TREESPARK: FROM ACCEPTANCE SCORES TO USEFUL WORK IN CONDITIONAL SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates autoregressive language models by using a small drafter to propose tokens that a larger target verifies in parallel. Diffusion-based drafters can produce a whole block in one forward pass, and lightweight conditional heads can model dependencies within that block. Extending these proposals to trees covers more possible continuations, but the extra construction and verification work can outweigh the benefit of accepting more tokens. We study this trade-off with TreeSpark, a conditional-tree decoding framework built on the DSpark drafter. TreeSpark preserves a primary chain and adds optional leaves using already computed parent distributions and the drafter's existing confidence head. A threshold chosen from the active batch size controls leaf admission, while ordered rejection verifies the sampled tree. We evaluate this design against chain and fixed-tree controls in the same optimized inference engine. On Qwen3-4B, selected fixed trees improve throughput over chain by 5.0–8.6%; confidence-guided trees gain 5.8–6.7% in sampling and 2.4–3.3% under fixed-rate serving. Matched static and constant-threshold controls do not establish a consistent additional benefit from adaptation. The results show why useful tree allocation must account for executed work, alongside acceptance predictions.

1 INTRODUCTION

Speculative decoding proposes tokens with a small drafter and verifies them in parallel with a target model (Leviathan et al., 2023; Chen et al., 2023). Trees cover several continuations in one verification pass (Miao et al., 2024; Cai et al., 2024), but coverage is only useful when its accepted tokens repay the work of proposing and verifying additional branches. Which predictions help make this allocation decision, and which costs must the decision account for?

This question is particularly consequential for diffusion-based parallel drafting. DFlash computes a draft block in one backbone pass (Chen et al., 2026); lightweight sequential corrections in semi-autoregressive drafters then condition proposals on their parents (Huang et al., 2026; Cheng et al., 2026). PCTree and DominoTree already reuse this architecture to construct conditional trees (Li et al., 2026b; Lin & Jang, 2026). Avoiding another backbone pass, however, does not make another parent free: it still requires a conditional vocabulary projection. Two trees with the same number of candidate nodes can therefore have different construction costs, in addition to differences in target verification and padding.

Prediction, allocation, and execution are distinct. We investigate these three stages in a common inference engine. First, a candidate's greedy-match probability, the proposal's average acceptance probability, and a node's probability of being committed under recursive rejection are different prediction targets. Conditioning a scalar fit on surviving greedy contexts improves its intended reliability diagnostic; applying that score to sampled trees introduces a separate hypothesis. Second, even accurate aggregate accepted-length predictions need not identify useful individual branches. We give a finite-vocabulary example where identical proposal–target overlap gives different optional-sibling values after observing the primary proposal. Third, an allocation with higher accepted length can lose in tokens/s when it prepares more parent distributions or verifies more positions.

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.

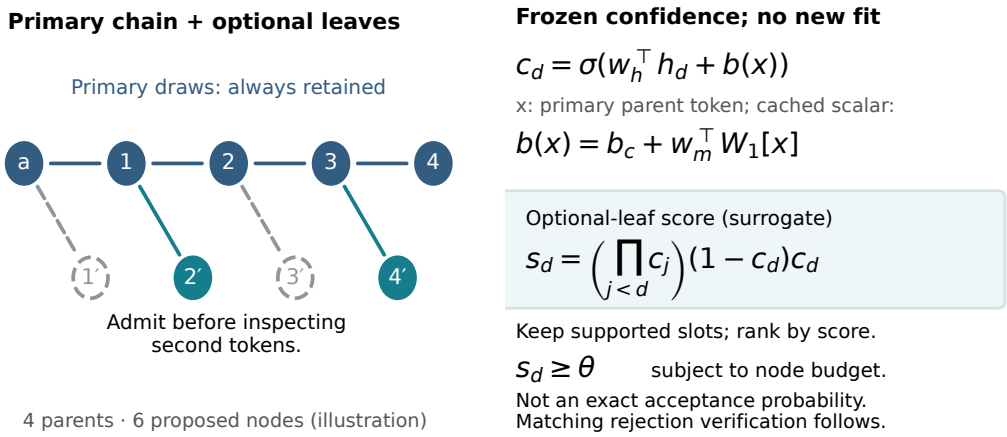

Figure 1: A fixed-parent construction separates branch allocation from additional conditional projections. The primary chain supplies the parent distributions; optional second siblings remain leaves. Static depth ordering and a frozen-confidence score select among these slots in the same builder. The illustration shows a four-depth example, not measured probabilities or an empirical speedup. Our experiments use up to seven draft depths.

TreeSpark provides a controlled study of these distinctions. We compare independently tuned raw, calibrated, depth-adjusted, and fixed-tree policies, then study the fixed-parent restriction in Figure 1. Its confidence variant reuses the released drafter head without fitting a new predictor; its static control uses the same execution path. This comparison tests the incremental value of content-dependent selection separately from the efficiency of the builder. The topology, confidence head, and recursive rejection mechanism build on existing work; we do not claim their invention.

Our contributions are a controlled test of calibration's decision value, explicit measurement of conditional-parent and target-node work, and correctness and diagnostic conditions for adaptive sampled trees. The evaluation freezes policies on validation prompts before model, dataset, temperature, context-length, and stationary-load transfers. Load policies observe active batch size and choose a threshold; they do not observe the benchmark's arrival-rate parameter. We retain the chain, strongest selected fixed tree, and constant-threshold controls, since adaptive selection has no oracle guarantee of dominating them.

The primary comparison finds gains over chain from fixed trees in all four temperature/batch strata. Confidence admission also gains over chain in sampling, but its differences from the optimized static tree remain uncertain. Calibrated best-first trees lose throughput despite improved greedy reliability. The practical implication is to evaluate a score through the work it admits and the execution-matched policy it improves upon.

2 Related Work

Draft quality, tree structure, and verification. Classical speculative sampling preserves a target distribution through proposal acceptance and residual correction (Leviathan et al., 2023; Chen et al., 2023). SpecInfer and Medusa organize candidates into trees (Miao et al., 2024; Cai et al., 2024); EAGLE-2, OPT-Tree, and Sequoia improve their shapes using confidence, acceptance, or hardware models (Li et al., 2024; Wang et al., 2025; Chen et al., 2024). DistillSpec improves draft–target agreement through distillation (Zhou et al., 2024), while Group Tree Optimization aligns drafter training with a tree objective (Hu et al., 2026b). Block Verification and optimal multi-draft sampling study how verification itself affects acceptance (Sun et al., 2025; Hu et al., 2025). Our comparisons leave drafter weights fixed; recursive rejection supplies the correctness mechanism rather than a new verification algorithm.

Conditional trees from parallel drafts. DFlash generates parallel block proposals, and DDTree builds a fixed-budget tree from their marginals (Chen et al., 2026; Ringel & Romano, 2026). DSpark and Domino add inexpensive sequential corrections to parallel backbones (Cheng et al., 2026; Huang et al., 2026). PCTree reuses DSpark's Markov head for parent-conditioned trees and accounts for their increased Markov batch sizes; DominoTree uses a path-dependent GRU correction and reports both research-harness and SGLang results (Li et al., 2026b; Lin & Jang, 2026). JetSpec trains a causal parallel head (Hu et al., 2026a). DARTree expands a conditional supertree by depth and prunes it, studying depth penalties and load-indexed width and budget schedules (Li et al., 2026a). Parent conditioning, batched construction, and load-dependent budgets therefore precede this work.

Adaptive computation budgets. CaDDTree and BASTION balance marginal-tree acceptance surrogates against verification latency to select budgets (Zhang et al., 2026; Oh et al., 2026). DSpark calibrates cumulative chain survival with Sequential Temperature Scaling and allocates verification prefixes across requests using a profiled engine-throughput curve. It also shows why admission must not anticipate the admitted draw (Cheng et al., 2026). TurboSpec and AdaSpec also adapt speculation to serving conditions (Liu et al., 2025; Huang et al., 2025); practical gains depend on the execution environment (Liu et al., 2026). DominoTree identifies overestimated conditional path scores as a cause of budget saturation and leaves calibration as an open repair (Lin & Jang, 2026). We study the connection between the event a score predicts, the branches it selects, and the conditional work actually executed. Surviving-context calibration and released overlap confidence are two tested allocation hypotheses. The comparison includes optimized static trees and independently tuned budget rules; neither calibration nor adaptive threshold selection alone establishes throughput optimality.

3 Conditional Trees: Outcomes, Allocation, and Work

3.1 Conditional proposals and measured work

A round proposes N non-anchor nodes, then commits an accepted path and a correction or bonus token. We report committed tokens per request-round τ, excluding the prefill token. A batch can contain several such rounds. Following DSpark and conditional-tree construction (Cheng et al., 2026; Li et al., 2026b), one backbone supplies logits $U_{1:b}$. Its Markov head conditions depth d on parent token x_p:

$$q_{0,d}(v \mid x_p) = \mathrm{softmax}(U_d + B(x_p, \cdot))_v, \qquad q_{T,d}(v \mid x_p) = \mathrm{softmax}((U_d + B(x_p, \cdot))/T)_v, \tag{1}$$

where $B(x, v) = \langle W_1[x], W_2[v] \rangle$ and $T > 0$. Here q_0 is the unscaled softmax, not the zero-temperature limit. Our block-7 checkpoint uses rank 256; Appendix E specifies row alignment.

A parent requires a conditional vocabulary projection, shared by its siblings. We distinguish P_{tree}, parents with retained children (including the anchor), from P_{prep}, actual nonpadding parent-projection rows, and P_{exec}, executed rows including padding. Preparation can include parents without retained children. Target verification processes $N + 1$ positions before padding; thus equal N need not mean equal work or latency. All counters are aggregated per request-round, separately from synchronized timing profiles. For matched batch-round starting contexts, let L_0, C_0 be mean emitted tokens and elapsed time. For $C_0 > 0$ and $C_0 + \Delta C > 0$, the token–time ratio improves exactly when $\Delta L\, C_0 > L_0\, \Delta C$. This local cost comparison does not account for policy-induced changes to later prefixes, active batches, or queues.

3.2 Predictable verification

Greedy verification follows target-argmax matches with a fixed tie rule. For sampling, retain each parent's actual FP32 proposal q_v as a snapshot and apply ordered recursive rejection (Jeon et al., 2024). For children $x^{(1)}, \ldots, x^{(m)}$, start from the target $p_{v,1} = p_T(\cdot \mid \mathrm{prefix}(v))$ and use

$$\begin{aligned} q_v^{(j)} &= \mathrm{norm}\Big(q_v \mathbb{1}[\cdot \notin \{x^{(1)}, \ldots, x^{(j-1)}\}]\Big), \quad a_{v,j} = \min\{1, p_{v,j}(x^{(j)})/q_v^{(j)}(x^{(j)})\}, \\ p_{v,j+1} &= \mathrm{norm}([p_{v,j} - q_v^{(j)}]_+) \qquad\qquad \text{after rejection.} \end{aligned} \tag{2}$$

Acceptance descends; exhausting children samples the final residual, and an accepted leaf samples a target bonus. Allocation scores never replace q_v. We specialize DSpark's non-anticipating admission condition (Cheng et al., 2026) to ordered tree siblings and hidden prefetching.

Proposition 1 (Predictable adaptive expansion)**.** *Assume each node uses the target conditional for its own causal prefix. In exact arithmetic, a finite construction preserves that law if every admitted draw has its stated proposal conditional on exposed construction history, admission cannot inspect that draw or correlated hidden sampling randomness, all admitted nodes are retained in sibling order, and verifier randomness is independent with matching residuals. Independent unexposed prefetching is permitted.*

The identity $\min(p, q) + [p - q]_+ = p$ applies recursively under these conditions (Appendix D); score accuracy is not a correctness assumption.

3.3 Prediction targets and allocation scores

Let Y_e indicate a target-argmax match and A_e matching ancestors. The greedy-calibration control fits on calibration edges with $A_e = 1$:

$$\hat{p}(e) = \sigma(\alpha \operatorname{logit} q_0(e) + \beta), \quad \alpha > 0, \qquad S(v) = \prod_{e \in \mathrm{path}(v)} \hat{p}(e). \tag{3}$$

Raw and depth-adjusted controls instead use $\prod_e q_0(e)$ and $e^{-\lambda d(v)} \prod_e q_0(e)$. For unused proposal-support tokens R_v, the calibrated best-first control prioritizes a pending slot by

$$M(v) = S(v) \max_{x \in R_v} \hat{p}(x \mid v). \tag{4}$$

Subject to depth, sibling and node caps, each step admits the largest eligible slot with $M \geq \theta$, then draws without replacement; greedy mode takes the largest unused proposal. Every admitted draw stays, even below threshold. This bounds a *score*, not true gain. Greedy edge calibration implies neither joint path calibration nor sampled rejection calibration. Algorithm 2 specifies this original comparison policy.

For a realized tree, target logits instead permit the diagnostic

$$\rho(\mathrm{root}) = 1, \qquad \rho(v_j) = \rho(v) a_{v,j} \prod_{i<j} (1 - a_{v,i}). \tag{5}$$

Without EOS or output truncation, $1 + \sum_{v \neq \mathrm{root}} \rho(v)$ is expected round length conditional on the realized tree, integrating verifier randomness. It is neither a construction-time oracle nor a target-output law conditional on that tree. Diagnostic target work is outside throughput timing; truncation and proof details are in Appendix A.

3.4 Primary chains with optional leaves

With $R \geq 1$ output slots left, set the effective node cap $N_{\mathrm{eff}} = \min\{N_{\max}, R - 1\}$ and choose prescribed depth $D \leq \min\{7, N_{\mathrm{eff}}\}$ ($D = 0$ uses a target-only step). Retain the complete primary chain $X_d \sim q_d = q_{T,d}(\cdot \mid X_{d-1})$ from anchor X_0. Each primary parent may add one second-draw leaf: $P_{\mathrm{tree}} = P_{\mathrm{prep}} = D$ and $D \leq N \leq \min\{2D, N_{\mathrm{eff}}\}$ for an unpadded request. Additional leaves still cost selection, target work and KV space. This is a restriction of the existing static parent-budget family, not a new topology. Unlike the original threshold, this policy never prunes the prescribed primary chain.

For contextual selection, reuse the released confidence head's frozen weights:

$$c_d = \sigma(w_h^\top h_d + b_c + w_m^\top W_1[X_{d-1}]), \qquad s_d = \left(\prod_{i<d} c_i \right) (1 - c_d) c_d. \tag{6}$$

Cache the token scalar and project already computed backbone features h_d once, without additional fitting. FP32 evaluation need not equal BF16 bitwise. Keep the highest-scoring supported leaves with $s_d \geq \theta$, at most $N_{\mathrm{eff}} - D$, breaking ties by depth. The same builder with $c_d = 0.8$ supplies the static depth control. This isolates selection from shared kernel improvements.

