FROM CHAINS TO TREES: PARENT-CONDITIONED DRAFTING FOR SEMI-AUTOREGRESSIVE SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates LLM inference only when drafted continuations survive target-model verification. Semi-autoregressive drafters such as DSpark predict a token block with one backbone forward and refine it with a lightweight Markov head. However, DSpark decodes this block as a chain, so an early mismatch invalidates the remaining suffix and limits the benefit of large draft blocks.

We show that the conditional structure already learned by DSpark can support multiple parent-consistent continuations without retraining or additional backbone passes. We introduce **P**arent-**C**onditioned Drafting **T**ree (PCTree), which uses the pretrained Markov head to score alternative children separately for each parent and allocates a fixed verification budget to the most probable paths. This converts DSpark's linear draft into a tree while preserving its one-pass parallel backbone.

Across Qwen3-{4B,8B,14B} and nine benchmarks, at $B{=}7$, measured speedup gains over autoregressive (AR) decoding, relative to matched DSpark, range from 3.1% to 29.5%. On Qwen3-4B GSM8K at $B{=}16$, PCTree increases mean acceptance length from 9.41 to 11.16 and three-run mean AR speedup from $6.14\times$ to $6.60\times$. These show that parent-conditioned branching can turn conditional capacity already present in a semi-autoregressive drafter into end-to-end inference gains through an inference-only change.

1 INTRODUCTION

Autoregressive decoding remains a major latency bottleneck for large language models because each step exposes little parallel work. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023; Xia et al., 2023) instead drafts several future tokens and verifies them together, but its benefit depends on how much of each draft survives verification. Tree verification can hedge against early mismatches by packing alternatives into one target forward (Miao et al., 2024). Autoregressive drafters such as EAGLE grow such trees through multiple sequential draft steps (Li et al., 2024a;b; 2025b), while parallel drafters trade stronger inter-token dependence for lower draft latency (Chen et al., 2026; Ringel & Romano, 2026).

DSpark (Cheng et al., 2026) combines a one-pass parallel backbone with a lightweight Markov head that conditions each draft position on its predecessor. However, its decoder uses this conditional model to construct only one chain, so a mismatch near the root invalidates the remaining suffix. This rejection cascade limits the number of tokens advanced per verification, especially for large draft blocks.

Branching alone does not fully solve the problem: if all parents at one depth share the same child distribution, individually likely tokens may form a conditionally unlikely path. Our key observation is that DSpark already provides the missing signal. We therefore introduce PCTree, which batches concrete parents through the pretrained Markov head, reuses the shared backbone logits, and retains a tree under a fixed verification budget (fig. 1). This inference-only conversion requires neither retraining nor additional backbone passes. It does add per-round tree and verification work, which must be offset by accepting longer paths in fewer rounds.

Experiments show that this trade-off is favorable in the evaluated settings. A matched Shared-Markov control attributes longer acceptance to parent-specific conditioning rather than branching alone.

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.

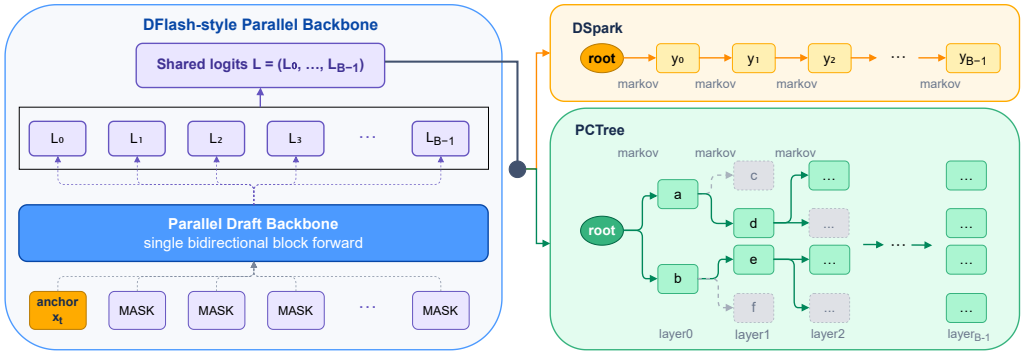

Figure 1: **From DSpark to PCTree.** One parallel backbone forward produces shared base logits $L_0, \ldots, L_{B-1}$. DSpark forms one chain with B batch-1 Markov stages, whereas PCTree batches frontier parents and retains a tree under node budget N. The same Markov head is reused at every stage; both use the same pretrained weights and number of sequential stages.

Across Qwen3-{4B,8B,14B} and nine benchmarks at B=7, PCTree improves acceptance in every model–task pair and raises measured end-to-end AR speedup by 3.1%–29.5% over DSpark. On Qwen3-4B GSM8K at B=16, it increases mean acceptance length from 9.41 to 11.16 and three-run mean AR speedup from 6.14× to 6.60×.

The paper makes three contributions:

- **A decoding perspective on semi-autoregressive drafting.** We show that DSpark's parent-dependent factorization can support tree-shaped proposals despite its original single-chain decoder.
- **Training-free parent-conditioned tree drafting.** We propose PCTree, which constructs a fixed-budget tree to improve acceptance length without changing the pretrained drafter.
- **Controlled mechanism and end-to-end evidence.** We show the importance of parent conditioning and the gain in acceptance and wall-clock through experimental evidence.

2 Related Work

Speculative decoding and tree verification. Early speculative decoding systems draft a single continuation and ask the target model to accept a prefix (Leviathan et al., 2023; Chen et al., 2023; Xia et al., 2023). Because later tokens are discarded after an early mismatch, chain drafts are brittle. Tree-based verification (Miao et al., 2024) packs alternatives into one attention pattern so that a rejected sibling does not invalidate an entire branch. Follow-up work improves the *drafter* with multi-head proposals, target features, dynamic tree shapes, and multi-layer fusion (Cai et al., 2024; Li et al., 2024a;b; Wang et al., 2025; Li et al., 2025b). These methods largely grow trees autoregressively, requiring additional draft forwards.

