From Long to Short: LLMs Excel at Trimming Own Reasoning Chains

Wei Han[*]
Independent Researcher

Geng Zhan
University of Sydney

Sicheng Yu
Singapore Management University

Chenyu Wang
University of Sydney

Bryan Hooi
National University of Singapore

Abstract

O1/R1-style large reasoning models (LRMs) demonstrate strong performance across a wide range of complex reasoning tasks, particularly by leveraging test-time scaling to generate extended reasoning paths. However, these models often suffer from *overthinking*. To address this issue, we conduct a systematic investigation into the reasoning efficiency of a broad set of LRMs and reveal a common dilemma. Motivated by the key findings, we propose a purely test-time computation method, EDIT(**E**fficient **D**ynamic **I**nference **T**rimming), that guides LRMs to identify the shortest correct reasoning paths at test time. EDIT employs constraint-guided generation while jointly tracking length and answer distributions under varying constraints, allowing it to select responses that strike an optimal balance between conciseness and correctness. Extensive experiments across diverse models and datasets show that EDIT substantially enhances reasoning efficiency, producing compact yet informative outputs that improve readability and user experience.

1 Introduction

The rapid emergence of large language models (LLMs) has revolutionized the development of artificial intelligence [39]. Recently, researchers have begun to push beyond traditional instruction-following and few-shot chain-of-thought. They explore more sophisticated application scenarios where LLMs must trigger deliberate thinking and reasoning to produce the correct answer [7, 35, 43].

However, the enhanced reasoning abilities of LRMs always come at the cost of increased output redundancy [3], leading to substantial computational overhead that grows quadratically with the output length. To alleviate this issue, we propose a test-time scaling method—EDIT (**E**fficient **D**ynamic **I**nference **T**rimming), which leverages the variation trends of solution lengths and answer confidence under parametric constraints. Building on this statistical discovery, we design a search algorithm that automatically identifies the critical point at which LRMs achieve *Pareto optimality* between reasoning cost and accuracy.

2 Related Work

Large Reasoning Models The first trial to elicit LLM's reasoning ability was on mathematical problems [14], where outcome- or process-based verifiers are trained to supervise LLM learning [20]. More recently, researchers believe that LLMs can simulate human's System 1 (fast, instinct) and System 2 (slow, delibrate) [41] to efficiently handle logical tasks. Reward-guided decoding [16,

[*]Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: NeurIPS 2025 ER Workshop.

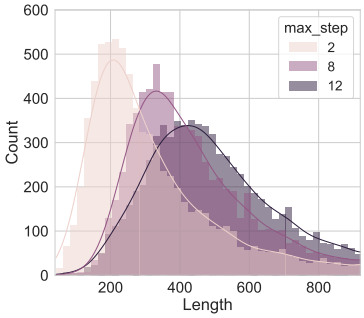
19, 38] and Monte Carlo tree search (MCTS) [12, 45] improve these base methods by introducing external or future feedback. CoT [36] and several subsequent enhancement work [32, 40, 2] managed to elicit the reasoning paths directed to the correct answer. The release of OpenAI-o1 [15] heightened the engagement from the research community in test-time scaling. DeepSeek-R1 [10] demonstrates the great role of reinforcement learning in LLM's grasping of reasoning capabilities [21], and there are a line of follow-up works to develop skilled reasoning models [27, 22]. Regards foundation LRM training, most efforts are put at the RL algorithm [42] and self-evolving pipelines [44, 26].

Underthinking and Overthinking Issue in LRMs. While O1/R1-like LRMs exhibit stunning capabilities in solving complex problems, they expose undesired weakness termed *overthinking* when encountering simple questions like "*what is the result of 2+3?*" [3, 29, 4]. Meanwhile, although LRMs are encouraged to generate complete reasoning paths based on the SFT versions, *underthinking* [34] is ubiquitous among LRMs, which could stagnate at shallow thinking. To alleviate this issue, the model should learn to avoid both *underthinking* and *overthinking*. Many solutions were proposed: direct tuning on pruned generation [37, 11, 23], latent-space reasoning [8, 13] and input-aware test-time searching [6, 33].

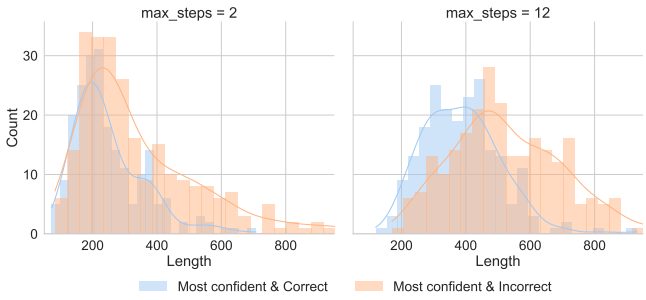
3 Methods

3.1 Statistical Findings from Constrained Reasoning

Since on a generic application scenario with deep thinking enabled, no explicit limitations on generation lengths are applied—LRMs are granted the full freedom to extend their reasoning chains. To dive deeper, we curate a simple parameterized prompt template as illustrated in Section A.2. We benchmark the responses by sampling multiple responses per question on the MATH500 dataset.



(a) Length distribution under three constraints parameterized by the maximum steps that can be taken to solve each problem.



