Don't Get Lost in the Trees: Streamlining LLM Reasoning by Overcoming Tree Search Exploration Pitfalls

Ante Wang[1,5*], **Linfeng Song**[2†], **Ye Tian**[2], **Dian Yu**[2], **Haitao Mi**[2], **Xiangyu Duan**[3]
Zhaopeng Tu[2†], **Jinsong Su**[1,4,5†] and **Dong Yu**[2]
[1]School of Informatics, Xiamen University, China [2]Tencent AI Lab
[3]School of Computer Science and Technology, Soochow University, China
[4]Shanghai Artificial Intelligence Laboratory, China
[5]Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan (Xiamen University), Ministry of Culture and Tourism, China

Abstract

Recent advancements in tree search algorithms guided by verifiers have significantly enhanced the reasoning capabilities of large language models (LLMs), but at the cost of increased computational resources. In this work, we identify two key challenges contributing to this inefficiency: *over-exploration* due to redundant states with semantically equivalent content, and *under-exploration* caused by high variance in verifier scoring leading to frequent trajectory switching. To address these issues, we propose FETCH – an **e**ffici**e**nt **t**ree sear**ch** framework, which is a flexible, plug-and-play system compatible with various tree search algorithms. Our framework mitigates over-exploration by merging semantically similar states using agglomerative clustering of text embeddings obtained from a fine-tuned SimCSE model. To tackle under-exploration, we enhance verifiers by incorporating temporal difference learning with adjusted λ-returns during training to reduce variance, and employing a verifier ensemble to aggregate scores during inference. Experiments on GSM8K, GSM-Plus, and MATH datasets demonstrate that our methods significantly improve reasoning accuracy and computational efficiency across four different tree search algorithms, paving the way for more practical applications of LLM-based reasoning. The code is available at https://github.com/DeepLearnXMU/Fetch.

1 Introduction

In recent months, the remarkable reasoning performance of OpenAI-o1 (OpenAI, 2024) has sparked significant research interest in enhancing the deductive capabilities of large language models (LLMs, Touvron et al. 2023; Jiang et al. 2023; Achiam et al. 2023) through inference-time scaling. One promising area in this field is the use of tree search algorithms guided by verifiers. Their effectiveness in finding optimal solutions for complex problems by exploring large search spaces has been demonstrated in various studies (Feng et al., 2023; Yao et al., 2024; Yu et al., 2024; Tian et al., 2024; Kang et al., 2024; Wang et al., 2024c; Zhang et al., 2024).

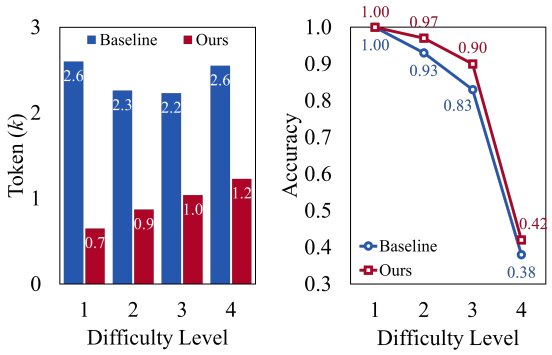


Figure 1: The effect of FETCH using Best-First Search as baseline on GSM8K problems of varying difficulty is illustrated. The left panel shows that FETCH reduces computational costs and prioritizes resource allocation to harder tasks (Levels 3–4) over simpler ones (Levels 1–2), addressing both over-exploration in basic problems and under-exploration in complex cases. This leads to better performance, as shown in the right panel.

However, the considerable increase in computational costs presents a significant challenge to the practical applications of these algorithms. Through our pilot study, we have identified the following two main issues that cause this inefficiency:

- *Over-exploration*: Due to the uncontrollable sampling of reasoning steps from LLMs, the search tree inevitably contains redundant states with semantically equivalent content. This leads to over-exploration of certain reasoning paths, as these redundant states are treated independently.

- *Under-exploration*: The verifiers used for search guidance may lack robustness and introduce unnecessary scoring variances due to differences

[*] Work dong during an internship at Tencent AI Lab
[†] Corresponding authors

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 23946–23959
July 27 - August 1, 2025

in reasoning style and phrasing choices, particularly when tackling challenging math problems. As a result, the search algorithm may frequently switch between different trajectories, leaving many potential paths unfinished by the end of the search.

In this paper, we propose FETCH, a flexible plug-and-play framework compatible with most existing tree search algorithms to tackle both over and under-exploration issues. With regard to over-exploration, the framework employs agglomerative clustering to merge similar states in a bottom-up style. Capturing accurate semantic-level similarity has always been a challenge in natural language processing (Dolan and Brockett, 2005; Agirre et al., 2014), especially when dealing with reasoning trajectories that usually involve complex deductions. We propose a lightweight sentence embedding model obtained by fine-tuning SimCSE (Gao et al., 2021) with signals from computationally intensive methods, such as LLM prompting or consistency checking over sampled subsequent steps. This shares the same spirit as training a reward model to represent expansive human feedback in reinforcement learning from human feedback (RLHF, Ouyang et al. 2022).

To tackle under-exploration, we enhance the verifier during both training and inference time to mitigate its variance. Here we focus on scoring-based verifiers, such as a value network or a process reward model (PRM), as they are widely adopted for tree search (Wang et al., 2023; Yu et al., 2024; Tian et al., 2024; Wang et al., 2024d). One major cause of high variance for such verifiers may be the noisy training signals acquired from Monte Carlo (MC) sampling, as the sample size is small compared to the vast space of possible trajectories. Drawing inspiration from Temporal Difference (TD) learning (Sutton, 1988), we enhance the verifier during *training* by incorporating an adjusted λ-return. This effectively balances short-term and long-term rewards, thereby reducing the variance. Furthermore, we propose verifier ensembling during *inference*, which aggregates multiple scores from different verifiers. This aggregation effectively averages out the individual biases and variances of each verifier, resulting in a more consistent and reliable estimate.

We conduct experiments on four representative search algorithms: Best-First Search (BFS, §2.1), Beam Search (Xie et al., 2024; Zhu et al., 2024), A* Search (Wang et al., 2024d,c), and Monte Carlo Tree Search (MCTS, Feng et al. 2023; Tian et al. 2024). Our experimental results on GSM8K (Cobbe et al., 2021), GSM-Plus (Li et al., 2024), and MATH (Hendrycks et al., 2021) demonstrate that our methods can substantially improve both accuracy and efficiency. Further analyses reveal that state merging facilitates more efficient search by preventing the over-exploration of redundant states. Concerning score variance reduction, both TD(λ) in training and verifier ensembling in inference contribute to generating more accurate and stable scores, with their combination yielding further improvements.

