SEED: Accelerating Reasoning Tree Construction via Scheduled Speculative Decoding

COLING 2025 • Abu Dhabi

Zhenglin Wang*, **Jialong Wu***,
Yilong Lai, Congzhi Zhang, Deyu Zhou(✉)

School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, China

Introduction

Large Language Models (LLMs) demonstrate remarkable emergent abilities across various tasks, yet fall short of complex reasoning and planning tasks. The tree-search-based reasoning methods address this by encouraging the exploration of intermediate steps, surpassing the capabilities of chain-of-thought prompting. However, significant inference latency is introduced due to the systematic exploration and evaluation of multiple thought paths.

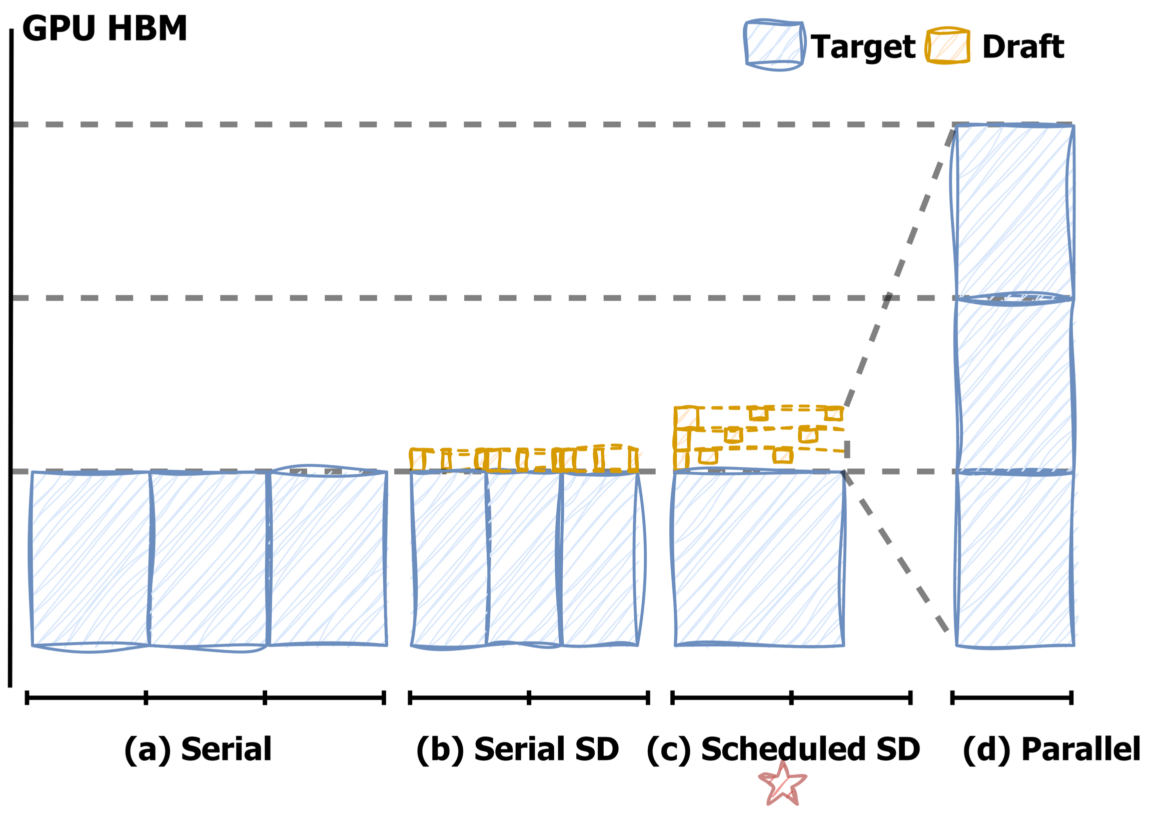

Figure 1. (a) Serial, where executions are operated one after another, simplifying resource management but increasing overall execution time; (b) Seiral SD, where speculative decoding is used for each execution; (c) Scheduled SD, which involves several parallel draft models and one target model; (d) Parallel, where multiple executions run concurrently, reducing completion time but increasing GPU HBM. ⎵ represents `a unit length` of execution time.

