A Reproduction of Tree of Thoughts

Sophia Baik*   William Leung*   Ashley Son*   Nicole Ku*

https://github.com/Tree_of_Thoughts-deeplearning

1   Introduction

Traditional large language models (LLMs) are trained for next-token or next-sentence prediction, resulting in sequential text generation. Hence, LLMs struggle with structured, multi-step reasoning, limiting performance on complex problems that require planning or exploring multiple solution paths. To address this, we reproduce the Tree of Thoughts (ToT) [6] framework introduced by Yao et al. in 2023, which enhances the Chain of Thought (CoT) [5] approach. Unlike CoT's linear reasoning, ToT enables language models to generate and explore intermediate "thoughts" through a structured branching process, enhancing problem-solving ability.

2   Chosen Result

In this paper, we reproduce the results from Section 4.1 of Yao et al.'s 2023 paper, which demonstrates the improved performance of ToT on the Game of 24. This game involves combining four numbers with basic arithmetic operations to create 24, a task that requires strategic planning and logical reasoning. The key result we aim to replicate is shown in Fig. 1, which compares the accuracy of ToT against IO prompting. IO prompting (N-shot prompting) only has an accuracy of 7.3%, while the paper's ToT achieves a much higher accuracy of 74% with $b = 5$ where b is the maximum breadth of the tree at any given level.

Fig. 1: Sec. 4.1 Results [6]

Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	**74%**
IO + Refine (k=10)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

3   Methodology

In Section 3.1, we describe our implementation of ToT, which includes multiple modifications from the original paper due to limitations in cost, API limits, and time. In Section 3.2, we explain how our changes deviate from the original method.

We use GPT-4o-mini via OpenAI's API [4], whereas the paper uses GPT-4 (2023). This distinction leads to expected differences in both performance and cost. To control this, we compare against 5-shot prompting as a baseline.

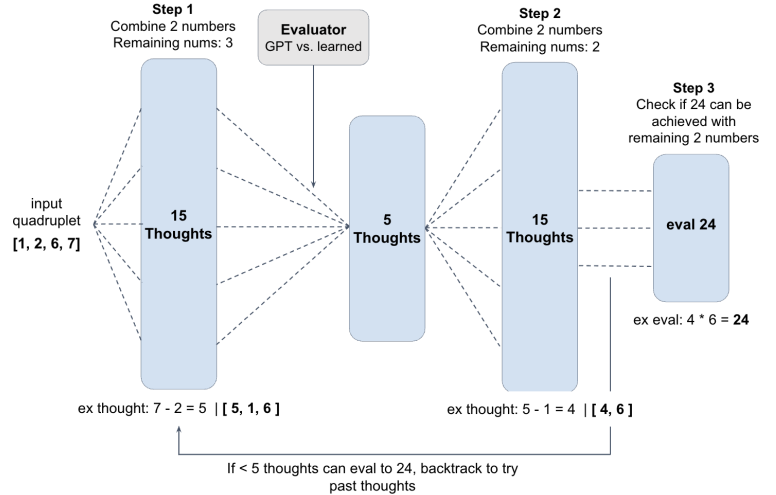Fig. 2: Visualization of ToT sampling and backtracking process.

3.1   Our Methodology

Dataset. We created a dataset of 1,362 solvable four-number combinations of the numbers 1-13, and we categorized each combination into 1 of 5 difficulty levels based on the number of possible solutions.

ToT setup. G24 can be split into three steps where in each step, two numbers are combined using an operation. These steps are connected by thoughts, with each thought merging two numbers into one. For example, starting with the numbers $[2, 4, 5, 6]$, a thought could be $2 + 4 = 6$.

In the first step, we sample $b \times 3$ thoughts, where breadth $b = 5$ for all trials, and only keep unique thoughts. To generate unique thoughts, each prior thought is appended onto the prompt for every subsequent sampling, and we encourage GPT not to choose an existing thought. Before step 2, we use an evaluator to prune the number of thoughts back down to b. In the second step, we sample 3 thoughts for each of the remaining b thoughts. At this point, each thought reduces the set to two remaining numbers. Then, we manually evaluate whether 24 can be obtained by iterating through all possible arithmetic operations. If less than b thoughts are viable, then we backtrack to the $b \times 3$ thoughts produced in step 1, and take the thoughts that were not used previously. Finally, we sample step 3 and check if GPT was able to obtain 24. A visualization of this process is shown in Fig. 2.

At each sampling step, we parse GPT's response and check for validity by ensuring exactly two available numbers are used and computational consistency. If any of these conditions are violated—for example, if GPT says $2 + 4 = 8$—we prune that thought immediately.

Evaluation: LLM. In coherence with Occam's Razor, we prompt GPT to evaluate whether the given thought can reach 24 on a scale of sure, likely, or impossible.