(b) Length distributions of "most confident and correct" and "most confident and wrong" answers under two constraints. The distribution is separable when constraint is loose (left). Serious distribution overlap exhibits when applying stronger constraint.

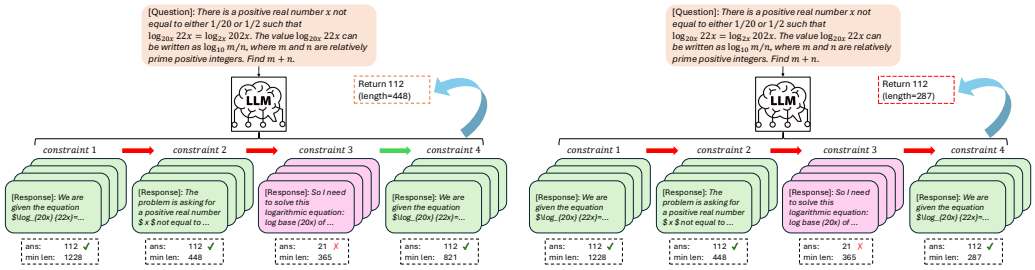
Figure 1: Histogram of length distributions on MATH500 dataset with LLaMA-3.1-8B: the generation length distributions for (a) all generated solutions, 16 samples per question (b) most confident answers, with binary justifications as correct/wrong.

The length distribution of outputs using this template in MATH500 test set and LLaMA-3.1-8B model is shown in Figure 1a. We also highlight the the most confident answer for each problem as well as the corresponding correctness in Figure 1b. Generally, the length distribution is long-tailed, especially when the constraint is stringent. Besides, there are two observations from the plots:

- In Figure 1a, LLM is sensitive to the imposed constraints and can adapt its output distribution accordingly—the overall length distribution shifts right (i.e., longer) as the constraint loosens (i.e., more steps allowed).
- In Figure 1b, as the constraint becomes more straingent (right to left), incorrect answers (orange bars) is prone to dominate the shorter part of answers, i.e. the model tends to prioritize the length and ignores correctness.

3.2 Dual Goal Search

Overview In EDIT, beyond the ultimate goal of solving the given problem, an additional length constraint is appended to the prompt templates. Take Figure 2a as an example. When the constraint is loose, LRMs work normally and enhance their reasoning ability by creating a long reasoning path. As we strengthen the constraint for length, LRMs successfully follow our guidance to shorten their output length while keeping the final answer correct (step 1 to step 2). However, when we further enforce constraints for shorter outputs, the LRMs cannot simultaneously satisfy both objectives forever. They tend to prioritize the goal it deems more important—typically the length during generation, and output the wrong answer (from step 2 to step 3). To address this, EDIT monitors the dynamics of two key statistics—answer confidence and length distribution—and adjusts the constraint strength accordingly, either relaxing or tightening it as needed.



(a) Cross-constraint checking fails ($\beta_0 = 0$). (b) Cross-constraint checking succeeds ($\beta_0 = 1$).

Figure 2: Two running examples of the EDIT framework, with and without patience respectively. Dark red and green arrows between search steps represent the constraint is tighter or relaxed. In figure (a), the consistency checking fails due to running-out of patience. The next step continues with a looser constraint and a sub-optimal response is returned. A higher value of initial patience in figure (b) and a better response is returned.

Algorithm The core algorithm in EDIT, *Dual-Goal Search* is described in Algorithm 1 in the appendix. The search process begins with the loosest constraint and full patience. At the start of every subsequent iteration, the algorithm checks the consistency of the predicted answers and length distributions by comparing the answer and length metrics. If the consistency checking passes, the algorithm proceeds with a stronger constraint. Otherwise, if the patience does not run out, the algorithm continues this procedure with a tighter constraint and reduced patience. If patience is exhausted, the algorithm compares the current answer with the most confident answer recorded in previous iterations from H.

The hyperparameter patience is introduced to prevent the misdirection of search and the return of suboptimal solutions. The indicator for length distribution is selected as follows:

$$lstat = \texttt{Ans\_Stat}(\mathcal{C}, \hat{a}) = \frac{\min(\mathcal{L}) + Q_1(\mathcal{L}) + \text{median}(\mathcal{L})}{3} \tag{1}$$

where $\mathcal{L}$ denotes the length of the collection of responses whose predicted answers equal the most confident answer $\hat{a}$.

4 Experiments

4.1 Inference Setup

Datasets We focus on mathematical reasoning tasks. Three publicly available datasets are evaluated in this stage: GSM8K [5], MATH500 [20] and AIMO [25], the super collection of AIME 2022-2024. Specifications of these datasets can be found in Section A.1.

Models We assess EDIT on seven LLMs, covering various parameter sizes and training patterns. The targeted models include: non-reasoning instruction-following models (LLaMA-3.1-8B [9]),

reinforcement learning enhanced reasoning models (QwQ-32B [31], Qwen3-8B/32B [30] and Phi-4-Reasoning [1]), LRM-distilled models [10] (R1-distilled-Qwen1.5B, R1-distilled LLaMA3-8B and Phi-4-reasoning-14B [1]). More specifications about targeted models can be found in Section A.5.

Generation Configuration We keep the same set of sampling parameters, such as temperatures and top_p values, across all baselines and our implementations to ensure a fair comparison. To ensure a fair comparison, the number of samples N is kept the same in all experiments. As a result, for methods using iterative algorithms, the number of samples per iteration is upper bounded.

