Thinking Economically: A Hierarchical Framework for Adaptive-Complexity Reasoning in LLMs

Yubo Gao[1,2]*, Haotian Wu[3]*, Hong Chen[1,2], Junquan Huang[1], Yibo Yan[1,2], Jungang Li[1], Zihao Dongfang[1], Sicheng Tao [1], Puay Siew Tan[4], Jie Zhang[3], Xuming Hu[1,2]

[1]The Hong Kong University of Science and Technology (Guangzhou),
[2]The Hong Kong University of Science and Technology,
[3]Nanyang Technological University,
[4]Singapore Institute of Manufacturing Technology, A*STAR

Correspondence: xuminghu@hkust-gz.edu.cn

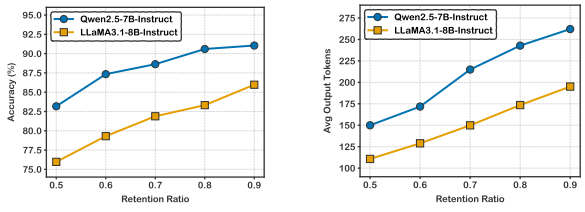
Abstract

Chain-of-Thought (CoT) has significantly enhanced LLM reasoning, yet often incurs substantial computational overhead due to "overthinking": generating excessively long rationales without commensurate accuracy gains. Existing efficiency methods typically apply uniform compression, which overlooks a critical observation that reasoning complexity is heterogeneous at two distinct granularities: across different problems and within individual reasoning steps. This motivates our principle of **Thinking Economically**: intelligently allocating computational resources based on intrinsic task and step demands rather than pursuing uniform brevity. We propose Hierarchical Adaptive Budgeter (HAB), a training framework that operationalizes this principle through coarse-to-fine budgeting. At the inter-step level, HAB predicts the optimal reasoning depth for each problem. At the intra-step level, HAB learns step-specific token budgeting signals from PPL-derived step comparisons and an adaptive Pareto optimization objective that captures the local quality-efficiency trade-off, while a Fisher Information-based pruner further provides fine-grained training-time guidance, thereby encouraging the generator to internalize more economical reasoning patterns. Experiments on GSM8K and MATH500 show that HAB not only surpasses standard CoT in accuracy but also reduces token usage, achieving a stronger performance-efficiency trade-off than the compared baselines.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable reasoning capabilities through Chain-of-Thought (CoT) prompting (Wei et al., 2022), which decomposes complex tasks into intermediate steps (Brown et al., 2020; Grattafiori et al., 2024). To further improve CoT, various extensions have been proposed, including self-consistency sampling (Wang et al., 2022), Tree-of-Thoughts (Yao et al., 2023), Graph-of-Thought (Besta et al., 2024), and instruction tuning with reasoning traces (Cai et al., 2024). However, these methods introduce significant inefficiencies. Specifically, they often engage in "overthinking" (Chen et al., 2024b; Team et al., 2025), generating excessively long reasoning chains without corresponding accuracy gains. Such redundant steps not only increase computational cost but also risk prematurely exhausting the token budget and introducing logical inconsistencies that undermine answer correctness and clarity (Sui et al., 2025).

A promising direction to address this issue is imposing token constraints or budgets. Existing methods fall into two categories: pre-defined and learning-based constraints. For pre-defined constraints, such as prompt-based length guidance (Xu et al., 2025a; Lee et al., 2025; Renze and Guven, 2024; Chen et al., 2024a) or post-processing techniques like TokenSkip (Xia et al., 2025), selecting an appropriate retention ratio remains challenging. Our pilot analysis in Figure 1 shows that a lower retention ratio reduces token count but causes a sharp drop in accuracy. Therefore, finding a satisfactory balance between efficiency and performance with a static, pre-defined constraint is difficult.



(a) Accuracy (b) Average Output Tokens

Figure 1: Comparison of performance of TokenSkip with different LLMs backbones under different retention ratios on the GSM8K dataset.

To address this, several methods based on **learned constraints** have been proposed. These

* These authors contributed equally to this work.

Findings of the Association for Computational Linguistics: ACL 2026, pages 39434–39444

include Reinforcement Learning (RL) (Luo et al., 2025; Yeo et al., 2025) approaches that penalize length, which in practice may be harder to stabilize than direct supervised objectives, and more recent methods like TALE (Han et al., 2025) that leverage the LLM's own capabilities to estimate a global budget first, and induce the LLM to reason under such a budget. While a significant step forward, these methods still focus on learning a single, *global* token budget for the entire reasoning chain.

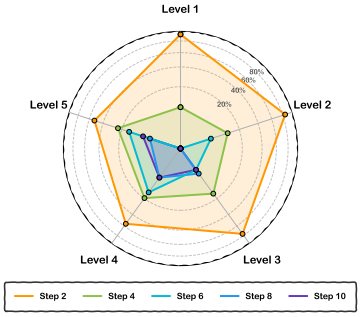
We contend that a global budget, whether predefined or learned, is fundamentally limited because it ignores the varying complexity at two granularities (especially the second granularity): **across different problems** and **within the reasoning steps of a single problem**. Our analysis on the MATH500 dataset (Figure 2) provides empirical motivation for hierarchical budgeting. At the inter-step level, shown in Figure 2(a), simpler problems require fewer reasoning steps, while complex ones need longer chains. At the intra-step level, as shown in Figure 2(b)(c), our experiments on Chain-of-Draft (CoD) show that allowing the model to allocate more tokens to difficult steps ($\text{CoD}_{w/relaxation}$) leads to significant performance gains. This indicates that different steps have varying token requirements, and a uniform constraint is suboptimal. These findings motivate our principle of "Thinking Economically": instead of uniformly pursuing brevity, we aim to adaptively allocate resources based on hierarchical task demands. The key is to concentrate computation on difficult *problems* and *reasoning steps*, while saving resources on simpler ones.

