@3	@5
Claude Sonnet 4.5	84.00	86.67	86.67	74.67	87.67	90.00	**95.61**	96.34	96.34	81.80	84.29	85.45	48.24	55.33	57.69
GPT-5-mini	88.67	93.33	93.33	73.33	83.50	86.11	92.68	94.88	95.12	82.38	86.19	87.57	64.84	74.40	77.20
Gemini 3 Flash	86.67	90.67	93.33	71.33	81.33	86.67	96.46	96.95	96.95	84.44	85.13	85.45	72.53	75.71	77.47
Deepseek v3.2 Exp	94.67	**96.67**	**96.67**	94.00	98.67	**100.00**	91.26	95.88	96.65	80.20	84.59	85.83	76.48	**86.65**	**88.46**
GPT OSS 20b	75.33	88.00	90.00	73.33	83.50	86.11	89.23	95.12	95.63	79.23	84.88	86.29	46.04	54.34	58.24
Qwen3-vl 235B	93.33	93.33	93.33	89.44	91.67	92.78	93.19	94.70	95.02	83.20	86.24	87.04	67.47	75.77	78.57
Phi-4	13.33	18.00	20.00	10.67	15.67	16.67	74.80	80.55	82.83	63.07	72.51	74.60	24.84	27.69	28.02
Theoretical Limit	96.67	–	–	100.00	–	–	98.17	–	–	93.65	–	–	90.66	–	–
Majority Voting	93.33	–	–	93.33	–	–	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
AgentVerse	80.00	–	–	60.00	–	–	91.46	–	–	79.37	–	–	65.93	–	–
Team-of-Thoughts	**96.00**	**96.67**	**96.67**	**95.33**	**99.33**	**100.00**	95.33	**97.47**	**98.07**	**85.93**	**88.84**	**89.68**	**77.91**	85.05	87.91

Table 2: **Orchestration performance across candidate models and budget constraints.** Calibration accuracy (%) is evaluated on AIME2024 and MBPP+ benchmarks under two distinct budgetary tiers (USD). Avg denotes the mean accuracy across all evaluated settings.

Model	**AIME2024**			**MBPP+**		
Budget Cost ($)	0.03	0.02	Avg	0.03	0.02	Avg
Claude Sonnet 4.5	86.67	46.67	66.67	78.38	78.38	78.38
GPT-5 Mini	86.67	93.33	90.00	86.49	83.78	**85.14**
Gemini 3 Flash	86.67	40.00	63.34	83.78	83.78	83.78
DeepSeek v3.2	93.33	93.33	**93.33**	81.08	81.08	81.08
GPT-OSS 20B	80.00	80.00	80.00	75.68	75.68	75.68
Qwen3-vl 235B	33.33	20.00	26.67	78.38	78.38	78.38
Phi-4	33.33	33.33	33.33	72.97	72.97	72.97

calibration. We find that increasing calibration density or employing categorized sampling effectively resolves this variance (Appendix C.2).

Overall, these results validate the probabilistic intuition behind Team-of-Thoughts: by leveraging heterogeneous agents to cover a broader solution space, the orchestrator successfully "jumps" to high-probability regions localized by specialized tools.

4.2 Orchestrator Calibration and Selection

To identify the optimal coordinator, we evaluate candidate models under a **fixed monetary budget**, normalizing costs into model-specific token limits to ensure a fair comparison.

As shown in Table 2, orchestration proficiency is highly task-dependent: DeepSeek-V3.2 dominates the mathematical domain, whereas GPT-5 Mini excels in code synthesis. These results suggest that effective orchestration is decoupled from raw model scale or general benchmark rankings. Instead, it relies on **integrative reasoning efficiency**: the ability to synthesize disparate trajectories into a cohesive whole. Consequently, we treat orchestrator selection as a domain-specific optimization problem rather than defaulting to the largest available model.

