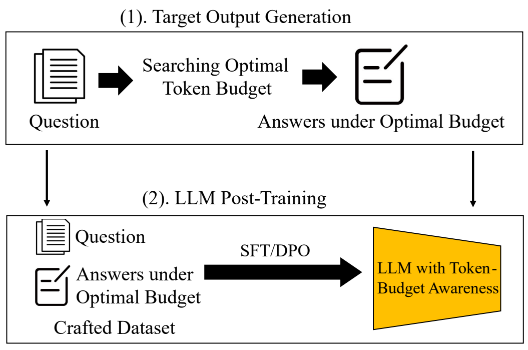
(1). Target Output Generation
Question
Searching Optimal Token Budget
Answers under Optimal Budget
(2). LLM Post-Training
Question
Answers under Optimal Budget
Crafted Dataset
SFT/DPO
LLM with Token-Budget Awareness

Figure 6: The workflow of TALE-PT. Given a set of questions, TALE-PT first generates target outputs in phase (1), searching the answers under optimal budget. In phase (2), TALE-PT uses the searched target outputs to craft a specialized dataset, then post-train the LLM (via SFT/DPO) to internalize token-budget awareness.

rithm 1 and Algorithm 2). Figure 1d illustrates an example. The resulting LLM output, constrained by the token budget specified in the prompt, is taken as the crafted target output y_i. This target output not only leads to the correct answer but also has minimal actual output token cost among our token elasticity-based search process, as described in Section 4. In the LLM post-training stage, we train the LLM $\mathcal{M}_\theta$ using the crafted target outputs from the first stage. We introduce two ways to conduct the token-budget awareness internalization during post-training, i.e., SFT-based and DPO-based method. Details of the hype parameters are in Section A.3.

SFT-based Internalization. To inject token-budget awareness into $\mathcal{M}_\theta$, we perform supervised fine-tuning with these target outputs. We post-train $\mathcal{M}_\theta$ to generate token-efficient outputs by minimizing the cross-entropy loss between the model's predictions and the target outputs. Given an input $\boldsymbol{x}$ and a target output y from the first stage (which reflects token-budget awareness), the cross-entropy loss is defined as:

$$\mathcal{L}_{\text{CE}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i}\log \mathbb{P}(y_{i,t}|y_{i,<t},\boldsymbol{x}_i),$$

where T_i means the length of the target sequence y_i for the i-th training example, $y_{i,t}$ the target token at position t of y_i, $y_{i,<t}$ means the sequence of tokens preceding the current token $y_{i,t}$, representing the context up to time step t for the i-th sample. $\mathbb{P}(y_{i,t}|y_{i,<t},x_i)$ represents the conditional probability predicted by the model $\mathcal{M}_\theta$ for the token $y_{i,t}$, given the input $\boldsymbol{x}_i$ and the preceding tokens $y_{i,<t}$. The loss is based on the next token prediction. The goal is to adjust the model parameters θ such that it produces concise and accurate responses that adhere to the token budget constraint. This is achieved through gradient descent, forcing the model to internalize the compact reasoning patterns from the token-efficient target outputs.

DPO-based Internalization. Another way to incentivize $\mathcal{M}_\theta$ to learn the token-budget preference is applying the DPO algorithm (Rafailov et al., 2023) to post-train the model. DPO directly refines the policy through a classification objective, aligning the model's behavior with the desired preferences. The goal of DPO here is to refine $\mathcal{M}_\theta$ so it can accurately solve a given problem $\boldsymbol{x}$ while adhering to an internalized token budget. We use the target outputs y_i from the searched optimal budget as positive samples, while outputs y_i' generated with the vanilla CoT prompt serve as negative samples. These positive-negative pairs are then used to create the pairwise preference data for DPO training. Given the crafted dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i, y_i')\}_{i=1}^{N}$, the objective is to maximize the likelihood that the model ranks the positive samples higher than the negative ones. Formally, we aim to optimize the following objective:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\log P_\theta(y_i \succ y_i'), \quad \textit{where}$$

$$P_\theta(y_i \succ y_i') = \frac{\exp\big(s(y_i,\mathbf{x}_i)\big)}{\exp\big(s(y_i,\mathbf{x}_i)\big)+\exp\big(s(y_i',\mathbf{x}_i)\big)}.$$