To realize this vision, we introduce Hierarchical Adaptive Budgeter (HAB), a two-stage training framework for adaptive-complexity reasoning that moves beyond a single global budget toward hierarchical control. HAB first performs inter-step control by predicting a coarse reasoning-depth category for each problem and converting it into a step-range instruction for generation, thereby determining how much reasoning the problem should receive at the global level. It then performs intra-step learning within the generated reasoning process: PPL-derived step comparisons provide supervision on relative step difficulty, an adaptive Pareto optimization objective models the local quality-efficiency trade-off, and a Fisher Information-based pruner supplies additional fine-grained training guidance. Through this coarse-to-fine design, HAB encourages the generator to internalize more economical reasoning behaviors, allocating more computation to harder problems and more demanding steps while avoiding unnecessary verbosity on simpler cases. Our main contributions are:

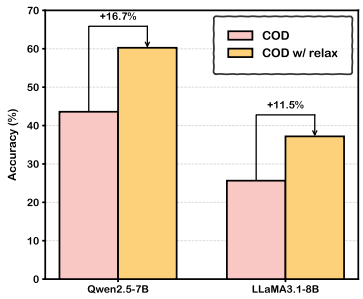
- We propose the principle of "Thinking Economically" and instantiate it with HAB, a coarse-to-fine hierarchical framework that first performs explicit inter-step reasoning-depth control and then learns finer-grained intra-step efficiency patterns during training.
- We design an adaptive Pareto optimization scheme to dynamically balance quality and efficiency for each reasoning step.
- Experiments on GSM8K and MATH500 show that HAB consistently achieves a stronger performance-efficiency trade-off than the compared baselines, improving accuracy while reducing token usage.

2 Related Work

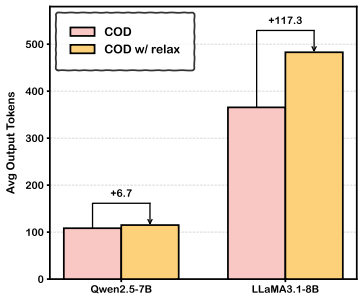
Chain-of-Thought Reasoning. Chain-of-Thought (CoT) prompting (Wei et al., 2022) enhances reasoning in Large Language Models (LLMs) by decomposing solutions of complex problems into intermediate steps. Early works demonstrated its effectiveness in both few-shot (Wei et al., 2022) and zero-shot (Kojima et al., 2022) settings, but standard CoT often suffers from error propagation due to its linear reasoning path. To mitigate this limitation, subsequent research has introduced structured reasoning strategies. Least-to-Most prompting (Zhou et al., 2022) addresses compositional generalization by sequentially solving decomposed sub-problems. Tree-of-Thoughts (ToT) (Yao et al., 2023) enables non-linear exploration through tree-search algorithms with backtracking. Self-Consistency (Wang et al., 2022) improves stability by sampling diverse reasoning trajectories and applying majority voting. Auto-CoT (Zhang et al., 2022) further automates exemplar construction via clustering and diversity selection, reducing manual annotation costs. Beyond inference-time prompting, training-stage methods have been proposed to internalize reasoning capabilities. Symbolic CoT Distillation (Li et al., 2023) transfers reasoning skills from teacher models to smaller students. Self-Taught Reasoner (STaR) (Zelikman et al., 2022) establishes an iterative bootstrapping loop where models improve by training on their own generated rationales. Recent



(a) **Inter-reasoning step**: Optimal step distribution by difficulty.



(b) **Intra-reasoning step**: CoD vs. $\text{CoD}_{w/\text{relaxation}}$ (Accuracy).



(c) **Intra-reasoning step**: CoD vs. $\text{CoD}_{w/\text{relaxation}}$ (Avg output tokens).

Figure 2: Exploratory experiments on the necessity of dynamically allocating the number of reasoning steps tailored to problems and allocating token budgets tailored to reasoning steps.

work addresses practical constraints such as robustness (Liu et al., 2024), generalization to unseen tasks (Yin et al., 2025), and inference efficiency (Xu et al., 2025b). CoT has also expanded into multimodal domains (Wang et al., 2025a) and lightweight adaptation for small-scale models (Zhuang et al., 2025).

