SHARED-PREFIX SELF-CONSISTENCY

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-consistency improves reasoning by sampling complete reasoning traces independently and aggregating their answers by majority vote, but each additional trace requires another full generation. We introduce Shared-Prefix Self-Consistency (SP-SC), a simple variant that generates complete reasoning trunks, retains a prefix from each, and samples additional continuations. Original trunks and continuations contribute equally to the final vote. Using Qwen3-14B on AIME 2025 and HMMT 2025, we examine how prefix length, trunk count, and continuation budget affect accuracy and generated-token cost. Offline evaluation over fixed generation pools shows that selected configurations improve the observed trade-off relative to standard, adaptive, and early-stopping self-consistency. For example, on AIME 2025, an eight-candidate configuration achieves 81.72% accuracy using 24.5% fewer generated tokens than eight-sample self-consistency, which achieves 79.44%. Longer prefixes can reduce generation cost while lowering accuracy; multiple trunks mitigate this sensitivity in several evaluated configurations. A separate rescue analysis shows that erroneous traces can contain prefixes from which correct continuations emerge.

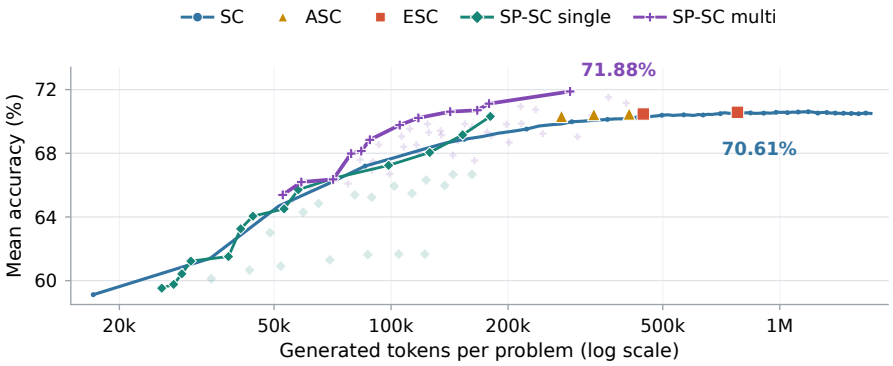

Figure 1: SP-SC achieves higher accuracy with fewer generated tokens, averaged across the AIME 2025 and HMMT 2025 mathematical reasoning datasets, using Qwen3-14B.

1 INTRODUCTION

Increasing inference time computation can substantially improve the reasoning ability of large language models (LLMs). A common approach is *self-consistency* (SC), which samples several independent thinking/reasoning traces and returns the answer with the most votes (Wang et al., 2023). SC is effective because errors made in one trace need not appear in others. Its cost, however, grows linearly with the number and length of the sampled traces: every additional vote normally requires another complete rollout.

Most efficient SC methods reduce the number of full rollouts. Adaptive Consistency, Early Stopping Self-Consistency, and related approaches stop sampling when the observed answers provide enough confidence or agreement (Aggarwal et al., 2023; Li et al., 2024; Wang et al., 2025a). These methods

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

reduce the *width* of sampling. We study a complementary question along its *depth*: must every vote regenerate the entire reasoning history?

We introduce *Shared Prefix Self-Consistency* (SP-SC), a training-free variant of SC that reuses rollout prefixes. SP-SC generates one or more independent reasoning trunks, retains a prefix from each trunk, and samples several continuations from the retained prefixes. All completed trunks and continuations contribute to a common vote. As illustrated in Figure 2, standard SC pays for an independent prefix for every vote, whereas SP-SC generates a prefix once and reuses it across several suffixes. The prefix length, number of trunks, and number of continuations are configurable. Longer prefixes provide greater token reuse but make the votes more correlated; additional trunks restore diversity at greater cost.

Our experiments evaluate this tradeoff with Qwen3-14B across competition mathematics benchmarks. SP-SC improves the empirical accuracy–token Pareto frontier relative to standard SC and other efficient SC variants. The improvement is strongest under low token budgets, where prefix reuse obtains more votes from the available generation budget. Single prefix configurations provide aggressive token reuse, while multiple prefix configurations recover reasoning diversity and remain competitive at larger budgets.

Our branching analysis helps explain this result. A trace ending in an incorrect answer may still contain a useful prefix from which correct continuations can be sampled. Sharing a prefix therefore does not merely propagate its original outcome. This finding challenges the view that SC efficiency must come from identifying and pruning unpromising rollouts: previously generated reasoning can instead be reused to produce additional votes.

Our contributions are:

- We introduce shared-prefix self-consistency (SP-SC), a configurable variant of self-consistency that reduces the tokens needed to obtain a pool of votes by reusing rollout prefixes.
- Our method improves the accuracy–token Pareto frontier relative to standard and efficient SC baselines, with its strongest gains under low token budgets.
- We show that prefixes from incorrect traces can still support correct continuations, providing an alternative to efficiency methods based only on pruning unpromising rollouts.

2 Related Work

Adaptive sampling for self-consistency. Self-consistency improves reasoning by sampling complete traces and voting over their answers, but its cost grows with the sampling budget (Wang et al., 2023). Adaptive Consistency, Dynamic Voting, and Early Stopping Self-Consistency stop when the observed answers provide sufficient agreement (Aggarwal et al., 2023; Xue et al., 2023; Li et al., 2024). Difficulty Adaptive Self-Consistency and Blend-ASC instead allocate samples across questions based on estimated difficulty or expected benefit (Wang et al., 2025a; Feng et al., 2025). These methods primarily reduce the number of independent rollouts. SP-SC changes the cost of each vote by allowing several rollouts to share an initial reasoning prefix.

Confidence weighting and trace pruning. Several methods use signals beyond answer counts. RASC evaluates reasoning paths, while CISC and ReASC incorporate confidence into voting or stopping (Wan et al., 2025; Taubenfeld et al., 2025; Kim et al., 2026). DeepConf filters low confidence traces using token probability signals (Fu et al., 2025). Its online variant reduces token costs by pruning traces below a confidence threshold, calibrated using 16 complete warmup traces per problem in the default configuration. As we'll see later selected SP-SC configurations require fewer tokens than the estimated cost of this warmup alone. Slim-SC detects similar intermediate thoughts and prunes redundant chains before completion (Hong et al., 2025). These methods improve efficiency by selecting, weighting, or terminating reasoning paths. SP-SC does not estimate path quality. It reuses generated reasoning to produce additional continuations, making prefix sharing complementary to confidence based selection and pruning.

