Project Proposal
Chopping Trees: Dynamic Pruning and Optimization Based Tree-of-Thought

Xirui Huang, Joongho Kim, Bhavesh Sharma, Abby Damodaran, Zarreen Reza
Team JOAB
zarreen@algoverseairesearch.org

1 Introduction

1.1 Motivation

Large Language Models (LLMs) have achieved notable success on a wide array of reasoning tasks, particularly when guided by structured prompting strategies. One such approach, Chain-of-Thought (CoT) prompting [Wang and Zhou, 2024], encourages step-by-step reasoning and has been shown to significantly enhance model accuracy on complex problems. Despite its effectiveness, CoT is vulnerable to error propagation, where an early mistake can cascade through subsequent steps and compromise the final output.

To mitigate this issue, Tree-of-Thought (ToT) prompting [Yao et al., 2023] was introduced, allowing the model to explore multiple reasoning trajectories in parallel and select the most promising outcome through tree search. While ToT improves robustness and decision quality, it comes with a substantial computational cost, as the number of branches increases rapidly with tree depth and exploration breadth.

In this work, we present a novel extension to the ToT framework called Semantic Similarity-based Dynamic Pruning (SSDP). Instead of exhaustively expanding all possible reasoning paths, SSDP dynamically prunes the search tree during inference by leveraging semantic similarity to merge redundant branches and applying node-level scoring to discard low-quality candidates. While the underlying components of SSDP—such as similarity-based merging and selective pruning—have been explored individually, their integration into a unified, real-time pruning framework for tree-based reasoning has not been previously realized.

To further improve efficiency, our approach incorporates complementary optimizations including semantic caching, representation compression, parallel branch evaluation, and convergence-based early stopping. Together, these techniques allow SSDP to retain the interpretability and diversity of ToT while significantly reducing its computational footprint.

Our goal is to demonstrate that it is possible to achieve the benefits of tree-structured reasoning, such as robustness and interpretability without incurring prohibitive resource costs. SSDP introduces a scalable and practical framework for structured multi-step reasoning, opening new avenues for efficient, high-fidelity inference in LLMs.

1.2 Contributions

In this work, we present Semantic Similarity-based Dynamic Pruning (SSDP)—a novel and efficient framework for optimizing Tree-of-Thought (ToT) reasoning in Large Language Models. Our approach addresses the computational inefficiencies of traditional ToT methods while preserving their robustness and interpretability. The key contributions of this paper are:

- **Unified Pruning Framework:** We introduce a dynamic pruning strategy that selectively trims unpromising reasoning paths during inference. This reduces unnecessary computation while retaining high-quality branches.
- **Semantic Similarity Merging:** We incorporate semantic similarity metrics to detect and merge redundant branches, effectively reducing duplication across the reasoning tree and improving path diversity.
- **Node-level Scoring Mechanism:** Each reasoning step is evaluated using a node-level scoring function, allowing the system to make fine-grained decisions about which paths to retain or discard.
- **Integrated Optimization Techniques:** To further enhance scalability and efficiency, SSDP combines several state-of-the-art optimization methods:
  - *Semantic caching* to avoid redundant computations,
  - *ChunkKV compression* for memory-efficient tree expansion on single GPUs,
  - *vLLM tree-attention* to enable parallel processing of reasoning branches,
  - *Ember compound-model coordination* to mitigate overthinking and stabilize reasoning behavior,
  - *Convergence-based dynamic stopping* to terminate inference when reasoning has stabilized.

Together, these components form a cohesive system that significantly reduces computational cost while maintaining or improving reasoning accuracy. SSDP establishes a new direction for scalable, structured reasoning in LLMs by unifying structural pruning, semantic understanding, and systems-level optimization.

2 Related Work

While ToT allows models to explore multiple reasoning paths simultaneously, many existing methods rely on Monte Carlo Tree Search (MCTS) to expand and evaluate paths. These methods, though robust, can be computationally inefficient, particularly when they continue expanding low-probability paths until predefined limits are reached Yao et al. [2023].

Dynamic Parallel Tree Search (DPTS) Ding et al. [2025] improves the scalability of ToT by enabling fine-grained parallel reasoning. It introduces two key innovations: a *Parallelism Streamline* that allows asynchronous expansion of reasoning paths with contextual alignment, and a *Search-and-Transition* module that halts low-confidence paths early while extending promising ones. This architecture significantly reduces inference time while preserving accuracy.