CoT Token Constraints. Existing methods for controlling token usage fall into two paradigms: pre-defined constraints and learned constraints. **Pre-defined Token Constraints.** These methods fix the efficiency strategy via a static constraint. The simplest form is *prompt-based length guidance*, which requests a token or step budget in the prompt (for example, "use fewer than 50 tokens") to shorten the rationale (Zhang et al., 2025a; Wang et al., 2025b). A second line uses *post hoc compression*: the model first produces a long CoT, then removes parts that appear redundant. TokenSkip (Xia et al., 2025) implements controllable CoT compression by skipping low-importance tokens. C3oT (Kang et al., 2025) trains a compressor and conditions generation on a target compression level to produce shorter CoT. The main limitation is that a single budget or ratio rarely fits all instances. It can truncate hard questions' reasoning or waste tokens on easy ones, which harms the efficiency-accuracy trade-off.

Learning-based Token Constraints. To reduce rigidity, learning-based methods optimize the trade-off during training. RL-style approaches incorporate length costs into the objective, so the model learns when to stop or how much to "think" for each input, such as DAST (Shen et al., 2025a), O1-Pruner (Luo et al., 2025), and related analyses that study length growth and stabilization in long-CoT training (Yeo et al., 2025). These methods are adaptive, but they often require careful reward design and may be unstable. TALE (Han et al., 2025) moves toward explicit budget control by estimating a global token budget per problem and enforcing it through budget-aware prompting or post-training, which improves the predictability of inference cost. *Latent reasoning* is a distinct branch within learned constraints. These methods perform intermediate reasoning in hidden states rather than generating explicit tokens/full textual rationale (Hao et al., 2024; Cheng and Van Durme, 2024; Shen et al., 2025b; Xu et al., 2025b; Zhang et al., 2025b). This reduces decoding cost substantially but sacrifices interpretability and fine-grained control over computational allocation.

3 Methodology

Our method, Hierarchical Adaptive Budgeter (HAB), is founded on the principle of "Thinking Economically", moving beyond the prevailing paradigm of uniform chain shortening to instead guide the model to dynamically allocate a computational budget tailored to a task's hierarchically intrinsic demands. HAB operationalizes this principle through a coarse-to-fine hierarchical process. At the **inter-step** level, it first predicts a coarse reasoning-depth category for each problem and converts it into a step-range instruction for generation. Subsequently, at the **intra-step** level, HAB allocates a specific token budget (retention ratio) to each individual step within the generated reasoning chain, tailored to their intrinsic difficulty. To realize this fine-grained control, HAB first derives step-level difficulty signals from **PPL-derived step comparisons**, which capture the relative complexity of different reasoning steps. It then learns the corresponding retention signals through

an **adaptive Pareto optimization** objective that models the local quality-efficiency trade-off, encouraging the model to preserve more computation for more demanding steps while compressing simpler ones more aggressively. During training, a **Fisher Information**-based pruner further provides token-level retention guidance within each step, and the resulting fine-grained compression patterns are internalized into the model parameters for inference. In this way, HAB unifies coarse-grained **inter-step** reasoning-depth control and fine-grained **intra-step** budget learning within a hierarchical framework, ultimately encouraging the generator to internalize economical reasoning behaviors.

3.1 Problem Formulation

Given an input question q and a LLM $\mathcal{M}$ with parameters θ, HAB performs hierarchical control at two levels. At the **inter-step** level, the model predicts a coarse reasoning-depth category $C_{\text{opt}} = f_{\text{macro}}(q;\theta)$, where C_{opt} denotes the preferred reasoning-depth category (e.g., short, medium, or long). This predicted category is then mapped to a step-range instruction for generation. At the **intra-step** level, HAB learns step-level retention signals over training-time reasoning steps $S = \{s_1, s_2, \ldots, s_N\}$, where each step s_j consists of L_j tokens. These retention signals are optimized during Stage-2 training to capture fine-grained quality-efficiency trade-offs and are subsequently internalized into the generator parameters for inference.

3.2 Inter-step: Requirement-Aware Reasoning Step Control

The first stage of HAB addresses the coarse-grained task of determining the optimal length of the reasoning path. The objective is to train the model to predict the most resource-efficient reasoning depth category, C_{opt}, which corresponds to the optimal number of steps, N_{opt}, required to solve a given problem correctly. To achieve this, we construct a high-quality dataset, $\mathcal{D}_{SCP}$, that is utilized to train the model for reasoning depth prediction. For each question q_i from benchmarks such as MATH500 and GSM8K, we first use a powerful LLM (e.g., Qwen-Max) to generate a portfolio of CoT solutions with a varying number of steps. To ensure that the step count reflects genuine reasoning granularity rather than superficial formatting, we impose a constraint on the average word count per step during generation. After verifying the correctness of each generated chain against the ground-truth answer, we select the chain with the fewest reasoning steps as the optimal reasoning chain (*Questions for which no correct reasoning chain was generated are excluded*). The step count from this chain is defined as the optimal number of steps, $N_{opt,i}$. Finally, we apply a series of data cleaning procedures, including standardizing step delimiters, removing redundant formatting, and normalizing mathematical expressions, to ensure dataset quality.

As shown in Figure 2(a), the distribution of optimal step lengths is heavily skewed, causing severe class imbalance. To address this, we group step counts into three balanced categories: short, medium, and long. The thresholds are dataset-specific: for **MATH**, we define 1-2 steps as *short*, 3-4 as *medium*, and 5+ as *long*; for **GSM8K**, the thresholds are 1, 2, and 3+ steps respectively. This bucketing transforms sparse step prediction into a balanced classification problem, enabling more robust learning.

