Revisiting Tree Search for LLMs: Gumbel and Sequential Halving for Budget-Scalable Reasoning

Leonid Ugadiarov[1,2], **Yuri Kuratov**[1,2], **Aleksandr Panov**[1,2], **Alexey Skrynnik**[1,2]

[1]AXXX, Moscow, Russia
[2]MIRAI, Moscow, Russia

Abstract

Neural tree search is a powerful decision-making algorithm widely used in complex domains such as game playing and model-based reinforcement learning. Recent work has applied AlphaZero-style tree search to enhance the reasoning capabilities of Large Language Models (LLMs) during inference, but we find that this approach suffers from a scaling failure: on GSM8K and Game24, accuracy drops as the search budget increases. In this paper, we present ReSCALE, an adaptation of Gumbel AlphaZero MCTS that replaces Dirichlet noise and PUCT selection with Gumbel sampling and Sequential Halving, restoring monotonic scaling without changes to the model or its training. ReSCALE reaches 58.4% on GSM8K and 85.3% on Game24 at budgets where the baseline degrades. Ablations confirm that Sequential Halving is the primary driver of the improvement.

Code — https://github.com/CognitiveAISystems/ReSCALE

1 Introduction

Large language models (LLMs) have achieved remarkable results on reasoning benchmarks (OpenAI 2024; Guo et al. 2025). However, even such reasoning-oriented models still struggle with problems that require multi-step reasoning and planning (Wan et al. 2024a; Kambhampati et al. 2024; Valmeekam, Stechly, and Kambhampati 2024). A common way to enable step-by-step reasoning in non-reasoning LLMs is chain-of-thought (CoT) prompting (Wei et al. 2022), where the model is asked to generate intermediate reasoning steps before providing an answer. CoT can substantially improve non-reasoning LLMs' accuracy on arithmetic, logical, and general tasks, but in its standard form it still represents a single reasoning attempt: the model follows one sampled trajectory of thoughts, with no mechanism to explore alternatives and backtrack.

To move beyond this one-shot CoT setting, recent work augments LLMs with search over multiple reasoning trajectories, ranging from sampling multiple CoT traces (Wang et al. 2023) to more structured tree-search LLM methods (Yao et al. 2023; Hao et al. 2023). In these methods, models explore alternative reasoning paths before producing a final answer. Tree-search-based approaches are appealing because they provide a principled way to allocate test-time compute (Snell et al. 2025), trading off exploration and exploitation under a limited budget (Pallagani et al. 2024).

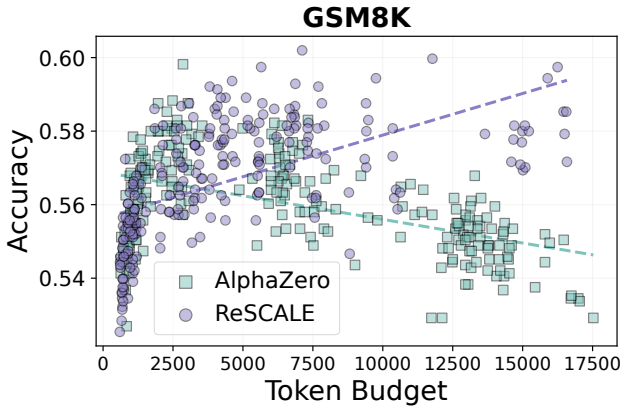

Figure 1: **Gumbel + Sequential Halving enables scaling of MCTS for reasoning.** Comparison of tree-search methods for LLM reasoning on GSM8K across increasing token budgets. The standard AlphaZero-style approach plateaus and declines at higher budgets, while proposed ReSCALE decoding achieves sustained accuracy gains, demonstrating better scaling with additional compute.

This idea builds on a long line of progress in Monte Carlo Tree Search (MCTS), from the upper confidence bound for trees algorithm (Kocsis and Szepesvári 2006) to AlphaZero-style methods that incorporate learned policies and value functions to achieve superhuman play in Go, chess, shogi, and Atari (Silver et al. 2017; Schrittwieser et al. 2020), and even to the discovery of improved matrix-multiplication algorithms (Fawzi et al. 2022).