Semantic Self-Consistency Knappe et al. [2025] enhances the reliability of CoT and ToT by aggregating not only final answers but also intermediate steps based on semantic similarity. This method reweights chains to prioritize semantically aligned rationales, improving performance on tasks that require nuanced reasoning.

FETCH: Semantic State Merging Wang et al. [2025] addresses over-exploration and under-exploration in ToT search spaces. FETCH identifies semantically redundant states using embedding-based similarity and merges them during inference. This reduces duplication in the tree and stabilizes path evaluation by ensemble verification, improving both efficiency and consistency.

Pruning in Long CoT Zhao et al. [2025] explores the impact of pruning different parts of long reasoning chains. By representing reasoning paths as logic graphs and removing superfluous steps, they show that pruning verification or auxiliary components can actually improve performance, while aggressive pruning of core reasoning steps can be detrimental. This motivates selective pruning strategies that balance conciseness with logical integrity.

Spectrum-Preserving Token Merging (PiToMe) Tran et al. [2024] proposes a compression strategy for transformer inference by merging similar token representations. Their technique significantly reduces FLOPs without compromising model accuracy, and provides inspiration for our compression and memory efficiency goals in reasoning tree traversal.

Chain-of-Draft (CoD) Xu et al. [2025] introduces a more efficient alternative to CoT, where models produce condensed "draft" reasoning steps instead of verbose chains. CoD achieves comparable

accuracy using less than 10% of the tokens typically required by CoT, showing that stepwise reasoning can be effective even with lightweight representations. We leverage this approach at the node level to limit token count and facilitate semantic similarity comparisons.

Monte-Carlo Graph Search for AlphaZero Czech et al. [2020a] extends traditional MCTS by implementing PUCT scoring mechanisms within directed acyclic graph structures rather than trees. Their approach addresses memory limitations in AlphaZero's search by storing Q-values on both edges and nodes, enabling transposition table usage while preventing information leaks. The work demonstrates PUCT's effectiveness in perfect-information games but focuses on game-playing contexts rather than language model reasoning applications.

Empirical Analysis of PUCT Algorithm Matsuzaki [2018] provides comprehensive evaluation of PUCT performance across different evaluation function qualities in strategic game environments. The study reveals how exploration-exploitation balance varies with evaluation function accuracy and demonstrates PUCT's superiority over standard UCT in scenarios with high-quality priors. However, the analysis remains confined to traditional game-playing domains without addressing computational efficiency challenges in transformer-based reasoning.

Entropy-Guided MCTS Enhancement Zhou et al. [2024] introduces information-theoretic improvements to AlphaZero by incorporating entropy-based action selection criteria alongside traditional PUCT scoring. Their *Entropy-Guided MCTS* balances immediate rewards with informational value through game state entropy metrics, showing improved performance in complex strategic scenarios. The approach enhances tree navigation efficiency but operates within conventional MCTS frameworks without semantic similarity considerations.

While existing methods individually tackle tree expansion, semantic coherence, or computational cost, our work differs by unifying all these components into a single framework. Specifically, we integrate dynamic pruning, semantic merging, node-level scoring, and system-level optimization strategies (e.g., semantic caching, ChunkKV compression, vLLM tree attention, Ember coordination, convergence-based early stopping). Unlike standard MCTS-based methods, our framework discards low-value nodes early, dynamically adjusts tree depth, and prunes redundant paths, resulting in both depth-wise and breadth-wise optimization. This yields a highly efficient, semantically consistent, and scalable approach to LLM reasoning.

3 Methodology

We propose **Semantically Similarity-based Dynamic Pruning** (SSDP), a novel framework designed to improve efficiency and robustness in ToT traversal. The key idea behind SSDP is to merge semantically similar reasoning paths during traversal and to dynamically prune low-quality branches, thereby reducing redundant computation while maintaining the diversity necessary for effective reasoning.

This section outlines our initial approach, which may evolve as we refine our implementation and evaluate empirical performance across tasks.

