Answer Probing-Guided Search for Diverse Solution Exploration of LLMs

Yi Fang[1,2], Que Shen[3], Chengpeng Li[3], Boyi Deng[1], Wei Shi[4], Wenjie Wang[1*], Fuli Feng[1], Fengli Xu[2,5*], Dayiheng Liu[3]

[1]University of Science and Technology of China [2]Zhongguancun Academy
[3]Alibaba Group [4]Shanghai Jiao Tong University [5]Tsinghua University

Abstract

Generating multiple diverse and high-quality solutions is valuable for many applications, such as code-test generation and drug discovery. However, Large Language Models (LLMs) tend to converge on a single high-confidence solution during inference, limiting exploration of alternative valid solution paths. Existing test-time methods promote diversity through tree-like search and prune semantically similar branches using response-level semantic embeddings. However, we find that such embeddings are easily confounded by linguistic and stylistic similarities, making it difficult to distinguish genuinely distinct solution paths. To address this, we introduce **Answer Probing**, which probes the potential answer an LLM would reach from an intermediate reasoning path. We demonstrate that the hidden states of probed answers more effectively differentiate distinct solution paths than semantic embeddings, and the perplexity of probed answers serves as a practical proxy for reasoning correctness. Based on these findings, we propose **A**nswer **P**robing-Guided **T**ree **S**earch (APTS), which guides the tree search by the probed answers' hidden state similarity and perplexity. Experiments on three reasoning tasks across two LLMs show that APTS consistently enhances solution diversity, demonstrating its effectiveness and robustness.

1 Introduction

Diverse solution generation requires Large Language Models (LLMs) to explore various solution paths and produce multiple high-quality solution candidates. This capability is highly valuable in real-world tasks where a set of high-quality candidates is needed rather than a single gold standard answer. For example, in code-test generation (Jain et al., 2025), diverse test cases enable more comprehensive software evaluation. In drug discovery (Drews, 2000), diverse candidates provide a broader range of chemical structures, helping experts identify novel drug leads. However, during inference, current LLMs tend to converge on a single high-confidence trajectory when sampling multiple times (Song et al., 2025; Jiang et al., 2025), limiting their ability to explore other valid solutions. To address this issue, it is important to develop methods that encourage LLMs to explore more diverse yet valid reasoning paths.

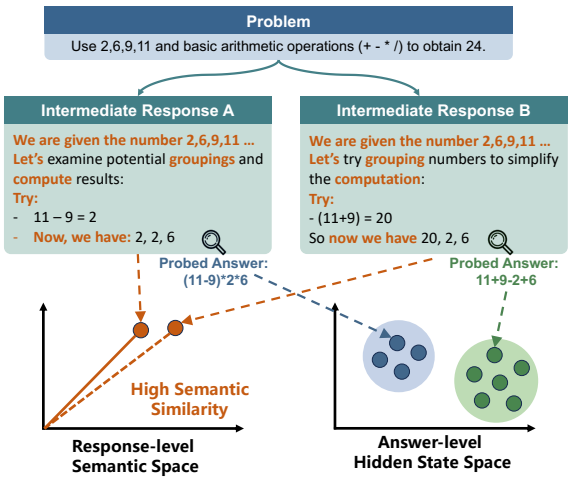


Figure 1: Illustration of Answer Probing. Compared with semantic embeddings, answer probing mitigates linguistic and stylistic noise in reasoning and better distinguishes different solution paths.

To improve reasoning diversity without incurring prohibitive training costs, several test-time methods have been explored. These approaches can be broadly categorized into decoding-based and search-based methods. Decoding-based methods, such as temperature sampling (Renze, 2024) and nucleus sampling (Holtzman et al., 2020), enhance diversity by introducing probabilistic perturbations into the token generation process. However, these methods often yield token-level variation without true semantic diversity: the generated reasoning paths may exhibit different token sequences but still reflect similar lines of reasoning (Yun et al.,

* Corresponding authors.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 31571–31587
October 24-29, 2026 ©2026 Association for Computational Linguistics

2025). Search-based methods address this limitation by leveraging semantic embeddings to guide the generation process (Shi et al., 2025; Vijayakumar et al., 2018). Specifically, they employ search algorithms such as beam search (Li et al., 2016) and Tree of Thoughts (Yao et al., 2023) to explore multiple candidate paths at each reasoning step. These paths are then pruned based on their semantic embedding similarity, retaining only semantically diverse branches for further reasoning, thereby improving the diversity of the final reasoning set. However, LLM responses to the same problem frequently exhibit high structural and stylistic homogeneity (Kirk et al., 2024; O'Mahony et al., 2024). As illustrated in Figure 1, shared reasoning signposts and surface patterns can induce high similarity in semantic space, reducing the effectiveness of semantic embeddings in identifying truly distinct solution paths (see detailed analysis in Section 2).

To address this limitation, we seek an alternative representation that better reflects underlying solution paths while being less affected by linguistic and stylistic noise. Our preliminary analysis (Section 2) shows that answer-level hidden states are more effective than response-level semantic embeddings at distinguishing divergent solution paths. Motivated by this finding, we propose **Answer Probing**: at each reasoning step, we probe the potential answers the LLM would reach from the current intermediate reasoning path. This process exposes the latent solution tendency encoded in the intermediate reasoning path. By representing candidate reasoning paths with the mean-pooled hidden states of probed answers, we can better distinguish paths leading to distinct solutions. We also find that the perplexity (PPL) of probed answers provides a useful proxy for the quality of a candidate reasoning path, with lower PPL generally associated with more accurate outcomes.

Based on these findings, we propose **A**nswer **P**robing-Guided **T**ree **S**earch (APTS), which improves solution diversity while maintaining solution quality during generation. At each tree search step, APTS performs answer probing to evaluate candidate nodes. It uses the PPL of the probed answer as a quality signal and the hidden state similarity between probed answers as a diversity signal. By selecting nodes that are both high quality and non-redundant for further expansion, APTS guides the search toward candidate paths that are both promising and diverse. Extensive evaluations on three cross-domain reasoning tasks and two LLMs show that APTS improves solution diversity with only a modest impact on accuracy, demonstrating its effectiveness and robustness. In summary, the contributions of this work are threefold:

- We propose **Answer Probing**, which probes intermediate answers to yield both a quality signal and a diversity signal, defining diversity at the level of underlying solution tendency rather than text-level semantic similarity.
- We propose **APTS**, a tree search algorithm which guides search using probed-answer hidden state similarity and PPL to improve solution diversity.
- We conduct extensive experiments across three domains and two LLM architectures, showing that APTS consistently improves diversity with minimal impact on accuracy.

2 Preliminary Analysis

In this section, we use Game of 24 as a diagnostic task to study how different representations capture solution diversity. We show that response-level semantic embeddings are often insufficient to distinguish different solution paths, whereas answer-level hidden states provide a more discriminative representation and better reveal how reasoning trajectories diverge during generation.

2.1 Experimental Setup

Following Yao et al. (2023), we use a multi-solution Game of 24 dataset collected from `4nums.com`. It contains 204 problems, each associated with at least four solution categories. Here, a solution category is defined by its underlying mathematical logic: expressions that differ in surface form but follow the same logic (*e.g.,* $7 * 5 - 10 - 1$ and $7 * 5 - (10 + 1)$) are merged according to standard *24-point equivalence rules*[1].

For each problem p, we sample $N = 512$ responses $\{r_1, \ldots, r_N\}$ from Qwen3-8B (Team, 2025). Each response is a token sequence $r_i = (t_{i,1}, \ldots, t_{i,T_i})$, where T_i denotes the response length. We extract the final answer and denote its start and end token positions by s_i and e_i, respectively. The corresponding answer segment is $a_i = (t_{i,s_i}, \ldots, t_{i,e_i})$. Let $h_{i,j}$ denote the hidden state of token $t_{i,j}$.[2] We then compute four representations: response-level semantic embed-

[1] `https://www.4nums.com/theory/`

[2] We use hidden states from the 32nd layer, which yields the highest separability in our layer-wise analysis (see Appendix A for details).