During training, we employ a LoRA-tuned LLM-based **Router** to classify the reasoning depth of a given problem. It outputs a probability distribution p_i over our three pre-defined categories: *short*, *medium*, *long*. We optimize this classification task using a standard Cross-Entropy (CE) loss, $\mathcal{L}_{SCP}$. Let y_i be the one-hot vector representing the ground-truth category for a given problem. The loss is computed over each training batch $\mathcal{B}$ as:

$$\mathcal{L}_{\text{SCP}} = -\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\sum_{k=1}^{3} y_{i,k}\log(p_{i,k}). \quad (1)$$

At inference time, the Router first predicts the most likely reasoning-depth category (e.g., '*medium*'). This prediction is then mapped to its corresponding step-count range under our bucketing rules and incorporated into the prompt as an explicit instruction, which guides the downstream LLM to reason within the predicted step range. This provides an elastic budget for the number of steps rather than an overly rigid one. For instance, if the Router predicts the '*medium*' category for a MATH problem, the prompt becomes: "Let's solve the problem step-by-step with 3 to 4 steps." Notably, the role of Qwen-Max is only to construct the supervision signal for reasoning-depth categories during data preparation; the CoT solutions generated by Qwen-Max are not directly used as reasoning traces during downstream generation.

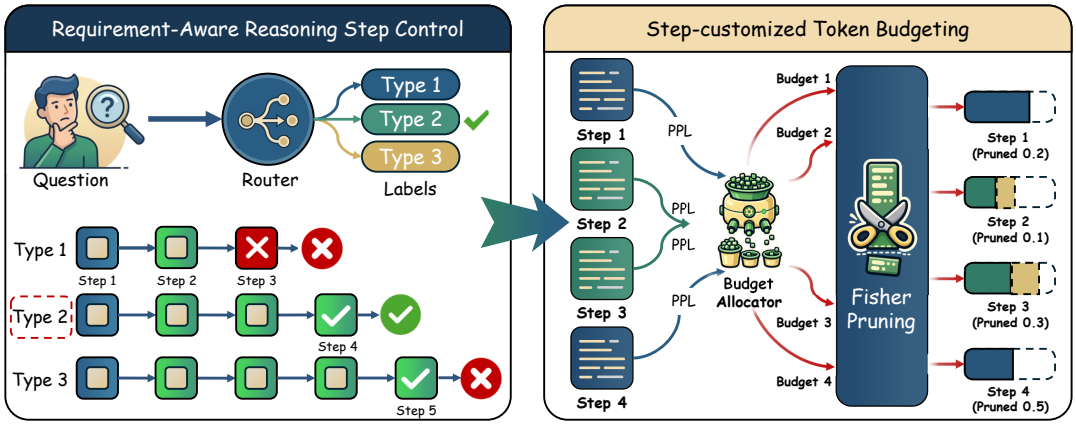


Figure 3: The overall framework of HAB. The Qwen-Max branch is used only during data preparation to construct supervision for reasoning-depth categories; its generated CoT solutions are not directly used as training traces for the downstream model.

3.3 Intra-step: Step-customized Token Budgeting

Once the inter-step plan is established, this stage learns a fine-grained, dynamic token budgeting mechanism over the reasoning process under the predicted step-range instruction. Our approach involves a two-phase process: (1) we first learn a robust indicator of each step's intrinsic complexity, and (2) we then use an adaptive Pareto optimization scheme to translate this complexity into a token budget (retention ratio) that balances performance and efficiency, with a Fisher Information-based pruner further providing token-level retention guidance during training.

3.3.1 Step Complexity Indicator

Complexity Metric Definition. Effective budget allocation requires accurate estimation of each step's intrinsic complexity. We develop a learnable complexity estimator that captures the computational demands of individual reasoning steps. Inspired by prior work (Cui et al., 2025), we adopt perplexity (PPL) as a practical complexity proxy, as it correlates with the relative difficulty of different reasoning steps. For step $s_{i,j}$ (the j-th step of sample i), we compute:

$$\mathrm{PPL}(s_{i,j}) = \exp\left(-\frac{1}{L_{i,j}}\sum_{k=1}^{L_{i,j}} \log P(t_{i,j,k} \mid \mathrm{context}_{<(i,j,k)}; \theta)\right), \tag{2}$$

where $L_{i,j}$ is the token length of step $s_{i,j}$, $t_{i,j,k}$ is the k-th token in step $s_{i,j}$, and $\mathrm{context}_{<(i,j,k)}$ includes all preceding tokens. In implementation, step-wise PPL is computed during Stage-2 data preparation and used only as a training-time supervision signal.