4.3 Ablation: Agent Profiling and Selection Policies

To optimize agent invocation, we compare three distinct policies for constructing and utilizing capability profiles:

- **Random Invocation (Baseline):** Tool models are sampled uniformly without prior profiling or weighting.
- **Orchestrator-Led Assessment:** A centralized model (DeepSeek-V3.2 for math; GPT-5 Mini for code) evaluates agents on a calibration set to map their core competencies and failure modes.
- **Agent Self-Assessment:** Each tool agent performs a structured self-audit. Provided with its own reasoning traces and ground truth, the agent critiques its own performance and identifies specific skill requirements.

As shown in Table 3, all structured policies significantly outperform the single-agent baseline. Notably, the **self-assessment** policy achieves the highest overall accuracy. We posit that self-assessment is superior because it **decouples capability profiling from the orchestrator's internal biases**, providing a more objective representation of agent expertise. Based on these findings, we adopt self-assessment as our default strategy.

In Appendix B, we present controlled ablations that

Table 3: **Performance evaluation of tool agent assessment strategies.** Comparison of Pass@1 accuracy (%) with no tool agent ("Single"), Orchestrator-Led Assessment, Agent Self-Assessment, and Random Invocation across AIME2024 and MBPP+ benchmarks.

Benchmark	Single	Self-Assess	Orch-Based	Random
AIME2024	94.67	**96.00**	92.67	94.00
MBPP+	82.38	**85.93**	85.77	84.44

Table 4: **Performance of orchestrator aggregation strategies.** Comparison of task accuracy (%) achieved when incorporating versus excluding the intermediate reasoning traces of tool agents.

	AIME 2025			MBPP+		
Config	pass@1	@3	@5	pass@1	@3	@5
No traces	**95.33**	**99.33**	**100.00**	**85.93**	**88.84**	**89.68**
Include traces	78.67	93.33	96.67	80.90	85.61	87.04

disentangle the individual contributions of model diversity, calibration, and routing to overall system performance.

4.4 The Impact of Information Density on Aggregation

We evaluate two protocols for information exchange between tool agents and the orchestrator: (1) returning only the final solution $\hat{X}_i$, or (2) providing the full reasoning trajectory $\mathbf{Z}^{(i)}$.

As summarized in Table 4, incorporating full reasoning trajectories **consistently degrades performance**. This drop suggests that verbose trajectories introduce significant **contextual noise**, distracting the orchestrator with intermediate logical errors or irrelevant sub-steps. Our results demonstrate that for multi-agent synthesis, "minimalist" tool responses are more effective. By filtering out fallible intermediate steps, the orchestrator maintains a cleaner global context, leading to more robust decision-making.

4.5 Scaling Limits and the Orchestration Bottleneck

We investigate whether performance scales monotonically with ensemble size ($|E|$) and activation density (k). We expanded our ensemble to include the 100 most popular models on OpenRouter.

Activation Density Fixing the ensemble at $|E| = 25$, we varied the number of activated models k from 2 to 16. Table 5 shows that increasing k yields negligible gains and eventually causes **performance decay**, with Pass@1 accuracy dropping significantly at $k = 8$. While higher activation density increases run-time variance (improving pass@5), it simultaneously compromises the robustness of the primary prediction.

Table 5: Performance under different numbers of activated tool agents, with 25 total tool agents available.

Activated tools	Pass@1	Pass@3	Pass@5
2	82.75%	85.58%	86.77%
4	82.96%	85.32%	85.98%
8	60.32%	83.76%	86.24%
16	60.63%	83.10%	85.71%

Table 6: Orchestrator performance and tool selection consistency (Agreement) across varying ensemble sizes. Performance is compared between configurations activating two and eight agents per tool call.

