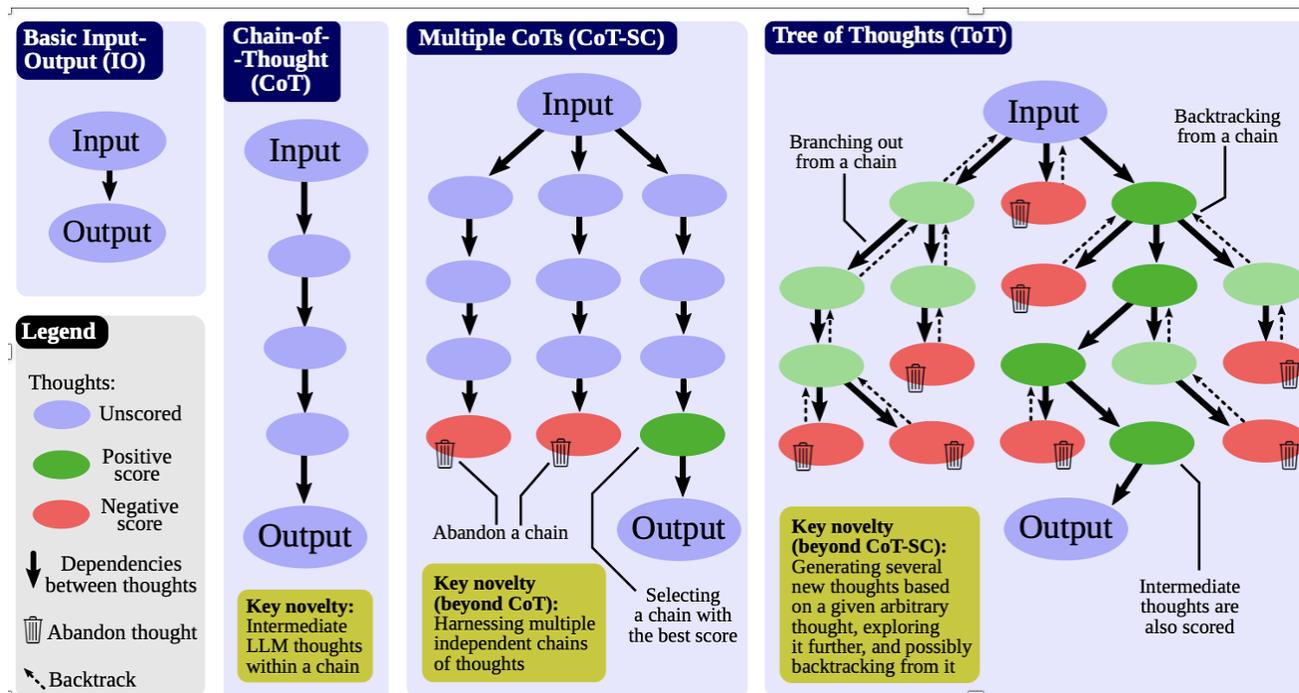


LLM tree-based reasoning

March 25, 2026

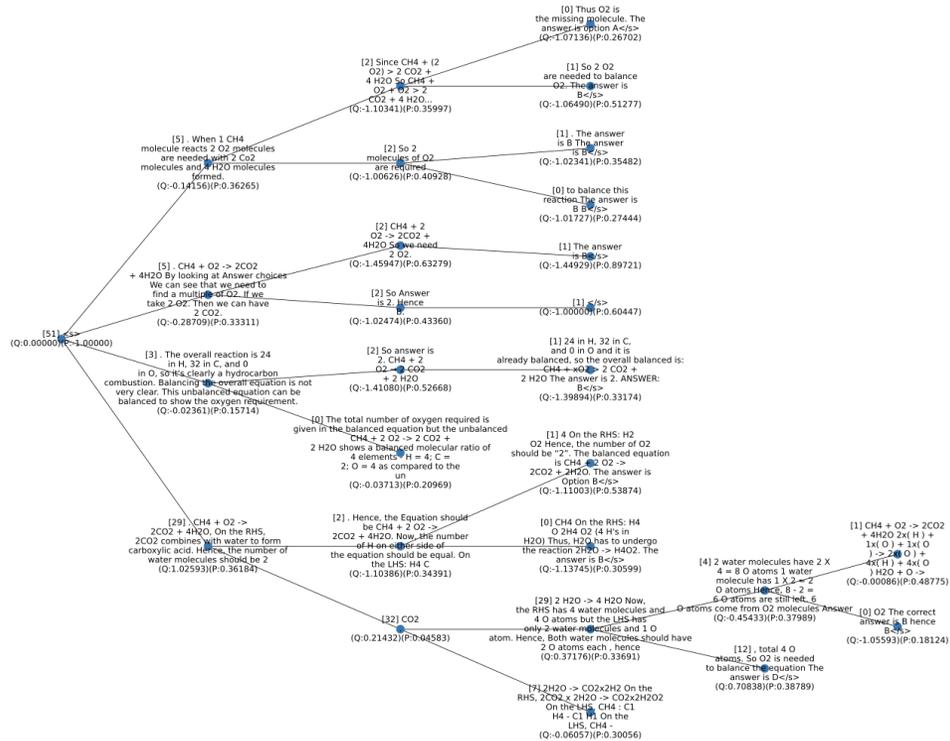
Introduction

- Token-level, left to right decision making process can fall short in tasks that **require exploration**, strategic lookahead, where initial decisions play a pivotal role.
- Consider **multiple different reasoning** paths and enable looking ahead and **backtracking**.



Introduction

A stronger tree generated by Marte Carlo Tree Search (MCTS):



Each reasoning step can be evaluated based on how likely it could lead to the correct answer.

Figure 5: Example of the result search tree of a science question “An unbalanced equation for the reaction of methane gas (CH_4) with oxygen is shown below. $\text{CH}_4 + \square \text{O}_2 \rightarrow 2\text{CO}_2 + 4\text{H}_2\text{O}$ How many molecules of oxygen gas (O_2) are needed to properly balance this equation? Answer Choices: (A) 1 (B) 2 (C) 3 (D) 4”. The ground truth answer is “(D) 4”. Here, we set the search breadth as $b_1 = 4, b_2 = 2$. The numbers at the beginning of each sequence indicate the visit count N of the corresponding node, while the Q and P values at the end of the sequence represent the Q values and the sentence probability, respectively.

Roadmap

- Tree of Thoughts. NeurIPS 2023
- Monte Carlo Tree Search Boosts Reasoning. NeurIPS 2024 Workshop.
- ReST-MCTS*. NeurIPS 2024
- Forest-of-Thought. ICML 2025
- Algorithm of Thoughts. ICML 2024

Motivation (ToT)

By the time of 2023, language models are confined to token-level, left-to-right decision processes during inference, and they can **fall short** in certain tasks:

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%

Table 2: Game of 24 Results.

Method	Success Rate (%)		
	Letter Word Game		
IO	38.7	14	0
CoT	40.6	15.6	1
ToT (ours)	78	60	20
+best state	82.4	67.5	35
-prune	65.4	41.5	5
-backtrack	54.6	20	5

Table 3: Mini Crosswords results.

Method (ToT)

Use LLM to generate and evaluate intermediate reasoning steps, and use BFS and DFS to **enable systematic exploration** of the ToT with lookahead and backtracking.

- Prompt Engineering
- 1) **Decompose** the intermediate process into thought steps;
 - 2) **Generate** potential thoughts from each state
 - 3) **Evaluate** state
 - 4) **Search** algorithm: use BFS/DFS to select node and query LLM with input question and current state (contains previous reasoning steps) to generate the next one.

Method(ToT):

A case study: cross – word task

T	A	S	K	S
M	O	T	O	R

Columns:

V1: to heap

...

V5: Desiccator

Rows:

h1: ...

h2: ...

h3: Pretentious; flowery

h4: ...

h5: ...

Method (ToT)

Step 1: Thought Decomposition

Specify the format in the prompt: ‘using the format "h1. apple (medium)”’

	Game of 24	Creative Writing	5x5 Crosswords
Input	4 numbers (4 9 10 13)	4 random sentences	10 clues (h1. presented;..)
Output	An equation to reach 24 (13-9)*(10-4)=24	A passage of 4 paragraphs ending in the 4 sentences	5x5 letters: SHOWN; WIRRA; AVAIL; ...
Thoughts	3 intermediate equations (13-9=4 (left 4,4,10); 10- 4=6 (left 4,6); 4*6=24)	A short writing plan (1. Introduce a book that connects...)	Words to fill in for clues: (h1. shown; v5. naled; ...)
#ToT steps	3	1	5-10 (variable)

Table 1: Task overview. Input, output, thought examples are in blue.