Parallel drafting. Parallel generation predicts multiple positions together to reduce draft latency. Non-autoregressive models remove within-sequence dependencies, whereas semi-autoregressive models retain dependencies between successive groups (Gu et al., 2018; Wang et al., 2018). Blockwise parallel decoding, multi-token prediction, and multi-head drafting similarly trade within-block dependence against latency (Stern et al., 2018; Gloeckle et al., 2024; Cai et al., 2024). DFlash emits an entire block in one forward, but its vanilla decoder still verifies essentially one trajectory per round (Chen et al., 2026).

Trees from parallel and semi-autoregressive drafts. DDTree constructs a budgeted tree from the per-position distributions of a block-diffusion drafter (Ringel & Romano, 2026), while TAPS selects a prefix-closed subtree using a lightweight target-aware scorer (Wang et al., 2026). DSpark instead combines a parallel backbone with a low-cost Markov head to restore within-block dependence (Cheng et al., 2026), but its original decoder materializes only one chain. PCTree reuses

Table 1: Conceptual comparison of parallel-block drafting strategies. DSpark and PCTree share the same parallel–Markov drafter but construct a single chain and a budgeted tree, respectively.

	DFlash	DDTree	DSpark	PCTree
Draft model	block diffusion	block diffusion	parallel+Markov	parallel+Markov
Candidate structure	chain	budgeted tree	chain	budgeted tree
Score source	per-position p_d	per-position p_d	Markov-refined logits	Markov-refined logits
Parent-specific scoring	no	no	one active parent	**batched parents**
Extra training for tree	–	no	–	**no**
Seq. draft stages / round	1 backbone	1 backbone	1 backbone + B Markov	1 backbone + B Markov

the same pretrained Markov head to score concrete parents separately, without additional training. Table 1 summarizes these factorization differences.

Semi-autoregressive speculative decoding. The Markov formulation in DSpark is particularly suitable for this conversion: its first-order state permits batched scoring of multiple frontier parents at each depth. PCTree therefore differs from DSpark in inference policy rather than draft-model training or backbone computation.

3 PRELIMINARIES

3.1 SPECULATIVE DECODING

Speculative decoding uses a lightweight draft model $\mathcal{M}_D$ to propose a block and a target model $\mathcal{M}_T$ to verify it in one forward pass (Leviathan et al., 2023; Chen et al., 2023). It commits the longest accepted prefix, so efficiency depends on draft cost, verification cost, and the number of tokens advanced per round.

3.2 DSPARK: SEMI-AUTOREGRESSIVE DRAFTING

We next recall the chain-style DSpark decoder (Cheng et al., 2026) that our method generalizes. Let $\mathcal{M}_D$ be a DSpark draft model with block size B and let x_t be the last verified token. Drafting has a parallel backbone stage followed by a lightweight sequential stage.

Parallel backbone. The DFlash-style backbone (Chen et al., 2026) processes the anchor token x_t and $B-1$ mask placeholders in one forward pass, producing base logits $L = (L_0, \ldots, L_{B-1})$ for B draft positions. Because the positions are computed in parallel, L_d does not depend on the concrete tokens selected earlier in the block.

Sequential Markov head. DSpark restores within-block dependence with a low-rank Markov head that adds a bias determined by the preceding token. Starting from $y_{-1} := x_t$, it applies the head sequentially for $d = 0, \ldots, B-1$:

$$z_d = L_d + \mathrm{Markov}(y_{d-1}), \qquad y_d = \arg\max_{v \in \mathcal{V}} (z_d)_v. \tag{1}$$

This produces one chain $(y_0, \ldots, y_{B-1})$: one backbone forward followed by B sequential batch-1 Markov calls (fig. 1).

We follow DSpark's default Markov head rather than its more complex RNN head; the same head is shared across all B sequential stages, and its first-order state permits batched scoring over multiple parents.

4 PCTREE

The top-1 choice in eq. (1) produces a single chain. PCTree replaces this choice with budgeted branching while reusing the same backbone logits, Markov operator, and pretrained parameters.

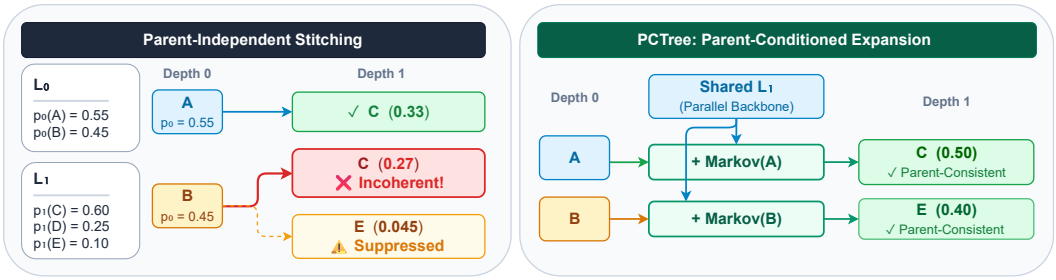

Figure 2: **Parent-independent stitching versus parent-conditioned expansion.** Reusing one depth-wise distribution can favor the incoherent path $B{\to}C$ and suppress $B{\to}E$ (left). PCTree reconditions the shared logits on each parent, recovering parent-consistent branches (right). Values are illustrative joint path probabilities.

Failure mode of parent-independent trees. Parallel block drafters produce one distribution for each future position in a single forward pass (Chen et al., 2026). A tree can be constructed directly from these depth-wise distributions, as in DDTree (Ringel & Romano, 2026). Without an additional parent-conditioned refinement, every parent at depth $d{-}1$ is expanded with the same distribution q_d; we refer to this as *parent-independent per-position scoring*. Figure 2 illustrates its possible failure mode: individually likely tokens can be combined into a locally incoherent path. In the example, the high position-wise score of C causes $B{\to}C$ to outrank the parent-consistent continuation $B{\to}E$, even though C is implausible after B. The probabilities are illustrative.

