# Confidence-Weighted Token Set Cover for Early Hypothesis Pruning in Self-Consistency

**Md Arafat Sultan**[◇] **Ramón Fernandez Astudillo**
IBM Research AI
[◇]arafat.sultan@ibm.com

## Abstract

Despite its simplicity and efficacy, the high token expenditure of self-consistency can limit its practical utility. We investigate whether early hypothesis pruning can improve the token efficiency of self-consistency for long chain-of-thought reasoning tasks, *while preserving its parallelism*. Concretely, we generate all solutions in parallel but periodically prune intermediate hypotheses based on two lightweight indicators: (*a*) the model's confidence in each hypothesis, and (*b*) the lexical coverage of all current hypotheses by candidate subsets. We design a fast weighted set cover algorithm that utilizes the two indicators; evaluation of five LLMs on three math benchmarks shows that our method improves token efficiency in most cases, with reductions of $10 - 35\%$ in many.

## 1 Self-Consistency and Efficiency

Self-consistency (Wang et al., 2023), also known as majority voting, generates multiple solutions to a given problem and selects the final answer that appears most frequently among them. A highly effective test-time scaling strategy for weak and strong models alike on a variety of reasoning tasks (Touvron et al., 2023; DeepSeek-AI, 2025; Mistral-AI, 2025), it is also in many ways a lightweight and low-cost method – unlike best-of-$N$ (BoN) sampling or beam search, it does not need a separate scoring model, for example. Yet, generating a large number of samples can itself be computationally expensive, especially with modern large language models (LLMs).

Existing approaches to sample-efficient self-consistency are predominantly recurrent in nature: given a sample budget $N$, a small batch of $M << N$ complete solutions is generated at a time; the process is repeated until strong consensus is observed among already generated samples, or in the worst case, the budget has been fully spent (Aggarwal et al., 2023; Li et al., 2024). Later work has incorporated utilities of the generated samples into the process, sourced either from the generator's own confidence scores (Wang et al., 2024; Huang et al., 2025) or from an external scorer, such as a feature-based classifier (Wan et al., 2025) or a reward model (Astudillo et al., 2025). Despite improving sample efficiency, the sequentiality of these methods remains their clear drawback, as turnaround time increases linearly with the number of batches.

Given today's GPU-powered, highly parallelized LLM compute environments, here we are interested in token-efficient self-consistency that better retains parallelism. Our specific context is that of reasoning tasks such as math problem solving, where solutions take the form of a relatively long chain of thought (CoT) followed by a short final answer. Concretely, we consider an approach where all solutions are still generated in parallel as in standard self-consistency, but hypotheses – sequences of generated tokens – unlikely to add to the predictive power of the cohort are pruned early. Similar strategies have been used to prune low-scoring hypotheses for efficient BoN sampling, *e.g.*, by Sun et al. (2024) and Wang et al. (2025b). Self-consistency poses different challenges than BoN, however, as an entire subset of hypotheses must be maintained through the end that are not only of high quality individually, but also diverse (Lau et al., 2024; Wang et al., 2025a) and sufficiently exploratory of the space of solutions collectively.

The key challenge in our approach lies in identifying the prunable hypotheses: Given a set $Y_t$ of parallel hypotheses for an input $x$ at timestep $t$, what subset $Y_t^* \subset Y_t$ should be retained for further expansion so that the rest can be safely pruned? We posit that a combination of two complementary properties can make strong $Y_t^*$ candidates: (*i*) Individual hypotheses in $Y_t^*$ are of high quality, and (*ii*) $Y_t^*$ retains the full diversity of "thoughts" verbalized across all hypotheses in $Y_t$. Using com-



*Findings of the Association for Computational Linguistics: ACL 2026*, pages 41148–41155

**Algorithm 1:** WEIGHTEDSETCOVER

**Input** : Universe $U$ of individual items, list of sets of items $S = [S_i]_{i=1}^{N}$ and their weights $[w_i]_{i=1}^{N}$
**Output** : Weighted set cover $C \subset S$ of $U$ (approx.)

