GAIR Lab

Dicussion on O1 series

Insights and Perspectives from Recent Papers/Blogs
especially about searching-based inference & (Long)CoT data synthesis

Zhen Huang

2024.10.05

Preliminaries

Enhancing LLMs' Reasoning Abilities (Review)

Pretraining:

- Large scale & high-quality reasoning data.
  - Domain-specific math data and synthetic data (e.g. MathPile)
  - Free-form math discussion text (why conversational abilities are needed?)
- Automatic data engineering (e.g. ProX, MAmmoTH2)

Post-training:

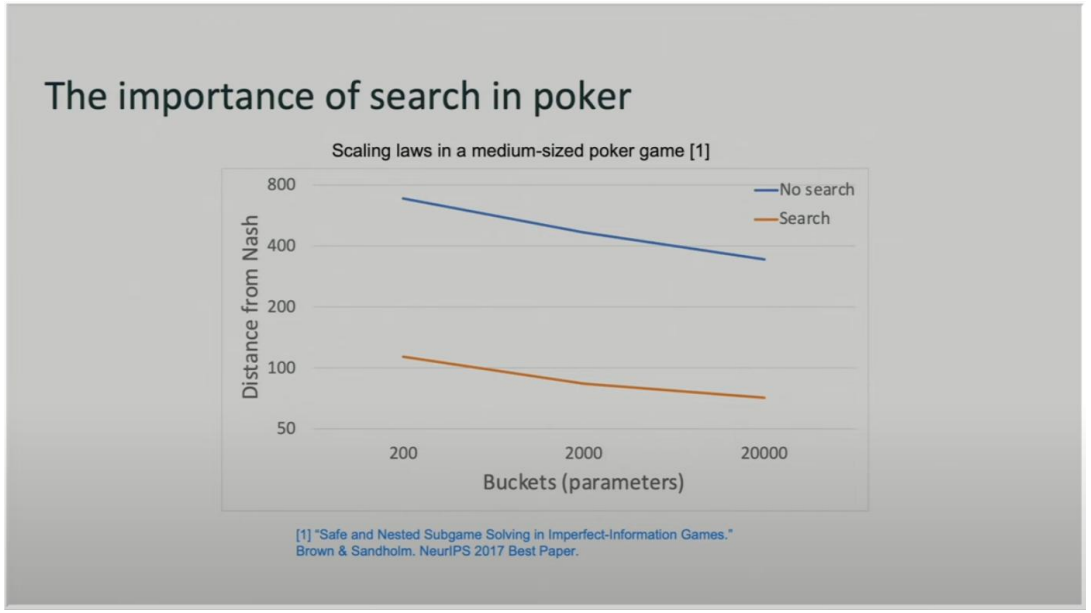
- SFT
- Preference learning like PPO, DPO...
  - Output-level, Process/Step-level

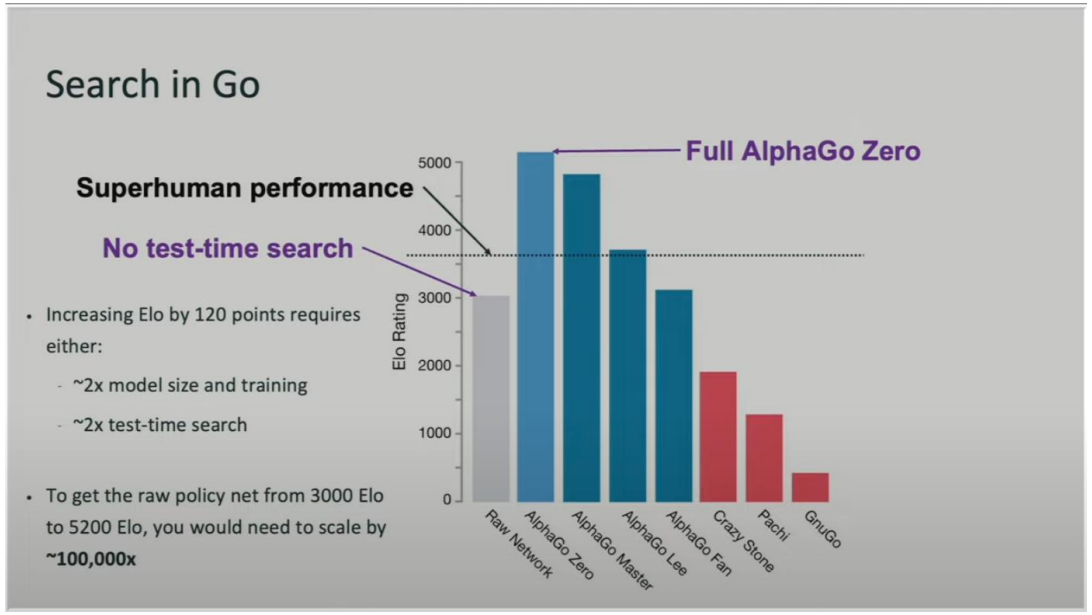
Inference-time:

- CoT, PoT
- Self-consistency + Majority voting
- ToT (BFS & DFS), MCTS

→ **Searching** through the problem space to find good problem-solving paths.

The Importance of Search





- The benefits brought by search are much greater than those gained from increasing the model size.
- It can provide nearly a 7-fold improvement with the same model size!
- In poker game: 100,000 times model scale from blue line to orange line!

Why Searching Works?

The Generator-Verifier Gap

- For some problems, verifying a good solution is easier than generating one
- Examples where verification is easier than generation:
  - Many puzzles (Sudoku, for example)
  - Math
  - Programming
- Examples where verification isn't much easier
  - Information retrieval (What's the capital of Bhutan
  - Image recognition
- When a generator-verifier gap exists *and we have a good verifier*, we can spend more compute on generation and then verify

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

- The difficulty of generating a good solution is usually greater than that of verifying a solution.
- A reliable verifier/reward model is necessary!

Searching in LLMs

	ORM	PRM	Majority Voting
% Solved (Best-of-1860)	72.4	**78.2**	69.6

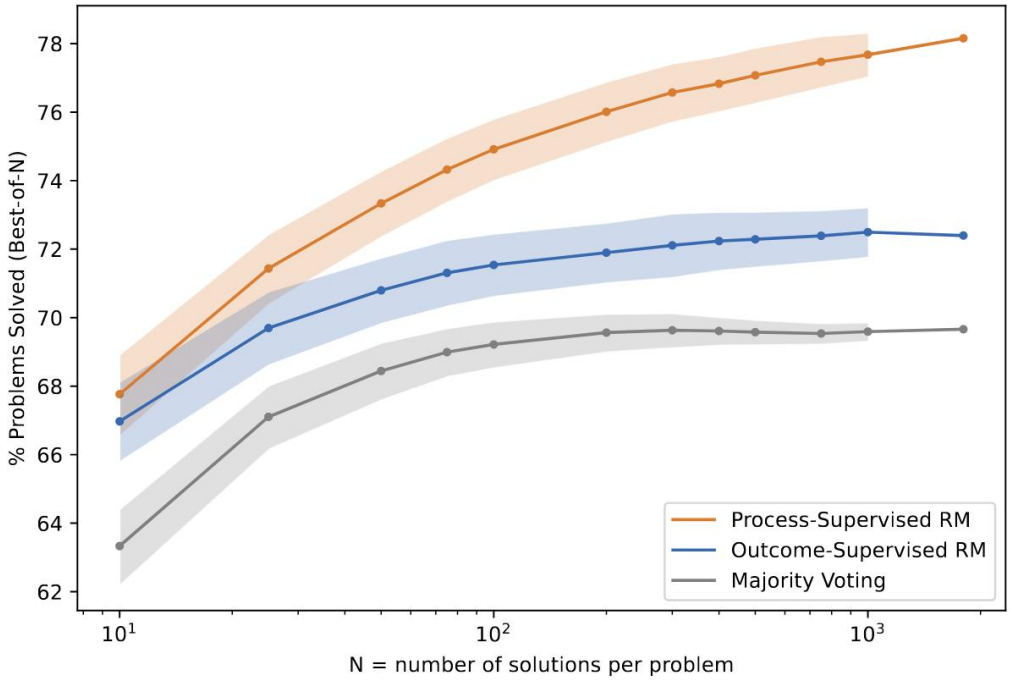
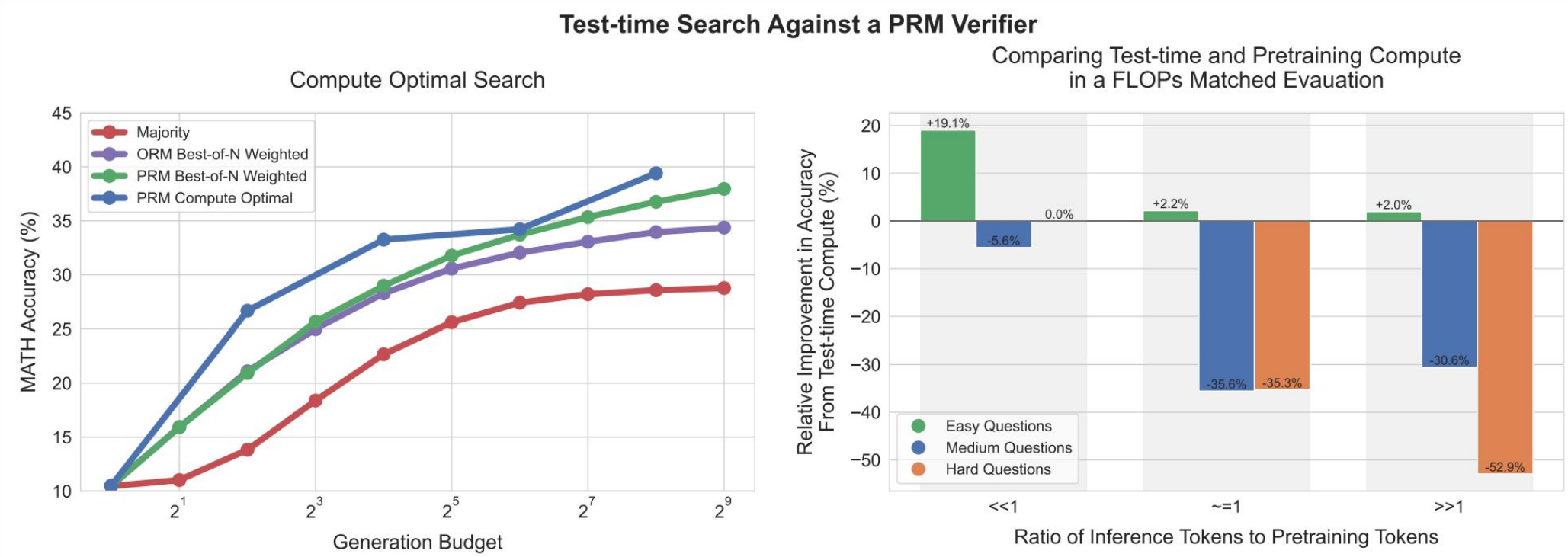


Figure 3: A comparison of outcome-supervised and process-supervised reward models, evaluated by their ability to search over many test solutions. Majority voting is shown as a strong baseline. For $N \leq 1000$, we visualize the variance across many subsamples of the 1860 solutions we generated in total per problem.