Method (ToT)

Step 1: Thought Decomposition

A Cross-Word Task

T	A	S	K	S
M	O	T	O	R

Tree

Input clues



h2.motor



h1.tasks

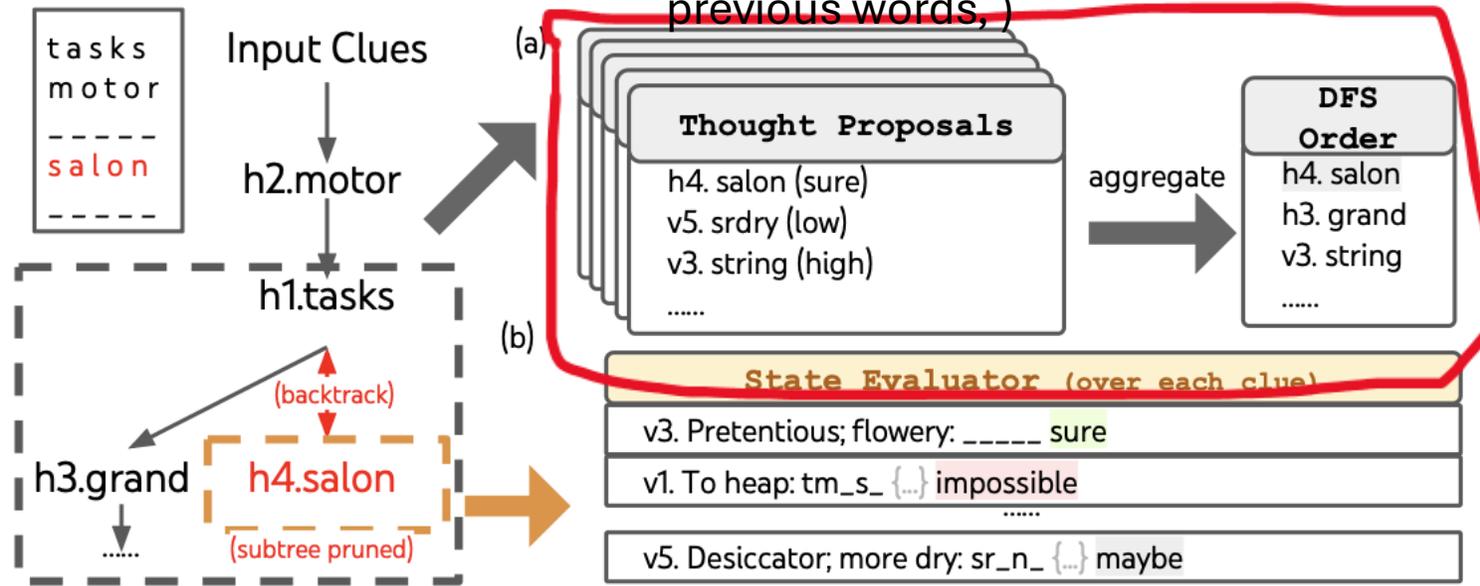
Method (ToT)

DFS (controller)

Python file.

Step 2: Thought Generator

Query LLM (question, clues, previous words,)



```
propose_prompt = '''Let's play a 5 x 5  
mini crossword, where each word  
should have exactly 5 letters.
```

{input}

Given the current status, list all possible answers for unfilled or changed words, and your confidence levels (certain/high/medium/low), using the format "h1. apple (medium)". Use "certain" cautiously and only when you are 100% sure this is the correct word. You can list more than one possible answer for each word.

```
'''
```

Method (ToT)

Step 2: Thought Generator

A Cross-Word Task

T	A	S	K	S
M	O	T	O	R
S	A	L	O	N

DFS search strategy

Tree

Input clues



h2.motor



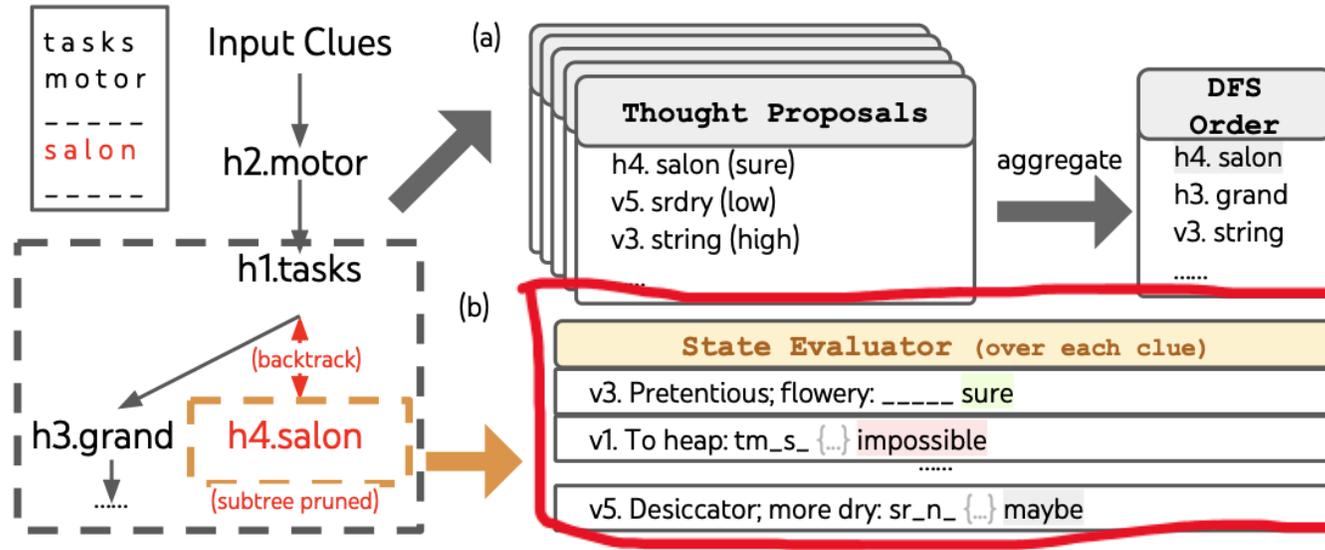
h1.tasks



h4.salon

Method (ToT)

Step 3: State Evaluator



```
value_prompt = '''Evaluate if there exists a five letter word of some meaning that fit some 1
```

```
Incorrect; to injure: w _ o _ g
```

```
The letter constraint is: 5 letters, letter 1 is w, letter 3 is o, letter 5 is g.
```

```
Some possible words that mean "Incorrect; to injure":
```

```
wrong (w r o n g): 5 letters, letter 1 is w, letter 3 is o, letter 5 is g. fit!
```

```
sure
```

```
A person with an all-consuming enthusiasm, such as for computers or anime: _ _ _ _ u
```

```
The letter constraint is: 5 letters, letter 5 is u.
```

```
Some possible words that mean "A person with an all-consuming enthusiasm, such as for comput
```

```
geek (g e e k): 4 letters, not 5
```

```
otaku (o t a k u): 5 letters, letter 5 is u
```

```
sure
```

```
Dewy; roscid: r _ _ _ l
```

```
The letter constraint is: 5 letters, letter 1 is r, letter 5 is l.
```

```
Some possible words that mean "Dewy; roscid":
```

```
moist (m o i s t): 5 letters, letter 1 is m, not r
```

```
humid (h u m i d): 5 letters, letter 1 is h, not r
```

```
I cannot think of any words now. Only 2 letters are constrained, it is still likely
```

```
maybe
```

```
A woodland: _ l _ d e
```

```
The letter constraint is: 5 letters, letter 2 is l, letter 4 is d, letter 5 is e.
```

```
Some possible words that mean "A woodland":
```

```
forest (f o r e s t): 6 letters, not 5
```

```
woods (w o o d s): 5 letters, letter 2 is o, not l
```

```
grove (g r o v e): 5 letters, letter 2 is r, not l
```

```
I cannot think of any words now. 3 letters are constrained, and _ l _ d e seems a common pat
```