$PQ \leftarrow$ An empty min-priority queue
**for** $S_i \in S$ **do**
  Enqueue $S_i$ into $PQ$ with priority $\frac{w_i}{|S_i|}$
$C \leftarrow \varnothing$ // *the cover (set of sets from $S$)*
$Covered \leftarrow \varnothing$ // *set of items from $U$*
**while** $Covered \neq U$ **do**
  $S^* \leftarrow$ Dequeue$(PQ)$
  **if** $S^* - Covered \neq \varnothing$ **then**
    $C \leftarrow C \cup \{S^*\}$ // *add $S^*$ to $C$*
    $Covered \leftarrow Covered \cup S^*$ // *new items*
**return** $C$

putationally inexpensive proxies to quantify the underlying attributes – the LLM's own token-level probabilities for individual hypothesis quality and the lexical coverage of $Y_t$ by $Y_t^*$ for group diversity – we propose an approximate confidence-weighted token set cover algorithm for subset selection (§2) that allows us to examine our proposition.

Experimental results (§3) show that the proposed method can improve token efficiency for five different LLMs in the 1.5B-14B parameters range from the Qwen2.5-Math (Yang et al., 2024), Granite3.3[1] and Phi4 (Abdin et al., 2024) families on multiple math benchmarks, often in the $10-35\%$ range for $32-64$ samples. We also report ablation results validating the individual utility of hypothesis quality and diversity, and uncover model attributes that dictate the efficacy of the method by analyzing the outcome of its individual steps.

## 2 Method

At a high level, our proposed method operates by generating multiple parallel hypotheses in a step-by-step fashion and pruning a subset of those after every step, as shown in Algorithm 2. The input to the algorithm is a question $x$, an LLM $\theta$ to answer it with, the sample budget $N$, and a schedule $\breve{C}$ that dictates the step size, *i.e.*, the number of tokens to generate at every step. For example, the step size can be kept fixed throughout execution or be gradually lowered to increase the frequency of pruning at later stages. Each iteration of the algorithm (line 3) consists of first generating $ss$ (step size) next tokens for each surviving incomplete hypothesis in $Y_t$, where $t$ is the current number of tokens per hypothesis (lines 4–8), and then identifying a subset $Y_t^* \subset Y_t$ to retain for the next iteration (lines

**Algorithm 2:** SELFCONSISTENCYWITHPRUNING

**Input** : Question $x$, LLM $\theta$, sample budget $N$, step size schedule $\breve{C}$
**Output** : Answer $a$

1 $t \leftarrow 0$
2 $Y_t \leftarrow [\varepsilon]_{i=1}^{N}$ // *list of $N$ empty strings*
3 **while** *$Y_t$ contains incomplete hypotheses* **do**
4   $ss \leftsquigarrow \breve{C}$ // *get step size for current iteration*
5   **for** *$i$ in $1:N$* **do**
6     **if** *$y_i$ is incomplete* **then**
7       $y_{i,t+1:t+ss} \sim P_\theta(\cdot \mid x, y_{i,\le t}, ss)$
8   $t \leftarrow t + ss$
9   **for** *$i$ in $1:N$* **do**
10     $S_i \leftarrow$ `unique-tokens`$(y_{i,1:t})$
11     $conf_i \leftarrow e^{\frac{1}{t}\sum_{j=1}^{t} \log P_\theta(y_{i,j} \mid x_i, y_{i,<j})}$
12     $w_i \leftarrow 1 - conf_i$
13   $U \leftarrow \bigcup_{i=1}^{N} S_i$
14   $keep \leftarrow$ WEIGHTEDSETCOVER$(U, [S_i]_{i=1}^{N}, [w_i]_{i=1}^{N})$
15   $Y_t^* \leftarrow [y_i \mid (y_i \in Y_t) \wedge (S_i \in keep)]$
16   $N \leftarrow$ `num-elements`$(Y_t^*)$
17   $Y_t \leftarrow Y_t^*$
18 **for** $y_i \in Y_t^*$ **do**
19   $a_i \leftarrow$ `extract-final-answer`$(y_i)$
20 $a \leftarrow$ `majority-element`$([a_i]_{i=1}^{N})$
21 **return** $a$

9–15). The loop terminates when all hypotheses in $Y_t$ are complete, *i.e.*, contain a final answer. A majority voting is then performed on the final answers extracted from all solutions in $Y_t$ and the most frequent answer is returned (lines 18–21).