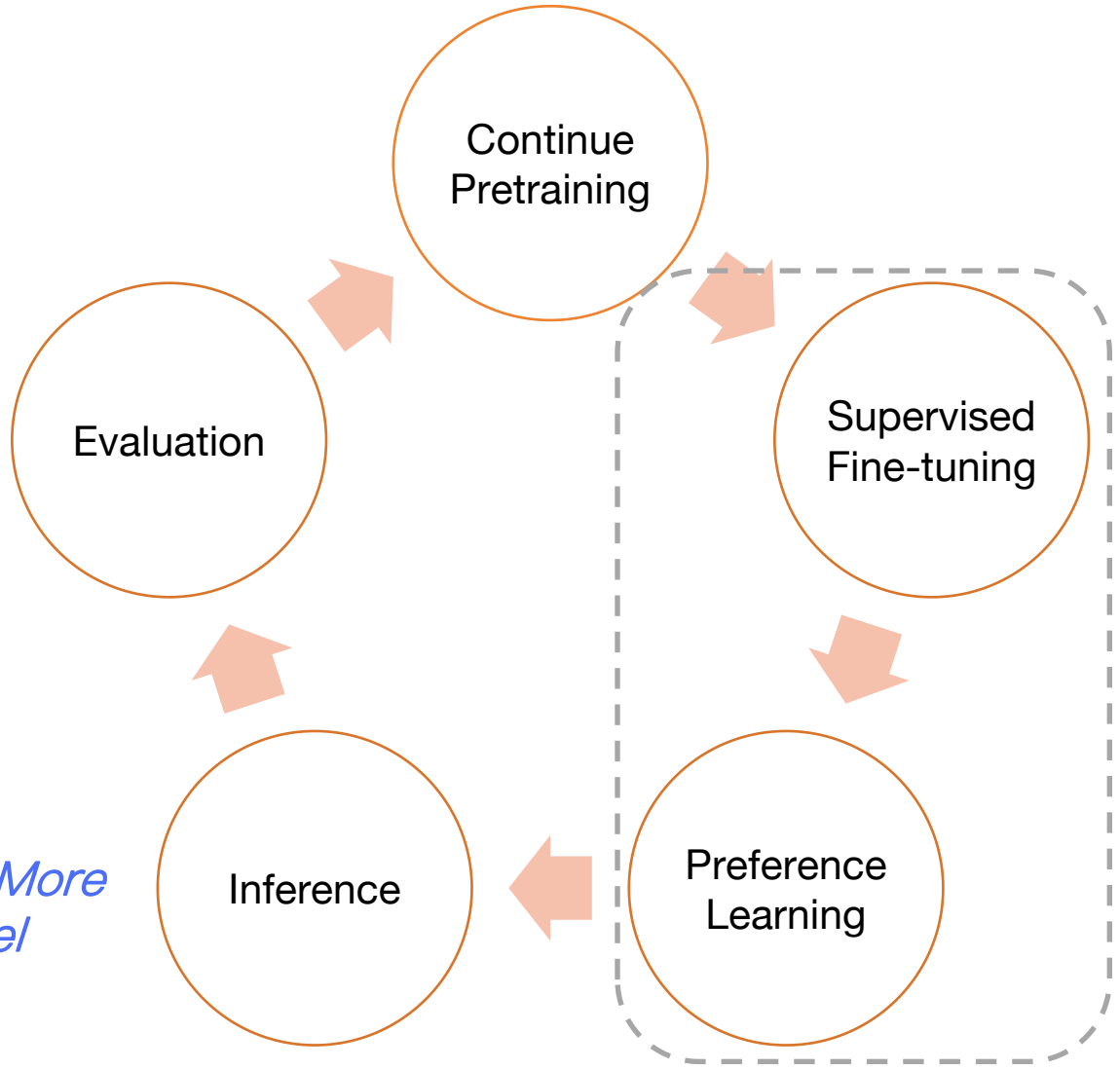
Inference-Time Scaling in LLMs



- Smaller models can outperform models 14 times larger when using more computational resources during the inference stage in MATH.

Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

Inference-Time Scaling in LLMs

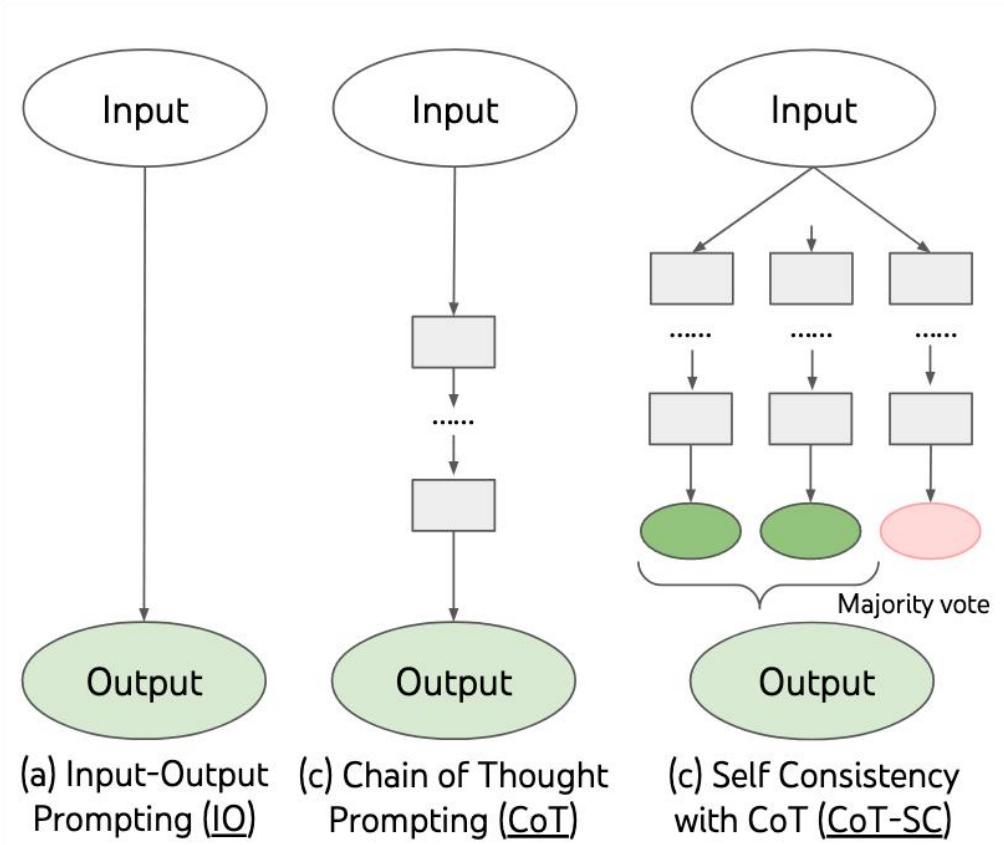


"Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters"

– Google DeepMind

Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

Chain-of-Thought (CoT) -> Tree of Thought (ToT)



Limitations:

- Locally: do not explore the branches of the tree.

  Diverse alternative reasoning paths.

Build the tree!

- Globally: do not incorporate any type of planning, lookahead, or backtracking to help evaluate these different branches.

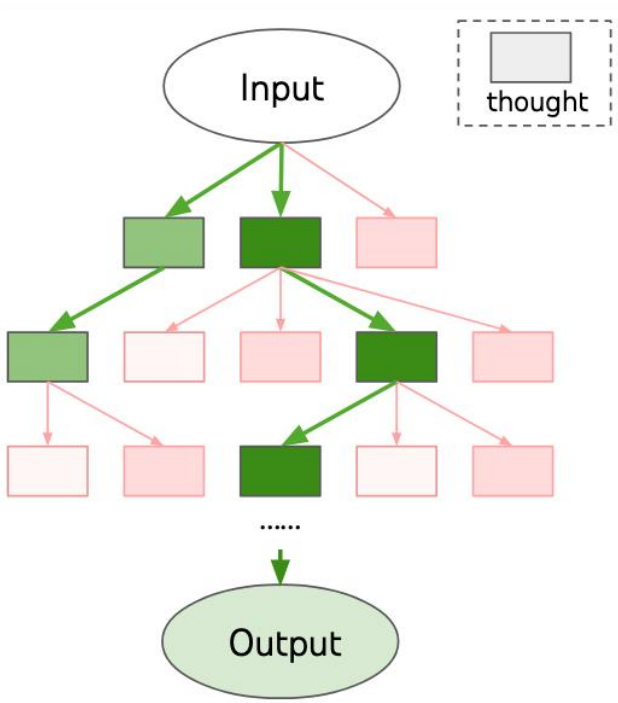
  Which branch to explore?

  When to change a reasoning path?

Conduct searching (BFS & DFS)!

Tree of Thoughts: Deliberate Problem Solving with Large Language Models

Chain-of-Thought (CoT) -> Tree of Thought (ToT)



(d) Tree of Thoughts (ToT)

Algorithm 1 ToT-BFS$(x, p_\theta, G, k, V, T, b)$

Require: Input x, LM p_θ, thought generator $G()$ & size limit k, states evaluator $V()$, step limit T, breadth limit b.

$S_0 \leftarrow \{x\}$

for $t = 1, \cdots, T$ **do**

$S'_t \leftarrow \{[s, z] \mid s \in S_{t-1}, z_t \in \mathrm{G}(p_\theta, s, k)\}$

$V_t \leftarrow V(p_\theta, S'_t)$

$S_t \leftarrow \arg\max_{S \subset S'_t, |S|=b} \sum_{s \in S} V_t(s)$

end for

return $G(p_\theta, \arg\max_{s \in S_T} V_T(s), 1)$

Algorithm 2 ToT-DFS$(s, t, p_\theta, G, k, V, T, v_{th})$

Require: Current state s, step t, LM p_θ, thought generator $G()$ and size limit k, states evaluator $V()$, step limit T, threshold v_{th}

if $t > T$ **then** record output $G(p_\theta, s, 1)$

end if

for $s' \in G(p_\theta, s, k)$ **do** ▷ sorted candidates

if $V(p_\theta, \{s'\})(s) > v_{thres}$ **then** ▷ pruning

DFS$(s', t+1)$

end if

end for

	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.

Better Tree Search Strategies?

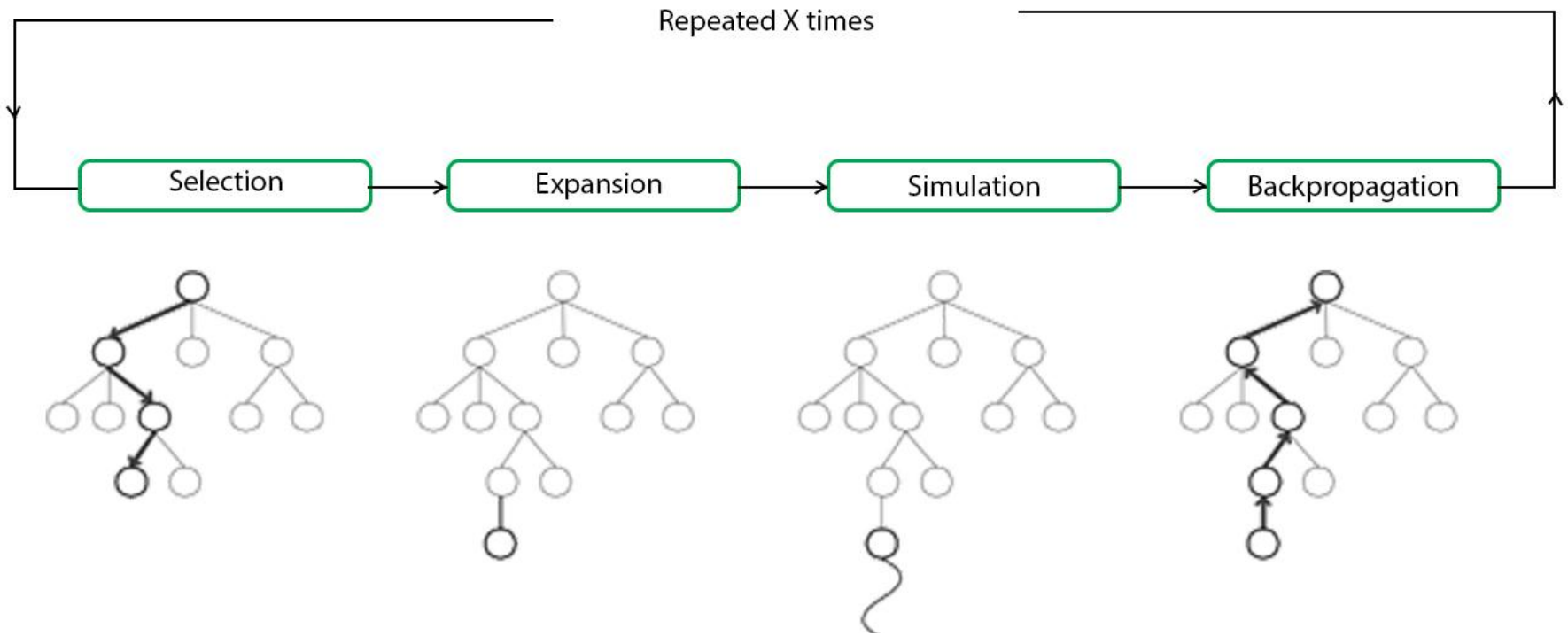
Limitations of BFS & DFS:

- Perform exhaustive search (DFS explores each branch deeply, and BFS explores all nodes level by level), leading to state explosion.
- DFS might get stuck in deep, irrelevant branches, while BFS could consume significant memory in large spaces.