Therefore, we propose SEED, a novel and efficient inference framework that utilizes SchEduled spEculative Decoding to manage the scheduling of parallel draft models, to address both runtime speed and GPU memory resource management concurrently in reasoning tree construction.

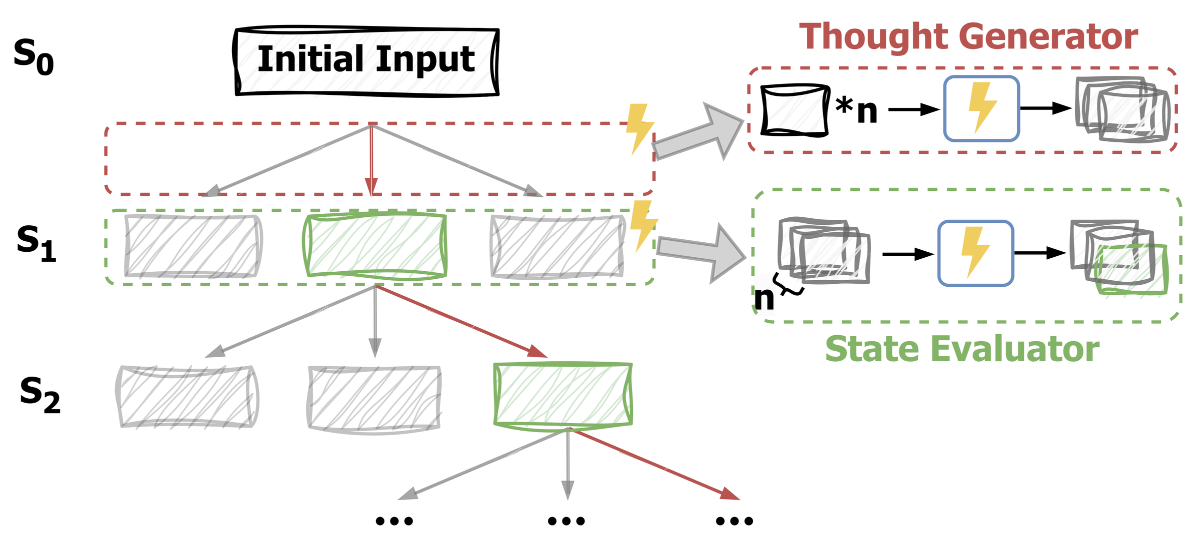

Figure 2. Two main components in reasoning tree construction, which are Thought Generator and State Evaluator, respectively.

SEED effectively handles two scenarios: (1) executing multiple iterations with the same prompt; (2) evaluating multiple iterations with different prompts.

Inspired by Operating System Scheduling

We utilize scheduled speculative decoding to manage the scheduling of parallel draft models. As depicted in Figure 1 (c), given that there is only one shared target model, which can not simultaneously verify multiple draft models, we address this limitation by drawing inspiration from process scheduling in operating system management.

