PREMISE: Scalable and Strategic Prompt Optimization for Efficient Mathematical Reasoning in Large Reasoning Models

Ye Yu
Siebel School of Computing and Data Science
University of Illinois Urbana-Champaign
Champaign, IL 61820
yeyu4@illinois.edu

Haibo Jin
School of Information Sciences
University of Illinois Urbana-Champaign
Champaign, IL 61820
haibo@illinois.edu

Yaoning Yu
School of Information Sciences
University of Illinois Urbana-Champaign
Champaign, IL 61820
yyn030600@gmail.com

Haohan Wang*
School of Information Sciences
University of Illinois Urbana-Champaign
Champaign, IL 61820
haohanw@illinois.edu

Abstract

Large Reasoning Models (LRMs) like Claude 3.7 Sonnet and OpenAI o1 achieve strong performance on mathematical tasks via long Chain-of-Thought (CoT), but often generate unnecessarily verbose reasoning traces. This inflates token usage and cost, limiting deployment in latency-sensitive or API-constrained settings. To address this issue, we present **PREMISE** (*PRompt-based Efficient Mathematical Inference with Strategic Evaluation*), an optimization framework designed specifically for black-box commercial LRMs. PREMISE reduces reasoning overhead without modifying model weights or requiring multiple queries. It combines trace-level diagnostics with gradient-based prompt optimization to minimize redundant computation while preserving answer accuracy. Across GSM8K, SVAMP, and Math500, PREMISE matches or exceeds baseline accuracy, while reducing reasoning tokens by up to **87.5%** and cutting dollar cost by **69–82%**.

1 Introduction

Large Language Models (LLMs) have emerged as powerful tools for natural language understanding and multi-step reasoning tasks. The recent development of reasoning-specialized LLMs—commonly referred to as Large Reasoning Models (LRMs) [1]—has pushed the frontier of system-2 reasoning, particularly in mathematics [2, 3] and programming [4, 5]. Models such as OpenAI's o1 [6] and DeepSeek-R1 [7] build on base pretrained models like LLaMA [8, 9] and use multi-stage supervised fine-tuning and reinforcement learning to encourage structured reasoning behaviors.

In many real-world settings—such as interactive assistants, robotic planning systems, or real-time retrieval applications—such inefficiencies are unacceptable. Token-based billing, latency constraints, and hardware bottlenecks limit the feasibility of long reasoning chains in commercial deployments. Thus, recent work has begun to explore efficient reasoning strategies, including length-constrained prompting [10, 11, 12], self-training with compressed CoT data [13, 14], latent-space reasoning [15, 16, 17], and dynamic test-time routing [18, 19, 20].

*Corresponding Author

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

However, most of these methods fall into two broad categories: (1) model-level adaptations that require access to internal weights (e.g., fine-tuning, RL, latent representation training), (2) prompt-based methods were either based on simple heuristics or imposed static length constraints without accounting for the internal structure of the reasoning process. The former are inapplicable to closed-source APIs, while the latter lack rigorous optimization and diagnostic tools for reasoning control.

In this paper, we present **PREMISE** (**PR**ompt-based **E**fficient **M**athematical **I**nference with **S**trategic **E**valuation), a optimization framework that can produce prompt solution designed for efficient reasoning in black-box LRMs. PREMISE introduces reasoning text level metrics that diagnose overthinking and underthinking in a model's output, then leverages these metrics within a reusable prompt structure that encourages strategic reasoning. The method explicitly guides models to avoid redundant branches and commit early to high-value solution paths. To further improve token efficiency, PREMISE incorporates multi-objective optimization via natural language gradients [21, 22], balancing correctness against reasoning length—all without modifying model weights.

We evaluate PREMISE across GSM8K, SVAMP, and Math500, showing that it matches or exceeds CoT [23] and SoT [24] prompting in accuracy while reducing reasoning token usage by up to 85%. PREMISE operates entirely through the prompt interface, making it suitable for any commercial LLM. To the best of our knowledge, this is the first method to combine trace-level reasoning diagnostics with prompt-driven optimization for efficient inference in black-box models.

Our contributions are three-fold:

- We introduce **PREMISE**, a optimization framework that produce prompt solution for efficient reasoning in black-box LLMs. PREMISE works without model fine-tuning or multi-sample decoding, making it applicable to commercial models such as Claude, GPT, and Gemini.
- We define and operationalize two trace-level metrics—*overthinking* and *underthinking*—to identify reasoning inefficiencies during inference. These metrics provide a principled diagnostic foundation for prompt-based reasoning control.
- We demonstrate that PREMISE achieves up to 87.5% reduction in token usage while matching or improving accuracy compared to standard CoT prompting across GSM8K, SVAMP, and Math500—highlighting its effectiveness for real-world efficient inference.

2 Method

Theoretical Framework We formalize reasoning efficiency through trace-level metrics. For a question q with ground-truth answer A, let a reasoning trace be

$$r = (t_1, \dots, t_{L(r)}),$$

which produces an answer $a(r)$. Define the correctness indicator as

$$\mathrm{acc}(r, q) = \begin{cases} 1, & \text{if } a(r) = A, \\ 0, & \text{otherwise.} \end{cases}$$

Among all correct traces, the most efficient is the shortest one.

Based on this, we introduce two inefficiency types:

- **Overthinking**: extra tokens beyond optimum reasoning trace in a correct trace.
- **Underthinking**: an early, irreversible deviation from any correct continuation in an incorrect trace.

These metrics are combined into an efficiency-aware loss, optimized jointly with the standard accuracy loss, yielding a multi-objective formulation. Complete definitions and analysis are in Appendix C.