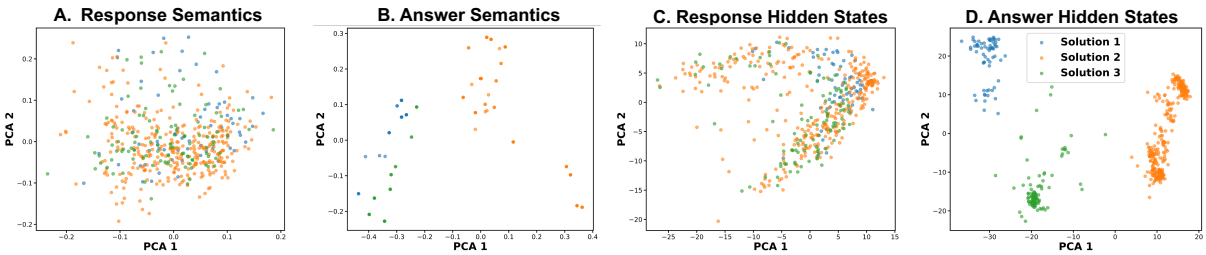


Figure 2: PCA visualization of solution-path clustering for problems (2,6,9,11) across different representation types. The Answer Semantics plots contain fewer points because many responses correspond to identical answers. Results for other problems are provided in Appendix Figure 14.

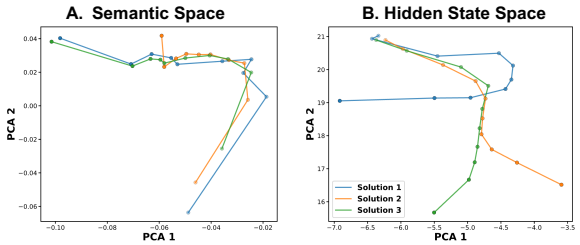


Figure 3: Visualization of solution-path trajectories for problems (2, 6, 9, 11) in semantic and hidden-state spaces. Lighter-colored nodes correspond to earlier prefixes. See Appendix Figure 15 for more results.

dings (RespSem), answer-level semantic embeddings (AnsSem), response-level hidden state averages (RespHid), and answer-level hidden state averages (AnsHid):

$$\begin{aligned}
\mathrm{RespSem}_i &= E_{\mathrm{sem}}(r_i), \\
\mathrm{AnsSem}_i &= E_{\mathrm{sem}}(a_i), \\
\mathrm{RespHid}_i &= E_{\mathrm{hid}}(r_i) = \tfrac{1}{T_i}\textstyle\sum_{j=1}^{T_i} h_{i,j}, \\
\mathrm{AnsHid}_i &= E_{\mathrm{hid}}(a_i) = \tfrac{1}{e_i - s_i + 1}\textstyle\sum_{j=s_i}^{e_i} h_{i,j}.
\end{aligned}$$

where $E_{\mathrm{sem}}(\cdot)$ denotes Qwen3-Embedding-0.6B (Zhang et al., 2025), and $E_{\mathrm{hid}}(\cdot)$ denotes mean pooling over token hidden states.

2.2 Answer-Level Representations Better Separate Solution Categories

We first examine whether answer-level representations better separate responses from different solution categories. Specifically, for each problem and representation type, we fit a 2D PCA transform on the corresponding representations and visualize their clustering patterns with respect to solution categories. As shown in Figure 2A and 2C, response-level representations are heavily overlapped and fail to distinguish different solution paths. In contrast, answer-level representations (Figure 2B and 2D) form clear clusters corresponding to each solution category. This visual observation is supported by quantitative analysis in Appendix B: using pairwise AUROC to measure solution separability, answer-level representations outperform response-level representations by 0.36 on average (0.94 vs. 0.58). These results indicate that semantic embeddings of full responses are less effective at distinguishing different solution paths. One plausible explanation is the high structural and stylistic homogeneity among responses to the same problem, which results in high semantic similarity (average cosine similarity of 0.95) and consequently limits response-level representation separability.

2.3 Hidden State Space Better Reveals Trajectory Divergence

We further analyze how different representation spaces capture trajectory divergence over the course of generation. Specifically, for each response $r_i = (t_{i,1}, \ldots, t_{i,T_i})$, we identify the token positions corresponding to 10%, 20%, …, 100% of its length, and use them to construct 10 cumulative prefixes:

$$r_i^{(k)} = (t_{i,1}, \ldots, t_{i,\lfloor kT_i/10 \rfloor}), \qquad k = 1, \ldots, 10.$$

We then group these prefixes by solution category and average their representations, yielding the semantic and hidden state representations of each solution at 10%, 20%, …, 100% of the reasoning process:

$$\begin{aligned}
s_{c,\mathrm{sem}}^{(k)} &= \tfrac{1}{|R_c|}\textstyle\sum_{r_i \in R_c} E_{\mathrm{sem}}(r_i^{(k)}), \\
s_{c,\mathrm{hid}}^{(k)} &= \tfrac{1}{|R_c|}\textstyle\sum_{r_i \in R_c} E_{\mathrm{hid}}(r_i^{(k)})
\end{aligned}$$

where R_c denotes the set of responses belonging to solution category c. For trajectory visualization, we fit PCA on answer-level representations (AnsSem or AnsHid) and project the reasoning trajectories (($s_{c,\mathrm{sem}}^{(k)}$ or $s_{c,\mathrm{hid}}^{(k)}$)) into this space to study how reasoning paths progress toward the final answer.

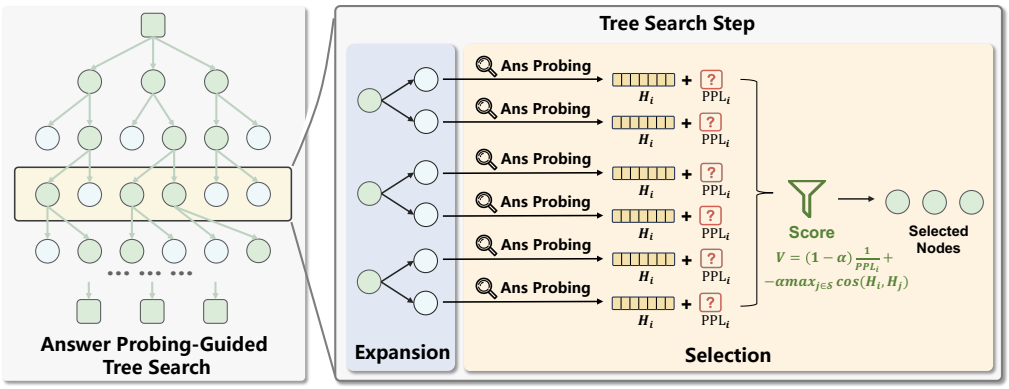


Figure 4: Overview of APTS.

Figure 3 shows that the hidden state space yields clearer separation among solution trajectories. Trajectories associated with different solutions originate from nearby regions and progressively diverge as generation proceeds, eventually reaching distinct endpoints. In contrast, trajectories in the semantic space are less coherent: they do not exhibit a clear common origin and remain highly overlapped throughout most of the generation process. Overall, these results suggest that hidden state representations provide a more discriminative view of intermediate reasoning development than semantic embeddings. This is consistent with prior work (Fang et al., 2026; Sheng et al., 2025), which shows that hidden state representations capture information about reasoning progress beyond semantic content alone. This finding justifies our use of hidden-state representations instead of semantic embeddings for Answer Probing.

3 Method

We present our method in two parts. First, motivated by the finding that answer-level hidden states better distinguish distinct solution paths, we propose *Answer Probing*, which characterizes an intermediate reasoning path by probing the answer that the LLM would generate from that path (Section 3.1). Second, we introduce APTS, which uses probed answers' hidden state similarity and PPL to guide tree search, thereby improving solution diversity in LLM generation (Section 3.2).

3.1 Answer Probing

As shown in Section 2, answer-level hidden states provide more discriminative representations of different solution paths than response-level semantic embeddings. However, during generation, the final answer is unavailable for an intermediate reasoning path. To address this issue, we introduce *Answer Probing*, which characterizes an intermediate reasoning path by probing the answer that the LLM would generate from that path.

Formally, given a problem p, let r denote an intermediate reasoning path, represented as a partially generated response. Answer Probing constructs a probing context $c = p \oplus r \oplus \pi_{\text{ans}}$ by appending an answer-inducing prompt π_{ans} (*e.g.*, "The final answer is")[3] to r, and then prompting the LLM to generate a probed answer PA:[4]

$$PA = \text{LLM}(c) = (a_1, a_2, \ldots, a_L),$$

where L is the length of the probed answer. We then use the hidden states of the probed-answer tokens to represent the reasoning path. Specifically, let h_i denote the hidden state of token a_i in PA. The representation of r is computed as:

$$H(r) = \frac{1}{L}\sum_{i=1}^{L} h_i.$$

By forcing the LLM to produce an answer directly from an intermediate reasoning path, Answer Probing exposes the latent solution tendency encoded in that state. As a result, two intermediate reasoning states that are linguistically similar but lead to different solutions can still be distinguished by their probed-answer representations.

An additional benefit of Answer Probing is that it provides a quality signal. For a probed answer $PA = (a_1, a_2, \ldots, a_L)$, we compute its PPL under the current reasoning path as

$$\text{PPL}(PA) = \exp\left(-\frac{1}{L}\sum_{i=1}^{L} \log p_\theta(a_i \mid c, a_{<i})\right).$$

[3]We analyze the sensitivity to the probing prompt formulation in Appendix C.