Algorithm 1 Primary chain with optional confidence-selected leaves

Require: Anchor X_0, backbone logits/features (U, h), node/depth caps $(N_{\max}, d_{\max})$, output budget R, temperature T, threshold θ

$N_* \leftarrow \min(N_{\max}, R-1)$; $D \leftarrow \min(b, d_{\max}, N_*)$; $L \leftarrow \min(D, N_* - D)$; $Q \leftarrow \varnothing$
if $D = 0$ **then**
  return empty proposal tree, Q {target-only step}
end if
for $d = 1, \ldots, D$ **do**
  Form $\ell_d \leftarrow U_d + W_2 W_1 [X_{d-1}]$; if $T > 0$, form q_d by Eq. 1
  Draw $X_d \sim q_d$ with fresh randomness if $T > 0$; otherwise take $\arg\max \ell_d$
  Retain a private snapshot $Q_d \leftarrow q_d$ if $T > 0$
end for
Compute c_d, s_d by Eq. 6; static control uses $c_d = 0.8$
$J \leftarrow$ up to L depths with second-token support and $s_d \geq \theta$, ordered by decreasing score, then increasing depth
Initialize tree $\mathcal{T}$ with the complete chain $X_0 \to X_1 \to \cdots \to X_D$
for admitted depths $d \in J$ in increasing depth **do**
  Draw $Z_d \sim \mathrm{norm}(q_d \mathbb{1}[\cdot \neq X_d])$ freshly if $T > 0$; otherwise take the second-largest logit
  Append Z_d as a leaf of X_{d-1}, after its primary sibling X_d
end for
return $\mathcal{T}, Q$ for ordered verification by Eq. 2

Use independent race rows across parents and requests. After exposing the complete primary chain, admit leaves before exposing their second tokens, race keys, or correlated information. An admitted leaf then retains conditional proposal $q_d(z)\mathbb{1}[z \neq X_d] / \sum_{u \neq X_d} q_d(u)$; empty support forbids it. Prefetching cannot leak second draws through later primary randomness.

Confidence targets first-draw mean overlap, not residual hazard. For example, $q = (1/3, 1/3, 1/3)$ and targets $p_A = (0, 1/3, 2/3)$, $p_B = (1/3, 0, 2/3)$ both have overlap $2/3$. Given primary token 1, an admitted leaf gains respectively $1/2$ and 0 expected tokens, while $(1-c)c = 2/9$ in both cases. Averaging over the primary draw gives $1/6$ for both: this is conditional non-identifiability. The geometric score is therefore a surrogate, including at $T = 1$. With a common prefix, primary proposals and acceptance uniforms coupled to a chain, uncapped non-EOS round gain is in $\{0, 1\}$: only the first rejected primary can be rescued by a leaf. This gives no same-seed, end-to-end τ, or throughput floor. Appendix I gives the exact budgeting and packing rules and the implementation's numerical qualifications.

Algorithm 1 states the logical sampling order: form the primary chain, select leaf slots, then draw their tokens. The implementation may prefetch independent second draws while keeping them hidden until admission (Appendix I). Both algorithms share the ordered verifier and commit only its accepted path's KV slots; all unused candidate slots are released.

Load-dependent threshold. At each scheduling round, a fixed strategy receives $\theta(B) = \theta_4$ for active batch $B \leq 4$, and θ_8 otherwise. Native-batch validation freezes $(\theta_4, \theta_8) = (.1, .05)$ for calibrated best-first and $(.025, .05)$ for confidence leaves. The controller changes the threshold within that strategy; it observes neither arrival rate nor queue length. Constant-threshold controls use the same builders and verifier.

4 Experiments

4.1 Setup and evidence scope

Completed diagnostics use Qwen3-4B/DSpark block-7 on an H100 80GB. The six-task splits contain 72 calibration, 72 validation and 144 test prompts; Arena-Hard-v2 and LiveCodeBench add 64 transfer prompts each. Excluding exact duplicates and earlier development indices does not establish pretraining decontamination. Implementation corrections after observed greedy runs and a long-context failure leave policies frozen; corrective reruns are not newly blinded (Appendix B).

Validation uses 128 output tokens; frozen tests use 256, two seed sets and three timings per seed. Mixed-task batches disable EOS and drain without refill; serving is evaluated separately. Within

Algorithm 2 Score-based best-first construction (original comparison policy)

Require: Anchor x_0, shared logits $U_{1:b}$, effective node cap N, sibling cap K, threshold θ, temperature T, frozen score map $\hat{p}$

$S(\text{root}) \leftarrow 1$; $\mathcal{T}, Q \leftarrow \varnothing$; initialize heap H with the root's eligible child slot, keyed by Eq. 4
while $|\mathcal{T}| < N$ and $H \neq \varnothing$ **do**
  Pop parent slot v of maximum current key $M(v)$
  if $M(v) < \theta$ **then**
    break
  end if
  Admit the slot before exposing its token or correlated hidden randomness
  If $T > 0$, retain original proposal $Q_v \leftarrow q_{T,d(v)+1}(\cdot \mid x_v)$ if absent
  Draw $x \sim \text{norm}(Q_v \not\vdash_{R_v})$ if $T > 0$; otherwise maximize $q_{0,d(v)+1}$ over unused tokens R_v
  Append $c = (v, x)$ in sibling order; $S(c) \leftarrow S(v)\hat{p}(c)$; remove x from R_v
  Enqueue c's eligible slot if $d(c) < b$; refresh and re-enqueue v if it has fewer than K children and $R_v \neq \varnothing$
end while
return $\mathcal{T}, Q$ for ordered verification by Eq. 2

Table 1: **Frozen primary comparison on Qwen3-4B.** The chain row gives tokens/s; other cells give percentage changes versus chain with paired 95% input-batch intervals. All 144 test prompts use 256 output tokens, two seed sets and three timing repetitions. Each family uses its validation-selected configuration. Greedy fixed trees have caps 14/11 at B4/B8; sampling fixed trees use the same-builder static cap 14. The T0/B4 depth choice aliases raw. Dashes denote untested families.

Policy	$T = 0$, B4	$T = 0$, B8	$T = 1$, B4	$T = 1$, B8
Chain (tokens/s)	1079.2	1706.8	958.0	1562.0
Selected fixed tree	$+5.0$ $[+2.1, +7.6]$ -4.9	$+8.6$ $[+3.8, +14.8]$ -4.2	$+8.2$ $[+4.2, +12.0]$ -5.8	$+5.1$ $[+1.7, +8.9]$ -8.3
Raw threshold	$[-7.1, -2.8]$ -4.2	$[-8.1, +1.4]$ -6.5	$[-8.9, -2.6]$ -5.4	$[-12.6, -3.0]$ -7.8
All-edge fit	$[-5.9, -2.4]$ -3.3	$[-8.6, -4.4]$ -4.5	$[-8.9, -1.9]$ -4.5	$[-11.0, -4.4]$ -11.7
Surviving-edge fit	$[-5.6, -1.2]$ -4.9	$[-8.4, -0.2]$ -4.5	$[-8.8, -0.9]$ -4.8	$[-14.8, -8.1]$ -9.3
Depth correction	$[-7.1, -2.8]$	$[-7.1, -1.6]$	$[-8.2, -1.2]$ $+6.7$	$[-13.2, -5.5]$ $+5.8$
Confidence threshold	—	—	$[+2.2, +11.2]$ $+6.8$	$[+1.3, +10.0]$ $+3.1$
Confidence budget	—	—	$[+1.7, +11.4]$	$[-2.0, +8.2]$

each stratum, methods share the target, drafter, verifier, cache and target graphs. Throughput divides output tokens by scheduler time, including prefill, drafting, construction, verification and KV commit; loading, warm-up, submission, decoding, cache flushing and assertions are excluded. We report τ and node/parent work as defined in Section 3.1. Paired 95% throughput intervals resample complete batches, retaining seeds and timings; partial batches remain separate. Diagnostic intervals resample prompts within tasks. Both use 2,000 replicates. Appendix B records frozen selections, source hashes and serving's distinct protocol.

4.2 Useful work and the controlled throughput comparison

Each score family is independently selected on validation against chain7, retuned raw/calibrated best-first trees, a jointly tuned depth penalty, and fixed node/parent budgets. Confidence admission also competes with all static leaf caps seven through fourteen in the *same fused builder*. This controls shared implementation gains; matching N or numerical θ alone would not.

The selected fixed trees improve throughput over chain in all four strata (Table 1). At $T = 1$, confidence thresholds improve throughput over chain by 6.7% at B4 and 5.8% at B8, but their incremental changes versus the optimized static tree are -1.38% $[-3.49, 0.88]$ and $+0.63\%$ $[-2.18, 3.73]$, re-

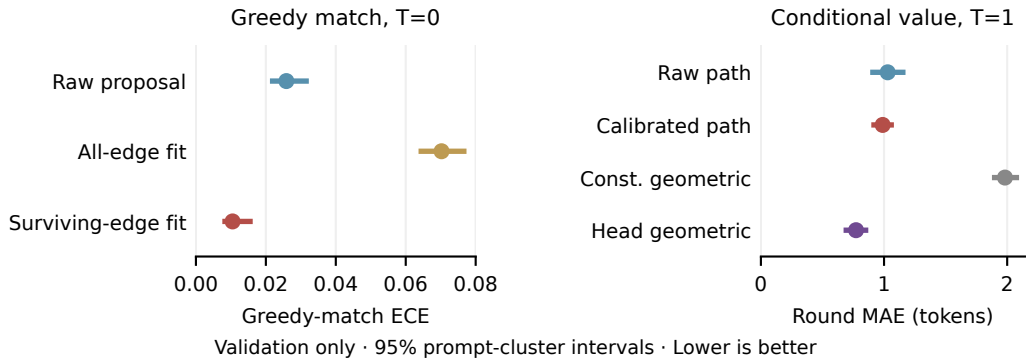

Figure 2: **Different prediction targets on validation data.** Left: greedy-match ECE on 19,144 surviving-context edges ($T = 0$). Right: round-level MAE against conditional verifier value on 1,560 realized trees ($T = 1$). Both use 72 prompts; bars are 95% task-stratified prompt-cluster intervals. The constant prior uses $c = 0.8$ without tuning; the released confidence head has no new fitting. These diagnostics do not measure alternate-policy throughput.

spectively. Neither interval establishes an adaptive-selection advantage. Surviving-context best-first calibration is slower than chain in all four strata despite its better greedy ECE.

4.3 Calibrating the right greedy population

On 72 validation prompts, surviving-context fitting reduces greedy ECE from 0.0259 to 0.0105: paired reductions are 0.0154 [0.0074, 0.0215] versus raw and 0.0598 [0.0530, 0.0637] versus all-edge fitting. Leave-one-task-out fitting yields pooled ECE 0.0108, but worsens GSM8K and MATH500; task-mean ECE changes only from 0.0305 to 0.0256. These improvements do not establish path calibration or better retuned allocation. Appendix F gives fitting populations, frozen coefficients and per-task results; Figure 2 summarizes the diagnostic.

4.4 Greedy calibration versus sampled commitment

At $T = 1$, we compare frozen surrogates on the same 1,560 realized trees from 72 validation prompts. Equation 5 integrates verifier randomness conditional on these trees; it does not replay an alternative policy. Construction and output-cap details are in Appendix G.

Calibrated paths predict mean value 5.875 versus 5.900, yet round MAE is 0.989 versus raw 1.030; the paired difference -0.041 $[-0.112, 0.030]$ does not establish improvement. Released-confidence geometric scoring lowers MAE to 0.773 (difference -0.216 $[-0.252, -0.182]$) but overpredicts the mean by 0.318 [0.225, 0.414]. Neither diagnostic measures throughput. The untuned $c = 0.8$ comparator does not represent all static priors, and confidence is not an exact residual-hazard predictor (Section 3.4; Appendix G).

4.5 Why longer paths need not run faster

Table 2 shows why acceptance and node count alone are insufficient. On these validation prompts, calibrated best-first $\theta = 0.05$ raises τ by 16.75% but lowers throughput by 8.59% versus chain7, using $1.963\times$ the executed parent rows per token. Confidence $\theta = 0.05$ retains 28.87% fewer nodes than same-builder fixed14, yet parent work rises 0.27% and throughput changes -0.44%. These are descriptive validation measurements; Appendix L gives the separate eight-prompt timing profile. Under a common-prefix coupling, pruning second leaves removes nonnegative round value and therefore needs cost savings; fixed attention tiles can make those savings non-proportional to node count (Appendix A).

Table 2: **Acceptance, retained nodes, and executed work.** Descriptive validation measurements on all 72 prompts, $T = 1$, initial batch eight, 128 output tokens, two seed sets and two timings. τ, N, P_{prep} and P_{exec} are per request-round; the last includes projection padding. Both threshold policies use $\theta = 0.05$. Synchronized profiles are excluded.

Policy	τ	N	P_{prep}	P_{exec}	Tokens/s
Chain7	4.81	6.79	6.79	7.23	1,607.1
Static leaves, cap 14	5.23	13.13	6.76	7.07	1,763.3
Confidence leaves	5.20	9.34	6.78	7.09	1,755.6
Calibrated best-first	5.62	13.69	12.93	16.57	1,469.0

Shared execution. All controls use paged attention that reads KV by slot and masks each request to its prefix and tree ancestors. Batched parent projections, target verification and KV commits combine work across requests; CUDA graphs replay common shapes. Negative padding slots skip cache writes, and only the verified path retains its KV. Both leaf policies share their builder. Appendix H gives the execution checks.

Frozen transfer. Transfer reuses frozen settings without refitting. The 128 dataset-transfer prompts overlap development's broad chat/coding domains; their 4,096-token prompt limit was chosen from lengths alone (primary limit: 512). Appendix B specifies each frozen contrast.

With the 4B sampling settings unchanged, selected fixed trees gain 7.3% [4.5, 10.4] on Qwen3-8B, 7.4% [0.5, 14.2] on Qwen3-14B, and 10.9% [7.3, 15.3] at $T = 0.5$ on Qwen3-4B versus their matched chains. Confidence gains are 5.9%, 5.8% and 8.7%, respectively, but no confidence-versus-static interval excludes zero. At the additional-dataset limit of 512 tokens, fixed and confidence changes versus chain are 3.2% $[-0.1, 6.0]$ and 2.8% $[-0.9, 6.7]$; these intervals do not establish a throughput gain.

At the 4,096-token limit, fixed and confidence trees gain 5.0% [2.2, 8.7] and 3.6% [1.4, 5.7] versus chain. Confidence changes -1.39% $[-3.94, 1.14]$ versus static. Figure 5 and Appendix C retain all frozen transfer families and their work measurements; no settings are reselected from these results.

4.6 Active-batch thresholds under fixed-rate arrivals

The controller uses its batch-four threshold at active $B \leq 4$ and batch-eight threshold otherwise, within one strategy. It observes neither queue length nor arrival rate and never switches to chain. Above four, its threshold equals the high-bin constant; earlier histories may differ, so adaptation alone implies no gain over that control. The lookup transfers native-batch validation choices; it is not the result of a serving-specific threshold search.