Structured branching and search. Tree of Thoughts and Graph of Thoughts explore alternatives from intermediate states and use search or evaluation signals to select promising paths (Yao et al.,

Table 1: Rescue from trunks with explicitly incorrect final answers, using 20 suffix continuations per prefix. Prefix tokens are measured from the start of the trunk. Fractions show successful groups out of all eligible groups. Trunks with missing answers are excluded; missing suffix answers count as failures. Selected trunks may differ across milestones.

	HMMT 2025		AIME 2025	
Prefix tokens	Any Rescue (≥ 1)	Strong Rescue (≥ 10)	Any Rescue (≥ 1)	Strong Rescue (≥ 10)
100	65.0% (13/20)	35.0% (7/20)	92.9% (13/14)	71.4% (10/14)
200	60.0% (12/20)	40.0% (8/20)	85.7% (12/14)	71.4% (10/14)
500	65.0% (13/20)	40.0% (8/20)	92.9% (13/14)	64.3% (9/14)
1000	65.0% (13/20)	35.0% (7/20)	85.7% (12/14)	64.3% (9/14)
2000	65.0% (13/20)	35.0% (7/20)	78.6% (11/14)	50.0% (7/14)
3000	60.0% (12/20)	40.0% (8/20)	85.7% (12/14)	50.0% (7/14)
4000	55.0% (11/20)	40.0% (8/20)	85.7% (12/14)	42.9% (6/14)
5000	55.0% (11/20)	20.0% (4/20)	78.6% (11/14)	42.9% (6/14)
7000	50.0% (10/20)	25.0% (5/20)	78.6% (11/14)	28.6% (4/14)
10000	50.0% (10/20)	15.0% (3/20)	64.3% (9/14)	21.4% (3/14)

2023; Besta et al., 2023). More targeted methods branch only where additional exploration appears useful. Cautious Next Token Prediction samples short alternatives at high entropy tokens and selects among them using perplexity (Wang et al., 2025b). DeepLook branches at uncertainty bottlenecks and ranks lookahead continuations, while DUCHESS uses a learned activation based controller to continue, terminate, or duplicate reasoning branches (Yang et al., 2026; Jiang et al., 2025). Unlike these search methods, SP-SC does not rank intermediate states or choose a single path. It retains the resulting continuations as votes.

3 Does a Wrong Answer Mean Wasted Reasoning?

Pruning methods such as DeepConf improve inference efficiency by identifying and pruning low-confidence reasoning traces before they finish (Fu et al., 2025). This approach saves tokens by treating an unpromising partial trace as having little future value. We ask whether that assumption is always justified: even when an LLM reasons extensively and ultimately produces an incorrect answer, its prefix may still contain a productive intermediate state from which an alternative continuation can succeed. We test this possibility through an oracle experiment.

3.1 Oracle Rescue Experiment

We conduct the rescue experiment with Qwen3-14B on AIME 2025 and HMMT 2025, using temperature $T = 0.6$ and top-$p = 0.95$. We first generate complete reasoning traces, which we call *trunks*. At each token milestone m, we select an incorrect trunk long enough to provide the prefix, retain its first m tokens, and sample 20 alternative suffixes using the same decoding settings. We report **Any Rescue** when at least one of these suffixes is correct and **Strong Rescue** when at least ten are correct, distinguishing prefixes that occasionally admit a correct outcome from those that do so consistently.

3.2 Prefixes Retain Alternative Outcomes

Table 1 shows that prefixes from incorrect trunks frequently yield correct continuations. At a 100-token prefix, Any Rescue occurs in 65.0% of HMMT groups and 92.9% of AIME groups; Strong Rescue occurs in 35.0% and 71.4%, respectively.

Rescue is generally less frequent for longer prefixes, although the pattern is not monotonic. This is consistent with longer prefixes constraining more of the subsequent reasoning, but the selected trunks can differ across milestones, so the table does not establish a causal effect of prefix length.

Thus, a reasoning prefix does not necessarily determine the final answer, even when its original continuation might be incorrect. This supports the premise that we can reuse a prefix without collapsing every candidate to the trunk's answer.