[4]We use greedy decoding here to eliminate sampling randomness.

A lower probed-answer PPL indicates that the LLM produces the answer more confidently from the current path, and thus serves as a useful proxy for the quality of that reasoning path.[5]

Overall, Answer Probing provides two useful signals: hidden-state similarity between probed answers, which reflects similarity between underlying solution paths, and probed-answer PPL, which reflects the quality of the current reasoning state.

3.2 Answer Probing-Guided Tree Search

Building on the two signals provided by Answer Probing, we propose APTS, which guides tree search toward high-quality and diverse candidate nodes, thereby improving solution diversity in LLM generation.

As shown in Figure 4, APTS adopts a breadth-first tree search in which each node represents an intermediate reasoning step.[6] For a problem p requiring M responses, APTS first samples M initial nodes, denoted by $\mathcal{N}^{(0)} = \{n_1^{(0)}, n_2^{(0)}, \ldots, n_M^{(0)}\}$. At each search step d, APTS performs *Expansion* and *Selection*. During the expansion stage, each node in $\mathcal{N}^{(d)}$ generates W child nodes, where W is the node expansion width. This yields a candidate set $\mathcal{C}^{(d+1)} = \{c_1^{(d+1)}, c_2^{(d+1)}, \ldots, c_{MW}^{(d+1)}\}$. During the selection stage, APTS chooses M nodes from $\mathcal{C}^{(d+1)}$ to form $\mathcal{N}^{(d+1)}$ for further exploration. The search terminates when the maximum search depth D is reached or all nodes in $\mathcal{N}^{(d)}$ have produced final answers.

To select $\mathcal{N}^{(d+1)}$, APTS performs M rounds of greedy diversity-aware selection while maintaining a selected set $\mathcal{S}$, initially empty. In each round, for each remaining candidate $c_i^{(d+1)} \in \mathcal{C}^{(d+1)} \setminus \mathcal{S}$, we compute the node value as:

$$V(c_i^{(d+1)}) = (1-\alpha) \cdot \frac{1}{\mathrm{PPL}_i} + \alpha \cdot DS_i,$$

$$DS_i = \begin{cases} 0, & \text{if } \mathcal{S} = \emptyset, \\ -\max\limits_{c_j \in \mathcal{S}} \cos(H_i, H_j), & \text{otherwise,} \end{cases}$$

where the first term measures node quality, the second term DS_i encourages diversity with respect to the set of already selected nodes, and $\alpha \in [0, 1]$ is a trade-off coefficient. In each round, the candidate with the highest node value is added to $\mathcal{S}$. After M rounds, $\mathcal{S}$ becomes $\mathcal{N}^{(d+1)}$. Full pseudocode of APTS is provided in Appendix, Algorithm 1.

[5]We analyze the distribution of inverse-PPL values across tasks in Appendix D.

[6]In this work, reasoning steps are separated using "\n\n" as the delimiter.

4 Experiments

In this section, we conduct experiments to address the following research questions:

- **RQ1:** How effectively does APTS enhance solution diversity compared to baseline methods?
- **RQ2:** What roles do the two signals in Answer Probing play in APTS, and is AnsHid a more effective representation than alternative representations for APTS?
- **RQ3:** How sensitive is APTS to variations in its key hyperparameters?

4.1 Experiment Setup

Datasets and Evaluation. We conduct experiments on three datasets spanning math, chemistry, and code: the multi-solution Game of 24 dataset introduced in Section 2, Forward Synthesis (Yu et al., 2024), and TestCaseGen (Huang et al., 2025). We evaluate on Qwen3-8B (Team, 2025) and GPT-OSS-120B (OpenAI, 2025), sampling 16 responses per problem. For Game of 24, we report accuracy (**Acc@16**) and solution category coverage (**Cov@16**). For Forward Synthesis, we report product quality (**Conf@16**) (Schwaller et al., 2021) and chemical space coverage (**NCircle@16**) (Xie et al., 2023). For TestCaseGen, we report test accuracy (**Acc@16**), line coverage (**L-Cov@16**), and branch coverage (**B-Cov@16**).

We use a task-specific diversity metric for each task rather than a single unified metric, because "genuinely distinct solutions" means different things across domains: distinct arithmetic structures in math, diverse chemical scaffolds in chemistry, and different program paths in code. Concretely, **Cov@16** $\in [0, 1]$ is the fraction of valid solution categories covered by the 16 answers; **NCircle@16** $\in [1, 16]$ is the size of the largest subset of generated molecules that are mutually dissimilar (Tanimoto similarity below 0.75) (Xie et al., 2023); and **L-Cov@16** and **B-Cov@16** $\in [0, 1]$ are the line and branch coverage of the target code achieved by the correct generated tests. For all diversity metrics, higher is better. Each metric is adopted from its task's original benchmark and is community-standard, consistent with prior work on diverse generation (Zhu et al., 2026). See Appendix E for more details.

Baselines and Implementation. We compare APTS with five representative methods spanning decoding-based, search-based, and steering-

Model	Method	Game of 24		Forward Synthesis		TestCaseGen		
		Acc@16	Cov@16	Conf@16	NCircle@16	Acc@16	L-Cov@16	B-Cov@16
Qwen3-8B	Repeated Sampling (T=0.5)	0.9479	0.6080	0.2820	7.275	0.7360	0.6889	0.5744
	Repeated Sampling (T=1.0)	**0.9559**	0.6179	**0.2823**	7.630	0.7394	0.7056	0.5896
	Repeated Sampling (T=1.5)	0.9289	0.6250	0.1513	7.130	0.6138	0.6859	0.5731
	Diverse Beam Search	0.9311	0.6238	0.2784	7.775	0.7172	0.7063	0.5957
	SemDiD (NeurIPS'25)	0.9553	0.6337	0.2782	7.930	**0.7484**	0.7188	0.6088
	GuidedSampling (ICLR'26)	0.9240	0.6383	0.2783	7.715	0.6956	0.7192	0.6106
	STARS (ICLR'26)	0.9525	0.6185	0.2794	7.800	0.7178	0.7118	0.6079
	APTS(Ours)	0.9445	**0.6913**	0.2808	**8.150**	0.7322	**0.7306**	**0.6232**
GPT-OSS-120B	Repeated Sampling (T=0.5)	**0.9507**	0.6046	0.4411	3.190	**0.9347**	0.7965	0.6927
	Repeated Sampling (T=1.0)	0.9292	0.6211	0.4423	3.255	0.9319	0.8150	0.7197
	Repeated Sampling (T=1.5)	0.9032	0.6695	0.2483	2.420	0.8753	0.8280	0.7363
	Diverse Beam Search	0.8986	0.6319	0.4369	3.290	0.9297	0.8242	0.7335
	SemDiD (NeurIPS'25)	0.9139	0.6544	0.4333	3.330	0.9294	0.8259	0.7368
	GuidedSampling (ICLR'26)	0.8879	0.6432	0.4214	3.300	0.8941	0.8234	0.7322
	STARS (ICLR'26)	0.9124	0.6344	0.4335	3.270	0.9281	0.8216	0.7324
	APTS(Ours)	0.9053	**0.6926**	**0.4499**	**3.665**	0.9309	**0.8335**	**0.7430**

Table 1: Comparison of different decoding methods on Game of 24, Forward Synthesis, and TestCaseGen. Bold indicates the best result and underline indicates the second best, and gray background denotes diversity metrics.

based approaches, including recent methods from NeurIPS 2025 and ICLR 2026: Repeated Sampling (Renze, 2024), Diverse Beam Search (Vijayakumar et al., 2018), SemDiD (Shi et al., 2025), GuidedSampling (Handa et al., 2026), and STARS (Zhu et al., 2026). Baseline and experimental details are provided in Appendix F.

4.2 Diversity Performance Compared with Baselines (RQ1)

Main Results. We evaluate the performance of different methods in Table 1, from which we draw the following observations: (1) **APTS achieves the best solution diversity across all three tasks and both LLMs.** Compared with Repeated Sampling at the same temperature ($T = 1.0$), APTS yields an average absolute improvement of over 7% in Cov@16 on Game of 24 across the two LLMs. On TestCaseGen, it yields an average absolute improvement of over 5% in the mean of L-Cov@16 and B-Cov@16. On Forward Synthesis, it improves NCircle@16 by 0.47 on average. These results demonstrate the effectiveness and robustness of APTS; (2) **Simply increasing the sampling temperature does not necessarily improve the diversity of valid solutions.** In fact, raising the temperature to $T = 1.5$ yields lower diversity than $T = 1.0$ on Forward Synthesis for both LLMs and on TestCaseGen for Qwen3-8B. This indicates that relying solely on sampling randomness is insufficient for increasing valid solution diversity, because excessive randomness yields superficially diverse yet incorrect outputs; (3) **APTS achieves a better trade-off between correctness and diversity.** Increasing the temperature to $T = 1.5$ improves diversity but typically causes a substantial drop in correctness (9% on average). In contrast, APTS improves diversity while incurring only a negligible correctness decrease of 0.6% relative to Repeated Sampling ($T = 1.0$). This is because APTS explicitly evaluates the quality of candidate paths during search and selects promising paths that maintain both correctness and diversity.