Frozen Poisson traces use 7, 9 and 11 requests/s, six arrival seeds, and 256 requests of 256 tokens. Paired intervals resample whole traces. Throughput uses the common 20–90% scheduled-arrival window; latency cohorts are followed to completion. Makespan throughput and whole-trace τ remain separate. Backlog growth accompanies latency: overloaded finite traces do not estimate stationary latency. These tests concern fixed-rate, not time-varying arrivals (Appendix B).

At $T = 0$, calibrated load thresholds reduce window throughput versus chain by 6.1%, 11.8% and 12.5% at 7, 9 and 11 requests/s; their paired 95% intervals are $[-9.3, -2.6]$, $[-12.4, -11.1]$ and $[-13.7, -11.7]$. Fixed-tree changes versus chain range from -0.33% to -0.12%, with all intervals containing zero. Thus the native batch advantage does not establish a serving advantage. The load controller spends 99.5% of measurement-window request-rounds above active batch four at rate seven, and 100% at rates nine and eleven: its threshold is effectively the high-bin constant over this window.

At $T = 1$, confidence load thresholds gain 2.4% [0.6, 4.4], 3.1% [1.8, 4.8] and 3.3% [1.7, 5.1] versus chain. Changes versus the static tree are -0.45% $[-1.31, 0.26]$, $+0.27\%$ $[-1.04, 1.89]$ and $+0.53\%$ [0.28, 0.78]. None establishes an advantage over constant θ_8; at rate nine, the controller is 1.34% [0.48, 1.98] slower than constant θ_4. Its high-bin occupancy is 98.3%, 100% and 100%. The calibrated load policy remains 14.8–17.4% below chain. These results distinguish the leaf strategy's benefit from any benefit of switching thresholds.

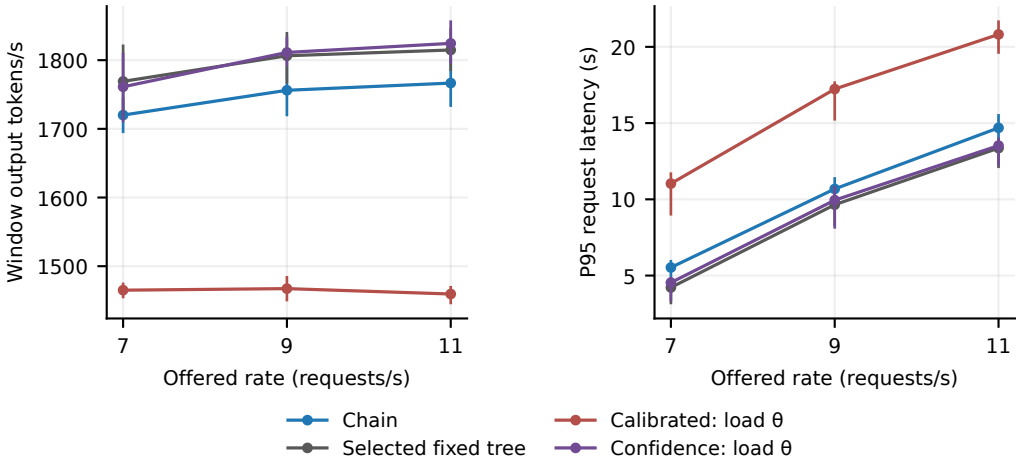

Figure 3: Fixed-rate GPU serving at $T = 1$. Points and bars show window throughput and arrival-cohort P95 latency with whole-trace 95% intervals. Backlog grows at rates nine and eleven, so these are finite-horizon overload measurements. Appendix C.1 retains all constant-threshold controls, paired throughput intervals and backlog counts.

5 Discussion and Limitations

Better greedy calibration and conditional sampled-round prediction need not improve throughput. The overlap counterexample rules out an exact scalar-utility interpretation, but approximate allocation may still be useful.

Optional leaves reuse primary-chain projections but still require selection, verification and cache space. Same-builder static controls separate these construction benefits from contextual selection. An interval containing zero is not equivalence. Our comparisons concern component policies on one H100 80GB, not separate production engines or alternative drafters; hardware and scheduler changes may alter the trade-offs.

The completed diagnostics concern Qwen3-4B/DSpark and seven draft depths. A frozen head can still encounter distribution shift. In the separate native-EOS sensitivity, fixed and confidence trees gain 6.8% [4.3, 9.5] and 3.9% [0.7, 7.2] tokens/s versus chain, while confidence loses 2.7% [0.3, 5.4] versus static. Across arms, 60.4–61.5% of requests reach the 256-token cap; this bounded stop convention establishes neither answer quality nor full generation-library parity. Fixed-rate traces do not establish time-varying-load robustness or stationary latency under overload. Frozen transfers cover Qwen3-8B/14B at $T = 1$, Qwen3-4B at $T = 0.5$, and 128 additional prompts under two input limits. They do not cover other model families or hardware. The larger limit is not a uniform long-context workload, and each checkpoint supplies its own released confidence head.

The target-law result requires the stated proposal, admission, randomness, and exact-arithmetic assumptions. Implementation checks do not prove universally exact finite-precision equivalence. Allocation accuracy is separate from correctness.

6 Conclusion

TreeSpark evaluates acceptance scores through the work they admit and the policies they improve upon. Preserving primary parents gives fixed and confidence-guided trees useful throughput gains over chain. Yet native confidence-versus-static intervals contain zero, and fixed-rate serving gains do not establish a benefit from switching thresholds when one batch bin dominates. Acceptance length and score accuracy are therefore insufficient: tree policies also need executed-work measurements and matched static controls.

AI USE STATEMENT

Generative AI assisted the preparation of this version through literature search, manuscript restructuring and editing, review of mathematical claims and their assumptions, interpretation of existing results, preparation of the method overview, experimental design, figure planning, and LaTeX formatting. AI assistance also contributed to implementation changes, experiment orchestration, finite-vocabulary verification checks, and diagnostic plotting. The manuscript distinguishes validation diagnostics from the frozen evaluation and reports the implementation corrections and corrective reruns. Experimental tables and figures were regenerated from checked raw records. The authors are responsible for independently checking the claims, code, data provenance, and final content before submission.

REFERENCES

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 (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR, 2024.

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.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. arXiv:2602.06036.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024.

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.

Lanxiang Hu, Zhaoxiang Feng, Yulun Wu, Haoran Yuan, Yujie Zhao, Yu-Yang Qian, Bojun Wang, Peng Zhao, Daxin Jiang, Yibo Zhu, Tajana Rosing, and Hao Zhang. JetSpec: Breaking the scaling ceiling of speculative decoding with parallel tree drafting. *arXiv preprint arXiv:2606.18394*, 2026a.

Shijing Hu, Jingyang Li, Zhihui Lu, and Pan Zhou. Bridging draft policy misalignment: Group tree optimization for speculative decoding. In *International Conference on Learning Representations (ICLR)*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/b6e67ae290635d0874c4cb43ba2a2cfb-Paper-Conference.pdf.

Zhengmian Hu, Tong Zheng, Vignesh Viswanathan, Ziyi Chen, Ryan A. Rossi, Yihan Wu, Dinesh Manocha, and Heng Huang. Towards optimal multi-draft speculative decoding. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/0907335ecf28faf15be54485dbcbe70e-Paper-Conference.pdf.

Jianuo Huang, Yaojie Zhang, Qituan Zhang, Hao Lin, Hanlin Xu, and Linfeng Zhang. Domino: Decoupling causal modeling from autoregressive drafting in speculative decoding. *arXiv preprint arXiv:2605.29707*, 2026.

Kaiyu Huang, Hao Wu, Zhubo Shi, Han Zou, Minchen Yu, and Qingjiang Shi. AdaSpec: Adaptive speculative decoding for fast, SLO-aware large language model serving. In *Proceedings of the 2025 ACM Symposium on Cloud Computing (SoCC)*, pp. 361–374. ACM, 2025. doi: 10.1145/3772052.3772239.

Wonseok Jeon, Mukul Gagrani, Raghavv Goel, Junyoung Park, Mingu Lee, and Christopher Lott. Recursive speculative decoding: Accelerating LLM inference via sampling without replacement. *arXiv preprint arXiv:2402.14160*, 2024. URL https://arxiv.org/abs/2402.14160.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023.

Tianyi Li, Yaxin Luo, Xinyi Shang, and Zhiqiang Shen. DARTree: Speculative diffusion decoding with autoregressive draft trees. *arXiv preprint arXiv:2608.13524*, 2026a. URL https://arxiv.org/abs/2608.13524.

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 (EMNLP)*, pp. 7421–7432. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.422.

Zixian Li, Tong Li, Chi Xie, Xiaohui Song, and Haonan Lu. From chains to trees: Parent-conditioned drafting for semi-autoregressive speculative decoding. *arXiv preprint arXiv:2608.02123*, 2026b.

Saw S. Lin and Jyh-Shing Roger Jang. DominoTree: Conditional tree-structured drafting with domino for speculative decoding. *arXiv preprint arXiv:2607.08642*, 2026.

Xiaoxuan Liu, Jongseok Park, Langxiang Hu, Woosuk Kwon, Zhuohan Li, Chen Zhang, Kuntai Du, Xiangxi Mo, Kaichao You, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Turbospec: Closed-loop speculation control system for optimizing LLM serving goodput. *arXiv preprint arXiv:2406.14066*, 2025.

Xiaoxuan Liu, Jiaxiang Yu, Jongseok Park, Ion Stoica, and Alvin Cheung. Speculative decoding: Performance or illusion? In *Proceedings of Machine Learning and Systems*, volume 8, pp. 1719–1741. MLSys, 2026.

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 (ASPLOS)*, pp. 932–949. ACM, 2024. doi: 10.1145/3620666.3651335.

Soowon Oh, Nam Cao, Yujin Kim, Hojung Jung, Huzama Ahmad, Sangmin Bae, and Se-Young Yun. BASTION: Budget-aware speculative decoding with tree-structured block diffusion drafting. *arXiv preprint arXiv:2605.29727*, 2026. URL https://arxiv.org/abs/2605.29727.

Liran Ringel and Yaniv Romano. Accelerating speculative decoding with block diffusion draft trees. *arXiv preprint arXiv:2604.12989*, 2026.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae H Ro, Ahmad Beirami, and Ananda Theertha Suresh. Block verification accelerates speculative decoding. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/3e710b42b1a9ed898f607ec0f4fcc971-Paper-Conference.pdf.

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.

Shuai Zhang, Huachuan Qiu, Hongliang He, and Yong Dai. Cost-aware diffusion draft trees for speculative decoding. *arXiv preprint arXiv:2606.01813*, 2026.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. DistillSpec: Improving speculative decoding via knowledge distillation. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/8766fbc68e1ed1cdef712ce273e0a363-Paper-Conference.pdf.

A CALIBRATION, DEPTH, AND THE MEANING OF NODE VALUE

Exact score decomposition. Write $q_e = q_{0,d}(x \mid x_p)$ for an edge probability. For $0 < q_e < 1$, the Platt map satisfies

$$\hat{p}(q_e) = \frac{e^{\beta} q_e^{\alpha}}{(1-q_e)^{\alpha} + e^{\beta} q_e^{\alpha}}. \tag{7}$$

For a depth-d path, this gives the algebraic identity

$$\log S(v) = \alpha \sum_{e \in \mathrm{path}(v)} \log q_e + \beta d - \sum_{e \in \mathrm{path}(v)} \log\left[(1-q_e)^{\alpha} + e^{\beta} q_e^{\alpha}\right]. \tag{8}$$

For fixed $\alpha > 0$ and sufficiently small q_e, each logarithm in the last sum approaches zero. The remaining approximation is a power of the raw path probability together with a depth term. It need not be accurate on high-probability edges. Consequently, independently retuning a threshold cannot in general reproduce the calibrated path ordering. Whether its additional nonlinear dependence improves inference is an empirical question.

The depth control uses $S_\lambda(v) = \exp(\sum_e \log q_e - \lambda d)$, with $\lambda \geq 0$ chosen independently on validation prompts. Its edge multipliers $q_e e^{-\lambda}$ remain at most one, allowing the same predictable max-unused bound as in Eq. 4. Both λ and the threshold must be tuned; the fitted $-\beta/\alpha$ is only a diagnostic reference. Allowing negative penalties would require checking the descendant bound and search order rather than silently reusing the same pruning argument.

Three distinct events. For a proposed edge, let Y_e denote a greedy target match, V_e denote that the verifier reaches and tests that candidate, and C_e denote that the candidate enters the committed path. The original fit estimates a greedy-match frequency conditional on surviving ancestors. Sampling instead has $\Pr(C_e) = \Pr(V_e)\Pr(C_e \mid V_e)$, where V_e includes the rejection of earlier siblings and $\Pr(C_e \mid V_e)$ depends on the residual proposal and residual target. A reliability diagram for Y_e is therefore not a reliability diagram for C_e. Our event diagnostics keep these labels, conditioning populations, and prediction rules separate.

Locally calibrated edges need not rank paths correctly. Consider two equally frequent prefix populations. A first edge has true survival probability and calibrated prediction 0.4 in population 1 and 0.6 in population 2. A second edge has prediction 0.5 in both populations, while its true conditional survival probabilities are 0.9 and $7/30$, respectively. Among surviving first edges, its match frequency is

$$\frac{0.4(0.9) + 0.6(7/30)}{0.4 + 0.6} = 0.5.$$

Thus both predictors are perfectly calibrated on their respective fitting populations. Their path products nevertheless rank population 2 above population 1 ($0.30 > 0.20$), whereas the true survival probabilities have the opposite order ($0.14 < 0.36$). This constructed example is not a model measurement. It shows why conditioning the fitting population is useful but insufficient: a scalar score can omit context that matters after conditioning on the particular preceding path. Neither zero edge ECE nor monotonicity of the calibrator alone guarantees good budget allocation.

Verifier-conditioned node values. For diagnostics under sampling, the target forward permits an exact value that averages out acceptance randomness. Fix a realized proposal tree $\mathcal{T}$, its ordered siblings, its proposal distributions, and all target distributions at its nodes. At a parent v, let $p_{v,j}$ be the target residual after hypothetical rejection of its first $j-1$ children, and let $q_{v,j}$ be the matching without-replacement proposal. Define

$$\begin{aligned} a_{v,j} &= \min\left\{1, \frac{p_{v,j}(x_{v,j})}{q_{v,j}(x_{v,j})}\right\}, \qquad \rho(\mathrm{root}) = 1, \\ \rho(x_{v,j}) &= \rho(v)\, a_{v,j} \prod_{i<j} (1 - a_{v,i}). \end{aligned} \tag{9}$$

The product accounts for sibling visitation as well as ancestor survival. If a preceding child has acceptance probability one, later children have value zero and their undefined zero-mass residuals need not be evaluated. Every retained candidate must have positive mass under its actual proposal at the time it was sampled.

