Slim-SC: Thought Pruning for Efficient Scaling with Self-Consistency

HyScale

Colin Hong[1], Xu Guo[2], Anand Chaanan Singh[1], Esha Choukse[3], Dmitrii Ustiugov[1]

[1]NTU Singapore [2]KTH Royal Institute of Technology [3]Microsoft

The Challenge: Improving LLM Reasoning

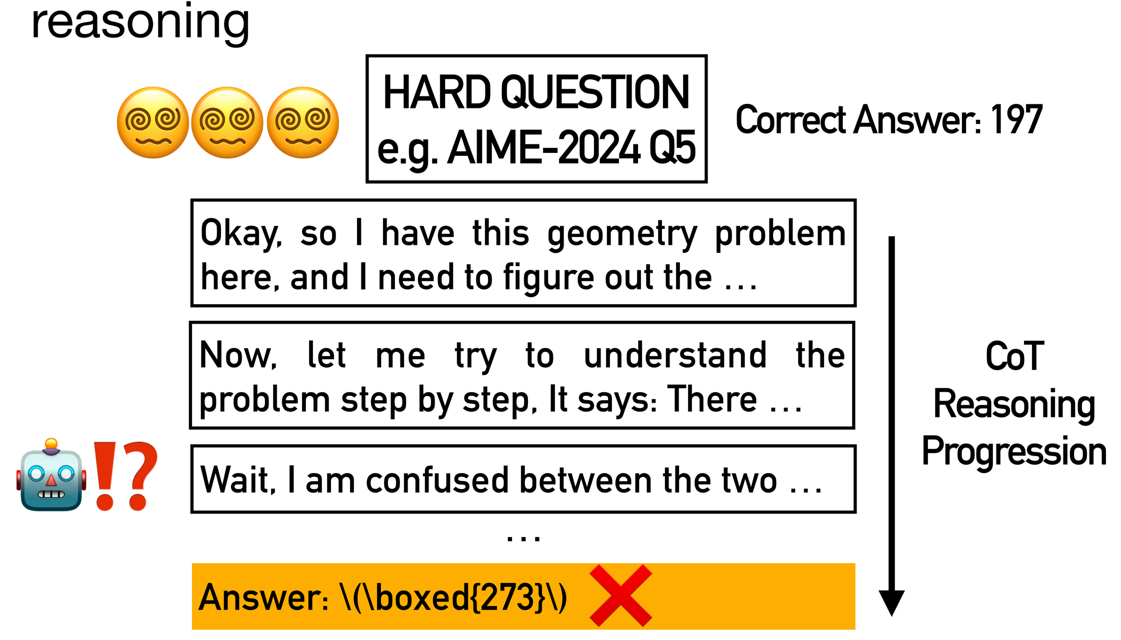
Challenging tasks require structured reasoning

Breakthrough: Chain-of-Thought (CoT) models

- Generate step-by-step intermediate thoughts
- Improves accuracy on hard problems

Limitation: CoT is brittle

- Limited perspectives from one reasoning chain
- One mistake in the chain can derail the entire reasoning



CoT improves reasoning, but its single-path approach is too brittle for complex problems

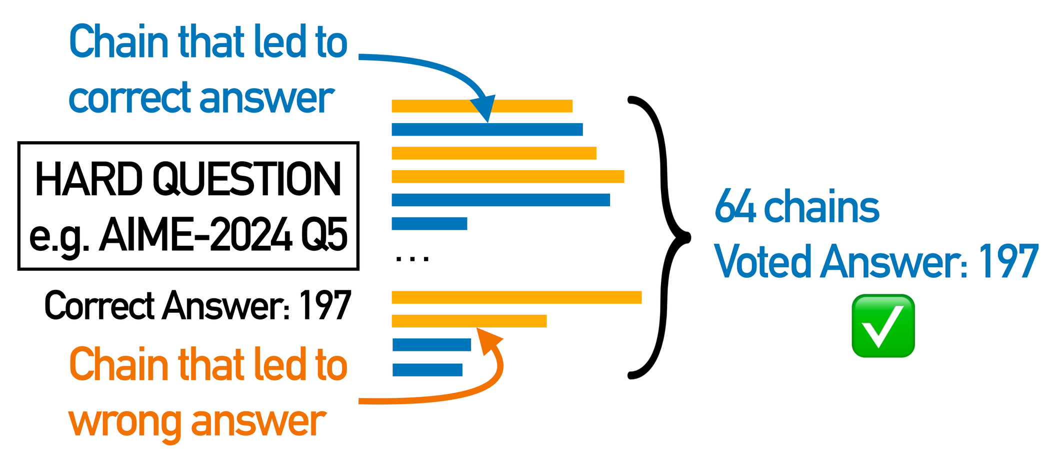
Self-Consistency: More Chains, More Power