The subset $Y_t^*$ is identified using a weighted set cover algorithm that is given three inputs: the set $U$ of all unique tokens across all current hypotheses in $Y_t$, a list $S = [S_i]_{i=1}^{N}$ where $S_i$ is the set of unique tokens in hypothesis $y_i \in Y_t$, and the weight $w_i$ of $S_i$ (line 14). $w_i$ is computed as $1 - conf_i$, where $conf_i \in [0, 1]$ is the model's length-normalized aggregate confidence in $y_i$ (lines 11, 12). Algorithm 1 details our fast approximate set cover implementation. All token sets $S_i \in S$ are first inserted into a min-priority queue $PQ$ according to both their coverage $|S_i|$ of $U$ and $w_i$. With a binary heap implementation, populating $PQ$ takes $O(N)$ time. Token sets are then dequeued from $PQ$ one at a time and added to the cover $C$ if they contain items not in $Covered$, the set of already covered tokens. The loop terminates as soon as $Covered = U$, running in $O(N \log N)$ worst-case time. This method essentially prefers to have hypotheses with jointly high confidence and coverage scores in $C$ provided they add to the ongoing coverage of $U$.

[1] https://huggingface.co/ibm-granite/granite-3.3-8b-instruct

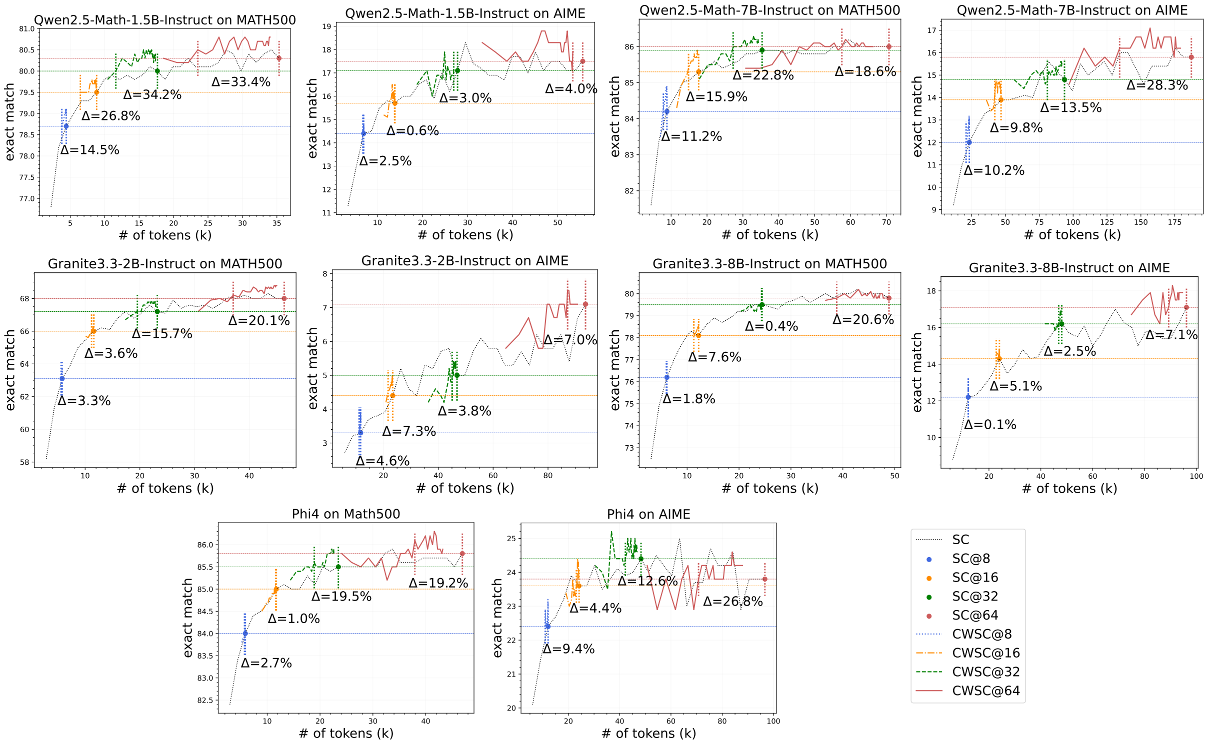

Figure 1: Token savings from early pruning. The four short colored polylines in each plot correspond to hypothesis pruning at sample budgets $N = 8, 16, 32, 64$. Each polyline depicts exact match (EM) as step size (and consequently, the number of tokens generated) increases. For each, $\Delta$ represents the $\%$ of tokens saved when it passes the corresponding majority voting EM without pruning (horizontal baseline) and subsequently never drops below it. The $\Delta$ values are also summarized in Table 6 (Appendix E). The long black dotted polyline depicts ordinary majority voting EM at different sample budgets $N \in [4, 64]$, and crucially, stays mostly below the early pruning polylines.