	Activate 2			Activate 8			Agreement
Avail Tools	Pass@1	@3	@5	Pass@1	@3	@5	$\mathcal{A}$ (%)
10	82.70	86.32	87.57	74.97	85.26	85.98	90.74
25	82.75	85.58	86.77	60.32	83.76	86.24	91.80
50	82.43	86.59	87.83	54.76	82.17	86.51	61.64
75	82.49	87.09	88.36	63.65	84.21	86.24	75.13
100	82.49	86.56	87.57	61.43	84.13	86.51	71.43

Ensemble Breadth and Selection Overload We further analyze how the orchestrator handles a broader pool of candidates. Sweeping over ensemble sizes for $k = 2$ and $k = 8$ (Table 6) reveals that $k = 8$ exhibits a sharp performance decline as $|E|$ increases. To isolate the cause, we define **Selection Agreement**:

$$\mathcal{A} = \frac{1}{|D|} \sum_{i=1}^{|D|} \mathbb{I}[S_2^{(i)} \subseteq S_8^{(i)}]$$

where $S_k^{(i)}$ is the set of tool agents invoked for query i at level k. Agreement sharply declines once $|E| > 50$, suggesting a **selection logic overload**: the orchestrator is overwhelmed by excessive options, leading to the invocation of suboptimal agents.

Our findings suggest an **orchestration bottleneck**: scaling both activation density and ensemble breadth eventually degrades performance by overwhelming the orchestrator's synthesis logic. Conversely, with a calibrated activation count (e.g., $k = 2$), the orchestrator can reliably extract high-utility responses from an arbitrarily large ensemble.

5 Related Work

Test-Time Scaling (TTS) TTS establishes that increasing inference-time computation can yield performance gains comparable to scaling model parameters (Snell et al., 2024; Wu et al., 2025). Approaches generally follow two trajectories: (1) **parallel sampling**, such as Best-of-N (Brown et al., 2024), which uses verification to select optimal outputs from a broad search space; and (2) **sequential reasoning**, which deepens the reasoning topology. This includes linear Chain-of-Thought (CoT) (Wei et al., 2023) and non-linear frameworks like Tree of Thoughts (ToT) (Yao et al., 2023) and Graph of Thoughts (GoT) (Besta et al., 2024). These methodologies enable complex operations like backtracking and information aggregation, mimicking human "System 2" cognition.

Multi-Agent Systems (MAS) Inspired by human dynamics, MAS utilizes agent ensembles to enhance task reliability and role-playing capabilities (Park et al., 2023; Li et al., 2023). Early frameworks leveraged conversational flows (Wu et al., 2023) or Standard Operating Procedures (SOPs) to reduce hallucinations in software engineering (Qian et al., 2024; Hong et al., 2024). More recent research emphasizes dynamic workflows; for instance, AgentVerse (Chen et al., 2024) introduces "expert recruitment," while AgentNet (Yang et al., 2025b) employs autonomous graph topologies for task decomposition. Theoretically, the "Society of Thoughts" (Kim et al., 2026) suggests that even single-model reasoning benefits from simulating diverse internal perspectives.

Diversity-Aware MAS A growing line of work incorporates the value of model diversity. Socratic Models (Zeng et al., 2022), Mindstorms (Zhuge et al., 2023), and Mixture-of-Agents (Wang et al., 2024) combine diverse models but rely on fixed pipelines, lacking principled agent selection. The Conductor (Nielsen et al., 2025) and TRINITY (Xu et al., 2025) address this by training dedicated coordinators via reinforcement learning and evolutionary strategies, respectively.

In contrast, our **Team-of-Thoughts** framework exploits **heterogeneous** model diversity while remaining entirely training-free. It uses lightweight inference-time calibration to profile agents and the standard tool-calling paradigm to dynamically select and aggregate them, making it immediately adaptable to new model pools and tasks.

6 Conclusion

We propose the Team-of-Thoughts framework, which explicitly leverages the skill diversity of a group of heterogeneous agent models. By dynamically selecting the most suitable orchestrator and making skill-dependent tool calls that can be executed in parallel, Team-of-Thoughts overcomes the limitations of fixed-role, sequential multi-agent systems. Our empirical results across reasoning and code generation tasks demonstrate that this approach consistently achieves more efficient cost usage and higher accuracy compared to both single-model baselines and homogeneous multi-agent systems, highlighting a novel paradigm for heterogeneous models to collaborate, paving the way for future exploration of multi-round, complex multi-agent tasks.