APTS on Open-Ended Tasks and Closed-Source LLMs. To further evaluate the generalizability of APTS, we conduct two additional experiments. First, we evaluate APTS on LiveIdeaBench (Ruan et al., 2026), an open-ended scientific idea generation benchmark, and find that APTS consistently improves diversity while maintaining quality (see Appendix G). Second, we adapt APTS to the closed-source GPT-o3-mini by replacing hidden-state signals with semantic embeddings and semantic entropy, and observe consistent diversity improvements (see Appendix H). These results show that APTS can be effectively applied to open-ended generation tasks and adapted to closed-source LLMs.

Latency and Efficiency Analysis. Considering that APTS introduces additional computational overhead due to tree search and answer probing, we measure the wall-clock latency of APTS. Taking Game of 24 on Qwen3-8B as an example, APTS introduces only 1.46× overhead over Repeated Sampling with the same sample count ($N = 16$), thanks to prefix caching and batching of short continu-

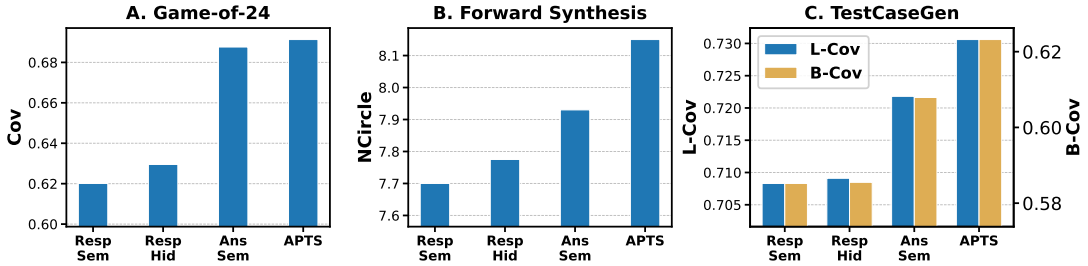


Figure 5: Comparison of different representations used as diversity signals in APTS on Qwen3-8B.

ations. We also measure the latency of budget-matched Repeated Sampling, where the number of samples is increased until its total token cost matches that of APTS ($N = 29$ for Game of 24), and find that APTS ($N = 16$) is faster than budget-matched Repeated Sampling ($N = 29$). Moreover, we compare the solution diversity under this budget-matched setting and find that APTS still achieves higher diversity than Repeated Sampling with more trials. This further demonstrates the effectiveness of APTS. See Appendix J for details.

4.3 Understanding the Role of Answer Probing in APTS (RQ2)

Effectiveness of probing-answer hidden states as diversity signals. We further assess the effectiveness of using probing-answer hidden states as the diversity signal for APTS by comparing it with alternative representations. Specifically, we replace it with alternative representations introduced in Section 2, including RespSem, AnsSem, and RespHid, and use them to compute the similarity penalty that guides node selection in APTS. The results are shown in Figure 5, from which we draw the following observations: **(1)** Answer-level representations consistently achieve higher solution diversity than response-level representations across all three tasks. This result is consistent with our analysis in Section 2, suggesting that answer-level representations better capture distinctions among solutions; and **(2)** Compared with AnsSem, APTS using the hidden states of probed answers as the diversity signal achieves clear improvements on TestCaseGen and Forward Synthesis. This indicates that the hidden-state space is more discriminative than the semantic embedding space, allowing APTS to better distinguish different reasoning trajectories.

Roles of the two Answer Probing signals in APTS. To understand the respective roles of the two Answer Probing signals, we evaluate APTS under different values of the trade-off coefficient

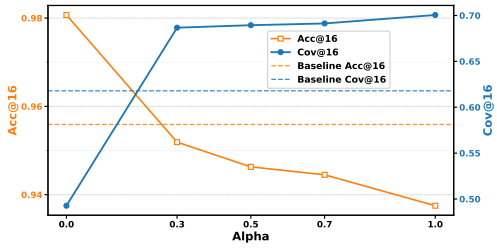


Figure 6: Effect of the trade-off coefficient α on Game of 24 with Qwen3-8B. Results for other datasets are provided in Appendix Figure 11.

α. Figure 6 presents the results on Game of 24 with Qwen3-8B, and we observe the same trend on TestCaseGen and Forward Synthesis (Appendix Figure 11). From these results, we draw the following observations: (1) When $\alpha = 0$, APTS selects nodes solely based on probed-answer PPL. In this setting, APTS achieves higher accuracy than Repeated Sampling (0.9807 vs. 0.9559), demonstrating that probed-answer PPL is an effective signal for estimating reasoning correctness; and (2) As α increases, the hidden-state similarity penalty in Answer Probing plays a larger role in node selection, leading to steadily improved solution coverage. Notably, once the similarity penalty is introduced (*i.e.*, $\alpha \neq 0$), APTS consistently achieves higher solution coverage than the baseline, demonstrating the effectiveness of hidden-state similarity for encouraging diverse exploration.

4.4 Impact of Hyperparameters (RQ3)

Influence of Node Expansion Width. We study how the node expansion width of APTS affects solution diversity. As shown in Figure 7, increasing the expansion width from 2 to 4 leads to substantial improvements in solution diversity. However, when the width is further increased from 4 to 5, the gains become much smaller on Game of 24 (with the same trend also observed on TestCaseGen; see Appendix Figure 12), and even slightly decrease on

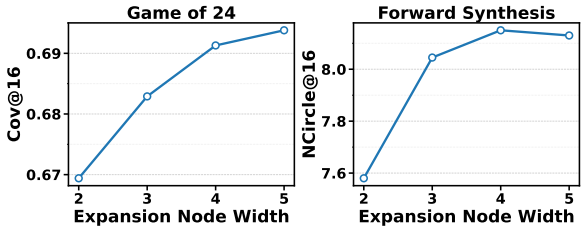


Figure 7: Effect of node expansion width on the solution diversity of APTS on Qwen3-8B.

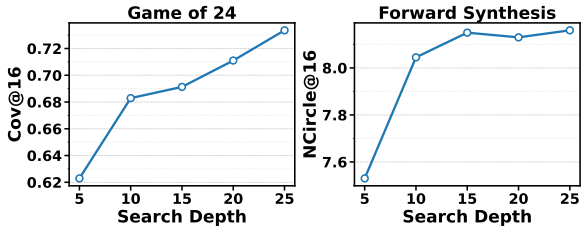


Figure 8: Effect of max search depth on the solution diversity of APTS on Qwen3-8B.

Forward Synthesis. These results suggest that setting the node expansion width to 4 provides a good trade-off. Larger widths incur additional reasoning overhead while bringing only marginal diversity improvements.

Influence of Max Search Depth. We also study how the maximum search depth of APTS affects solution diversity. As shown in Figure 8, we observe two distinct trends across tasks. For Game of 24, solution diversity increases steadily as the search depth grows. In contrast, for Forward Synthesis, solution diversity improves notably as the depth increases up to 15, but further increasing the depth beyond 15 brings only marginal gains (we observe a similar trend on TestCaseGen; see Appendix Figure 13). This suggests that the effect of search depth is task-dependent.

We think this may be because, for Game of 24, different solution paths may diverge relatively late, so deeper search can continue to uncover new valid solutions. However, for tasks such as Forward Synthesis and TestCaseGen, most useful branching may occur within a moderate depth, after which the LLM tends to follow previously explored reasoning patterns and produce similar answers. These results suggest that the optimal search depth should be selected according to the task. In practice, it is preferable to tune the search depth on a validation set and choose the smallest value that captures most of the diversity gains.

5 Related Work

Diversity Generation of LLMs. Prior work improves LLM output diversity at inference time mainly through two directions: *prompt-side diversification*, which varies the input prompt (Naik et al., 2023; Handa et al., 2026), and *generation-side diversification*, which modifies the generation process under a fixed prompt. These directions are complementary, and we focus on the latter in this work. Generation-side methods can be further divided into *decoding-based* and *search-based* approaches. Decoding-based methods adjust parameters such as temperature (Renze, 2024) and top-p (Holtzman et al., 2020), but often yield only token-level variation. Search-based methods explore multiple reasoning paths and prune similar ones using semantic embeddings (Shi et al., 2025). However, semantic similarity can be confounded by linguistic or stylistic noise, making truly distinct reasoning paths hard to detect. Answer probing addresses this issue by focusing on answer-level signals from intermediate states.