## 3 Experiments

**Setup:** We evaluate five different LLMs of various sizes on three public math benchmarks of varying difficulty. The LLMs include Alibaba's math-specific Qwen2.5-Math-Instruct with 1.5B and 7B parameters, IBM's general-purpose Granite3.3-Instruct with 2B and 8B parameters that have strong math performance for their size and class, and Microsoft's Phi4 with 14B parameters as our largest model. Our benchmarks are MATH500 (Hendrycks et al., 2021; Lightman et al., 2024) with 500 test problems, and AIME24[2] and AIME25[3] with 30 problems each. We merge the two AIME sets into one and report results on the merged set. All model predictions are evaluated against ground truth answers using exact match (EM) via a custom fork of `math-evaluation-harness`[4].

We report results for sample budgets $N = 8$, 16, 32 and 64. For each problem, we generate a total of 256 samples with each LLM (temperature 1.0, top-$p$ 0.95) from 16 different seeds so that evaluation scores can be averaged over multiple runs, *e.g.*, 16 runs for $N = 16$ and 4 runs for $N = 64$. See Appendix B for length statistics. Finally, we experiment with two step size schedules $\breve{C}$. The first maintains a fixed step size $ss$ throughout execution. The other starts with a pre-specified initial value and cuts it by half after every step as more hypotheses get pruned, until $ss$ reaches a floor (8 in our experiments). The main idea behind this dynamic schedule is to prune more frequently during later rounds of generation as hypotheses get longer and more representative of their final forms.

**Results:** Figure 1 shows our main results for the dynamic step size schedule. Each plot depicts the accuracy of the proposed confidence-weighted set cover method (CWSC) and of ordinary self-consistency with no pruning (SC) for a unique model-test set pair, with sample budgets $N = 8, 16, 32, 64$. For each value of $N$, we run CWSC with different initial step sizes ranging from 64 to 512, shown as a short colored polyline in the plots. It should be noted here that smaller step sizes gener-

[2] https://huggingface.co/datasets/math-ai/aime24
[3] https://huggingface.co/datasets/math-ai/aime25
[4] https://github.com/ZubinGou/math-evaluation-harness

**MATH500**

| | CWSC@16 | AC@16 | CWSC@32 | AC@32 |
|---|---|---|---|---|
| Qwen2.5-Math-1.5B-Instruct | (73.2%, 96.0%) | (54.5%, 581.5%) | (65.8%, 94.6%) | (40.3%, 774.4%) |
| Qwen2.5-Math-7B-Instruct | (84.1%, 96.7%) | (67.3%, 513.4%) | (77.2%, 95.5%) | (53.0%, 713.1%) |
| Granite3.3-2B-Instruct | (96.4%, 97.7%) | (78.6%, 468.0%) | (84.3%, 91.7%) | (65.6%, 595.3%) |
| Granite3.3-8B-Instruct | (92.4%, 97.9%) | (64.8%, 541.3%) | (99.6%, 99.9%) | (50.3%, 671.9%) |
| Phi4 | (99.0%, 99.4%) | (55.2%, 369.7%) | (80.5%, 92.1%) | (40.3%, 426.8%) |