$P_\theta(y_i \succ y_i')$ is the preference function. Here, $s(y_i,\boldsymbol{x}_i)$ is defined as $\sum_{t=1}^{T_i}\log \mathbb{P}(y_{i,t}|y_{i,<t},\boldsymbol{x}_i)$, and it represents the log-probability of the model generating y_i for input $\boldsymbol{x}_i$, which serves as the preference score assigned to y_i. This score measures how strongly the model favors that output. The objective ensures that the model prioritizes concise and token-efficient outputs while maintaining high-quality reasoning and correctness. During training, the LLM is encouraged to internalize the token budget constraint and adopt a more compact reasoning process guided by the target outputs generated in the first stage. This two-stage process effectively trains the LLM to produce concise yet accurate responses, striking a balance between reasoning quality and token efficiency during inference. More details are in Section A.3.

6 Evaluation

In this section, we provide the experiment results to evaluate the effectiveness of two versions of TALE,

Table 3: Comparison of TALE-EP (estimation and prompting) and other prompt engineering methods. "Directly Answering" means prompting LLM without any reasoning process. "Vanilla CoT" means the vanilla CoT prompting without budget. The model used in our evaluation is GPT-4o-mini (OpenAI, 2024a). Observe that TALE-EP achieves an average accuracy (ACC) of 80.22%, with an average output token cost of 138.53 and an average expense of 118.46. TALE-EP reduces output token costs by 67%, lowers expenses by 59%, and maintains competitive performance compared to the vanilla CoT approach. ACC ↑, Output Tokens ↓, Expense (10^{-5}\$ / sample) ↓.

Dataset	Directly Answering			Vanilla CoT			TALE-EP		
	ACC ↑	Output Tokens ↓	Expense ↓	ACC ↑	Output Tokens ↓	Expense ↓	ACC ↑	Output Tokens ↓	Expense ↓
GSM8K	28.29%	12.46	39.43	81.35%	318.10	541.09	84.46%	77.26	279.84
GSM8K-Zero	97.21%	18.85	91.69	99.50%	252.96	886.79	98.72%	22.67	276.12
MathBench-Arithmetic	59.67%	41.10	9.78	75.00%	313.51	78.58	73.67%	39.60	18.62
MathBench-Middle	33.33%	5.00	3.58	84.67%	553.93	68.22	79.33%	238.14	42.95
MathBench-High	51.33%	5.00	4.07	84.00%	653.24	82.44	80.00%	254.82	47.61
MathBench-College	44.00%	5.00	3.68	78.00%	675.78	81.56	70.00%	259.85	45.60
Average	52.31%	14.57	25.37	83.75%	461.25	289.78	81.03%	148.72	118.46

Table 4: The generalization of TALE-EP (estimation and prompting) across different LLMs. Yi-lightning (Wake et al., 2024), GPT-4o-mini (OpenAI, 2024a), GPT-4o (OpenAI, 2024b) and o3-mini (OpenAI, 2025) are taken into consideration. We conduct following evaluations on the MathBench-College dataset. ACC ↑, Output Tokens ↓, Expense (10^{-5}\$ / sample) ↓.