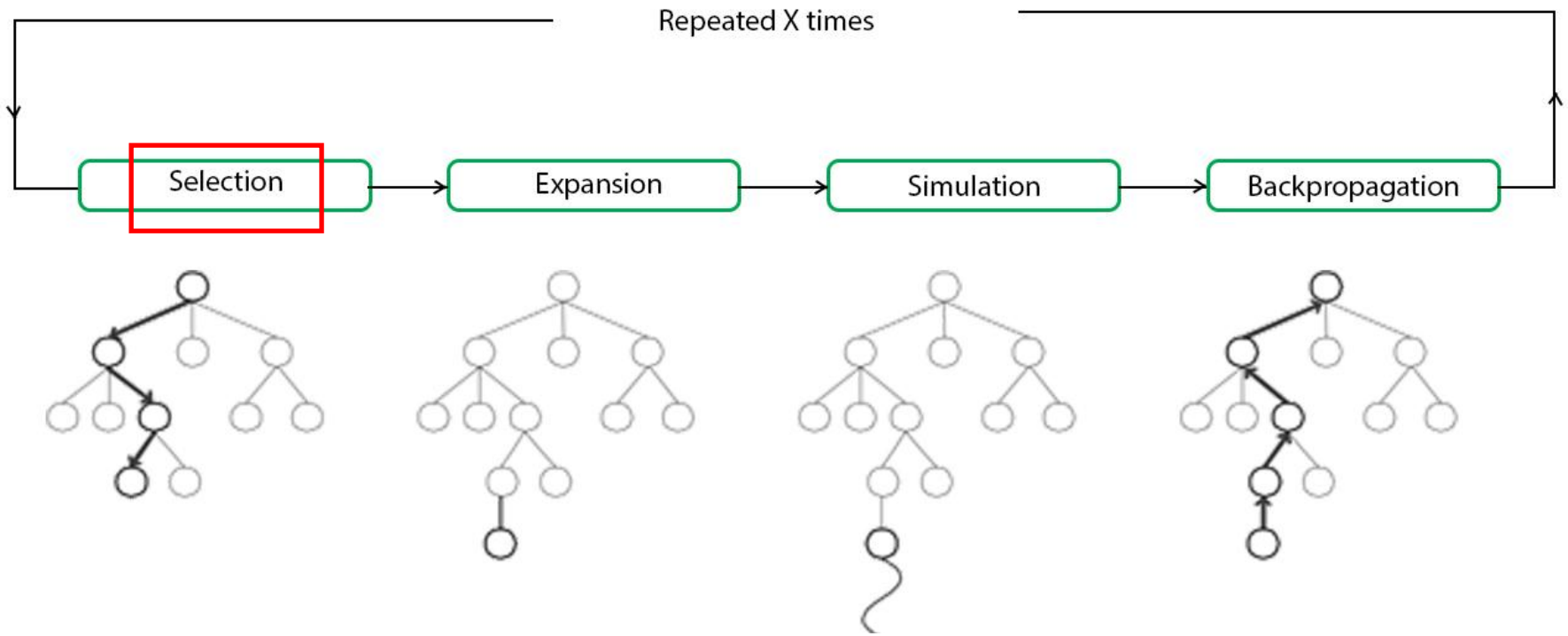
Our desire:

- Focus on simulating and expanding only the most promising parts of the tree, avoiding the inefficiencies of global search. This allows it to manage large and complex spaces more intelligently.

Monte Carlo Tree Search (MCTS)



Monte Carlo Tree Search (MCTS)



Selection: start from root **R** and select successive child nodes (based on a larger UCB value) until a leaf node **L** is reached.

Exploitation Exploration

$$UCB(i) = w_i + C * \sqrt{2 * \ln \frac{N_i}{n_i}}$$

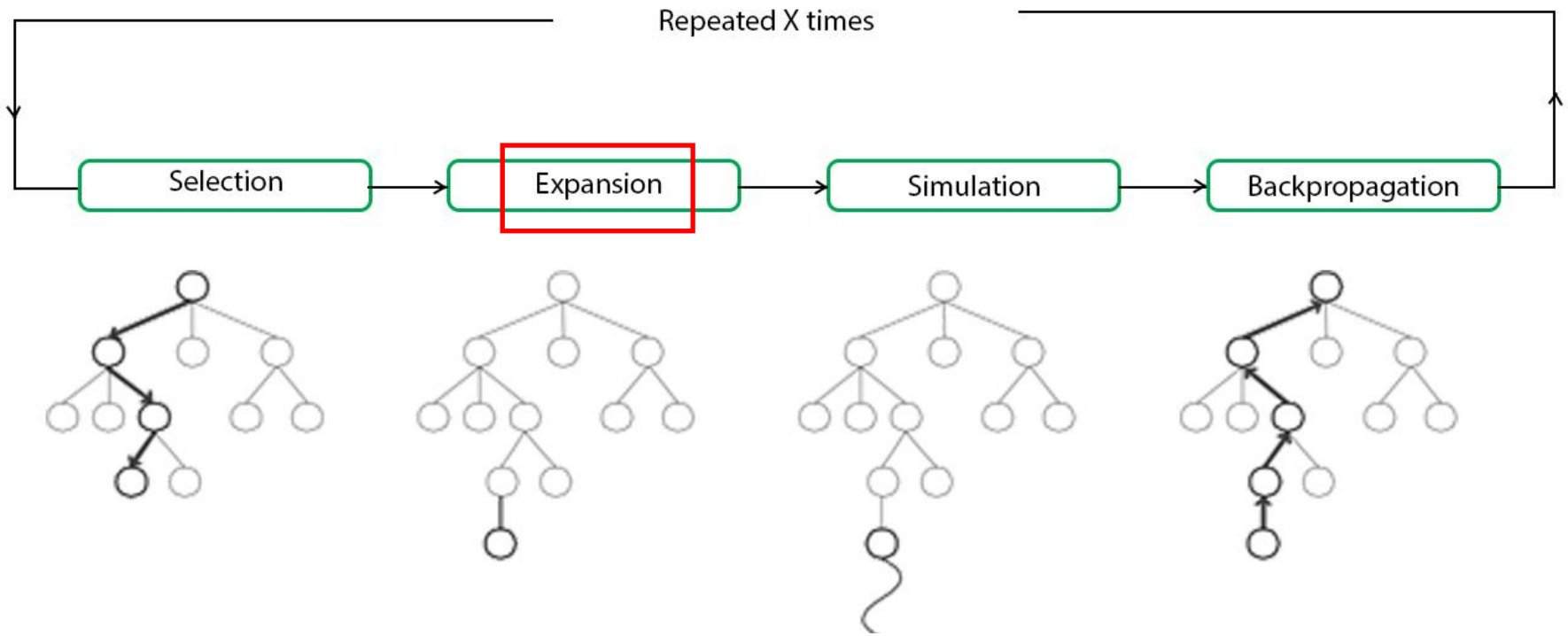
N_i: visit counts for i's parent

n_i: visit counts for i

w_i: average value

C: hyper-parameter

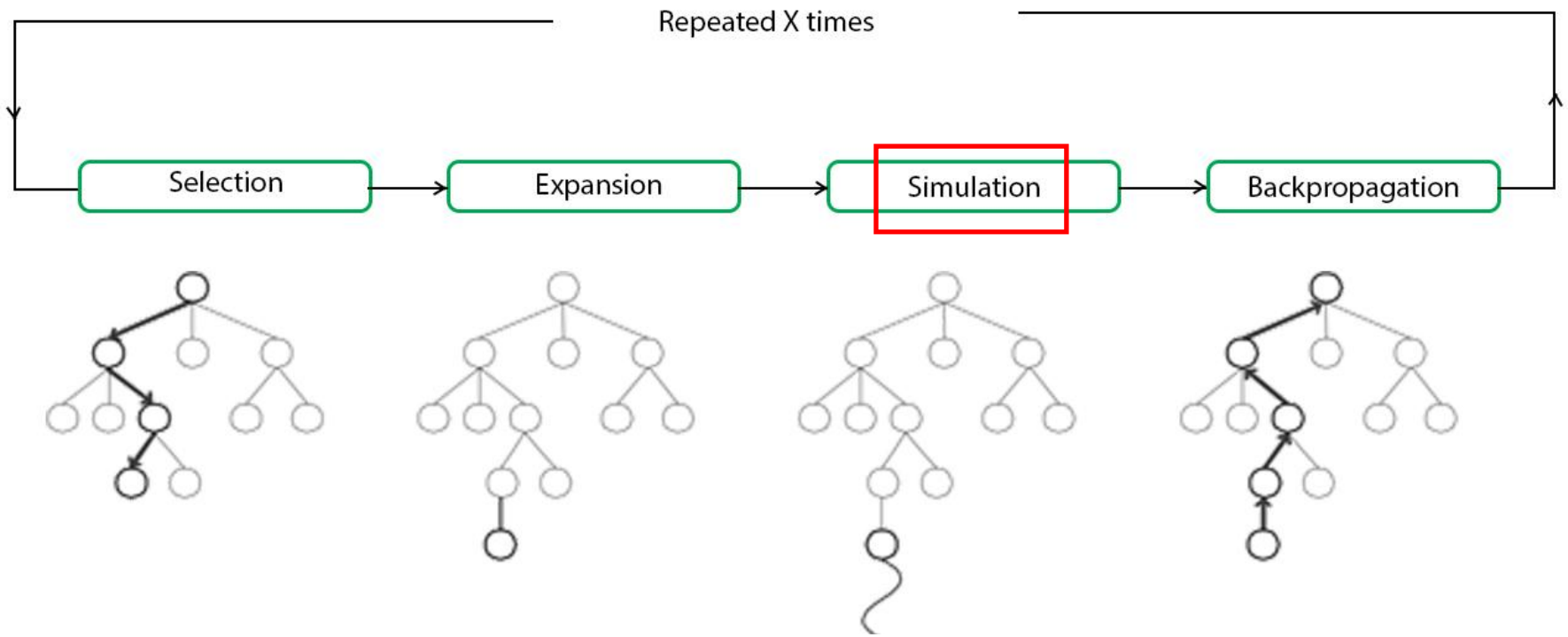
Monte Carlo Tree Search (MCTS)



Expansion: Unless **L** is a terminated state, create one (or more) child nodes and select node C from among them for further exploration.