**AIME**

| | CWSC@16 | AC@16 | CWSC@32 | AC@32 |
|---|---|---|---|---|
| Qwen2.5-Math-1.5B-Instruct | (91.1%, 97.6%) | (93.2%, 903.9%) | (97.0%, 99.1%) | (84.8%, 1473.3%) |
| Qwen2.5-Math-7B-Instruct | (90.2%, 100.0%) | (97.8%, 1174.0%) | (86.5%, 100.0%) | (93.5%, 2228.9%) |
| Granite3.3-2B-Instruct | (92.7%, 97.3%) | (98.7%, 591.5%) | (96.2%, 98.9%) | (93.8%, 899.0%) |
| Granite3.3-8B-Instruct | (94.9%, 98.5%) | (97.5%, 633.7%) | (97.5%, 99.6%) | (90.9%, 938.7%) |
| Phi4 | (95.6%, 98.1%) | (89.5%, 509.6%) | (87.4%, 96.0%) | (80.2%, 721.2%) |

Table 1: Trade-offs between CWSC (proposed) and adaptive-consistency (AC). Each pair $(p_1, p_2)$ consists of the following two quantities: $p_1$ = token expenditure as a % of that of self-consistency (SC); $p_2$ = number of tokens generated sequentially, as a % of the same in SC. While AC exhibits the best token efficiency, it results in a marked increase in the number of sequential LLM calls. CWSC reduces token expenditure without sacrificing parallelism.

ally lead to more aggressive early hypothesis pruning – as groups of short solution prefixes tend to be more homogeneous than longer ones – resulting in high token efficiency but low accuracy. We use the symbol $\Delta$ and two parallel vertical dotted lines in the plots to show token savings, which is computed as the % by which CWSC reduces token count over SC. This is determined at the point where CWSC's EM score surpasses that of SC and subsequently never falls behind. CWSC almost always yields token savings, by a sizable $10-35\%$ in many cases. Savings are understandably higher at larger values of $N$, but are non-negligible ($> 5\%$) even in many lower-budget settings.

The long black dotted polyline in each plot shows the accuracy of SC for different token counts, corresponding to different values of $N \in [4, 64]$. Importantly, more often than not, and particularly within the $\Delta$ range, the CWSC polylines stay above this SC polyline for an equal number of generated tokens, indicating a degree of robustness to the selection of step size.

In Appendix A, we report similar gains under a fixed step size schedule $\breve{C}$, further demonstrating the robustness of CWSC to the choice of schedule.

***Trade-offs with sequential methods.*** To examine the trade-offs of CWSC relative to a sequential approach to token-efficient SC, we also evaluate Adaptive-Consistency (AC) (Aggarwal et al., 2023), which generates solutions one at a time until a sufficiently strong consensus is reached. Table 1 compares AC and CWSC, where each cell reports a pair of percentage values $(p_1, p_2)$ for the respective method. $p_1$ is the method's token expenditure as a % of that of vanilla SC, where AC clearly outperforms. AC has zero parallelism, however, which naturally increases the number of sequential LLM calls made (and therefore latency). This latter effect is captured in $p_2$, which is the number of tokens in the longest sequentially generated chain, as a % of the same in SC. For SC and CWSC, the numerator is simply the token count of the longest solution. For AC, it is the sum of token counts over all generated solutions. As the results clearly show, AC's overall higher token efficiency comes at the cost of a many-fold increase in the number sequential LLM calls, whereas CWSC reduces token expenditure without sacrificing parallelism.

| Method 1 | Method 2 | Winner | Win % |
|---|---|---|---|
| CWSC | USC | CWSC | 82.5 |
| CWSC | CW | CWSC | 65.0 |
| USC | CW | CW | 62.5 |

Table 2: Ablation results. CWSC: confidence-weighted set cover (proposed); USC: unweighted set cover; CW: confidence weighting with no set cover. Both confidence weighting and set cover contribute to the overall performance of CWSC (rows 1, 2), with the former playing a more important role (rows 1, 2, 3).

***Ablation.*** To gauge the individual importance of confidence weighting and token set cover for the algorithm, we ablate each and evaluate the resulting method. For *set cover only*, we assign a weight of 1 to all hypotheses uniformly in line 12 of Algorithm 2. For *confidence weighting only*, we substitute the set $keep$ in line 14 with the same number of highest-confidence hypotheses.

Across our 5 LLMs, 2 test sets and 4 different sample budgets, we have a total of 40 test settings.