For details of baselines and metrics, please refer to Section A.4.

Table 1: Result on three math problem datasets. The **best accuracy** among all baseline methods are underlined. Accuracy and relative length variation with the best baselines are highlighted in the brackets.

Dataset	Model	DP		BoN		ST		EDIT	
		Acc↑	Length↓	Acc↑	Length↓	Acc↑	Length↓	Acc↑	Length↓
GSM8K	Qwen3-32B	88.2	725	94.2	181	94.0	195	94.4 (↑ 0.2)	83 (↓54.1%)
	Llama-3.1-8B	67.2	302	72.3	120	68.4	84	71.4 (↓0.9)	87 (↓27.5%)
	Qwen3-8B	88.2	380	93.4	202	93.3	163	93.4	103 (↓36.8%)
	QwQ-32B	88.0	706	93.0	378	92.6	765	93.3 (↑ 0.3)	223 (↓41.0%)
	Phi-4-Reasoning	82.6	524	91.1	419	91.5	370	91.8 (↑ 0.3)	286 (↓22.7%)
	RD-Qwen-1.5B	74.1	476	86.4	290	85.9	236	85.7 (↓0.7)	154 (↓46.9%)
	RD-Llama-3.1-8B	65.2	420	89.1	330	88.2	295	88.7 (↓0.4)	262 (↓20.6%)
MATH	Qwen3-32B	52.0	480	71.6	328	68.8	368	70.8 (↓0.8)	297 (↓9.4%)
	Llama-3.1-8B	30.2	110	52.6	131	51.0	98	52.4 (↓0.2)	108 (↓17.6%)
	Qwen3-8B	59.8	480	73.4	353	74.2	291	73.2 (↓1.0)	207 (↓28.9%)
	QwQ-32B	43.8	612	64.4	394	61.0	327	64.8 (↑ 0.4)	298 (↓24.4%)
	Phi-4-Reasoning	73.5	1033	81.2	796	78.6	535	80.2 (↓1.0)	335 (↓57.9%)
	RD-Qwen-1.5B	48.4	520	60.4	396	62.6	284	62.4 (↓0.2)	249 (↓12.3%)
	RD-Llama-3.1-8B	51.4	406	62.4	411	63.0	374	65.0 (↑ 2.0)	337 (↓9.9%)
AIMO	QwQ-32B	72.2	7352	84.4	5480	86.7	4980	86.7	3791 (↓30.8%)
	Llama-3.1-8B	5.6	13	7.8	47	7.8	32	11.1 (↑ 3.3)	40 (↑25.0%)
	Qwen3-8B	58.9	3335	76.7	3187	75.6	3045	74.4 (↓2.3)	2284 (↓28.3%)
	Qwen3-32B	77.8	6189	82.2	5533	83.3	4980	83.3	4140 (↓16.9%)
	Phi-4-reasoning	68.9	3565	82.2	3870	77.8	2921	83.3 (↑ 1.1)	2533 (↓34.5%)
	RD-Qwen-1.5B	22.9	2843	42.2	1728	36.7	1155	41.1 (↓1.1)	1376 (↓20.4%)
	RD-Llama-3.1-8B	51.1	2578	60.0	2348	60.0	2149	60.0	1770 (↓17.6%)

4.2 Main Results

The results averaged over five runs are shown in Table 1. The key findings from these results are as follows:

- RL-enhanced models (e.g., the Qwen-3 8B/32B, QwQ-32B and Phi-4-Reasoning) consistently show better capabilities than models in other categories. Models trained with reasoning data distillation (e.g., R1-Distill-Llama-3.1-8B and R1-Distill-Qwen1.5B) perform worse than RL-enhanced models but still outperform the SFT-only baseline (Llama-3.1-8B).

- While showing the highest accuracy in every test case, EDIT shows superior reasoning efficiency—it manages to preserve accuracy to the maximum extent and cut off the reasoning path. In most cases, EDIT nearly achieves or surpasses the best baseline with a concise reasoning process. Specifically, the largest performance gap is below relatively 2% with 20% fewer tokens produced in the reasoning paths.

- EDIT struggles to make decisions on non-reasoning models. The only outlier in length is on AIMO using Llama-3.1-8B, where EDIT generates 25% more tokens to pursue a significant accuracy boost. In contrast, on MATH and GSM8K, it works similarly to other models that strive to reach Pareto optimality.

5 Conclusion

In this paper, we propose EDIT (**E**fficient **D**ynamic **I**nference **T**rimming), a test-time scaling method aimed at generating concise and correct reasoning paths. Triggered by the phenomenon distributional

shifts discovered in our preliminary experiments, EDIT designs a dual-goal search algorithm to achieve an optimal tradeoff between simplicity and correctness. Experimental results across a broad range of models and datasets demonstrate the advantage and robustness compared to other strong baselines.

References

[1] Marah Abdin, Sahaj Agarwal, Ahmed Awadallah, Vidhisha Balachandran, Harkirat Behl, Lingjiao Chen, Gustavo de Rosa, Suriya Gunasekar, Mojan Javaheripi, Neel Joshi, et al. Phi-4-reasoning technical report. *arXiv preprint arXiv:2504.21318*, 2025.

[2] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 17682–17690, 2024.

[3] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

[4] Cheng-Han Chiang and Hung-Yi Lee. Over-reasoning and redundant calculation of large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–169, 2024.