2 Pilot Study

This section aims to analyze the over-exploration and under-exploration issues in tree search reasoning. We first introduce definitions and experiment setup, taking Best-First Search (BFS) as an example (§2.1). Then, we conduct comprehensive analyses about these two issues on the popular GSM8K (Cobbe et al., 2021) benchmark (§2.2).

2.1 Definitions and Setup

We formulate the search process of a reasoning problem q as a tree search task. Each tree node (state) $s_i = q, o_1, \ldots, o_i$ represents a partial or full reasoning process, where o_i denotes the i-th reasoning step. Most tree search algorithms follow the iteration of two operations: *selection* and *expansion*. For clarity, we introduce the BFS as a typical example of tree search algorithms.

In the selection phase, BFS selects the most promising node s_i from the set of unexplored nodes $\mathbf{s}$ in the search space based on the feedback from a verifier $v(*)$:

$$i = \arg \max_{1 \le j \le |\mathbf{s}|} v(s_j). \quad (1)$$

Then, in the expansion phase, N (expansion size) child nodes are expanded from the selected node by sampling corresponding new reasoning steps from the LLM (or referred to as policy) $\pi(*)$. Each step is sampled via nucleus sampling (Holtzman et al., 2020) with a temperature τ:

$$o_{i+1} \sim \pi(s_i). \quad (2)$$

By iteratively choosing and expanding the most promising node, BFS efficiently navigates toward the goal state, making it particularly effective for optimization problems.

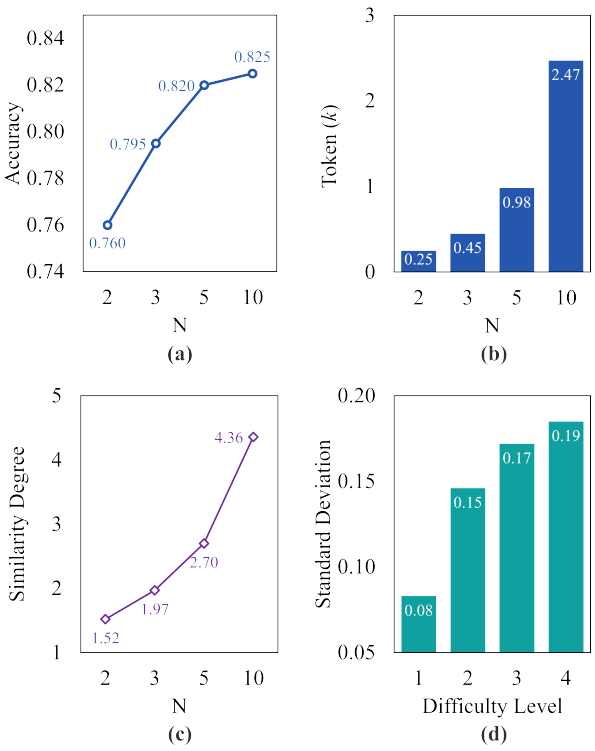


Figure 2: Pilot experiment results using BFS on GSM8K, including: (a & b) Performances and costs when using different expansion size N, where the averaged number of generated token #Token (k) is used to estimate the computational cost. (c) The averaged similarity degree of N sub-nodes from 1000 randomly selected non-leaf nodes, which affects the severity of over-exploration. (d) Standard deviation of verifier scores for 10 sampled correct trajectories from each question at different difficulty levels. High-variance scores can lead to under-exploration of promising nodes.

2.2 Results and Analyses

We adopt LLaMA-3-8B (Meta, 2024), fine-tuned on the GSM8K training set, as the policy. Following Tian et al. (2024), we construct a value network by equipping LLaMA-3-8B with a regression head to serve as the verifier for BFS. Results on the GSM8K test set with different expansion sizes N are shown in Fig. 2(a) and Fig. 2(b).

As N increases, BFS consistently achieves better performance. However, the computational costs rise dramatically. When comparing $N=5$ and $N=10$, there is a $3\times$ to $9\times$ increase in token consumption compared to greedy decoding, yet the improvement in accuracy is merely 0.5%. This raises concerns about the inefficiency of tree search algorithms in addressing search tasks within larger search spaces (i.e., larger N) for LLM reasoning.

A closer examination of the different problem difficulty categories in Fig. 1 reveals that the computational costs are quite similar across the categories, contradicting the intuition that resource allocation should increase with growing difficulty. This indicates that current algorithms struggle to balance resource allocation, suffering from over-exploration in basic problems and under-exploration in complex cases. Upon analyzing the constructed search trees, we discover two critical issues contributing to these problems.

Redundant Search States Result in Over-Exploration Due to the uncontrollable sampling of steps from the policy model, a large number of nodes convey identical semantic meanings (e.g., "*3+4=7*" and "*4+3=7*"). Expanding such nodes leads to redundant exploration.

To further investigate this, we randomly select 1,000 non-leaf nodes from BFS trees, expand $N=2,3,5,10$ sub-nodes for each, and apply agglomerative clustering on their vanilla SimCSE (Gao et al., 2021) embeddings. We then define the overall similarity degree for a group of sub-nodes as $\frac{N}{C}$, where C represents the cluster number. As shown in Fig. 2(c), the overall similarity degree increases linearly with the growth of N. This indicates that the over-exploration issue can be more severe when taking a larger N for higher accuracy.

High-Variance Verifier Scores Lead to Under-Exploration We observe that verifier scores often lack stability. For instance, even though both trajectories lead to the correct answer, their scores may differ greatly. As depicted in Fig. 2(d), we illustrate the standard deviation of scores for 10 sampled correct trajectories for each math problem across various difficulty levels[1]. This highlights the high variance of verifier scores, especially for challenging problems. Guided by inaccurate scores, the search process might be trapped in a locally optimal branch, leaving many valuable paths unexplored and resulting in their under-exploration.

This issue can be attributed to the high-variance training objective of the verifier produced by Monte Carlo (MC) rollouts, which has been thoroughly discussed in reinforcement learning theory (Sutton, 2018). Besides, due to the high costs of rolling out, the number of rollouts is often limited, which further increases the randomness of MC, especially for more difficult math problems where the policy is uncertain about the answer.

[1]It is measured by the frequency of wrong answers within 64 sampled trajectories following Wang et al. (2024c).

3 FETCH

We propose redundant state merging and score variance reduction to address the over-exploration and under-exploration issues analyzed in the pilot study. Both methods are complementary and orthogonal with widely used tree search algorithms.