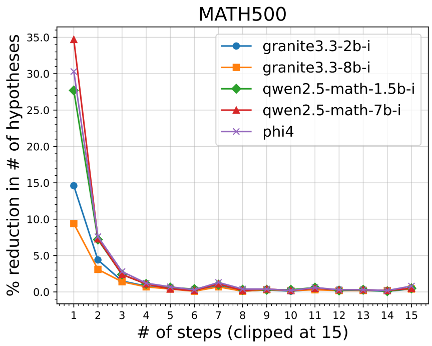


Figure 2: % reduction in the number of hypotheses as generation progresses for MATH500 (initial step size = 256). More hypotheses are expectedly pruned in earlier steps; % reduction saturates near a small but non-zero value after ~6 steps. Qwen and Phi4 generations are pruned more aggressively as individual hypotheses provide greater lexical coverage of the cohort (§3). AIME results are quite similar; c.f. Appendix D.

By first averaging over various amounts of tokens generated within each setting, and then further averaging over all 40 settings, we compare the performance of different methods in Table 2. Row 1, for example, shows that the full method (CWSC) outperforms *set cover only* (USC) in 82.5% of all settings. Rows 1 and 2 validate the need for both components in the algorithm, while all rows together point to confidence weighting as the more important component. See also Appendix C for random pruning results.

***A step-by-step look.*** To further examine the inner working of our method, we take a closer look at the outcome of its individual steps with an initial step size of 256. First, as Figure 2 shows for MATH500, the rate of pruning is expectedly higher in earlier steps of inference, falling gradually over time – as many prunable items have already been removed – eventually plateauing near a small non-zero value after about 6 steps. See also Appendix D.

Second, we observe slightly different outcomes for different models: Qwen and Phi4 hypotheses get pruned more aggressively than those of Granite, for instance. To examine why, we inspect the content of the first 256 tokens generated by all models with $N = 64$. We find that individual hypotheses from Qwen and Phi4 provide a higher token coverage of the group of all 64 hypotheses compared to Granite. For instance, while an average Phi4 solution prefix of 256 tokens contains 26.5% of all unique tokens in all 64 solution prefixes combined, the corresponding number for Granite3.3-8B-Instruct is only 15.2%. Thus, when a model's individual hypotheses are lexically more unique within their cohort, the algorithm has to naturally retain more of them to ensure full coverage.

## 4 Conclusion

We show that token expenditure in self-consistency can be reduced without sacrificing parallelism, by computing a confidence-weighted token set cover over partial hypotheses. Future work could explore methods for learning optimal step size schedules tailored to specific model-problem pairs. Applying the proposed method to related tasks, such as code generation, is another natural direction.

## Limitations

A primary limitation of the proposed method is that it expects a step size schedule as input, which could in principle be estimated using validation data from a diverse set of domains. While we report extensive evaluation and empirical analysis on the important domain of mathematical problem solving, other long-CoT reasoning tasks such as code generation may also benefit from our approach, which we did not explore in this paper.

## References

Marah Abdin, Jyoti Aneja, and 1 others. 2024. Phi-4 Technical Report. *Preprint*, arXiv:2412.08905.

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. 2023. Let's Sample Step by Step: Adaptive-Consistency for Efficient Reasoning and Coding with LLMs. In *EMNLP*.

Ramón Fernandez Astudillo, Md Arafat Sultan, Aashka Trivedi, Yousef El-Kurdi, Tahira Naseem, Radu Florian, and Salim Roukos. 2025. Optimal Policy Minimum Bayesian Risk. *Preprint*, arXiv:2505.17242.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Preprint*, arXiv:2501.12948.

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 *NeurIPS Track on Datasets and Benchmarks (Round 2)*.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. 2025. Efficient Test-Time Scaling via Self-Calibration. *Preprint*, arXiv:2503.00031.

Gregory Kang Ruey Lau, Wenyang Hu, Liu Diwen, Chen Jizhuo, See-Kiong Ng, and Bryan Kian Hsiang

Low. 2024. Dipper: Diversity in Prompts for Producing Large Language Model Ensembles in Reasoning tasks. In *NeurIPS Workshop on Foundation Model Interventions*.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape Sky-high Cost: Early-stopping Self-Consistency for Multi-step Reasoning. In *ICLR*.

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 *ICLR*.

Mistral-AI. 2025. Magistral. *Preprint*, arXiv:2506.10910.

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. In *NeurIPS*.