Limitations

The proposed Team-of-Thoughts framework relies on a central LLM as an orchestrator, which introduces a performance bottleneck as the agent pool scales. In Section 4.5, we characterize the limitations of this centralized architecture when scaled to 100 tool-agents. Specifically, we observe two primary failure modes: The orchestrator often fails to navigate expansive tool selection spaces, leading to suboptimal tool-calling sequences. Large-scale tool orchestration induces high variance in output quality as shown in pass@1 performance, suggesting a lack of robustness in the model's iterative reasoning process under high context load. These results suggest that the scalability of the system is currently constrained by the state of frontier models. Future advancements in long-context reasoning and hierarchical information aggregation will be essential to fully realize the potential of massive tool-agent ensembles.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Anthropic. 2026. Claude code agent teams documentation. https://code.claude.com/docs/en/agent-teams. Accessed: 2026-03-15.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten

Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *Preprint*, arXiv:2407.21787.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pages 20094–20136.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Tianxiang Gao, Xiaokai Huo, Hailiang Liu, and Hongyang Gao. 2023. Wide neural networks as gaussian processes: Lessons from deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 36, pages 54918–54951. Curran Associates, Inc.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. *Preprint*, arXiv:2308.00352.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *Preprint*, arXiv:2403.07974.

Junsol Kim, Shiyang Lai, Nino Scherrer, Blaise Agüera y Arcas, and James Evans. 2026. Reasoning models generate societies of thought. *Preprint*, arXiv:2601.10825.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pages 51991–52008. Curran Associates, Inc.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, Hongxuan Lu, Tianrui Qin, Chenghao Zhu, Yi Yao, Shuying Fan, Xiaowan Li, Tiannan Wang, Pai Liu, King Zhu, and 11 others. 2025. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl. *Preprint*, arXiv:2508.13167.

Stefan Nielsen, Edoardo Cetin, Peter Schwendeman, Qi Sun, Jinglue Xu, and Yujin Tang. 2025. Learning to orchestrate agents in natural language with the conductor. *arXiv preprint arXiv:2512.04388*.

Mathematical Association of America. 2024. AIME 2024.

Mathematical Association of America. 2025. AIME 2025.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA. Association for Computing Machinery.

Anthropic PBC. 2025. Introducing claude sonnet 4.5.

Sundar Pichai, Demis Hassabis, and Koray Kavukcuoglu. 2025. A new era of intelligence with gemini 3.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand. Association for Computational Linguistics.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, and 465

others. 2025. Openai gpt-5 system card. *Preprint*, arXiv:2601.03267.

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

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024. Mixture-of-agents enhances large language model capabilities. *Preprint*, arXiv:2406.04692.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *Preprint*, arXiv:2408.00724.

xAI. 2026. Grok (version 4.2 beta 0309). `https://x.ai`. Accessed: 2026-03-15.

Jinglue Xu, Qi Sun, Peter Schwendeman, Stefan Nielsen, Edoardo Cetin, and Yujin Tang. 2025. Trinity: An evolved llm coordinator. *arXiv preprint arXiv:2512.04695*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. 2025b. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems. *Preprint*, arXiv:2504.00587.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, and 1 others. 2022. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö. Arı k. 2024. Chain of agents: Large language models collaborating on long-context tasks. In *Advances in Neural Information Processing Systems*, volume 37, pages 132208–132237. Curran Associates, Inc.

Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, and 1 others. 2023. Mindstorms in natural language-based societies of mind. *arXiv preprint arXiv:2305.17066*.

A Experiment Setup Details

A.1 Additional Experiment Setup

Main experiment setting We evaluated Team-of-Thoughts MAS across a diverse suite of seven model families, comprising three closed-source models: Claude-Sonnet-4.5 (PBC, 2025), GPT-5-mini (Singh et al., 2025), and Gemini-3-Flash-Preview (Pichai et al., 2025) and four open-source models: DeepSeek-V3.2-Exp (DeepSeek-AI et al., 2025), GPT-OSS-20B (OpenAI et al., 2025), Qwen3-VL-235B-A22B-Thinking (Yang et al., 2025a), and Phi-4 (Abdin et al., 2024). Two tool agents are activated on each tool call.