DSpark already provides the missing conditional signal: its Markov head can re-score the shared logits for each concrete parent. We use this signal to construct the tree described next.

4.1 Markov Tree Expansion

Let k be the local branching factor and N the verification node budget (including the root). We index Markov stages by $d = 0, \dots, B-1$; stage d constructs non-root tree level $d+1$. Thus, block size B is also the maximum number of drafted tree levels. Let $\mathcal{F}_{d-1}$ be the parent frontier used at stage d, with $\mathcal{F}_{-1} = \{x_t\}$ at the root. Its batch size is $b_d = |\mathcal{F}_{d-1}|$, with $b_0 = 1$ and $1 \le b_d \le k$. Every parent $p \in \mathcal{F}_{d-1}$ shares the same base logits L_d but receives its own Markov bias:

$$z_d(p) = L_d + \mathrm{Markov}(p), \qquad \log \pi_d(\cdot \mid p) = \log \mathrm{softmax}\big(z_d(p)\big). \tag{2}$$

Equivalently, stack the frontier tokens into $P_{d-1} = (p_1, \dots, p_{b_d}) \in \mathcal{V}^{b_d}$. At sequential Markov stage d, one call batches the current b_d parents and computes

$$Z_d = \mathbf{1}_{b_d} L_d^\top + \mathrm{Markov}(P_{d-1}) \in \mathbb{R}^{b_d \times |\mathcal{V}|}, \tag{3}$$

where the shared L_d is broadcast across the b_d frontier parents.

Each candidate node c is ranked by its *joint path score*

$$s(c) = \sum_{(u \to v) \in \mathrm{path}(c)} \log \pi_{d(u)}\big(v \mid u\big), \tag{4}$$

where $d(u)$ is the Markov stage that expands parent u. We take the local top-k children of each parent under $\log \pi_d(\cdot \mid p)$ and append them to a candidate pool with updated scores $s(\mathrm{child}) = s(p) + \log \pi_d(\mathrm{child} \mid p)$. Without pruning, the frontier entering stage d would contain up to k^d parents, and expanding it would produce up to k^{d+1} nodes. This exponential growth is prohibitive for large block sizes. Following the score-guided dynamic-tree principle of EAGLE-2 (Li et al., 2024b), we allocate expansion capacity to the most promising current nodes: after adding all children to the pool, only the k candidates with the highest joint scores become the next frontier (fig. 3). The frontier is therefore bounded by k, and the number of candidate nodes added to the pool is at most $k + (B-1)k^2$ rather than growing exponentially with depth. This bound matters especially for parallel drafters with large B. Without layer-wise pruning, the number of active parents would grow

Algorithm 1 PCTree draft construction (training-free)

Require: Draft model $\mathcal{M}_D$, root token x_t, block size B, top-k, budget N

$L \leftarrow$ one parallel-block forward of $\mathcal{M}_D$ on $(x_t, \text{MASK}, \ldots)$
frontier $\leftarrow \{x_t\}$ with joint score 0; pool $\leftarrow \{x_t\}$
for $d = 0$ to $B - 1$ **do**
    Sequential stage d: batch all $p \in$ frontier in one Markov call and compute $z_d(p) \leftarrow L_d + \text{Markov}(p)$
    Expand local top-k children; push into pool with updated joint scores (4)
    if $d < B - 1$ **then**
        frontier $\leftarrow$ top-k nodes of the current layer by joint score
    end if
end for
$S \leftarrow$ global top-$\min\{N, |\text{pool}|\}$ pool nodes by (score desc., depth asc., stable ID)
return draft tokens, retrieve indices, and tree attention mask induced by S

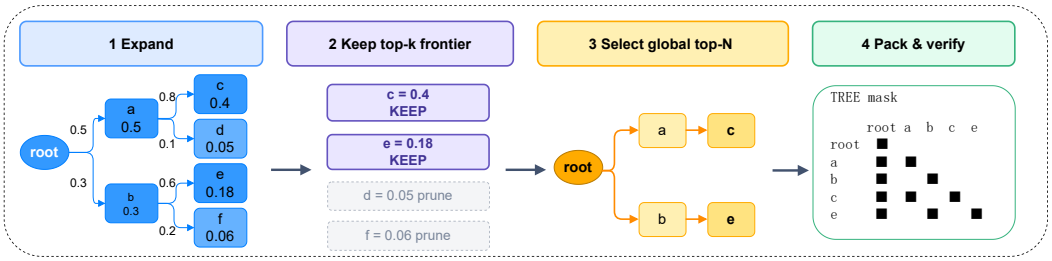

Figure 3: **Budgeted tree construction and verification.** Each frontier parent proposes top-k Markov-conditioned children; layer-wise pruning controls expansion, and global top-N selection yields an ancestor-closed tree. The selected nodes are packed with an ancestor-only mask for one target verification.

rapidly, and all of their previous-token IDs would enter the next batched Markov call. An overly wide frontier would therefore increase both the candidate count and the latency of every subsequent Markov stage. Thus the Markov head is still called once per depth—B calls in total—but its batch size is $b_d \in \{1, \ldots, k\}$ (typically k after the first step), rather than always 1.

Relation to dynamic trees and beam search. Frontier pruning is beam-like, but it controls only which nodes are expanded. Unlike beam search, which returns terminal sequences, we retain scored internal nodes because they are themselves verification candidates and ancestors of deeper nodes. This follows EAGLE-2's score-guided dynamic-tree principle (Li et al., 2024b), but uses parent-conditioned Markov scores over shared block logits.

4.2 Budgeted Selection and Tree Verification

After B stages, we rank all candidate-tree nodes, including the root, by decreasing joint score, breaking ties by shallower depth and then a stable node identifier, and retain the global top $\min\{N, |\text{pool}|\}$ nodes. The root has score $s(x_t) = 0$ and is therefore always selected. Because $s(c) = s(p) + \log \pi_{d(p)}(c \mid p) \leq s(p)$, every ancestor precedes its descendants, so the selected nodes form a prefix-closed tree (Li et al., 2024b).