Improvement: Test-Time Scaling (TTS)

- Boosts performance at inference time
- Uses more compute, but no model retraining

Self-Consistency (SC): A Powerful TTS Method

- Explore many paths to reduce CoT's brittleness
- **How it works:**
  1. Generate **many** reasoning chains **in parallel**
  2. Take a **majority vote** on the final answers



SC improves accuracy by exploring many reasoning paths, but this brute-force approach comes at a cost

SC is Expensive: Latency & Waste

Problem 1: Correct Chains May Be Outvoted

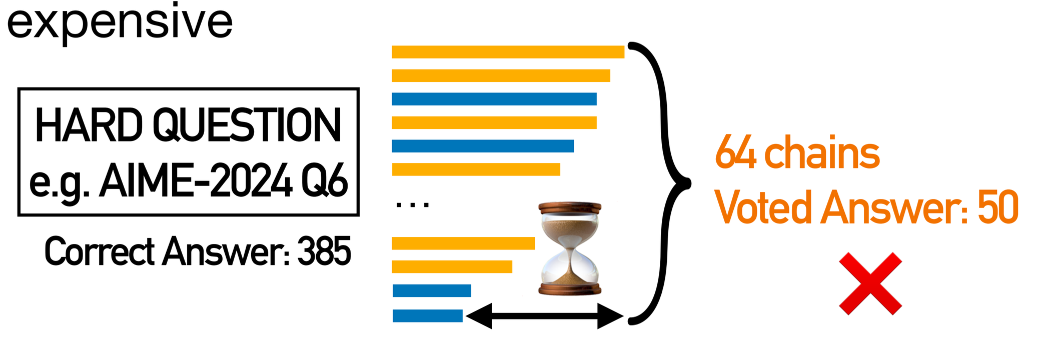
- Multiple incorrect chains may converge on the same wrong answer
- Wrong answers can dominate the majority vote

Problem 2: "Wait-for-all" Latency Bottleneck

- Voting happens after all chains are complete
- Long, incorrect "stragglers" block fast, correct chains

Problem 3: Massive Computational Cost

- Resources (latency, tokens, memory) scale linearly with chains
- Scaling SC to many chains is prohibitively expensive



SC's brute-force approach is wasteful, slow, and can still be outvoted by flawed logic

Insight: Semantic Clusters Enable Pruning

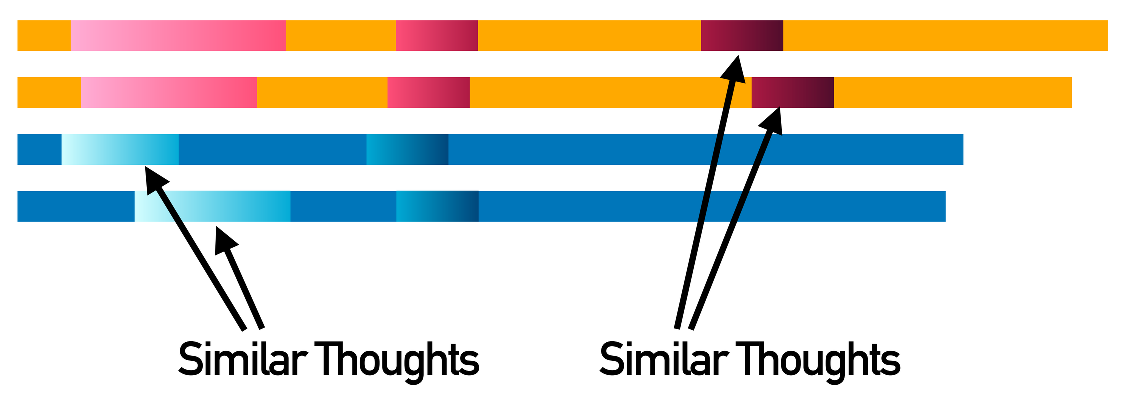
First, What is a "Thought"?