[5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[6] Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of lm computation. *arXiv preprint arXiv:2410.04707*, 2024.

[7] Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, and Yongfeng Zhang. Nphardeval: Dynamic benchmark on reasoning ability of large language models via complexity classes. *arXiv preprint arXiv:2312.14890*, 2023.

[8] Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

[9] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[10] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[11] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

[12] Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, 2023.

[13] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

[14] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[15] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[16] Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. Args: Alignment as reward-guided search. In *The Twelfth International Conference on Learning Representations*, 2024.

[17] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[18] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[19] Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

[20] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[21] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

[22] Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-rft: Visual reinforcement fine-tuning. *arXiv preprint arXiv:2503.01785*, 2025.

[23] Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*, 2025.

[24] MAA. American invitational mathematics examination- aime, 2025.

[25] Bhrij Patel, Souradip Chakraborty, Wesley A Suttle, Mengdi Wang, Amrit Singh Bedi, and Dinesh Manocha. Aime: Ai system optimization via multiple llm evaluators. *arXiv preprint arXiv:2410.03131*, 2024.

[26] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[27] Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

[28] Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[29] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

[30] Qwen Team. Qwen3: Think deeper, act faster.

[31] Qwen Team. Qwq-32b: Embracing the power of reinforcement learning.

[32] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[33] Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. Sampling-efficient test-time scaling: Self-estimating the best-of-n sampling in early decoding. *arXiv preprint arXiv:2503.01422*, 2025.

[34] Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*, 2025.

[35] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

[36] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[37] Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

[38] Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650, 2023.

[39] Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, et al. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*, 2025.

[40] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[41] Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.

[42] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[43] Weizhe Yuan, Jane Yu, Song Jiang, Karthik Padthe, Yang Li, Dong Wang, Ilia Kulikov, Kyunghyun Cho, Yuandong Tian, Jason E Weston, et al. Naturalreasoning: Reasoning in the wild with 2.8 m challenging questions. *arXiv preprint arXiv:2502.13124*, 2025.

[44] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

[45] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

A Experiment Details

A.1 Dataset Statistics

We use three datasets for evaluation, all of which are in the mathematical domain:

- GSM8K [5]: The Grade School Math dataset contains 8.5K high-quality, linguistically diverse grade school math word problems, which are divided into the training set of 7,500 samples and the test set of 1,000 samples. Problems in GSM8K can be solved from 2 to 8 steps, and all solutions primarily require performing a sequence of elementary calculations using basic arithmetic operations to reach the final answer. We use the test set for evaluation, which has 1,319 examples in total.

- MATH500 [20]: This dataset was released by OpenAI and is commonly used for verifier-based training. We use a subset of the whole original dataset for evaluation, which comprises 500 examples.

- AIMO [24]: The AIMO (American Invitational Mathematics Olympics) dataset consists of 90 problems from AIME 2022-2024 competitions. This is the most challenging dataset among all the datasets.

A.2 Prompt Template

The prompt template for constrained generation is below.

Table 2: The prompt template to apply constraints and elicit principle-guided responses. Variables with curly braces surrounded are placeholders to be filled. “num_step" is the upper limit of reasoning steps we assign to LRMs.

[System] <*The system prompt is omitted*>.

[User] Solve the given math problem step by step. You must output the final answer in a box "\\{{final_answer}}". You are limited to at most {num_step} reasoning steps. Stop generating immediately and output the answer if you reach the maximum {num_step} steps or obtained the final answer early.

Question: {question}

Solution:

A.3 Search Algorithm

A.4 Baselines and Metrics

Baselines We compare our methods with a series of advanced baselines: Direct CoT Prompting (**DP**) [17] forces the model to generate a step-wise thinking path. For Best-of-N (**BoN**) [28], we gather a collection of solutions that obtain the most common answer according to the voting mechanism as in self-consistency [32]. We then select the shortest solution from that collection as the final generation result. In Self-Truncation (**ST**), similar to the implementation in [37], we first prompt LRMs to generate reasoning chains and select the chains that produce the most common answers. Then we continue to prompt LRMs to prune these solution paths. All baselines are running under a fixed budget of $N = 64$ samples (except for direct prompting, where LRM generates solutions using greedy decoding), regardless of how many iterations each method has gone through.

Metrics The basic metrics are accuracy and average length, both of which are averaged over the entire datasets. However, for the length comparison, we find that treating correct and incorrect responses equally causes inconsistency in the intuitive expression of the models’ overall performance.