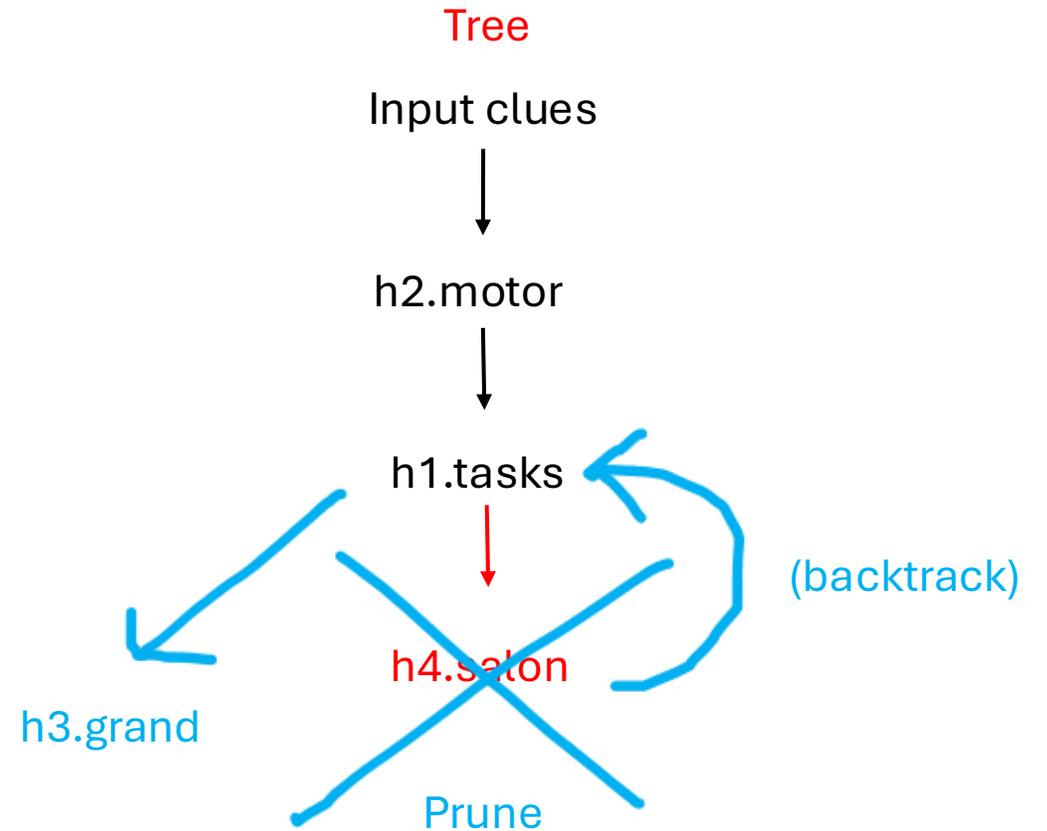
```
maybe
```

Method (ToT)

Step 4: Evaluate and Search

A Cross-Word Task

T	A	S	K	S
M	O	T	O	R
S	A	L	E	N



Experiments (ToT)

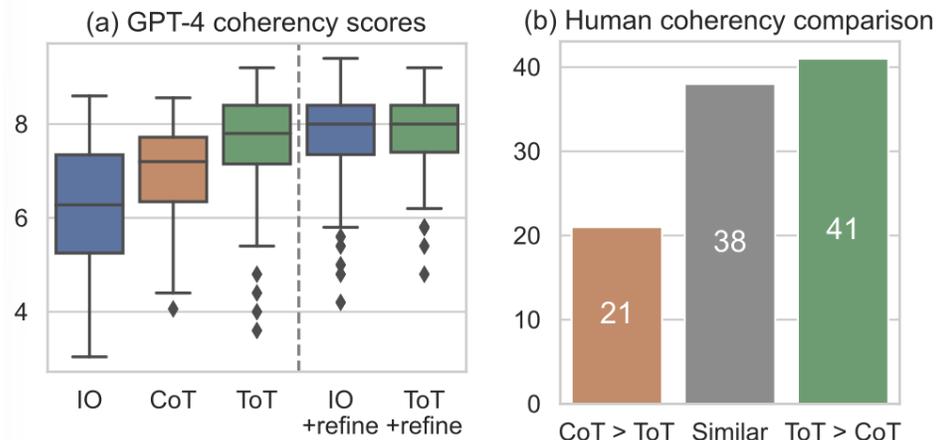


Figure 5: Creative Writing results.

Method	Success Rate (%)		
	Letter	Word	Game
IO	38.7	14	0
CoT	40.6	15.6	1
ToT (ours)	78	60	20
+best state	82.4	67.5	35
-prune	65.4	41.5	5
-backtrack	54.6	20	5

Table 3: Mini Crosswords results.

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%

Keep b among multiple candidates

Table 2: Game of 24 Results.

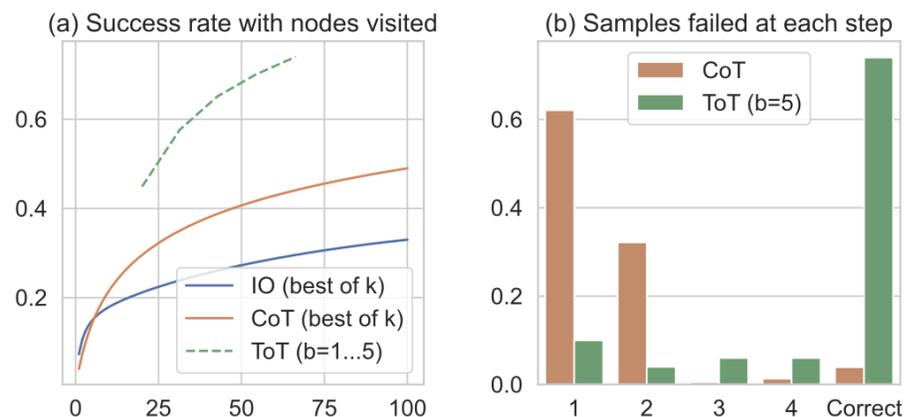


Figure 3: Game of 24 (a) scale analysis & (b) error analysis.

Contribution (ToT)

- First to model reasoning path with tree structure;
- Discuss tasks that are suitable for deliberate search ;
- Simple yet effective design;

Transit from ToT to MCTS modeling....

B.1 Extension to new tasks (GSM8k, StrategyQA) with zero-shot ToT

While more common NLP tasks might be too easy for GPT-4 and do not require ToT (which is why we considered harder new tasks), we believe applying ToT to new tasks could be straightforward. For example, we implemented a simple and generic zero-shot ToT-BFS similar to creative writing (sample 5 problem solving strategies then vote for the best one; then sample 5 solutions based on the best strategy then vote for the best one) for GSM8K and StrategyQA with few extra lines of code:

```
# define the answer format of new tasks
gsm8k_format = "the answer is n" where n is a number'
strategyqa_format = 'either "the answer is yes" or "the answer is no"'

# define zero-shot io prompting
standard_prompt = 'Answer the following question with {format}: {input}'

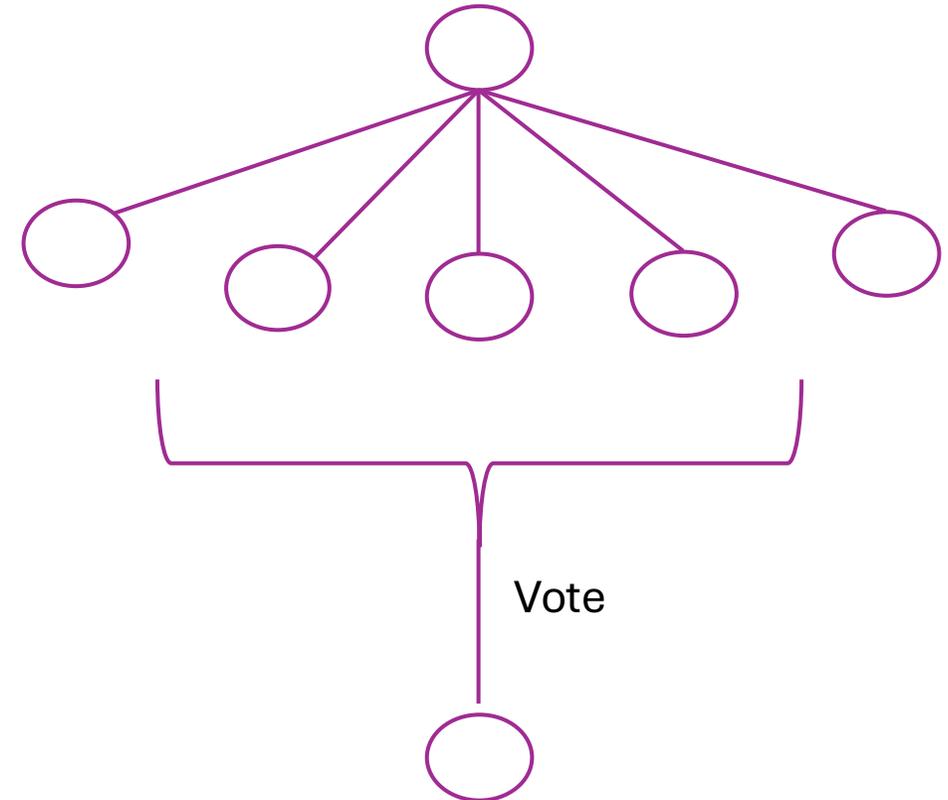
# define thought format for zero-shot cot and zero-shot tot
cot_prompt = "'Answer the following question: {input}'

Make a strategy then write. Your output should be of the following format:

Strategy:
Your strategy about how to answer the question.

Answer:
Your answer to the question. It should end with {format}.
'''

# define zero-shot voting used for zero-shot tot
vote_prompt = "'Given an instruction and several choices,
decide which choice is most promising.
Analyze each choice in detail, then conclude in the last line
"The best choice is {s}", where s the integer id of the choice.
'''
```