3.1 Handling Over-Exploration via Redundant State Merging

This method aims to merge semantically equivalent states to avoid over-exploration of them. In the following subsection, we first introduce how to achieve this by utilizing a clustering algorithm and the step representations produced by an embedding model. Then, we further investigate post-training the embedding model for better clustering.

3.1.1 Semantic-Based State Merging

As shown in Fig. 3, state merging is implemented after each node expansion. For newly created nodes $s_1, \ldots, s_N$, we first employ an embedding model, such as SimCSE (Gao et al., 2021), to encode their reasoning processes. Then, we utilize agglomerative clustering to partition them into M groups:

$$\begin{aligned} \bar{\mathbf{s}}_1, \ldots, \bar{\mathbf{s}}_M &= \texttt{Agglomerative}(e_1, \ldots, e_N), \\ e_1, \ldots, e_N &= \texttt{Emb}(s_1), \ldots, \texttt{Emb}(s_N). \end{aligned} \tag{3}$$

In particular, we choose agglomerative clustering as it dynamically determines the cluster number based on the linkage distance threshold d between clusters. This offers a more flexible and adaptive solution, eliminating the need to pre-specify the number of clusters for accurate clustering.

Next, each group $\bar{\mathbf{s}}_i$ is used to create a new hyper-node, comprising multiple original nodes $\bar{\mathbf{s}}_i = \{s^i_1, \ldots, s^i_{|\bar{\mathbf{s}}_i|}\}$. The corresponding verifier score $\bar{v}_i$ is calculated from the original nodes represented by the hyper-node: $\bar{v}_i = f(v^i_1, \ldots, v^i_{|\bar{\mathbf{s}}_i|})$, where f can be $\texttt{max}(*)$, $\texttt{avg}(*)$, or $\texttt{min}(*)$.

When expanding a hyper-node, we sequentially choose original nodes in the order of their verifier scores to obtain child nodes. This facilitates the sampling of more diverse actions, thereby expanding the search space more effectively.

3.1.2 Embedding Model Fine-Tuning

Although we can directly adopt a pretrained embedding model for producing step embeddings, this results in sub-optimal clustering because of the domain discrepancy between the general corpus and target content. For instance, we notice that SimCSE is often insensitive to numbers, which are critical for the math domain. To address this issue, we propose two alternative data collection approaches to post-train the embedding model: *prompting-based* and *consistency-based*.

Prompting-Based Approach This approach leverages the instruction-following capability of LLMs for semantic equivalence judgment. Given reasoning states s_i and s_j, we formulate the verification task as $\texttt{LLM}(I, s_i, s_j)$, where I denotes the prompt provided in Appendix A. Though conceptually straightforward, its effectiveness hinges on the availability of a general LLM and a carefully engineered prompt.

Consistency-based Approach Inspired by the empirical finding that the same states often yield similar actions, we propose to craft labels for s_i, s_j by comparing their impact on subsequent actions when swapped. Formally, given K rollouts $\mathbf{a}_1, \ldots, \mathbf{a}_K$ sampled either from s_i or s_j, we have

$$\delta = \mathbb{E}_{k \in 1 \ldots K}(|p(\mathbf{a}_k \mid s_i; \pi) - p(\mathbf{a}_k \mid s_j; \pi)|), \tag{4}$$

where δ quantifies the influence of swapping s_i and s_j on rollout predictions of the policy π. Intuitively, the smaller the δ, the more similar the states are. Thus, we consider s_i and s_j as identical when $\delta < \alpha$ and distinct when $\delta > \beta$, where α and β are hyperparameters used to control the label quality.

After obtaining the labels of state pairs from the training corpus, we post-train the model using the standard cross-entropy loss:

$$\mathcal{L} = y \log g(e_i, e_j) + (1 - y) \log(1 - g(e_i, e_j)), \tag{5}$$

where $g(*, *)$ is the cosine similarity function and $y \in \{0, 1\}$ denotes state equivalence.

3.2 Handling Under-Exploration via Score Variance Reduction

Previous studies (Wang et al., 2023; Tian et al., 2024) train their verifiers with inspiration drawn from the value function, which seeks to approximate the expected cumulative reward starting from a state s and following a policy model π thereafter. This can be represented as $v(s) = \mathbb{E}_\pi [G_i \mid s_i = s]$, where G_i is the discounted return starting from state s_i and can be estimated using MC methods: $G_i \approx \frac{1}{\rho} \sum_{j=1}^{\rho} [\mathbf{a}_j \text{ is correct}]$, where $\mathbf{a}_j \sim \pi(s_i)$ and ρ is the number of rollouts. As discussed in our pilot study, the high variance of verifiers trained in this way can mislead the search process, thus

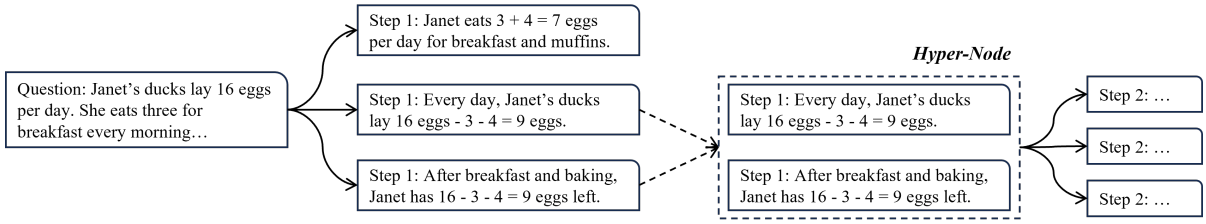


Figure 3: Illustration of redundant state merging. When new nodes are expanded, we merge semantically equivalent nodes into hyper-nodes using agglomerative clustering based on their embeddings.

raising the under-exploration issue. To address it, we propose leveraging *TD(λ)* and *Ensembling* to reduce the score variance during training and inference, respectively.

Training Time TD(λ) is a reinforcement learning algorithm blending key aspects from both MC methods and standard TD learning. In contrast to MC methods that rely on the outcome reward only, TD(λ) balances immediate and delayed rewards by facilitating the allocation of credit for a reward back to previous states and actions. Here, λ refers to the trace decay parameter, with $0 \leqslant \lambda \leqslant 1$, allowing for a flexible trade-off between bias and variance in the estimates. Formally, this process can be formulated as

$$G_i^\lambda = (1-\lambda) \sum_{t=1}^{T-i-1} \lambda^{t-1} G_i^{(t)} + \lambda^{T-i-1} G_i, \quad (6)$$

where T is the total number of steps and $G_i^{(t)}$ is equal to $v(s_{i+t})$. Then, for state s_i, we train the verifier utilizing the standard mean-square error (MSE) loss:

$$\mathcal{L}_{\mathrm{TD}(\lambda)} = (\min(G_i^\lambda, 1) - v(s_i))^2, \quad (7)$$

where the λ-return G_i^λ is constrained within the interval $[0, 1]$ to satisfy the requirements of certain search algorithms.