Algorithm 1 Dual-Goal Search

Input: Model $\mathcal{M}_\theta$, Question q, Maximum Number of Iterations T, Total Sampling Budget N, Constraint Template Ψ with Parameter $\tau = \tau_{max}$ (smaller τ means tighter constraint), patience $\beta = \beta_0$.
Output: Final Answer $\hat{a}$
Initialize Number of Samples in each iteration $n = \frac{N}{T}$, History Record Memory H
for $t = 1$ to T **do**
    Sample candidates with parameterized constraints $\mathcal{C}_t = \{c_{t,1}, \ldots, c_{t,n}\} \sim \mathcal{M}_\theta(q, \Psi(\tau_t))$
    $answer\_conf = \texttt{Cal\_Confidence}(\mathcal{C}_t)$
    $\hat{a}_t = \arg\max(answer\_conf)$
    $lstat_t = \texttt{Ans\_Stat}(\mathcal{C}_t, \hat{a}_t)$ ▷ obtain the statistics of length distribution
    if $\texttt{Check\_Consistency}(H, \hat{a}_t, lstat_t)$ **then**
        $\tau_{max} = \tau_t$, $\tau_t = (\tau_{max} + \tau_{min})/2$
    else if $\beta == 0$ **then** ▷ checking consistency failed, patience spent up
        if $\hat{a}_i == \texttt{most\_confident}(H)$ **then**
            $\tau_{max} = \tau_t$
        else
            $\tau_{min} = \tau_t$
            continue
        end if
        $\beta = \beta_0$
    else ▷ patience does not run out, continue to searching in shorter generations.
        $\tau_t = (\tau_{t-1} + \tau_{min})/2$, $\beta = \beta - 1$
    end if
    $H = H \cup \{(\hat{a}_i, lstat_i)\}$
end for
$\hat{a} = \texttt{most\_confident}(H)$
return $\hat{a}$

Take Llama-3.1-8B on the MATH500 test set as an example, the DP has an average response length of 110 tokens, while EDIT spends more tokens (121) and achieves twice the accuracy (61.7% v.s. 30.2%). The measurement of length becomes meaningless at this time. Hence, a calibration approach is required—one that considers both accuracy and the costs associated with incorrect predictions.

A.5 Model Specifications

There are seven targeted models being tested. All targeted models can be divided into three categories:

- Instruction tuned model (Llama series): This category presents the most typical instruction-following models. They only experience the pretraining and supervised fine-tuning stages. Additional red-teaming and human-preference alignment may be further applied to make its output fulfill the HHH (Helpful, Honest, and Harmless) principle. Although demonstrating the powerful general task solving capability, they show poor performance when encountering complex reasoning problems.

- Reinforcement Learning (RL) enhanced reasoning model (QwQ-32B, Qwen3 series): These models are endowed with the better reasoning capabilities through an additional RL stage on large-scale reasoning data before traditional SFT and RL-based value-alignment stages.

- Distilled model (R1-distill-llama3-8B, R1-distill-Qwen-1.5B, Phi-4-reasoning-14B): These models are released along with DeepSeek-R1 [10], which are fine-tuned on the automatically generated prompt-response pairs as illustrated in their papers.

A.6 Evaluation Configurations

We present all pivotal inference hyper-parameters in Table 3. The inference scripts are implemented using the vLLM [18] library on four NVIDIA A100 80G GPUs with tensor-level parallelism. We keep the same set of configurations for all tested models on the same dataset.

We put the core code at `https://anonymous.4open.science/r/EDIT-F90B`.

Dataset	GSM8K	MATH	AIMO
temperature		0.7	
top_p		0.9	
num samples		64	
max token	2048	2048	16384

Table 3: Inference settings across three datasets.

B More Statistical Results

B.1 Distribution of Length on Correct and Incorrect Answers

We further plot the length distribution of correct/incorrect answers on GSM8K test set in Figure 3 and Figure 4. It can be found that for all models, compared to other two baselines, the distribution of EDIT shifts to the left—indicating there the huge reduction in lengths. Besides, the area ratio between the blue and orange bars also increases sharply, which means EDIT gets more correct answers.

B.2 Generation Examples

We show some typical examples of responses from these three methods, as shown in Tables 4 and 5. In the simple GSM8K example, EDIT reaches the correct answer using much less tokens than the other methods. In the hard MATH500 example, where the model has to understand the code in the question, BoN fails to obtain the correct answer due to insufficient uncertainty regarding the correct answer. In contrast, EDIT successfully yields the correct answer.

Table 4: An GSM8K problem solved by EDIT and several baselines and R1-Distill-Qwen1.5B. Green titled boxes represent that the solution is both correct and the shortest. We remove some special tokens for better readability. Blue titled boxes mean that only answer is correct. Red titled boxes fail to produce the correct answer.

GSM8K Question

There are some oranges in a basket. Ana spends 3 minutes peeling an orange and Jane spends 4 minutes doing the same. If Ana and Jane start picking oranges from this basket to peel at the same time, how many more oranges will Ana have peeled than Jane after an hour?
Answer: 5

EDIT (142 Tokens)

Step 1: Determine the peeling rates of Ana and Jane.
- Ana peels 1 orange per 3 minutes.
- Jane peels 1 orange per 4 minutes.

Step 2: Calculate the number of oranges each can peel in an hour (60 minutes).
- Ana: $\frac{60}{3} = 20$ oranges.
- Jane: $\frac{60}{4} = 15$ oranges.

Step 3: Find the difference in oranges peeled.
- Ana peels $20 - 15 = 5$ more oranges than Jane.

Answer: $\boxed{5}$

BoN (226 Tokens)

First, Let's define, Ana's rate = 1 orange per 3 minutes, Jane's rate = 1 orange per 4 minutes. In 1 hour (60 minutes), Ana peels 60 ÷ 3 = 20 oranges. Jane peels 60 ÷ 4 = 15 oranges. Therefore, Ana peeled 5 more oranges than Jane.

Step-by-Step Explanation:

1. **Determine Individual Rates:**
- Ana peels 1 orange in 3 minutes.
- Jane peels 1 orange in 4 minutes.