Wan et al. (2024b) apply AlphaZero-like tree search to LLM decoding, using a learned value model to guide MCTS. With small to moderate simulation budgets, ranging from 5 to 15 simulations and involving around 500 tokens, this approach substantially outperforms standard decoding strategies. However, we show that when the computation budget is scaled beyond this range by increasing the number of simulations or the search depth, accuracy on the GSM8K dataset plateaus and even slightly declines (Figure 1), while the number of visited nodes and generated tokens contin-

ues to grow. We observe similar degradation at large budgets on the Game24 dataset. This behavior resembles the lookahead pathology phenomenon, where increased search effort can paradoxically degrade decision quality (Nguyen and Ramanujan 2024).

A well-designed search algorithm should exhibit the opposite behavior: performance should improve as the search budget increases. Classical work on planning and MCTS has introduced techniques that are well suited to this fixed-budget perspective, including Gumbel-based root sampling (Danihelka et al. 2022) and Sequential Halving (Karnin, Koren, and Somekh 2013) algorithms for best-arm identification. To the best of our knowledge, Gumbel sampling and Sequential Halving have not yet been integrated into AlphaZero-like tree-search frameworks for LLM decoding. In this study, we examine the effects of incorporating these two techniques into such a framework. We find that these components can significantly improve performance with larger simulation budgets.

Our contributions are:

- We identify a scaling failure in AlphaZero-style tree search for LLM reasoning: on GSM8K and Game24, accuracy *decreases* beyond a moderate simulation budget, even as the algorithm explores more tree nodes.
- We trace this failure to the action selection mechanism and show that replacing it with Gumbel-based sampling and Sequential Halving restores monotonic scaling, without any changes to the model or its training.
- We extensively evaluate ReSCALE on GSM8K and Game24, demonstrating that it effectively converts additional compute into accuracy gains in regimes where the baseline degrades.

2 Method

We first describe the background components shared with prior work: the MDP formulation, action space, value network, and standard AlphaZero tree search, and then present our Gumbel AlphaZero adaptation for LLM decoding.

Background

MDP Formulation We formulate language generation as a token-level Markov Decision Process (MDP) (Puterman 1994). Both actions and states reside in token space: the state $s_t = [q_1, \ldots, q_M, o_0, \ldots, o_{t-1}]$ concatenates the input prompt and previously generated tokens, and the LLM serves as a policy π_θ^{token} that autoregressively selects the next token o_t from vocabulary $\mathcal{O}$, yielding a deterministic transition $s_{t+1} = s_t \oplus o_t$.

The reward function $r(s_t)$ represents the task objective (defined by rules or a pre-trained reward model) and can be sparse (e.g., binary correctness of the final state s_T) or dense. Our method uses tree search to guide LLM generation toward maximizing the cumulative reward $\sum_{t=0}^{T} r(s_t)$.

Action Space Following Wan et al. (2024b), we define an action as a sequence of tokens ending with a special delimiter `DELIM` (e.g., a newline) or a `STOP` token. The full model output is thus a sequence of sentence-level actions $[a_0, a_1, \ldots, a_{d-1}]$. Compared to token-level actions ($a_i = o_i$), this reduces the search tree depth at the cost of a larger branching factor. During tree expansion, we sample w actions and normalize their probabilities to sum to 1, denoting the result as $p(a)$. We also restrict the maximum tree depth to d, making the search computationally tractable.

Value Network AlphaZero-like tree search (Silver et al. 2018) requires a value network v_ϕ that estimates the expected cumulative reward from a given state. We use a value network that shares the policy architecture but adds a scalar value head. The network is trained with Monte Carlo return targets (Sutton and Barto 2018) using a mean-squared error objective; training details are provided in Section 3.