The resulting set induces: (1) a flat draft-token tensor for the target; (2) retrieve indices enumerating root-to-leaf paths; (3) an ancestor-only tree attention mask for a single target forward. The packed tree is then verified in one target-model forward using the standard greedy tree speculative-decoding rule (Miao et al., 2024; Li et al., 2024b).

Algorithm 1 summarizes the procedure. When k=1, expansion collapses to the unique greedy child at every depth and recovers DSpark (up to the global budget N).

4.3 Why Trees Help and Complexity

Nodes at a fixed depth share L_d; branches from different parents differ through parent-specific Markov biases, so branching reuses the parallel backbone. Let C_{bb} be the backbone cost and $C_{\mathrm{mk}}(b)$ the Markov cost at batch size b. DSpark costs $C_{\mathrm{bb}} + \sum_{d=0}^{B-1} C_{\mathrm{mk}}(1)$, whereas PCTree costs $C_{\mathrm{bb}} + \sum_{d=0}^{B-1} C_{\mathrm{mk}}(b_d)$ with $b_0{=}1$ and $b_d \leq k$. In addition to batched Markov projections, tree construction incurs vocabulary-wide normalization and top-k, candidate management, and packing overhead; these costs are implementation- and hardware-dependent. Thus it preserves B sequential stages and increases only bounded Markov batches.

5 Experiments

PCTree changes only the inference policy. Within each matched comparison, DSpark and PCTree use the same draft checkpoint, with no additional training for tree expansion.

5.1 Setup

Models. We evaluate Qwen3-{4B,8B,14B} (Yang et al., 2025). The $B{=}7$ experiments use the official DSpark checkpoints released by DeepSeek. For $B{=}16$, we train target-matched drafts with the official DSpark recipe for each target size. Checkpoint and training details are given in section A.

Baselines. DSpark (Cheng et al., 2026) is the matched control and differs from PCTree only in chain versus tree expansion. EAGLE-3 (Li et al., 2025b) and DFlash (Chen et al., 2026) are published references reproduced from Cheng et al. (2026), not strict same-environment speed comparisons.

Decoding hyperparameters. All matched main comparisons use a fixed reference configuration with local branching factor $k{=}4$ and verification budget $N{=}32$ (including the root). This reference budget is held constant across models and benchmarks and is not claimed to maximize throughput on every workload; the full budget sweep is reported in section E. The chain baseline is **DSpark** only; $k{=}1$ is mathematically the same greedy Markov chain and is not evaluated as a second baseline. We report greedy target verification. Confidence scheduling is disabled, so every comparison uses the same fixed verification policy.

Metrics. Following Cheng et al. (2026), mean acceptance length τ includes the target bonus token. We report end-to-end speedup relative to the specified HuggingFace target-only autoregressive baseline, which generates one token per step without speculative drafting. This absolute ratio depends on the baseline implementation; the PCTree–DSpark comparison uses the same speculative harness and therefore isolates their relative gain.

Hardware. All latency and AR-speedup measurements use a single NVIDIA H20 with bfloat16 and SDPA attention. Additional checkpoint identifiers are listed in section A.

Measurement protocol and uncertainty. The Qwen3-4B $B{=}16$ main configurations are averaged over three full runs. For the smallest original margin, Qwen3-4B on GSM8K at $B{=}7$, we likewise ran DSpark and PCTree three times with the same prompts, prompt order, generated-token cap, checkpoint, and decoding settings. Other main-table configurations currently have one full run, so their decimal precision should not be interpreted as statistical confidence. The aggregate results appear in table 3; the individual repeated runs are reported in table 8 in the appendix. Each point in the verification-budget sweep is likewise repeated three times. Timing, warmup, and execution-order details are provided in section A.

Benchmarks. Following the DSpark evaluation protocol (Cheng et al., 2026), we cover **Math**: GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2024), and AIME25 as distributed by MathArena (Balunović et al., 2025); **Code**: MBPP (Austin et al., 2021), HumanEval (Chen et al., 2021), and LiveCodeBench (LCB) (Jain et al., 2025); **Chat**: MT-Bench (Zheng et al., 2023), Alpaca (Taori et al., 2023), and Arena-Hard (Li et al., 2025a).

Table 2: Mean acceptance length τ. †: published values reproduced from Cheng et al. (2026), not strict same-environment comparisons. Non-dagger rows are our matched runs; bold marks PCTree.

		Math			Code			Chat		
Target	Drafter	GSM8K	MATH-500	AIME25	MBPP	HumanEval	LCB	MT-Bench	Alpaca	Arena-Hard
Qwen3-4B	EAGLE-3†	5.14	4.62	3.92	3.69	4.16	3.77	2.39	2.26	2.55
	DFlash†	5.40	4.85	4.15	4.40	4.74	4.18	3.07	2.96	2.83
	DSpark (B=7)	6.31	6.24	5.47	5.33	5.60	5.30	3.82	3.67	3.81
	PCTree (B=7)	**7.24**	**7.22**	**6.77**	**6.53**	**6.78**	**6.30**	**5.07**	**4.83**	**4.83**
	DSpark (B=16)	9.41	9.03	6.87	6.90	7.42	6.71	4.28	3.99	4.30
	PCTree (B=16)	**11.16**	**10.85**	**8.80**	**8.43**	**9.21**	**8.05**	**5.50**	**5.21**	**5.44**
Qwen3-8B	EAGLE-3†	5.30	4.77	3.91	3.96	4.33	4.17	2.66	2.54	2.54
	DFlash†	5.33	4.91	4.07	4.36	4.64	4.39	3.11	2.98	2.81
	DSpark (B=7)	6.46	6.35	5.57	5.46	5.85	5.48	3.87	3.77	3.88
	PCTree (B=7)	**7.36**	**7.28**	**6.77**	**6.68**	**6.96**	**6.51**	**4.98**	**4.93**	**4.91**
	DSpark (B=16)	9.75	9.13	6.93	6.97	7.88	6.80	4.35	4.07	4.37
	PCTree (B=16)	**11.50**	**10.93**	**8.74**	**8.76**	**9.65**	**8.28**	**5.63**	**5.40**	**5.52**
Qwen3-14B	EAGLE-3†	5.24	4.60	3.71	3.81	4.14	4.01	2.62	2.47	2.48
	DFlash†	5.41	4.84	3.98	4.44	4.59	4.33	3.10	2.94	2.72
	DSpark (B=7)	6.48	6.41	5.76	5.48	5.79	5.45	3.97	3.76	3.92
	PCTree (B=7)	**7.38**	**7.30**	**6.82**	**6.71**	**6.93**	**6.50**	**5.14**	**4.92**	**4.92**
	DSpark (B=16)	9.94	9.22	7.11	7.04	7.72	6.77	4.43	4.08	4.36
	PCTree (B=16)	**11.61**	**10.94**	**9.12**	**8.81**	**9.54**	**8.19**	**5.72**	**5.34**	**5.54**