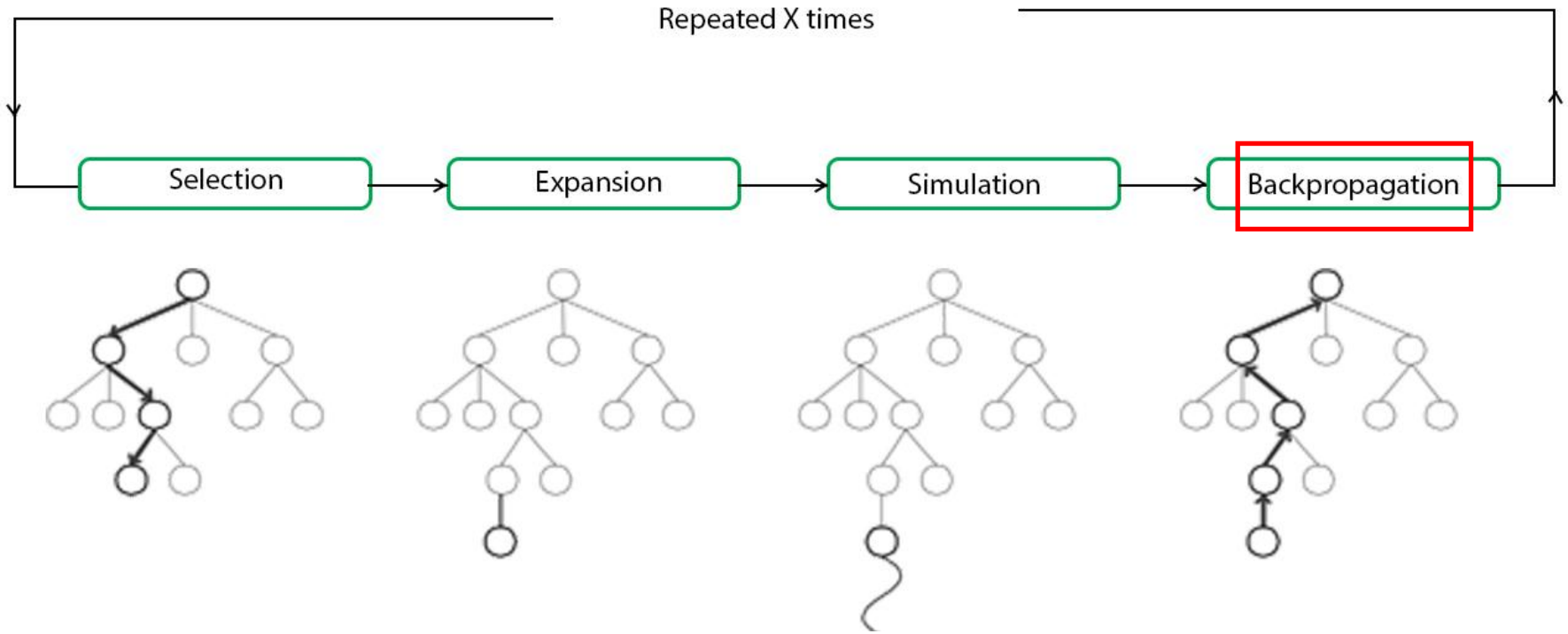
The tree expands at this stage.

Monte Carlo Tree Search (MCTS)



Simulation: Complete one random rollout from the current selected node **C** and conduct evaluation.

Monte Carlo Tree Search (MCTS)



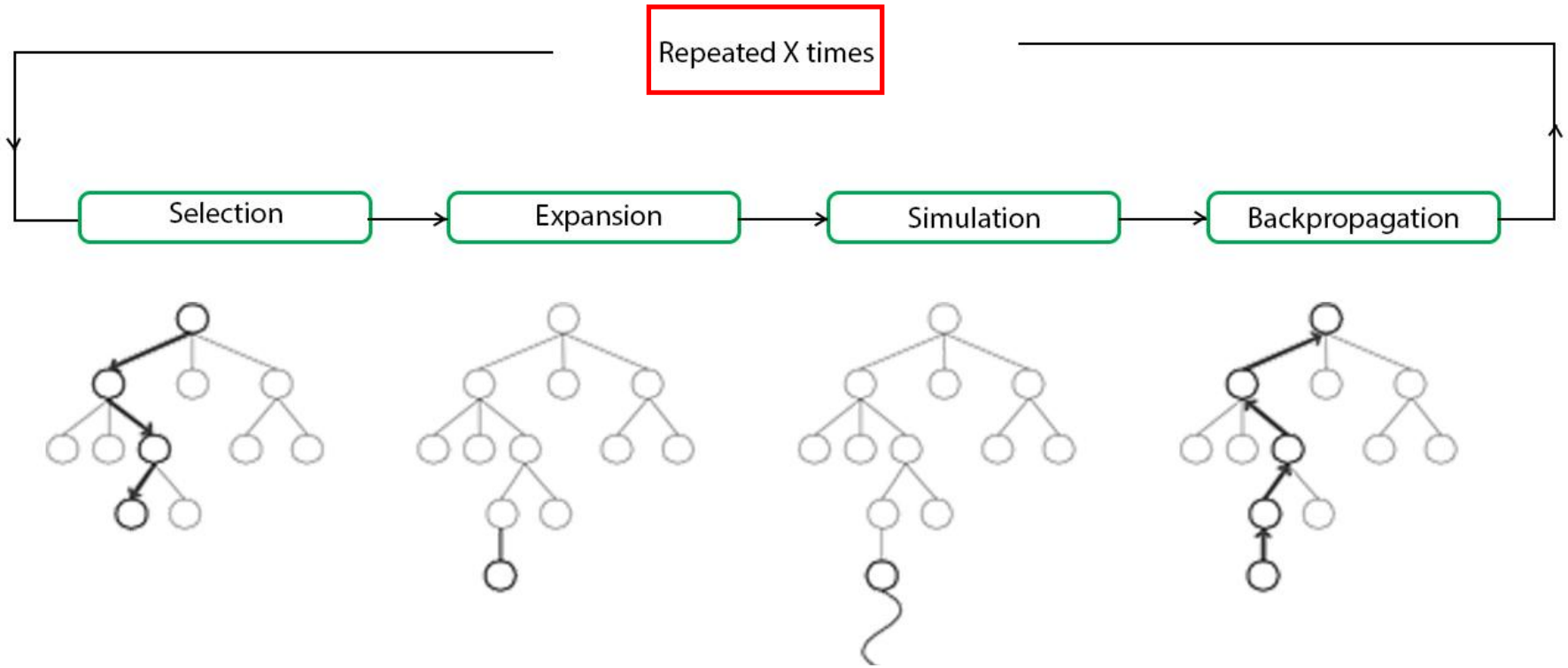
Backpropagation: Use the result of the rollout to update information in the nodes on the path from **C** to **R**.

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

$$N(s_t) \leftarrow N(s_t) + 1$$

The value and visit count of nodes can only be updated at this stage.

Monte Carlo Tree Search (MCTS)



When does the algorithm end?

- Predetermined number of iterations.
- Maximum depth constraint of the tree.
- .....

Apply MCTS to LLM for Reasoning

Node:
- A whole solution of a given problem.

Expansion:
- Self-refine

Simulation (rollout)
- Self-evaluation

Drawback:
- The granularity of the node is too big!

Furthur discussion:
- Can MCTS be high-quality **data generator**?

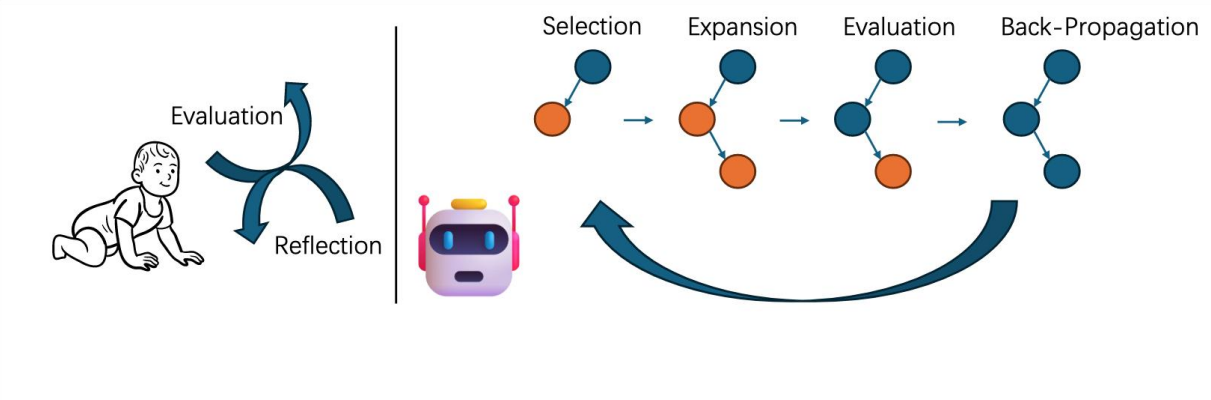


Figure 1: Agents can learn decision-making and reasoning from the trial-and-error as humans do.

Datasets	Zero-Shot CoT	One-turn Self-refine	4-rollouts MCTSr	8-rollouts MCTSr	Example Nums
GSM8K	977	1147	1227	1275	1319
	74.07%	86.96%	93.03%	96.66%	
GSM-Hard	336	440	526	600	1319
	25.47%	33.36%	39.88%	45.49%	

Table 1: Performance of MCTSr on the GSM Dataset
Base model: LLaMA3-8B

Accessing GPT-4 level Mathematical Olympiad Solutions via Monte Carlo Tree Self-refine with LLaMa-3 8B: A Technical Report

Enhancing LLMs' Reasoning Abilities (Review)

Pretraining:

- Large scale & high-quality reasoning data.
  - Domain-specific math data and synthetic data (e.g. MathPile)
  - Free-form math discussion text (why conversational abilities are needed?)
- Automatic data engineering (e.g. ProX, MAmmoTH2)

Post-training:

- SFT
- Preference learning like PPO, DPO...
  - Output-level, Process/Step-level

Inference-time:

- CoT, PoT
- Self-consistency + Majority voting
- ToT (BFS & DFS), MCTS

Self-generated data:
- Synthetic CoT (LongCoT?)
- Preference Data (+/-)

Iterative self-improvement

Searching through the problem space to find good problem-solving paths.

Overview (Paper)

- ✓ Toward Self-Improvement of LLMs via Imagination, Searching, and Criticizing

  MCTS -> SFT Data

- ✓ Chain of Preference Optimization: Improving Chain-of-Thought Reasoning in LLMs

  Tree -> Step-DPO Data

- ✓ Monte Carlo Tree Search Boosts Reasoning via Iterative Preference Learning

  MCTS -> Step-DPO Data

- ✓ SELF-EXPLORE to Avoid the PIT: Improving the Reasoning Capabilities of Language Models with Fine-grained Rewards

  Self-Explore -> Step-DPO Data

- ✓ Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents

  MCTS + DPO Data -> Solving Real-World Agentic Problems

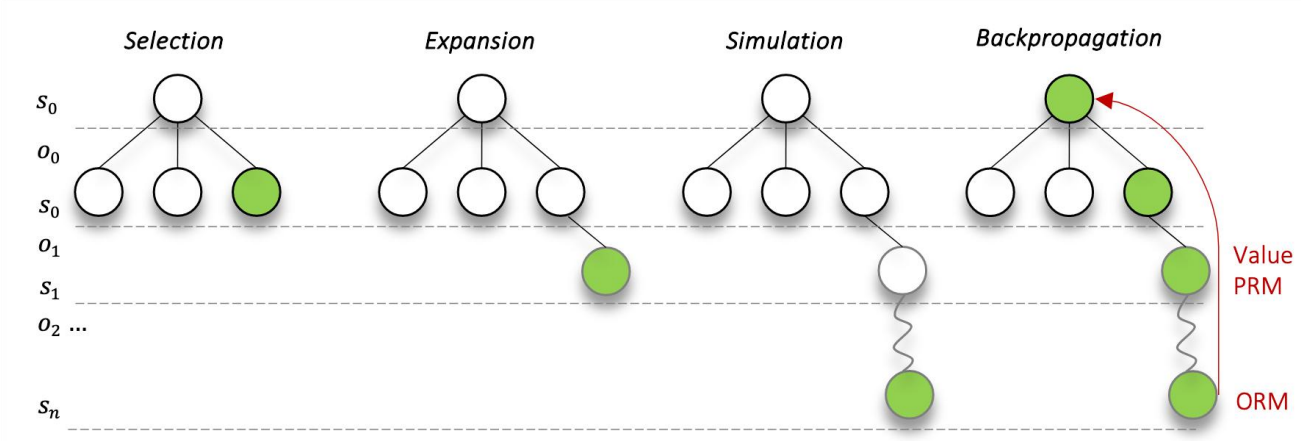
MCTS -> SFT Data

MCTS can be fine-grained data annotator.