2. **Calculate Oranges Plucked in an Hour:**
- One hour = 60 minutes.
- **Ana:** $60 \div 3 = 20$ oranges.
- **Jane:** $60 \div 4 = 15$ oranges.

3. **Find the Difference:** - $20 - 15 = 5$ oranges.
Final Answer: $\boxed{5}$

ST (278 Tokens)

First, Let's define, Ana's rate = 1 orange per 3 minutes, Jane's rate = 1 orange per 4 minutes. In 1 hour (60 minutes), Ana peels 60 ÷ 3 = 20 oranges. Jane peels 60 ÷ 4 = 15 oranges. Therefore, Ana peeled 5 more oranges than Jane.

Step-by-Step Explanation:

1. **Determine Individual Rates:**
- Ana peels 1 orange in 3 minutes.
- Jane peels 1 orange in 4 minutes.

2. **Calculate Oranges Plucked in an Hour:**
- One hour = 60 minutes.
- **Ana:** $60 \div 3 = 20$ oranges.
- **Jane:** $60 \div 4 = 15$ oranges.

3. **Find the Difference:**
- $20 - 15 = 5$ oranges.

Final Answer: $\boxed{5}$

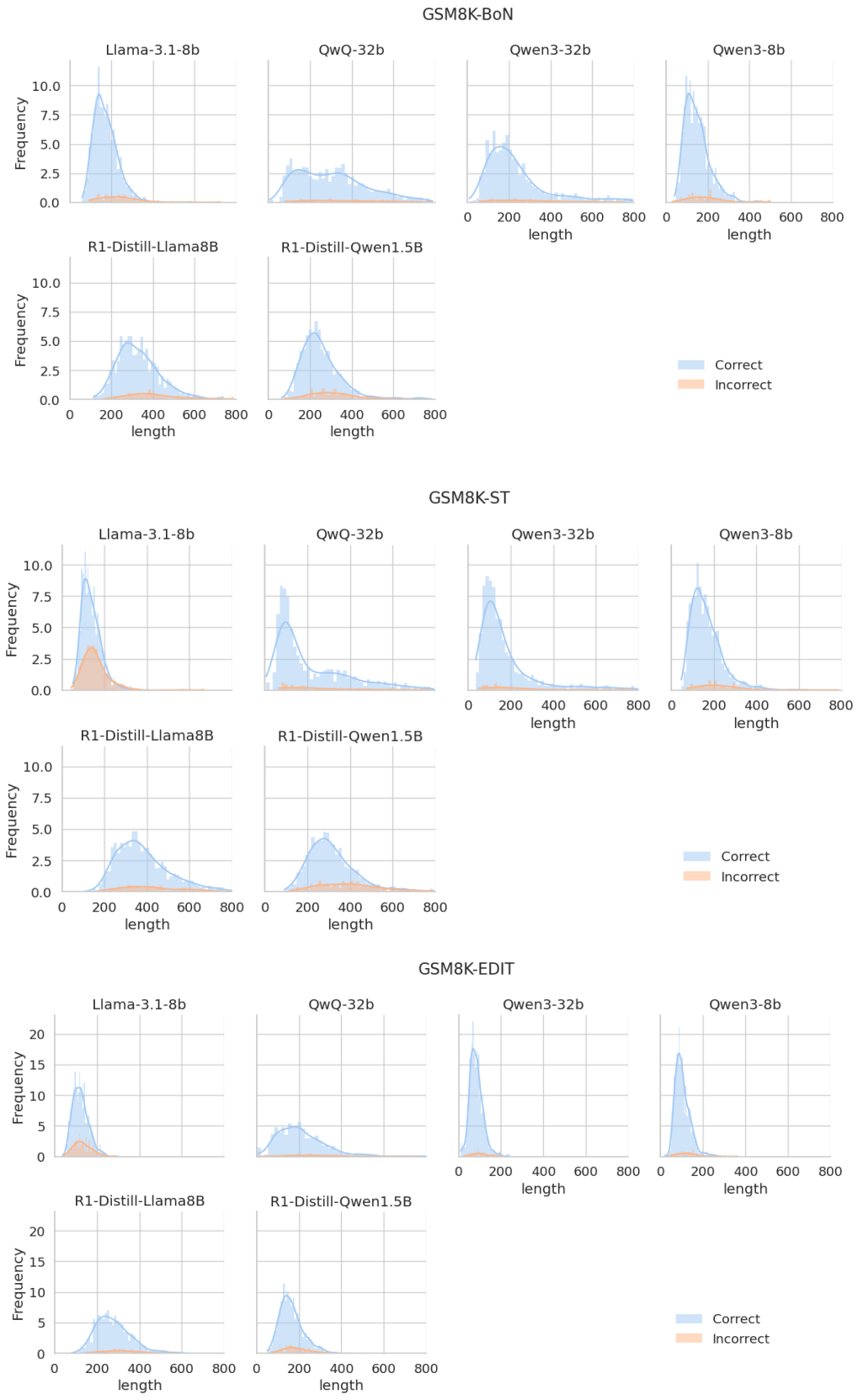


Figure 3: Length distribution for correct and incorrect responses from three methods on GSM8K.