Our assessment spanned two domains: mathematical reasoning (AIME2024 (of America, 2024), AIME2025 (of America, 2025)) and code generation (Humaneval+ (Chen et al., 2021), MBPP+ (Austin et al., 2021), and LiveCodeBench v6 (Jain et al., 2024) with problems released between 2025/01/01 and 2025/05/01). Unless stated otherwise, we set a standardized context window for each tool-agent: 20,000 tokens for AIME tasks and 4,096 tokens for coding tasks. For the orchestrator, we used a 16,384 token context window across all tasks to ensure sufficient capacity for processing tool descriptions and making informed selection and reasoning decisions. For reasoning models, we applied the default "medium" effort setting, capping the reasoning token budget at 50% of the maximum generation length to ensure consistent comparisons across all baselines.

By default, we randomly sample 10% of the target tasks to construct a calibration dataset. However, for AIME2024 and AIME2025, 10% corresponds to only three data points, which is too small to provide a reliable characteristic modeling. We instead sample 50% of the problems from AIME2025 and use them as the calibration dataset for evaluating AIME2024 and vice versa for AIME2025. This cross-calibration setup ensures sufficient calibration data while avoiding exposure of the test data.

Orchestration selection In the orchestration agent selection experiment, we judge the performance of the orchestrator by activating all tool agents. The max generation token is set based on each agent's cost, ensuring it will not exceed the cost budgets.

Scaling tool agents experiment setting For this analysis, we select the top 100 OpenRouter models ranked by popularity that provide a context length greater than 16K tokens and cost less than $1 USD per million input tokens. The orchestrator context window is set as 16,384 unless stated otherwise. Due to varying max context lengths of different models, we employ Orchestrator-Led Assessment using GPT-5 Mini. The orchestrator is provided with the tool agents with top single-model performance on MBPP+ when a subset of the models is made available.

A.2 AgentVerse Setup

We employ GPT-5-Mini (Singh et al., 2025) as the backbone language model of agents in AgentVerse (Chen et al., 2024). We use a maximum token limit of 512 for the role assigner agent, and 4,096 for the rest of the agents. For invalid agent outputs, such as invalid role assignments, unparseable answers, or errors in code, AgentVerse retries generation a limited number of times: 10 times on math tasks and 1,000 times on coding tasks.

B Disentangling Performance Gains: Diversity, Calibration, and Routing

To isolate the individual contributions of model diversity, calibration, and routing to overall system performance, we conduct controlled ablation experiments on MBPP+. All experiments share the same inference budget as detailed (number of tool calls, backend token budget, and orchestration token budget) in Section A.1.

B.1 Diversity: Single-Family to Multi-Family Backends

We evaluate the Team-of-Thoughts framework under three backend configurations—a single model family, two families, and the full Team-of-Thoughts setup—while holding all other factors constant.

Table 7 shows that as backend diversity increases, the fraction of questions that all backends answer incorrectly decreases, and the theoretical upper bound on system performance rises accordingly. The gap between this upper bound and actual performance reflects the quality of calibration and orchestration aggregation.

B.2 Calibration: Self-Assessment vs. Random Routing

To isolate the contribution of calibration, we reuse the same backend execution traces so that the

Table 7: Performance scaling with increasing backend diversity on MBPP+. All settings use a fixed number of tool calls and token budgets.

	GPT-5-mini only	GPT-5-mini + Gemini	Team-of-Thoughts
All backends wrong	47	38	24
Theoretical Limit[†] (%)	87.57%	89.90%	93.65%
MBPP+ performance	86.24%	87.83%	89.68%

[†] Fraction of questions where at least one backend produces a correct answer.

theoretical upper bound is identical across conditions. Any gain from self-assessment is therefore attributable entirely to routing quality.