LLM	Directly Answering			Vanilla CoT			TALE-EP		
	ACC ↑	Output Tokens ↓	Expense ↓	ACC ↑	Output Tokens ↓	Expense ↓	ACC ↑	Output Tokens ↓	Expense ↓
Yi-lightning	66.67%	80.01	3.09	79.33%	998.10	21.55	76.67%	373.52	17.25
GPT-4o-mini	44.00%	5.00	3.68	78.00%	675.78	81.56	70.00%	259.85	45.60
GPT-4o	57.33%	5.00	61.34	84.00%	602.29	1359.42	80.00%	181.61	759.95
o3-mini	96.00%	601.51	336.69	97.33%	1163.55	638.46	96.66%	677.65	385.12

TALE-EP and TALE-PT. The comparisons of the two implementations are detailed in Section A.6.

6.1 Experiment Setup

Datasets. To evaluate the LLM performance, three most challenging mathematical datasets are taken into consideration: GSM8K (Cobbe et al., 2021), GSM8K-Zero (Chiang and Lee, 2024), and MathBench (Liu et al., 2024). GSM8K-Zero, derived from the GSM8K dataset, specifically targets the analysis of over-reasoning and redundancy in LLM-generated outputs. In short, GSM8K-Zero is designed so that the answers are embedded within the questions themselves. LLMs can easily generate correct responses without complicated additional reasoning or redundant calculations.

Models. We conduct experiments on five state-of-the-art LLMs (i.e., GPT-4o (OpenAI, 2024b), GPT-4o-mini (OpenAI, 2024a), Yi-lightning (Wake et al., 2024), o3-mini (OpenAI, 2025)), and Lllama-3.1-8B-Instruct (Dubey et al., 2024).

Metrics. The target of TALE is to balance the LLM correctness performance and extra redundant token costs. Specifically, TALE seeks to minimize *Number of Output Tokens* while maintaining comparable *Accuracy (Acc)* simultaneously.

Accuracy (Acc). This metric is calculated as the following: $Accuracy = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{\mathcal{M}(\boldsymbol{x}_i) = y_i\}$, where $(\boldsymbol{x}_i, y_i) \in \mathcal{X}$. $\boldsymbol{x}_i$ is the math question from dataset $\mathcal{X}$ and y_i the ground truth answer. $\mathcal{M}(\cdot)$ returns the answer for a given question. $\mathbb{I}\{\cdot\}$ represents an indicator function. This function evaluates whether the inside given condition holds. Specifically, it returns **1** if the condition is true and **0** if the condition is false. For a better evaluation, we format the LLM output by crafting an elaborate instruction detailed in Figure 8.

Number of Output Tokens. We evaluate the token costs by calculating the average output token consumption for each specific task. The output token costs are measured as follows: $Number\ of\ Output\ Tokens = \frac{1}{N}\sum_{i=1}^{N} \mathbb{T}(\mathcal{M}(\boldsymbol{x}_i))$, where $\boldsymbol{x}_i$ represents the given question, and $\mathbb{T}$ is a function that measures the number of tokens. Intuitively, the more output tokens, the higher the costs incurred by $\mathcal{M}$. To evaluate costs more precisely, we calculate the average expense per sample. The total token expense includes both input and output tokens used during the query process.

6.2 Effectiveness of TALE-EP

Table 3 compares TALE-EP with other prompt engineering methods across seven datasets, evaluating accuracy, output tokens, and expenses. Effective prompts should maximize accuracy while minimizing token usage and cost. Direct Answering is the most cost-efficient (14.57 tokens, 25.37 expense) but with low accuracy (52.31%). Vanilla

Table 5: Comparison of TALE-PT (post-training to internalize token-budget awareness) and other prompt engineering methods. Two different post-training methods, SFT and DPO, are taken into consideration.