Evaluation: Learned Value Function. We trained a neural network to estimate the value function $V(state)$, which represents the probability of reaching 24 from a given set of numbers. The network consists of a series of fully connected layers with Leaky ReLU activations, Layer Normalization, and dropout (with $p = 0.3$) for regularization. The dataset maps states to a normalized value representing the probability of reaching 24. During training, rewards propagate from leaf states that reach 24, back to their ancestors.

3.2 Modifications from original paper

BFS. The paper uses a BFS (Big Friendly Student) algorithm to explore the entirety of the tree. We decided that in the context of G24, this approach was equivalent to brute-forcing a solution by exploring all combinations of numbers and operations, causing the computation time and cost to increase significantly. However, to allow for exploration of previous thoughts, we implement a backtracking procedure, which jumps from step 3 to step 1, revisiting previous thoughts if there are insufficient results.

Extra evaluation step. In the paper, an evaluation step follows each sampling step. However, to increase efficiency in our G24 implementation, we omitted the evaluation step after sampling step 2. At this point, only two numbers remain, and it is trivial to determine whether 24 can be reached.

Learned Value Function. The paper does not implement a learned evaluator nor does it experiment with any other evaluation methods: it only uses an LLM evaluator. However, we additionally implemented a learned evaluator, which we predicted would be cheaper and potentially improve performance.

4 Results & Analysis

All of our ToT models outperform 5-shot prompting with GPT at every difficulty level as shown in Fig. 3. Our backtracking approach achieved accuracy comparable to the original paper where LLM evaluator > Learned evaluator > 5-shot. ToT with the LLM evaluator model is also 100 times less expensive than the original paper.

Datasets. We used two datasets for evaluation. Table 1 uses the paper's dataset, a subset of 100 problems with an average difficulty score of 2.88. Fig. 3 uses our dataset, where we conducted batches of testing on ten random G24 problems for each difficulty level.

Fig. 3: ToT vs. 5-shot GPT accuracy by G24 difficulty.

Limitations. The biggest limitation was OpenAI API's rate limit, which restricted the number of daily requests and limited our ability to test on more data. Additional constraints included time and cost. The original paper estimates a cost of \$0.74 per problem, which would amount to \$74.00 for just 100 problems. In addition, our model took over a minute to solve one problem with a simpler implementation, suggesting that the original paper's method took longer.

Analysis. Our results reinforce the core results of the original paper: ToT's framework offers substantial improvements over conventional prompting techniques. Moreover, our simplified setup demonstrates that similar performance gains can be achieved at a fraction of the computational cost, indicating that ToT is viable for broader use cases, even without expensive tools or setups.

Table 1: Metric Comparisons

Method	Accuracy	Cost per problem	Time per problem
Paper's ToT	**0.74**	$0.74	–
Our ToT w/ Backtracking	0.68	$0.0064	81.69 secs
Our ToT w/ LLM Eval	0.49	$0.0049	60.62 secs
Our ToT w/ Learned Eval	–	**$0.0020**	**23.53 secs**

5 Reflections

Our reproduction confirmed that ToT outperforms traditional LLMs on structured reasoning tasks like G24. The systematic exploration of reasoning paths and backtracking were key to performance gains for G24 and, more broadly, complex reasoning tasks. It is unsurprising that follow-up research such as Boosting of Thoughts[2] and Graph of Thoughts[1] seeks to expand upon ToT's performance gains.

During re-implementation, we were surprised by how much GPT struggled with the seemingly simple arithmetic reasoning of G24, and equally impressed by how much the ToT framework improved performance. While the paper's performance ultimately eclipsed ours, our results demonstrate that even a scaled-down version of ToT, operating at a fraction of the cost, can still achieve similar performance.

A promising future use case for ToT is LLM improvement. Since ToT is expensive, it should not be used when the LLM has a high confidence in its answer, rather, only in times of uncertainty. If future research can find a way to measure uncertainty, ToT and similar branching frameworks could be critical for improving LLM accuracy. Furthermore, we would be interested in generalizing ToT to other games such as Sudoku, word ladders, or even scientific reasoning tasks such as evaluating fluid mechanics questions and answers.

References

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

[2] Sijia Chen, Baochun Li, and Di Niu. Boosting of thoughts: Trial-and-error problem solving with large language models, 2025.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[4] OpenAI. Openai api. https://platform.openai.com, 2024. Computer software.

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

[6] 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.

A Additional image

pick which one is funniest:
Clash trophies
Pokemon ranked
Drinks on slope day
recovery time after slope day
Distance to the stage on slope day
DL attendance

for context, slope day is a music festival hosted at Cornell university

Funniest one: **"Recovery time after Slope Day"**

Why? It pokes fun at the universally shared post–Slope Day struggle — it's relatable, dramatic, and implicitly chaotic, which makes it hit harder than just naming events or stats. It's a meme-worthy metric.

Runner-up: **"Drinks on Slope Day"** — also funny, but more expected.

Want me to help turn this into a graph or meme format?

Fig. 4: ChatGPT choosing the metric that determines the author ordering.