Inference Time To further reduce the variance and bias arising from TD learning, we propose ensembling multiple verifiers during the inference time. This approach can also be employed to avoid the need for training when open-source verifiers are accessible. Given N_{vn} verifiers, the ensemble score for state s is the average of their predictions: $\frac{1}{N_{vn}} \sum_{i=1}^{N_{vn}} v_i(s)$. To avoid the impact on speed caused by running multiple verifiers serially, we deploy them in parallel on different devices.

4 Experiment

4.1 Setup

Datasets We conduct experiments on three popular test sets: GSM8K (Cobbe et al., 2021), GSM-Plus (Li et al., 2024), and MATH (Hendrycks et al., 2021). GSM8K contains 1,319 grade school math problems taking between 2 and 8 steps to solve. GSM-Plus is an enhancement of GSM8K by introducing 8 variations. We randomly sample one variant for each test case, resulting in 1,319 instances. MATH consists of more challenging math problems from high school math competitions. Following previous work (Lightman et al., 2023), we test on a subset of 500 cases, named MATH500.

Models and Hyperparameters For GSM8K and GSM-Plus, the policy and value network are fine-tuned from LLaMA-3-8B (Meta, 2024) on the GSM8K training set. For MATH, we employ the policy released by Wang et al. (2023), which is based on Mistral-7B (Jiang et al., 2023), and adopt a PRM as the verifier due to its better performance on MATH. Detailed parameter settings for training and search are reported in Appendix A and B.

Baselines We report conventional greedy decoding and self-consistency (Wang et al., 2022, 2024a,b), which only adopt the policy for inference. Guided by verifiers, the sampling-based approach can be further enhanced using Best-of-N and Weighted Voting. These methods select the solution with the highest score or ensemble the predicted answers by considering their respective scores as weights. For tree search algorithms, we consider the most popular choices: Beam Search (Xie et al., 2024; Zhu et al., 2024) and MCTS (Feng et al., 2023; Tian et al., 2024). We also experiment with the typical BFS (§2.1) and recent LiteSearch (Wang et al., 2024c), which is an optimized implementation of A* (Nilsson et al., 1984) that carefully manages the expansion budget to speed up the search process.

	GSM8K		GSM-Plus		MATH	
	Accuracy ↑	#Token (k) ↓	Accuracy ↑	#Token (k) ↓	Accuracy ↑	#Token (k) ↓
Greedy Decoding	.657	0.08	.500	0.09	.302	0.25
Self-Consistency	.753	0.83	.601	0.95	.340	2.44
Best-of-N	.819	0.83	.634	0.95	.352	2.44
Weighted Voting	.817	0.83	.620	0.95	.372	2.44
BFS	.824	2.47	.655	2.84	.382	6.51
w/ State Merge	.836	0.88	.664	1.12	.380	**2.89**
w/ Var Reduce	.847	2.25	.676	2.89	**.388**	7.63
w/ FETCH	**.852**	**0.87**	**.684**	**1.09**	.384	3.04
Beam Search	.828	1.55	.669	1.88	.378	6.50
w/ State Merge	.832	**1.17**	.674	**1.61**	.384	**6.36**
w/ Var Reduce	**.846**	1.58	.680	1.89	.386	7.45
w/ FETCH	.842	1.19	**.682**	1.62	**.396**	7.04
LiteSearch	.818	0.55	.660	0.92	.372	2.68
w/ State Merge	.825	0.48	.657	**0.69**	.378	**1.58**
w/ Var Reduce	**.838**	0.53	.670	0.89	.380	3.53
w/ FETCH	.837	**0.46**	**.677**	**0.69**	**.384**	2.00
MCTS	.838	10.7	.676	14.1	.374	59.0
w/ State Merge	.845	3.03	.670	**3.43**	.380	**20.3**
w/ Var Reduce	**.854**	8.84	.686	14.4	.394	66.6
w/ FETCH	.853	**2.60**	**.688**	3.53	**.400**	22.9

Table 1: Main test results. We emphasize the best results in **bold** and the second-best ones with underlining.

Evaluation Metrics We report the accuracy of answers (**Accuracy**) to evaluate performance and follow (Kang et al., 2024; Wang et al., 2024c) to adopt the averaged number of generated tokens (**#Token** (k)) to estimate computational costs.

4.2 Main Results

Table 1 shows the main test results on the three datasets. Generally, leveraging verifiers effectively improves performance over conventional methods relying only on the policy. Besides, comparing these tree search methods, we can usually observe the performance gains when scaling the inference computation. When applying our methods to these algorithms, we draw the following conclusions.

State Merging Effectively Decreases Computational Costs While Maintaining Performance State merging consistently reduces computational costs across all algorithms and datasets by enabling smarter node selection, which prevents redundant states from over-exploration. For example, in BFS applied to GSM8K, state merging reduces #Token (k) from 2.47 to 0.88, a nearly *3× reduction* in computational overhead. Beyond efficiency gains, we observe accuracy improvements in certain cases, even when fewer nodes are explored. This suggests that merging redundant states allows algorithms to allocate their computation budget more effectively, prioritizing promising paths over unproductive ones.

Among the four search algorithms tested, Beam Search shows the smallest token savings. This stems from its inherent rigidity: it strictly expands a fixed number of nodes (determined by the beam size), limiting opportunities for optimization. Nevertheless, Beam Search still benefits from state merging, as avoiding redundant node exploration enhances accuracy by redirecting resources toward more valuable states.

Score Variance Reduction Substantially Improves Search Performance The application of score variance reduction leads to consistent performance gains across all tested algorithms, improving accuracy by approximately 1∼2 points. Notably, this method maintains stable token consumption on the GSM8K and GSM-Plus datasets, suggesting it efficiently allocates computational resources to valuable nodes rather than aimlessly switching trajectories, thereby overcoming the under-exploration issue.

However, on the more challenging MATH dataset, we observe a slight increase in computational costs. This is likely due to the inherent difficulty of MATH problems, where search algorithms often struggle to converge on confident solutions. To avoid under-exploration, our method would allocate more computations on unfinished trajectories as demonstrated by results in Fig. 8.