LLM	Dataset	Directly Answering		Vanilla CoT		TALE-PT-SFT		TALE-PT-DPO	
		ACC ↑	Output Tokens ↓	ACC ↑	Output Tokens ↓	ACC ↑	Output Tokens ↓	ACC ↑	Output Tokens ↓
Llama-3.1-8B-Instruct	GSM8K	21.00%	38.54	77.56%	241.51	78.57%	139.63	74.11%	149.93
	GSM8K-Zero	70.32%	13.49	65.04%	251.08	78.43%	77.85	78.41%	113.41

CoT achieves the highest accuracy (83.75%) but at a high cost (461.25 tokens, 289.78 expense). TALE-EP balances performance and efficiency, achieving 81.03% accuracy while reducing token usage to 32% and expenses to 41% of Vanilla CoT. On GSM8K, it even surpasses Vanilla CoT with 84.46% accuracy. Note that expense is not directly proportional to output tokens because it also accounts for input and cached tokens. TALE-EP reduces token costs by 68.64% on average, offering a scalable, cost-effective solution for budget-constrained reasoning tasks. For resource-rich scenarios, we evaluate TALE-EP under a larger token budget as detailed in Section A.8.

To further evaluate the generalization of TALE-EP across different LLMs. We conduct experiments across Yi-lightning, GPT-4o-mini, GPT-4o and o3-mini on MathBench-College. Table 4 illustrates the results, showing TALE-EP's ability to reduce output tokens and expenses while maintaining competitive accuracy significantly. TALE-EP achieves substantial token savings, reducing output tokens by 64.63% on average, compared to Vanilla CoT. Expense reductions are equally notable, with costs decreasing by 45.30% on average. Despite these cost savings, TALE-EP maintains strong accuracy, achieving 76.67% on Yi-lightning, 70.00% on GPT-4o-mini, and 80.00% on GPT-4o, comparable to Vanilla CoT. These results highlight TALE-EP's effectiveness in balancing cost efficiency and reasoning performance across diverse LLM architectures. The observed accuracy drop is most significant for GPT-4o-mini. This could be attributed to its smaller number of parameters, which makes it more challenging to answer correctly within a limited response reasoning length. We also evaluate the applicability of TALE on more tasks, the results are illustrated in Section A.5. The efficiency analysis of TALE-EP is in Section A.7

6.3 Effectiveness of TALE-PT

Table 5 compares TALE-PT methods with Vanilla CoT and Direct Answering on GSM8K and GSM8K-Zero using Llama-3.1-8B-Instruct. For GSM8K, Direct Answering demonstrates the lowest token usage (38.54) but at the cost of significantly reduced accuracy (21.00%). In contrast, Vanilla CoT achieves much higher accuracy (77.56%) but incurs a significant increase in token cost (241.51). Note that on GSM8K-Zero, the accuracy of Vanilla CoT drops below Direct Answering. This drop can be attributed to overthinking, as GSM8K-Zero is simpler, with answers often implied directly within the question. In such cases, a long reasoning process can introduce unnecessary complexity, leading to reduced accuracy. Among the TALE-PT methods, TALE-PT-SFT achieves the best accuracy (78.57%, 78.43%) with reduced tokens, while TALE-PT-DPO balances accuracy (74.11%, 78.41%) and token efficiency, cutting token consumption by over 50% on GSM8K-Zero compared to Vanilla CoT.

7 Conclusion

In this paper, we introduce TALE, a framework that reduces token redundancy in Chain-of-Thought (CoT) reasoning by incorporating token budget awareness. TALE dynamically adjusts the number of reasoning tokens based on the reasoning complexity of each problem, balancing token efficiency and answer correctness. Experiments show that TALE reduces output token usage and expense significantly with acceptable accuracy loss, outperforming Vanilla CoT in cost-effectiveness while generalizing well across various LLMs.

8 Limitations

The experiments of our proposed token-budget-aware reasoning framework currently focus on LLMs that process only text as input and output. While the results demonstrate significant improvements in efficiency and cost reduction, it does not account for models that have multimodal output content. Such as the models generate interleaved images and text as output. In future work, we will extend token-budget awareness to such LLMs with multimodal output by introducing modality-specific budget constraints and designing adaptive strategies to optimize token efficiency for different modality types, such as images and videos.