Learning Complexity Indicator. We train a difficulty prediction head $h_{\text{diff}} : \mathbb{R}^d \to \mathbb{R}$ to estimate step complexity from the last hidden states. Direct regression on raw PPL values often suffers from high variance and poor generalization. Instead, to capture relative difficulty differences between reasoning steps, we adopt a ranking-based objective. For each sample i with N_i steps, we consider all $\frac{N_i(N_i-1)}{2}$ unique pairs of steps $(s_{i,j}, s_{i,k})$. To further refine the learning signal, we treat pairs with smaller PPL differences as harder to distinguish and therefore assign them larger weights. Specifically, we define

$$w_{jk} = \frac{1}{|\mathrm{PPL}(s_{i,j}) - \mathrm{PPL}(s_{i,k})| + \epsilon}, \tag{3}$$

where ϵ is a small smoothing constant. These weights are then normalized within each sample using a softmax function to produce w'_{jk}. The final ranking loss encourages the predicted score of the harder step, d_{hard}, to be greater than that of the easier one, d_{easy}, by at least a margin m:

$$\mathcal{L}_{\text{pair}} = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\sum_{(j,k)} w'_{jk} \cdot \max(0,\ m - (d_{\text{hard}} - d_{\text{easy}}))\,, \tag{4}$$

where $d_{i,j} = h_{\text{diff}}(\mathrm{StepEncoder}(s_{i,j}))$ is the predicted difficulty score, and the harder/easier ordering is determined by the relative PPL ranking of the compared steps.

The resulting latent score $d_{i,j}$ is an unbounded value. To map this score to a normalized token retention ratio $r_{i,j} \in [0, 1]$, we apply a learnable affine transformation followed by a sigmoid function (Budget Allocator):

$$r_{i,j} = \sigma(w \cdot d_{i,j} + b). \tag{5}$$

This provides a simple and effective trainable mapping from a raw difficulty score to an actionable token budget, and in implementation serves as the budget allocator during Stage-2 training.

3.3.2 Adaptive Pareto Optimization for Budget Allocation

The token budget for each step, represented by a retention ratio $r_{i,j} \in [0, 1]$, is learned by balancing two conflicting objectives: reasoning quality ($\mathcal{L}_{\text{qual},i}$) and computational efficiency ($\mathcal{L}_{\text{eff},i}$). Intuitively, complex steps require more tokens to preserve quality, while simple steps can be compressed more aggressively with minimal quality loss. Traditional methods use fixed trade-off weights, which cannot adapt to varying step complexities. Pareto optimization provides a principled perspective for such multi-objective problems by characterizing the trade-off between competing objectives (Sener and Koltun, 2018; Lin et al., 2019; Pimentel et al., 2020; Zhou et al., 2024). Inspired by this perspective, we design an *adaptive Pareto optimization* scheme that dynamically adjusts the quality-efficiency balance according to local step difficulty.

First, we define the two conflicting losses. **Quality** is measured by the standard language modeling loss (negative log-likelihood) under pruning, while **efficiency** is the average token retention ratio:

$$\begin{aligned} \mathcal{L}_{\text{qual},i} &= -\sum_{j=1}^{N_i} \sum_{k=1}^{L_{i,j}} \log P(t_{i,j,k} \mid S'_{\text{pruned},i,<(j,k)}) \\ \mathcal{L}_{\text{eff},i} &= \frac{1}{N_i} \sum_{j=1}^{N_i} r_{i,j}. \end{aligned} \tag{6}$$

To find the optimal trade-off, we introduce a **Pareto Curvature Probe**. The core idea is to estimate the marginal cost of efficiency: "*If we tighten our budget by a tiny amount δ, how much quality do we lose?*" We approximate this using a lightweight perturbation-based probe to compute the local slope (ϕ_i) of the Pareto frontier:

$$\phi_i = \frac{\mathcal{L}'_{\text{qual},i} - \mathcal{L}_{\text{qual},i}}{\delta}, \tag{7}$$

where $\mathcal{L}'_{\text{qual},i}$ is the quality loss under a more aggressive budget $r'_{i,j} = \max(r_{i,j} - \delta, 0)$. This slope quantifies the marginal cost of efficiency. A high, positive slope indicates that we are in a "steep" region of the trade-off curve, where even small efficiency gains are very costly to quality. A low slope indicates that we are in a "flat" region, where efficiency can be improved at relatively low quality cost. We use this signal to dynamically set the loss weights ($\alpha_{\text{qual}}, \alpha_{\text{eff}}$).

$$\begin{aligned} \alpha_{\text{eff},i} &= \sigma\left(-k \cdot \phi_i + b'\right) \\ \alpha_{\text{qual},i} &= 1 - \alpha_{\text{eff},i}. \end{aligned} \tag{8}$$

This sigmoid function acts as a smooth switch. When the slope is high, the input to the sigmoid becomes a large negative value, driving α_{eff} towards 0. This forces the optimizer to prioritize quality. Conversely, when the slope is low, α_{eff} increases, encouraging the model to pursue higher efficiency. This leads to the final Pareto-balanced loss:

$$\mathcal{L}_{\text{pareto},i} = \alpha_{\text{qual},i} \cdot \mathcal{L}_{\text{qual},i} + \alpha_{\text{eff},i} \cdot \mathcal{L}_{\text{eff},i}. \tag{9}$$

3.3.3 Pruning Execution via Fisher Information

During Stage 2 training, once the retention ratio $r_{i,j}$ for step $s_{i,j}$ is determined, we perform token-level pruning within that step. Specifically, we retain the tokens that are most critical to the model's reasoning process. The importance of each token $t_{i,j,k}$ is approximated by the Fisher Information (Brunel and Nadal, 1998; Ohno, 2024), which we estimate using the squared norm of the gradient of the quality loss with respect to the token embedding $\mathbf{e}(t_{i,j,k})$ (Liu et al., 2021):

$$I(t_{i,j,k}) \approx \left\| \nabla_{\mathbf{e}(t_{i,j,k})} \mathcal{L}_{\text{qual},i} \right\|^2. \tag{10}$$

For each step $s_{i,j}$, we compute token importance scores, rank all tokens within the step, and retain the top $k_{i,j} = \lfloor L_{i,j} \cdot r_{i,j} \rfloor$ tokens to form the pruned step $s'_{i,j}$. This Fisher-guided pruning is used only during training to provide fine-grained token-level compression supervision; during inference, the learned compression behavior is internalized into the model parameters rather than executed through explicit pruning.