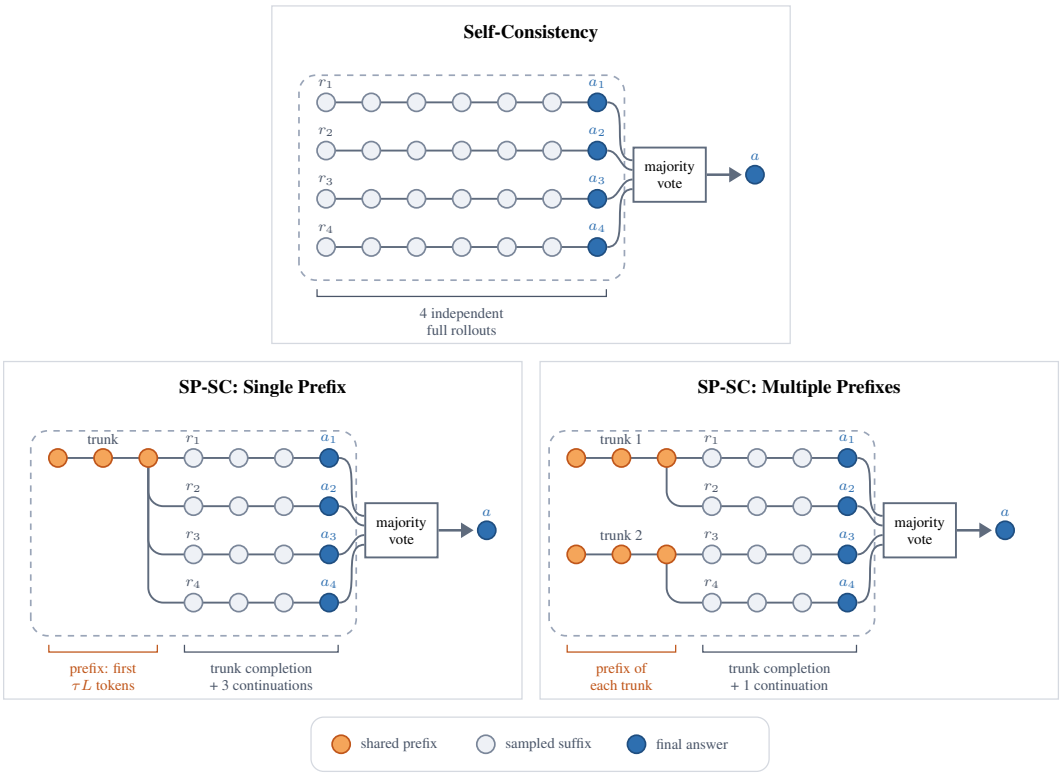

Figure 2: SC samples $N = 4$ independent rollouts. SP-SC generates a full *trunk*, reuses its first $\text{round}(\tau L)$ tokens (orange) as a prefix for $k - 1$ further continuations, and keeps the trunk itself as a candidate. Single-prefix SP-SC uses one trunk ($S = 1$, $k = 4$); multi-prefix SP-SC uses $S = 2$ trunks with $k = 2$ each ($K = Sk = 4$). All candidates are combined by majority vote.

4 Shared-Prefix Self-Consistency

We hypothesize that, for many complex reasoning problems such as mathematics, the early stages of reasoning are strongly guided by the question, while meaningful diversity emerges later in the reasoning process. Reusing one or a small number of representations of these early stages may therefore preserve the diverse later-stage reasoning on which self-consistency relies. The rescue analysis above further shows that prefixes from traces with incorrect final answers can still produce correct continuations, suggesting that early reasoning may remain useful even when the original rollout fails. Motivated by these observations, we introduce Shared-Prefix Self-Consistency (SP-SC), illustrated in Figure 2. Whereas standard self-consistency generates each complete response independently, SP-SC shares the initial segment of one or more reasoning trunks across several independently sampled continuations. Throughout, N denotes the number of independent complete responses used by standard SC. For SP-SC, S is the number of independently generated trunks, k is the number of candidates contributed by each trunk, and $K = Sk$ is the total candidate count. The prefix fraction τ, with $0 < \tau \leq 1$, determines how much of each trunk is shared.

4.1 Single-Prefix Sampling

Given a problem, we generate a complete response, called a *trunk*. If its generated length is L, we retain the first $p = \text{round}(\tau L)$ tokens, where p is the retained prefix length. We then sample $k - 1$ continuations from the original prompt and this prefix. Together with the completed trunk, these form k answer candidates. Thus, the shared prefix is generated once, while only the suffix after token p is resampled for the additional $k - 1$ candidates. The original trunk contributes to the final vote, but its answer is not used to select the trunk or determine the branching point.

4.2 Multi-Prefix Sampling

Multi-prefix SP-SC generates S independent trunks and applies the same procedure to each. With $k-1$ continuations per trunk, the total number of answer candidates is $K = Sk$. Single-prefix SP-SC is the case $S = 1$. We report SP-SC configurations as (τ, S, K); consequently, the per-trunk candidate count is $k = K/S$ for the uniformly allocated configurations studied here. In particular, each trunk contributes one original candidate and $k-1$ newly sampled continuations. Using multiple trunks provides different starting paths while retaining prefix reuse within each group. Increasing S also incurs additional full-trunk generation cost, creating a trade-off between reuse and diversity.

4.3 Answer Aggregation and Token Cost

We return the most frequent answer across the original trunks and their continuations. Each candidate contributes one vote, regardless of whether it is an original trunk or a continuation. Missing answers abstain. Ties follow first occurrence.

Let L_i denote the full generated length of trunk i, and U_{ij} the newly generated length of its continuation j. The generated-token cost is

$$C_{\text{SP–SC}} = \sum_{i=1}^{S} L_i + \sum_{i=1}^{S} \sum_{j=1}^{k-1} U_{ij}. \tag{1}$$

The retained prefix is already part of L_i and is not charged again. This includes the full trunk generation needed to determine the fractional branching point. Candidates without extracted answers still incur cost.

5 Experimental Setup

5.1 Models and Datasets

Model. We choose a reasoning model that balances capacity for challenging mathematical tasks with the practicality of inference on a single high-memory GPU. We use Qwen3-14B with thinking enabled. Generation uses vLLM with temperature 0.6, nucleus sampling with $p = 0.95$, and a maximum output budget of 32,000 tokens per trace, subject to available context. For continuation suffixes, the available budget accounts for the original prompt and retained prefix.

Datasets. We choose challenging competition-level mathematics benchmarks to evaluate whether sharing reasoning prefixes can reduce token costs while preserving accuracy on problems requiring extended reasoning. AIME 2025[1] and HMMT February 2025[2] each contain 30 problems in MathArena, and we evaluate every method on all 30 problems from each benchmark. AIME answers are integers, whereas HMMT answers can be symbolic expressions, including fractions and radicals, such as $\frac{1}{2}$, $\sqrt{2}$, or $3 + \sqrt{5}$. We therefore use integer normalization for AIME and symbolic normalization for HMMT.

Prompting and grading. We use the model's chat template and ask it to reason step by step and place its final answer in `\boxed{}`. We extract the last brace-balanced boxed answer. Mathematically equivalent answers are grouped together before voting, whether correct or incorrect, using exact arithmetic and supported symbolic normalization. Responses without extracted answers do not vote, but their tokens still count toward cost. An all-abstaining prediction is incorrect.

5.2 Baselines

Self-consistency (SC): SC votes over N independent complete responses (Wang et al., 2023). The main figures show $N = 1, \ldots, 100$.