3.1 Preliminary Design

Our current design for SSDP consists of two primary mechanisms:

- **Semantic Similarity Merging**: Inspired by FETCH's semantic state merging [Wang et al., 2024], we address the problem of redundant exploration of semantically equivalent but syntactically different reasoning paths by comparing nodes for semantic similarity at each tree level, and merging those that are closely aligned. This is intended to compress the reasoning space and reduce duplicate paths.
- **Dynamic Confidence-based Pruning**: Nodes with low confidence (or high entropy) are pruned during expansion. In addition, we consider pruning parent nodes whose children are predominantly low-quality.

Traversal proceeds in a breadth-first manner. At each step, we prioritize exploration of high-confidence nodes and defer or discard paths with low promise.

3.2 Key Components

- **Parallel Node Expansion**: From each current node, k child nodes will be generated in parallel using a decomposition-based prompting strategy (e.g., Chain-of-Thought). The specific generation strategy may be adjusted for efficiency or coherence.
- **Similarity-Based Merging**: Nodes at the same level will be compared using a semantic similarity function $\text{Sim}(a, b)$. If their similarity exceeds a threshold σ, they will be merged. The merging function (e.g., how we aggregate confidence or merge traces) is an area for ongoing experimentation.
- **Confidence-Guided Pruning**: Each node is assigned a score $S(n)$ based on a measure of quality, such as model-generated confidence or entropy. Nodes with $S(n) < \tau$ are pruned. We are exploring heuristics to determine when pruning a parent node (due to low-quality children) is justified.
- **Priority Queue Traversal**: Nodes are added to a processing queue in descending order of $S(n)$, enabling a best-first traversal without rollback. Alternative queueing strategies (e.g., incorporating exploration bonuses) are also under consideration.
- **Scoring Strategy**: To score branches, we can leverage Preference Learning to train a separate scoring model, inspired by the MCTS paper. We first run a regular ToT algorithm and seeing which reasoning steps yield the best result. We can then take those reasoning steps to train a smaller separate "scoring model" that will then be used to grade each step (0 to 1) during inference on another model. The UCT (Upper Confidence for Trees) scoring strategy will also be explored, where the mathematical scoring formula ranges from 0 to 1, and scores a branch based on a mix of exploration value and exploitation value.

$$\text{UCT}(a) = \frac{Q(a)}{N(a)} + c \cdot \sqrt{\frac{\log N_{\text{parent}}}{N(a)}}$$

$\text{UCT}(a)$: The UCT score for action or branch a
$Q(a)$: Total reward accumulated from choosing action a
$N(a)$: Number of times action a has been selected
N_{parent} : Total number of visits to the parent node of a
c : Exploration constant (typically a positive scalar like $\sqrt{2}$ or 1.4)

Note that we can also use PUCT, a strategy used by the DPTS paper and Alphazero [Czech et al., 2020b] that combines both scoring strategies, where the UCT formula features an additional linear term multiplied by the scoring model's score given by

$$\text{PUCT}(a) = \frac{Q(a)}{N(a)} + c \cdot P(a) \cdot \frac{\sqrt{N_{\text{parent}}}}{1 + N(a)}$$

$\text{P}(a)$: Prior probability of choosing action a, predicted by a separate scoring model

- **Stopping Criteria**: Traversal is terminated when: (1) a sufficiently confident leaf node is reached, (2) nodes converge to similar answers, (3) the score distribution stabilizes, or (4) a compute/token budget is reached. These conditions will be subject to ablation and refinement.
- **Multi-Layer Self-Correction Framework**: Each reasoning branch undergoes systematic validation through three correction layers: (1) Self-Rewarding Correction [Liu et al., 2025] with 5-word constraint corrections, (2) cross-branch validation through semantic similarity and logical consistency checking, and , mathematical, and commonsense reasoning perspectives. This addresses fundamental self-consistency limitations including systematic bias and independence assumption violations.
- **Computational Efficiency Optimizations**: We implement five optimization protocols: semantic caching, ChunkKV compression enabling larger trees on single GPUs, vLLM tree-attention for parallel branch processing, Ember compound-model coordination preventing

overthinking, and convergence-based dynamic stopping. Node-level confidence scoring with multi-criteria thresholds (confidence, entropy, token budget) enables intelligent pruning while preserving reasoning diversity, ensuring both computational efficiency and solution quality. We will implement intelligent parent node elimination when child nodes consistently score below threshold, and employ multi-armed bandit strategies for exploration-exploitation balance in branch selection.