Optimization Pipeline Overview We adopt the general pipeline of textual optimization for LLM-based systems [21, 22], where prompts are optimized using feedback in the form of natural language gradients, with three key phases:

- **Forward Pass:** Inputs are processed sequentially through the system's computation graph, producing a trajectory of intermediate reasoning states.
- **Language Loss Computation:** An evaluator LLM provides textual feedback on the quality of outputs, serving as a loss that reflects alignment with task objectives.

- **Backward Pass:** Textual gradients, expressed as natural language instructions, are backpropagated to adjust system variables such as prompts, decisions, or tool calls. Unlike numerical optimization, updates are guided entirely by natural language feedback.

PREMISE: Multi-Objective, Trace-Aware Optimization Building on TextGrad and REVOLVE, PREMISE extends textual optimization in two critical ways. First, it introduces *trace-aware variables*:

$$\mathcal{V}_{thinking} = \{\text{value}, \text{trace}, \text{token\_count}, \text{role\_description}\},$$

where *trace* encodes the model's hidden reasoning tokens and *token_count* quantifies computational cost. This representation allows the optimizer to diagnose inefficiencies such as redundant steps (*overthinking*) or premature errors (*underthinking*).

Second, PREMISE employs a *multi-objective loss* that balances accuracy and efficiency. Accuracy loss penalizes incorrect answers, while efficiency loss penalizes both *overthinking* (redundant reasoning tokens) and *underthinking* (early irreversible deviations). Dynamic weighting allows the optimizer to adapt emphasis between the two objectives.

In contrast to prior approaches that optimize for correctness alone, PREMISE explicitly learns the Pareto frontier of accuracy–efficiency trade-offs, enabling deployment in settings where both solution quality and inference cost matter. For implementation and analysis details, see Appendix D.

3 Experiments

Dataset	Model	Method	Acc. (%)	Input	Thinking	Completion	Cost per iteration ($)
GSM8K	Claude-3.7-sonnet	Normal	94	**74**	1,023	230	0.01902
		SoT	**96**	624	487	156	0.01152
		PREMISE	95	650	**218**	**49**	**0.00596**
	OpenAI o1	Normal	96	**68**	**249**	114	0.02280
		SoT	96	535	556	77	0.04601
		PREMISE	**97**	519	1,012	**35**	**0.07061**
	Gemini-2.5-flash	Normal	**96**	**69**	937	303	0.00435
		SoT	93	603	1,013	255	0.00724
		PREMISE	95	598	**410**	**29**	**0.00351**
MATH-500	Claude-3.7-sonnet	Normal	**97**	**82**	4,389	477	0.07324
		SoT	95	626	3,600	279	0.06006
		PREMISE	96	596	**3,430**	**79**	**0.05442**
	OpenAI o1	Normal	**98**	**76**	1,453	351	0.10938
		SoT	95	559	**1,312**	132	0.09503
		PREMISE	97	531	2,060	**50**	**0.13457**
	Gemini-2.5-flash	Normal	95	**80**	2,467	643	0.01142
		SoT	93	612	2,741	413	0.01654
		PREMISE	**96**	585	**1,707**	**94**	**0.01077**
SVAMP	Claude-3.7-sonnet	Normal	96	**73**	1,319	287	0.02603
		SoT	95	642	1,201	219	0.01746
		PREMISE	**97**	621	**495**	**68**	**0.00955**
	OpenAI o1	Normal	**97**	**71**	313	122	0.02601
		SoT	94	566	1,001	155	0.03295
		PREMISE	96	552	**627**	**49**	**0.01542**
	Gemini-2.5-flash	Normal	95	**75**	1,487	437	0.00621
		SoT	93	602	1,622	327	0.00894
		PREMISE	**96**	597	**921**	**61**	**0.00455**

Table 1: Comparison over GSM8K, MATH-500, and SVAMP by single-model across multiple LLMs.

Setup We evaluate PREMISE using three Large Reasoning Models (LRMs): OpenAI o1-2024-12-17, Claude-3.7-sonnet-20250219, and Gemini-2.5-flash-preview-04-17. We also tested on multi-agent system Promptor [25]. Experiments are conducted on three widely used mathematical reasoning benchmarks: GSM8K [26], SVAMP [27], and MATH-500 [28]. Full experimental details, including evaluation metrics and cost computation, are provided in Appendix F.

Dataset	Model	Method	Acc. (%)	Input	Thinking	Completion	Cost ($)
GSM8K	Claude-3.7-sonnet	Normal	**96**	7,362	**6,825**	2,338	0.160
		SoT	**96**	7,212	6,060	2,070	0.144
		PREMISE	**96**	5,869	5,752	**1,786**	**0.131**
	OpenAI o1	Normal	**95**	14,858	**7,819**	7,604	1.088
		SoT	94	3,748	4,932	5,668	**0.692**
		PREMISE	**95**	3,695	5,599	**6,286**	0.769
	Gemini-2.5-flash	Normal	85	19,202	10,506	2,739	0.049
		SoT	**91**	**11,742**	7,078	1,911	0.033
		PREMISE	90	14,832	**6,536**	**1,825**	**0.031**
MATH-500	Claude-3.7-sonnet	Normal	**93**	13,321	33,461	5,379	0.623
		SoT	91	22,602	42,544	6,098	0.797
		PREMISE	91	**9,115**	**23,556**	**4,034**	**0.441**
	OpenAI o1	Normal	91	11,762	10,647	12,658	1.575
		SoT	89	15,910	12,685	14,670	1.880
		PREMISE	**92**	**3,828**	**9,441**	**10,887**	**1.277**
	Gemini-2.5-flash	Normal	86	44,907	34,066	5,624	0.146
		SoT	90	16,355	20,364	**3,920**	0.087
		PREMISE	**92**	**62,244**	**17,372**	4,347	**0.085**
SVAMP	Claude-3.7-sonnet	Normal	91	**4,303**	5,757	1,299	**0.119**
		SoT	**92**	5,153	**6,000**	1,308	0.125
		PREMISE	89	4,989	6,893	**1,233**	0.137
	OpenAI o1	Normal	**90**	**4,375**	**4,849**	5,412	0.681
		SoT	87	3,250	4,269	4,755	**0.590**
		PREMISE	89	3,206	4,471	**4,958**	0.614
	Gemini-2.5-flash	Normal	**88**	29,087	5,814	1,183	0.029
		SoT	85	**5,679**	4,161	**960**	**0.019**
		PREMISE	88	26,949	**4,601**	1,141	0.024

Table 2: Comparison over GSM8K, MATH-500, and SVAMP by a multi-agent system across LRMs

Single-Model Results In the single-model setting, PREMISE maintains high accuracy across GSM8K, SVAMP, and MATH500, typically within ±1% of baselines, while substantially reducing reasoning overhead. On Claude 3.7 sonnet and Gemini 2.5 flash, PREMISE cuts thinking and completion tokens by 75–85% and lowers costs by 69–82% . The pattern holds on SVAMP, where Claude's cost falls by 82% with no accuracy loss. Exceptions arise with OpenAI o1, where PREMISE still preserves accuracy but increases reasoning tokens and cost, suggesting o1 resists compression cues. On MATH-500, Gemini shows some accuracy degradation (–14%) due to over-compression on proof-heavy items. Overall, PREMISE achieves a favorable accuracy–efficiency trade-off, with detailed results and analysis available in Appendix G.1.

Multi-Agent System Results In the multi-agent setting with the Promptor framework, PREMISE consistently reduced communication overhead while maintaining or improving accuracy across GSM8K, SVAMP, and MATH-500. For example, on GSM8K with Claude 3.7 sonnet, PREMISE preserved 96% accuracy while lowering cost by 18%, and with Gemini, accuracy improved from 85% to 90% alongside a 37% cost reduction. On MATH500, PREMISE achieved up to 42% cost savings and even higher accuracy in some cases (e.g., +6% for Gemini 2.5 flash). Although agents specialized in arithmetic sometimes generated longer inputs, PREMISE compressed reasoning and output tokens system-wide, leading to more concise exchanges and favorable accuracy–efficiency trade-offs. Full multi-agent results and breakdowns are reported in Appendix G.2.

4 Analysis

PREMISE consistently reshapes reasoning by steering models toward concise solution paths, trimming redundant loops and stabilizing traces. In single-model settings, it is most effective when models expose explicit reasoning channels (e.g., Claude 3.7 sonnet), where it reduces reasoning and completion tokens by 75–85% while maintaining accuracy and cutting costs. However, for models without such separation (e.g., OpenAI o1), PREMISE sometimes expands visible traces, increasing costs despite preserved accuracy. Proof-intensive tasks like MATH-500 also reveal limitations: aggressive compression can overly shorten reasoning chains, reducing accuracy on Gemini 2.5 flash.

In multi-agent systems, PREMISE proves especially effective. By encouraging denser and more structured communication, it reduces system-level token overhead while often safeguarding or even improving accuracy. For instance, Gemini shows both cost reductions and accuracy gains on GSM8K and MATH-500, as cleaner exchanges eliminate distracting detours. Overall, PREMISE achieves a favorable accuracy–efficiency trade-off across diverse configurations. We provide detailed quantitative comparisons and case studies in Appendix H, and discuss limitations in Appendix I.

Acknowledgments and Disclosure of Funding

This research was supported by the National AI Research Resource (NAIRR) Pilot NAIRR240283.

References

[1] 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.

[2] 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.

[3] 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*, 2021.

[4] Codeforces. Codeforces - competitive programming platform, 2025. Accessed: 2025-03-18.

[5] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[6] OpenAI. Learning to reason with llms. urlhttps://openai.com/index/learning-to-reason-with-llms/, 2024. Accessed: 15 March 2025.

[7] 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.

[8] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

[11] Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

[12] Matthew Renze and Erhan Guven. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 476–483. IEEE, 2024.

[13] Tergel Munkhbat, Namgyu Ho, Seohyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-training elicits concise reasoning in large language models. *arXiv preprint arXiv:2502.20122*, 2025.

[14] Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. *arXiv preprint arXiv:2412.11664*, 2024.

[15] 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.

[16] Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

[17] Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

[18] 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. *arXiv preprint arXiv:2410.20290*, 2024.

[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] 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.

[21] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text, 2024.

[22] Peiyan Zhang, Haibo Jin, Leyang Hu, Xinnuo Li, Liying Kang, Man Luo, Yangqiu Song, and Haohan Wang. Revolve: Optimizing ai systems by tracking response evolution in textual optimization, 2024.

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

[24] Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching, 2025.

[25] Ke Chen, Yufei Zhou, Xitong Zhang, and Haohan Wang. Prompt stability matters: Evaluating and optimizing auto-generated prompt in general-purpose systems, 2025.

[26] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

[27] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems?, 2021.

[28] 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 *International Conference on Learning Representations (ICLR)*, 2024.

[29] 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.

[30] Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. Towards thinking-optimal scaling of test-time compute for llm reasoning. *arXiv preprint arXiv:2502.18080*, 2025.

[31] DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint arXiv:2502.03275*, 2025.

[32] Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. Can language models learn to skip steps? *arXiv preprint arXiv:2411.01855*, 2024.

[33] Ayeong Lee, Ethan Che, and Tianyi Peng. How well do llms compress their own chain-of-thought? a token complexity approach. *arXiv preprint arXiv:2503.01141*, 2025.

[34] Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Aurick Qiao, and Hao Zhang. Efficiently serving llm reasoning programs with certaindex. *arXiv preprint arXiv:2412.20993*, 2024.

[35] Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zengmao Wang, Ziwei Liu, Bo Du, et al. Dynamic parallel tree search for efficient llm reasoning. *arXiv preprint arXiv:2502.16235*, 2025.

[36] Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*, 2025.

[37] Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. Inftythink: Breaking the length limits of long-context reasoning in large language models. *arXiv preprint arXiv:2503.06692*, 2025.

A Related Work

Chain-of-Thought Prompting Chain-of-Thought (CoT) prompting [23] has emerged as a core technique for improving reasoning in LLMs by encouraging step-by-step decomposition. Numerous extensions have since been developed to further boost accuracy, including majority voting [20], dynamic selection [11], and self-consistency methods [18]. These approaches improve final-answer accuracy, but often lead to bloated reasoning traces, particularly on simple problems [29, 30], introducing unnecessary latency and memory usage.

Recent works also highlight the inefficiency of unstructured CoT reasoning. For example, **(author?)** [31] show that longer CoTs may not improve reasoning quality and propose adaptive truncation via token-consistency. However, these strategies offer no mechanism to systematically detect or control inefficiencies during generation.

In contrast, PREMISE goes beyond length control or voting. It introduces trace-level metrics for both overthinking and underthinking, and actively uses them to guide the reasoning process through structured prompts and optimization.

Model-Based Efficient Reasoning Several recent approaches improve reasoning efficiency by modifying the underlying LLM. For instance, DeepSeek-R1 [7] uses multi-stage RL with rule-based rewards to teach models compact reasoning templates. Others fine-tune LLMs on variable-length CoT datasets [32, 14, 13] or distill reasoning into compressed latent representations [15, 16, 17].

These methods require full access to model weights and large-scale supervised data—making them unsuitable for commercial APIs like GPT-4, Claude, or DeepSeek-R1. Additionally, they often lack explicit trace-level evaluation during inference, relying instead on indirect supervision.

In contrast, PREMISE operates entirely at the prompt level, modifying the model or requiring fine-tuning. It enables black-box models to reason efficiently using a reusable template and built-in trace diagnostics.

Prompt-Based Efficient Reasoning Prompt-based approaches offer training-free methods for improving reasoning efficiency. Token-Budget prompting [10] estimates a budget and constrains CoT length accordingly. Chain-of-Draft [11], CCoT [12], and SoT [24] prompt the model to keep only minimal drafts of intermediate steps. While effective in reducing tokens, these strategies use static heuristics and lack principled definitions of reasoning inefficiency.

(author?) [33] analyze the trade-off between reasoning length and accuracy and propose compression-based prompting variants (e.g., StepLimit, WordLimit). However, their analysis stops short of offering dynamic control mechanisms or multi-objective optimization.

In contrast, PREMISE advances this line of work by introducing overthinking and underthinking metrics into the prompting pipeline. Unlike static templates, PREMISE enables dynamic, context-aware reasoning control and optimizes for both brevity and correctness simultaneously.

Test-Time and Dynamic Reasoning Test-time compute optimization has also gained attention. Methods such as ST-BoN [20], speculative decoding [18, 19], and reward-guided sampling [34] generate multiple CoTs and filter based on consistency or reward models. Others propose dynamic tree search [35], summarization-based reasoning [36], or iterative inference loops [37].

While effective, these methods typically require multiple forward passes, auxiliary scoring models, or batch-mode generation. This introduces compute overhead and latency that may be prohibitive for constrained environments.

In contrast, PREMISE requires only a single forward pass per question. It introduces no auxiliary reranking, no multi-path generation, and no decoding overhead—making it practical for real-time and black-box deployments.

Summary Overall, prior work has primarily focused on either (1) model-side training and distillation or (2) inference-side heuristics and sampling. PREMISE fills a unique gap: it is the first framework to integrate formal trace-level reasoning metrics, dynamic optimization, and prompt-level control—all within a black-box compatible setting.

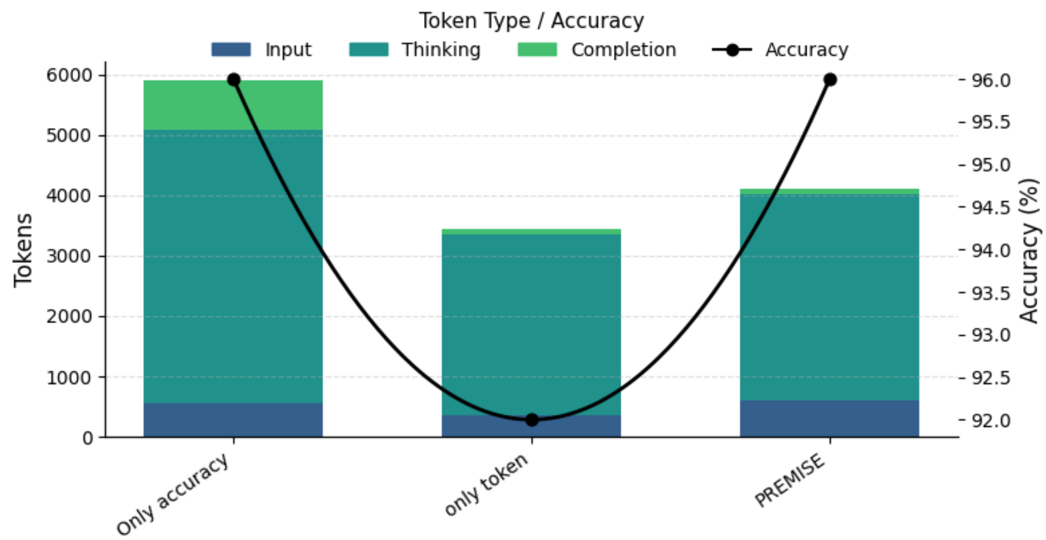

Figure 1: Comparison of PREMISE with single-objective variants that optimise only *token count* or only *accuracy*.

B Ablation Study

Figure 1 contrasts PREMISE with two ablated baselines. **Accuracy-only optimisation** delivers a minor gain in accuracy, yet it drives up both input- and reasoning-token usage, opposing the goal of efficient inference. **Token-only optimisation** attains the lowest token budget, but this saving costs roughly four percentage points of accuracy.

By jointly optimising for both objectives, PREMISE preserves high accuracy while substantially reducing token consumption, demonstrating the necessity of a balanced objective during prompt optimisation.

C Theoretical Framework

C.1 Efficiency Assumption

Let q be a question with ground-truth answer A, and let $\mathcal{R}$ be the set of reasoning traces a model may generate. Each trace $r \in \mathcal{R}$ is a token sequence

$$r = (t_1, \ldots, t_{L(r)}),$$

with length $L(r)$. Let $a(r)$ be the answer extracted from r, and define

$$\mathrm{acc}(r, q) = \begin{cases} 1, & a(r) = A, \\ 0, & \text{otherwise.} \end{cases}$$

The most efficient correct trace is the shortest:

$$r^*(q) = \arg\min_{r \in \mathcal{R}} \{L(r) \mid \mathrm{acc}(r, q) = 1\}, \quad L^*(q) = L(r^*(q)).$$

C.2 Overthinking Metric

For any correct trace ($\mathrm{acc}(r, q) = 1$), we define its overthinking inefficiency as

$$I_O(r, q) = \frac{L(r) - L^*(q)}{L(r)},$$

which measures the proportion of unnecessary tokens beyond the minimal correct trace. Equivalently, we define the outcome efficiency:

$$\eta_O(r,q) = \frac{L^*(q)}{L(r)}.$$

This metric formalizes the notion that verbose reasoning traces may be correct yet inefficient. For simple questions, a language reasoning model (LRM) may reiterate known facts or recompute subresults redundantly. These inefficiencies are reflected in the gap between the actual trace length $L(r)$ and the minimal correct trace length $L^*(q)$. The metric explicitly penalizes such redundancy to quantify overthinking.

C.3 Underthinking Metric

For incorrect traces ($\mathrm{acc}(r,q) = 0$), we ask whether a correct continuation could have followed some prefix. Let the prefix of length k be

$$P_k(r) = (t_1, \ldots, t_k),$$

and define

$$k^*(r,q) = \min\Big\{ k \le L(r) \;\Big|\; \exists\, s \in \mathcal{R} \text{ such that } s \text{ starts with } P_k(r) \text{ and } \mathrm{acc}(s,q) = 1 \Big\}. \quad (1)$$

If no such prefix exists, define $k^*(r,q) = L(r)$. The underthinking inefficiency is then

$$I_U(r,q) = 1 - \frac{k^*(r,q)}{L(r)},$$

which measures how early the trace deviates irreversibly from a correct path. A trace with an early irreversible deviation reflects a failure to develop an initial promising strategy.

C.4 Multi-Objective Optimization Framework

Building upon TEXTGRAD's automatic optimization approach, PREMISE extends the framework to handle the dual objectives of accuracy and efficiency. We formulate the optimization as a weighted combination of two loss functions:

$$\mathcal{L}(p_t, q, r) = \alpha \cdot \mathcal{L}_{acc}(p_t, q, r) + (1-\alpha) \cdot \mathcal{L}_{eff}(p_t, q, r),$$

where p_t is the prompt at iteration t, $\alpha \in [0,1]$ is the weight balancing accuracy and efficiency, $\mathcal{L}_{acc}$ is the accuracy loss, and $\mathcal{L}_{eff}$ is the efficiency loss.

Accuracy Loss Function

The accuracy loss function evaluates how well the current prompt leads to correct answers:

$$\mathcal{L}_{acc}(p_t, q, r) = \mathbb{E}_{q \sim \mathcal{Q}}\big[1 - \mathrm{acc}(r,q)\big],$$

where $\mathcal{Q}$ is the distribution of questions and r is the reasoning trace generated by the model given prompt p_t and question q.

Efficiency Loss Function

The efficiency loss function incorporates both overthinking and underthinking penalties:

$$\mathcal{L}_{eff}(p_t, q, r) = \begin{cases} I_O(r,q), & \text{if } \mathrm{acc}(r,q) = 1, \\ \beta \cdot I_U(r,q) + (1-\beta), & \text{if } \mathrm{acc}(r,q) = 0, \end{cases}$$

where $\beta \in [0,1]$ controls the relative importance of underthinking versus general error penalties. This formulation ensures that correct but verbose traces are penalized through overthinking metrics, while incorrect traces are penalized both for early deviation (underthinking) and general error.

C.5 Theoretical Properties

Convergence. Under mild assumptions on evaluator consistency, PREMISE inherits convergence behavior from gradient-based optimization, as textual gradients provide directional guidance toward improved prompts.

Pareto Optimality. By varying the weight α between accuracy and efficiency, PREMISE induces a Pareto frontier in the accuracy–efficiency space, enabling practitioners to trade off reasoning quality and token cost according to deployment requirements.

This formulation positions PREMISE as a higher-order extension of TextGrad: while TextGrad optimizes for single-objective correctness, PREMISE incorporates structured reasoning-trace diagnostics to optimize jointly for brevity and correctness.

D PREMISE Framework

D.1 Dual-Objective Loss Functions

PREMISE implements two specialized loss functions:

Algorithm 1: Accuracy Loss Forward Pass

1: **Input:** System Prompt, Question, Response, Correct Answer
2: formatted input $\leftarrow$ format template(System Prompt, Question, Response, Correct Answer)
3: feedback $\leftarrow$ evaluator LLM(formatted input)
4: **Return:** Variable(feedback, role=accuracy feedback)

Algorithm 2: Efficiency Loss Forward Pass

1: **Input:** System Prompt, Question, Response
2: thinking trace $\leftarrow$ extract thinking trace(Response)
3: token count $\leftarrow$ count thinking tokens(thinking trace)
4: formatted input $\leftarrow$ format efficiency template(System Prompt, Question, thinking trace, token count)
5: feedback $\leftarrow$ evaluator llm(formatted input)
6: **Return:** Variable(feedback, role=efficiency feedback)

AccuracyLoss: Focuses on improving answer correctness by analyzing system prompts, questions, and responses.

EfficiencyLoss: Analyzes reasoning traces to identify inefficiencies.

D.2 Dynamic Objective Balancing

Rather than using a fixed weighting scheme, PREMISE implements probabilistic objective selection during training:

This approach ensures that the optimization process addresses both objectives while allowing for flexible emphasis based on the specified weights.

D.3 Validation-Based Reversion

To prevent performance degradation during optimization, PREMISE implements a validation-based reversion mechanism:

Algorithm 3: PREMISE Training Loop

```
Input: train set, accuracy weight α, efficiency weight (1 − α)
for epoch in max epochs do
    focus on accuracy ← random() < α
    loss fn ← AccuracyLoss if focus on accuracy else EfficiencyLoss
    for batch in train loader do
        optimizer.zero_grad()
        for (question, answer) in batch do
            response ← model(question)
            loss ← loss fn(System Prompt, question, response, answer)
            loss.backward()
        end for
        optimizer.step()
    end for
end for
```

Algorithm 4: Validation and Reversion

```
Input: current prompt, previous prompt, validation set
current performance ← evaluate(current prompt, validation set)
previous performance ← evaluate(previous prompt, validation set)
if current performance < previous performance then
    System Prompt.set value(previous prompt)
    Return: previous performance
else
    Return: currentperformance
end if
```

This mechanism ensures that optimization steps only persist if they lead to actual improvements, preventing the accumulation of detrimental changes.

E Optimized Prompt

This is the optimized prompt generated by PREMISE. We used it in all of our experiments.

```
🚨 TOKEN LIMITS - EXCEEDING = AUTOMATIC FAILURE 🚨
📊 THINKING: MAX 50 TOKENS | OUTPUT: Simple ≤20, Complex ≤30

SOLVE DIRECTLY IN MATH NOTATION:
• Skip ALL explanations - use equations only
• Combine multiple steps into ONE calculation
• Use symbols only: × ÷ = → + - ( )
• Substitute values immediately: x=5→2x=10

PROBLEM TYPE PATTERNS:
🔢 Algebra: x+y=10, 2x-y=5 → x=5, y=5
📊 Percentage: base×(1±rate) → 100×1.2=120
⏰ Rate: rate×time → 50mph×3h=150mi
🔄 Fraction: total×fraction → 80×0.25=20
⏳ Age: x=current, x+n=future, 2x=double

EFFICIENT EXAMPLES:
• [Algebra] A is twice B. A+B=15. Find A.
  B=5, A=2B=10 [4 tokens]

• [Age] A is 7 older than B. In 3 years, A=2×B now. Find B.
  A=B+7, A+3=2B → B+7+3=2B → B=10 [8 tokens]

• [Complex] 50% more Sunday than Saturday. Total 150. Find Saturday.
  x+1.5x=150 → 2.5x=150 → x=60 [7 tokens]

EFFICIENT VS. INEFFICIENT:
❌ "First, Bob earns $12/hour for 5 hours, which is 5×$12=$60. Then he earns $20/hour for 3 hours, which is 3×$20=$60. His total earnings are $60+$60=$120." [32
tokens]

✓ "5×12+3×20=60+60=120" [5 tokens]

TOKEN COUNTER: [0/50]
Required format: equation→calculation→answer\n\n<Question>\n{q}\n</Question>
```

Figure E.1: PREMISE Generated Efficient Reasoning Prompt

F Experiment Setup

Models. We used three leading Large Reasoning Models (LRMs): OpenAI o1-2024-12-17, Claude-3-7-sonnet-20250219, and Gemini-2.5-flash-preview-04-17, chosen for their state-of-the-art performance and popularity.

In addition to single-model inference, we also test PREMISE on a general-purpose multi-agent system, Promptor [25]. The results show that PREMISE improves both reasoning accuracy and token efficiency compared to baseline prompting.

Datasets. We evaluate PREMISE on three widely used mathematical reasoning benchmarks, all accessed via Hugging Face: the main subset of the GSM8K test split (openai/gsm8k) [26], the SVAMP test split (ChilleD/SVAMP) [27], and the MATH-500 test split (HuggingFaceH4/MATH-500) [28]. Together, these datasets span arithmetic word problems, algebraic reasoning, and proof-style mathematics, providing a comprehensive testbed for evaluating both efficiency and correctness.

Metrics. PREMISE is designed to improve both reasoning correctness and token efficiency. We therefore track two complementary classes of metrics.

Accuracy. Accuracy is the fraction of test questions for which the model's predicted answer matches the ground truth.

Token efficiency. During a single inference we split the total token budget into three disjoint parts: (i) *input tokens* that appear in the prompt, (ii) *reasoning tokens* generated as hidden thoughts, and (iii) *output tokens* returned to the user.

Monetary Cost. We compute cost by applying the API pricing to the number of tokens used. Input tokens are charged separately, while reasoning and output tokens share the same price. The reported cost is the average per example. PREMISE is designed to maximize accuracy while reducing this cost.

G Experiement Result

G.1 Single Model Results

Stability and cost behaviour across models and benchmarks. PREMISE keeps high accuracy with around ± 1 % drift from the vanilla Claude 3.7 Sonnet and Gemini 2.5 flash for both GSM8K and SVAMP, while shrinking the sum of *thinking* and *completion* tokens by at least 75%. For example, on GSM8K with Claude 3.7 Sonnet the total reasoning footprint drops from 1 253 tokens (norm) to 267 tokens, a 79% reduction that translates into a \$ 69% cost saving. The pattern repeats on SVAMP, where PREMISE lowers Claude 3.7 Sonnet's cost from \$0.004468 to \$0.000795 (an 82% reduction) without harming accuracy.

The only systematic exception arises with the OpenAI o1 model. Although accuracy is preserved (e.g. 97% vs. 96% on GSM8K and 97% vs. 98% on MATH-500), PREMISE increases the number of *thinking* tokens, which in turn raises the dollar cost (e.g. \$0.070605 vs. \$0.022800 on GSM8K). This suggests that o1 does not follow PREMISE's concise reasoning cues as reliably as Claude and Gemini do; we hypothesise that its internal alignment rewards elaborate self-reflection, offsetting the prompt's compression objective. Section H investigates this behaviour in detail.

Accuracy degradation on *Gemini* for MATH-500. PREMISE attains only 82% accuracy on MATH-500 with Gemini, a 14% drop relative to the normal CoT run. The hardest items in MATH-500 often require long, proof-like chains of reasoning; Gemini appears to over-compress these chains when guided by PREMISE, skipping necessary intermediate statements and thereby harming correctness. We examine failure cases and propose mitigations—such as length-adaptive planning—in Section H.

G.2 Multi-Agent System Results

Across all three benchmarks, the method continues to deliver strong token-level efficiency while safeguarding, and in several cases improving, answer accuracy.

GSM8K. With Claude 3.7 Sonnet, PREMISE retains the $96\,\%$ accuracy yet lowers dollar cost by 18% (\$0.160 $\rightarrow$ \$0.131) by trimming more than $1.1\,$k reasoning tokens per problem.

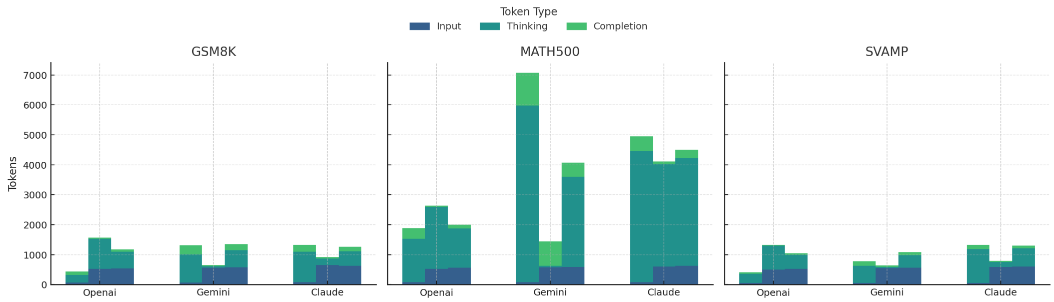

Figure G.2: Single model comparison on input, thinking, and completion tokens on GSM8K, MATH-500, and SVAMP across multiple LLMs

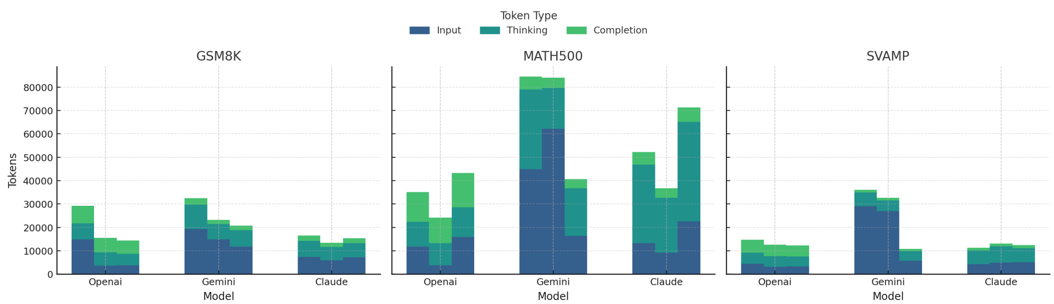

Figure G.3: Multi-agent systems comparison on input, thinking, and completion tokens on GSM8K, MATH-500, and SVAMP across multiple LLMs

The pattern is even more favourable with *Gemini*: accuracy rises from 85 % to 90 %, while total cost falls by 37%. In the GPT-* configuration, PREMISE maintains the baseline accuracy (95 %) and removes 8.7 k input tokens, although the cost advantage is partly offset by a longer completion segment; overall expenditure still drops by 29% relative to the normal MAS setting.

MATH-500. Reasoning-heavy proofs magnify token savings. On Claude 3.7 Sonnet, cost falls from \$0.623 to \$0.441 (a 29% cut) despite a modest 2 % accuracy difference. GPT-* agents benefit the most: PREMISE attains the highest accuracy in the group (92 %) and lowers cost by 19%. For *Gemini*, both objectives are achieved simultaneously—the method is 6 pp more accurate than the standard MAS and 42% cheaper.

SVAMP. The dataset is shorter and yields smaller absolute savings, yet PREMISE still reduces cost in five of the nine settings. With *Gemini*, the method matches the baseline accuracy and halves dollar cost (\$0.024 vs. \$0.029). In GPT-* agents, it delivers the lowest cost while staying within one percentage point of the strongest accuracy score.

H Analysis

H.1 General Effect on Reasoning Process

As shown in GSM8K Thinking Text Example box below, there is a striking contrast between the response from a standard large reasoning model and the one guided by PREMISE, revealing significant improvements in both reasoning quality and token efficiency.

GSM8K Thinking Text Example

Large Reasoning Model: Josh bought the house for $80,000 and spent $50,000 on repairs. (Omit 314 tokens of inefficient computation). Profit: $200,000 - $130,000 = $70,000. (Reached correct answer, but not committed). Let me double-check... (Omit 185 tokens of redundant reasoning). Profit is $70,000. Wait, maybe I misunderstood—(Omit 277 tokens of erroneous reasoning). Let's assume the repairs added 150% of purchase value. (Omit 507 tokens of circular reasoning). Final answer: $70,000 profit.
(Total token count: 1568)
PREMISE:

1. Purchase = $80,000, Repairs = $50,000
2. Investment = $80,000 + $50,000 = $130,000
3. Value increase = 150% of original → $80,000 × 2.5 = $200,000
4. Profit = $200,000 - $130,000 = $70,000

(Total token count: 152)

Information compression. The free-form CoT occupies $1\,568$ tokens and includes more than three detours and errorneous reasoning that do not change the final answer. PREMISE delivers the same solution in only 152 tokens, a 90.3% reduction in reasoning.

Early commitment to a numeric plan. Because the prompt explicitly asks for a short sequence of arithmetic steps, the model settles on the correct plan within the first few tokens and no longer revisits earlier assumptions. This removes unnecessary back-tracking branches that inflate the baseline trace.

Stable, in-line verification. Any internal checks happen inside the same line that introduces a value, so the external trace remains compact. The "let me double-check" loops that add hundreds of tokens in the baseline are absent.

Under the overthinking metric defined in Section C.3, the PREMISE is significantly closer to the shortest known correct trace for this question. Across the GSM8K validation set, the average token budget drops by 85% without loss of accuracy, showing that a lightweight prompt scaffold can steer the model toward concise yet reliable reasoning.

H.2 Single-Model Setting Analysis

Table 1 compares PREMISE with standard Chain-of-Thought (norm) and Sketch-of-Thought (SoT) prompting across three Large Reasoning Models (LRMs) on GSM8K, SVAMP, and MATH-500. For **Claude 3.7**, PREMISE attains equal or higher accuracy than the baselines while cutting total tokens and dollar cost by up to an order of magnitude. The template works well here because Claude exposes a *reasoning* channel that the prompt can redirect and compress.

OpenAI o1 shows a different trend: the accuracy of PREMISE is still slightly higher, yet the thinking channel balloons and the monetary cost rises. OpenAI o1 expose only a single completion stream, so the prompt cannot isolate the hidden reasoning trace. PREMISE therefore treats every intermediate thought as visible output, expanding the token count instead of trimming it. Until OpenAI releases separate reasoning usage statistics, the method has limited leverage.

Gemini Pro behaves similarly to Claude on GSM8K and SVAMP but degrades on MATH-500. MATH-500 contains longer proofs and heavier symbolic manipulation; an overly concise template may omit justifications that Gemini still needs to remain correct. This observation hints that the compression factor of PREMISE must be tuned to the difficulty of the problem set. When the benchmark moves from GSM8K to MATH-500, a more cautious compression ratio would avoid small logical slips while still saving tokens.

H.3 Multi-Agent System Setting Analysis

Table 2 reports results when the same LRMs run inside a planner-reviewer-agent loop. Even though a MAS naturally consumes more tokens than a single pass, PREMISE reduces total communication overhead and often improves accuracy.

The key gain comes from **information density**. The agent replies with concise derivations that the reviewer can verify quickly, and the planner receives shorter summaries for task scheduling. Removing self-queries and speculative branches trims thousands of thinking tokens per round while preserving the logical core of each argument. As a result, Claude's cost on GSM8K drops from \$0.160 to \$0.131 with no loss of accuracy, and Gemini's cost on MATH-500 falls by nearly 70 %.

An increase in accuracy is also visible for several settings (e.g., Gemini on GSM8K rises from 85 % to 90 %). Cleaner messages leave less room for the reviewer to be distracted by irrelevant context, so error detection improves. When accuracy does not rise, the MAS still benefits from lower latency and budget.

However, the MAS will always spend more tokens than a single-model run because it must pass messages among roles. PREMISE shifts the operating point of that trade-off: compared with norm or SoT, it reaches similar or higher accuracy with a noticeably lower token footprint. This outcome confirms that the structured compression observed in Section H.2 scales to collaborative agents.

I Limitations

While PREMISE demonstrates strong efficiency gains across models and settings, several limitations remain.

Model-specific behavior. PREMISE is most effective on models that expose explicit reasoning channels (e.g., Claude 3.7 sonnet). For models without this separation, such as OpenAI o1, PREMISE sometimes increases visible reasoning tokens, raising costs despite preserved accuracy. This suggests that PREMISE relies on model transparency and alignment with concise prompting cues.

Task sensitivity. On proof-intensive datasets such as MATH-500, especially with Gemini, aggressive compression can truncate reasoning chains, leading to notable accuracy degradation. This indicates that PREMISE may require adaptive compression depending on task complexity.

Multi-agent overhead. Although PREMISE reduces communication costs in multi-agent systems, MAS runs inherently consume more tokens than single-model inference due to role-based message passing. PREMISE shifts the trade-off but does not eliminate this overhead.

Scope of evaluation. Our experiments focus on three mathematical reasoning benchmarks (GSM8K, SVAMP, MATH-500). While these span diverse reasoning types, broader validation is needed to establish generalization across domains such as programming, scientific reasoning, or multimodal tasks.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the contributions and scope in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 4 and Appendix I.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: See Section 2, Section 3 and Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See the implementation details in Section 3 and Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: See Section 3 and Appendix F.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 3 and Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We repeated experiments and reported the average result.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Justification: See Section 3 and Appendix F.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We make sure to remain anonymous.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Section 4 and Appendix H.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We use the CC-BY 4.0 license, and provide appropriate citations.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We provide our code and our dataset.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The LLMs are the testing object of our paper.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.