[1] https://huggingface.co/datasets/MathArena/aime_2025

[2] https://huggingface.co/datasets/MathArena/hmmt_feb_2025

Early-stopping self-consistency (ESC): ESC processes nonoverlapping windows of responses and returns the first unanimous, nonmissing window answer (Li et al., 2024). Otherwise, it votes over all consumed responses at its cap. We show window/cap settings $(5, 40)$ and $(8, 64)$.

Adaptive self-consistency (ASC): ASC processes responses sequentially and stops when a beta-based confidence criterion exceeds 0.90, 0.95, or 0.99, or after 40 responses (Aggarwal et al., 2023). It votes over the consumed responses.

5.3 Shared-Prefix Sampling Protocol

We use 316 saved independent full response traces per problem for both datasets. All baselines sample from this pool, and four traces in the pool serve as SP-SC trunks, with no correctness-based filtering. At each $\tau \in \{0.2, 0.3, 0.4, 0.5\}$, the designated first trunk has 12 saved continuations and each other trunk has six. This gives 120 trunks (30×4) and 3,600 continuations ($30\times4\times(12+3\times6)$) per dataset.

Using the notation of Section 4, single-prefix evaluation fixes the designated first trunk and sweeps $k \in \{2, 3, 4, 5, 7, 9, 11, 13\}$, giving 32 configurations (4 prefix fractions $\times$ 8 budgets). Multi-prefix evaluation sweeps $S \in \{2, 3, 4\}$ and $k \in \{2, 3, 4, 7\}$, giving 48 configurations (4 prefix fractions $\times$ 3 trunk counts $\times$ 4 budgets). The main comparison therefore displays 185 configurations per dataset: 80 SP-SC configurations ($32 + 48$) and 105 baseline configurations, comprising 100 SC budgets ($N = 1, \ldots, 100$), three ASC thresholds, and two ESC settings. Each SC budget samples from the full 316-response pool.

5.4 Evaluation Protocol

We evaluate each configuration over 500 offline resampling trials to reduce dependence on any one selection of saved traces. For each problem and trial, SC, ESC, and ASC use the same randomly shuffled response pool. SP-SC samples continuations from the saved pools, with multi-prefix configurations also sampling trunks without replacement. For a given method and prefix fraction, increasing the sampling budget keeps the previously selected candidates and adds more from the same shuffled pools. We report accuracy and token cost averaged over all 500 trials. These trials reuse existing traces and measure variability due to sampling, rather than generating new traces.

We report accuracy and generated tokens per problem, averaged over trials. For $T = 500$ trials and $P = 30$ problems, the plotted cost is

$$\overline{C} = \frac{1}{TP} \sum_{t=1}^{T} \sum_{p=1}^{P} C_{t,p}. \tag{2}$$

Costs follow Equation 1 for SP-SC and include all responses consumed by a baseline. Input prompt tokens and unused pool samples are excluded. These are generated-token measurements, not measured latency or the total expense of constructing the experimental pools.

Error bars in Appendix Figure 7 and 8 show standard deviations across the 500 resampling trials. These trials reuse the same problems and saved traces, so they measure variation from trace selection rather than independent generation experiments.

A configuration is Pareto-optimal if no other configuration in the plot achieves higher accuracy at equal or lower token cost, or equal accuracy at lower token cost. We identify the Pareto frontier separately for the configurations shown in each plot. Our comparisons are restricted to the evaluated settings and should be interpreted as descriptive results of a sweep.

6 Results and Analysis

Figures 3 and 4 show the accuracy–token trade-offs on AIME 2025 and HMMT 2025, respectively. SP-SC contributes new points to the observed Pareto frontier on both datasets, although the gains depend on the token budget and configuration.

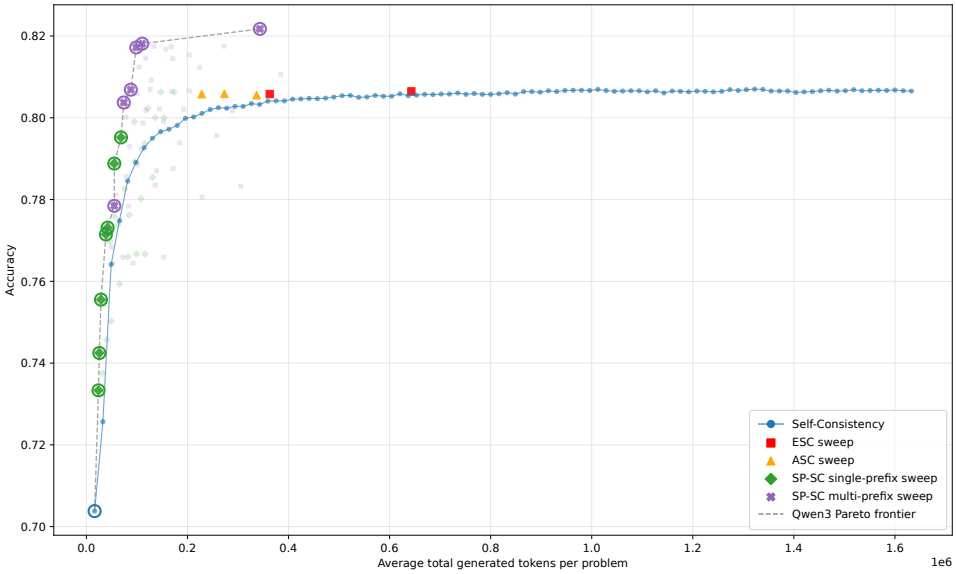

Figure 3: Qwen3-14B accuracy versus mean generated tokens per problem on AIME 2025, averaged over 500 resampling trials. SP-SC includes original trunk votes and full-trunk generation cost. Circled points mark the observed joint Pareto frontier.