Table 3: Matched gains over DSpark on three representative tasks (Qwen3-4B, k=4, N=32). All B=16 entries and GSM8K at B=7 report three-run means. Deltas use unrounded measurements.

Suite	B	τ_{DS}	τ_{PC}	$\Delta\tau$	AR speedup (DSpark/PCTree)	Δ AR speedup (%)
GSM8K	7	6.31	7.24	+0.93 (+14.8%)	4.24×/4.50×	+6.1%
	16	9.41	11.16	+1.75 (+18.6%)	6.14×/6.60×	+7.5%
HumanEval	7	5.60	6.78	+1.18 (+21.2%)	3.74×/4.27×	+14.3%
	16	7.42	9.21	+1.79 (+24.1%)	4.89×/5.47×	+11.8%
MT-Bench	7	3.82	5.07	+1.24 (+32.5%)	2.48×/3.20×	+29.3%
	16	4.28	5.50	+1.23 (+28.8%)	2.82×/3.25×	+15.1%

5.2 Main Results: Acceptance Length

Table 2 reports mean acceptance length τ for all drafters. PCTree improves τ across all nine benchmarks for each evaluated target/block-size setting. Table 3 reports the corresponding Qwen3-4B acceptance and AR-speedup gains on three representative tasks, including the three-run B=16 measurements and the repeated GSM8K measurement at B=7. AR-speedup results for all models and benchmarks are deferred to table 7 in the appendix.

Relative improvement by block size. Table 3 reports *absolute* τ and AR target-only speedup on GSM8K, HumanEval, and MT-Bench. Across suites, PCTree consistently raises τ at B=7. On GSM8K, HumanEval, and MT-Bench, the corresponding τ gains are 14.8%, 21.2%, and 32.5%, respectively. The corresponding AR-speedup gains are 6.1%, 14.3%, and 29.3%; the GSM8K speedup is the three-run mean, while the other short-block measurements are single runs. At B=16, PCTree raises τ by 18.6%–28.8% and three-run mean AR speedup by 7.5%–15.1% relative to matched DSpark on the same three representative suites. The detailed phase-cost breakdown is reported in section D; the wider Markov stage costs more per round, but the longer accepted paths reduce the number of proposal rounds enough to improve overall AR speedup.

Scaling across target sizes. The acceptance gains in table 2 persist from Qwen3-4B to Qwen3-8B and Qwen3-14B at both block sizes. This consistency indicates that the benefit is not confined to one target scale. The complete AR-speedup results for these larger targets are reported in table 7.

Table 4: Mechanism isolation on GSM8K (B=16, k=4, N=32). The DSpark tree controls share the checkpoint, search, packing, and verifier; only parent-specific reconditioning differs. DFlash+DDTree uses an external checkpoint and is a system-level reference. Utilization is the committed fraction of verified non-root nodes.

Construction	Mean τ	Verified-node utilization (%)	Rounds/sample
Within-checkpoint DSpark controls			
DSpark	9.410	53.78	26.812
Shared-Markov tree	10.225	29.88	24.718
PCTree	**11.156**	32.90	**22.632**
External tree baseline (different draft checkpoint)			
DFlash+DDTree (B=16)	7.485	20.92	33.826

5.3 Mechanism Isolation: Shared vs. Parent-Conditioned Markov Trees

Figure 2 illustrates that reusing one position-wise distribution under different parents can stitch individually likely tokens into an incoherent path. We test this failure mode on model outputs using three variants that share the DSpark checkpoint, prompts, block size B, target verifier, and packing code, thereby separating the effect of branching from that of parent-specific reconditioning:

1. **DSpark**: the original top-1 Markov chain;
2. **Shared-Markov tree**: first construct the greedy DSpark reference chain using eq. (1). At each depth, reuse its Markov-refined distribution q_d for every tree parent. Thus different parents receive the same distribution and child ranking: $q_d(\cdot \mid p) = q_d(\cdot \mid p') = q_d(\cdot)$;
3. **PCTree**: following eq. (2), parent-conditioned scoring computes a separate distribution for every concrete frontier parent. Parents at the same depth share L_d but may induce different child rankings. Parent computations within a depth are batched, while the B depth-wise Markov stages remain sequential.

These variants form a controlled comparison: Shared-Markov removes only parent-specific reconditioning, and both tree variants use the same frontier width, budget, selection rule, packing implementation, and target verifier. Separately, DFlash+DDTree is an external reference under the same target, GSM8K protocol, and N=32 budget, but uses a different drafter and checkpoint; we therefore use the official Qwen3-4B DFlash B=16 checkpoint only for system-level context.

We report mean acceptance length, rounds per sample, accepted-depth survival $S(d) = \Pr[A \geq d]$, and verified non-root utilization (the fraction of verified non-root nodes committed to the output path). Figure 4 and table 4 report these measurements.