Node:
- A step of a given problem.

Expansion:
- Action: generate the next step given the previous reasoning path.

Simulation:
- Rollout: generate until the solution ends.
- Evaluation:
  - Self-evaluation
  - Value function (reward model predicting the future reward)
  - PRM (estimate node quality)
  - ORM (assess the overall trajectory quality)



MCTS -> SFT Data

Node:
- A step of a given problem.

Expansion:
- Action: generate the next step given the previous reasoning path.

Simulation:
- Rollout: generate until the solution ends.
- Evaluation:
  - Self-evaluation
  - Value function (reward model predicting the future reward)
  - PRM (estimate node quality)
  - ORM (assess the overall trajectory quality)

Importance Weighted Expansion:

$$I(\boldsymbol{s}_t) = \max_{\boldsymbol{o}_t} |v^{\pi}([\boldsymbol{s}_t, \boldsymbol{o}_t]) - v^{\pi}(\boldsymbol{s}_t)|$$

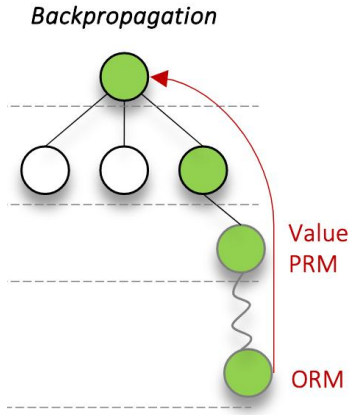
$$n(\boldsymbol{s}_t) = \max\left(c_{\texttt{min}}(t), \min\left(\lfloor \alpha I(\boldsymbol{s}_t) \rfloor, c_{\texttt{max}}(t)\right)\right)$$

State Merge:

Algorithm 2: Find Action with Minimum Distance Larger Than Threshold

Input max number of trails max_trials, threshold $thres$
Output pool of children nodes
$n \leftarrow 0$
$min\_dist \leftarrow 0$
while $n < max\_trials$ *and* $min\_d \leq thres$ **do**
  $\boldsymbol{o}_t \sim \pi(s_t)$
  $min\_d \leftarrow \min_{\boldsymbol{o} \in A_{t,\texttt{pool}}} \texttt{Dist}(\boldsymbol{o}_t, \boldsymbol{o})$
  $n \leftarrow n + 1$
end
Add $\boldsymbol{s}_{t+1} = [\boldsymbol{s}_t, \boldsymbol{o}_t]$ to the pool of children nodes

MCTS -> SFT Data



Node:
- A step of a given problem.

Expansion:
- Action: generate the next step given the previous reasoning path.

Simulation:
- Rollout: generate until the solution ends.
- Evaluation:
  - Self-evaluation
  - Value function (reward model predicting the future reward)
  - PRM (estimate node quality)
  - ORM (assess the overall trajectory quality)

Value function training:

$$\mathcal{D}_{\texttt{value}} = \{(\boldsymbol{s}_{it}, v_{it}) | i \in [N], t \in [T]\}$$

$$\boldsymbol{s}_{it} = [\boldsymbol{x}_i \cdot \boldsymbol{o}_{<it}] \quad v_{it} = \frac{1}{J}\sum_{j=1}^{J} r_{iT}^{j}$$

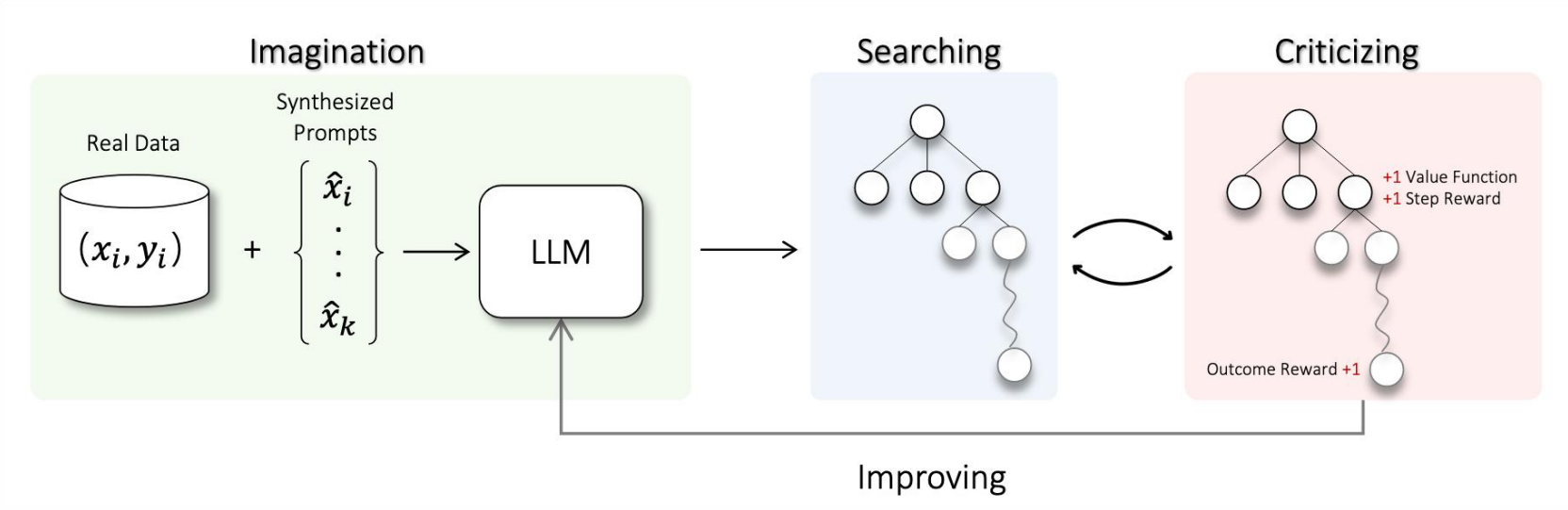
PRM training:

$$\mathcal{D}_{\texttt{PRM}} = \{(\boldsymbol{s}_{it}, \boldsymbol{o}_t, r_t^{\texttt{PRM}}) | i \in [N], t \in [T]\}$$

ORM training:

$$\mathcal{D}_{\texttt{ORM}} = \{(\boldsymbol{x}_i, \boldsymbol{o}_{1:T}^{i}, r_i^{\texttt{ORM}}) | i \in [N]\}$$

MCTS -> SFT Data



Algorithm 1: LLM self-improving loop

Input Initial dataset $\mathcal{D}^0 = \{(\boldsymbol{x}_i^0, \boldsymbol{y}_i^0) \mid i \in [N]\}$, policy model π_θ^0, reward model R, number of self-improving training loop K

Output θ^k

for $k \leftarrow 1, \ldots, K$ **do**

  Generate synthetic prompts $[\boldsymbol{x}^k] = \texttt{SYN}(\pi_\theta^{k-1}, \mathcal{D}^{k-1})$

  Collect trajectories with search algorithm, *e.g.*, MCTS guided by R.

  $[\hat{\boldsymbol{y}}^k] = \texttt{MCTS}(\pi_\theta^{k-1}, [\boldsymbol{x}^k])$

  Construct dataset $\mathcal{D}^k = \{(\boldsymbol{x}^k, \hat{\boldsymbol{y}}^k)\}$

  Update policy $\theta^k = \arg\min_\theta L(\pi_\theta^{k-1}, \mathcal{D}^k)$

end

Toward Self-Improvement of LLMs via Imagination, Searching, and Criticizing

MCTS -> SFT Data

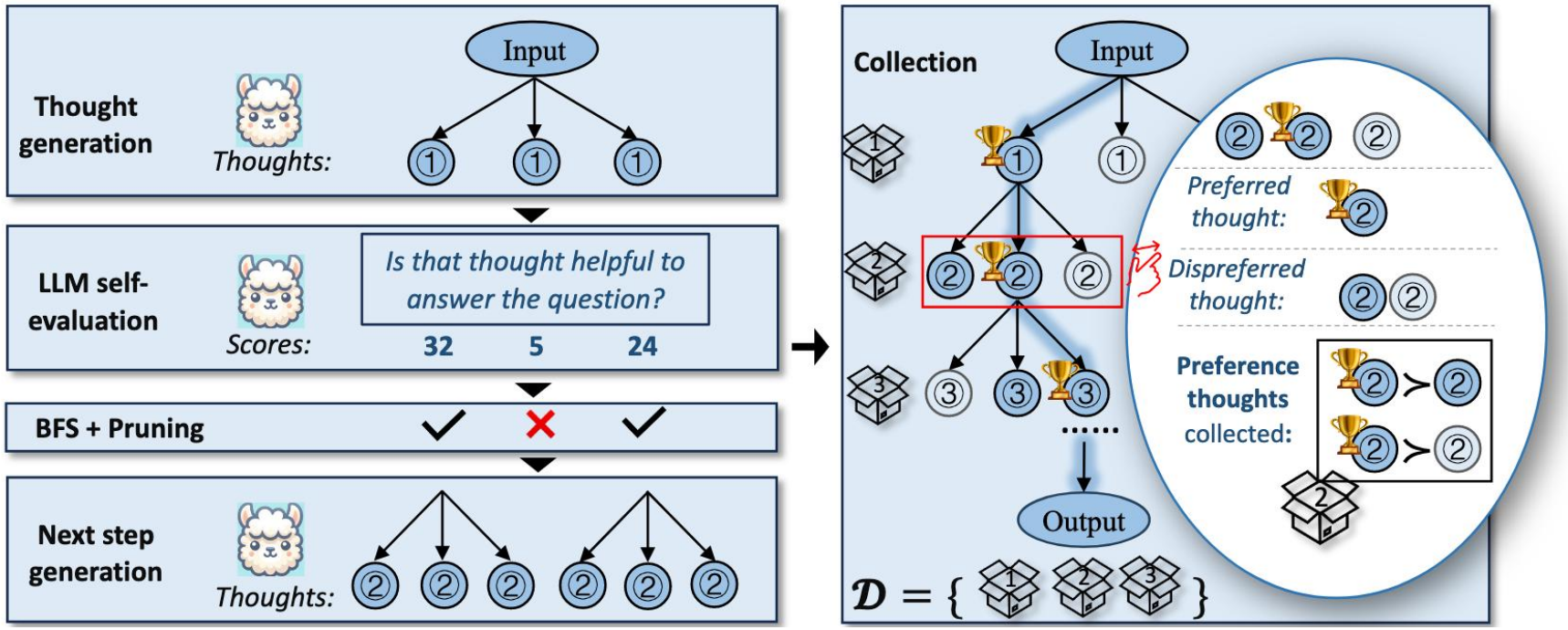
Model	Decoding	#Annotation	RN	FA	SYN	GSM8K	MATH
GPT-3.5	Sampling	-	-	-	-	80.8	35.5
GPT-4	Sampling	-	-	-	-	92.0	42.5
GPT-4 (PAL)	Sampling	-	-	-	-	94.2	51.8
Gemini 1.0 Pro	Sampling	-	-	-	-	77.9	32.6
Gemini 1.0 Ultra	Sampling	-	-	-	-	88.9	53.2
Gemini 1.5 Pro	Sampling	-	-	-	-	92.5	58.5
Claude-2	Sampling	-	-	-	-	85.2	32.5
PaLM-2 540B	Sampling	-	-	-	-	80.7	34.3
LLaMA-2 70B	Greedy	0	×	×	×	57.8	-
LLaMA-2 70B SFT	Greedy	7.5k	✓	✓	×	69.3	-
WizardMath 70B V1.0	Greedy	96k	✓	✓	×	-	20.7
ALPHALLM	Greedy	7.5k/3k	×	✓	✓	73.7	23.6
ALPHALLM	ηMCTS	7.5k/3k	×	✓	×	88.9	48.7
ALPHALLM	ηMCTS	7.5k/3k	×	✓	✓	92.0	51.0