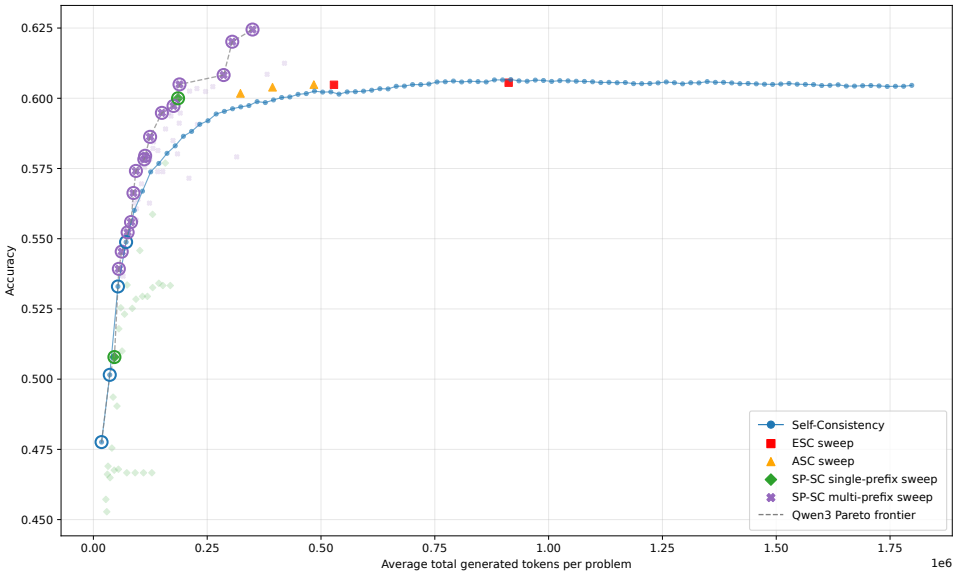

Figure 4: Corresponding accuracy–token comparison on HMMT 2025 under the same evaluation protocol as Figure 3. The advantage of SP-SC emerges at a larger budget than on AIME, but multi-prefix configurations reach the highest observed accuracies.

To make the trade-offs in these curves more concrete, Table 2 presents selected SP-SC and baseline configurations. The complete per-dataset curves and all observed Pareto-frontier configurations are reported in Appendix A, Figures 5–6, and Tables 5–6.

Table 2: Selected configurations from the 500-trial sweep. Tokens are thousands of generated tokens per problem. SP-SC labels give (τ, S, K), with K including original trunk candidates.

Dataset	Configuration	Accuracy (%)	Tokens (k)
AIME	SC $N = 8$	79.44	130.6
	SC $N = 71$	80.72	1159.7
	ASC $c = .90$	80.53	227.7
	ESC $(8, 64)$	80.62	643.1
	SP-SC $(.5, 4, 8)$	81.72	<u>98.7</u>
	SP-SC $(.3, 4, 28)$	**82.17**	343.4
HMMT	SC $N = 7$	57.36	125.8
	SC $N = 65$	60.55	1168.2
	ASC $c = .99$	60.38	483.1
	ESC $(8, 64)$	60.53	913.2
	SP-SC $(.3, 3, 6)$	57.41	<u>93.4</u>
	SP-SC $(.4, 4, 28)$	**62.45**	349.6

6.1 Accuracy–Token Trade-offs

To compare low- to moderate-budget settings, for each SP-SC configuration in Table 3, we searched SC over $N = 1, \ldots, 100$ and selected the cheapest setting within 0.5 percentage points of its accuracy. On AIME 2025, SP-SC saves 43.6% of tokens near 79% accuracy and 61.5% near 81% accuracy. On HMMT 2025, savings reach 25.8% near 57% accuracy and 54.2% near 60% accuracy. Thus prefix reuse is most useful when the desired accuracy would otherwise require several complete rollouts.

At higher accuracy levels, the comparison changes from cost reduction to the best accuracy each method reaches. The best observed SP-SC means are 82.17% on AIME and 62.45% on HMMT, compared with 80.72% and 60.55% for SC across the full sweep through $N = 100$. No SC configuration falls within 0.5 percentage points of either SP-SC result. The highest-accuracy SP-SC settings use multiple prefixes, which extend the frontier beyond both standard and adaptive baselines.

Table 3: SP-SC configurations and the cheapest SC setting within ± 0.5 percentage points of the same accuracy.

		SP-SC		Matched SC			
Dataset	(τ, S, K)	Acc. (%)	Tokens (k)	N	Acc. (%)	Tokens (k)	Savings
AIME	$(.2, 1, 4)$	78.88	55.3	6	78.73	98.0	43.56%
AIME	$(.2, 3, 6)$	80.69	88.0	14	80.19	228.7	61.54%
AIME	$(.5, 4, 8)$	81.72	98.7	–	–	–	–
AIME	$(.3, 4, 28)$	82.17	343.4	–	–	–	–
HMMT	$(.3, 3, 6)$	57.41	93.4	7	57.36	125.8	25.77%
HMMT	$(.5, 4, 16)$	60.49	189.2	23	59.99	413.4	54.24%
HMMT	$(.4, 4, 28)$	62.45	349.6	–	–	–	–

6.2 Effect of Prefix Length

Prefix sharing creates a direct trade-off: retaining a larger fraction of the trunk reduces the number of new suffix tokens, but may constrain the alternative reasoning paths. Table 4 isolates this effect by fixing the total candidate count at $K = 9$ and varying the prefix fraction τ for single-prefix ($S = 1$) and multi-prefix ($S = 3$) SP-SC.

Table 4: Effect of prefix fraction with $K = 9$ fixed. Each cell reports accuracy (%) / generated tokens (thousands).

τ	AIME, $S = 1$	AIME, $S = 3$	HMMT, $S = 1$	HMMT, $S = 3$
0.2	80.23 / 121.4	80.92 / 128.1	55.87 / 130.1	57.39 / 141.4
0.3	78.03 / 107.9	80.18 / 117.9	52.95 / 118.6	58.45 / 132.0
0.4	78.93 / 95.9	79.24 / 108.4	52.95 / 107.9	57.83 / 124.0
0.5	76.60 / 82.5	80.59 / 100.1	46.67 / 91.8	57.54 / 112.9

Increasing τ from 0.2 to 0.5 lowers mean token cost in all four columns. With one prefix, accuracy falls by 3.63 percentage points on AIME and 9.20 points on HMMT. With three prefixes, the corresponding endpoint changes are only -0.33 and $+0.15$ points. Although the intermediate values are not monotonic, the central pattern is consistent across datasets: longer prefix reuse is substantially more fragile when all candidates share the same trunk.