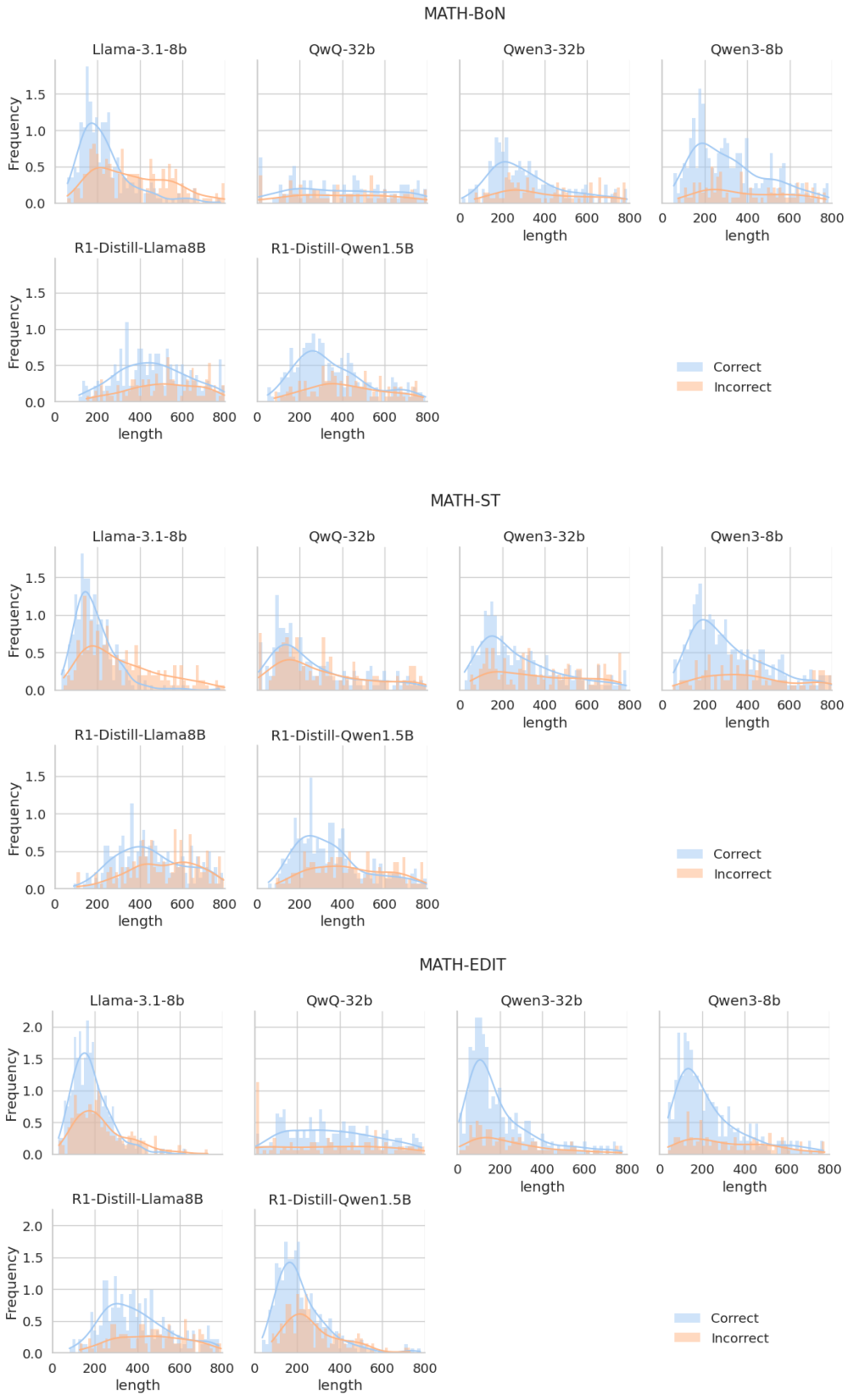


Figure 4: Length distribution for correct and incorrect responses from three methods on MATH.

Table 5: A MATH500 problem solved by EDIT and several baselines and R1-distill-Qwen-1.5B.

MATH500 Question

Let $z = 2 + \sqrt{2} - (3 + 3\sqrt{2})i$, and let $c = 2 - 3i$. Let w be the result when z is rotated around c by $\frac{\pi}{4}$ counter-clockwise.
[asy]
unitsize(0.6 cm);
pair C, W, Z;

$Z = (2 + \sqrt{(2)}, -3 - 3 * \sqrt{(2)}); C = (2, -3); W = \text{rotate}(45, C) * (Z);$
draw(Z–C–W);
dot("c", C, N);
dot("w", W, SE);
dot("z", Z, S);
label("$\frac{\pi}{4}$", C + (0.6,-1));
[/asy]
Find w.
Answer: $6 - 5i$

EDIT (667 Tokens)

Step 1: Identify the given points and transformations.
We are given: - $z = 2 + \sqrt{2} - (3 + 3\sqrt{2})i$ - $c = 2 - 3i$ - w is the result of rotating z around c by $\frac{\pi}{4}$ counter-clockwise.
Step 2: Translate the complex number z so that c is at the origin.
Translation is done by subtracting c from z:

$$z' = z - c = (2 + \sqrt{2} - (3 + 3\sqrt{2})i) - (2 - 3i) = (\sqrt{2}) - (3\sqrt{2})i$$

Step 3: Rotate the translated complex number z' by $\frac{\pi}{4}$.
The rotation formula is:

$$w' = z' \cdot \left(\cos\left(\frac{\pi}{4}\right) + i \sin\left(\frac{\pi}{4}\right)\right) = z' \cdot \left(\frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2}\right)$$