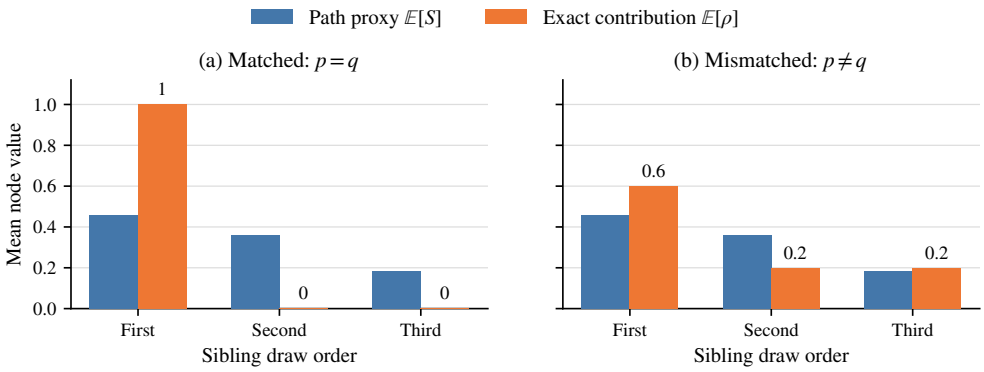

Figure 4: Illustrative exact finite-vocabulary example, not a model experiment. A one-level tree samples three siblings without replacement from $q = (0.6, 0.3, 0.1)$. Bars average over all six ordered draws, with their exact probabilities. An identity edge map gives $S(v) = q(x_v)$ and isolates sibling visitation from calibration. (a) For $p = q$, the first child is always accepted; later positive proxy scores correspond to zero commitment value. (b) For $p = (0.2, 0.4, 0.4)$, the exact contributions are $0.6, 0.2, 0.2$, obtained by enumerating the ordered proposals and integrating their accept/reject probabilities.

Proposition 2 (Conditional expected committed length)**.** *For a fixed realized tree with matching proposal reconstruction and independent acceptance random variables, let A count committed draft tokens before a correction or bonus token. In the absence of EOS and output-budget truncation,*

$$\Pr(v \textit{ is committed} \mid \mathcal{T}) = \rho(v),$$
$$\mathbb{E}[A \mid \mathcal{T}] = \sum_{v \in \mathcal{T}} \rho(v), \qquad \mathbb{E}[\tau \mid \mathcal{T}] = 1 + \sum_{v \in \mathcal{T}} \rho(v). \tag{10}$$

The sums exclude the root.

Proof. Conditional on reaching v, child j is committed exactly when all preceding siblings are rejected and its own independent acceptance test succeeds. This gives Eq. 9 by induction from the root. Writing A as the sum of node-commitment indicators and applying linearity of expectation proves the second identity. An uncapped round emits one additional correction or bonus token, proving the third. □

These are expectations over verification randomness conditional on a realized tree; they do *not* assert that output tokens are target-distributed conditional on that tree. Distributional correctness instead averages over proposal construction as well (Appendix D). In particular, recording $1 + \sum_v \rho(v)$ on a final output-capped round gives its hypothetical *untruncated* length, not its expected actual committed length. It must not be aggregated as the reported τ. If $R \geq 1$ output tokens remain and EOS is disabled, the capped round length L_R instead satisfies

$$\mathbb{E}[L_R \mid \mathcal{T}] = 1 + \sum_{v:\, \mathrm{depth}(v) < R} \rho(v). \tag{11}$$

Indeed, $L_R = 1 + \min(A, R - 1)$: each committed draft node before depth R permits one further emitted token. With EOS enabled, a path that has already emitted EOS must additionally be terminated; the uncapped identity cannot be used unchanged. Evaluating ρ at every materialized parent is an offline diagnostic with additional reconstruction cost, not free production-time knowledge available before the target forward.

The quantity $\sum_v \rho(v)$ is the conditional expectation of the uncapped, non-EOS accepted count in the same sampling experiment. Its variance across proposal trees is no greater than the variance of that count, by the law of total variance. It can therefore support less noisy diagnostics of $S(v)$ and sibling-rank effects without fitting another predictor. Across trees, uncertainty must still respect the prompt/run sampling unit.

Figure 4 isolates the difference between path scores and verifier values. This finite example motivates measuring visitation and commitment separately; it does not establish the size or direction of a model-level speed improvement.

A legal local marginal-gain probe. The general marginal value of adding a candidate is a difference in expected committed length between two specified construction policies, averaged over their proposal and verification randomness. An interpretable probe can append one last sibling slot or extend an existing leaf, admitting the slot before sampling its token and retaining all previous nodes and sibling order. Absent EOS and output truncation, the increase in expected committed length for this isolated extension equals the probability that the added node is committed. This is an expectation identity, not a claim that independently sampled runs differ by exactly one token. Arbitrary insertion before an existing sibling, reordering, or rebuilding the rest of the tree changes other opportunities and does not isolate the value of one node. Such a probe would need to resample the complete legal construction at fixed target prefixes; conditioning on an already realized sampled tree does not generally preserve the target law.

The completed diagnostic in Appendix G compares scores with conditional verifier values on recorded trees. It does not perform this randomized construction intervention and should not be interpreted as a causal estimate of another allocation policy.

The cost requirement for pruning optional leaves. Fix a common request prefix, primary chain C of depth D, and latent support-valid second draws Z, with independent correct conditional proposal laws. Couple verification uniforms across a full-leaf construction and a predictably admitted subset S of its second leaves. Assume a common causal target, exact arithmetic, and no EOS or output truncation. If a_d is the first-child acceptance probability at primary parent d and b_d is second-child acceptance under its matching rejection residuals, then

$$\mathbb{E}[L_{\text{full}} - L_S \mid C, Z, S] = \sum_{d \notin S} \left(\prod_{i<d} a_i \right) (1 - a_d) b_d \geq 0. \tag{12}$$

The contribution is zero when rejection is impossible or second-proposal support is empty. The two coupled rounds agree until the first primary rejection. An accepted second leaf omitted from S adds exactly one token; otherwise the lengths agree. Thus the coupled difference belongs to $\{0, 1\}$. With the same positive one-round execution cost, pruning cannot improve conditional expected tokens per cost; it must save enough work. This orders neither complete generation trajectories nor measured throughput under different batch schedules or finite-precision logits.

The measured attention kernel also illustrates why node savings need not translate proportionally to cost savings. Its query tile has 16 rows; an anchor, seven primary nodes and at most seven leaves occupy one active tile per request/head. Removing leaves leaves that tile count unchanged. Key tiles of width 64, masked traffic, graph buckets and full-model projection shapes can still change, so this source observation is not a constant-cost claim or a measured attribution of runtime to padding.

B Controlled Evaluation and Reproducibility

Data isolation. For each of GSM8K, MATH500, HumanEval, MBPP, MT-Bench, and Alpaca, the split excludes the first 25 records used in earlier development, then assigns 12 calibration, 12 validation, and 24 test prompts, with normalized exact duplicates removed across splits. This gives 72 calibration, 72 validation, and 144 test prompts; only the first user turn is used. Arena-Hard-v2 and LiveCodeBench contribute 64 additional prompts each, reserved for transfer. These are unseen datasets in this protocol, not necessarily unseen semantic domains or pretraining data. The primary input protocol disables thinking and retains the final 512 prompt tokens. The separately declared 4,096-token transfer limit was selected from prompt lengths, without inspecting generated outputs. It is an upper limit, not a uniform 4,096-token input workload: the 128 tokenized prompts have median length 412 and maximum 3,439; 43 exceed 512 tokens and three exceed 2,048. The two limits therefore test different truncation of this fixed input set, rather than general long-context scaling.

Fitting, selection, and freezing. Calibration labels are used only to fit the two-parameter maps. Validation selects thresholds, node budgets, and depth penalties separately for each method and

batch/temperature stratum; test labels cannot refit these maps or select configurations. The completed 24-prompt screen uses four validation prompts per task, 128 output tokens, $T \in \{0, 1\}$, and batches four and eight. Its leading settings are candidates for validation refinement, not held-out winners. Fixed controls include probability-ranked fixed-node trees and static rank-prior topologies; the depth-penalty family includes raw scoring as its zero-penalty case. Each method receives independent selection rather than a shared numerical threshold.

The confidence-leaf extension is declared before final evaluation. It uses the released confidence head without fitting additional weights. Its static control tests every cap from seven to fourteen; confidence thresholds are selected from $\{0, .005, .01, .025, .05, .1, .2\}$, and confidence-ranked fixed budgets test caps eight through thirteen. The 24-prompt screen advances two candidates per added family, retaining the original comparison controls. After eliminating redundant constant scoring work from the static implementation, all prior nominees and all eight static caps are rerun together: eighteen arms per sampling stratum, 72 validation prompts, two seed sets, and two timings per seed. Old and new implementation timings are not pooled. Source-transfer declarations preserve unchanged original greedy configurations for their final rerun.

Final evaluation uses the validation-selected configurations in Table 3; no test-set argmax is selected. Calibration coefficients and their fitting population are given in Appendix F. Implementation corrections are disclosed below, separately from configuration selection.

Corrective source transfer. The first two greedy test strata prompted the numerical audit in Appendix H. The subsequent correction promotes Q and K to FP32 before regular decode accumulation; the tree path used by the measured policies is unchanged. All selected policies, inputs, seeds, budgets and contrasts are preserved. All nine native strata, both serving suites and the natural-EOS sensitivity were rerun under the corrected source, without pooling earlier timings. The two earlier greedy strata and their numerical diagnostics have been inspected: the reruns are not newly blinded evaluation. No threshold, calibration map or policy is selected from these test outcomes.

The 4,096-token input-limit stratum subsequently failed during an incomplete prefill chunk. Its repair adds a scheduler stream dependency when no request finishes a chunk (Appendix H). The complete stratum was rerun with the same configuration, cases, seeds, output/context budgets and contrasts. Failed timings and untimed diagnostics are excluded. Each reported stratum compares policies under a common source version; the long-context repair is not pooled with its failed predecessor or with short-context timings.

Frozen configurations. Table 3 gives the selected parameters for all four primary strata. Best-first families allow 64 nodes and eight siblings. Greedy fixed controls use rank-prior trees with four siblings and prior acceptance 0.8; sampling fixed controls use the same-builder static leaf construction. Leaf families allow at most two siblings and 14 nodes unless a smaller fixed budget is specified. All families use at most seven draft depths. These are nominal caps: remaining output budgets and exhausted proposal support still apply. All timed arms share a 1,024-token target graph capacity and the execution optimizations described in Appendix H.

Table 3: Validation-selected primary configurations. Each row names its parameter; $N_{\max}$ excludes the anchor. The depth control uses raw path scores multiplied by $e^{-\lambda d}$. At $T = 0$, B4 it is identical to the raw control. Dashes denote families not evaluated in greedy mode.

Policy parameter	$T = 0$, B4	$T = 0$, B8	$T = 1$, B4	$T = 1$, B8
Chain $N_{\max}$	7	7	7	7
Fixed tree $N_{\max}$	14	11	14	14
Raw θ	0.2	0.2	0.2	0.2
All-edge fit θ	0.05	0.05	0.05	0.05
Surviving-edge fit θ	0.1	0.05	0.1	0.05
Depth control (λ, θ)	$(0, 0.2)$	$(0.5, 0.01)$	$(0.5, 0.025)$	$(0.1, 0.1)$
Confidence leaves θ	—	—	0.025	0.05
Confidence budget $N_{\max}$	—	—	11	10

The five transfer strata reuse the batch-eight, $T = 1$ settings without retuning: Qwen3-8B, Qwen3-14B, two input limits on the additional datasets, and $T = .5$ on Qwen3-4B. Each model uses its own

released DSpark checkpoint and frozen confidence head; this is transfer of the allocation rule and settings, not reuse of identical 4B head weights in larger models. Primary and temperature-transfer strata have 144 prompts; the two additional-dataset strata share the same 128 prompts. Native final runs use 256 output tokens, two seed sets, and three timings per seed.

Matched execution and estimators. The native comparison uses the same target and drafter checkpoints, conditional proposal head, verifier, cache implementation, and target graph configuration for chain and tree methods. Each contrast uses the same tokenized inputs, request-seed assignments and hardware. The throughput timer covers scheduler work, including prefill, drafting, construction, verification, and KV commit; it excludes model loading, graph warm-up, submission, text decoding, cache flushing, assertions, and serialization. Synchronized phase profiles are separate measurements and do not enter the throughput estimate. Reported throughput is total emitted tokens divided by total timed seconds. Reported τ is total post-prefill committed tokens divided by request-rounds, including correction/bonus tokens and actual output-limit truncation.

Paired comparisons require identical complete prompt-batch and repetition sets in every arm. Partial final batches are reported separately. Throughput bootstrap units are complete input batches within each temperature/batch-size stratum, retaining seeds and timing repetitions inside their batch cluster. Calibration and node-value diagnostics instead resample prompts within tasks, retaining all their edges or rounds. They use 2,000 paired bootstrap replicates. Neither edges nor timing repetitions are treated as independent prompts. Shared seeds across methods do not imply identical sampled outputs or a tokenwise coupling. The intervals are marginal paired 95% intervals, not simultaneous family-wise confidence bands. The full set of frozen contrasts is retained; an interval containing zero does not establish equivalence.

Stationary serving scope. The serving protocol uses fixed-rate Poisson arrivals and identical arrival traces across methods. A load controller may observe the actual active requests and temperature, but not the generator's request-rate parameter. Its lookup and coefficients must be selected on validation and frozen before evaluation. Whole independent traces, not individual requests within one trace, are the replication units. Throughput counts useful generated tokens; latency runs from arrival through queueing and completion. Ramp, the fixed measurement window, arrival tail, and drain are recorded separately. Below saturation, reduced latency alone does not establish increased capacity. No time-varying-load generalization claim follows from this protocol.

The chain-only validation pilot fixes rates at 7, 9, and 11 requests/s. Two-bin policies use the batch-four validation threshold for at most four active requests and the batch-eight threshold otherwise. These thresholds come from native-batch validation. A separate serving validation runs all predeclared nominees, which remain in the final comparison regardless of their speed; it does not reselect the lookup from a serving threshold grid. The experiment therefore tests this transferred two-bin rule, not the best possible load controller. Both constant thresholds remain measured controls when distinct. Under sustained occupancy above four requests the policy uses its high-bin constant; it need not improve on that constant. Final traces contain 256 requests of 256 output tokens, with six independent arrival seeds. The measurement window is 20–90% of the scheduled-arrival span and is identical across paired arms. Window throughput and complete-trace makespan throughput are distinct estimands. Arrival-window latency cohorts are followed to completion, and backlog growth is reported: overloaded finite traces do not define stationary queueing latency. The original calibrated load policy switches between .1 and .05; the sampling confidence policy switches between .025 and .05. The low-bin threshold is used at $B \leq 4$ in each case. Both constants, chain, and the validation-selected batch-eight fixed tree remain explicit controls. Appendix C.1 reports every frozen contrast, the constant-threshold controls and active-batch occupancy.