6.3 Single versus Multiple Prefixes

Multiple prefixes spend additional tokens on independent trunks, but recover diversity that a single shared prefix can lose. At the matched candidate count in Table 4, $S = 3$ is more accurate than $S = 1$ for every tested prefix fraction on both datasets. The difference is especially large for long prefixes on HMMT: at $\tau = 0.5$, three-prefix SP-SC improves accuracy from 46.67% to 57.54%, while increasing cost from 91.8k to 112.9k tokens.

The same pattern appears in the full sweep. The best single-prefix means are 80.63% on AIME and 60.00% on HMMT, whereas multi-prefix SP-SC reaches 82.17% and 62.45%, respectively. Single-prefix sampling can still be useful when minimizing cost is the primary objective, but the results favor multiple prefixes as the more reliable choice when accuracy and robustness to prefix length matter.

7 Conclusion

Shared-Prefix Self-Consistency is a simple modification to self-consistency: instead of generating every reasoning trace from scratch, it reuses early reasoning prefixes to obtain additional votes at lower generation cost. Despite its simplicity, SP-SC achieves strong accuracy–token trade-offs on challenging mathematical reasoning benchmarks, often matching or exceeding standard and adaptive self-consistency while using fewer generated tokens. The same idea is extendable beyond competition mathematics to other tasks that benefit from sampling and aggregating multiple lines of reasoning. Our results also show that reasoning diversity is important: across the tested prefix lengths, using multiple independent trunks is consistently more accurate and less sensitive to prefix choice than relying on a single shared trunk. Multi-prefix SP-SC therefore provides the most reliable practical setting, combining prefix reuse with enough independent starting paths to preserve the diversity that makes self-consistency effective.

AI use statement

We used Claude Code and Codex to assist with implementing and debugging code, preparing figures and tables, drafting and editing the manuscript, and providing feedback on experimental methodology and interpretation of results. All generated content was reviewed and revised thoroughly by the authors to ensure accuracy and clarity. The authors take responsibility for the final content of this work.

Ethics statement

Our experiments use publicly available, open-weight language models and existing mathematical reasoning benchmarks, without private or sensitive data. All experimental hyperparameters are reported to support reproducibility. Model-generated reasoning may contain errors, and benchmark

performance does not establish reliability in safety-critical applications. Reported token savings should not be interpreted as equivalent reductions in latency or energy consumption.

Reproducibility Statement

We describe the sampling procedure, answer aggregation, and generated-token accounting in Section 4. Section 5 provides the model and dataset details, decoding settings, generation pools, configuration grid, and 500-trial offline resampling protocol. Section 3 specifies how the rescue cohort is constructed and how its metrics are computed. Together, these details specify the generation and evaluation procedures used to obtain our results.

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12375–12396. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.761.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023.

Austin Feng, Marius Alonso, and Ambroise Odonnat. Optimal self-consistency for efficient reasoning with large language models. *arXiv preprint arXiv:2511.12309*, 2025.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.

Colin Hong, Xu Guo, Anand Chaanan Singh, Esha Choukse, and Dmitrii Ustiugov. Slim-SC: Thought pruning for efficient scaling with self-consistency. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 34500–34517. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1750.

Weifan Jiang, Rana Shahout, Yilun Du, Michael Mitzenmacher, and Minlan Yu. Intra-request branch orchestration for efficient LLM reasoning. *arXiv preprint arXiv:2509.24957*, 2025.

Junseok Kim, Nakyeong Yang, Kyungmin Min, and Kyomin Jung. Reliability-aware adaptive self-consistency for efficient sampling in LLM reasoning. *arXiv preprint arXiv:2601.02970*, 2026.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *International Conference on Learning Representations*, 2024.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20090–20111. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1030.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3613–3635. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.184.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6919–6932. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.findings-naacl.383.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yizhou Wang, Lingzhi Zhang, Yue Bai, Mang Tik Chiu, Zhengmian Hu, Mingyuan Zhang, Qihua Dong, Yu Yin, Sohrab Amirghodsi, and Yun Fu. Cautious next token prediction. In *Findings of the Association for Computational Linguistics: ACL 2025*. Association for Computational Linguistics, 2025b. URL https://aclanthology.org/2025.findings-acl.1318/.

Mingfeng Xue, Dayiheng Liu, Wenqiang Lei, Xingzhang Ren, Baosong Yang, Jun Xie, Yidan Zhang, Dezhong Peng, and Jiancheng Lv. Dynamic voting for efficient reasoning in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3085–3104. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.203.

Tingxin Yang, Zefeng Wang, Mengyue Wang, Xingcheng Zhou, and Yunpu Ma. DeepLook: Deeper thinking with lookahead. *arXiv preprint arXiv:2607.22602*, 2026.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023.

A Additional Accuracy–Token Results

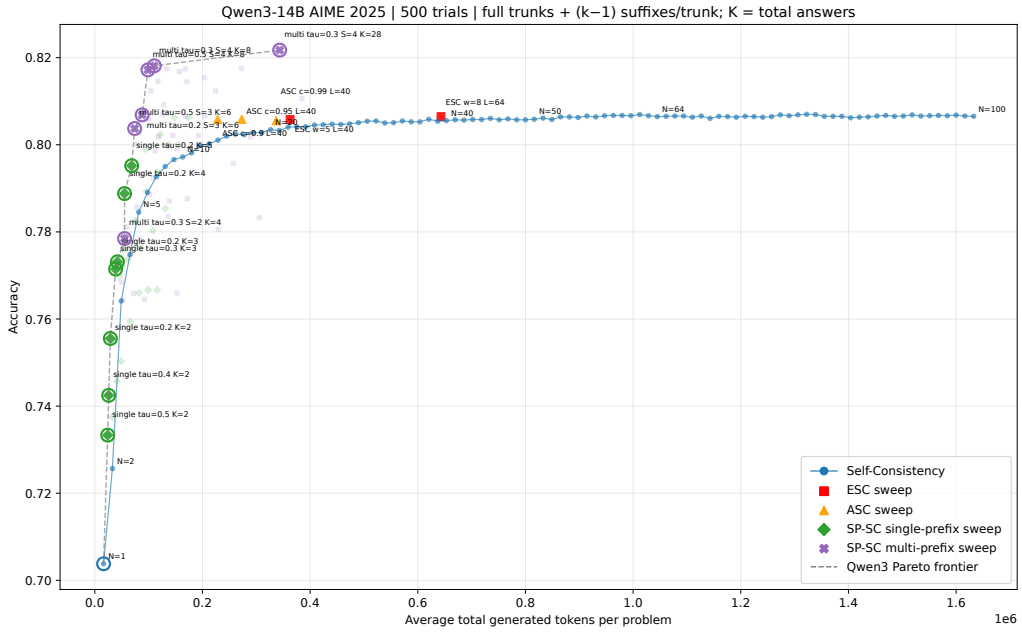

Figure 5: Accuracy–token tradeoffs for Qwen3-14B on AIME 2025, averaged over 500 resampling trials. SP-SC costs include complete trunks and newly generated suffixes.