Base model: LLaMA2-70B

Toward Self-Improvement of LLMs via Imagination, Searching, and Criticizing

Tree -> Step-DPO

From SFT data to Step-level DPO data



Searching algorithm: BFS with pruning

$$\mathcal{L}_i(\pi_\theta; \pi_{\text{ref}}) = -\log \sigma \left(\beta \log \frac{\pi_\theta(z_i^w | x, s_{i-1}^w)}{\pi_{\text{ref}}(z_i^w | x, s_{i-1}^w)} - \beta \log \frac{\pi_\theta(z_i^l | x, s_{i-1}^w)}{\pi_{\text{ref}}(z_i^l | x, s_{i-1}^w)} \right)$$

$$\mathcal{L}_{\text{CPO}}(\pi_\theta; \pi_{\text{ref}}) = \mathbb{E}_{(x, z_i^w, z_i^l, s_{i-1}^w) \sim \mathcal{D}} \left[\mathcal{L}_i(\pi_\theta; \pi_{\text{ref}}) \right]$$

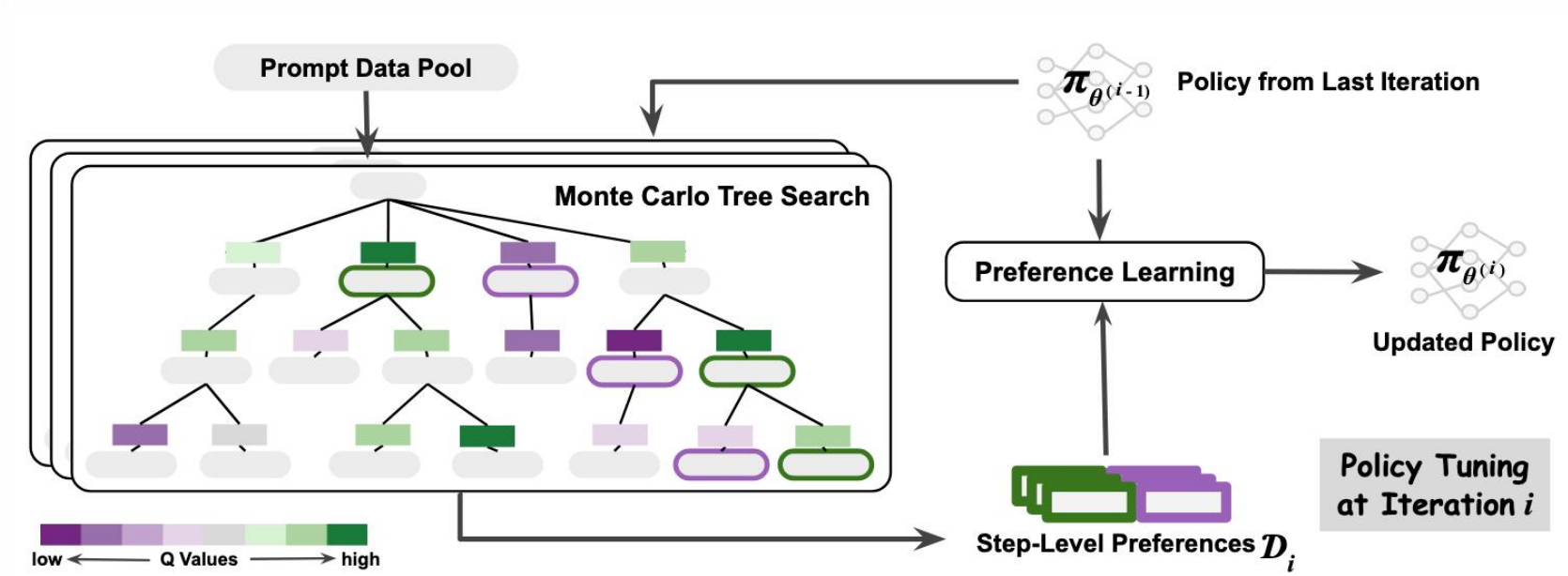
Chain of Preference Optimization: Improving Chain-of-Thought Reasoning in LLMs

Tree -> Step-DPO

		ToT [48]		CoT [40]		TS-SFT [14]		CPO (ours)	
		Acc. (%)↑	Latency (s/ins.)↓	Acc. (%)↑	Latency (s/ins.)↓	Acc. (%)↑	Latency (s/ins.)↓	Acc. (%)↑	Latency (s/ins.)↓
LLaMA2-7B									
Question Answering	Bam.	**33.6**	1168.4	29.6	37.2	30.4	36.5	**32.0***	38.2
	2Wiki.	28.6	847.6	26.3	35.7	27.6	35.5	**29.7***	35.7
	Hot.	23.0	1100.7	21.0	45.5	22.7	44.8	**24.0***	41.1
Fact Verification	FVR.	47.3	2087.1	45.8	33.8	47.5	34.0	**53.2***	36.8
	FVRS.	47.5	2539.5	44.3	40.6	46.0	40.4	**49.0***	41.2
	Vita.	50.7	2639.3	47.3	35.9	51.0	40.1	**52.7***	40.1
Arithmetric	SVA.	42.7	1861.1	37.7	33.3	43.1	30.2	**46.0***	32.1
Average Performance		39.1	1749.1	36.0	37.4	38.3	37.4	**40.9***	37.9
LLaMA2-13B									
Question Answering	Bam.	53.8	1318.3	48.0	50.5	50.8	49.6	**52.0***	50.3
	2Wiki.	**36.3**	1097.1	28.3	67.0	29.0	66.5	**30.3***	60.4
	Hot.	**32.0**	1271.0	29.0	65.2	**30.3**	65.5	**30.3**	63.8
Fact Verification	FVR.	48.8	3139.8	48.2	45.4	48.8	44.0	**49.2**	43.9
	FVRS.	**51.3**	3433.2	50.0	61.1	48.8	58.0	**50.7***	68.2
	Vita.	52.5	2933.6	46.3	48.3	49.7	51.5	**54.0***	58.6
Arithmetric	SVA.	45.7	2115.3	40.3	46.2	44.6	46.4	**50.0***	48.1
Average Performance		**45.8**	2186.9	41.4	54.8	43.1	54.5	**45.2***	56.2
Mistral-7B									
Question Answering	Bam.	**46.4**	4399.6	41.6	46.2	41.6	45.0	**45.6***	47.0
	2Wiki.	28.4	2356.9	26.7	45.1	31.0	44.2	**31.7**	46.3
	Hot.	**30.0**	4698.0	28.0	58.4	28.6	56.2	**29.4**	56.9
Fact Verification	FVR.	**61.4**	3291.3	57.9	41.8	**60.2**	41.7	59.9	40.6
	FVRS.	50.5	5537.8	48.0	51.0	49.7	49.5	**54.0***	53.0
	Vita.	52.2	4698.2	47.7	44.8	50.3	45.9	**53.7***	46.6
Arithmetric	SVA.	66.0	4623.7	65.3	41.4	59.0	41.3	**69.3***	44.9
Average Performance		47.8	4229.4	45.0	47.0	45.8	46.3	**49.1***	47.9

Chain of Preference Optimization: Improving Chain-of-Thought Reasoning in LLMs

MCTS -> Step-DPO



$$h_{\pi_\theta}^{y_w,y_l} = \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}$$

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

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

label smoothing

Monte Carlo Tree Search Boosts Reasoning via Iterative Preference Learning

MCTS -> Step-DPO

Results on math and commonsense reasoning tasks

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**

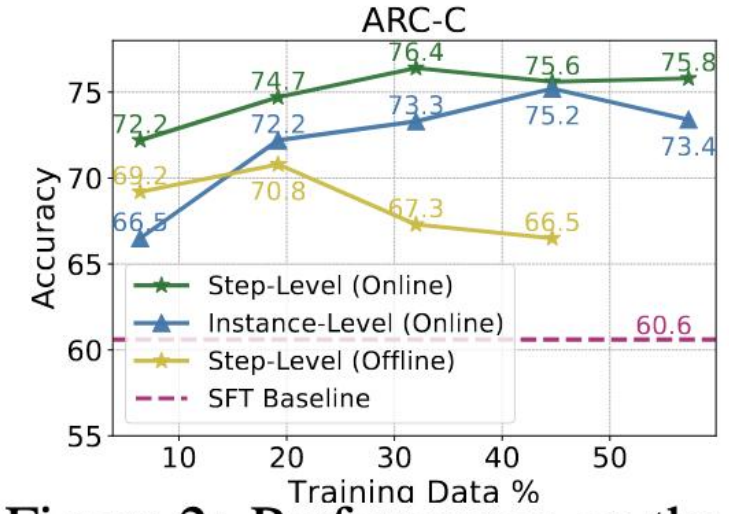
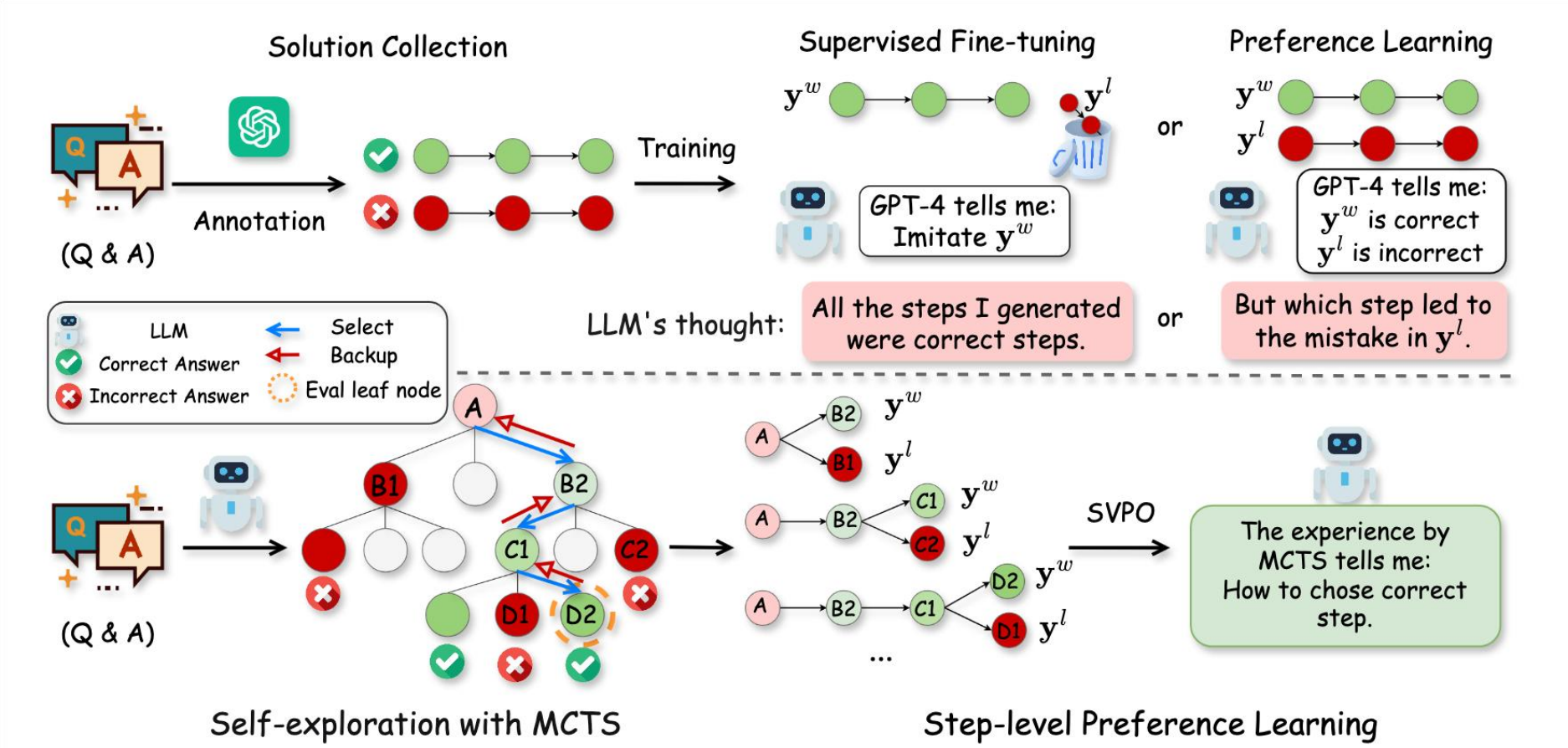