AlphaZero Tree Search The standard AlphaZero tree search (Silver et al. 2018) performs multiple simulations from the root state s_t, where each simulation is a single traversal from the root to an unexpanded node. During traversal, actions are selected via the PUCT algorithm (Rosin 2011); at the unexpanded node, new children are added to the tree and the value estimate is backpropagated to all ancestors. After all simulations, the action is chosen proportionally to exponentiated visit counts: $a_{t+1} \sim N(s_t, a)^{1/\tau} / \sum_b N(s_t, b)^{1/\tau}$, where $N(s_t, a)$ is the number of times action a was selected from s_t.

ReSCALE: Gumbel MCTS with Sequential Halving for LLM Decoding

An overview of the ReSCALE (Reasoning via Scalable Compute Allocation for LLM Exploration) approach is shown in Figure 2. Compared to the standard AlphaZero tree search, Gumbel AlphaZero (Danihelka et al. 2022) replaces Dirichlet noise and PUCT-based selection at the root with Gumbel noise and Sequential Halving, uses an improved policy with mixed value approximation at non-root nodes, and selects the final action deterministically.

Selection at the Root Node Each action a^i at the root is scored by $f^i = g(a^i) + \log p(a^i)$, where $g(a^i)$ is sampled Gumbel noise. After simulations, scores are refined with the value-based monotonic function σ, where $c_{\text{visit}} = 50$:

$$\sigma(v(s_t \oplus a)) = (c_{\text{visit}} + \max_{a^*} N(s_t, a^*))\, v(s_t \oplus a).$$

> ***Takeaway:*** Gumbel noise enables sampling actions without replacement while providing policy improvement guarantees, replacing the heuristic Dirichlet exploration noise of standard AlphaZero.

The simulation budget N is distributed via Sequential Halving. Starting with the top $M \leq w$ actions ranked by their Gumbel scores, the procedure runs $N/\lceil \log_2 M \rceil$ simulations split evenly among surviving actions, then eliminates the bottom half based on updated scores $f^i + \sigma(v(s_t \oplus a^i))$. This repeats until a single action remains: $s_{t+1} = s_t \oplus a_{\text{next}}$.