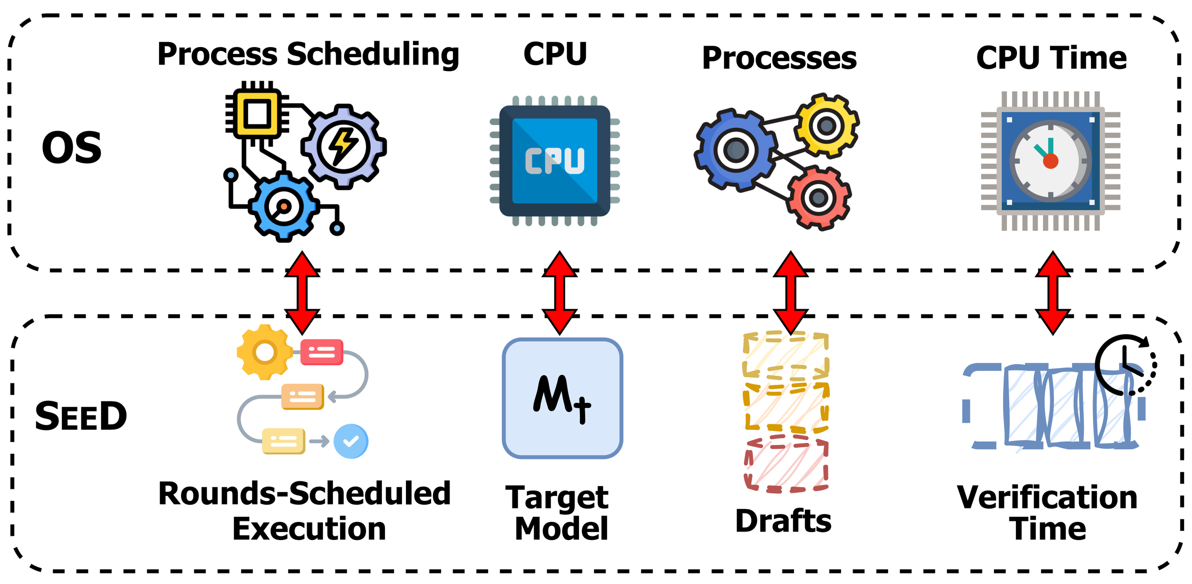

Figure 3. Analogy between the Operation System scheduler with our proposed SEED.

As shown in Figure 3, each component in SEED can be mapped to a corresponding component in the operating system scheduler. We elaborate on each component individually as below:

- The rounds-scheduled execution in SEED corresponds to the process scheduling in OS. Both use an FCFS queue to control and maintain the overall execution flow. A key distinction exists: in SEED, after the drafting tokens are processed by the verification phase, the draft model is returned to the queue, *i.e.*, "*rounds*". In contrast, in OS scheduling, a process that has been handled by the CPU is marked as completed.
- The verification of draft tokens mirrors an operating process in OS scheduling.
- The target model serves M_t analogously to the CPU.
- The total verification time of M_t resembles the CPU time in OS process scheduling.