Mechanism result. Under the same DSpark checkpoint and 32-node budget, PCTree increases mean acceptance length over Shared-Markov from 10.225 to 11.156 (+9.1%), raises verified-node utilization from 29.88% to 32.90%, and reduces rounds per sample from 24.718 to 22.632 (−8.4%). Survival at depths 8 and 12 rises from 57.1% to 64.8% and from 40.3% to 46.8%, respectively. Since the controls differ only in reconditioning, these gains isolate the effect of conditioning on the concrete parent.

External tree reference. The DFlash+DDTree curve ends at d=15 because its official B=16 convention uses one fixed root position and 15 masked future positions. Its results provide external context, but differences from the DSpark variants cannot be attributed solely to tree scoring because the draft architecture and checkpoint also change.

5.4 Ablation Studies

All ablations use matched large-block drafts at B=16 and report GSM8K. The branching-factor sweep uses Qwen3-4B as a representative target. We focus on B=16, where the larger candidate space makes branching choices most consequential.

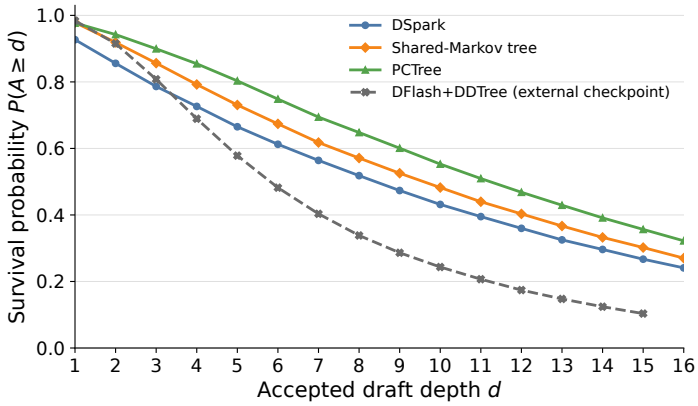

Figure 4: **Accepted-depth survival on GSM8K.** Each curve gives $S(d) = \Pr[A \geq d]$. The DSpark variants share the checkpoint, 32-node budget, and search policy; Shared-Markov reuses one distribution per depth, whereas PCTree reconditions on each parent. DFlash+DDTree (gray dashed) is a nonmatched external reference.

Table 5: Ablation on branching factor k (B=16, N=32, GSM8K). k=1 ≡ DSpark (chain); k≥2 are tree policies. The shared k=1, 4 endpoints use the three-run main measurements.

Target	Metric	1 (≡ DSpark)	2	4	8
Qwen3-4B	Mean τ	9.41	10.85	**11.16**	11.14
	AR speedup	6.14×	6.58×	**6.60**×	6.59×

5.4.1 Branching factor k

Table 5 varies $k \in \{1, 2, 4, 8\}$ at fixed N=32 on Qwen3-4B with B=16; k=1 is the DSpark chain endpoint. Increasing k from 1 to 2 yields most of the acceptance and AR-speedup gain, while performance saturates at the default k=4 and k=8 adds no benefit beyond timing variability.

5.4.2 Verification budget N

At fixed k=4, increasing N improves acceptance until saturation, whereas AR speedup is non-monotonic because additional tree construction and verification eventually outweigh the acceptance gain. We use N=32 as a common operating point; the full sweep is reported in section E.

6 Limitations

There are some limitations in this work. (1) Tree quality remains bounded by the pretrained draft distributions; a training-free expansion policy cannot fix a systematically weak Markov head. (2) Very large k or N can make draft construction and verify attention non-negligible on small targets or memory-tight devices. (3) Although AR speedup improves in the present BF16 runs, the gain depends on hardware, precision, attention kernels, and the tree budget.

7 Conclusion

We presented PCTree, a training-free method that turns DSpark's semi-autoregressive chain drafts into budgeted draft trees. By expanding top-k Markov children on shared backbone logits and verifying the resulting tree in one target forward, PCTree increases acceptance length without any retraining. AR-speedup measurements suggest that the acceptance gains can translate into end-to-end improvements at B=7 and B=16 across Qwen3-{4B,8B,14B}. More generally, a parallel drafter with cheap parent-dependent refinement can form tree proposals at inference time, with the largest gains when early rejections waste long chain suffixes.

AI USE STATEMENT

We used generative AI tools to polish the writing and improve the visual presentation of figures. We did not use generative AI tools to generate experimental data, design the methodology or experiments, implement the method, or interpret the results. All authors reviewed and verified the final manuscript and take responsibility for its content.

REPRODUCIBILITY STATEMENT

The draft factorization, tree search and selection rules, verification packing, and decoding settings are specified in Section 4; the experimental protocol, dataset sources, software versions, and hardware are documented in Appendix A. We report repeated timing measurements where available and identify the remaining single-run measurements in section 5. The short-block DSpark checkpoints and the external DFlash baseline are public and linked in section A. The $B{=}16$ DSpark drafts are internally trained and are not public at submission; we plan to release anonymized training configurations, data-processing details, evaluation scripts, and the corresponding artifacts in accordance with the conference's anonymity policy.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1d27c01ebd3e3aebe226b44fc970d803-Abstract-Datasets_and_Benchmarks_Track.html.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 5209–5235. PMLR, 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023. URL https://arxiv.org/abs/2302.01318.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*, 2026. URL https://arxiv.org/abs/2602.06036. Accepted at ICML 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, Qinyu Chen, Chengqi Deng, Jiping Yu, Damai Dai, Zhengyan Zhang, Yixuan Wei, Yixuan Tan, Wenkai Yang, Runxin Xu, Yu Wu, Zhean Xu, Xuanyu Wang, Muyang Chen, Rui Tian, Xiao Bi, Zhewen Hao, Shaoyuan Chen, Huanqi Cao, Wentao Zhang, Anyi Xu, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. DSpark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026. URL https://arxiv.org/abs/2607.05147.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve

math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15706–15734. PMLR, 2024. URL https://proceedings.mlr.press/v235/gloeckle24a.html.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida I. Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-Hard and benchbuilder pipeline. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 34209–34231. PMLR, 2025a. URL https://proceedings.mlr.press/v267/li25h.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR, 2024a. URL https://proceedings.mlr.press/v235/li24bt.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432, 2024b. doi: 10.18653/v1/2024.emnlp-main.422.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. doi: 10.52202/085713-4562. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c7b5a35ea98b62512a869c19ea7b03cb-Abstract-Conference.html.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949, 2024. doi: 10.1145/3620666.3651335.