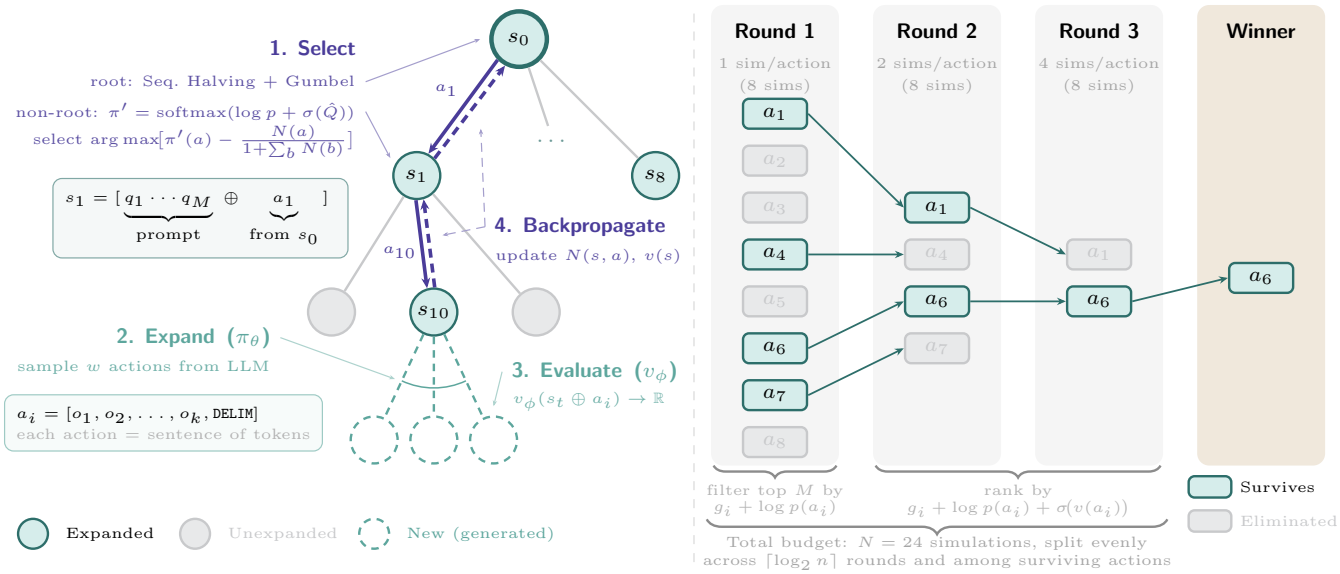

Figure 2: **Left:** A single simulation in tree search with sentence-level actions. Each simulation traverses from the root to a leaf, selecting actions via Sequential Halving with Gumbel noise at the root and an improved policy at non-root nodes. The selected leaf is expanded by sampling w actions from the LLM (π_θ), each evaluated by the value network v_ϕ, and the resulting values are backpropagated to update ancestor statistics. **Right:** Sequential Halving at the root node. From $M = 8$ candidate actions, the budget of $N = 24$ simulations is split evenly across $\lceil \log_2 M \rceil = 3$ rounds. After each round, the bottom half of actions (by score $g_i + \log p(a_i) + \sigma(v(a_i))$) is eliminated, until a single winner remains.

Takeaway: Sequential Halving focuses the simulation budget on the most promising actions by progressively eliminating weaker ones, improving efficiency — important given the cost of LLM forward passes.

Selection at Non-Root Nodes At a non-root node s_t, actions are selected using an improved policy that combines the prior action probabilities with value estimates:

$$\pi'(a) = \text{softmax}\big(\log p(a) + \sigma(v_c(s_t \oplus a))\big),$$

where v_c is a *completed* value: for visited actions ($N(s_t, a) > 0$) we use the backpropagated value $v(s_t \oplus a)$, and for unvisited actions we use a mixed approximation:

$$v_c(s_t \oplus a) = \begin{cases} v(s_t \oplus a), & \text{if } N(s_t, a) > 0, \\ v_{\text{mix}}(s_t \oplus a), & \text{otherwise.} \end{cases}$$

Let $N_\Sigma = \sum_b N(s_t, b)$ denote the total visit count at s_t and $\bar{v}_t = \frac{\sum_{b:\, N(s_t,b)>0} p(b)\, v(s_t \oplus b)}{\sum_{b:\, N(s_t,b)>0} p(b)}$ the policy-weighted mean value over visited sibling actions, serving as an empirical baseline for unvisited ones. Then:

$$v_{\text{mix}}(s_t \oplus a) = \frac{v_\phi(s_t \oplus a) + N_\Sigma \cdot \bar{v}_t}{1 + N_\Sigma}.$$

The final action is chosen in a deterministic way according to π', with a correction that uses the normalized visit counts:

$$a_{\text{next}} = \arg\max_a \left(\pi'(a) - \frac{N(s_t, a)}{1 + \sum_b N(s_t, b)} \right).$$

Takeaway: The mixed value approximation interpolates between the value network's prediction (when few actions are visited) and the observed mean of visited siblings (as data accumulates), preventing overcommitment to inaccurate estimates for unexplored actions.

Expansion and Evaluation When the selection phase reaches a non-terminal leaf node s_t, we expand it by sampling w actions $\{a_i\}_{i=0}^{w-1}$ from the LLM and adding child nodes $\{s_t \oplus a_i\}_{i=0}^{w-1}$ to the tree. Each child is then evaluated by the value network, yielding estimates $v_\phi(s_t \oplus a_i)$ that are used in backpropagation.

Backpropagation Once a leaf is expanded and evaluated, its value v_{leaf} is propagated upward along the path back to the root. At each ancestor node s on this path, the visit count $N(s, a)$ for the action a that was chosen at s is incremented, and the value estimate is updated as a running mean:

$$v(s) \leftarrow \frac{v(s) \cdot N(s, a) + v_{\text{leaf}}}{N(s, a) + 1}.$$

Here v_{leaf} is either v_ϕ for non-terminal leaves or the environment reward for terminal states. The same v_{leaf} is used at every node along the path.

3 Experiments

The goal of our experiments is to compare the performance of the standard AlphaZero tree search with its Gumbel AlphaZero variant. We follow the experimental setup from prior work (Wan et al. 2024b) where the AlphaZero tree search for LLM decoding was proposed.