Hugo Touvron, Thibaut Lavril, and 1 others. 2023. LLaMA: Open and Efficient Foundation Language Models. *Preprint*, arXiv:2302.13971.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2025. Reasoning Aware Self-Consistency: Leveraging Reasoning Paths for Efficient LLM Sampling. In *NAACL*.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2024. Soft Self-Consistency Improves Language Models Agents. In *ACL*.

Tianchun Wang, Zichuan Liu, Yuanzhou Chen, Jonathan Light, Haifeng Chen, Xiang Zhang, and Wei Cheng. 2025a. Diversified Sampling Improves Scaling LLM inference. *Preprint*, arXiv:2502.11027.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *ICLR*.

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. *Preprint*, arXiv:2503.01422.

An Yang, Beichen Zhang, and 1 others. 2024. Qwen2.5-Math Technical Report: Toward Mathematical Expert Model via Self-Improvement. *arXiv preprint arXiv:2409.12122*.

## A Results with a Fixed Step Size

In Table 3 we compare the token savings from the proposed method for the two step size schedules $\breve{C}$ described in §3. Across all models and benchmarks and for two different sample budgets 16 and 32, savings remain equally robust for both schedules.

## B Response Length Statistics

Table 4 shows the average response length (in number of tokens; without hypothesis pruning) of all LLMs we evaluate on the two test sets. Qwen2.5-Math-7B-Instruct is the most verbose of the five models while Qwen2.5-Math-1.5B-Instruct produces the most succinct responses. Interestingly, we see no clear correlation between response length and performance (reported in §3) for these models.

## C Comparison with Random Pruning

In Table 2, we presented results of ablating the two components of the algorithm – confidence weighting and token set cover – separately. Here we ablate both together by substituting the set $keep$ in line 14 of Algorithm 2 with an equal number of randomly chosen hypotheses. This is in essence also a random baseline that, like *confidence weighting only*, borrows the exact number of hypotheses to prune from the full method, for which it does not have a mechanism of its own. As the last row of Table 5 shows, this random baseline is outperformed by the full method in $92.5\%$ of all $40$ evaluations described in §3. We also show the original ablation results in this table for reference.

## D Hypothesis Pruning at Different Steps

Figure 3 depicts the average $\%$ reduction in number of hypotheses during step-by-step generation of solutions for both MATH500 and AIME. A similar trend can be seen on both test sets: a larger fraction of hypotheses get pruned in early stages and fewer in later stages of generation. On both test sets, Qwen and Phi4 solutions are pruned more aggressively than Granite solutions; see §3 for a detailed discussion on this.

## E Overall Token Savings

Table 6 summarizes token savings from the proposed method for all models on both test sets. The same information is presented in Figure 1 using the symbol $\Delta$ and the two parallel vertical lines corresponding to each short polyline.

## F Compute Environment

We ran inference with all models on a single A100-80GB GPU; models were served using VLLM[5].

[5] https://github.com/vllm-project/vllm

**MATH500**

| | Dynamic@16 | Fixed@16 | Dynamic@32 | Fixed@32 |
|---|---|---|---|---|
| Qwen2.5-Math-1.5B-Instruct | 26.8% | 25.6% | 34.2% | 32.6% |
| Qwen2.5-Math-7B-Instruct | 15.9% | 17.6% | 22.8% | 30.3% |
| Granite3.3-2B-Instruct | 3.6% | 4.7% | 15.7% | 18.3% |
| Granite3.3-8B-Instruct | 7.6% | 7.2% | 0.4% | 0.4% |
| Phi4 | 1.0% | ~0% | 19.5% | 19.8% |

**AIME**

| | Dynamic@16 | Fixed@16 | Dynamic@32 | Fixed@32 |
|---|---|---|---|---|
| Qwen2.5-Math-1.5B-Instruct | 0.6% | 3.0% | 8.2% | 2.7% |
| Qwen2.5-Math-7B-Instruct | 9.8% | 13.5% | 31.5% | 48.1% |
| Granite3.3-2B-Instruct | 7.3% | 3.8% | 7.0% | 3.4% |
| Granite3.3-8B-Instruct | 5.1% | 2.5% | 5.0% | 12.3% |
| Phi4 | 4.4% | 12.6% | 5.7% | 5.5% |

Table 3: Token savings from confidence-weighted token set cover for dynamic and static step size schedules with sample budgets 16 and 32. Savings remain equally robust for both schedules.