Liran Ringel and Yaniv Romano. Accelerating speculative decoding with block diffusion draft trees. *arXiv preprint arXiv:2604.12989*, 2026. URL https://arxiv.org/abs/2604.12989.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/c4127b9194fe8562c64dc0f5bf2c93bc-Abstract.html.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following LLaMA model. GitHub repository, 2023. URL https://github.com/tatsu-lab/stanford_alpaca.

Chunqi Wang, Ji Zhang, and Haiqing Chen. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 479–488. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1044. URL https://aclanthology.org/D18-1044/.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. OPT-Tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199, 2025. doi: 10.1162/tacl_a_00735.

Zhuoyu Wang, Junnan Huang, and Xinyu Chen. TAPS: Target-aware prefix tree selection for diffusion-drafted speculative decoding. *arXiv preprint arXiv:2606.00487*, 2026. URL https://arxiv.org/abs/2606.00487.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3909–3925, 2023. doi: 10.18653/v1/2023.findings-emnlp.257.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

Table 6: Evaluation datasets and their upstream releases. "Rows" denotes the size of the corresponding evaluation file.

Dataset	Upstream release	Rows
GSM8K	openai/gsm8k	1,319
MATH-500	HuggingFaceH4/MATH-500	500
AIME25	MathArena/aime_2025	30
MBPP	google-research-datasets/mbpp	257
HumanEval	openai/openai_humaneval	164
LiveCodeBench	livecodebench/code_generation_lite	1,055
MT-Bench	HuggingFaceH4/mt_bench_prompts	80
Alpaca	tatsu-lab/alpaca	52,002
Arena-Hard	lmarena-ai/arena-hard-auto	750

A Full Experimental Protocol

Prompts and decoding. We apply the target model's chat template with thinking disabled and evaluate the first user turn. Decoding is greedy with batch size one, a 512-token output cap, and the model's EOS stopping rule. DSpark and PCTree use identical prompts, ordering, and stopping criteria. The pre-specified reference tree uses $k{=}4$ and $N{=}32$ including the root; confidence scheduling and near-tie re-verification are disabled. The target and draft models, their KV caches, and hidden states use bfloat16.

Evaluation data. Table 6 records the upstream releases used by our evaluation harness.

Software and hardware. Experiments run on one NVIDIA H20 in bfloat16 using PyTorch 2.11, Transformers 5.5, CUDA 12.8, and SDPA attention. End-to-end timing excludes model and dataset loading but includes prompt processing, data transfer, and generation.

Warmup and repeated timing. We perform three matched full-run repetitions for all Qwen3-4B $B{=}16$ main configurations and for Qwen3-4B GSM8K at $B{=}7$. Other main-table configurations currently have one run; each configuration in the verification-budget sweep is also repeated three times. Model and dataset loading occur before timing, but no prompt-level warmup is used, so the first prompt is included in the measured loop. The dedicated $B{=}7$ repeatability run alternates DSpark/PCTree execution order; GPU persistence mode and application clocks remain at the host defaults. Each run reports total generated tokens divided by end-to-end wall time, and AR speedup is computed within each repetition before reporting mean$\pm$standard deviation.

Checkpoints. Target and draft checkpoint identifiers:

- Target models: Qwen/Qwen3-4B, Qwen/Qwen3-8B, and Qwen/Qwen3-14B;
- Official short-block DSpark drafts ($B{=}7$) for Qwen3-4B, Qwen3-8B, and Qwen3-14B: 4B, 8B, and 14B;
- Qwen3-{4B,8B,14B} + internally trained large-block DSpark drafts ($B{=}16$): separate target-matched checkpoints trained with the official DSpark recipe on mlabonne/open-perfectblend after regenerating assistant responses with the corresponding target model.
- DFlash+DDTree external baseline: official Qwen3-4B DFlash $B{=}16$ checkpoint z-lab/Qwen3-4B-DFlash-b16.

Tensor implementation. Selected tree nodes are stored in a flat token tensor. Each node's position identifier is its tree depth plus the verified prefix length, so siblings share a position while retaining distinct ancestry. The tree mask permits a node to attend to the verified prefix, itself, and its root-to-node ancestors. Retrieve indices enumerate padded root-to-leaf rows used to compare each proposed child against the preceding target logit; after selecting the longest accepted row, KV tensors and target hidden states are gathered along that row and compacted after the verified prefix.

B Complete AR-Speedup Results

Table 7: End-to-end AR speedup for all nine tasks. Each pair uses a matched DSpark/PCTree harness. Qwen3-4B B=16 and Qwen3-4B GSM8K at B=7 report three-run means.