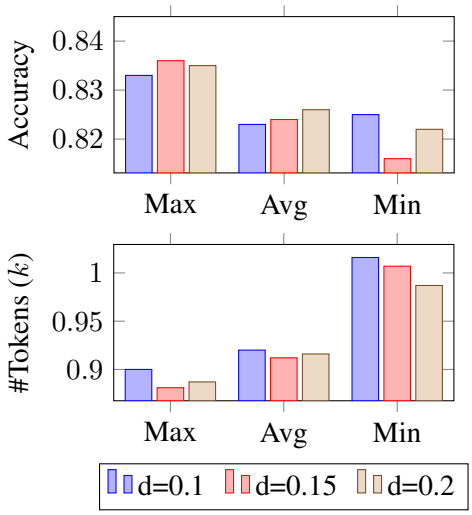


Figure 4: Ablation studies on parameter selection of f and d for state merging.

	Accuracy ↑	#Token (k) ↓
BFS (w/o state merge)	.824	2.47
BFS + Agglomerative		
Pretrained SimCSE	.827	1.06
w/ Prompt. FT	.835	**0.88**
w/ Consist. FT	**.836**	**0.88**
BFS + k-means (SimCSE w/ Consist. FT)		
k=2	.832	0.91
k=4	.830	1.05

Table 2: Ablation studies of using different clustering algorithms and post-training strategies for state merging.

FETCH Mitigates Over- and Under-Exploration By combining the two methods, we validate our approach enables more efficient and effective searching. Furthermore, we observe better resource allocation for problems of varying difficulties, as demonstrated in Fig. 8. The results on both the GSM8K and MATH datasets indicate that our methods not only significantly reduce the token consumption for simple tasks but also ensure sufficient exploration of complex ones. This strikes a balance between the problems of over-exploration and under-exploration, thereby enhancing the practicality of the tree search algorithms.

4.3 Ablation Studies and Analyses

In this part, we investigate the hyperparameter selection of our methods and conduct in-depth analyses of the working mechanism, mainly on the GSM8K dataset using the BFS algorithm.

Influences of Hyper Parameter Selection for State Merging In Fig. 4, d and f determine the distances of nodes within each cluster during clustering and the verifier scores of the clustered hyper-nodes. Increasing d results in fewer clusters but higher probabilities of erroneously grouping different states into one cluster. However, we notice our method is robust to this parameter. For example when $f = \texttt{max}(*)$, there is limited influence on both performance and efficiency by ranging d from 0.1 to 0.2. For f, the best choice is $\texttt{max}(*)$ regarding both Accuracy and #Token (k) among all examined values of d. It might be because it better guarantees that the promising state (with a higher score) can be explored first.

	Accuracy ↑	#Token (k) ↓
BFS + State Merge		
w/ MC	.836	0.88
w/ TD(λ)	.843	0.89
w/ MC+Ensem	.848	0.88
w/ TD(λ)+Ensem	**.852**	**0.87**

Table 3: Ablation studies of score variance reduction strategies.

In Table 2, we also compare different clustering algorithms. We notice that agglomerative clustering works better than k-means. Partial reason is that k-means requires specifying the number of clusters in advance, resulting in inflexible clustering and higher clustering error rates. Nevertheless, k-means is also significantly better than the "w/o state merge" setting, proving the necessity of merging redundant states.

Influences of Post-Training Embedding Models We compare different strategies to post-train the embedding models in Table 2. Prompting-based and consistency-based approaches achieve competitive performance, and both of them significantly outperforms the baseline, which only adopts the pretrained SimCSE for clustering. To further validate these findings, we conduct a human evaluation to examine the correlation between similarity scores produced by these embedding models and human judgments in Appendix D. Results in Table 5 again demonstrate the effectiveness of our post-training approaches. Notably, the traditional rule-based method, edit distance, is very sensitive to dataset variations, limiting its robustness. Besides, scaling the fundamental model size from RoBERTa-Base to Large yields measurable performance gains, suggesting future opportunities to advance state merging through more powerful embedding models.

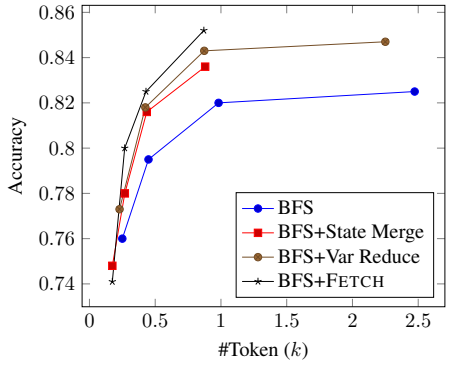


Figure 5: Inference-time scaling for the BFS algorithm equipped with our methods. The expansion budget N is set as $2, 3, 5, 10$.

Comparison of Different Score Variance Reduction Approaches We observe that both TD(λ) and verifier ensembling effectively enhance search performance (Table 3) and improve verifiers for better step-level alignment with final accuracy (Fig. 6). Notably, ensembling shows better performance relative to TD(λ) but at the cost of deploying multiple verifiers. Combining these methods leads to further performance gains, demonstrating their complementary nature. We also illustrate the score variance across problems of various difficulties in Fig. 7, which again validates the effectiveness.

Inference-Time Scaling We validate the performance of our methods when scaling inference computations. As shown in Fig. 5, our methods have a more rapid increase in Accuracy when increasing the expansion budget. Besides, both state merging and variance reduction can lead to 1$\sim$2 times less computational costs with competitive performance. Further combining both methods yields the best overall performance, again validating the conclusion of previous analyses.

5 Related Work

Deductive Reasoning with Search Algorithms LLMs have demonstrated remarkable potential in tackling complex reasoning tasks (Achiam et al., 2023; Touvron et al., 2023; Jiang et al., 2023). However, they frequently fail on even simple reasoning steps, significantly degrading performance. To mitigate this, researchers have proposed inference-time scaling methods that enhance computational effort during the reasoning process before finalizing answers. Early approaches, such as Self-Consistency (Wang et al., 2022), scaled computations by sampling multiple solutions. This approach is inefficient because it requires exploring full solution paths, even if a mistake has occurred early on. Later work introduced advanced tree search algorithms, including Beam Search (Yao et al., 2024; Zhu et al., 2024; Yu et al., 2024), MCTS (Tian et al., 2024; Zhang et al., 2024), and A* (Wang et al., 2024d,c), to strategically allocate computational resources to critical reasoning steps, thereby improving efficiency.