Positioning of Answer Probing. Several recent works explore mechanisms related to Answer Probing. Certaindex (Fu et al., 2025) introduces a probe-in-the-middle mechanism that extracts intermediate answers during reasoning to estimate certainty and enable early stopping. Answer Probing extends this probing mechanism to serve as both a diversity signal (via hidden-state similarity between probed answers) and a quality signal (via probed-answer PPL), rather than solely estimating certainty. On the diversity side, prior work mainly studies diversity in semantic embedding space (*e.g.,* SD-E^2 (Mishra et al., 2026)), or, when using hidden states, focuses on global trajectory geometry rather than solution-level convergence (*e.g.,* VERL (Huang et al., 2026)). In contrast, Answer Probing defines diversity at the level of underlying solution tendency—whether two intermediate states are likely to converge to the same final solution. As shown in Section 2, this representation better separates distinct solution paths than semantic embeddings, which are easily confounded by linguistic and stylistic similarities. Additionally, tree-structured generation has been adopted for credit assignment in LLM reasoning (Guo et al., 2025) and RL post-training of visual generative models (Ding and Ye, 2026). While APTS also employs tree search, its contribution lies in the

answer-probing-based signals used to guide node selection.

6 Conclusion

In this work, we propose Answer Probing, which probes the potential answers an LLM would reach from an intermediate reasoning state. The probed answers provide two useful reasoning signals: the hidden-state similarity between probed answers reflects the diversity of reasoning trajectories, while the PPL of a probed answer reflects the quality of the underlying reasoning. Based on these two signals, we further propose APTS, which guides tree search toward candidate nodes that are both high-quality and diverse, thereby improving solution-level diversity in LLM generation while maintaining strong overall performance.

Limitations

While APTS improves solution diversity in LLM generation, it still has several limitations. First, APTS requires sampling multiple candidate reasoning paths at each search depth and probing their potential answers, which introduces additional computational overhead. Second, APTS mainly focuses on solution diversity rather than semantic diversity, and thus may be less suitable for non-reasoning tasks such as creative writing. Third, the original APTS formulation depends on access to intermediate hidden states and log probabilities, limiting its direct applicability to closed-source models. Fourth, the trade-off coefficient α may not transfer reliably across all tasks and models; in practice, a suitable α can be selected using a small validation set. Fifth, the probed-answer PPL used as a quality signal relies on the model's own confidence, which may be unreliable in rare or challenging scenarios where the model has blind spots. We provide a detailed case study and error analysis in Appendix K.

Acknowledgments

This work is supported by the Zhongguancun Academy (Grant No. C20250401).

References

Dávid Bajusz, Anita Rácz, and Károly Héberger. 2015. Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics*, 7(1):20.

Zheng Ding and Weirui Ye. 2026. TreeGRPO: Tree-advantage GRPO for online RL post-training of diffusion models. In *ICLR*.

Jürgen Drews. 2000. Drug discovery: A historical perspective. *Science*.

Yi Fang, Wenjie Wang, Mingfeng Xue, Boyi Deng, Fengli Xu, Dayiheng Liu, and Fuli Feng. 2026. Controllable llm reasoning via sparse autoencoder-based steering.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*.

Yichao Fu, Junda Chen, Siqi Zhu, Fu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana S Rosing, Ion Stoica, and Hao Zhang. 2025. Efficiently scaling llm reasoning programs with certaindex. In *NeurIPS*.

Yiran Guo, Lijie Xu, Jie Liu, Ye Dan, and Shuang Qiu. 2025. Segment policy optimization: Effective segment-level credit assignment in RL for large language models. In *NeurIPS*.

Divij Handa, Mihir Parmar, Aswin RRV, Md Nayem Uddin, Hamid Palangi, and Chitta Baral. 2026. Guided-sampling: Steering LLMs towards diverse candidate solutions at inference-time. In *ICLR*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *ICLR*.

Dong Huang, Jie M Zhang, Mark Harman, Qianru Zhang, Mingzhe Du, and See-Kiong Ng. 2025. Benchmarking llms for unit test generation from real-world functions. *arXiv preprint arXiv:2508.00408*.

Fanding Huang, Guanbo Huang, Xiao Fan, Yi He, Xiao Liang, Xiao Chen, Qinting Jiang, Faisal Nadeem Khan, Jingyan Jiang, and Zhi Wang. 2026. Semantic-space exploration and exploitation in rlvr for llm reasoning.

Kush Jain, Gabriel Synnaeve, and Baptiste Rozière. 2025. Testgeneval: A real world unit test generation and test completion benchmark. In *ICLR*.

Liwei Jiang, Yuanjun Chai, Margaret Li, Mickel Liu, Raymond Fok, Nouha Dziri, Yulia Tsvetkov, Maarten Sap, Alon Albalak, and Yejin Choi. 2025. Artificial hivemind: The open-ended homogeneity of language models (and beyond). *NIPS*.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2024. Understanding the effects of rlhf on llm generalisation and diversity. *ICLR*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Kshitij Mishra, Nils Lukas, and Salem Lahlou. 2026. SD-e2: Semantic exploration for reasoning under token budgets. In *Findings of the Association for Computational Linguistics: EACL 2026*.

Ranjita Naik, Varun Chandrasekaran, Mert Yüksekgönül, Hamid Palangi, and Besmira Nushi. 2023. Diversity of thought improves reasoning abilities of large language models. *arXiv preprint arXiv:2310.07088*.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card.

Laura O'Mahony, Leo Grinsztajn, Hailey Schoelkopf, and Stella Biderman. 2024. Attributing mode collapse in the fine-tuning of large language models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Matthew Renze. 2024. The effect of sampling temperature on problem solving in large language models. In *EMNLP (Findings)*.

Kai Ruan, Xuan Wang, Jixiang Hong, Peng Wang, Yang Liu, and Hao Sun. 2026. Evaluating llms' divergent thinking capabilities for scientific idea generation with minimal context. *Nature communications*.

Philippe Schwaller, Benjamin Hoover, Jean-Louis Reymond, Hendrik Strobelt, and Teodoro Laino. 2021. Extraction of organic chemistry grammar from unsupervised learning of chemical reactions. *Science Advances*.

Leheng Sheng, An Zhang, Zijian Wu, Weixiang Zhao, Changshuo Shen, Yi Zhang, Xiang Wang, and Tat-Seng Chua. 2025. On reasoning strength planning in large reasoning models. *NeurIPS*.

Weijie Shi, Yue Cui, Yaguang Wu, Jingzhi Fang, Shibo Zhang, Mengze Li, Sirui Han, Jia Zhu, Jiajie Xu, and Xiaofang Zhou. 2025. Semantic-guided diverse decoding for large language model.

Yuda Song, Julia Kempe, and Rémi Munos. 2025. Outcome-based exploration for LLM reasoning. *CoRR*, abs/2509.06941.

Qwen Team. 2025. Qwen3 technical report.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search: Decoding diverse solutions from neural sequence models.

Wenhan Wang, Chenyuan Yang, Zhijie Wang, Yuheng Huang, Zhaoyang Chu, Da Song, Lingming Zhang, An Ran Chen, and Lei Ma. 2025. Testeval: Benchmarking large language models for test case generation. *NAACL (Findings)*.

Yutong Xie, Ziqiao Xu, Jiaqi Ma, and Qiaozhu Mei. 2023. How much space has been explored? measuring the chemical space covered by databases and machine-generated molecules. *ICLR*.

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

Botao Yu, Frazier N Baker, Ziqi Chen, Xia Ning, and Huan Sun. 2024. Llasmol: Advancing large language models for chemistry with a large-scale, comprehensive, high-quality instruction tuning dataset. *COLM*.

Longfei Yun, Chenyang An, Zilong Wang, Letian Peng, and Jingbo Shang. 2025. The price of format: Diversity collapse in llms. In *EMNLP (Findings)*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Dongxuan Zhu, Ly Tran Ho Khanh, Andy Yat-Ming Cheung, Man-Chung Yue, and Viet Anh Nguyen. 2026. Exploring diverse generation paths via inference-time stiefel activation steering. In *ICLR*.

A Layer-wise Analysis of Hidden-State Separability

To determine which hidden layer provides the best separability across solution categories, we conduct a layer-wise analysis on the Game of 24 dataset using Qwen3-8B. For each problem, we sample 512 responses and extract their answer-level hidden-state representations from every transformer layer. For each solution category, we then compute the mean of all answer-level hidden-state representations within that category and use it as the category-level representation.