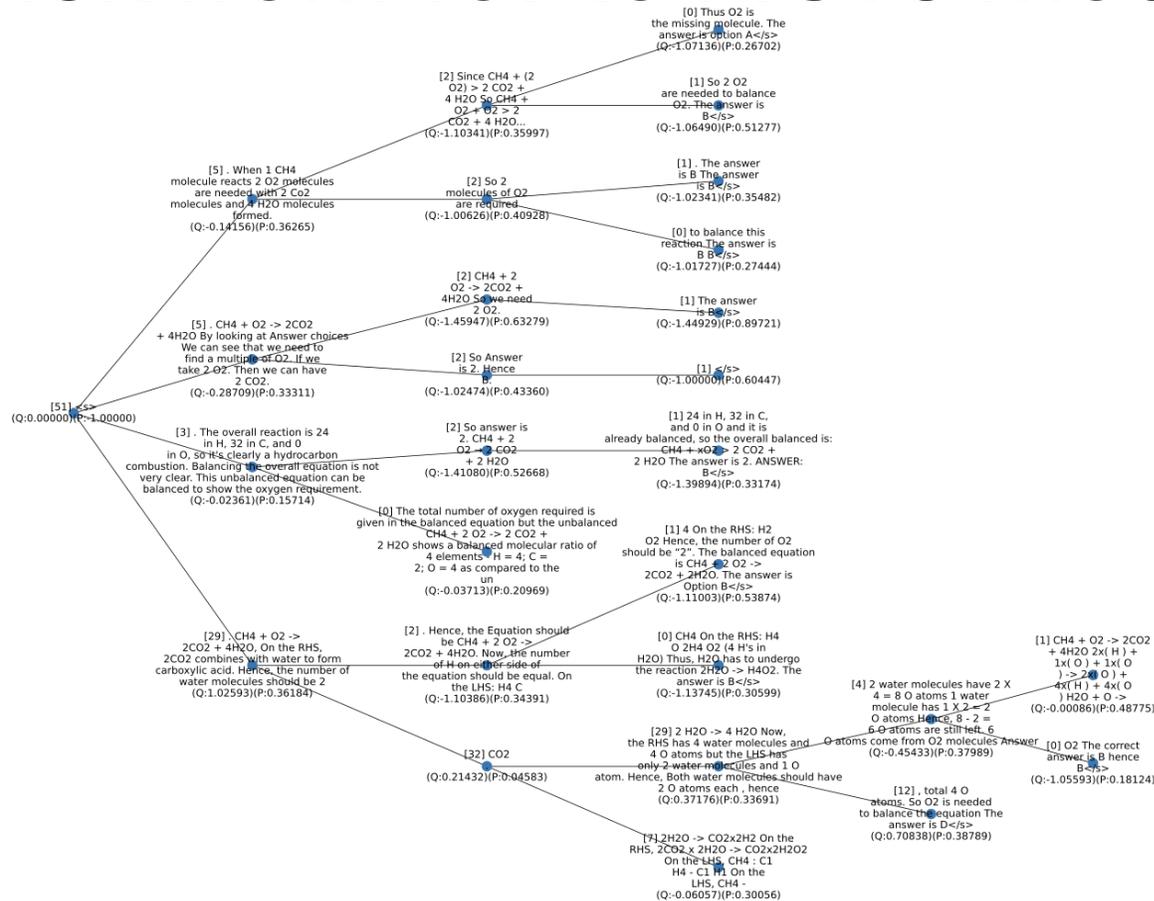
Transit from ToT to MCTS modeling...

Problem: Solve for x in $2x + 10 = 20$.

Flawed Reasoning Trace:

- 1) First, we add 10 to 20 to get 30 (**Error 1**);
- 2) Then, we divide 30 by 6 [**Error 2**] to get $x = 5$ [**Correct Answer**].

Transit from ToT to MCTS modeling...



- We want a stronger tree that :
- 1) works for **more complex** problems,
 - 2) with **step-level signal**,
 - 3) and **more advanced search** strategy

Figure 5: Example of the result search tree of a science question “An unbalanced equation for the reaction of methane gas (CH₄) with oxygen is shown below. CH₄ + □O₂ → 2CO₂ + 4H₂O How many molecules of oxygen gas (O₂) are needed to properly balance this equation? Answer Choices: (A) 1 (B) 2 (C) 3 (D) 4”. The ground truth answer is “(D) 4”. Here, we set the search breadth as $b_1 = 4, b_2 = 2$. The numbers at the beginning of each sequence indicate the visit count N of the corresponding node, while the Q and P values at the end of the sequence represent the Q values and the sentence probability, respectively.

MCTS-Enhanced Iterative Preference Learning

Method: iteratively conducts step-level preference data sampling via MCTS and preference learning via DPO to update the policy.

Algorithm 1 . MCTS-Enhanced Iterative Preference Learning. Given an initial policy $\pi_{\theta^{(0)}} = \pi_{\text{sft}}$, our algorithm iteratively conducts step-level preference data sampling via MCTS and preference learning via DPO to update the policy.

Input: $\mathcal{D}_{\mathcal{P}}$: prompt dataset; $q(\cdot | x)$: MCTS sampling strategy that constructs a tree-structured set of possible responses given a prompt x , where q_{π} represents that the strategy is based on the policy π for both response generation and self-evaluation; $\ell_i(x, y_w, y_l; \theta)$: loss function of preference learning at the i -th iteration, where the corresponding sampling policy is $\pi^{(i)}$; M : number of iterations; B : number of samples per iteration; T : average number of steps per sample

Train π_{θ} on $\mathcal{D}_{\mathcal{P}}$ using step-level preference learning.

for $i = 1$ **to** M **do**

$\pi^{(i)} \leftarrow \pi_{\theta} \leftarrow \pi_{\theta^{(i-1)}}$

Sample a batch of B samples from $\mathcal{D}_{\mathcal{P}}$ as $\mathcal{D}_{\mathcal{P}}^{(i)}$.

/ MCTS for Step-Level Preference Data Collection */*

For each $x \in \mathcal{D}_{\mathcal{P}}^{(i)}$, elicit a search tree of depth T via $q_{\pi_{\theta}}(\cdot | x)$.

Collect a batch of preferences $\mathcal{D}_i = \{ \{ (x^j, y_w^{(j,t)}, y_l^{(j,t)}) |_{t=1}^T \} |_{j=1}^B \}$ s.t. $x^j \sim \mathcal{D}_{\mathcal{P}}^{(i)}, y_w^{(j,t)} \neq y_l^{(j,t)} \sim q_{\pi_{\theta}}(\cdot | x^j)$, where $y_w^{(j,t)}$ and $y_l^{(j,t)}$ is the nodes at depth t , with the highest and lowest Q values, respectively, among all the children nodes of their parent node.

/ Preference Learning for Policy Improvement */*

Optimize θ by minimizing $J(\theta) = \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}_i} \ell_i(x, y_w, y_l; \theta)$.

Obtain the updated policy $\pi_{\theta^{(i)}}$

end for

$\pi_{\theta} \leftarrow \pi_{\theta^{(M)}}$

Output: Policy π_{θ}

MCTS-Enhanced Iterative Preference Learning

Step 0:

Finetune Mistral-7B on Arithmo dataset (540 K math and code problems) as an **initial policy** $\pi_{\theta(0)} = \pi_{\text{sft}}$.