Evidence scope. The completed calibration, sampling-value, and numerical-reference checks are reported with their populations in the following appendices. Development screens and a chain-only capacity pilot do not establish a frozen adaptive policy's held-out throughput or serving advantage. Earlier implementation timings and incomplete runs are excluded from the controlled comparisons in Appendix C.

C Complete Frozen Comparisons

Reading the tables. All configurations were selected on validation and preserved for these tests. The following work tables include every frozen native arm, merging aliases only when they denote the identical configuration. Tokens/s is total output tokens divided by total scheduler time, including prefill. The other columns are aggregate request-round quantities: committed length τ excludes the prefill anchor, N counts non-root candidate nodes, and P_{tree}, P_{prep} and P_{exec} distinguish retained parents, prepared parent distributions and executed projection rows including padding. None is a measured target-attention padding count. Separate synchronized profiles do not enter these tables.

Table 4: Primary native work measurements: Qwen3-4B, 144 test prompts, 256 output tokens, two request-seed sets and three timing repetitions. Raw/depth aliases share one row at $T = 0$, B4. Fixed at $T = 1$ is the optimized same-builder static leaf tree. Main-text Table 1 gives paired throughput intervals.

Policy	Tokens/s	τ	N	P_{tree}	P_{prep}	P_{exec}
Qwen3-4B, $T = 0$, B4						
Chain	1079.2	4.74	6.88	6.88	6.88	7.05
Fixed	1132.7	5.13	13.52	9.72	9.72	9.92
Raw / Depth	1026.5	4.98	7.03	6.52	7.24	7.61
All-edge fit	1034.0	5.31	10.93	7.55	11.17	11.72
Surviving fit	1043.5	5.18	8.46	7.21	8.59	8.94
Qwen3-4B, $T = 0$, B8						
Chain	1706.8	4.68	6.88	6.88	6.88	7.32
Fixed	1854.3	5.00	10.70	7.83	7.83	8.28
Raw	1634.8	4.96	7.02	6.52	7.22	7.90
All-edge fit	1595.4	5.25	10.89	7.49	11.16	12.37
Surviving fit	1630.1	5.46	12.63	9.74	12.16	13.64
Depth	1630.8	5.30	10.49	8.12	10.87	12.01
Qwen3-4B, $T = 1$, B4						
Chain	958.0	4.54	6.89	6.89	6.89	7.08
Fixed	1036.1	4.91	13.52	6.87	6.87	7.02
Raw	902.2	4.70	7.72	6.47	7.85	9.02
All-edge fit	906.2	5.08	11.73	7.35	11.90	14.03
Surviving fit	914.6	4.96	9.34	7.12	9.36	10.82
Depth	911.5	4.71	7.97	5.91	8.64	10.10
Confidence threshold	1021.8	4.85	10.35	6.87	6.87	7.05
Confidence budget	1023.2	4.84	10.72	6.88	6.88	7.07
Qwen3-4B, $T = 1$, B8						
Chain	1562.0	4.55	6.88	6.88	6.88	7.31
Fixed	1641.5	4.90	13.54	6.88	6.88	7.24
Raw	1433.1	4.75	7.74	6.49	7.87	9.85
All-edge fit	1440.6	5.14	11.74	7.36	11.88	15.36
Surviving fit	1379.7	5.29	14.33	9.72	13.65	17.55
Depth	1416.0	4.82	8.79	6.96	8.90	11.27
Confidence threshold	1651.9	4.82	9.55	6.87	6.87	7.27
Confidence budget	1610.0	4.78	9.76	6.88	6.88	7.26

Table 5: Model and temperature transfer with unchanged 4B, $T = 1$, B8 policy settings. Each model uses its own released target/drafter pair; the temperature transfer uses Qwen3-4B. Every stratum retains all 144 test prompts and the primary output/seed/timing budgets.

Policy	Tokens/s	τ	N	P_{tree}	P_{prep}	P_{exec}
Qwen3-8B, $T = 1$, B8						
Chain	1570.4	5.12	6.90	6.90	6.90	7.25
Fixed	1684.5	5.49	13.58	6.89	6.89	7.18
Raw	1371.3	5.20	7.81	6.59	7.92	9.95
All-edge fit	1365.1	5.45	11.82	7.55	11.98	15.55
Surviving fit	1381.8	5.77	14.33	9.95	13.67	17.68
Depth	1412.9	5.35	8.98	7.19	9.07	11.60
Confidence threshold	1663.7	5.40	9.39	6.89	6.89	7.17
Confidence budget	1636.5	5.37	9.78	6.89	6.89	7.20
Qwen3-14B, $T = 1$, B8						
Chain	965.4	4.56	6.88	6.88	6.88	7.30
Fixed	1036.6	4.86	13.53	6.87	6.87	7.24
Raw	944.3	4.66	7.48	6.14	7.70	9.70
All-edge fit	931.1	4.88	12.08	7.09	12.32	16.04
Surviving fit	950.6	5.19	14.64	9.57	14.10	18.32
Depth	1009.0	4.88	8.86	6.86	9.02	11.58
Confidence threshold	1021.3	4.78	9.58	6.87	6.87	7.26
Confidence budget	1002.8	4.76	9.76	6.88	6.88	7.27
Qwen3-4B, $T = 0.5$, B8						
Chain	1530.9	4.64	6.87	6.87	6.87	7.26
Fixed	1698.1	5.08	13.54	6.88	6.88	7.25
Raw	1401.3	4.91	7.14	6.47	7.35	8.71
All-edge fit	1398.0	5.28	11.08	7.52	11.29	13.98
Surviving fit	1322.5	5.45	13.14	9.77	12.64	15.48
Depth	1433.8	5.07	8.07	7.02	8.26	9.86
Confidence threshold	1663.5	4.99	9.47	6.87	6.87	7.25
Confidence budget	1658.1	4.97	9.75	6.87	6.87	7.27

Table 6: Additional-dataset work measurements on the same 128 Arena-Hard-v2/LiveCodeBench prompts at two truncation limits. The 4,096-token limit is an upper bound, not a uniform long-input workload. Its complete rerun uses the documented prefill lifetime repair; each stratum uses one matched source version and failed timings are excluded.

Policy	Tokens/s	τ	N	P_{tree}	P_{prep}	P_{exec}
Qwen3-4B, dataset transfer, limit 512, $T = 1$, B8						
Chain	1315.1	3.78	6.88	6.88	6.88	7.32
Fixed	1356.6	4.08	13.56	6.88	6.88	7.25
Raw	1146.7	3.88	7.19	5.72	7.52	9.58
All-edge fit	1136.1	4.31	12.26	6.67	12.57	16.44
Surviving fit	1108.9	4.39	15.08	9.15	14.61	19.15
Depth	1235.9	4.14	8.65	6.46	8.93	11.53
Confidence threshold	1352.5	4.05	9.94	6.88	6.88	7.22
Confidence budget	1353.9	4.01	9.77	6.88	6.88	7.25
Qwen3-4B, dataset transfer, limit 4,096, $T = 1$, B8						
Chain	1171.0	3.74	6.89	6.89	6.89	7.37
Fixed	1229.9	4.04	13.57	6.89	6.89	7.31
Raw	1076.6	3.89	7.20	5.75	7.53	9.54
All-edge fit	1087.9	4.30	12.15	6.62	12.45	16.29
Surviving fit	1047.6	4.41	14.87	9.07	14.40	18.80
Depth	1094.7	4.06	8.55	6.36	8.82	11.31
Confidence threshold	1212.8	4.02	9.84	6.88	6.88	7.30
Confidence budget	1219.7	3.98	9.77	6.88	6.88	7.33

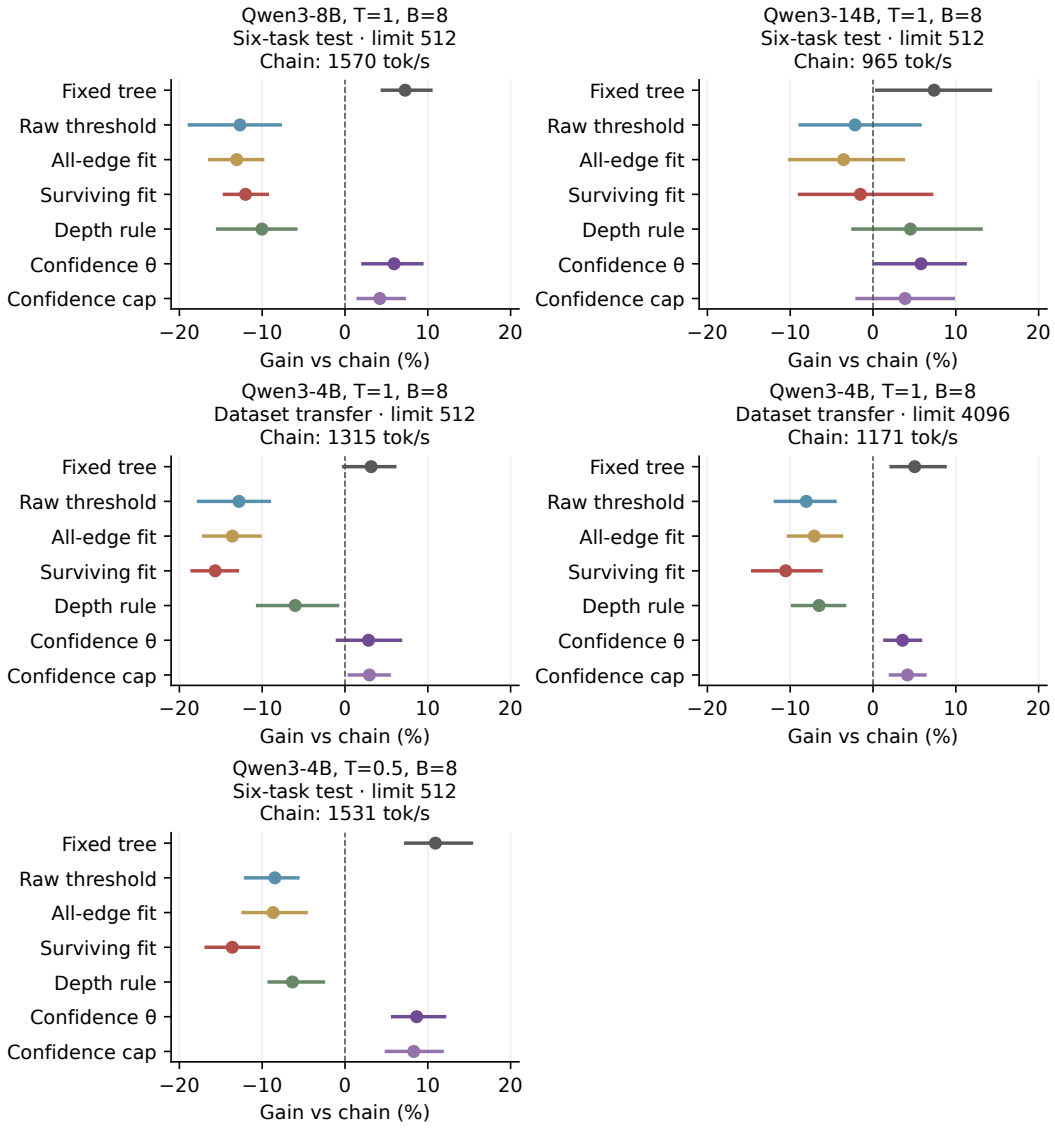

Figure 5: All frozen transfer families relative to their matched chain. Points and bars are percentage changes and paired 95% input-batch bootstrap intervals. Seeds and timing repetitions remain nested within input batches (18 six-task or 16 additional-dataset clusters). Each panel states its own model, temperature and input limit; timings are not pooled across panels. Intervals are descriptive and do not adjust for multiple comparisons. No configuration is chosen from these results.

C.1 SERVING CONTROLS AND LATENCY

The complete curves retain both constant thresholds for each controller, alongside chain and the selected fixed tree. Every rate/temperature/arm uses six independent Poisson traces of 256 requests. Output-window throughput covers the same scheduled-arrival 20–90% window; the latency cohort is followed to completion. Whole-trace τ and makespan throughput are distinct estimands. Positive backlog growth indicates finite-horizon overload, so these latencies are not stationary queueing estimates.

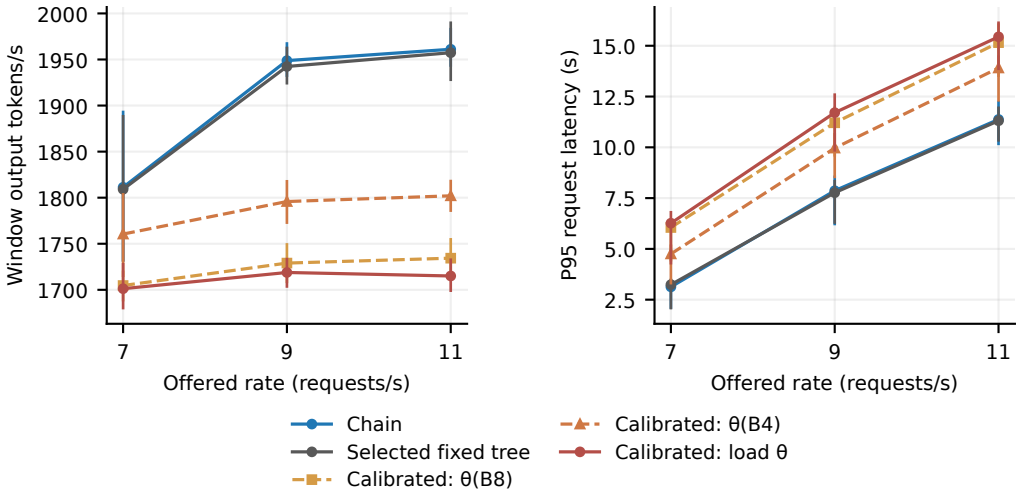

Figure 6: Greedy serving, all five frozen controls. Error bars are trace-bootstrap 95% intervals for each arm's window throughput and cohort P95 latency. A load controller can approach a constant rule when its active-batch occupancy concentrates in one bin.