Despite these advancements, most existing methods fail to address the challenges of over-exploration and under-exploration, leading to sub-optimal efficiency. To our knowledge, only Tian et al. (2024) explicitly tackle over-exploration by pruning repeated reasoning states. However, their solution relies on either edit distance, which is prone to robustness issues (Table 5), or LLM-based similarity checks, which incur significant computational overhead. This underscores the absence of a lightweight and robust mechanism to address this problem.

Training Verfiers to Guide Searching Previous studies have mainly utilized two types of verifiers: value networks (Yu et al., 2024; Tian et al., 2024) and process reward models (PRMs, Lightman et al. 2023; Wang et al. 2023). Typically, both approaches leverage LLMs augmented with classification or regression heads to produce scalar scores to guide searching.

To collect training labels, Lightman et al. (2023) employed human annotators to identify reasoning errors in intermediate steps. However, this approach is cost-prohibitive, hindering the scalability of training datasets. Therefore, subsequent research (Yu et al., 2024; Wang et al., 2023; Tian et al., 2024; Wang et al., 2024c) has instead adopted MC methods to automate label collection. While effective, they overlook the high variance associated with MC estimation, which influences the stability of trained verifiers and can lead to under-exploration. In this study, we are the first to incorporate TD(λ) and verifier ensembling to address this concern.

6 Conclusion

This work introduces FETCH, which addresses the over-exploration and under-exploration issues faced by tree search reasoning algorithms in LLMs by introducing two novel enhancements: state merging and variance reduction. By mitigating redundant search states and stabilizing verifier score estimation, these methods improve the practical

feasibility of implementing advanced reasoning strategies, significantly enhancing accuracy and efficiency. Our experiments demonstrate the potential of these techniques to refine the deductive reasoning capabilities of LLMs without incurring prohibitive computational costs. Future explorations may further optimize these strategies and extend them to more complex tasks and domains.

Discussion on Integrating Tree Search with Large Reasoning Models With the rapid advancement of this field, significant progress has been achieved very recently by leveraging reinforcement learning to train large reasoning models (LRMs, Guo et al. 2025; Team et al. 2025; El-Kishky et al. 2025). In this study, however, we exclude LRMs from our analysis. Unlike traditional LLMs, LRMs generate long chains of thought (CoTs), which can be interpreted as linearized representations of search trajectories across search trees (Xiang et al., 2025). The integration of tree search algorithms with LRMs presents a promising avenue for improving the management of search states and addressing the efficacy challenges inherent to these models (Chen et al., 2024; Yan et al., 2025; Wang et al., 2025). However, efficiently segmenting long CoTs into manageable fragments remains an under-explored challenge. Furthermore, the heuristic functions employed by search algorithms require a more advanced verification mechanism to evaluate the complex search states of long CoTs. Given these limitations, this avenue warrants further investigation to unlock its full potential.

Limitation

As shown in Fig. 5, the Pearson Correlation Coefficient (PCC) scores of predicted similarities from our embedding models, when evaluated against human annotations on MATH, are much lower compared to those on GSM8K. This discrepancy likely arises because reasoning steps in MATH problems are typically longer and more complex, resulting in less accurate clustering and suboptimal final performance. To address this limitation, one potential approach is to employ larger-scale embedding models and curate higher-quality post-training data to enhance their capabilities.

Ensembling multiple verifiers can improve the quality of scores as shown in Table 3. Even though this method marginally impacts search speed when deployed separately, it also increases computational resource consumption, which is not eco-friendly. Consequently, distilling the capabilities of ensemble models into a single, efficient model can be a promising way for further improvement.

In this study, we follow common practices by testing on mathematical tasks. Nevertheless, over- and under-exploration can occur in a variety of tasks. Extending FETCH to other reasoning tasks would thus hold great value. We leave these explorations as future work.

Acknowledgments

Ante Wang and Jinsong Su were also supported by National Key R&D Program of China (No. 2022ZD0160501), Natural Science Foundation of Fujian Province of China (No. 2024J011001), and the Public Technology Service Platform Project of Xiamen (No. 3502Z20231043). We also thank the reviewers for their insightful comments.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of SemEval 2014*, pages 81–91.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of IWP 2005*.

Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, et al. 2025. Competitive programming with large reasoning models. *arXiv preprint arXiv:2502.06807*.

Xidong Feng, Ziyu Wan, Muning Wen, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like

tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of EMNLP 2021*, pages 6894–6910.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Proceedings of NeurIPS 2021*.

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

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jikun Kang, Xin Zhe Li, Xi Chen, Amirreza Kazemi, and Boxing Chen. 2024. Mindstar: Enhancing math reasoning in pre-trained llms at inference time. *arXiv preprint arXiv:2405.16265*.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *arXiv preprint arXiv:2402.19255*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Nils J Nilsson et al. 1984. *Shakey the robot*, volume 323. Sri International Menlo Park, California.

OpenAI. 2024. Openai o1 system card.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *Proceedings of NeurIPS 2022*, pages 27730–27744.

Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44.

Richard S Sutton. 2018. Reinforcement learning: An introduction. *A Bradford Book*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. 2024. Toward self-improvement of llms via imagination, searching, and criticizing. *arXiv preprint arXiv:2404.12253*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ante Wang, Linfeng Song, Baolin Peng, Lifeng Jin, Ye Tian, Haitao Mi, Jinsong Su, and Dong Yu. 2024a. Improving llm generations via fine-grained self-endorsement. In *Findings of ACL 2024*, pages 8424–8436.

Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Lifeng Jin, Haitao Mi, Jinsong Su, and Dong Yu. 2024b. Self-consistency boosts calibration for math reasoning. In *Findings of EMNLP 2024*, pages 6023–6029.

Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Dian Yu, Haitao Mi, Jinsong Su, and Dong Yu. 2024c. Litesearch: Efficacious tree search for llm. *arXiv preprint arXiv:2407.00320*.

Chaojie Wang, Yanchen Deng, Zhiyi Lyu, Liang Zeng, Jujie He, Shuicheng Yan, and Bo An. 2024d. Q*: Improving multi-step reasoning for llms with deliberative planning. *arXiv preprint arXiv:2406.14283*.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. 2023. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *arXiv preprint arXiv:2312.08935*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of ICLR 2022*.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, et al. 2025. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-though. *arXiv preprint arXiv:2501.04682*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2024. Self-evaluation guided beam search for reasoning. In *Proceedings of NeurIPS 2024*.

Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. 2025. Inftythink: Breaking the length limits of long-context reasoning in large language models. *arXiv preprint arXiv:2503.06692*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of NeurIPS 2024*.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2024. Ovm, outcome-supervised value models for planning in mathematical reasoning. In *Findings of NAACL 2024*, pages 858–875.

Longhui Yu, Weisen Jiang, Han Shi, YU Jincheng, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. In *Proceedings of ICLR 2023*.

Di Zhang, Jiatong Li, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*.

Tinghui Zhu, Kai Zhang, Jian Xie, and Yu Su. 2024. Deductive beam search: Decoding deducible rationale for chain-of-thought reasoning. *arXiv preprint arXiv:2401.17686*.

A Parameter Settings

For GSM8K, the policy model is fine-tuned from Llama-3-8B using standard cross-entropy loss. We train the model for 2 epochs, adopting the AdamW optimizer with a linear scheduler using an initial learning rate of 5e-6. The value network is trained on sampled rollouts from the policy, following previous studies (Yu et al., 2024; Tian et al., 2024). Specifically, for each question, we sample 16 rollouts with a temperature of 1 only from the initial state (i.e., $s_0 = q$). This model is trained for 1 epoch using typical mean square error loss with a learning rate of 2e-6.

For MATH, we directly adopt the policy model and PRM training data released by (Wang et al., 2023). The policy model is based on Mistral-7B (Jiang et al., 2023) trained from MetaMath (Yu et al., 2023). We adopt the same value training script to train our PRM, and the trained PRM shows competitive performance as the official results (Wang et al., 2023).

When state merging, by default, the clustering parameters d is 0.15 and f uses $\texttt{max}(*)$ across all experiments. The embedding models are based on RoBERTa-Large (Liu, 2019) pretrained using SimCSE (Gao et al., 2021). For each dataset, we collect around 100k step pairs to post-train the embedding models using prompting-based or consistency-based methods. The models are trained using binary cross-entropy loss with a learning rate of 1e-6. For the prompting-based method, we adopt Llama-3-8B-Instruct (Meta, 2024), which is trained on extensive post-training data, including publicly available instruction datasets and over 10 million human-annotated examples. The instruction is designed as

> *Given two reasoning steps (Step A and Step B) from two rationales of a math word problem, respectively. Your task is to judge whether these two steps are semantically equivalent.*
> *Step A:* $\{\texttt{STEP}_\texttt{A}\}$
> *Step B:* $\{\texttt{STEP}_\texttt{B}\}$
> *Are Steps A and B semantically equivalent? Answer with Yes or No.*

For the consistency-based method, we set $\alpha = 0.02$, $\beta = 0.08$, and $K = 1, 2$ for GSM8K and MATH, respectively. By default, we adopt the consistency-based method as it does not rely on the other model.

As for score variance reduction, we adopt a combination of TD(λ) and verifier ensembling for GSM8K, where $\lambda = 0.8$ and $N_{nv} = 2$. We only use verifier ensembling with $N_{nv} = 2$ for MATH because more accurate process rewards have been provided by (Wang et al., 2023), which are collected using very heavy MC sampling with enough rollouts for each intermediate state to reduce variance.

B Implementation of Search Algorithms

We report the parameter settings for the search algorithms used in this paper. Except for greedy decoding, we set the generation temperature τ to 0.8 across all experiments. We set the expansion budget $N = 10$ for Self-Consistency, Best-of-N. For BFS, we set $N = 10$ and $N = 5$ for GSM8K

	Accuracy ↑	#Token (k) ↓
Greedy Decoding	.841	0.12
Self-Consistency	.893	1.86
Best-of-N	.857	1.86
Weighted Voting	.899	1.86
Beam Search	.911	1.44
w/ FETCH	**.922**	**0.98**
MCTS (w/o simulation)	.904	2.54
w/ FETCH	**.923**	**1.01**

Table 4: Test results on GSM8K when employing Qwen2.5-32B as the policy model.

	GSM8K	MATH
BM25+	0.211	0.205
Edit Distance	0.307	0.500
SimCSE (RoBERTa-Base)	0.388	0.379
SimCSE (RoBERTa-Large)	0.464	0.414
w/ Prompt. FT	0.872	**0.663**
w/ Consist. FT	**0.878**	0.658

Table 5: Pearson Correlation Coefficient of similarity scores produced by different methods and human annotations on 200 sampled pairwise samples from GSM8K or MATH.

(or GSM-Plus) and MATH, respectively. For BeamSearch, we set both the expansion budget and beam size to 5. For LiteSearch (Wang et al., 2024c), we set the maximum expansion budget $N = 10$ and $N = 5$ for GSM8K (or GSM-Plus) and MATH, and the expected accuracy $\epsilon = 0.95$. For MCTS (Tian et al., 2024), we set the expansion budget $N = 8$ for the root node and $N = 4$ for the others. During simulation, we use a rollout number of 2.

C Experiments on Qwen2.5-32B

To evaluate the efficacy of FETCH on larger models, we conduct experiments using Qwen2.5-32B as the policy model. Due to computational resource constraints, we directly utilize verifiers based on Llama-3-8B, which were trained in the main experiments. For testing, we select the two most widely used methods, Beam Search and MCTS (without simulation to accelerate inference). The results are shown in Table 4, further demonstrating the effectiveness of FETCH.

D Human Evaluation on Embedding Models

We randomly sample 200 step pairs from both GSM8K and MATH and hire annotators with natural language processing backgrounds to manually annotate whether they are the same steps with 0 or 1. Each annotation is double-checked to ensure its correctness. Then, we ask the embedding models to provide similarity scores for them and calculate the Pearson Correlation Coefficient (PCC) with human annotations. As shown in Table 5, we again validate the effectiveness of post-training. Besides, by comparing RoBERTa-Base and Large, we also demonstrate that scaling the model parameter may further enhance the model performance. We leave it for future work.

E Computational Costs for Clustering and Ensembling

In this work, clustering is performed with just a single forward pass of RoBERTa-large, while ensembling can be efficiently achieved by deploying multiple verifiers in parallel. Our experiments using Beam Search show that state merging adds less than 0.2% to the total inference time. Additionally, results from evaluating verifiers on 1,000 randomly selected cases indicate that ensembling increases the overall time by approximately 3% compared to the baseline. Both findings demonstrate that the impact on computational efficiency is negligible.

F Effect of FETCH on Reasoning Paths with Different Lengths

We conducted experiments on GSM8K using BFS and categorized the test cases into different groups according to the number of reasoning steps required for their solutions. Results in Table 6 demonstrate that our method consistently improves performance across all groups, achieving greater gains in both performance and efficiency, particularly for solutions requiring longer reasoning steps.