The framework of SEED

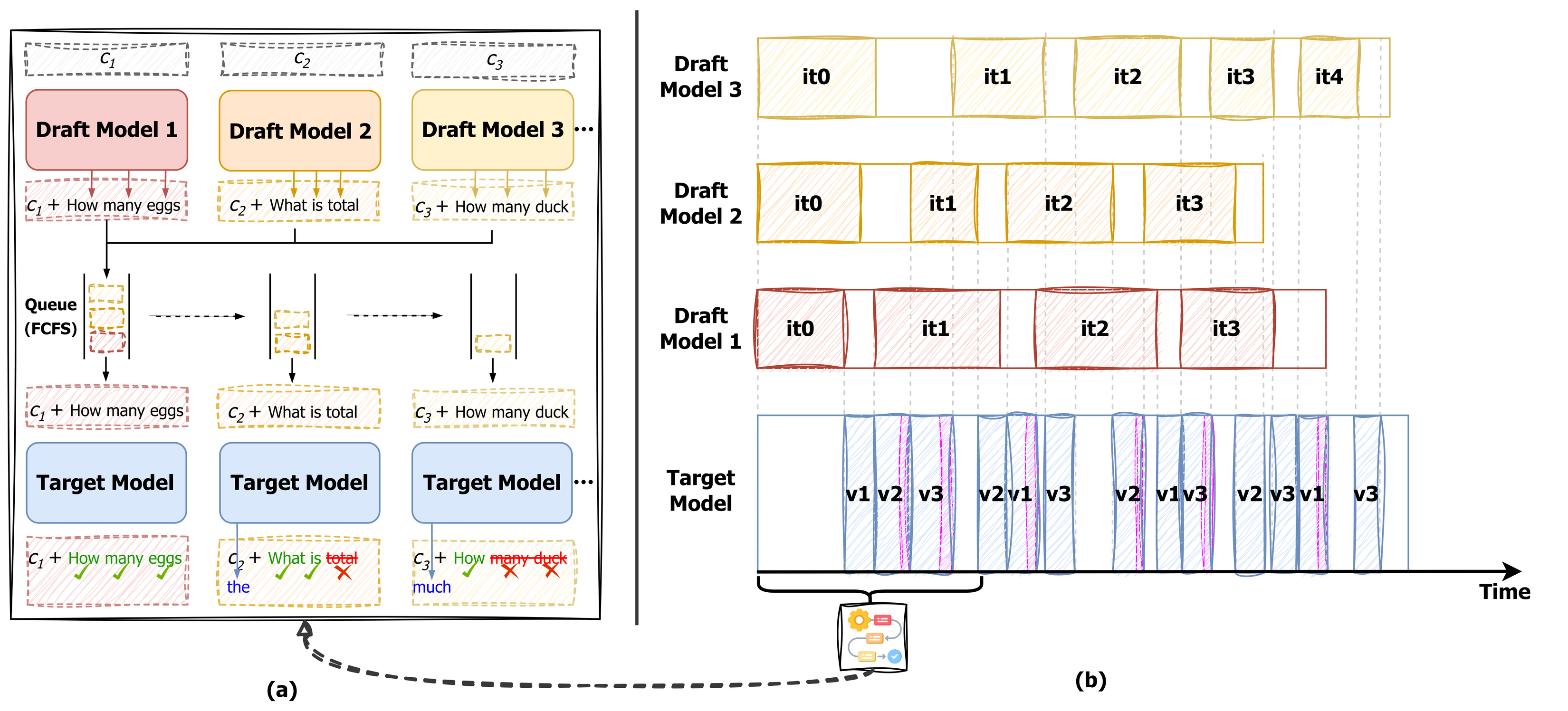

Figure 4. (a) The scenario where the target model manages the verification of target models at the beginning; (b) Overall scheduling diagram for one target model and three draft models. ▭, ▭, ▭ represent Draft Model 1, Draft Model 2, Draft Model 3, respectively. ▯, ▯, ▯ denotes the execution times of drafting for each corresponding draft model. ▭ refers to Target Model. ▯ represents the execution time of the verification phase, while ▯ specifies the resampling time in cases of rejection.

Parallel Drafting Phase: Each draft model generates its own tokens while the target model M_t verifies the tokens of other draft models.

Sequential Verification Phase: The target model first verifies the tokens generated by the draft model at the front of the queue.

Rounds-Scheduled Strategy: We leverage a Rounds-Scheduled Strategy integrated with the FCFS scheduling policy to manage the verification process efficiently. When a draft model completes its drafting phase and is ready for verification, the draft sequences along with c are placed into a queue.

Experimental Result

Extensive experiments and analysis studies are conducted to demonstrate the effectiveness of SEED. SEED achieves 1.1–1.5× speedups, generating up to 20 additional tokens per second across three reasoning datasets.