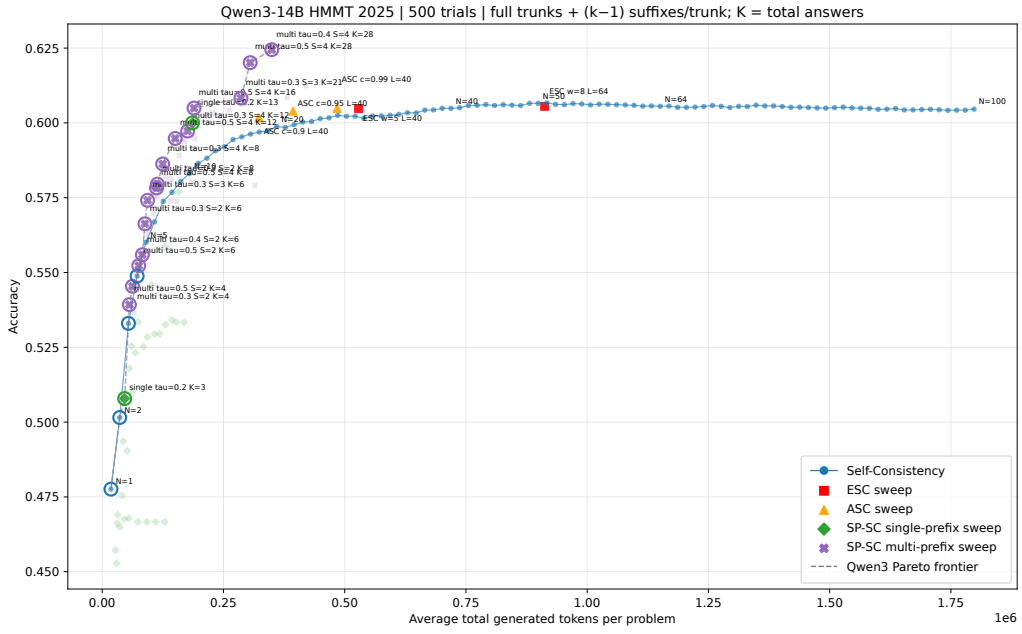

Figure 6: Accuracy–token tradeoffs for Qwen3-14B on HMMT 2025, averaged over 500 resampling trials. SP-SC costs include complete trunks and newly generated suffixes.

This appendix provides per-dataset curves, resampling-variability plots, and complete Pareto tables underlying the selected comparisons in Tables 2 and 3. Figures 5 and 6 show the complete AIME 2025 and HMMT 2025 trade-off curves.

A.1 Variability Across Resampling Trials

Figures 7 and 8 add error bars to the per-dataset comparisons. The bars show variability caused by selecting different subsets from the fixed generation pools; they are not confidence intervals over new benchmark problems or newly generated traces.

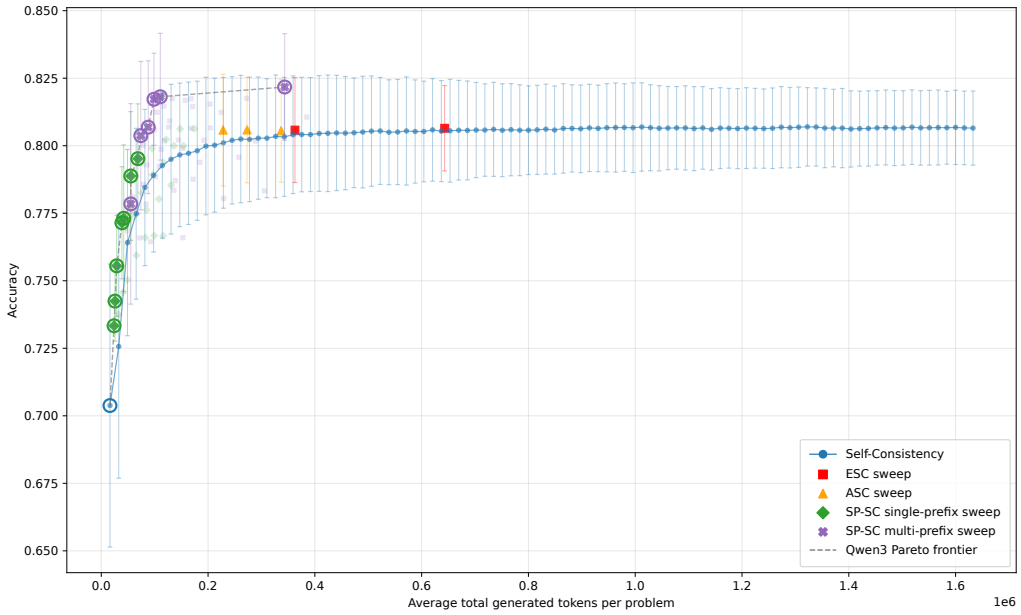

Figure 7: AIME 2025 accuracy–token trade-offs with ± 1 standard deviation across 500 resampling trials.