We quantify how well different solution categories are separated in each layer using three metrics: average inter-class Euclidean distance, minimum inter-class Euclidean distance, and silhouette score. The first two metrics measure the overall and worst-case separation, respectively, among category-level representations. Larger values indicate better separability between different solution categories. The silhouette score measures how well individual answer-level hidden-state representations cluster according to their solution categories, jointly reflecting intra-category compactness and inter-category separation. Larger values indicate better clustering quality. As shown in Figure 9, the 32nd layer achieves both strong inter-class separation and high clustering quality. We therefore use the 32nd layer in our main experiments.

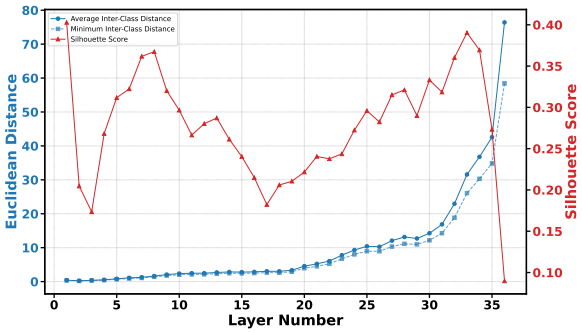


Figure 9: Layer-wise separability analysis of answer-level hidden states on Game of 24 using Qwen3-8B.

B Quantitative Analysis of Solution Separability

To quantitatively evaluate how well different representations separate solution categories, we conduct a pairwise separability analysis. Specifically, for each problem, we randomly sample 10,000 response pairs from the 512 responses generated by Qwen3-8B on the Game of 24 dataset. We compute the cosine similarity for each pair under four representations: RespSem, RespHid, AnsSem and AnsHid. We treat response pairs from the same solution category as positive examples and use their cosine similarity as the prediction score. We then compute the AUROC for distinguishing same-category pairs from different-category pairs under each representation. A higher AUROC indicates that the representation better captures solution-level distinctions.

Table 2 shows that answer-level representations substantially outperform response-level representations in terms of solution separability. We believe this is mainly because many responses share the same final answer, resulting in identical answer-level semantic embeddings. This makes same-category response pairs especially easy to distinguish from different-category pairs under AnsSem, thereby slightly improving the AUROC.

However, for APTS, AnsHid achieves better performance than AnsSem, as shown in Figure 5. This suggests that while AnsSem is more effective at separating responses based on their final answers, AnsHid is better at distinguishing differences in their intermediate reasoning processes. We hypothesize that this is because hidden-state representations preserve richer reasoning-related information beyond surface-level answer semantics. Consequently, AnsHid is more advantageous for answer probing.

Table 2: Pairwise AUROC for solution separability under different representations on the Game of 24 dataset using Qwen3-8B. Higher is better.

RespSem	RespHid	AnsSem	AnsHid
0.57	0.58	0.95	0.92

C Sensitivity to Probing Prompt Formulation

To examine whether APTS is sensitive to the exact wording of the probing prompt, we conduct experiments on Game of 24 with Qwen3-8B using three different probing prompts: “Final answer:”, “Based on the reasoning so far, the answer is”, and “My current best answer is”.

We first compare the probed-answer hidden states across prompt variants. For 100 randomly selected intermediate reasoning trajectories, the resulting probed-answer hidden states exhibit an

average cosine similarity of 0.9516 across prompts, indicating that the hidden-state representation is highly consistent despite changes in prompt wording.

We further evaluate the full APTS pipeline using each probing prompt. As shown in Table 3, the final Acc@16 and Cov@16 are very close across all three formulations, suggesting that APTS is not highly sensitive to the exact wording of the probing prompt.

Table 3: Effect of probing prompt formulation on Game of 24 with Qwen3-8B.

Probing Prompt	Acc@16	Cov@16
Final answer	0.9445	0.6913
Based on the reasoning so far, the answer is	0.9439	0.6927
My current best answer is	0.9463	0.6952

D Inverse-PPL Distribution Analysis

One potential concern is that the inverse-PPL term in the node value function is unbounded, while the cosine similarity term is bounded in $[-1, 1]$, leading to an unpredictable diversity–quality trade-off. However, because the probed answer is generated via greedy decoding, the token probabilities along the probed answer tend to be relatively high, making the resulting inverse-PPL naturally concentrated and close to 1.

We verify this empirically on Qwen3-8B across all three tasks. As shown in Table 4, the actual inverse-PPL values used by APTS are tightly concentrated within roughly $[0.85, 1.00]$ across tasks, and do not vary by orders of magnitude. This suggests that the quality term remains well-behaved in practice, and a fixed α can work reasonably across different tasks.

Table 4: Distribution of inverse-PPL values used by APTS on Qwen3-8B.

Task	Avg	Min	Max
Game of 24 (Math)	0.95	0.87	1.00
TestCaseGen (Code)	0.96	0.88	1.00
Forward Synthesis (Chem)	0.94	0.85	1.00

E Detailed Evaluation Protocol

We conduct experiments on three datasets. We select these tasks because they require LLM reasoning to ensure correctness, while solution diversity is also valuable in these settings.

Game of 24. The multi-solution Game of 24 dataset is introduced in Section 2. We report **Acc@16**, which measures the average correctness of the 16 responses, and **Cov@16**, which measures coverage over all possible solution categories.

Forward Synthesis. In forward synthesis prediction (Yu et al., 2024), LLMs are asked to generate multiple plausible products from given reactants and reagents, so both chemical validity and diversity are crucial. We report **Conf@16**, which evaluates the quality of generated products using RXN-Mapper confidence (Schwaller et al., 2021), and **NCircle@16** (Xie et al., 2023), which measures the extent of chemical space explored by computing the largest subset of generated molecules such that no two molecules have Tanimoto similarity (Bajusz et al., 2015) above a predefined threshold 0.75.

TestCaseGen. In TestCaseGen (Huang et al., 2025), the goal is to generate unit tests for code, where more diverse test cases can improve coverage of different program behaviors and edge cases. Following prior work (Wang et al., 2025), we report **Acc@16**, which measures whether the generated test cases are executable and contain correct test assertions, as well as **L-Cov@16** and **B-Cov@16**, which measure the overall line coverage and branch coverage achieved by all correct generated test cases on the target code.

F Implementation Details

Baselines. We compare APTS with the following methods:

- **Repeated Sampling** (Renze, 2024): generates multiple samples using high temperature sampling.
- **Diverse Beam Search** (Vijayakumar et al., 2018): uses a tree-search framework and encourages diversity through token-level diversity penalties.
- **SemDiD** (Shi et al., 2025): adopts a tree-search framework, using semantic embeddings to guide exploration and prune semantically redundant branches.
- **GuidedSampling** (Handa et al., 2026): first generates diverse solution concepts and then samples answers conditioned on them.
- **STARS** (Zhu et al., 2026): steers hidden activations at inference time to encourage divergent generation.

Hyperparameters. Following prior work (Shi et al., 2025), we test temperatures of 0.5, 1.0, and 1.5 for Repeated Sampling. For Diverse Beam Search, SemDiD, and APTS, we set the sampling temperature to 1.0, the node expansion width W to 4, and the maximum search depth D to 15. All other hyperparameters for Diverse Beam Search and SemDiD follow their original papers. For APTS, we set the trade-off coefficient $\alpha = 0.7$.

G Evaluation on Open-Ended Generation

To evaluate whether APTS generalizes beyond explicit multi-solution settings, we conduct experiments on LiveIdeaBench (Ruan et al., 2026), a benchmark that requires LLMs to generate scientific ideas from carefully designed keywords. This setting better reflects open-ended generation scenarios where "solution diversity" is less well-defined.

Following the original LiveIdeaBench evaluation protocol, both quality and diversity are scored by multiple LLM judges on a 1–10 scale. Quality evaluates the generated ideas in terms of originality, feasibility, and clarity, while diversity measures how different the ideas generated for the same keyword are. For each keyword, we generate 16 ideas.

As shown in Table 5, APTS achieves the best or near-best quality while consistently improving diversity on both models, suggesting that it is also effective for open-ended generation tasks beyond explicit multi-solution settings.

Table 5: Results on LiveIdeaBench. Bold indicates the best diversity score.