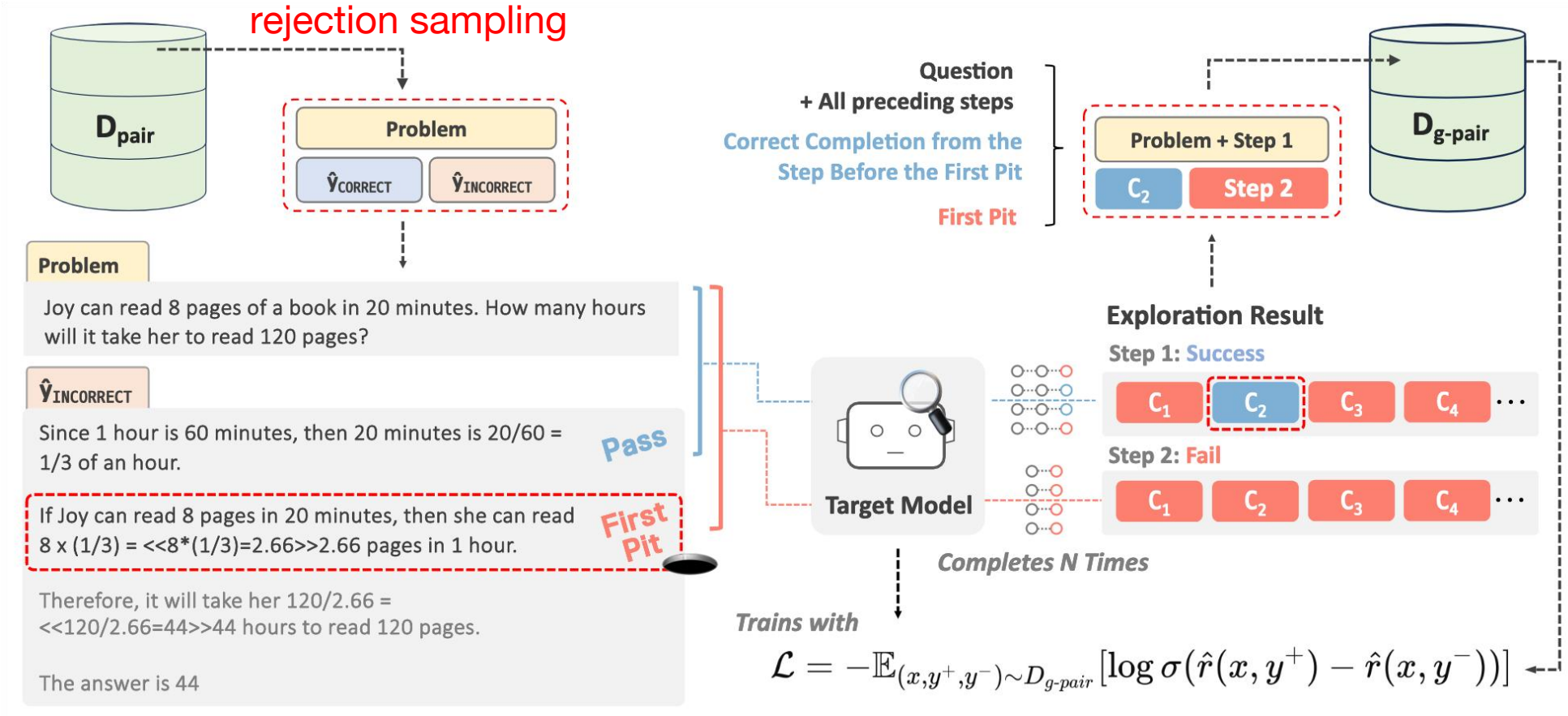
Figure 2: Performance on the validation set of ARC-C via training with different settings.

MCTS -> Step-DPO



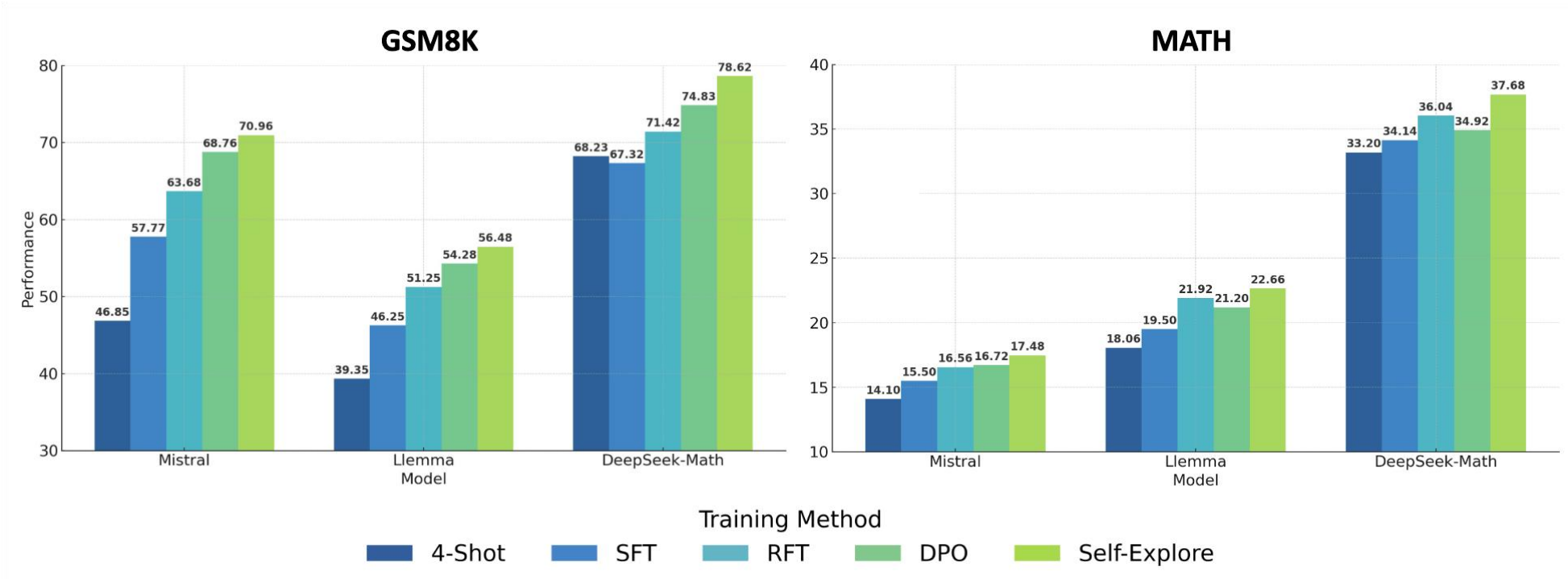
Self-Explore -> Step-DPO

Generate Step-DPO pair data without tree.



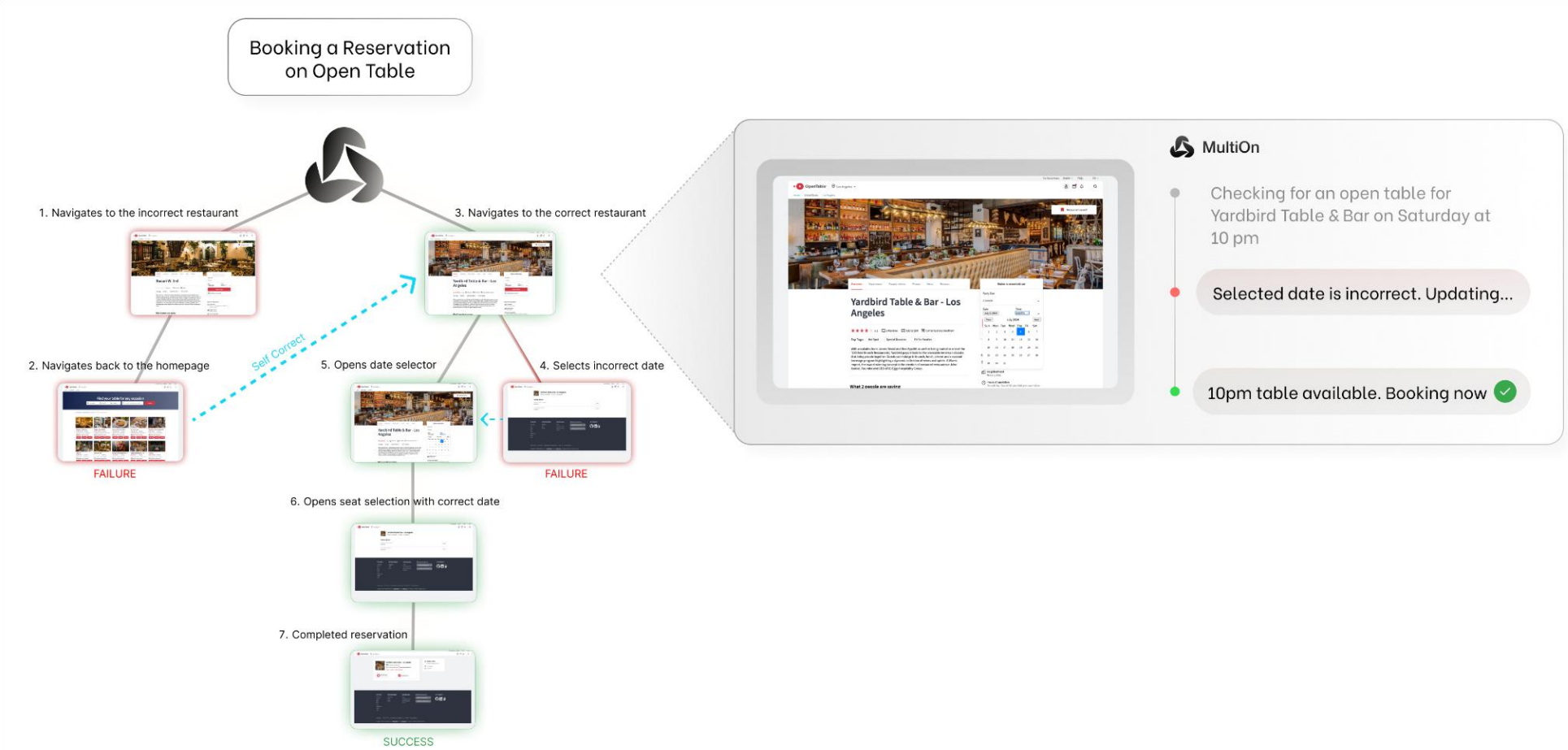
SELF-EXPLORE to Avoid the PIT: Improving the Reasoning Capabilities of Language Models with Fine-grained Rewards