		GSM8K			Game24		
Tree Search	Budget	Tokens	Acc., %	Max. Acc., %	Tokens	Acc., %	Max. Acc., %
AlphaZero	Small	0.5K–2K	56.0 ± 1.2	58.8	0.2K–2K	74.4 ± 12.0	86.7
ReSCALE			55.1 ± 1.1	58.6		71.6 ± 10.5	81.2
AlphaZero	Medium	6K–8K	56.7 ± 0.9	58.3	2K–4K	84.3 ± 1.0	86.2
ReSCALE			57.9 ± 1.0	60.2		83.4 ± 1.4	85.9
AlphaZero	Large	16K–18K	53.6 ± 0.7	55.0	4K–6K	82.9 ± 1.1	84.8
ReSCALE			58.4 ± 0.9	59.7		85.3 ± 0.6	85.9
Best-of-N	N = 32	3.5K	53.4 ± 0.4	-	2.5K	54.1 ± 1.0	-

Table 1: Combined results on GSM8K and Game24. For each budget level, mean accuracy $\pm$ std is computed across hyperparameter configurations; for Best-of-N, across three random seeds. Best single-configuration accuracy (Max. Acc.) is also reported. On GSM8K, AlphaZero accuracy drops at Large budget (53.6%) while ReSCALE continues to improve (58.4%), both surpassing the Best-of-N baseline (53.4%). On Game24, AlphaZero similarly degrades at Large budget (82.9%), whereas ReSCALE continues to improve (85.3%), both outperforming the Best-of-N baseline (54.1%).

In particular, we evaluate both methods on tasks that use sentence-level actions. We consider the mathematical reasoning dataset GSM8K (Cobbe et al. 2021) and the mathematical planning task Game24 (Yao et al. 2023). We use Llama2-7B (Touvron et al. 2023) as the base language model. The `DELIM` token is the newline character.

Experimental Setup We first perform supervised fine-tuning (SFT) of Llama2-7B on each task's training set. For every example, the target is constructed by concatenating all chain-of-thought steps (separated by the `DELIM` token), followed by the final line: "The answer is {answer}." SFT is performed for three epochs with end-of-epoch checkpoints.

We then train the value network v_ϕ. To construct the value-training dataset, we sample 100 completions per example from the task's training set using each SFT checkpoint with temperature 0.7, remove duplicates, and retain 17 unique completions per checkpoint (51 in total). The completions are split into sentence-level actions using the `DELIM` token. For each completion, we locate the first action containing "The answer is {answer}" and check whether the predicted answer is correct. If the answer is correct, that state receives a reward of 1; otherwise, it receives 0. All preceding states receive a reward of 0. Value targets are computed using a Monte Carlo estimate of the return. The value network shares the LLM backbone with an additional MLP head and is trained for three epochs.

Experimental Results We compare AlphaZero and ReSCALE tree search across three budget levels, progressively increasing tree width, depth, and the number of simulations. We group these hyperparameters into three levels, called Small, Medium, and Large, based on their token usage. As an extra baseline, we evaluate Best-of-N by generating candidates with the SFT Llama2-7B at temperature 1.0 and scoring them with the value network v_ϕ. Results are reported in Table 1.

On GSM8K and Game24, ReSCALE exhibits a steady increase in accuracy as the compute budget grows. In contrast, AlphaZero's accuracy plateaus and even degrades as the budget increases from Medium to Large on GSM8K and at Large budget on Game24.

Takeaway: We analyzed MCTS trees for examples where AlphaZero MCTS produced incorrect answers and found that it overexploits a small subset of actions, leading to a large imbalance in visit counts between explored and under-explored nodes. This issue worsens with larger budgets, as root selection favors the most visited node and ignores potentially better alternatives. ReSCALE mitigates this by using Sequential Halving to preserve exploration diversity at each step.

Ablation Studies We perform ablations to isolate the effects of Gumbel noise and Sequential Halving. To remove Gumbel noise, we set it to zero. To remove Sequential Halving, we replace it at the root node with the PUCT rule from AlphaZero MCTS while still adding Gumbel noise to the root prior.