Temp.	k_{config}	Methods	CW($\mathcal{T}=2$) Tokens/s	CW($\mathcal{T}=2$) Speedup	GSM8K($\mathcal{T}=4$) Tokens/s	GSM8K($\mathcal{T}=4$) Speedup	BW($\mathcal{T}=7$) Tokens/s	BW($\mathcal{T}=7$) Speedup
0.2	-	AR	38.42	1.000×	42.31	1.000×	34.19	1.000×
	(1,1,1)	SD	39.96	1.040×	51.11	1.208×	36.28	1.061×
		w. SEED	**41.53**	**1.081×**	**53.14**	**1.256×**	**36.93**	**1.080×**
		MCSD	40.19	1.046×	52.42	1.239×	36.04	1.054×
		w. SEED	**41.46**	**1.079×**	**53.78**	**1.271×**	**36.96**	**1.081×**
	(2,2,1)	SD	46.22	1.203×	60.63	1.433×	40.04	1.171×
		w. SEED	**48.60**	**1.265×**	**65.24**	**1.542×**	**44.24**	**1.294×**
		MCSD	46.80	1.218×	60.88	1.439×	40.79	1.193×
		w. SEED	**48.79**	**1.270×**	**65.58**	**1.550×**	**44.75**	**1.309×**
1.0	-	AR	39.47	1.000×	47.81	1.000×	34.62	1.000×
	(1,1,1)	SD	45.90	1.163×	55.32	1.157×	35.14	1.015×
		w. SEED	**46.77**	**1.185×**	**61.01**	**1.276×**	**38.94**	**1.125×**
		MCSD	45.63	1.156×	58.47	1.223×	38.05	1.099×
		w. SEED	**46.54**	**1.179×**	**65.50**	**1.370×**	**40.02**	**1.156×**
	(2,2,1)	SD	57.39	1.454×	66.74	1.396×	45.98	1.328×
		w. SEED	**58.89**	**1.492×**	**72.62**	**1.519×**	**47.22**	**1.364×**
		MCSD	56.24	1.425×	67.36	1.409×	46.18	1.334×
		w. SEED	**59.76**	**1.514×**	**74.44**	**1.557×**	**47.71**	**1.378×**

Table 1. The speedup performance of our proposed SEED and baselines, with settings of SEED for M_d and M_t being LLaMA-68M and LLaMA2-7B, respectively.

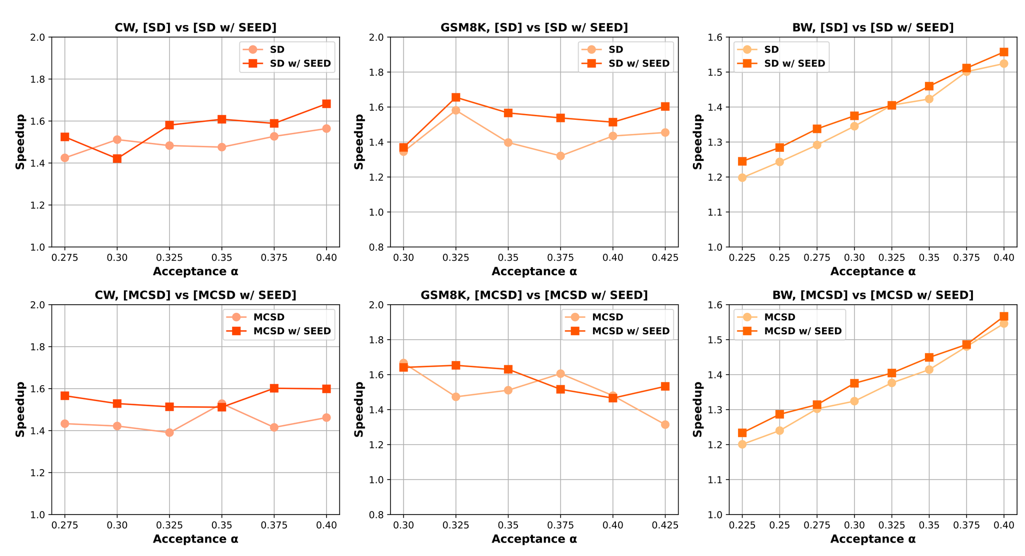

Figure 5. The variation of speedup performance across three datasets at different acceptance rates α.

More Details

More details can be found in our paper and code below:

arXiv GitHub

 {zhenglin,jialongwu}@seu.edu.cn