This comprehensive framework enables SSDP to maintain ToT's reasoning benefits while achieving computational cost reduction through systematic semantic optimization and intelligent resource management. By addressing over-exploration through semantic merging, SSDP provides a scalable solution for complex reasoning tasks across varying computational constraints.

3.3 Stopping Criteria

Traversal stops when:

- A leaf node exceeds a confidence threshold.
- Active branches converge on the same or semantically similar answer.
- Score variance across nodes stabilizes (i.e., convergence).
- Token or computational budget is exhausted.

An ablation study will be conducted to assess the sensitivity of the SSDP algorithm to these criteria and determine optimal settings for different reasoning tasks.

3.4 Algorithm

Let T represent the search tree with root node r. Let $\text{Sim}(a, b)$ denote a semantic similarity function. The following pseudocode summarizes the SSDP algorithm.

Algorithm 1 SSDP: Semantic Similarity-based Dynamic Pruning

Initialize tree T with root node r
Initialize priority queue $Q \leftarrow [r]$
while Q is not empty **do**
  Pop node n from Q
  if n is a terminal node **then**
    Save n as a candidate answer
    continue
  end if
  Generate k children $\{c_1, \ldots, c_k\}$ via parallel reasoning
  for each child c_i **do**
    Compute confidence score $S(c_i)$
    if $S(c_i) < \tau$ **then**
      Discard c_i
    end if
  end for
  Initialize merged_nodes $\leftarrow [\,]$
  for each remaining pair (c_i, c_j) **do**
    if $\text{Sim}(c_i, c_j) > \sigma$ **then**
      Merge c_i and c_j into m
      Update $S(m)$ and append m to merged_nodes
    end if
  end for
  Discard parent n if most children were pruned
  Add nodes in merged_nodes to Q, sorted by $S(\cdot)$ descending
end while
return leaf node with highest score

4 Experimental Setup

Our experiments will be conducted using a range of open-source large language models (LLMs) to assess the generalizability and performance of our approach across model sizes and architectures. Specifically, we will evaluate our method on the following models:

- **Qwen2.5-1.5B**
- **Qwen2.5-7B**
- **LLaMA-3.1-8B**
- **LLaMA-3.2-3B**

These models were selected for their state-of-the-art performance and accessibility in research settings, providing a representative sample of both mid- and large-scale LLMs.

We will test our proposed method against several established Tree-of-Thought (ToT) traversal strategies, including:

- **Monte Carlo Tree Search (MCTS)** [Sprueill et al., 2023, Xie et al., 2024]
- **Best-of-N sampling** [Yao et al., 2023]
- **Beam Search** [Chan et al., 2025]
- **FETCH** [Wang et al., 2025]
- **DPTS** [Ding et al., 2025]

By running our experiments across this suite of models and traversal methods, we aim to evaluate the effectiveness and efficiency of our approach under varying computational and architectural conditions.

5 Datasets, Benchmark and Evaluation

To evaluate our proposed approach, we will conduct experiments on two widely adopted benchmarks for mathematical reasoning: GSM8K and MATH500. GSM8K consists of grade school math word problems requiring multi-step reasoning, while MATH500 includes higher-level mathematical problems that test more complex symbolic manipulation and problem-solving abilities. These datasets have become standard in the evaluation of reasoning methods and are frequently used in recent works exploring Tree-of-Thought (ToT) traversal techniques, including Beam Search, Monte Carlo Tree Search (MCTS), and Best-of-N sampling. We can also test them on datasets that feature more complicated questions that require further reasoning, such as AIME, GPQA. The AIME dataset consists of difficult mathematical problems where each answer is an integer, and the GPQA dataset consists of physics, chemistry, and biology questions that have been verified by experts. Testing on these more complex tasks will hopefully yield improved results than the traditional reasoning model strategies, thus showing the advantage of our proposed SSDP framework.

We can also test them on common sense reasoning and logical coherence benchmarks.

We will compare the performance of our method against these ToT traversal baselines to assess improvements in both reasoning quality and computational efficiency. The primary evaluation metrics will be the following:

- **Accuracy**: the percentage of correctly solved problems
- **Latency**: the time taken to reach a final answer
- **Compute cost**: measured in terms of tokens

Additionally, we will conduct an **ablation study** for different node-level scoring mechanisms.