Why: 1) Otherwise the model may fail to produce correct reasoning among all the paths;
2) Align the format to generate intermediate reasoning steps;

MCTS-Enhanced Iterative Preference Learning

Step 1: MCTS:

- 1) sample a batch of samples $\mathcal{D}_{\mathcal{P}}$ as $\mathcal{D}_{\mathcal{P}}^{(i)}$
- 2) for each $x \in \mathcal{D}_{\mathcal{P}}^{(i)}$, elicit a search tree of max depth T via $q_{\pi_{\theta}}(\cdot | x)$

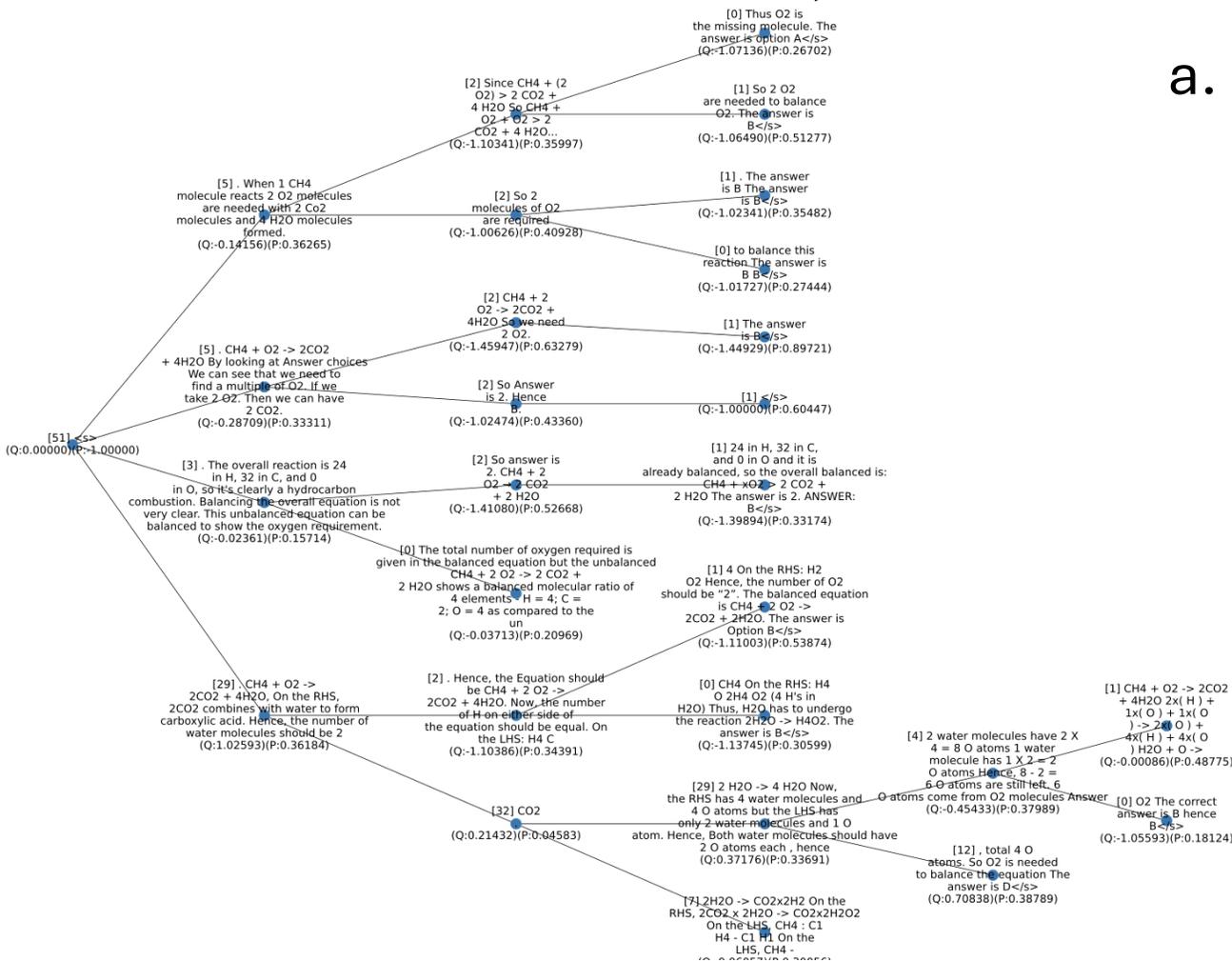
$q_{\pi_{\theta}}(\cdot | x)$: MCTS sampling strategy that construct a reasoning tree using the policy (finetuned LLM) for response generation and self evaluation.

Call LLM (x , previous reasoning step) to generate next step reasoning

MCTS-Enhanced Iterative Preference Learning

To construct a search tree, we have the following steps:

- Selection with PUST (Predictor + Upper Confidence bounds applied to Trees)



Exploration

Exploitation

$$s_{t+1}^* = \arg \max_{s_t} \left[Q(s_t, a) + c_{\text{puct}} \cdot p(a | s_t) \frac{\sqrt{N(s_t)}}{1 + N(s_{t+1})} \right]$$

where $Q(s_t, a)$ is the value of taking action a in state s_t ;

$p(a | s_t) = \pi_{\theta}(a | x, s_t) / |a|^{\lambda}$ denotes the policy π_{θ} 's probability distribution for generating a step a ;

$N(s_t)$ is the visit frequency of state s_t ;

MCTS-Enhanced Iterative Preference Learning

c. Backup : carry out a **bottom-up update** from the terminal node back to the root:

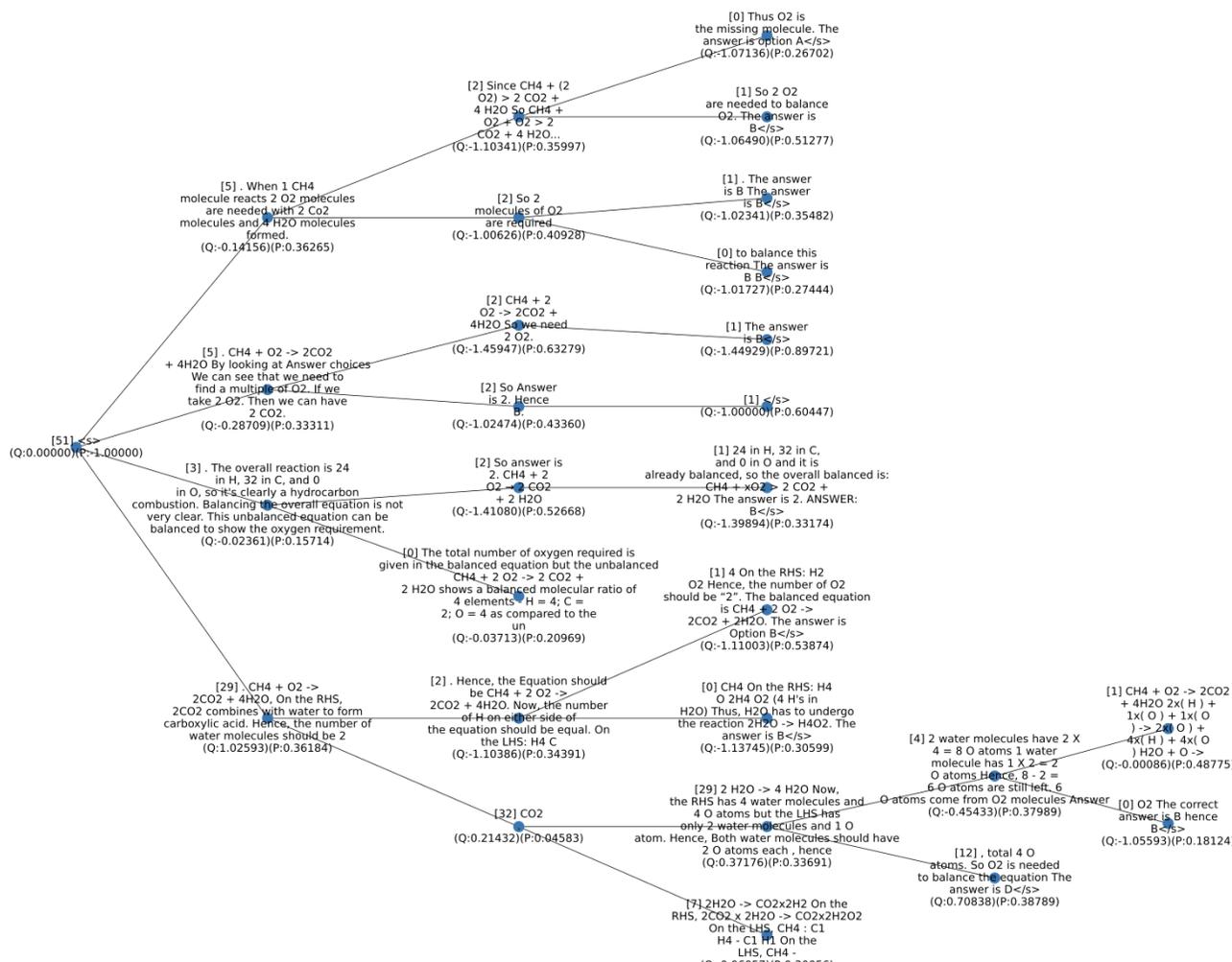
- Transition value: $Q(s_t, a) \leftarrow r(s_t, a) + \gamma V(s_{t+1})$