Model	Method	Quality@16	Diverse@16
Qwen3-8B	Repeated Sampling	6.27	4.00
	Diverse Beam Search	6.11	4.08
	SemDiD	6.23	4.13
	APTS (Ours)	6.25	**4.19**
GPT-OSS-120B	Repeated Sampling	6.81	6.22
	Diverse Beam Search	6.76	6.31
	SemDiD	6.79	6.37
	APTS (Ours)	6.82	**6.44**

H Adaptation to Closed-Source Models

The original APTS formulation relies on hidden states and log probabilities, which are typically unavailable for closed-source models. To extend APTS to this setting, we adapt it by (1) replacing hidden-state embeddings with semantic embeddings of probed answers, and (2) replacing PPL with the semantic entropy (Farquhar et al., 2024) of probed answers.

We evaluate this adapted variant on GPT-o3-mini. As shown in Table 6, the adapted APTS still improves diversity over Repeated Sampling across all three tasks, suggesting that the core idea of answer probing for search guidance transfers beyond open-weight models.

Table 6: Adapted APTS on GPT-o3-mini. Bold indicates the best diversity score.

	Game of 24		Forward Synthesis		TestCaseGen		
Method	Acc@16	Cov@16	Conf@16	NCircle@16	Acc@16	L-Cov@16	B-Cov@16
Repeated Sampling	0.9084	0.6796	0.4092	5.23	0.9175	0.7288	0.6120
APTS (Ours)	0.9105	**0.7162**	0.4033	**5.75**	0.9153	**0.7390**	**0.6329**

I Evaluation on an Additional Model Family

To assess whether the effectiveness of APTS generalizes across model families, we further evaluate it on DeepSeek-R1-Distill-Llama-8B, a reasoning model from a different family than Qwen3-8B and GPT-OSS-120B. We report results on Game of 24 and Forward Synthesis. As shown in Table 7, APTS achieves the best solution diversity across both tasks (Cov@16 and NCircle@16), confirming that answer-probing-guided search transfers across model families. Together with Qwen3-8B, GPT-OSS-120B, and GPT-o3-mini (Appendix H), our evaluation covers four models spanning different families, scales (8B to 120B), and access types (open-weight and closed-source).

Table 7: Results on DeepSeek-R1-Distill-Llama-8B. Gray background denotes diversity metrics; bold indicates the best diversity score.

	Game of 24		Forward Synthesis	
Method	**Acc@16**	**Cov@16**	**Conf@16**	**NCircle@16**
Repeated Sampling (T=0.5)	0.5564	0.5564	0.2124	6.140
Repeated Sampling (T=1.0)	0.7518	0.6526	0.1710	5.205
Repeated Sampling (T=1.5)	0.0064	0.0199	0.0006	0.035
Diverse Beam Search	0.7411	0.6694	0.2105	6.745
SemDiD	0.7439	0.6727	0.2111	6.970
APTS (Ours)	0.7475	**0.6927**	0.2111	**7.175**

J Latency and Efficiency Analysis

Wall-Clock Latency. We measure the end-to-end inference latency of APTS on Game of 24 with Qwen3-8B, using vLLM on a single A100-80GB GPU under the same decoding settings as in the main experiments. Both APTS and Repeated Sampling reach a peak GPU memory of 72GB.

As shown in Table 8, APTS introduces 1.46× wall-clock overhead over Repeated Sampling with the same sample count, and is faster than budget-

Table 8: Average per-problem wall-clock time on Game of 24 with Qwen3-8B.

Method	Wall-clock time
Repeated Sampling ($N = 16$)	8.64s
Repeated Sampling (budget-matched, $N = 29$)	13.90s
APTS ($N = 16$)	12.61s

matched Repeated Sampling. The latency gap is narrowed by two implementation optimizations:

- **Prefix caching.** Because tree nodes share long prefixes, both node expansion and answer probing can reuse cached prefixes and only decode the short suffix specific to each node.
- **Batching.** Since both stages generate only short continuations, many tree nodes can be packed into a single LLM generation call. In our setup, we can batch up to 640 expansions in one call on a single A100-80GB GPU without exceeding memory limits.

Budget-Matched Comparison. Since APTS incurs additional token overhead due to tree search, a natural question is whether its gains could be achieved simply by spending the same budget on more repeated samples. To answer this, we conduct a budget-matched comparison by increasing the number of samples in Repeated Sampling until its total token cost matches that of APTS. Figure 10 presents the results on Game of 24 and TestCaseGen. We exclude Forward Synthesis from this analysis because its evaluation metric, N-Circle, is strongly correlated with the number of sampled solutions: as K increases, N-Circle naturally increases as well, making it difficult to disentangle improvements due to more effective search from those caused simply by returning more samples.

When K is small (*e.g.*, , $K = 5$), budget-matched Repeated Sampling can be comparable to or even slightly better than APTS, likely because direct sampling is still generate diverse solutions at low sample counts. However, as K grows, the marginal benefit of repeated sampling decreases due to increasingly frequent duplicate solutions, whereas APTS can make better use of the same budget by explicitly exploring alternative reasoning trajectories. As a result, APTS consistently outperforms budget-matched Repeated Sampling at larger K on both tasks. These results suggest that the gains of APTS do not arise merely from using more decoding tokens, but from answer-probing-guided search.

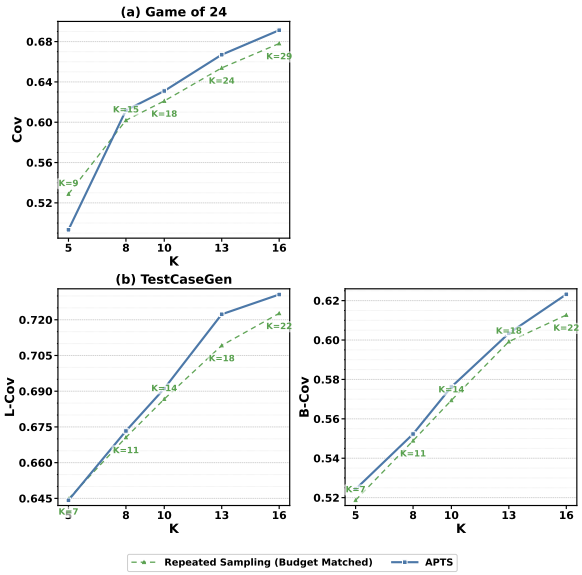


Figure 10: Budget-matched comparison on Qwen3-8B.

More importantly, APTS is more sample-efficient. For example, on Game of 24, APTS achieves higher solution coverage using only 16 final answers than budget-matched Repeated Sampling using 29 samples (0.6913 vs. 0.6781). Similarly, on TestCaseGen, APTS achieves higher line coverage and branch coverage with 16 generated test cases than budget-matched Repeated Sampling with 22 samples (0.7306 vs. 0.7227 in L-Cov and 0.6232 vs. 0.6127 in B-Cov). This is particularly desirable in applications where each generated candidate must be further validated, executed, or reranked.

K Case Study and Error Analysis

We present two representative failure cases from Game of 24 on Qwen3-8B ($\alpha = 0.7$) to provide deeper insights into APTS's behavior.

Case 1: Insufficient exploration under current α. For the problem [2, 6, 10, 12], there are 9 solution categories. As shown in Table 9, APTS covers only 3 of them, with the remaining 6 categories unexplored. The 16 answers concentrate on a few dominant categories (Cat 2 appears 7 times), while categories involving more creative use of division (*e.g.*, $(10 + 2) \times 12/6$, $(12/6 + 10) \times 2$) are not discovered. This suggests that the current α may not provide a sufficiently strong diversity incentive for problems with many solution categories, and increasing α could encourage broader exploration.

Table 9: APTS outputs for problem [2, 6, 10, 12] (9 categories, 3 covered, $\alpha = 0.7$).

Index	APTS Output
0	$(12-10)\times 6\times 2$
1	$(10-6)\times 12/2$
2–5	$(12-10)\times 6\times 2$
6	$(12+2-10)\times 6$
7–8	$(10-6)\times 12/2$
9	Error
10	$(12+2-10)\times 6$
11	Error
12	$(10-6)\times 12/2$
13–15	$(12-10)\times 6\times 2$

Case 2: Limited solution diversity in sampling. For the problem [2, 2, 4, 6], there are 5 solution categories, including $(4+2)\times(6-2)$, $(6+4+2)\times 2$, and $(4-2)\times 6\times 2$. However, as shown in Table 10, all 16 APTS answers produce the identical expression $6\times 4+2-2$, which exploits a trivial cancellation pattern ($a\times b+c-c$). This pattern recurs across problems with repeated numbers (*e.g.,* [2,6,6,12] $\rightarrow 12\times 2+6-6$; [4,6,8,8] $\rightarrow 6\times 4+8-8$). In these cases, the trivial cancellation acts as a strong attractor in the model's sampling behavior, and APTS cannot steer the model toward alternative solutions regardless of α.

Table 10: APTS outputs for problem [2, 2, 4, 6] (5 categories, 1 covered, $\alpha = 0.7$).