	4B, B=7		4B, B=16		8B, B=7		8B, B=16		14B, B=7		14B, B=16	
Suite	DSpark	**PCTree**	DSpark	**PCTree**	DSpark	**PCTree**	DSpark	**PCTree**	DSpark	**PCTree**	DSpark	**PCTree**
GSM8K	4.24×	**4.50×**	6.14×	**6.60×**	4.43×	**4.86×**	6.66×	**7.09×**	4.28×	**4.88×**	6.93×	**7.21×**
MATH-500	4.21×	**4.34×**	6.02×	**6.37×**	4.36×	**4.71×**	6.26×	**6.63×**	4.37×	**4.91×**	6.20×	**6.76×**
AIME25	3.65×	**4.17×**	4.51×	**5.07×**	3.86×	**4.34×**	4.71×	**5.39×**	3.97×	**4.46×**	4.78×	**5.46×**
MBPP	3.61×	**4.06×**	4.67×	**5.04×**	3.75×	**4.50×**	4.64×	**5.58×**	3.75×	**4.38×**	4.70×	**5.55×**
HumanEval	3.74×	**4.27×**	4.89×	**5.47×**	4.17×	**4.60×**	5.45×	**5.83×**	4.08×	**4.69×**	5.40×	**5.62×**
LCB	3.49×	**3.74×**	4.34×	**4.51×**	3.61×	**4.01×**	4.30×	**4.58×**	3.45×	**3.75×**	4.07×	**4.36×**
MT-Bench	2.48×	**3.20×**	2.82×	**3.25×**	2.60×	**3.34×**	3.00×	**3.50×**	2.74×	**3.42×**	3.10×	**3.47×**
Alpaca	2.39×	**3.02×**	2.69×	**3.09×**	2.65×	**3.26×**	2.76×	**3.37×**	2.61×	**3.38×**	2.89×	**3.40×**
Arena-Hard	2.52×	**2.93×**	2.86×	**3.16×**	2.67×	**3.12×**	2.98×	**3.38×**	2.64×	**3.16×**	2.91×	**3.30×**

C Repeatability Check

We repeat the Qwen3-4B B=16 main evaluation across all nine tasks. We also repeat GSM8K at B=7 because it had the smallest AR-speedup margin in the initial evaluation and was therefore the most susceptible to timing noise. Within each matched repetition, DSpark and PCTree use the same checkpoint, evaluation inputs in the same order, generated-token cap, decoding settings, and execution environment. Table 8 reports the GSM8K runs individually; the main tables retain the more compact three-run means.

Table 8: Repeatability of Qwen3-4B GSM8K (k=4, N=32). Mean±standard deviation is computed over three full runs.

B	Run	DSpark	PCTree	Gain
7	1	4.31×	4.54×	+5.4%
	2	4.22×	4.47×	+5.8%
	3	4.18×	4.48×	+7.1%
	Mean±std.	4.24±0.07×	**4.50±0.04×**	**+6.1±0.9%**
16	1	6.09×	6.64×	+9.0%
	2	6.22×	6.57×	+5.7%
	3	6.10×	6.57×	+7.8%
	Mean±std.	6.14±0.07×	**6.60±0.04×**	**+7.5±1.7%**

D Phase Cost Breakdown

Table 9 decomposes average per-round latency on GSM8K. Because PCTree changes only candidate expansion, its backbone cost matches DSpark at each block size. Branching raises the Markov+tree cost, but longer accepted paths reduce the average number of rounds per sample: from 39.9 to 35.1 at B=7 and from 26.8 to 22.6 at B=16. The Markov+tree measurement includes selection, pruning, bookkeeping, and packing in addition to the Markov projection; consequently, k=4 does not imply exactly 4× DSpark's batch-1 Markov latency.

E Verification-Budget Sweep

Figure 5 separates acceptance gains from their end-to-end cost by sweeping N at fixed k=4. For B=7, this search generates at most 101 nodes, so we omit the N=128 point rather than pad the verification tree beyond its candidate pool. Increasing N generally lengthens the accepted prefix, but the improvement gradually saturates. AR speedup is therefore non-monotonic: it peaks at N=32 for B=7 and at N=64 for B=16, then declines as extra tree-construction and verification work outweighs the remaining acceptance gain. The sweep-specific optima are hardware- and workload-dependent operating points, rather than retroactive tuning of the main comparisons.

Table 9: Phase cost breakdown on GSM8K (Qwen3-4B, k=4, N=32). Phase latency is wall time per proposal round. AR speedups are three-run means; phase latencies are representative matched measurements.

				Avg. phase latency (ms/rd)		
B	Method	Mean τ	Avg. rounds/sample	Backbone	Markov+tree	AR speedup
7	DSpark	6.31	39.9	3.46	0.65	4.24×
	PCTree	7.24	35.1	3.46	3.52	**4.50**×
16	DSpark	9.41	26.8	3.55	1.09	6.14×
	PCTree	11.16	22.6	3.55	5.24	**6.60**×

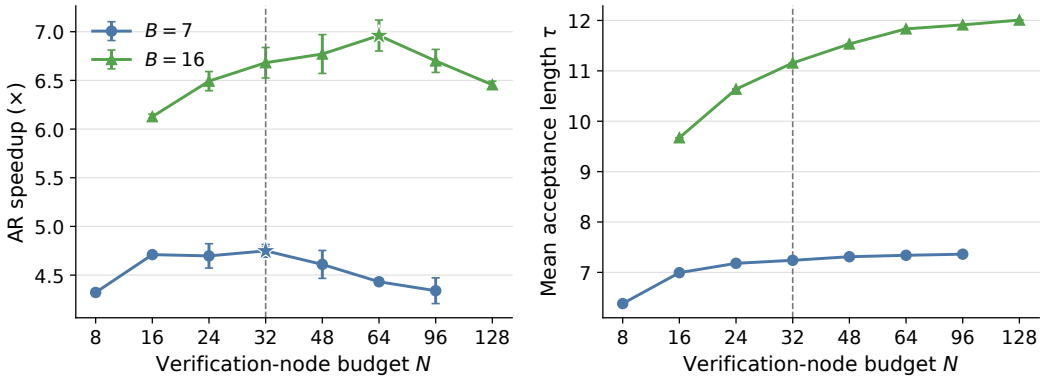

Figure 5: **Scaling with verification-node budget on GSM8K** (Qwen3-4B, k=4). Points are three-run means; AR-speedup error bars show one standard deviation. The dashed line marks the main setting N=32. For B=7, the N=128 point is omitted because the candidate pool is capped at 101 nodes. This sweep was timed independently from the main tables.