3.4 Staged Training Strategy

We train HAB via two-stage sequential fine-tuning with LoRA to reduce task interference between inter-step planning (Stage 1) and intra-step budgeting (Stage 2). Stage 1 learns a global skill, namely classifying the appropriate reasoning-depth category for the entire problem, and is optimized with the cross-entropy loss $\mathcal{L}_{\text{SCP}}$. Stage 2 then learns a local skill: estimating step-level retention ratios and performing Fisher-guided token pruning within each step to support adaptive budgeting. Training both stages jointly with a unified objective

could introduce destructive gradient interference, since global reasoning-depth classification and local token-level compression follow different learning dynamics. Therefore, we first adapt the model to predict reasoning depth, and then continue fine-tuning for intra-step adaptive budgeting using the combined objective $\lambda_1 \mathcal{L}_{\text{pair}} + \lambda_2 \mathcal{L}_{\text{pareto}}$.

4 Experiments

4.1 Baselines and Evaluation Metrics

We compare HAB against six representative methods spanning diverse strategies. (1) **Vanilla CoT (Zero-shot)** (Wei et al., 2022): the standard zero-shot CoT prompting baseline, which generates unconstrained reasoning paths and serves as our primary reference point for both performance and efficiency. **Pre-defined constraint methods**: (2) **TokenSkip** (Xia et al., 2025): prunes a fixed ratio of tokens from CoT chains and fine-tunes on compressed data. (3) **Chain of Draft (CoD)** (Xu et al., 2025a): instructs the model to generate minimalistic drafts for each step. (4) **Skeleton-of-Thought (SoT)** (Ning et al., 2023): first generates an answer skeleton, then expands each point in parallel. (5) **Sketch-of-Thought** (Aytes et al., 2025): uses a router to select a reasoning paradigm from a predefined library. **Learning-based constraint methods:** (6) **O1-Pruner** (Luo et al., 2025): an RL-based method using length-harmonizing reward within PPO to shorten reasoning paths. Following prior work (Han et al., 2025), we use two metrics: *Accuracy*, the percentage of correctly solved problems via exact match, and *Average output tokens*, the mean number of generated tokens per problem.

4.2 Implement Details and Datasets

Experiments are conducted on dual NVIDIA A800 GPUs (80GB each). We apply LoRA (Hu et al., 2022) fine-tuning to Qwen2.5-7B-Instruct[1] and Llama3.1-8B-Instruct[2], targeting attention projection layers (q_{proj}, k_{proj}, v_{proj}, o_{proj}) with rank 16, alpha 32, and dropout 0.1. For Stage 1 (inter-step), we train for 5 epochs with learning rate of 3×10^{-5}, batch size 4, gradient accumulation steps 2, and weighted cross-entropy loss to address class imbalance. For Stage 2 (intra-step), we train for 3 epochs with learning rate 2×10^{-5}, λ_1=0.03, λ_2=0.01, m=0.5, and δ=0.05. All models use AdamW with weight decay 0.01, gradient clipping 1.0, and maximum sequence length 512.

Datasets. Following prior work (Munkhbat et al., 2025; Xia et al., 2025), we evaluate on two mathematical reasoning benchmarks. (1) **MATH500** (Hendrycks et al., 2021): 500 high-school competition problems, split into 233/77/78 for train/val/test after excluding instances without a verifiable correct reasoning chain in Qwen-Max (see Section 3.2 or the corresponding subsection label). (2) **GSM8K** (Cobbe et al., 2021): grade-school math word problems; we randomly sample 2,000 instances and, after applying the same filtering procedure, remaining examples are split into 1007/434/462 for train/val/test.

4.3 Results and Discussions

Comparison with Baselines. Our experimental evaluation, summarized in Table 1 for the GSM8K dataset and Table 2 for the MATH500 dataset, provides a comprehensive comparison of HAB against a suite of SOTA baselines.

Table 1: Performance comparison on GSM8K dataset.

Method	Qwen2.5-7B-Instruct		Llama3.1-8B-Instruct	
	Acc (%)	Tokens	Acc (%)	Tokens
Vanilla CoT (Zero-Shot)	93.94	283.0	83.16	244.5
CoD	74.24	**44.2**	64.29	124.3
Skeleton-of-Thought	71.65	105.1	61.04	290.4
TokenSkip	90.04	165.9	81.17	124.3
Sketch-of-Thought	73.81	97.3	62.55	**112.8**
O1-Pruner	84.63	198.7	79.22	205.4
HAB (Ours)	**95.24**	211.1	**87.23**	232.2

Table 2: Performance comparison on MATH500 dataset.