Experiments are conducted on GSM8K using three seeds, with 50 simulations, a maximum width of 24, and a maximum depth of 16. Our method achieves $60.1 \pm 0.9\%$ accuracy. Removing Sequential Halving reduces accuracy by 4.7%, while removing Gumbel noise reduces it by 1.7%. These results highlight the importance of both Gumbel noise and Sequential Halving for the performance of ReSCALE.

4 Conclusion

We presented ReSCALE, which replaces PUCT selection and Dirichlet noise with Sequential Halving and Gumbel sampling for LLM reasoning. On GSM8K and Game24, it restores monotonic scaling, reaching 58.4% and 85.3%, respectively, where the baseline degrades to 53.6% and 82.9%. Ablations confirm Sequential Halving as the primary contributor. These results show that root action-selection design is critical for tree search in LLM decoding, and that fixed-budget best-arm identification techniques can address scaling pathologies.

References

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

Danihelka, I.; Guez, A.; Schrittwieser, J.; and Silver, D. 2022. Policy improvement by planning with Gumbel. In *International Conference on Learning Representations*.

Fawzi, A.; Balog, M.; Huang, A.; Hubert, T.; Romera-Paredes, B.; Barekatain, M.; Novikov, A.; R. Ruiz, F. J.; Schrittwieser, J.; Swirszcz, G.; et al. 2022. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930): 47–53.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Wang, P.; Zhu, Q.; Xu, R.; Zhang, R.; Ma, S.; Bi, X.; et al. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081): 633–638.

Hao, S.; Gu, Y.; Ma, H.; Hong, J.; Wang, Z.; Wang, D.; and Hu, Z. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 8154–8173.

Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L. P.; and Murthy, A. B. 2024. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Forty-first International Conference on Machine Learning*.

Karnin, Z.; Koren, T.; and Somekh, O. 2013. Almost optimal exploration in multi-armed bandits. In *International conference on machine learning*, 1238–1246. PMLR.

Kocsis, L.; and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, 282–293. Springer.

Nguyen, K. P. N.; and Ramanujan, R. 2024. Lookahead Pathology in Monte-Carlo Tree Search. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 414–422.

OpenAI. 2024. OpenAI o1 System Card. arXiv:2412.16720.

Pallagani, V.; Muppasani, B. C.; Roy, K.; Fabiano, F.; Loreggia, A.; Murugesan, K.; Srivastava, B.; Rossi, F.; Horesh, L.; and Sheth, A. 2024. On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS). In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Puterman, M. L. 1994. *Markov decision processes*. Wiley Series in Probability & Mathematical Statistics: Applied Probability & Statistics. Nashville, TN: John Wiley & Sons.

Rosin, C. D. 2011. Multi-armed bandits with episode context. *Ann. Math. Artif. Intell.*, 61(3): 203–230.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T.; Simonyan, K.; and Hassabis, D. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Snell, C. V.; Lee, J.; Xu, K.; and Kumar, A. 2025. Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning. In *The Thirteenth International Conference on Learning Representations*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book. ISBN 0262039249.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

Valmeekam, K.; Stechly, K.; and Kambhampati, S. 2024. LLMs Still Can't Plan; Can LRMs? A Preliminary Evaluation of OpenAI's o1 on PlanBench. In *NeurIPS 2024 Workshop on Open-World Agents*.

Wan, Y.; Wang, W.; Yang, Y.; Yuan, Y.; Huang, J.-t.; He, P.; Jiao, W.; and Lyu, M. 2024a. LogicAsker: Evaluating and improving the logical reasoning ability of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2124–2155.

Wan, Z.; Feng, X.; Wen, M.; Mcaleer, S. M.; Wen, Y.; Zhang, W.; and Wang, J. 2024b. AlphaZero-Like Tree-Search can Guide Large Language Model Decoding and Training. In *International Conference on Machine Learning*, 49890–49920. PMLR.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q. V.; Chi, E. H.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36: 11809–11822.