, where $r(s_t, a) = R(s_{t+1}) - R(s_t)$;
 $V(s_{t+1})$ is state value for the next step;

- state value : weighted average of Q values of all its immediate successor nodes;

$$V(s_t) \leftarrow \sum_a N(s_{t+1})Q(s_t, a) / \sum_a N(s_{t+1})$$

- Increment visit frequency $N(s_t) \leftarrow N(s_t) + 1$



MCTS-Enhanced Iterative Preference Learning

Step 2: collect preference pairs (y_w, y_l) for each step:

- y_w : node with highest Q
- y_l : node with lowest Q

Step 3: Preference Learning (DPO) :

$$\ell_i(\theta) = - \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}_i} \left[(1 - \alpha_{x, y_w, y_l}) \log \sigma(\beta h_{\pi_\theta}^{y_w, y_l}) + \alpha_{x, y_w, y_l} \log \sigma(-\beta h_{\pi_\theta}^{y_w, y_l}) \right]$$

$$\alpha_{x, y_w, y_l} = \frac{1}{N(x, y_w)/N(x, y_l) + 1}$$

MCTS-Enhanced Iterative Preference Learning

Table 1: Result comparison (accuracy %) on arithmetic tasks. We supervised fine-tune the base model Mistral-7B on Arithmo data, while Math-Shepherd (Wang et al., 2023a) use MetaMATH (Yu et al., 2023b) for SFT. We highlight the advantages of our approach via conceptual comparison with other methods, where NR, OG, OF, and NS represent “w/o Reward Model”, “On-policy Generation”, “Online Feedback”, and “w/ Negative Samples”.

Approach	Base Model	Conceptual Comparison				GSM8K	MATH
		NR	OG	OF	NS		
LMSI	PaLM-540B	✓	✓	✗	✗	73.5	–
SFT (MetaMath)	Mistral-7B	–	–	–	–	77.7	28.2
Math-Shepherd		✗	✓	✗	✓	84.1	33.0
SFT (Arithmo)	Mistral-7B	–	–	–	–	75.9	28.9
MCTS Offline-DPO		✓	✗	✗	✓	79.9	31.9
Instance-level Online-DPO		✓	✓	✓	✓	79.7	32.9
Ours		✓	✓	✓	✓	80.7	32.2
Ours (w/ G.T.)		✓	✓	✓	✓	81.8	34.7

- **NR**: without training a separate reward model;
- **OG**: training data is generated by the updated policy;
- **OF**: Online feedback -- updated based on the content (Q) generated by the latest policy ;
- **NS**: use both positive and negative samples (many self-training algorithm only use positive samples)
- **w/ G.T**: Add the ground-truth reasoning step as one of the branch

- 1) Compare with the baseline:
75.9 %-> 80.7% on GSM8k;
28.9 %-> 32.2% on MATH.
- 2) instance -level V.S step-level:
79.7% (Instance-level Online-DPO);
80.7% (Ours)
- 3) w/G. T. : add the ground-truth reasoning
as one of the branch:
80.7% (Ours)
81.8% (Ours w/ G.T.)

MCTS-Enhanced Iterative Preference Learning

Table 2: Result comparisons (accuracy %) on commonsense reasoning tasks. The results based on GPT-3-curie (Brown et al., 2020) and T5 (Raffel et al., 2020) are reported from Liu et al. (2023b). For CSQA, we also include the GPT-J (Wang and Komatsuzaki, 2021) results reported by Zelikman et al. (2022). We follow Liu et al. (2023b) to combine the training data of ARC, AI2Sci, OBQA, and CSQA for training , while STaR (Zelikman et al., 2022) only use CSQA for training.

Approach	Base Model	Conceptual Comparison				ARC-c	AI2Sci-m	CSQA	SciQ	Train Data Used (%)
		NR	OG	OF	NS					
CoT Tuning	GPT-3-curie (6.7B)	✓	✗	✗	✗	—	—	56.8	—	100
Direct Tuning	GPT-J (6B)	✓	✗	✗	✗	—	—	60.0	—	100
STaR		✓	✓	✓	✗	—	—	72.5	—	86.7
Direct Tuning	T5-11B	✓	✗	✗	✗	72.9	84.0	82.0	83.2	100
Crystal		✗	✓	✓	✓	73.2	84.8	82.3	85.3	100
SFT Base (Arithmo)		—	—	—	—	60.6	70.9	54.1	80.8	—
Direct Tuning	Mistral-7B	✓	✗	✗	✗	73.9	85.2	79.3	86.4	100
MCTS Offline-DPO		✓	✗	✗	✓	70.8	82.6	68.5	87.4	19.2
Instance-level Online-DPO		✓	✓	✓	✓	75.3	87.3	63.1	87.6	45.6
Ours		✓	✓	✓	✓	76.4	88.2	74.8	88.5	47.8

MCTS-Enhanced Iterative Preference Learning

- Iterative Learning (Cumulative): ensemble of models at all checkpoints;
- Iterative Learning (Pass@1): Inference once;
- Sampling Only (cumulative): SC;

1) Iterative Learning (pass @1)
 \approx Sampling 60+; (ARC-C)

2) For complex math competition problems that requires long reasoning chain, training fail to internalize the reasoning ability;

3) SciQ as unseen validation set: demonstrate the training improves general reasoning ability;

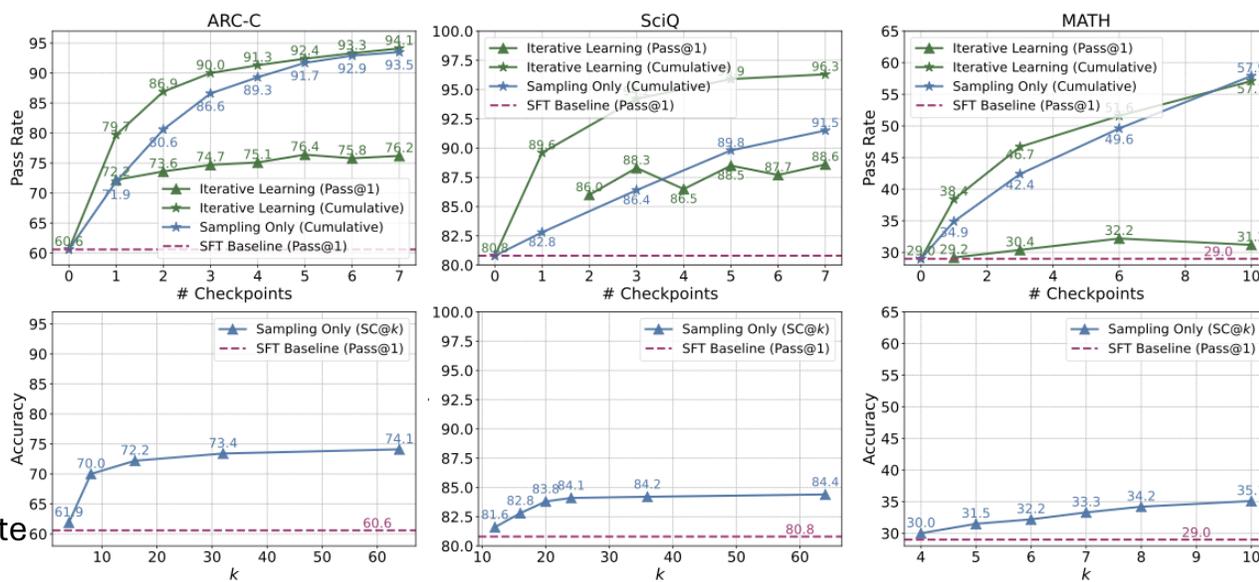


Figure 3: Training- vs. Test- Time Compute Scaling on ARC-C, SciQ, and MATH evaluation sets. The cumulative pass rate of our iterative learning method can be seen as the pass rate of an ensemble of different model checkpoints. We use greedy decoding to obtain the inference time performance of our method of iterative learning.