Method	Qwen2.5-7B-Instruct		Llama3.1-8B-Instruct	
	Acc (%)	Tokens	Acc (%)	Tokens
Vanilla CoT (Zero-Shot)	79.49	482.2	46.15	572.9
CoD	43.59	**108.3**	25.64	365.5
Skeleton-of-Thought	47.44	135.3	39.74	529.7
TokenSkip	75.64	354.6	48.72	**282.2**
Sketch-of-Thought	52.56	237.3	35.9	418.5
O1-Pruner	69.23	350.8	47.44	381.7
HAB (Ours)	**82.05**	327.3	**51.28**	424.8

First, HAB achieves superior performance-efficiency trade-offs on both datasets. On GSM8K with Qwen2.5-7B-Instruct, HAB improves accuracy over Vanilla CoT while reducing token usage. This simultaneous improvement in both metrics distinguishes HAB from all baselines, which invariably sacrifice one for the other. **Second**, pre-defined constraint methods suffer severe accuracy degradation. For example, CoD shows substantial

[1]https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
[2]https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

drops on both GSM8K and MATH500, sacrificing a large amount of accuracy in exchange for aggressive token reduction, highlighting the limitations of uniform compression approaches. **Third**, O1-Pruner, representing RL-based approaches, underperforms HAB despite comparable token usage. This indicates that global budget optimization, even when learned through RL, cannot match HAB's hierarchical coarse-to-fine budgeting control. This suggests that global RL-based budget optimization is less effective than HAB's hierarchical budgeting in our setting. **Finally**, cross-model evaluation with Llama3.1-8B-Instruct shows that HAB remains effective across backbones. HAB consistently outperforms all baselines on both GSM8K and MATH500. Notably, HAB uses more tokens on MATH500 than on GSM8K, demonstrating its ability to adaptively allocate more computation to harder problems.

4.3.1 Broader Adaptability for HAB

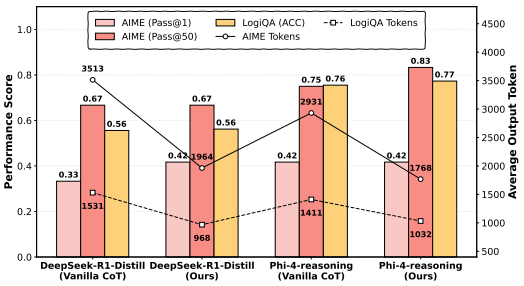


Figure 4: Broader adaptability of HAB across datasets and reasoning backbones.

To further demonstrate the robustness of HAB, we extend our evaluation to specialized reasoning-oriented models and high-difficulty benchmarks. The objective is to verify whether the "Thinking Economically" principle holds value for models already optimized for complex reasoning, which are still prone to "overthinking" and generating redundant computational overhead. We incorporate DeepSeek-R1-Distill-Qwen-7B[3] and Phi-4-reasoning[4] as backbones, testing them on the AIME (mixture of AIME 2025[5] and 2024[6], 15/5/12 for train/val/test after filter) and LogiQA (Liu et al., 2020) datasets (1172/381/331 for train/val/test after filter). As shown in Figure 4, HAB successfully maintains or even improves the accuracy of these advanced models while significantly reducing token usage. This demonstrates that even for models with strong inherent reasoning capabilities, HAB's hierarchical budgeting effectively identifies and prunes logical redundancies. These results confirm that our framework is a versatile paradigm applicable to the next generation of reasoning-heavy models and highly complex logical tasks.

[3]https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B

[4]https://huggingface.co/microsoft/Phi-4-reasoning

[5]https://huggingface.co/datasets/opencompass/AIME2025

[6]https://huggingface.co/datasets/Maxwell-Jia/AIME_2024

4.3.2 Ablation Studies

The ablation study demonstrates that HAB achieves optimal efficiency-performance balance through adaptive step prediction, a capability that fixed-step configurations fundamentally lack. While certain fixed configurations occasionally match or marginally exceed HAB's accuracy, they invariably do so at the expense of computational efficiency, validating our core principle of thinking economically. On GSM8K with Qwen2.5-7B-Instruct, the 1-step configuration achieves slightly higher accuracy than HAB. However, this marginal gain comes at a substantially higher token cost, indicating a much weaker efficiency-performance trade-off. Similarly, on MATH500 with Llama3.1-8B-Instruct, the 2-step configuration underperforms HAB while also consuming more tokens. These results highlight the inefficiency of static policies that cannot differentiate computational investment based on problem complexity.

Table 3: Impact of Fix Length parameter on our HAB's performance.

Fix Length_Type = 0				
Model	GSM8K		MATH500	
	Acc (%)	Tokens	Acc (%)	Tokens
Qwen2.5-7B-Instruct	95.02	172.6	79.21	309.5
Llama3.1-8B-Instruct	84.42	159.6	48.15	313.8
Fix Length_Type = 1				
Model	GSM8K		MATH500	
	Acc (%)	Tokens	Acc (%)	Tokens
Qwen2.5-7B-Instruct	95.89	310.9	81.77	496.6
Llama3.1-8B-Instruct	87.01	245.1	50.72	478.2
Fix Length_Type = 2				
Model	GSM8K		MATH500	
	Acc (%)	Tokens	Acc (%)	Tokens
Qwen2.5-7B-Instruct	95.45	316.3	80.49	514.7
Llama3.1-8B-Instruct	89.18	290.8	49.44	563.3