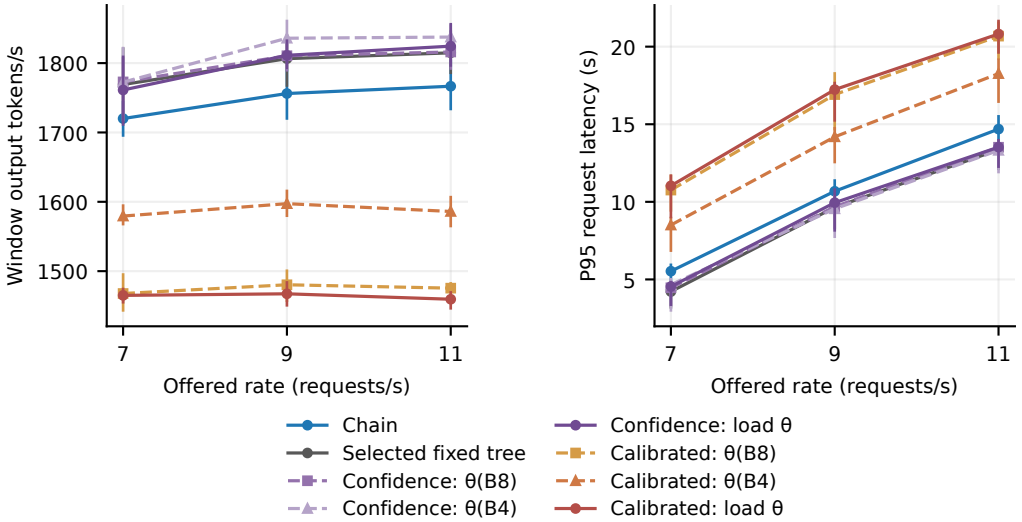

Figure 7: Sampling serving at $T = 1$, all eight frozen controls. Constant-threshold controls use the same strategy and execution path as their load controller. The six whole arrival traces are the bootstrap units; individual requests are not independent replicates.

Table 7: All seven predeclared greedy serving contrasts. Each cell gives percentage change in window throughput (candidate/reference minus one) and its paired 95% whole-trace interval. Cal. denotes calibrated best-first; θ_4 and θ_8 mean constant use of the validation-selected threshold for that batch bin. Positive values favor the candidate. These are marginal intervals without multiplicity adjustment.

Candidate / reference	7 req/s	9 req/s	11 req/s
Cal. load / Chain	-6.1 $[-9.3, -2.6]$	-11.8 $[-12.4, -11.1]$	-12.5 $[-13.7, -11.7]$
Cal. load / Fixed	-6.0 $[-9.1, -2.7]$	-11.5 $[-12.0, -11.0]$	-12.4 $[-13.9, -11.3]$
Cal. load / Cal. θ_8	-0.2 $[-1.5, +1.5]$	-0.6 $[-1.0, -0.2]$	-1.1 $[-2.4, +0.1]$
Cal. load / Cal. θ_4	-3.4 $[-5.3, -1.4]$	-4.3 $[-4.8, -3.8]$	-4.8 $[-5.4, -4.3]$
Fixed / Chain	-0.1 $[-0.4, +0.1]$	-0.3 $[-0.7, +0.1]$	-0.2 $[-0.9, +0.4]$
Fixed / Cal. θ_8	$+6.1$ $[+2.1, +10.3]$	$+12.3$ $[+11.5, +13.1]$	$+12.9$ $[+11.7, +14.0]$
Fixed / Cal. θ_4	$+2.8$ $[+0.8, +5.0]$	$+8.2$ $[+7.7, +8.7]$	$+8.6$ $[+7.4, +10.1]$

Table 8: All nineteen predeclared sampling serving contrasts, with the same estimand and interval construction as Table 7. Conf. denotes confidence leaves. The comparison of the two load controllers is retained alongside each controller's constant-threshold and fixed-tree controls.

Candidate / reference	7 req/s	9 req/s	11 req/s
Conf. load / Chain	$+2.4$ $[+0.6, +4.4]$	$+3.1$ $[+1.8, +4.8]$	$+3.3$ $[+1.7, +5.1]$
Conf. load / Fixed	-0.4 $[-1.3, +0.3]$	$+0.3$ $[-1.0, +1.9]$	$+0.5$ $[+0.3, +0.8]$
Conf. load / Conf. θ_8	-0.7 $[-2.2, +0.7]$	$+0.0$ $[-0.8, +0.8]$	$+0.5$ $[-0.2, +1.0]$
Conf. load / Conf. θ_4	-0.6 $[-1.9, +0.4]$	-1.3 $[-2.0, -0.5]$	-0.7 $[-1.7, +0.1]$
Conf. load / Cal. θ_8	$+20.0$ $[+15.6, +24.6]$	$+22.4$ $[+20.4, +24.3]$	$+23.7$ $[+21.8, +25.7]$
Conf. load / Cal. θ_4	$+11.5$ $[+7.8, +15.4]$	$+13.4$ $[+12.3, +14.7]$	$+15.0$ $[+12.4, +17.3]$
Conf. load / Cal. load	$+20.2$ $[+16.8, +23.7]$	$+23.4$ $[+22.2, +24.6]$	$+25.0$ $[+22.3, +28.4]$
Cal. load / Chain	-14.8 $[-16.0, -13.5]$	-16.4 $[-17.4, -15.4]$	-17.4 $[-19.9, -15.1]$
Cal. load / Fixed	-17.2 $[-19.6, -14.7]$	-18.8 $[-19.5, -17.6]$	-19.6 $[-21.6, -17.8]$
Cal. load / Conf. θ_8	-17.4 $[-19.0, -15.5]$	-19.0 $[-19.2, -18.7]$	-19.6 $[-21.4, -18.2]$
Cal. load / Conf. θ_4	-17.3 $[-19.7, -14.9]$	-20.1 $[-20.8, -19.2]$	-20.6 $[-22.1, -19.5]$
Cal. load / Cal. θ_8	-0.2 $[-1.8, +1.4]$	-0.9 $[-2.5, +1.1]$	-1.1 $[-2.4, +0.1]$
Cal. load / Cal. θ_4	-7.2 $[-8.0, -6.4]$	-8.1 $[-8.9, -7.4]$	-8.0 $[-10.0, -6.2]$
Fixed / Chain	$+2.9$ $[+1.0, +4.6]$	$+2.9$ $[+2.1, +3.8]$	$+2.7$ $[+1.0, +4.7]$
Fixed / Conf. θ_8	-0.2 $[-1.8, +0.8]$	-0.2 $[-1.4, +0.5]$	-0.1 $[-0.6, +0.3]$
Fixed / Conf. θ_4	-0.2 $[-0.8, +0.6]$	-1.6 $[-3.4, -0.3]$	-1.2 $[-2.2, -0.3]$
Fixed / Cal. θ_8	$+20.5$ $[+15.9, +25.5]$	$+22.0$ $[+18.6, +24.9]$	$+23.0$ $[+21.2, +25.0]$
Fixed / Cal. θ_4	$+12.0$ $[+8.2, +15.9]$	$+13.1$ $[+11.8, +14.4]$	$+14.4$ $[+11.7, +16.9]$
Fixed / Cal. load	$+20.8$ $[+17.2, +24.4]$	$+23.1$ $[+21.4, +24.3]$	$+24.3$ $[+21.6, +27.5]$

Table 9: Greedy output-window throughput, whole-trace τ, latency (arrival to completion, seconds), and mean change in backlog between the common window boundaries. All constant-threshold and adaptive controls are retained. Cal./Conf. denote calibrated/confidence policies; θ_4 and θ_8 denote constant use of the corresponding validation-selected threshold.

Req/s	Policy	Tok/s	τ	Mean (s)	P95 (s)	P99 (s)	Δ backlog
7	Chain	1811.27	4.71	1.65	3.13	3.63	-5.67
7	Fixed	1809.14	4.96	1.70	3.24	3.74	-4.83
7	Cal. θ_8	1704.64	5.49	3.30	6.05	6.61	5.33
7	Cal. θ_4	1760.51	5.16	2.54	4.75	5.38	-0.33
7	Cal. load	1701.22	5.46	3.27	6.26	7.05	6.83
9	Chain	1948.77	4.72	4.65	7.86	8.77	25.33
9	Fixed	1942.41	4.96	4.63	7.77	8.52	25.67
9	Cal. θ_8	1729.01	5.48	7.08	11.21	12.20	41.33
9	Cal. θ_4	1795.82	5.15	6.20	9.96	10.80	36.67
9	Cal. load	1718.82	5.48	7.20	11.71	12.82	42.17
11	Chain	1961.09	4.72	7.32	11.37	12.50	52.00
11	Fixed	1957.38	4.95	7.30	11.31	12.29	51.50
11	Cal. θ_8	1734.34	5.50	9.73	15.15	16.36	66.17
11	Cal. θ_4	1802.03	5.15	8.89	13.91	14.77	60.83
11	Cal. load	1715.03	5.47	9.92	15.44	16.44	66.67

Table 10: Sampling serving latency and backlog, with the same definitions as Table 9. Negative or uncertain policy differences remain part of the comparison.

Req/s	Policy	Tok/s	τ	Mean (s)	P95 (s)	P99 (s)	Δ backlog
7	Chain	1720.05	4.55	2.96	5.53	6.17	4.50
7	Fixed	1769.12	4.91	2.44	4.22	4.70	-0.83
7	Conf. θ_8	1772.88	4.82	2.53	4.46	5.01	-0.83
7	Conf. θ_4	1772.56	4.87	2.37	4.66	5.21	-0.67
7	Conf. load	1761.24	4.83	2.60	4.54	5.02	0.67
7	Cal. θ_8	1467.75	5.27	6.69	10.76	11.91	29.50
7	Cal. θ_4	1579.44	4.98	4.92	8.50	9.22	18.17
7	Cal. load	1465.06	5.27	6.67	11.03	11.90	29.50
9	Chain	1756.14	4.52	6.63	10.68	11.54	39.50
9	Fixed	1806.43	4.89	6.12	9.65	10.47	35.33
9	Conf. θ_8	1810.64	4.82	6.01	9.68	10.54	35.33
9	Conf. θ_4	1835.88	4.88	5.83	9.55	10.57	32.83
9	Conf. load	1811.24	4.83	6.02	9.94	10.98	35.00
9	Cal. θ_8	1480.36	5.25	10.81	16.91	18.70	59.50
9	Cal. θ_4	1597.32	5.00	9.02	14.19	14.94	51.50
9	Cal. load	1467.33	5.22	11.05	17.23	18.04	61.67
11	Chain	1766.65	4.54	9.35	14.69	15.76	63.33
11	Fixed	1814.83	4.90	8.68	13.36	14.06	60.33
11	Conf. θ_8	1816.05	4.82	8.69	13.52	14.21	60.50
11	Conf. θ_4	1837.56	4.88	8.57	13.32	14.07	59.50
11	Conf. load	1824.45	4.84	8.66	13.52	14.19	59.83
11	Cal. θ_8	1475.48	5.26	13.51	20.68	22.25	83.00
11	Cal. θ_4	1586.07	4.98	11.71	18.26	19.67	75.50
11	Cal. load	1459.50	5.27	13.67	20.82	22.00	84.50

Table 11: Measurement-window active-batch occupancy, weighted by tree request-rounds. Top: $T = 0$. Bottom: $T = 1$. Mean N is retained non-root nodes per request-round. These observations describe the executed controller, not a new threshold selection.

Req/s	Controller	$B \leq 4$ (%)	$B > 4$ (%)	Mean N
7	Calibrated, load theta	0.50	99.50	12.63
9	Calibrated, load theta	0.00	100.00	12.63
11	Calibrated, load theta	0.00	100.00	12.67

Req/s	Controller	$B \leq 4$ (%)	$B > 4$ (%)	Mean N
7	Confidence, load theta	1.66	98.34	9.53
7	Calibrated, load theta	0.00	100.00	14.33
9	Confidence, load theta	0.00	100.00	9.54
9	Calibrated, load theta	0.00	100.00	14.26
11	Confidence, load theta	0.00	100.00	9.53
11	Calibrated, load theta	0.00	100.00	14.33

C.2 NATIVE EOS STOPPING SENSITIVITY

This separate frozen comparison uses all 144 six-task test prompts, $T = 1$, initial batch eight, two request-seed sets and two timing repeats. Each initial batch drains without refill, and each request is capped at 256 output tokens. Native token 151645 stops generation; token 151643 does not. Terminal EOS counts as a committed token. This tests the engine's declared stop convention, not complete Hugging Face generation parity, uncapped response quality or a serving-arrival workload.

Table 12: EOS sensitivity. Length and stopping statistics count each prompt/seed once; throughput includes both timing repeats. Requests/s includes both EOS-ended and length-capped requests. The lower table gives percentage changes with paired 95% intervals from 10,000 initial-batch bootstrap resamples (18 clusters, with seeds/timings nested). Intervals are descriptive, with no multiplicity adjustment.

Policy	Tokens/s	Requests/s	Mean tokens	EOS (%)	Capped (%)
Chain	1634.4	7.75	210.8	39.6	60.4
Static leaves	1746.1	8.26	211.4	38.5	61.5
Confidence threshold	1698.9	8.02	211.7	39.6	60.4

Contrast	Tokens/s change (%)	Requests/s change (%)
Confidence threshold / Static leaves	-2.70 [$-5.44, -0.33$]	-2.86 [$-5.64, -0.40$]
Static leaves / Chain	$+6.84$ [$+4.33, +9.49$]	$+6.53$ [$+3.92, +9.37$]
Confidence threshold / Chain	$+3.95$ [$+0.73, +7.17$]	$+3.48$ [$-0.22, +6.99$]

D CORRECTNESS OF PREDICTABLE SAMPLED EXPANSION

We give the probability-mass argument underlying Proposition 1. Non-anticipating admission and a selection-bias counterexample already appear in Cheng et al. (2026, Appendix A); here we spell out the conditions for ordered siblings and hidden prefetching. The claim concerns the distribution obtained after averaging over proposal draws and verification randomness. It does not assert target-distributed output conditional on an arbitrary realized tree, whose token values already reveal proposal randomness.

The target is one fixed causal conditional kernel, including the chosen temperature and filtering rule. A node's target distribution must equal that kernel at its own ancestor prefix, and committed KV must represent that prefix. Proposal cancellation cannot compensate for incorrect target attention, positions, cache state, or a different nodewise target kernel.

Conditional proposal law and admissible information. The proposal condition is conditional, not merely marginal. Let $\mathcal{H}_t$ contain the committed prefix and all proposal values *exposed to construction* before the tth admission. It may include tokens on branches that will eventually be rejected. The next parent, its sibling rank, and the decision to admit or stop are measurable with respect to $\mathcal{H}_t$. If a slot at parent v and sibling rank j is admitted, its token must satisfy

$$\Pr(X_{v,j} = x \mid \mathcal{H}_t, \text{admit } (v, j)) = q_{v,j}(x). \tag{13}$$

Here $q_{v,j}$ is the base conditional at the already materialized parent path with its preceding sibling tokens removed and the remaining mass normalized. It is fixed when the draw is admitted and is not changed by later tree decisions. In the implementation it depends on the shared backbone logits, parent token, depth, temperature, and preceding siblings.