: # candidate answers

MCTS-Enhanced Iterative Preference Learning

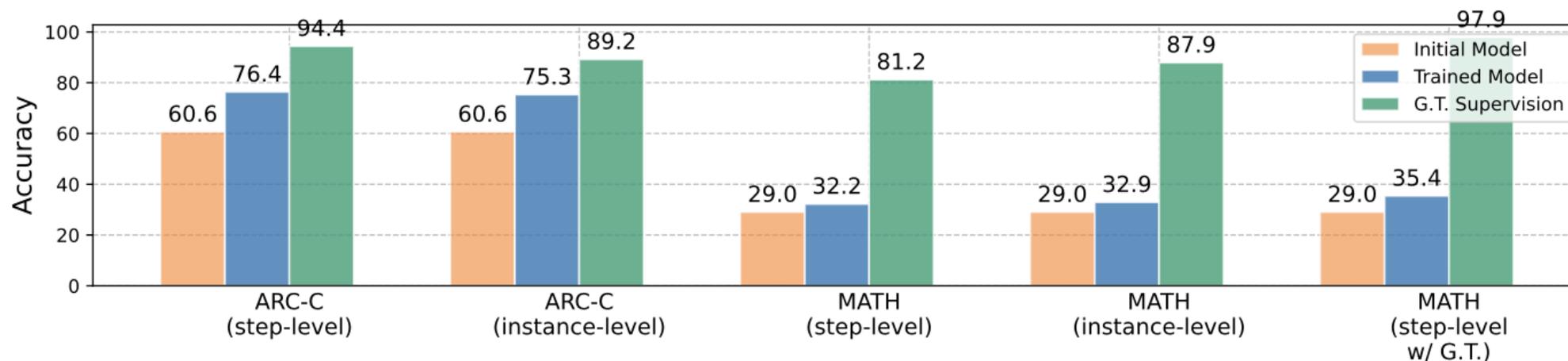


Figure 4: Ablation study on step- vs. instance-level supervision, online vs. offline learning, and w/ vs. w/o G.T. supervision. We also compare the accuracy of the training signals collected via MCTS or instance-level sampling.

Transit from MCTS via PL to ReST-MCTS* ...

*“When compared to Math-Shepherd, which also utilizes process supervision in preference learning, our approach **achieves similar performance** enhancements **without the necessity** of training separate reward or value networks. **This suggests the potential of integrating trained reward model signals into our MCTS stage to further augment performance** “*

-- MCTS via PL

ReST-MCTS*

Motivation: To solve the cost problem in process supervision

Method: For each iteration:

1) use MCTS to automatically mark the reward score for each step

(difference : not preference $A > B$, but the probability to reach to a correct answer from the current state);

2) After each round of MCTS, they use the actual success rate as ground truth to finetune the reward function;

(difference: instead of collect preference pairs, they train a reward function);

3) self-training: only use good traces to finetune;

ReST-MCTS*

Algorithm 1: Mutual self-training ReST-MCTS* for value model and policy model.

Input: base LLM π , original dataset for policy model D_{S_0} , original dataset for value model D_0 , new problem set D_G , number of solutions N , j -th solution A_j , correct solution a^* , value model V_θ , weighted value function w , quality value function v , number of iterations T .

```
1:  $\pi_{S_0} \leftarrow \text{SFT}(\pi, D_{S_0})$  // fine-tune generator
2:  $D_{V_0} \leftarrow \text{generate\_value\_data}(D_0, w, v)$  // initialize train set for value model
3:  $V_0 \leftarrow \text{train\_value\_model}(V_\theta, D_{V_0})$  // initialize value model
4: for  $i = 1$  to  $T$  do
5:    $D_{G_i} \leftarrow \text{generate\_policy\_data}(\pi_{S_{i-1}}, V_{i-1} \text{ guided MCTS}^*, D_G, N)$  // generate synthetic data for policy model
6:   for  $j = 1$  to  $N$  do
7:      $D_{G_i(A_j=a^*)} \leftarrow \text{label\_correctness}(D_{G_i})$  // match and select correct solutions
8:   end for
9:    $\pi_{S_i} \leftarrow \text{SFT}(\pi_{S_{i-1}}, D_{G_i(A_j=a^*)}|_{j=1}^N)$  // self-training policy model
10:   $D_{V_i} \leftarrow \text{extract\_value\_data}(D_{G_i})$  // collect process reward and extract value data
11:   $V_i \leftarrow \text{train\_value\_model}(V_{i-1}, D_{V_i})$  // self-training value model
12: end for
Output:  $\pi_{S_T}, V_T$ 
```

Step 1: Use MCTS to construct reasoning trees and collect reward data;

Difference: MCTS-via-IPL update Q based on self evaluation and correctness, while this method use reward function to estimate the probability of success from current state;

Step 2: Collect the most promising path instead of preference pairs;

Step 3: Finetune policy model good traces only;

Step 4: Update reward function

ReST-MCTS*

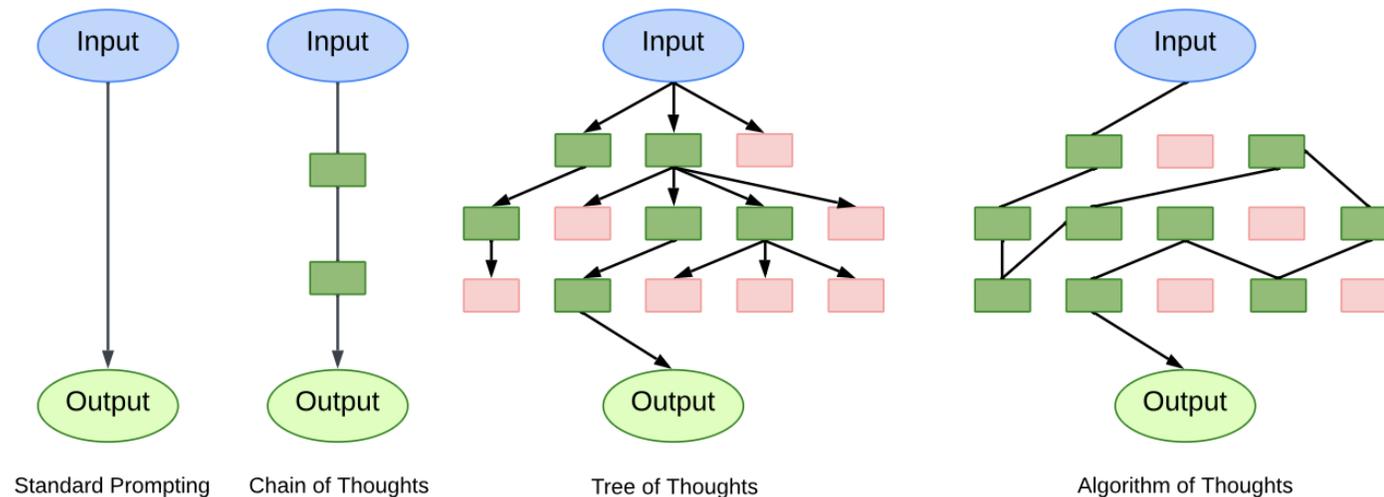
Model	Self-Training Methods	MATH	GPQA _{Diamond}	CEval-Hard	Ave.	
LLaMA-3-8B-Instruct	0th iteration (zero-shot)	20.76	27.27	26.32	24.78	
	0th iteration (few-shot)	30.00	31.31	25.66	28.99	
	(Below are fine-tuned from model of previous iteration with self-generated traces)					
	w/ ReST ^{EM} (1st iteration)	30.84	26.77	21.05	26.22	
	w/ Self-Rewarding (1st iteration)	30.34	26.26	25.66	27.42	
	w/ ReST-MCTS* (1st iteration)	31.42	24.24	26.97	27.55	
	w/ ReST ^{EM} (2nd iteration)	33.52	25.25	21.71	26.83	
	w/ Self-Rewarding (2nd iteration)	33.89	26.26	23.03	27.73	
	w/ ReST-MCTS* (2nd iteration)	34.28	27.78	25.00	29.02	
	Mistral-7B: MetaMATH	0th iteration (zero-shot)	29.34	27.78	9.87	22.33
0th iteration (few-shot)		28.28	29.29	9.21	22.26	
(Below are fine-tuned from model of previous iteration with self-generated traces)						
w/ ReST ^{EM} (1st iteration)		23.84	26.26	20.39	23.50	
w/ Self-Rewarding (1st iteration)		25.70	27.78	19.74	24.40	
w/ ReST-MCTS* (1st iteration)		31.06	26.26	17.11	24.81	
w/ ReST ^{EM} (2nd iteration)		23.86	26.26	22.37	24.16	
w/ Self-Rewarding (2nd iteration)		23.90	26.77	25.00	25.22	
w/ ReST-MCTS* (2nd iteration)		24.40	28.79	26.32	26.50	
SciGLM-6B		0th iteration	25.18	23.74	51.97	33.63
	(Below are fine-tuned from model of previous iteration with self-generated traces)					
	w/ ReST ^{EM} (1st iteration)	22.72	24.75	51.32	32.93	
	w/ Self-Rewarding (1st iteration)	22.50	26.26	47.37	32.04	
	w/ ReST-MCTS* (1st iteration)	24.86	25.25	51.32	33.81	
	w/ ReST ^{EM} (2nd iteration)	25.86	25.25	48.68	33.27	
	w/ ReST-MCTS* (2nd iteration)	23.90	31.82	51.97	35.90	

Algorithm of Thought

Motivation:

ToT use search algorithm (BFS/DFS) as external controller to assign reasoning task, which require frequent LLM queries.