- A "thought" is a semantic unit of reasoning
- We segment chains using keywords LLMs use to pivot, such as: "Alternatively", "Another", "Wait"

Insight: These thoughts form semantic clusters

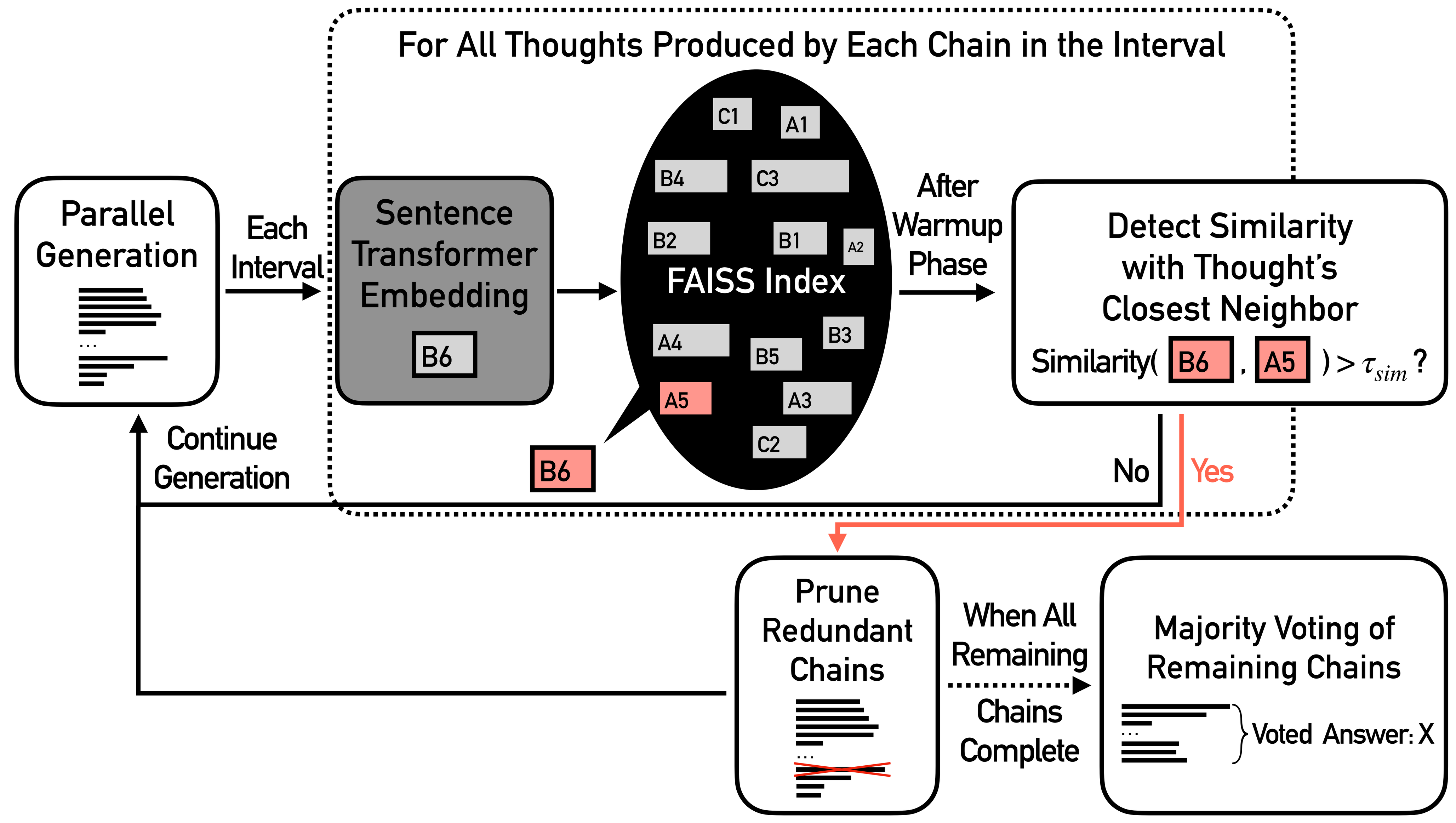
- Correct chains cluster around similar, valid logic
- Incorrect chains form separate, denser clusters around common flawed logic

Opportunity: Prune redundant chains within a cluster without harming reasoning diversity



Chains form distinct semantic clusters, making computational redundancy identifiable