Multiply z' by this rotation factor:

$$w' = (\sqrt{2} - 3\sqrt{2}i) \cdot \left(\frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2}\right)$$

Step 4: Perform the multiplication.
Multiply the terms:

$$w' = \sqrt{2} \cdot \frac{\sqrt{2}}{2} + \sqrt{2} \cdot i\frac{\sqrt{2}}{2} - 3\sqrt{2}i \cdot \frac{\sqrt{2}}{2} - 3\sqrt{2}i \cdot i\frac{\sqrt{2}}{2}$$

Simplify each term: - $\sqrt{2} \cdot \frac{\sqrt{2}}{2} = 1$ - $\sqrt{2} \cdot i\frac{\sqrt{2}}{2} = i1$ - $-3\sqrt{2}i \cdot \frac{\sqrt{2}}{2} = -3i1$ - $-3\sqrt{2}i \cdot i\frac{\sqrt{2}}{2} = 3$ (since $i^2 = -1$)

$$w' = 1 + i - 3i + 3 = 4 - 2i$$

Step 5: Translate back by adding c.
Add c to w':

$$w = w' + c = (4 - 2i) + (2 - 3i) = 6 - 5i$$

Answer: $\boxed{6 - 5i}$

BoN (610 Tokens but wrong answer)

Let's find the value of w after rotating z around c by $\frac{\pi}{4}$ counter-clockwise.
Step 1: Identify the given points and complex numbers
- $z = 2 + \sqrt{2} - (3 + 3\sqrt{2})i$ - $c = 2 - 3i$
Step 2: Translate z so that the center c becomes the origin
To rotate around c, we translate z by subtracting c:

$$z' = z - c = (2 + \sqrt{2} - 2) + (-3 - 3\sqrt{2} - (-3))i = \sqrt{2} - 3\sqrt{2}i$$

Step 3: Apply the rotation by $\frac{\pi}{4}$
The rotation formula for a complex number z' by θ radians is:

$$w = z' \cdot e^{i\theta} = z' \cdot (\cos\theta + i\sin\theta)$$

For $\theta = \frac{\pi}{4}$:

$$e^{i\frac{\pi}{4}} = \cos\frac{\pi}{4} + i\sin\frac{\pi}{4} = \frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2}$$

Multiply z' by this:

$$w = (\sqrt{2} - 3\sqrt{2}i)\left(\frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2}\right)$$

Step 4: Perform the multiplication
Multiply each term:

$$w = \sqrt{2}\cdot\frac{\sqrt{2}}{2} + \sqrt{2}\cdot i\frac{\sqrt{2}}{2} - 3\sqrt{2}i\cdot\frac{\sqrt{2}}{2} - 3\sqrt{2}i\cdot i\frac{\sqrt{2}}{2}$$

Simplify each term:

$$w = \frac{2}{2} + i\frac{2}{2} - i\frac{6}{2} - i^2\frac{6}{2}$$

$$w = 1 + i - 3i - (-1)\cdot 3$$

$$w = 1 - 2i + 3 = 4 - 2i$$

Final Answer: $\boxed{4 - 2i}$

ST (705 Tokens)

Step 1: Identify the given points and transformations.
We are given: - $z = 2 + \sqrt{2} - (3 + 3\sqrt{2})i$ - $c = 2 - 3i$ - w is the result of rotating z around c by $\frac{\pi}{4}$ counter-clockwise.
Step 2: Translate the complex number z so that c is at the origin.
Translation is done by subtracting c from z:

$$z' = z - c = (2 + \sqrt{2} - (3 + 3\sqrt{2})i) - (2 - 3i) = (\sqrt{2}) - (3\sqrt{2})i$$

Step 3: Rotate the translated complex number z' by $\frac{\pi}{4}$.
The rotation formula is:

$$w' = z' \cdot \left(\cos\left(\frac{\pi}{4}\right) + i\sin\left(\frac{\pi}{4}\right)\right) = z' \cdot \left(\frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2}\right)$$

Multiply z' by this rotation factor:

$$w' = (\sqrt{2} - 3\sqrt{2}i) \cdot \left(\frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2}\right)$$

Step 4: Perform the multiplication.
Multiply the terms:

$$w' = \sqrt{2} \cdot \frac{\sqrt{2}}{2} + \sqrt{2} \cdot i\frac{\sqrt{2}}{2} - 3\sqrt{2}i \cdot \frac{\sqrt{2}}{2} - 3\sqrt{2}i \cdot i\frac{\sqrt{2}}{2}$$

Simplify each term: - $\sqrt{2} \cdot \frac{\sqrt{2}}{2} = 1$ - $\sqrt{2} \cdot i\frac{\sqrt{2}}{2} = i1$ - $-3\sqrt{2}i \cdot \frac{\sqrt{2}}{2} = -3i1$ - $-3\sqrt{2}i \cdot i\frac{\sqrt{2}}{2} = 3$ (since $i^2 = -1$)

$$w' = 1 + i - 3i + 3 = 4 - 2i$$

Step 5: Translate back by adding c.
Add c to w':

$$w = w' + c = (4 - 2i) + (2 - 3i) = 6 - 5i$$

Final Answer: $\boxed{6 - 5i}$