A sufficient implementation uses fresh proposal randomness conditional on $\mathcal{H}_t$, parent-before-child construction, and the original sibling draw order. Every admitted node remains eligible for verification; none is removed or reordered after inspecting its value or descendants. The complete construction is independent of the random variables used for verification. Batched or prefetched draws are permitted only when unadmitted values remain hidden from *all* admission decisions. For example, drawing K siblings in advance is safe when only an exposed prefix of that draw sequence can influence the next decision. Selecting a parent by comparing its still-unadmitted prefetched token against another parent's prefetched token violates Eq. 13, even if both tokens were initially drawn by a correct

sampler. Prefetching is an execution optimization, not permission to condition admission on unseen randomness.

One rejection stage. Fix a construction history just before an admitted proposal is drawn. Let p be the current residual target distribution and q the proposal specified at this history. Draw $X \sim q$ and independently draw the acceptance random variable. The acceptance probability at X is $\min\{1, p(X)/q(X)\}$. For each vocabulary item z, the unconditional mass emitted by acceptance at this stage is

$$q(z)\min\{1, p(z)/q(z)\} = \min\{p(z), q(z)\}. \tag{14}$$

Terms with $q(z) = 0$ contribute zero. The rejection probability is

$$Z = 1 - \sum_z \min\{p(z), q(z)\} = \sum_z [p(z) - q(z)]_+. \tag{15}$$

When $Z > 0$, define $r(z) = [p(z) - q(z)]_+/Z$. Any continuation that emits r after rejection completes the identity

$$\min\{p(z), q(z)\} + Zr(z) = p(z). \tag{16}$$

If $Z = 0$, rejection has probability zero and no residual needs to be normalized. This is the standard speculative-sampling identity (Leviathan et al., 2023; Chen et al., 2023).

Ordered siblings and stopping. After rejecting earlier siblings, the next sibling is drawn from the base conditional with the previously drawn tokens removed and the remaining mass renormalized. This is the proposal $q^{(j)}$ used in the acceptance ratio and residual update in Section 3.2. Although the proposal depends on preceding draws, identity 16 holds for each such history and its specified proposal. The tree has a finite node cap. Backward induction on the remaining proposal stages therefore proves that a sequence of valid rejection stages, followed by a draw from the final residual, emits the initial target distribution. Stopping immediately is the base case: sample the current residual directly. If the next slot is available, applying a valid stage and then a valid continuation preserves that same residual.

The slot-admission condition is needed for this induction. Randomness exposed on other branches may affect whether a later slot is admitted, but the new token still has the law in Eq. 13. Off-path construction can therefore be integrated out as auxiliary randomness. Stopping after earlier exposed proposals chooses a valid continuation or direct residual sampling; it cannot select the next proposal by its own value. Computing proposals before verification changes their evaluation time, not this generative law, provided the information restriction is respected. Conditioning on the complete final tree instead would generally change the proposal law and is not a valid step in this proof.

Descent and multiple rounds. The target distribution at a tree node conditions on the verified prefix and that node's root path. The event that a parent is accepted depends on its ancestors, preceding siblings at those ancestors, their fixed proposal kernels, and independent acceptance random variables. By causal construction order, none of these tests uses the parent's descendant draws. Subsequent expansion cannot change that event by deleting or reordering admitted nodes. The sibling argument thus applies again at the accepted parent, with its target conditional, including when construction on other branches influenced its available budget. At a leaf, the bonus token is sampled directly from the target conditional; when no sibling survives, the correction comes from the last residual. Composition over depth and subsequent rounds gives the target sequence distribution. EOS and a fixed output limit may stop that sequence using its already emitted tokens; they do not permit value-dependent changes to unverified proposals.

Counterexample: retaining only a favorable prefetched token. Let $p = (1/2, 1/2)$ and $q = (9/10, 1/10)$ over $\{a, b\}$. Draw $X \sim q$, retain the slot only when $X = a$, and otherwise sample directly from p. On the retained branch, use the ordinary p/q rejection rule. The probability of emitting a is then

$$\frac{9}{10}\frac{1/2}{9/10} + \frac{1}{10}\frac{1}{2} = \frac{11}{20} \neq \frac{1}{2}.$$

The first term is acceptance of the retained a; rejection produces only b. Both the initial sampler and the rejection formula are individually standard, but admission changed their joint law. Thresholding a

sampled node by its own score, or pruning a sampled supertree without accounting for this selection, can have this failure. It is not a limitation of deterministic candidate trees verified by direct target-sample matching.

Likewise, matching each candidate's *unconditional* marginal is insufficient. With an iid uniform binary target, reuse one fair coin for both tokens of a two-node chain while claiming a uniform proposal at each depth. Both ratios equal one, but the output contains only aa and bb, each with probability $1/2$, instead of all four pairs with probability $1/4$. Fresh conditional proposal laws exclude this construction.

An exact-rational check enumerates a binary adaptive tree in which a descendant controls admission of a later root sibling, checks the resulting two-token law, verifies the node-value identity in Proposition 2, and reproduces these counterexamples. These finite examples are checks of the stated reasoning, not a substitute for the general argument or an end-to-end numerical verification.

These statements concern the specified algorithm in exact arithmetic. Numerically consistent reconstruction of the proposals used for sampling is an implementation requirement. Calibration accuracy and the particular threshold are not assumptions of the proof.

Greedy and finite-precision scope. With the same causal target decisions and tie rule at every reached prefix, greedy tree verification follows the target-only greedy sequence. Repeated outputs within one execution mode do not establish agreement across modes. Small logit errors or small distributional KL also do not guarantee matching argmax near a tie. Any observed disagreement requires checking target evaluation and cache state at the first shared divergent prefix; the sampling proof does not establish bitwise finite-precision equivalence.

E Block Semantics and Proposal Storage

Next-token alignment. The released DSpark training code assigns label offsets $1, \dots, b$ to backbone rows $0, \dots, b-1$ after an anchor at position a. Its teacher-forced Markov inputs are the anchor followed by all but the last of those labels. Thus zero-based row $d-1$ predicts the token at $a+d$, conditioned by the Markov head on the parent at depth $d-1$. The input block itself contains the anchor and $b-1$ mask tokens at positions $a, \dots, a+b-1$; a row's input position and the position of its next-token label are different. The training definitions are `label_offsets` and `prev_token_ids` in the released Qwen3 DSpark model, and the inference helper `build_dspark_proposal` starts its sequential head evaluation with the anchor token.

Substituting a generated parent token for the teacher-forced parent gives Eq. 1, using U_d for zero-based tensor row $d-1$. In a tree, every node at depth d uses that same backbone row, while its Markov term uses its own parent token. This establishes the indexing rule without choosing an alignment from downstream speed measurements. The native chain and tree builders use this convention. Target verification separately evaluates each node at position $a+d$ with its own ancestors; the independent numerical checks in Appendix H compare these tree forwards against linear reference prefixes.

Retained proposals and work accounting. The current sampled builder retains a float32 base proposal distribution for each prepared parent needed by verification. Siblings share that immutable base distribution. The verifier removes earlier sibling tokens and renormalizes in draw order to obtain $q^{(j)}$ in Eq. 2; it does not replace the proposal by its calibrated allocation score. Retaining these snapshots also prevents a subsequent CUDA-graph replay from overwriting a proposal awaiting verification. This description concerns the current native implementation; historical recomputation variants are not used to describe its memory cost.

A node cap N bounds successful admissions, not all heap operations: lazy preparation can refresh and reinsert a slot before admission. Each new parent distribution requires a low-rank projection and vocabulary normalization; siblings reuse it, while the backbone is shared within a request-round. The materialized node count, prepared parent rows, and graph-executed rows including padding can therefore differ. Appendix H specifies the recorded counters and the scope of the numerical checks.

F Cross-Task Transfer of Greedy-Match Calibration

We evaluate whether the two-parameter calibration map transfers across the six benchmark tasks without collecting calibration examples from the evaluation task. This is *cross-task benchmark transfer*, not a test of previously unseen semantic domains: the datasets share mathematical, coding, and instruction-following content.

Protocol. The experiment uses the aligned native Qwen3-4B/DSpark block-7 collector at $T = 0$, with 12 calibration and 12 validation prompts per task. Generation is limited to 128 tokens, EOS stopping is disabled, and inputs longer than 512 tokens are left-truncated after chat templating. Trees use a 64-node cap, branching cap 8, and $\theta = 0$. We retain edges whose actual tree ancestors match the target's greedy tokens; the binary label indicates whether the edge itself matches. There are 19,305 calibration and 19,144 validation edges in this population. For each of six folds, we fit $\hat{p} = \sigma(a \operatorname{logit}(q_0) + b)$, $a > 0$, on the other five tasks' 60 calibration prompts and evaluate only the excluded task's 12 validation prompts. Every fit minimizes mean binary cross-entropy plus $10^{-6}[(a-1)^2 + b^2]$, with q_0 clipped to $[10^{-8}, 1 - 10^{-8}]$. Comparators are raw q_0 and the frozen fit from all 72 calibration prompts ($a = 0.6595$, $b = -0.4153$). Neither test/domain-holdout prompts nor threshold tuning enter this experiment.

The calibration collector contains 69,167 edges, including 19,305 with matching ancestors. The surviving-edge fit is $(\alpha, \beta) = (0.65954, -0.41530)$; the all-edge fit is $(0.36697, -0.80253)$. Both maps use the same labels and functional family and are frozen before validation.

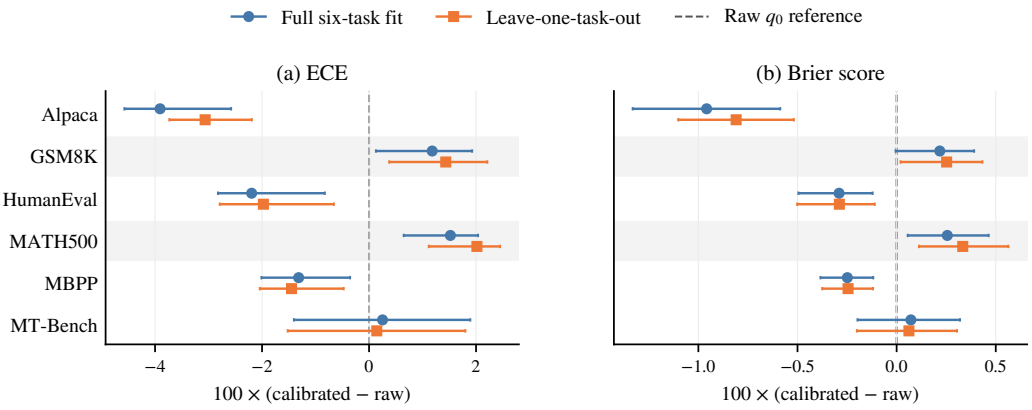

Figure 8: Cross-task calibration transfer on validation surviving edges. Points show calibrated-minus-raw ECE (15 equal-width bins) and Brier score; both axes multiply differences by 100, and negative values are better. Error bars are paired 95% percentile intervals from 2,000 prompt-cluster bootstrap samples, conditional on fixed fitted coefficients. Shaded rows identify GSM8K and MATH500, where leave-one-task-out calibration worsens both metrics. Each row has 12 validation prompts.

Results and limits. LOTO reduces pooled ECE from 0.02586 to 0.01076 and Brier score from 0.06358 to 0.06225 (Table 13). The paired LOTO-minus-raw differences are -0.01510 $[-0.02147, -0.00744]$ for ECE and -0.00133 $[-0.00225, -0.00042]$ for Brier, using a task-stratified prompt bootstrap. However, GSM8K and MATH500 exhibit negative transfer in both metrics (Figure 8); MT-Bench's small increases have intervals spanning zero. ECE improvement is smaller when averaging tasks separately (0.03045 to 0.02563), since pooled bin residuals can cancel across tasks. The full-six-task fit is modestly better than LOTO on pooled Brier (0.06182 versus 0.06225). These results support aggregate transfer within this benchmark mixture, not uniform improvement across tasks. The folds share calibration data; intervals resample validation prompts with the fitted coefficients held fixed and do not include refitting uncertainty. This diagnostic establishes neither sampling-temperature transfer nor a decoding-throughput improvement.

	ECE ↓			Brier ↓		
Task	Raw	Full six	LOTO	Raw	Full six	LOTO
Alpaca	.0691	.0300	.0384	.1156	.1060	.1075
GSM8K	.0183	.0301	.0326	.0448	.0470	.0473
HumanEval	.0326	.0107	.0129	.0642	.0613	.0613
MATH500	.0156	.0308	.0357	.0406	.0432	.0439
MBPP	.0285	.0153	.0140	.0523	.0498	.0498
MT-Bench	.0187	.0212	.0201	.0582	.0589	.0588
Pooled	.0259	.0105	.0108	.0636	.0618	.0623
Task mean	.0305	.0230	.0256	.0626	.0610	.0614

Table 13: Measured validation metrics. Full six uses all tasks' calibration prompts; LOTO excludes the evaluation task during fitting. Pooled metrics weight edges equally; task mean averages six separately computed metrics.

Table 14: Greedy-match calibration on the same 19,144 ancestor-matching validation edges from 72 prompts. Brackets are 95% task-stratified prompt-cluster intervals, conditional on the frozen fits. All three maps have AUC 0.9735: monotone calibration changes probabilities, not single-edge ranking.

Predictor / fitting population	ECE [95% CI]	Brier score
Raw Markov q_0	0.0259 [0.0219, 0.0315]	0.0636
Platt map / all edges	0.0702 [0.0644, 0.0766]	0.0731
Platt map / surviving edges	0.0105 [0.0083, 0.0155]	0.0618

G SAMPLING-TIME VALUE DIAGNOSTICS ON REALIZED TREES

Protocol and estimand. We evaluate four frozen surrogates on the *same* sampled trees from Qwen3-4B with its released DSpark block-7 drafter: 72 validation prompts (12 per task), 1,560 rounds, and 72,333 materialized nodes. Construction uses raw-score best-first expansion, a maximum of 64 proposed nodes, at most eight siblings, and $\theta = 0$, with $T = 1$, no top-k/top-p filtering, and EOS disabled. Prompts are capped at 512 tokens and outputs at 128 tokens. No new head is fitted or alternative allocation policy executed in this diagnostic.

The reference value is verifier-conditioned commitment probability $\rho(v)$ from Eq. 9, evaluated using the actual target and proposal distributions. Round value is the conditional potential length $\tau_{\mathcal{T}} = 1 + \sum_v \rho(v)$, whose mean is 5.9004 (95% CI $[5.6643,\ 6.1601]$). Generally, output truncation requires Eq. 11. In this artifact, every node has depth below its round's remaining output budget, so the cap does not further reduce this conditional expectation. It remains an expectation over verification randomness, not an observed token count or a target-distribution claim conditional on a fixed tree.