Results in Table 8 show that the primary benefit of calibration lies in the *some backends wrong* category—cases where the orchestrator must select among a mix of tools that produce correct and incorrect answers. Here, self-assessment yields a +15.3% improvement over random routing.

Table 8: Performance comparison between random sampling and self-assessment routing on MBPP+, using identical backend traces.

Category	Random	Self-assessment	Gain
All backends correct (256)	251 (98.0%)	256 (100%)	+5
Some backends wrong (98)	68 (69.4%)	83 (84.7%)	+15.3%
MBPP+ performance	84.39%	89.68%	+5.29%

B.3 Per-Backend Call Distribution

Self-assessment enables strategic call allocation across backends. As shown in Table 9, calibrated routing almost completely avoids the weakest backend (Phi-4, 75% accuracy) on MBPP+ tasks, while heavily favouring the strongest backend (Backend 6, 86% accuracy). Random routing, by contrast, distributes calls uniformly and invokes the weak model approximately one-third of the time.

Table 9: Per-backend call distribution under random vs. calibrated routing on MBPP+.

Backend	Random calls	Calibrated calls
Backend 3 (Phi-4, 75% acc.)	239	6
Backend 6 (DS, 86% acc.)	229	366
Backend 5 (GPT-5-mini, 88% acc.)	144	163
Remaining backends	174	221

C Impact of Calibration Dataset Choice

The calibration scheme is more sensitive on LiveCodeBench than on other benchmarks. We investigate the source of this sensitivity and evaluate two mitigation strategies: categorized sampling and increased sample size.

C.1 Calibration Sensitivity and Taxonomic Diversity

We hypothesise that the sensitivity stems from LiveCodeBench's higher taxonomic diversity relative to HumanEval+. With limited calibration samples, greater label diversity leads to sparse category coverage and higher estimation variance.

To verify this, we constructed a data-driven taxonomy independently for each benchmark. We sampled problems in batches of 20 and used GPT-4o-mini to tag each along four dimensions: algorithm techniques, data structures, problem domain, and problem type. Sampling continued until fewer than two new tags were discovered per round, ensuring saturation. We then grouped similar labels using GPT-5-mini and computed per-dimension coverage across both datasets.

Table 10: Taxonomic coverage across HumanEval+ and LiveCodeBench. LiveCodeBench exhibits substantially higher diversity along all four dimensions.

Dimension	HumanEval+	LiveCodeBench
Algorithm techniques	59%	76%
Data structures	41%	59%
Problem domain	64%	70%
Problem type	67%	89%

As shown in Table 10, LiveCodeBench is significantly more diverse along all four dimensions, with 22 percentage points more diversity in problem type than HumanEval+, supporting the hypothesis that calibration sensitivity correlates with taxonomic diversity.

C.2 Categorized Calibration

Motivated by the diversity gap, we construct a categorized calibration set that ensures coverage across problem types. We randomly sampled 10% of LiveCodeBench v6 (18 questions), used GPT-5 to group them by primary algorithmic concept (e.g., Arrays / Data Structures, Strings / Sequences, Math / Logic, Grid / Matrix, Simulation / Greedy, Knapsack / Optimization, Intervals / Range), and selected one representative question per category.

Table 11: Performance in pass@k on LiveCodeBench v6 using different calibration construction strategies.

Construction Method	pass@1	pass@3	pass@5
Random Sample	77.91%	85.05%	87.91%
Categorized Sample	81.65%	85.93%	86.81%

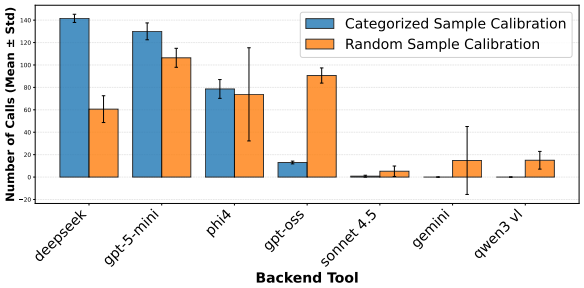

Figure 4: Comparing the times of tool agent invocations between two calibration dataset construction methods.