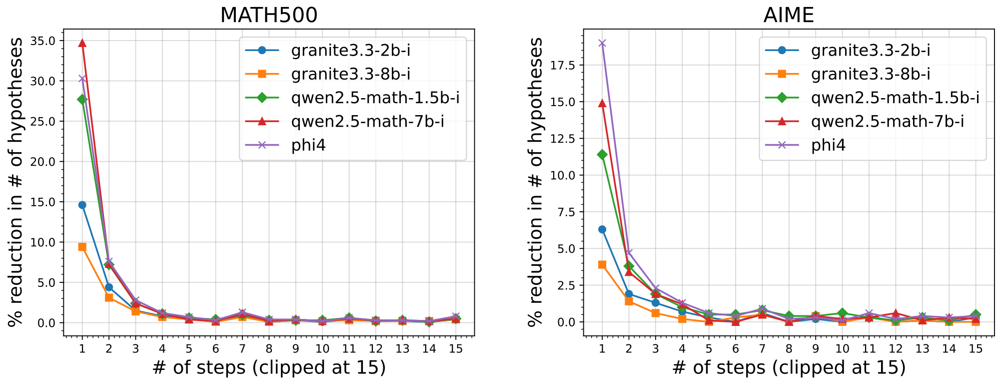


Figure 3: % reduction in the number of hypotheses as generation progresses (initial step size = 256). More hypotheses are pruned in earlier steps of generation; % reduction saturates near a small but non-zero value after about 5 or 6 steps. Qwen and Phi4 generations are pruned more aggressively as individual hypotheses provide greater lexical coverage of the cohort (§3).

| Model | MATH500 | AIME |
|---|---|---|
| Qwen2.5-Math-1.5B-Instruct | 551.9 | 867.9 |
| Qwen2.5-Math-7B-Instruct | 1104.6 | 2925.7 |
| Granite3.3-2B-Instruct | 724.7 | 1463.6 |
| Granite3.3-8B-Instruct | 763.8 | 1502.2 |
| Phi4 | 733.3 | 1510.1 |

Table 4: Average response length (# of tokens) of different LLMs measured in two test sets.

| Method 1 | Method 2 | Winner | Win % |
|---|---|---|---|
| CWSC | USC | CWSC | 82.5 |
| CWSC | CW | CWSC | 65.0 |
| USC | CW | CW | 62.5 |
| CWSC | Random | CWSC | 92.5 |

Table 5: Extended ablation results. CWSC: confidence-weighted set cover (proposed); USC: unweighted set cover; CW: confidence weighting with no set cover; Random: random hypothesis pruning. The proposed method outperforms all the simpler methods.

| Benchmark | Model | % **token savings** | | | |
|---|---|---|---|---|---|
| | | *sample budget* | | | |
| | | **8** | **16** | **32** | **64** |
| Math500 | Qwen2.5-Math-1.5B-Instruct | 14.5 | 26.8 | 34.2 | 33.4 |
| | Qwen2.5-Math-7B-Instruct | 11.2 | 15.9 | 22.8 | 18.6 |
| | Granite3.3-2B-Instruct | 3.3 | 3.6 | 15.7 | 20.1 |
| | Granite3.3-8B-Instruct | 1.8 | 7.6 | 0.4 | 20.6 |
| | Phi4 | 2.7 | 1.0 | 19.5 | 19.2 |
| Aime | Qwen2.5-Math-1.5B-Instruct | 2.5 | 0.6 | 3.0 | 4.0 |
| | Qwen2.5-Math-7B-Instruct | 10.2 | 9.8 | 13.5 | 28.3 |
| | Granite3.3-2B-Instruct | 4.6 | 7.3 | 3.8 | 7.0 |
| | Granite3.3-8B-Instruct | 0.1 | 5.1 | 2.5 | 7.1 |
| | Phi4 | 9.4 | 4.4 | 12.6 | 26.8 |

Table 6: $\%$ token savings from the proposed confidence-weighted token set cover method for all models and test sets. This is a tabular summary of all $\Delta$ values in Figure 1. We observe substantial gains of over $10\%$, reaching as high as $34\%+$, in many cases