6 Ideal Results

We expect our method to achieve accuracy that is on par with or better than existing Tree-of-Thought traversal strategies such as Monte Carlo Tree Search (MCTS), Beam Search, and Best-of-N sampling.

At the same time, we aim to significantly improve efficiency by reducing the number of model calls, the total number of tokens processed, and overall inference time. Ideally, our approach will demonstrate that it can match the reasoning performance of these baselines while being more resource-efficient, thereby offering a more scalable and practical solution for structured reasoning with language models.

7 Limitations

While SSDP improves computational efficiency and offers a more strategic approach to reasoning, several limitations remain. The method assumes that semantic similarity between intermediate steps implies logical equivalence, which may not hold in tasks where subtle variations in language carry distinct implications. Our reliance on model-generated confidence scores and entropy for pruning introduces potential noise, especially when used with less calibrated models. Additionally, the framework assumes a breadth-first expansion strategy with no rollback, which limits its adaptability to problems requiring backtracking or recursive reasoning. Finally, while our proposal applies well to math questions, it remains empirical and task-specific, and would require further validation before it can generalize across domains or model scales.

References

Xuezhi Wang and Denny Zhou. Chain-of-thought reasoning without prompting, 2024. URL https://arxiv.org/abs/2402.10200.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zengmao Wang, Ziwei Liu, Bo Du, Xianglong Liu, and Dacheng Tao. Dynamic parallel tree search for efficient llm reasoning, 2025. URL https://arxiv.org/abs/2502.16235.

Tim Knappe, Ryan Li, Ayush Chauhan, Kaylee Chhua, Kevin Zhu, and Sean O'Brien. Semantic self-consistency: Enhancing language model reasoning via semantic weighting, 2025. URL https://arxiv.org/abs/2410.07839.

Ante Wang, Linfeng Song, Ye Tian, Dian Yu, Haitao Mi, Xiangyu Duan, Zhaopeng Tu, Jinsong Su, and Dong Yu. Don't get lost in the trees: Streamlining llm reasoning by overcoming tree search exploration pitfalls, 2025. URL https://arxiv.org/abs/2502.11183.

Shangziqi Zhao, Jiahao Yuan, Guisong Yang, and Usman Naseem. Can pruning improve reasoning? revisiting long-cot compression with capability in mind for better reasoning, 2025. URL https://arxiv.org/abs/2505.14582.

Hoai-Chau Tran, Duy M. H. Nguyen, Duy M. Nguyen, Trung-Tin Nguyen, Ngan Le, Pengtao Xie, Daniel Sonntag, James Y. Zou, Binh T. Nguyen, and Mathias Niepert. Accelerating transformers with spectrum-preserving token merging, 2024. URL https://arxiv.org/abs/2405.16148.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less, 2025. URL https://arxiv.org/abs/2502.18600.

Johannes Czech, Patrick Korus, and Kristian Kersting. Monte-carlo graph search for alphazero. *arXiv preprint arXiv:2012.11045*, 2020a. URL https://arxiv.org/abs/2012.11045.

Kiminori Matsuzaki. Empirical analysis of puct algorithm with evaluation functions of different quality. In *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE, 2018.

Andy Zhou et al. Entropy-guided exploration in alphazero: Enhancing mcts with information gain for strategic decision making. In *Proceedings of the 2024 7th International Conference on Machine Learning and Machine Intelligence (MLMI)*. ACM, 2024. URL https://dl.acm.org/doi/10.1145/3696271.3696296.

Johannes Czech, Patrick Korus, and Kristian Kersting. Monte-carlo graph search for alphazero, 2020b. URL https://arxiv.org/pdf/2012.11045.

Henry W. Sprueill, Carl Edwards, Mariefel V. Olarte, Udishnu Sanyal, Heng Ji, and Sutanay Choudhury. Monte carlo thought search: Large language model querying for complex scientific reasoning in catalyst design, 2023. URL https://arxiv.org/abs/2310.14420.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P. Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning, 2024. URL https://arxiv.org/abs/2405.00451.

Brian J Chan, Jui-Hung Cheng, Mao Xun Huang, Chao-Ting Chen, and Hen-Hsen Huang. Efficient beam search for large language models using trie-based decoding, 2025. URL https://arxiv.org/abs/2502.00085.