As shown in Table 11, the categorized calibration dataset leads to a noticeable improvement in pass@1 accuracy. Figure 4 compares the distribution of tool selections under the two strategies. With the categorized set, the system selects the strongest single model approximately twice as often, explaining the improved pass@1.

However, the categorized set also reduces selection diversity, resulting in slightly lower pass@5 than the random set. This suggests a bias–variance trade-off: categorized calibration biases toward stronger single-model decisions but reduces exploration across tools.

C.3 Effect of Calibration Sample Size

As an alternative to categorized sampling, we show that simply increasing the calibration sampling fraction also resolves the instability.

Table 12: LiveCodeBench performance as a function of calibration sample size. Pass@1 performance improves steadily up to a fraction of 20%.

Sampling fraction	0.05	0.10	0.15	0.20	0.30
Performance	76.37%	76.92%	79.67%	80.21%	79.67%

Table 12 confirms that the sensitivity is a coverage effect: performance improves steadily as more samples are randomly drawn, plateauing around a fraction of 20%. Beyond this point, additional samples provide diminishing returns as the calibration set already covers most taxonomic categories.

D Longer Context Window with 100-Tool Agent Ensemble

To verify whether the performance decline with eight activated agents stems from context window limitations, we evaluated the 25-model ensemble using extended orchestration contexts. As shown in Table 13, increasing context length yielded no performance gains. This confirms that the bottleneck resides in the orchestrator's aggregation capability rather than spatial context constraints.

Table 13: Performance under different orchestrator max context lengths, with eight of 25 available tool agents activated on each tool call.

Orch context len	Pass@1	Pass@3	Pass@5
16K	60.32%	83.76%	86.24%
32K	59.79%	82.99%	85.71%
64K	59.42%	83.39%	86.51%
128K	60.32%	82.96%	85.45%

E Experiment Prompt

Here is an example of the language model-based tool agent assessment prompts.

Assessment Prompt

Instructions: You will be provided with a series of problems. For each problem, the Subject Agent has provided an answer. Some are correct, and some are incorrect.

Part 1: Per-Problem Analysis

For every problem provided, generate a structured audit containing:

1. **Taxonomy:** Classify the problem type (e.g., Arithmetic, Logical Reasoning, Creative Writing, Coding) and specific skill required.
2. **Performance Verdict:** Clearly state [PASS] or [FAIL].
3. **Gap Analysis:**
   - *If Correct:* Briefly explain why the agent succeeded (e.g., "Good step-by-step reasoning," "Robust knowledge retrieval").
   - *If Incorrect:* Pinpoint exactly *where* and *why* the agent failed. Was it a calculation error? A logic jump? A hallucination? A misunderstanding of constraints? Compare the Subject's logic to the Ground Truth.

Part 2: Executive Summary

After analyzing all problems, synthesize a "Model Persona" profile:

1. **Core Competencies:** List specific categories where the agent consistently succeeds.

2. **Blind Spots & Failure Modes:** Describe the patterns in the agent's errors (e.g., "The agent struggles with negative integers," or "The agent is verbose but inaccurate").
3. **Final Verdict:** A 2-sentence summary of the agent's reliability.

—

Input Data:

- Problem 1: *Question, Subject Agent Answer, Ground Truth/Solution...*
- Problem 2: *...repeat for all samples...*

—

COMMAND:
Based on the data above, proceed with the **Per-Problem Analysis** followed by the **Executive Summary**.
Reminders:

1. **Be Specific:** Do not just say "The agent failed." Identify *why* (e.g., Logic Error vs. Calculation Error).
2. **Be Critical:** Compare the Subject Answer against the Ground Truth rigorously.
3. **Format:** Use the headers `## Part 1: Per-Problem Analysis` and `## Part 2: Executive Summary`.

GENERATE REPORT: