前提：可扩展且战略性的提示优化，以提高大型模型中数学推理的效率

Ye Yu[1] **Yaoning Yu**[1] **Haohan Wang**[1]

[1]University of Illinois at Urbana–Champaign

Abstract

大型推理模型（LRMs）如 Claude 3.7 Sonnet 和 OpenAI o1 通过长链式思维（CoT）在数学任务上取得强劲表现，但经常生成不必要的冗长推理轨迹。这增加了标记使用和成本，限制了在对延迟敏感或 API 受限环境中的部署。我们引入了 PREMISE（基于提示的高效数学推理与战略评估），一个不修改模型权重的纯提示框架，旨在减少推理开销。PREMISE 结合了轨迹级诊断与基于梯度的提示优化，以在保持答案准确性的同时，最小化冗余计算。为了同时优化简洁性和正确性，PREMISE 使用了一种多目标文本优化过程，通过自然语言梯度平衡标记长度和答案有效性。与先前的方法不同，PREMISE 完全在单通道黑箱接口内运行，允许在商业 LLM 中进行高效推理。在 GSM8K、SVAMP 和 Math500 等数据集上，PREMISE 匹配或超过基线准确率（例如，Claude 在 GSM8K 上的 96% → 96%，Gemini 在 Math500 上的 91% → 92%），同时减少推理标记最多达 87.5 %，并削减 69-82 % 的美元成本。这些结果证明了提示级优化是提高 LRM 推理效率而不影响推理质量的实用且可扩展的途径。

1 介绍

大语言模型（LLMs）已成为自然语言理解和多步推理任务的强大工具。最近开发的专门用于推理的 LLMs——通常称为大推理模型（LRMs） (Xu et al., 2025a) ——推动了系统 2 推理的前沿，特别是在数学 (Cobbe et al., 2021b; Hendrycks et al., 2021) 和编程 (Codeforces, 2025; Chen et al., 2021) 领域。诸如 OpenAI 的 o1 (OpenAI) 和 DeepSeek-R1 (Guo et al., 2025) 等模型基于经过预训练的基础模型，例如 LLaMA (Touvron et al., 2023; Grattafiori et al., 2024)，并使用多阶段的有监督微调和强化学习来促进结构化的推理行为。

支持这些模型的核心策略是链式思考（CoT）提示 (Wei et al., 2022)，它将问题分解为明确的、逐步的推理。尽管 CoT 大大提高了推理的准确性，但也引入了低效性。即便是简单的算术问题也可能引发冗长且冗余的推理轨迹 (Chen et al., 2024)，这增加了令牌使用、推断延迟和内存消耗。这种“过度思考”的行为经常出现在较小的模型中 (Xu et al., 2025a)，但即便是最先进的 LRMs 也会表现出过度思考或相反的“思考不足”现象——即推理链过早截断并未能继续推理 (Wang et al., 2025; Su et al., 2025)。

在许多现实世界的环境中，如交互式助手、机器人规划系统或实时检索应用程序，这种低效率是不可接受的。基于令牌的计费、延迟限制和硬件瓶颈限制了商业部署中进行长推理链的可行性。因此，最近的研究开始探索高效的推理策略，包括长度受限的提示 (Han et al., 2024; Xu et al., 2025b; Renze and Guven, 2024)、使用压缩的 CoT 数据进行自我训练 (Munkhbat et al., 2025; Kang et al., 2024)、潜在空间推理 (Hao et al., 2024; Shen et al., 2025; Cheng and Van Durme, 2024) 和动态测试时路由 (Sun et al., 2024; Liao et al., 2025; Wang et al., 2025)。

然而，这些方法大多可以分为两大类：

1. 需要访问内部权重的模型级别适应（例如，微调，强化学习，潜在表示训练）
2. 基于提示的方法要么基于简单的启发式，要么在未考虑推理过程内部结构的情况下施加静态长度约束。

前者不适用于像 Claude 或 GPT 这样的闭源 API，而后者缺乏严格的优化和诊断工具来进行推理控制。

在本文中，我们提出了 PREMISE（基于提示的高效数学推理及策略性评估），这是一种专门设计用于在黑箱 LRM 中进行高效推理的仅限提示框架。PREMISE 引入了推理文本层面的指标，以诊断模型输出中的过度思考和思维不足，然后利用这些指标在一个可重复使用的提示结构中促使策略性推理。该方法明确指导模型避免冗余的分支，并提前承诺于高价值的解决路径。为了进一步提高符号效率，PREMISE 通过自然语言梯度 (Zhang et al., 2024) 进行多目标优化，平衡正确性与推理长度——所有这

些都无需修改模型权重。

我们在 GSM8K、SVAMP 和 Math500 上评估了 PREMISE，结果表明它在准确性上与或超过标准的 CoT 提示，同时将推理标记的使用减少高达 85 %。PREMISE 完全通过提示接口操作，使其适用于任何商业 LLM。据我们所知，这是第一个将轨迹级推理诊断与提示驱动的优化相结合的方法，用于在黑盒模型中实现高效推理。

我们的贡献有三个方面：

2 相关工作

2.1 思维链提示及其扩展

链式思维（CoT）提示已成为通过鼓励逐步分解来提高大型语言模型（LLM）推理能力的核心技术。自此，开发了许多扩展方法以进一步提高准确性，包括多数投票、动态选择和自一致性方法。这些方法提高了最终答案的准确性，但往往导致冗长的推理过程，特别是在简单问题上，引入了不必要的延迟和内存使用。

最近的研究也强调了非结构化链式推理的低效性。例如，Su et al. (2025) 表明较长的链式推理可能不会改善推理质量，并通过令牌一致性提出了自适应截断。然而，这些策略没有提供系统地检测或控制生成过程中低效率的机制。

与之相反，PREMISE 不仅仅局限于长度控制或投票。它引入了针对过度思考和思考不足的全新的轨迹级别指标，并主动使用这些指标通过结构化提示和优化来引导推理过程。

2.2 基于模型的高效推理

一些近期的方法通过修改底层的大型语言模型来提高推理效率。例如，DeepSeek-R1 (Guo et al., 2025) 使用多阶段的强化学习以及基于规则的奖励教模型紧凑的推理模板。其他方法则在变长的 CoT 数据集上微调大型语言模型 (Liu et al., 2024; Kang et al., 2024; Munkhbat et al., 2025) 或者将推理压缩成潜在表示中 (Hao et al., 2024; Shen et al., 2025; Cheng and Van Durme, 2024) 。

这些方法需要完全访问模型权重和大规模监督数据，这使得它们不适用于像 GPT-4、Claude 或 DeepSeek-R1 这样的商业 API。此外，它们在推理过程中通常缺乏明确的逐步评估，而是依赖于间接监督。

相比之下，PREMISE 完全在提示层面上运行，不需要修改模型或进行微调。它通过使用可重用的模板和内置的跟踪诊断，使黑箱模型能够高效推理。

2.3 基于提示的高效推理

基于提示的方法提供了无需训练的方式来提高推理效率。Token-Budget 提示 (Han et al., 2024) 估计预算并相应地限制 CoT 的长度。Chain-of-Draft (Xu et al., 2025b) ，CCoT (Renze and Guven, 2024) ，以及 SoT (Aytes et al., 2025) 提示模型仅保留中间步骤的最小草稿。虽然在减少标记方面有效，这些策略使用静态启发法且缺乏对推理低效的原则性定义。

Lee et al. (2025) 分析了推理长度和准确性之间的权衡，并提出了基于压缩的提示变体 (例如，StepLimit、WordLimit)。然而，他们的分析未能提供动态控制机制或多目标优化。

PREMISE 通过在提示流程中引入过度思考和思考不足的指标，推进了这一研究方向。与静态模板不同，PREMISE 实现了动态的、上下文感知的推理控制，同时优化简洁性和正确性。

2.4 测试时及动态推理

测试时的计算优化也引起了关注。方法如 ST-BoN (Wang et al., 2025) 、投机解码 (Sun et al., 2024; Liao et al., 2025) 和奖励引导采样 (Fu et al., 2024) 生成多个 CoT 并基于一致性或奖励模型进行筛选。其他方法提出了动态树搜索 (Ding et al., 2025) 、基于总结的推理 (Zhang et al., 2025) 或迭代推理循环 (Yan et al., 2025) 。

虽然有效，但这些方法通常需要多次前向传播、辅助评分模型或批处理生成。这会引入计算开销和延迟，对于受限环境可能是不可接受的。

相比之下，PREMISE 每个问题只需要一次前向传递。它不需要辅助重排序、不需要多路径生成，也没有解码开销——这使其在实时和黑盒部署中是实用的。

2.5 摘要

总体而言，之前关于高效推理的研究主要集中于（1）模型端的训练和蒸馏，或者（2）推理端的启发式和采样。PREMISE 填补了一个独特的空白：它是第一个在黑箱兼容环境中集成形式化的跟踪级推理度量、动态优化和提示级控制的框架。

3 方法

PREMISE 使用基于令牌化推理路径的跟踪级指标定义了过度思考和思考不足。我们首先建立整个分析中使用的基本符号和假设。

3.1 问题设置

设 q 是一个问题，其标准答案为 A ，并且设 $\mathcal{R}$ 表示模型可能为 q 生成的可能推理路径的集合。每个路径 $r \in \mathcal{R}$ 是一个令牌序列：

$$r = (t_1, t_2, \ldots, t_{L(r)}),$$

，其中 $L(r) \in \mathbb{N}$ 是 r 的令牌长度。设 $a(r)$ 表示从 r 中提取的答案，并定义二进制正确性指示符：

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

3.2 效率假设

在给定问题的所有正确推理路径中，我们将令最有效的路径是指标记数量最少的路径：

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

$$L^*(q) = L(r^*(q)).$$

3.3 过度思考度量

对于任何正确的轨迹（$\mathrm{acc}(r, q) = 1$ ），我们定义其过度思考效率低下为：

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

，这测量了超出最小正确轨迹的不必要标记的比例。同样地，我们定义结果效率为：

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

3.4 低思维度量

对于不正确的路径（$\mathrm{acc}(r, q) = 0$ ），我们询问是否一些前缀可以正确地继续。设长度为 k 的前缀为

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

并设置

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

如果不存在这样的前缀，定义 $k^*(r, q) = L(r)$ 。那么欠思考低效性就是

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

，它衡量了路径偏离正确路径并不可逆的时刻有多早。

3.5 汇总指标

对于问题-轨迹对的数据分布 $\mathcal{D}$ ，我们计算期望的低效性：

$$\Xi_O = \mathbb{E}_{(q,r)\sim\mathcal{D}} \left[I_O(r, q) \cdot \mathbf{1}_{\mathrm{acc}(r,q)=1} \right],$$
$$\Xi_U = \mathbb{E}_{(q,r)\sim\mathcal{D}} \left[I_U(r, q) \cdot \mathbf{1}_{\mathrm{acc}(r,q)=0} \right].$$

3.6 多目标优化

为了优化推理轨迹的正确性和简洁性，我们将生成过程表述为一个在 $\mathcal{R}$ 上的多目标优化问题。定义目标向量：

$$F(r) = \big(L_{\mathrm{acc}}(r),\, L_{\mathrm{len}}(r)\big) = \big(1 - \mathrm{acc}(r, q),\, L(r)\big),$$

，其中 L_{acc} 惩罚错误答案，L_{len} 惩罚较长的轨迹。

我们寻求帕累托最优前沿：

$$\{r^* \in \mathcal{R} \mid \nexists\, r \in \mathcal{R} :\; F(r) \prec F(r^*)\},$$

，其中 $F(r) \prec F(r^*)$ 表示帕累托优势（即，两项目标均不差，且至少一项更优）。

为了探索这一前沿，我们使用基于梯度的提示优化方法 (Zhang et al., 2024) 在自然语言空间中执行可微优化。令 $\delta_{\mathrm{acc}} = \nabla_{\mathrm{text}} L_{\mathrm{acc}}(r)$ 和 $\delta_{\mathrm{len}} = \nabla_{\mathrm{text}} L_{\mathrm{len}}(r)$ 表示每个目标的文本梯度。通过凸组合将其标量化：

$$\delta = \lambda\, \delta_{\mathrm{acc}} + (1 - \lambda)\, \delta_{\mathrm{len}}, \qquad \lambda \in [0, 1].$$

然后通过迭代更新这一轨迹：

$$r \leftarrow \mathrm{TGD\_step}(r,\, \delta),$$

允许在准确性和令牌效率之间进行权衡。通过调整 λ ，我们可以生成在部署环境的约束下平衡这些相互竞争目标的轨迹。

4 实验

4.1 实验装置

模型。我们使用了三个领先的大型推理模型（LRMs）：OpenAI o1-2024-12-17、Claude-3-7-sonnet-20250219 和 Gemini-2.5-flash-preview-04-17，它们因其先进的性能和受欢迎程度而被选中。

除了单模型推理之外，我们还在通用多代理系统 Promptor (Chen et al., 2025) 上测试了 PREMISE。结果显示，与基线提示相比，PREMISE 提高了推理准确性和令牌效率。

数据集。为了全面评估我们方法的效率和正确性，我们在三个广泛使用的数学推理数据集上进行了实验：GSM8K (Cobbe et al., 2021a)

Table 1: 在 GSM8K、MATH-500 和 SVAMP 上的比较，针对多个大型语言模型的单一模型比较。

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

，SVAMP (Patel et al., 2021)，以及 MATH-500 (Lightman et al., 2024)。

指标。PREMISE 旨在提高推理的正确性和令牌的效率。因此，我们跟踪两个互补的指标类别。

准确性。给定数据集 $\{(x_i, y_i)\}_{i=1}^{N}$，模型 M 达到

$$\text{Acc} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{ M(q(x_i)) = y_i \},$$

其中 x_i 和 y_i 是一个数学问题-答案对，q 是推理模式，$\mathbb{I}\{\cdot\}$ 是指示函数。

令牌效率。在一次推理过程中，我们将总令牌预算分为三个不相交的部分：(i) 出现在提示中的输入令牌，(ii) 作为隐藏思维生成的推理令牌，以及 (iii) 返回给用户的输出令牌。这些计数的提取取决于提供者：

- OpenAI。`prompt_tokens` 表示输入计数；`reasoning_tokens`（如果有的话）记录隐藏的想法；输出计数是 `completion_tokens` − `reasoning_tokens`。
- Claude。客户报告 `input_tokens` 和 `output_tokens`。我们利用提供的 `count_tokens` 例程，应用于流式隐秘密迹来近似推理标记。
- 双子座。元数据中的 `prompt_token_count`、`thoughts_token_count` 和 `candidates_token_count` 分别直接映射到输入、推理和输出部分。

货币成本。对于每个模型，我们将相应的 API 价格应用于每个分段的令牌使用。由于推理令牌和输出令牌的成本相同，我们定义两个价格 $w_{\mathrm{I}}, w_{\mathrm{O}} \in \mathbb{R}_{>0}$，其中 w_{I} 是每个输入令牌的成本，而 w_{O} 是每个推理或输出令牌的成本。设 $\overline{C}_{\mathrm{I}}$、$\overline{C}_{\mathrm{R}}$ 和 $\overline{C}_{\mathrm{O}}$ 分别表示每个例子的输入、推理和输出令牌的平均数量。每个例子预期的总成本由以下公式给出：

$$\text{Cost} = w_{\mathrm{I}} \cdot \overline{C}_{\mathrm{I}} + w_{\mathrm{O}} \cdot \left(\overline{C}_{\mathrm{R}} + \overline{C}_{\mathrm{O}}\right).$$

PREMISE 的目标是在最大化 Acc 的同时最小化 $\mathcal{C}$。

4.2 单模型结果

在不同模型和基准之间的稳定性和成本行为。对于 GSM8K 和 SVAMP，PREMISE 在保持高准确度的同时，与基础模型 Claude 3.7 Sonnet 和 Gemini 2.5 flash 的漂移约为 ±1 %，同时将思考和完成令牌的总和缩减至少 75% 。例如，在 GSM8K 上，使用 Claude 3.7 Sonnet，总推理

Table 2: 通过一个多代理系统对比多个大型语言模型在 GSM8K、MATH-500 和 SVAMP 上的表现

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

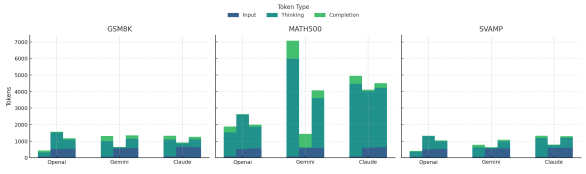

Figure 1: 单模型在 GSM8K、MATH-500 和 SVAMP 上的输入、思考及完成标记的多种大语言模型比较

占用从 1 253 令牌（标准）减少到 267 令牌，减少间隔为 79%，成本节省为 $ 69%。此模式在 SVAMP 上重复出现，PREMISE 将 Claude 3.7 Sonnet 的成本从 $ 0.004468 降低到 $ 0.000795（减少 82%）而不损害准确性。

唯一的系统性例外出现于 OpenAI 的 o1 模型。虽然准确性得以保持（例如，在 GSM8K 上的 97% 对比 96%，以及在 MATH-500 上的 97% 对比 98%），但是 PREMISE 增加了思考标记的数量，这反过来提高了美元成本（例如，在 GSM8K 上的 $ 0.070605 对比 $ 0.022800）。这表明 o1 并没有像 Claude 和 Gemini 那样可靠地遵循 PREMISE 的简明推理提示；我们假设其内部对齐奖励了详尽的自我反省，抵消了提示的压缩目标。第 5 节详细调查了这一行为。

对于 MATH-500，该方法在 Gemini 上的准确性下降。在 MATH-500 的 Gemini 上，PREMISE 仅达到 82% 的准确率，相比正常的 CoT 运行下降了 14%。MATH-500 中最难的题目通常需要长的、类似证明的推理链；而 Gemini 在 PREMISE 的指导下似乎过度压缩了这些推理链，跳过了必要的中间步骤，从而影响了准确性。我们在 5 节中检查失败案例并提出缓解措施，如长度自适应规划。

4.3 多智能体系统

在所有三个基准测试中，该方法继续提供强大的令牌级效率，同时在多个情况下保护并提高答案准确性。

GSM8K。通过 Claude 3.7 Sonnet，PREMISE 保持了 96% 的准确性，同时通过每个问题减少超过 1.1 k 的推理 tokens，将成本降低了 18%（$ 0.160 → $ 0.131）。对于 Gemini，该模式更加有利：准确性从 85% 提升到 90%，而总成本下降了 37%。在 GPT-* 配置中，PREMISE 保持了基线精度（95%）并去除了 8.7 k 输入 tokens，尽管由于完成段更长，成本优势部分被抵消；总体支出仍然比正常的 MAS 设置降低了 29%。

MATH-500。基于推理的证明放大了标记节省。在 Claude 3.7 Sonnet 上，成本从 $ 0.623 下降到 $ 0.441（减少了 29% ），尽管准确率仅有适度的 2 XMATHPC 差异。GPT-* 代理收益最大：PREMISE 在组中取得了最高的准确率（92 % ）并降低了成本 19% 。对于 Gemini，两个目标同时实现——该方法比标准 MAS 准确了 6 pp 并且更便宜了 42% 。

SVAMP。数据集较短，绝对节省更小，但在九种设置中仍有五种减少了成本。使用 Gemini，该方法匹配基线精度并将成本减半（$ 0.024 对比 $ 0.029）。在 GPT-* 代理中，该方法在保持最强精度分数的一个百分点内的同时提供了最低成本。

5 分析

5.1 对推理过程的一般影响

GSM8K Thinking Text Example

Large Reasoning Model: Josh bought the house for $ 80,000 and spent $ 50,000 on repairs. （省略 314 个无效计算的标记）. Profit: $ 200,000 - $ 130,000 = $ 70,000. 达到了正确答案，但未确认. Let me double-check...（省略 185 个冗余推理词）. Profit is $ 70,000. Wait, maybe I misunderstood—（省略 277 个错误推理的标记）. Let's assume the repairs added 150 % of purchase value.（省略 507 个循环推理的标记）. Final answer: $ 70,000 profit.
（总标记数：1568）

PREMISE:
1. 购买 = $ 80,000, 修理 = $ 50,000
2. 投资 = $ 80,000 + $ 50,000 = $ 130,000
3. 价值增加 = 原来的 150 % $ 80,000 Œ 2.5 = $ 200,000
4. 利润 = $ 200,000 - $ 130,000 = $ 70,000

(总字数: 152)

如上图所示，标准的大型推理模型的响应与由 PREMISE 指导的响应之间存在显著的对比，显示出推理质量和标记效率的显著提高。

信息压缩。自由形式的 CoT 占据了 1 568 个符号，并包含了三个以上的曲折和错误推理，这些并不改变最后的答案。PREMISE 在仅仅 152 个符号中提供了相同的解决方案，推理减少了 90.3% 。

及早确定一个数字计划。因为提示明确要求提供一个简短的算术步骤序列，模型在最开始的几个标记中就确定了正确的计划，并且不再重新考虑早期的假设。这消除了那些使基准路径膨胀的不必要的回溯分支。

稳定的在线验证。任何内部检查都发生在引入值的同一行内，因此外部跟踪保持简洁。基线中增加数百个标记的“让我仔细检查”循环则不存在。

根据在第 3.4 节中定义的过度思考指标，PREMISE 明显更接近此问题已知最短的正确路径。在 GSM8K 验证集中，平均标记预算在不损失准确性的情况下减少了 85% ，表明一个轻量级的提示框架可以引导模型进行简洁而可靠的推理。

5.2 单模型设置分析

表 1 比较了在 GSM8K、SVAMP 和 MATH-500 上的三个大型推理模型（LRM）的 PREMISE 与标准的思维链（norm）和思维草图（SoT）提示。对于 Claude 3.7，PREMISE 达到了与基线相等或更高的准确性，同时减少了总代币和美元成本达一个数量级。该模板在这里效果良好，因为 Claude 暴露了一个推理通道，可以让提示重定向和压缩。

OpenAI 展示了不同的趋势：PREMISE 的准确率仍稍高，但思维通道膨胀，货币成本上升。GPT 模型仅暴露单一的完成流，因此提示无法隔离隐藏的推理轨迹。因此，PREMISE 将每一个中间思维视为可见输出，扩大了标记数而非减少它。在 OpenAI 发布单独的推理使用统计之前，该方法的杠杆效用有限。

Gemini Pro 在 GSM8K 和 SVAMP 上的表现与 Claude 类似，但在 MATH-500 上有所退化。MATH-500 包含更长的证明和更复杂的符号操作；一个过于简明的模板可能会省略 Gemini 仍需保持正确的理由。这个观察表明，PREMISE 的压缩系数必须根据问题集的难度进行调整。当基准从 GSM8K 转向 MATH-500 时，更谨慎的压缩比可以避免小的逻辑失误，同时仍能节省标记。

5.3 多智能体系统设置分析

表格 2 显示了在规划者-审核者-代理循环中运行相同 LRM 的结果。尽管 MAS 自然会比单次通过消耗更多的 token，PREMISE 减少了总通信开销，且常常提高了准确性。

关键的收益源于信息密度。代理使用简洁的推导进行回应，评审者可以快速验证这些推导，而计划者可以获得更短的任务调度摘要。去除自我查询和推测分支在每个回合中减少了数千个思考标记，同时保留了每个论点的逻辑核心。结果，Claude 在 GSM8K 上的成本从 $ 0.160 降至 $ 0.131，而准确性没有损失，Gemini 在 MATH-500 上的成本下降了近 70 %。

在多个设置中，准确性也有提高（例如，Gemini 在 GSM8K 上从 85 % 提升到 90 %）。更清晰的信息减少了审阅者被不相关背景分散注意力的可能性，从而提高了错误检测能力。当准确性没有上升时，MAS 仍然受益于更低的延迟和预算。

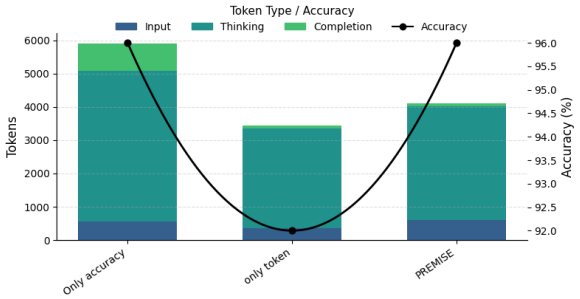

Figure 2: 将 PREMISE 与仅优化 token 数量或仅优化准确性的单目标变体进行比较。

然而，MAS 总是比单一模型运行消耗更多的代币，因为它必须在各个角色之间传递信息。PREMISE 转移了该权衡的操作点：与 norm 或 SoT 相比，它以明显较低的代币使用量达到相似或更高的准确性。这个结果证实了在第 5.2 节观察到的结构化压缩可以扩展到协作代理。

6 消融研究

图 2 将 PREMISE 与两个简化的基线进行对比。单纯优化准确性虽然带来了轻微的准确性提升，但同时也增加了输入和推理标记的使用，这与高效推理的目标相悖。单纯优化标记使用达到了最低的标记预算，但这种节省大约以四个百分点的准确率为代价。

通过同时优化两个目标，PREMISE 在保持高准确性的同时大幅减少了 token 消耗，证明了在提示优化过程中保持平衡目标的必要性。

7 结论

我们提出了 PREMISE，一个仅使用提示的框架，在不更改模型权重的情况下提高大型推理模型（LRMs）的数学推理效率。通过将过度思考和不足思考的跟踪级诊断与多目标自然语言优化方案结合，PREMISE 引导生成简洁而准确的解决路径。

在 GSM8K、SVAMP 和 MATH-500 上，PREMISE 在答案准确率上与标准的链式思维提示相匹配或超越，同时减少了多达 87.5 % 的推理标记，并降低了 69–82 % 的货币成本。这些节省在单次传递设置和多代理系统中都适用，表明仅仅通过提示级别的控制，在接口限制为黑箱 API 调用时，也可以获得显著收益。

研究也揭示了一些局限性。当没有显式的推理渠道暴露时——以基于 GPT 的模型为例——当前的模板可能会延长可见轨迹并增加成本。同样，在使用 Gemini 的以证明为主的 MATH-500 集上，过于激进的压缩率会导致漏掉中间的理由和准确性的损失。这些情况突显了需要适应性的压缩，以使令牌预算与任务难度和给定模型的接口特征保持一致。

未来的工作将把诊断扩展到符号或多模态推理任务。我们相信，这样的方向将在保持逐步推理的透明性和可靠性的同时，进一步降低推理成本。

References

Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *Preprint*, arXiv:2503.05179.

Ke Chen, Yufei Zhou, Xitong Zhang, and Haohan Wang. 2025. Prompt stability matters: Evaluating and optimizing auto-generated prompt in general-purpose systems. *Preprint*, arXiv:2505.13546.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

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

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. 2021a. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021b. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Codeforces. 2025. Codeforces - competitive programming platform. Accessed: 2025-03-18.

Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zengmao Wang, Ziwei Liu, Bo Du, and 1 others. 2025. Dynamic parallel tree search for efficient llm reasoning. *arXiv preprint arXiv:2502.16235*.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

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

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

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

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

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

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*.

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

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

OpenAI. Learning to reason with llms. urlhttps://openai.com/index/learning-to-reason-with-llms/. Accessed: 15 March 2025.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *Preprint*, arXiv:2103.07191.

Matthew Renze and Erhan Guven. 2024. 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.

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

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

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. 2024. Fast best-of-n decoding via speculative rejection. *arXiv preprint arXiv:2410.20290*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, and 1 others. 2025a. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*.

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

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

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

Peiyan Zhang, Haibo Jin, Leyang Hu, Xinnuo Li, Liying Kang, Man Luo, Yangqiu Song, and Haohan Wang. 2024. Revolve: Optimizing ai systems by tracking response evolution in textual optimization. *Preprint*, arXiv:2412.03092.

A 附录

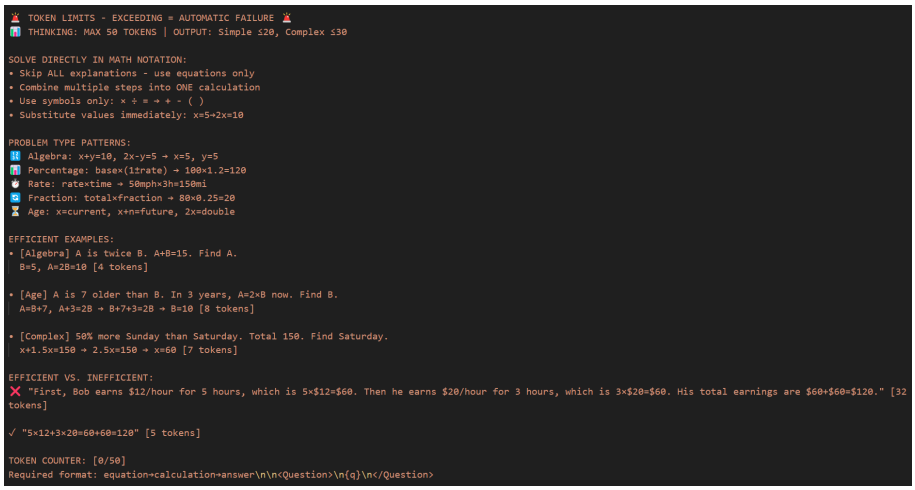

Figure A.1: 生成的高效推理提示