Furthermore, the experimental results reveal a critical insight: more reasoning steps do not monotonically improve performance. With Qwen2.5-7B-

Instruct on MATH500, accuracy decreases when moving from 1 step to 2 steps, despite increased token usage. This non-monotonic relationship suggests that excessive reasoning steps can introduce errors or logical inconsistencies, further illustrating the overthinking phenomenon. The model's best performance therefore requires not simply more steps, but an appropriate number of steps for each specific problem. HAB's adaptive mechanism naturally captures this complexity by learning to allocate reasoning budget based on hierarchical task requirements rather than applying uniform treatments. The framework achieves consistently strong performance across both datasets while maintaining computational efficiency, avoiding both the under-reasoning that can hurt complex problems and the over-reasoning that wastes resources on simpler ones.

5 Conclusions

We introduce Hierarchical Adaptive Budgeter (HAB), a novel framework that enables LLMs to "Think Economically" through adaptive reasoning resource allocation. HAB employs a coarse-to-fine budgeting process: it first predicts a coarse reasoning-depth category at the inter-step level and maps it to a step-range instruction, then allocates a dynamic token budget for each step at the intra-step level. The budget allocation is governed by an adaptive Pareto optimization scheme that balances reasoning quality and computational efficiency. Experiments on GSM8K and MATH500 demonstrate that HAB achieves a stronger performance-efficiency trade-off than the compared baselines, improving accuracy while reducing token usage compared to Vanilla CoT.

6 Limitations

Our work has several limitations that suggest directions for future research. First, HAB currently focuses on linear CoT structures. Extending the adaptive budgeting paradigm to non-linear reasoning frameworks such as Tree-of-Thoughts (Yao et al., 2023) and Graph-of-Thought (Besta et al., 2024) remains an open challenge. Second, while the adaptive Pareto optimization scheme is effective, it incurs modest additional training overhead. Moreover, HAB requires extra data preparation effort to construct the CoT dataset and derive reasoning-depth labels, which currently relies on external LLM-generated candidate chains and subsequent filtering. Developing more lightweight optimization and annotation strategies is therefore a promising direction.

7 Acknowledgements

This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-RP-2022-031), and A*STAR under its MTC Programmatic (Award M23L9b0052). This research is supported in part by the Singapore MOE AcRF Tier 1 funding (RG16/25).

References

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

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Nicolas Brunel and Jean-Pierre Nadal. 1998. Mutual information, fisher information, and population coding. *Neural computation*, 10(7):1731–1757.

Huanqia Cai, Yijun Yang, and Zhifeng Li. 2024. System-2 mathematical reasoning via enriched instruction tuning. *arXiv preprint arXiv:2412.16964*.

Qiguang Chen, Libo Qin, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. 2024a. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. *Advances in Neural Information Processing Systems*, 37:54872–54904.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024b. 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, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, and 1 others. 2025. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2502.13260*.

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*.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.

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. *arXiv preprint arXiv:2103.03874*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2025. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24312–24320.

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

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*.

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2665–2679.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. 2019. Pareto multi-task learning. *Advances in neural information processing systems*, 32.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. 2021. Group fisher pruning for practical network compression. In *International Conference on Machine Learning*, pages 7021–7032. PMLR.

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*.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*.

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

Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. 2023. Skeleton-of-thought: Prompting llms for efficient parallel generation. *arXiv preprint arXiv:2307.15337*.

Hiroshi Ohno. 2024. Adaptive pruning algorithm using a quantum fisher information matrix for parameterized quantum circuits. *Quantum Machine Intelligence*, 6(2):77.

Tiago Pimentel, Naomi Saphra, Adina Williams, and Ryan Cotterell. 2020. Pareto probing: Trading off accuracy for complexity. *arXiv preprint arXiv:2010.02180*.

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.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. 2025a. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*.

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

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Hanjie Chen, Xia Hu, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

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

Yaoting Wang, Shengqiong Wu, Yuecheng Zhang, Shuicheng Yan, Ziwei Liu, Jiebo Luo, and Hao Fei. 2025a. Multimodal chain-of-thought reasoning: A comprehensive survey. *arXiv preprint arXiv:2503.12605*.

Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. 2025b. 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.

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

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

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025b. Softcot: Soft chain-of-thought for efficient reasoning with llms. *arXiv preprint arXiv:2502.12134*.

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

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.

Maxwell J Yin, Dingyi Jiang, Yongbing Chen, Boyu Wang, and Charles Ling. 2025. Enhancing generalization in chain of thought reasoning for smaller models. *arXiv preprint arXiv:2501.09804*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

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

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. 2025b. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *arXiv preprint arXiv:2505.15778*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. In *The eleventh international conference on learning representations*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Han Zhou, Xingchen Wan, Ivan Vulić, and Anna Korhonen. 2024. Autopeft: Automatic configuration search for parameter-efficient fine-tuning. *Transactions of the Association for Computational Linguistics*, 12:525–542.

Xianwei Zhuang, Zhihong Zhu, Zhichang Wang, Xuxin Cheng, and Yuexian Zou. 2025. Unicott: A unified framework for structural chain-of-thought distillation. In *The Thirteenth International Conference on Learning Representations*.