G Analysis on Variance of Verifier Scores

Temporal Difference (TD) learning reduces variance compared to Monte Carlo (MC) methods by updating predictions incrementally using immediate rewards and bootstrapped estimates (current value predictions) instead of waiting for full-episode returns. This localizes variance to single steps, mitigating the compounding effect of stochastic noise in long trajectories, but introduces bias from imperfect estimates. MC methods eliminate bias but suffer from higher variance, as their dependence on full trajectories accumulates stochastic errors proportional to episode length. The TD(λ) algorithm bridges this trade-off via

	[1,4)		[4,6)		[6,∞)	
	Accuracy	#Token (k)	Accuracy	#Token (k)	Accuracy	#Token (k)
BFS	.898	1.92	.832	2.46	.639	3.74
w/ State Merge	.895	.58	.844	**0.89**	.686	1.59
w/ Var Reduce	**.914**	1.48	.855	2.38	.670	3.46
w/ FETCH	**.914**	**0.55**	**.858**	0.91	**.696**	**1.44**

Table 6: Performance of FETCH on reasoning paths from GSM8K with different step numbers.

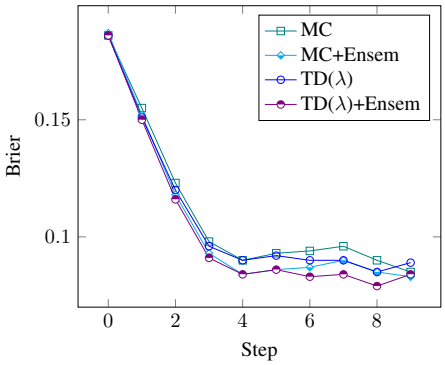


Figure 6: Brier scores of estimated values from different verifiers at different reasoning steps and final accuracy, where brier scores can be calculated via mean square error (MSE).

the parameter λ ($0 \leq \lambda \leq 1$), which interpolates between TD (biased, low-variance) and MC (unbiased, high-variance) updates. By adjusting λ, TD(λ) allows optimization of the bias-variance trade-off based on task-specific needs. In this study, we further propose a verifier ensemble approach to mitigate individual biases and variances by averaging predictions across multiple verifiers. This reduces the frequent path-switching phenomenon during search caused by score variance, thereby addressing the under-exploration issue.

Fig. 7 shows the variance of scores produced from verifiers trained in different methods. We observe both TD(λ) and verifier ensemble have lower variance than the baseline across subsets of various difficulties. Their combination also makes further progress in most cases. These results demonstrate that our methods can effectively reduce variance.

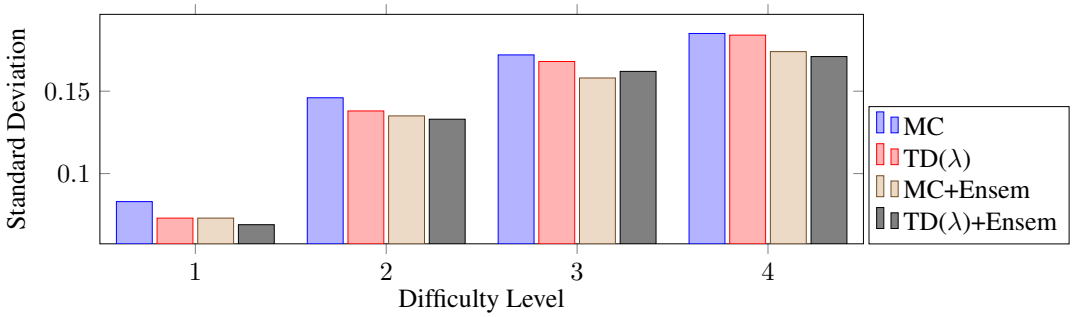
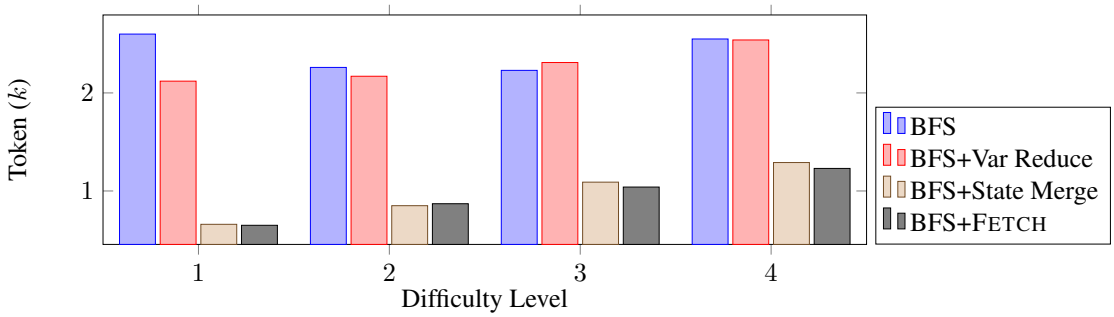
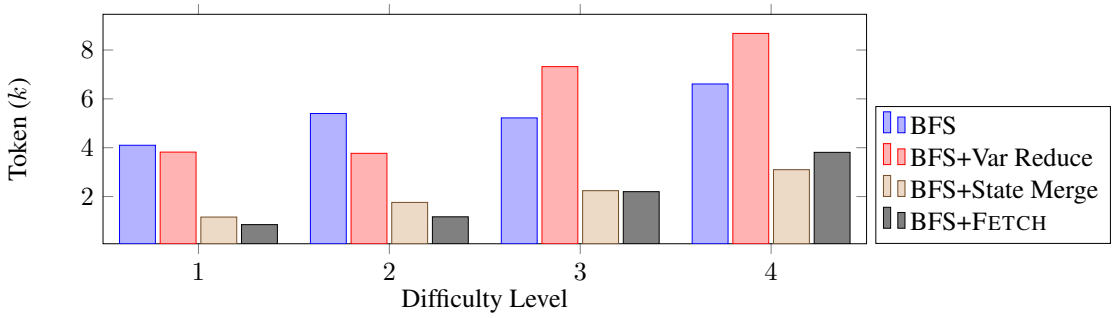


Figure 7: Comparison of the standard deviation of score estimation using different verifiers for 20 sampled reasoning paths of GSM8K questions at different difficulty levels.



(a) GSM8K



(b) MATH

Figure 8: Ablation study of our methods on token consumption across GSM8K and MATH problems of different difficulties when using BFS.