Index	APTS Output
0–15	$6\times 4+2-2$

L Additional Results

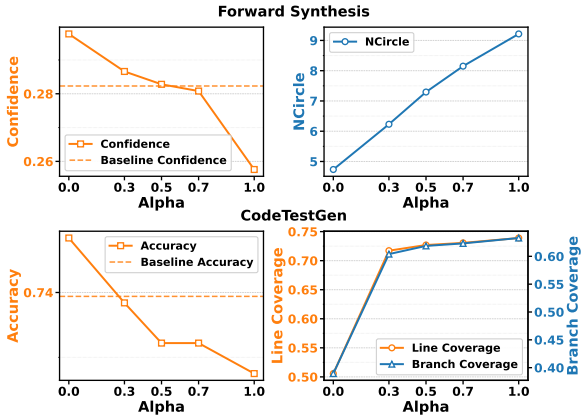


Figure 11: Effect of the trade-off coefficient α on Forward Synthesis and TestCaseGen with Qwen3-8B.

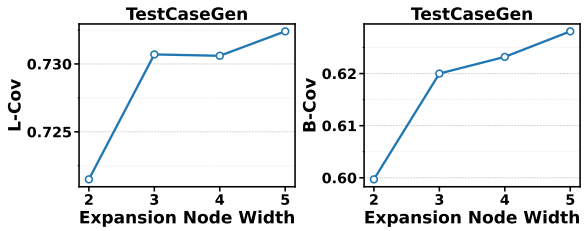


Figure 12: Effect of node expansion width on the solution diversity of APTS on TestCaseGen with Qwen3-8B

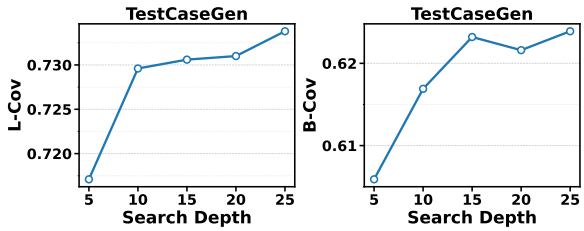
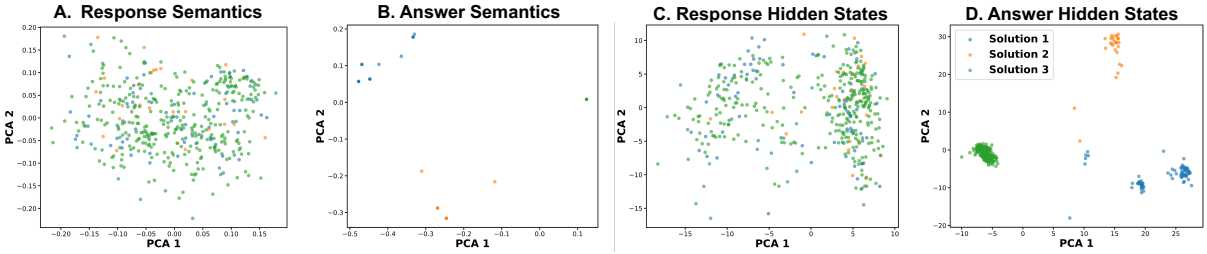
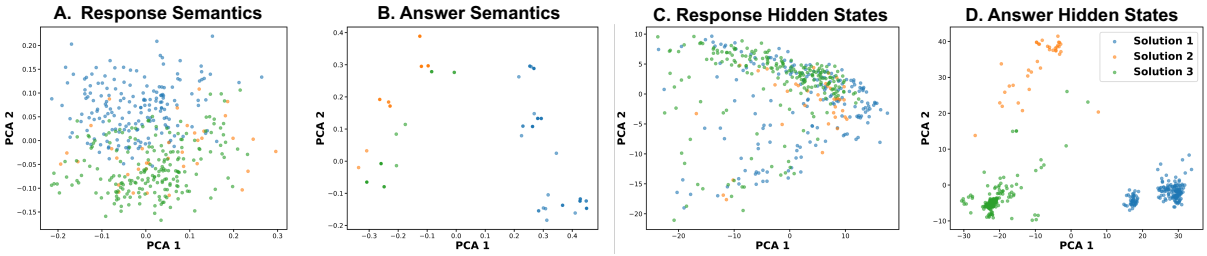


Figure 13: Effect of max search depth on the solution diversity of APTS on TestCaseGen with Qwen3-8B

(a) Problem (1, 3, 4, 8)



(b) Problem (2, 6, 7, 9)



(c) Problem (5, 5, 7, 7)

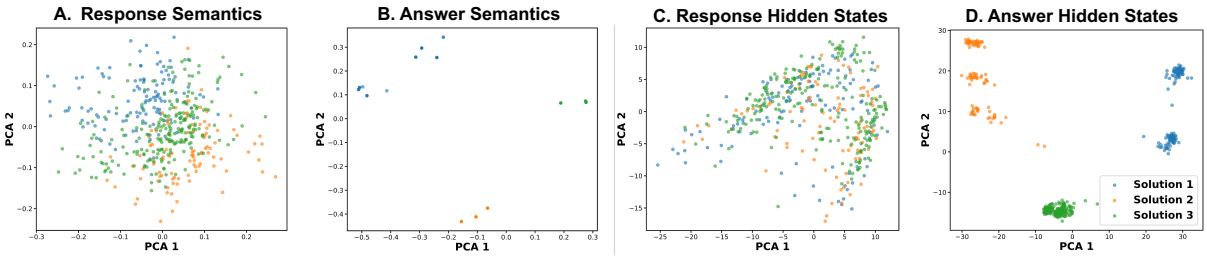
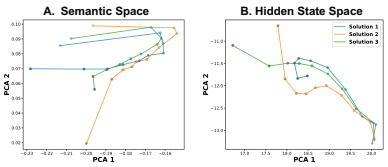
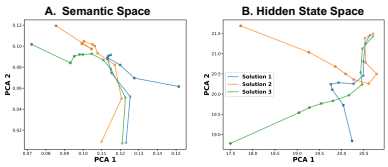


Figure 14: PCA visualizations of solution-path clustering across different representation types.

(a) Problem (1, 3, 4, 8)



(b) Problem (2, 6, 7, 9)



(c) Problem (5, 5, 7, 7)

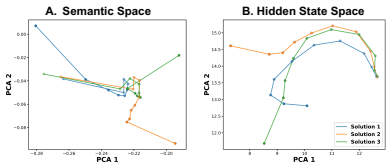


Figure 15: Visualizations of solution-path trajectories in semantic and hidden-state spaces for different problem groups.

Algorithm 1 APTS Algorithm

Input: Problem p, target response number M, node expansion width W, maximum search depth D, trade-off coefficient α

$\mathcal{N}^{(0)} \leftarrow \mathrm{LLM}(p, M) = \{n_1^{(0)}, \ldots, n_M^{(0)}\}$
for $d = 0$ to $D - 1$ **do**
  if all nodes in $\mathcal{N}^{(d)}$ are complete **then**
    break
  end if

  #### Expansion Stage ####
  $\mathcal{C}^{(d+1)} \leftarrow \emptyset$
  for each $n_i^{(d)} \in \mathcal{N}^{(d)}$ **do**
    $\text{Children}\left(n_i^{(d)}\right) \leftarrow \mathrm{GenerateNextStep}(p \oplus n_i^{(d)}, W)$
    $\mathcal{C}^{(d+1)} \leftarrow \mathcal{C}^{(d+1)} \cup \text{Children}\left(n_i^{(d)}\right)$
  end for
  for each $c_i \in \mathcal{C}^{(d+1)}$ **do**
    $(H_i, \mathrm{PPL}_i) \leftarrow$ **AnswerProbing**(p, c_i)
  end for

  #### Selection Stage ####
  $\mathcal{S} \leftarrow \emptyset$
  for $m = 1$ to M **do**
    for each $c_i \in \mathcal{C}^{(d+1)} \setminus \mathcal{S}$ **do**
      if $\mathcal{S} = \emptyset$ **then**
        $DS_i \leftarrow 0$
      else
        $DS_i \leftarrow -\max_{c_j \in \mathcal{S}} \cos(H_i, H_j)$
      end if
      $V(c_i) \leftarrow (1 - \alpha)/\mathrm{PPL}_i + \alpha DS_i$
    end for
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\arg\max_{c_i \in \mathcal{C}^{(d+1)} \setminus \mathcal{S}} V(c_i)\}$
  end for
  $\mathcal{N}^{(d+1)} \leftarrow \mathcal{S}$
end for
for each $n_i^{(d)} \in \mathcal{N}^{(d)}$ **do**
  Complete the response from $p \oplus n_i^{(d)}$
end for

Output: the resulting M responses