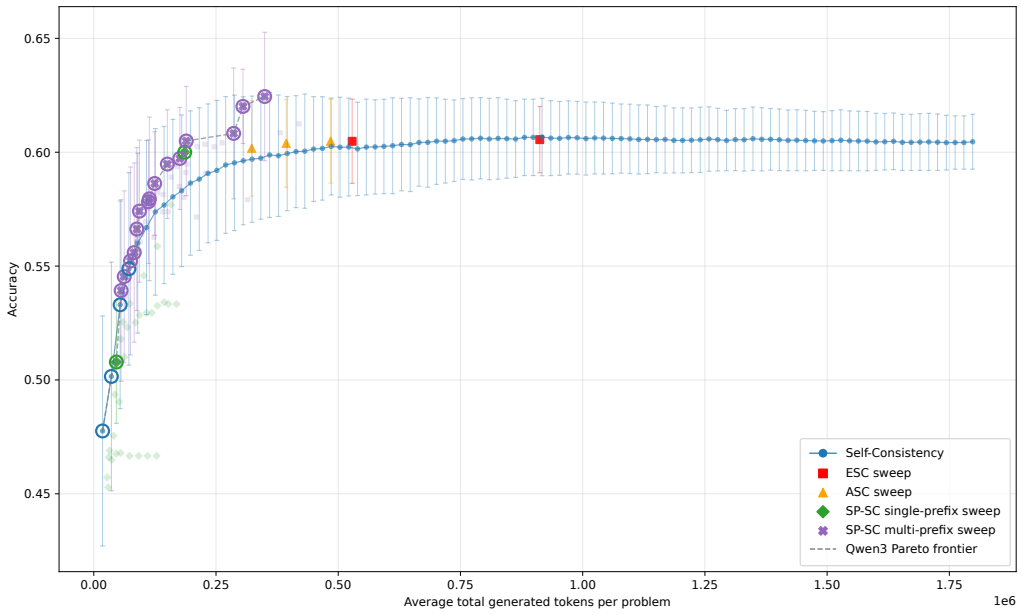

Figure 8: HMMT 2025 accuracy–token trade-offs with ± 1 standard deviation across 500 resampling trials.

A.2 Complete Pareto Configurations

Tables 5 and 6 list all joint Pareto points, sorted by mean token cost. Accuracy is mean ± standard deviation across 500 resampling trials, not a confidence interval. The candidate column is N for SC and K for SP-SC, while Δ is the gain over the best tested SC, ASC, or ESC setting at equal or lower cost. The 14 AIME and 21 HMMT points are distinct from the frontier of the dataset-averaged first-page figure.

Table 5: Complete Pareto frontier on AIME 2025. Tokens are thousands per problem and Δ is measured in percentage points.

Method	τ	S	K or N	Acc. (%)	Tokens (k)	Δ (pp)
SC	–	–	1	69.96 ± 4.98	16.3	+0.00
SP-SC single	0.5	1	2	73.33 ± 0.00	24.0	+3.37
SP-SC single	0.4	1	2	74.25 ± 1.49	25.7	+4.29
SP-SC single	0.2	1	2	75.55 ± 1.87	28.9	+5.59
SP-SC single	0.3	1	3	77.15 ± 2.07	38.7	+5.05
SP-SC single	0.2	1	3	77.31 ± 2.72	42.1	+5.22
SP-SC multi	0.3	2	4	77.85 ± 3.71	55.26	+1.83
SP-SC single	0.2	1	4	78.88 ± 2.38	55.29	+2.87
SP-SC single	0.2	1	5	79.52 ± 2.03	68.5	+2.20
SP-SC multi	0.5	3	6	80.37 ± 2.75	74.0	+3.05
SP-SC multi	0.2	3	6	80.69 ± 2.46	88.0	+2.42
SP-SC multi	0.5	4	8	81.72 ± 1.71	98.7	+2.99
SP-SC multi	0.3	4	8	81.81 ± 2.35	110.6	+3.09
SP-SC multi	0.3	4	28	82.17 ± 1.97	343.4	+1.59

Table 6: Complete Pareto frontier on HMMT 2025. Tokens are thousands per problem and Δ is measured in percentage points.

Method	τ	S	K or N	Acc. (%)	Tokens (k)	Δ (pp)
SC	–	–	1	48.27 ± 5.08	18.0	+0.00
SC	–	–	2	50.70 ± 4.86	35.9	+0.00
SP-SC single	0.2	1	3	50.79 ± 2.69	46.2	+0.09
SC	–	–	3	53.36 ± 4.38	53.9	+0.00
SP-SC multi	0.5	2	4	53.93 ± 3.99	55.9	+0.57
SP-SC multi	0.3	2	4	54.54 ± 3.76	62.3	+1.18
SC	–	–	4	54.81 ± 4.31	71.9	+0.00
SP-SC multi	0.5	2	6	55.23 ± 4.13	75.2	+0.42
SP-SC multi	0.4	2	6	55.59 ± 3.94	82.7	+0.79
SP-SC multi	0.3	2	6	56.63 ± 3.58	88.0	+1.82
SP-SC multi	0.3	3	6	57.41 ± 3.12	93.4	+1.29
SP-SC multi	0.5	4	8	57.83 ± 2.71	111.8	+1.05
SP-SC multi	0.3	2	8	57.95 ± 3.59	113.7	+1.18
SP-SC multi	0.3	4	8	58.63 ± 2.28	124.6	+1.85
SP-SC multi	0.5	4	12	59.48 ± 2.38	150.5	+1.67
SP-SC multi	0.3	4	12	59.73 ± 2.24	176.0	+1.43
SP-SC single	0.2	1	13	60.00 ± 0.00	186.0	+1.71
SP-SC multi	0.5	4	16	60.49 ± 2.40	189.2	+2.20
SP-SC multi	0.3	3	21	60.83 ± 2.87	286.1	+1.35
SP-SC multi	0.5	4	28	62.01 ± 1.63	305.1	+2.19
SP-SC multi	0.4	4	28	62.45 ± 2.82	349.6	+2.33

B Computational Resources

We generated Qwen3-14B SP-SC traces using one NVIDIA H100 GPU with 80 GB of memory per generation job. Inference used vLLM with BF16 precision and no tensor parallelism. The 500-trial resampling evaluations, answer aggregation, and figure generation ran on CPUs using saved traces, without additional model inference. Reported token costs measure generated tokens rather than GPU time.