Slim-SC: Step-wise Thought Pruning



Slim-SC operationalizes our insight by using semantic similarity to proactively prune redundant chains during generation

Faster Inference, Similar (or Better) Accuracy

Datasets: GPQA, AIME-2024, AQUA-RAT

Baselines: SC, ESC, CCoT-SC

- ESC (Early-Stopping SC):
  - Generates chains in sequential batches
  - Stops early if a consensus is reached
  - Saves tokens but introduces high latency as batches must wait for each other
- CCoT-SC (Concise CoT):
  - Adds "Be concise" to the SC prompt to try and shorten chains

Methods	Datasets & Metrics		
	GPQA-D	AIME'24	AQuA
	Accuracy (%)		
R1-Distill			
CoT	58.8±2.3	67.8±9.6	89.8±0.4
SC	63.0±0.6	**82.2**±1.9	**90.6**±0.4
ESC	62.0±0.8	81.1±1.9	89.5±0.5
CCoT-SC	60.4±1.5	80.0±3.3	89.9±0.2
Slim-SC	**63.5**±1.8	**82.2**±1.9	90.2

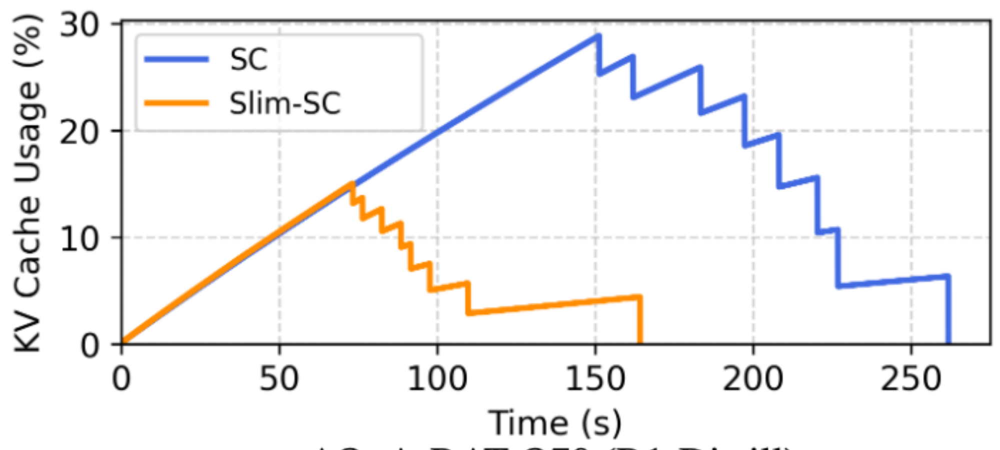
Up to 2.3x faster than ESC

Up to 1.4x faster than SC

Slim-SC achieves significant latency improvements, with minimal accuracy impact

Reduced GPU-time Costs

- By terminating redundant chains early, Slim-SC frees up GPU resources much sooner than SC
- This translates directly into significant, measurable savings in the latency and mean KV Cache usage



AQuA-RAT Q70 (R1-Distill)

Methods	Datasets & Metrics					
	GPQA-D	AIME'24	AQuA	GPQA-D	AIME'24	AQuA
	Latency (s)			Mean KVC usage (%)		
R1-Distill						
CoT	112±3	172±18	38±2	2	2	1
SC	536±16	942±29	61±3	47±2	56±2	4
ESC	876±17	1542±100	**51**±1	**10**	**13**	**1**
CCoT-SC	505±32	797±64	58±2	46±3	44±15	3
Slim-SC	**381**±126	**664**±139	56±1	43	49±8	3

Up to 25% less memory

Resource Savings:

- 29% faster than SC on GPQA and AIME-2024
- Up to 25% reduction in mean KV Cache usage vs SC
- In contrast, SC holds memory until the very last straggler finishes

Slim-SC's proactive pruning delivers a double win with lower latency and lower memory usage

Takeaways

SC is highly accurate but inefficient due to long chains & redundant computation

Key insight: Chains cluster, which the system can identify and prune

Slim-SC proactively prunes redundant chains with a minimal overhead

Key advantages:

- **Lower Latency & Cost:** Reduces the latency and GPU-time cost by up to 45%
- **Robust Accuracy:** Closely matches or exceeds SC's accuracy
- **Production-Ready:** Easy to integrate into popular frameworks (e.g. vLLM)

Code & Paper

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

KTH VETENSKAP OCH KONST

Microsoft