Self-Explore -> Step-DPO



SELF-EXPLORE to Avoid the PIT: Improving the Reasoning Capabilities of Language Models with Fine-grained Rewards

Agent Q



Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents

Agent Q

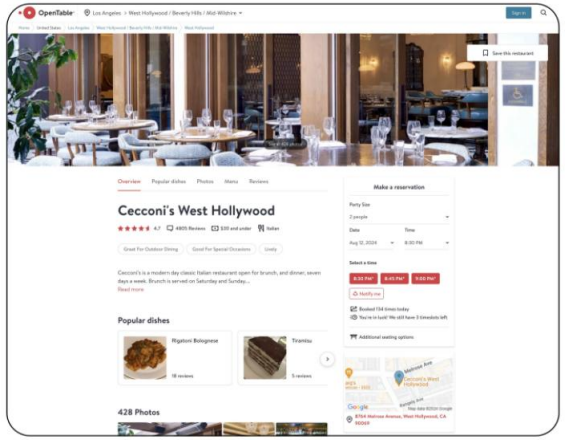
Observation:
- HTML DOM language

Action:
- Planning
- Reasoning
- Environment Action
- Explanation Action

State:
- History action + Current Observation

Agent Input

```
<SYSTEM PROMPT>
<EXECUTION HISTORY>

USER QUERY: Book a reservation for the
restaurant Ceconni's on OpenTable for
2 people on June 17 2024 at 7:00pm

CURRENT OBSERVATION:
```



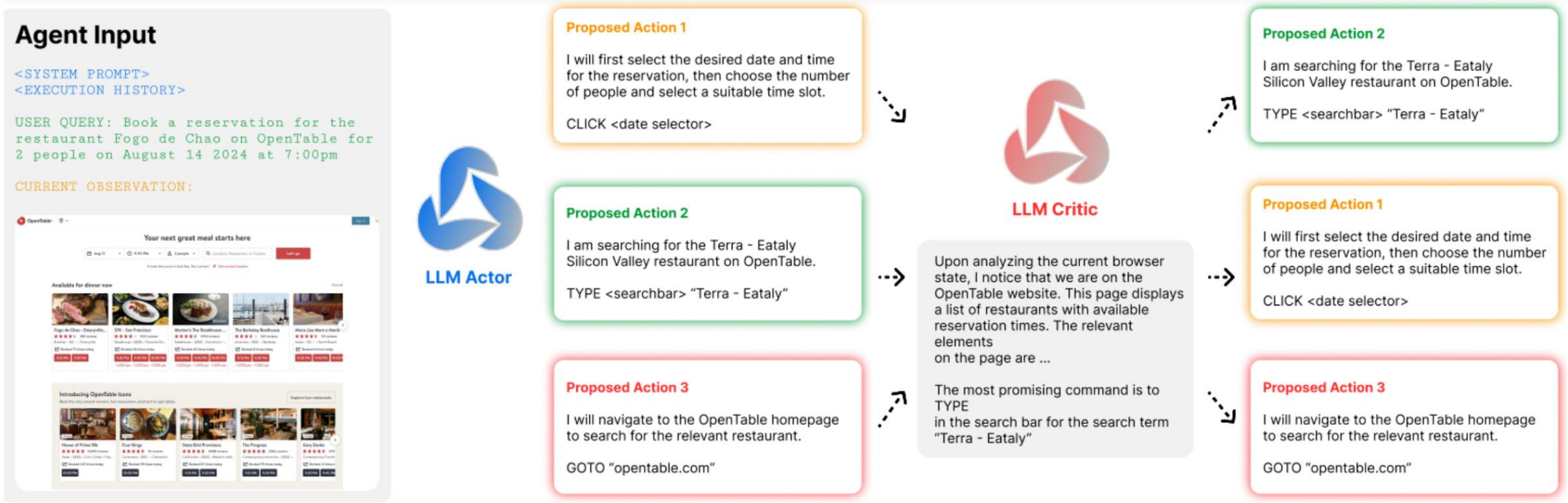
Agent Output

```
PLAN:
1. Select the date (May 22, 2024) for the
   reservation.
2. Choose the time (7:00 PM) for the
   reservation.
3. Select the party size (4 people) for the
   reservation.
4. Click on the "Find a table" button to book
   the reservation.

THOUGHT: I am currently on the Cecconi's
restaurant page on OpenTable, and I need to
select the date and time for the reservation
and choose the number of guests. I will focus
on selecting the date, time, and party size
for the reservation.

COMMANDS: CLICK <select>Date</select>

STATUS: CONTINUE
```

$$\log \pi(\mathbf{a}_1|\mathbf{h}_1) = \log \pi(\mathbf{a}_1^{\text{expl}}|\mathbf{h}_1, \mathbf{a}_1^{\text{env}}, \mathbf{a}_1^{\text{tht}}, \mathbf{a}_1^{\text{plan}}) + \log \pi(\mathbf{a}_1^{\text{env}}|\mathbf{h}_1, \mathbf{a}_1^{\text{tht}}, \mathbf{a}_1^{\text{plan}}) + \log \pi(\mathbf{a}_1^{\text{tht}}|\mathbf{h}_1, \mathbf{a}_1^{\text{plan}}) + \log \pi(\mathbf{a}_1^{\text{plan}}|\mathbf{h}_1)$$

$$\log \pi(\mathbf{a}_t|\mathbf{h}_t) = \log \pi(\mathbf{a}_t^{\text{expl}}|\mathbf{h}_t, \mathbf{a}_t^{\text{env}}, \mathbf{a}_t^{\text{tht}}) + \log \pi(\mathbf{a}_t^{\text{env}}|\mathbf{h}_t, \mathbf{a}_t^{\text{tht}}) + \log \pi(\mathbf{a}_t^{\text{tht}}|\mathbf{h}_t)$$

Agent Q



MCTS+Self-Critique+Iterative DPO

Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents

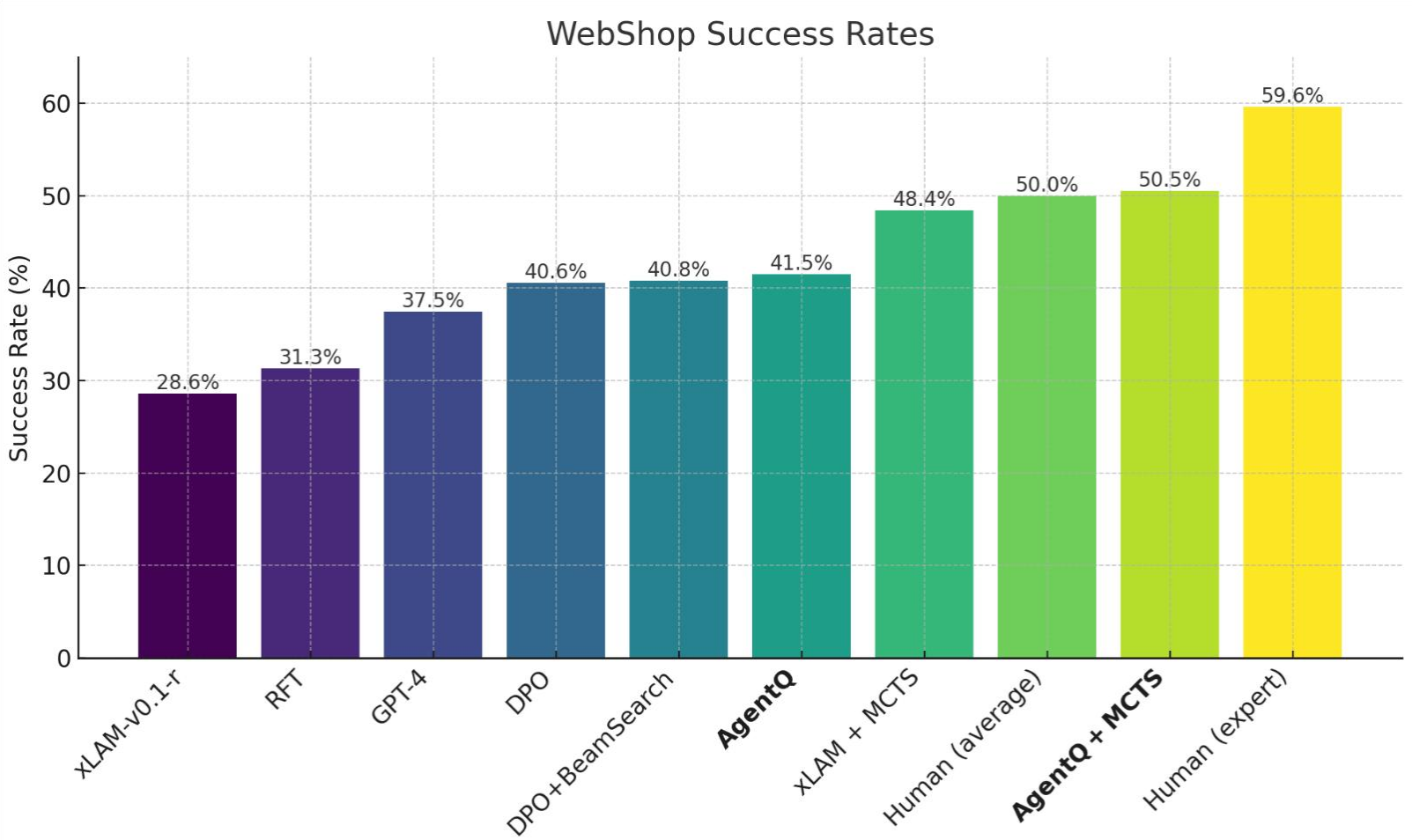
Agent Q

Algorithm 1 MCTS Guided Direct Preference Optimization

Input: π_{θ_0}: initial LLM policy, $\mathcal{D}_T$: dataset of tasks the agent must complete in the environment, N: number of iterations, B: number of samples per iteration, T: MCTS tree depth, $\mathcal{B}$: replay buffer, $\theta_{\text{threshold}}$: value threshold in (10), K: number of actions to sample for MCTS
Output: π_{θ_N}, the trained LLM policy
for $i = 1$ to N **do**
  $\pi_{\text{ref}} \leftarrow \pi_{\theta_i}$, $\pi_{\theta_i} \leftarrow \pi_{\theta_{i-1}}$
  Sample a batch of B tasks from $\mathcal{D}_T$
  for each task in batch **do**
    Initialize the root node $\mathbf{h}_0$
    for $t = 1$ to T **do**
      Selection: Traverse tree from the root node to a leaf node using tree policy (UCB1; 7)
      Trajectory Rollout: From the selected node's trace, roll out the trajectory using π_{θ_i} until a terminal state is reached
      Backpropagation: Backpropagate the value estimate bottom-up (8)
    end for
    Collect trajectories from rollouts and store them in replay buffer $\mathcal{B}$
  end for
  Construct preference pairs $\mathcal{D}_P = \{(\mathbf{h}_t, \mathbf{a}_t^w, \mathbf{a}_t^l)\}_{t=1}^{T-1}$ where $\mathbf{h}_t \sim \mathcal{D}_P$. For each node at step level t, compare each pair of child nodes, and construct the pair of generated actions $(\mathbf{a}^w, \mathbf{a}^l)$ if the values of taking the action, $|Q(\mathbf{h}_t, \mathbf{a}^w) - Q(\mathbf{h}_t, \mathbf{a}^l)| > \theta_{\text{threshold}}$, where $Q(\mathbf{h}_t, \mathbf{a}^w)$ and $Q(\mathbf{h}_t, \mathbf{a}^l)$ are computed using (10)
  Optimize LLM policy π_{θ_i} using DPO objective in Eq. (5) with $\mathcal{D}_P$ and π_{ref}
end for

Use MCTS to generate pair-wise trajectories for DPO.

Agent Q



Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents

Thanks.