Method: encode tree structure and perform search inside LLM.



Algorithm of Thought

AoT (DFS)

System:

Use numbers and basic arithmetic operations (+ - * /) to obtain 24. When considering the next steps, do not choose operations that will result in a negative or fractional number. In order to help with the calculations, the numbers in the parenthesis represent the numbers that are left after the operations and

16

Algorithm of Thoughts

they are in descending order.

Another thing we do is when there are only two numbers left in the parenthesis, we check whether we can arrive at 24 only by using basic arithmetic operations (+ - * /). Some examples regarding this idea:

(21 2) no
since $21 + 2 = 23$, $21 - 2 = 19$, $21 * 2 = 42$, $21 / 2 = 10.5$, none of which is equal to 24.
(30 6) $30 - 6 = 24$ yes
(8 3) $8 * 3 = 24$ yes
(12 8) no
(48 2) $48 / 2 = 24$ yes

Most importantly, do not give up, all the numbers that will be given has indeed a solution.

User:

14 8 8 2

Assistant:

Trying a promising first operation:

1. $8 + 2$: (14, 10, 8)
- $14 + 10$: (24, 8) 32, 16, 192, 3
- $14 - 10$: (8, 4) 12, 8, 32, 2
- $14 * 10$: (140, 8) 148, 132, 1120, fractional
- $14 / 10$: (8, 1.4) fractional
- $14 + 8$: (22, 10) 32, 12, 220, fractional
- $14 - 8$: (10, 6) 16, 4, 60, fractional
- $14 * 8$: (112, 10) 122, 102, 1120, fractional
- $14 / 8$: fractional
- $10 + 8$: (18, 14) 32, 4, 252, fractional
- $10 - 8$: (14, 2) 16, 12, 28, 7
- $10 * 8$: (80, 14) 94, 66, big, fractional
- $10 / 8$: fractional

Trying another promising first operation:

2. $8 / 2$: (14, 8, 4)
- $14 + 8$: (22, 4) 26, 18, 88, fractional
- $14 - 8$: (6, 4) 10, 2, $24 = 6 * 4$ -> found it!

Backtracking the solution:

Step 1:

$8 / 2 = 4$

Step 2:

$14 - 8 = 6$

Step 3:

$6 * 4 = 24$

Considering these steps: $24 = 6 * 4 = (14 - 8) * 4 = (14 - 8) * (8 / 2) = 24$.

answer: $(14 - 8) * (8 / 2) = 24$.

User:

9 5 5 5

Assistant:

Trying a promising first operation:

Algorithm of Thought

Method	W. Success	Queries	PTs	CTs
I/O	14%	1	790.3	30.5
CoT-SC	15.6%	1	1,400	1,600
ToT	46.5%	> 200	96,700	21.8k
AoT (ours)	52%	2	3,800	975.6

Table 3. 5×5 mini crosswords word: word success rates and the average number of LLM queries for each example. We give the average query count, prompt tokens (PT), and completion tokens generated by the LLM (CT).

Forest of Thought

Motivation: Previous work typically perform a single pass of reasoning and may fail to revisit flawed paths;

Method: introduce a dynamic self-correction strategy that enables real-time error correction

Forest of Thought

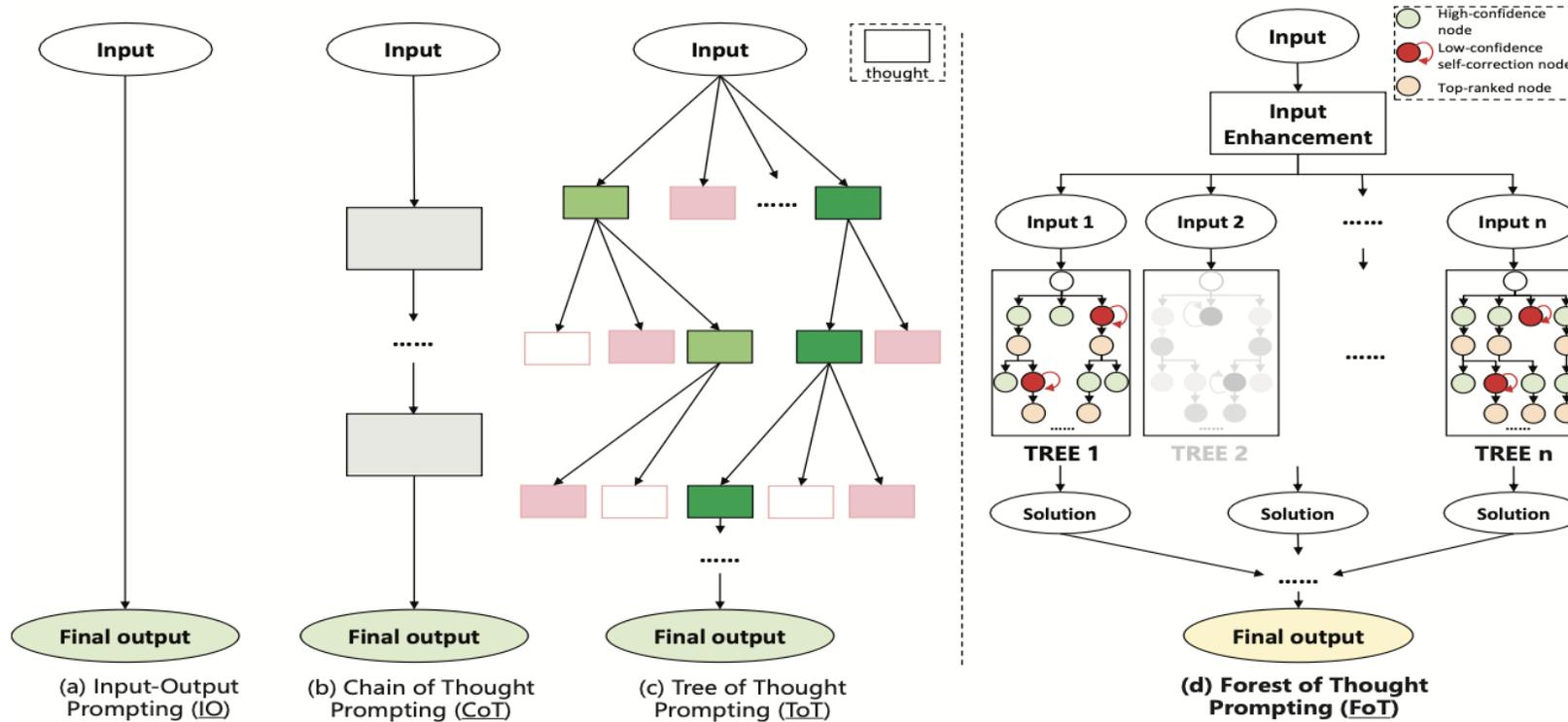


Figure 1. Schematic illustration of various LLM reasoning approaches including IO prompting, CoT, ToT and the proposed FoT.

Forest of Thought

Table 2. Performance Comparison on Game of 24: Our method, with 8 activated subtrees, achieved the highest average ranking across different inference frameworks.

Method	LLM invoked	Success
IO	1.00	10.22%
CoT (Kojima et al., 2023)	1.00	4.38%
CoT-SC (Wang et al., 2023)	10.00	4.38%
GoT (k=1) (Besta et al., 2024a)	7.00	5.26%
ToT (b=5) (Yao et al., 2024)	13.74	74.00%
BoT (Yang et al., 2024)	3.00	82.40%
XoT (w/ 3 r) (Ding et al., 2024)	1.78	85.40%
Ours	25.64	96.84%