Four prediction rules. Raw and greedy-calibrated scores multiply $q_0(x_v)$ and $\hat{p}(q_0(x_v))$, respectively, along the actual ancestry. The latter uses the frozen surviving-edge fit from disjoint $T = 0$ calibration prompts ($\alpha = 0.6595$, $\beta = -0.4153$). For both geometric rules, initialize $m(\text{root}) = 1$ and assign the jth ordered child of v the score $m(v_j) = m(v)a_v(1-a_v)^{j-1}$. One rule fixes $a_v = 0.8$; the other uses the released parent confidence c_v, whose original training target is first-draw distributional overlap (Cheng et al., 2026). Reusing that scalar for later residual hazards is a surrogate assumption, not an exact RRS identity.

Table 15: Fixed-tree $T = 1$ prediction diagnostics. Bias is predicted minus verifier-conditioned round value; MAE is mean absolute error. Bracketed intervals are prompt-cluster 95% confidence intervals. No row reports execution speed.

Surrogate	Mean $\widehat{\tau}$	Round bias	Round MAE	Node MAE
Raw q path	6.7787	$+0.8782$ $[0.7513, 1.0099]$	1.0295 $[0.9092, 1.1527]$	0.0690
Greedy-calibrated q path	5.8754	-0.0250 $[-0.1339, 0.0874]$	0.9888 $[0.9182, 1.0584]$	0.0642
Constant 0.8 geometric	4.7012	-1.1992 $[-1.4151, -1.0044]$	1.9820 $[1.8979, 2.0748]$	0.0867
Released-confidence geometric	6.2183	$+0.3179$ $[0.2249, 0.4136]$	0.7725 $[0.6930, 0.8504]$	0.0553

Interpretation and uncertainty. The greedy-calibrated rule nearly matches the aggregate round mean, but this does not imply accurate values for individual rounds or nodes (Figure 9). Released-confidence geometric scoring lowers round MAE relative to that rule by a paired difference of -0.2163 (95% CI $[-0.2522, -0.1824]$); the node-MAE difference is -0.0089 ($[-0.0107, -0.0072]$). It still overpredicts total value on average, and underpredicts later sibling ranks. Its advantage over the constant-0.8 rule does not establish superiority to tuned static priors: 0.8 was not optimized for this population. The results motivate a sampling-specific allocation experiment; they do not establish that changing the policy improves throughput or generalizes to another model or workload.

Intervals use 2,000 percentile-bootstrap replicates, resampling whole prompts within each task and keeping their rounds and nodes together; paired differences use the same resamples. Metrics pool nodes or rounds, as indicated, rather than treating them as independent samples. Uncertainty is conditional on the frozen fit, collected trees and model, and does not include hardware or transfer variation.

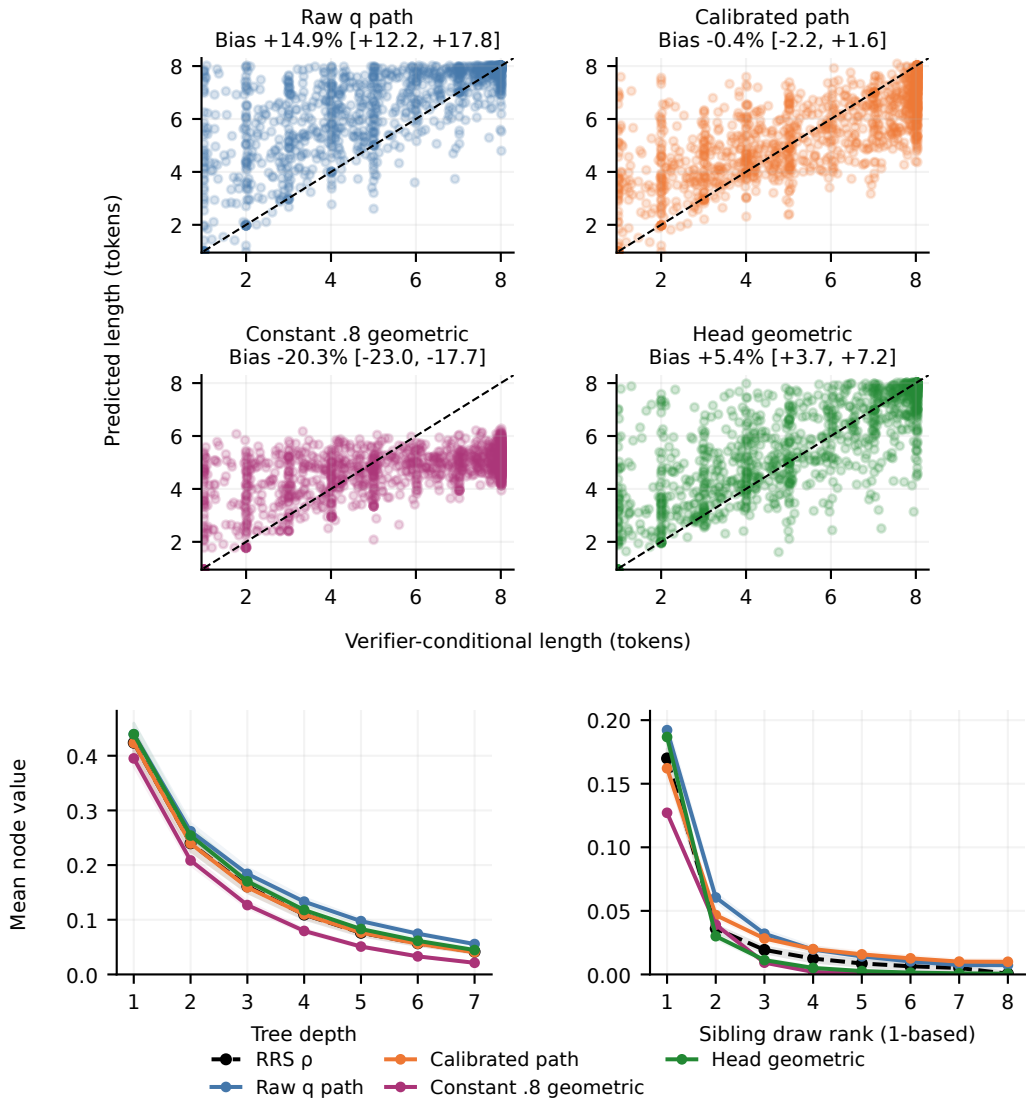

Figure 9: Sampling-value diagnostics on the same 72 validation prompts. **Top:** each point is one realized tree; the diagonal indicates agreement between predicted and verifier-conditioned potential length. Panel titles report relative mean bias with prompt-cluster 95% intervals. **Bottom:** node values averaged by depth and ordered sibling rank; bands show prompt-cluster 95% intervals. These are forecasts on the original trees, not trajectories produced by the four scoring policies.

G.1 Scalar overlap does not identify conditional leaf utility

Even a perfect prediction of first-draw overlap does not determine the value of an optional second leaf after the primary token is observed. Consider a depth-one primary chain, before EOS or output truncation, with

$$q = (1/3, 1/3, 1/3), \qquad p_A = (0, 1/3, 2/3), \qquad p_B = (1/3, 0, 2/3).$$

Both targets have overlap $c = \sum_z \min\{p(z), q(z)\} = 2/3$. Given the observed primary draw $X = 1$, the second proposal is $q^{(2)} = (0, 1/2, 1/2)$. Admit its slot without observing the second token or its hidden randomness, and use independent verification randomness. For a fixed primary x that can reject, the expected additional token count from this leaf is

$$G(x; p, q) = (1 - \min\{1, p(x)/q(x)\}) \sum_z \min\{r(z), q^{(2)}(z)\}, \qquad r = \mathrm{norm}([p - q]_+).$$

Set $G = 0$ when the primary accepts surely, without constructing an empty residual. A successful leaf adds one token relative to the corresponding chain round: the accepted leaf is followed by a bonus token.

Under p_A, the primary rejects surely and $r_A = (0, 0, 1)$, giving $G(1; p_A, q) = 1/2$. Under p_B, it accepts surely, giving $G(1; p_B, q) = 0$. The geometric surrogate $(1 - c)c = 2/9$ is identical for both. Thus even the full q, observed primary identity, and exact scalar overlap do not identify this conditional utility. The confidence score remains a surrogate rather than a certified residual hazard or an optimal allocation rule.

Any deterministic score using only $(q, c, X = 1)$ returns the same value s in these two cases, so $\max\{|s|, |s - 1/2|\} \geq 1/4$ by the triangle inequality. This is a worst-case conditional bound for that information set; it neither bounds model-average error from below nor rules out useful confidence heuristics or additional features.

The conditioning matters: averaging over $X \sim q$ gives $1/6$ for both targets, which are permutations under a uniform proposal. This pair therefore establishes conditional non-identifiability, not an unconditional utility difference. It also does not invalidate exact sampling: averaged over proposal draws, RRS retains the target law under each predictable admission rule. Exact enumeration verifies the stated fractions and the target first-token law under three rules: never admit the leaf, always admit it, or admit it only when $X = 1$.

H Implementation and Numerical Checks

Independent target reference. We compare the native target against the Transformers implementation of the same checkpoint, loading the two implementations sequentially on one H100 80GB. Four fixed diagnostic prompts exercise prefill, three cached decode steps, and a three-node branching forward, giving 28 comparisons per model. Each branching node is also evaluated with its own linear prefix in the reference. We compare vocabulary logits and the target hidden-state taps consumed by the drafter. The diagnostic gates are finite outputs, target-to-native KL at most 0.02, and relative RMS error at most 0.05 for each tap. These gates were fixed before interpreting the corrected implementation's results; they do not assert bitwise equality.

Target	Comparisons	Matching argmax	Maximum KL
Qwen3-4B	28	28	0.00516
Qwen3-8B	28	28	0.00759
Qwen3-14B	28	28	0.00737

Table 16: Completed numerical diagnostics on four reference prompts per checkpoint. Every comparison also satisfies the hidden-tap error gate. These small checks are implementation evidence, not a test of task accuracy or a proof of end-to-end distributional equivalence in finite precision.

The audit found two implementation details that matter for reproducibility: the 8B and 14B checkpoints require their untied output heads, and the fused BF16 RMSNorm must round the normalized activation before multiplying its weight to match the reference cast order. We corrected these details

Native FP32 path	Maximum KL	Matching argmax
Full-prefix prefill	8.23×10^{-10}	$6/6$
Teacher-forced cached decode	5.89×10^{-9}	$6/6$
Rebuilt forest, eager	3.80×10^{-9}	$18/18$

Table 17: Selected-prefix numerical diagnosis. KL is directed from HF FP32 to native FP32 after unit-temperature softmax. Eighteen forests share six reference prefixes; they are not independent samples.

and repeated calibration; earlier calibration artifacts are superseded rather than pooled with the corrected results. The table uses a later isolated decode-accumulation correction described below; all three short checks were repeated under the same diagnostic gates. This correction leaves the tree path used by calibration and timed policies unchanged.

Numerical fidelity beyond trace self-consistency. During the implementation audit, long greedy replays reproduced all 720 recorded requests and 184,320 emitted tokens, but exposed 318 first-disagreement pairs with identical prefixes across configurations, spanning 115 prompts. Every emitted token matched its own recorded target argmax; this establishes trace self-consistency, not a common target distribution across execution paths. Fresh BF16 Hugging Face (HF) references also exposed outliers: maximum HF-to-native KL was 0.607 with SDPA and 0.776 with eager attention over the selected saved snapshots.

We then froze six diagnostic prefixes and 18 recorded forests, promoting the original BF16 checkpoint values to FP32. Native parameters, floating buffers and KV used FP32; TF32 was disabled, and prefill/tree dot products explicitly used IEEE precision. Native cache slots, positions and ancestry construction were retained. Table 17 compares native paths with HF FP32 eager full-prefix evaluation.

This convergence supports consistency of the tested native equations and reconstructed masks, with arithmetic precision a principal explanation for the earlier discrepancies. It does not isolate a single operation: several numerical components change together. The prefixes were deliberately selected for diagnosis, including a post-EOS case, and do not estimate population error. Unit-temperature softmax KL describes logit geometry, not the temperature-zero output distribution. Reconstructed forests use freshly recomputed prefix KV; historical BF16 KV contents and graph replay remain unverified by this probe. These observations establish neither global sampling equivalence nor throughput, and introduce no numerical acceptance threshold. A separate compiler audit found BF16 multiplication and accumulation in the regular single-token decode QK reduction. Promoting its operands before the reduction lowers the two largest selected cached KL values relative to the FP32 reference from $0.858/1.057$ to $0.006/0.080$, without monotone improvement on every prefix. All 60 collected fresh and rebuilt-tree vectors remain bitwise unchanged, as expected: timed chain and tree policies share the tree attention path. We adopt this isolated decode correction. A further FP32-PV candidate has mixed full-model effects and is not adopted. The correction does not establish bitwise cross-tree equality in BF16; the exact-arithmetic sampling statement remains conditional on a common causal target law.

Incomplete prefill chunks. The 4,096-token input-limit transfer exposed a separate lifetime error: a prefill chunk that completes no request did not retain its input buffers behind a completion event. Those buffers could be reused on their allocation stream before the compute stream finished. The original asynchronous replay fails after 24 batches; a blocking reference and a conditional return-stream dependency both complete 32 diagnostic batches, with identical outputs and request RNG states. A delayed-consumer regression corrupts two of three original cases and none of the three repaired cases. Diagnostic timings are excluded from performance estimates. The long-context recovery uses the two-line scheduler repair in a separately frozen source snapshot, retaining all original policies and the complete evaluation protocol. Other strata use their recorded source and 512-token prompt limit; no comparison pools source versions. Appendix B records this correction separately from the decode-precision change.

Shared execution and measured work. The engine uses paged tree attention with request-local ancestry masks, batched proposal preparation and verification, and batched KV commits. The target

graph capacity is 1,024 tokens for every timed arm, with common buckets including 520 tokens for eight requests with 64 proposed nodes plus anchors. Padding uses negative write slots, so it cannot overwrite live cache entries. Only the verified path is committed; unused tree slots are released. Each run checks full KV recovery, graph-cache stability, and unchanged source snapshots. Same-seed offline timing repetitions must reproduce token outputs and final request RNG states.

Node count alone is not a sufficient cost measure. We separately record parents retained in the final tree, nonpadding parent distributions actually prepared, and executed graph rows including padding. Preparing an unused parent is still charged to the method. Instrumented static and leaf builders count actual eager projection rows or captured projection shapes at replay; scalar confidence projections are excluded from parent-query counts. Uninstrumented historical counters remain missing rather than being replaced by theoretical counts. Separate synchronized profiles diagnose costs and are excluded from throughput estimates.

Leaf-builder reproducibility. The confidence-guided leaf candidate shares a fixed tensor bank across graph records for depths up to seven and request buckets of one, two, four, and eight. Graphs are captured only during explicit warm-up; a missing graph uses eager execution. Replays are sequential, request RNGs generate independent parent races, and returned float32 proposal snapshots are cloned before another replay can overwrite the bank. The candidate reuses the frozen DSpark confidence head, with factored float32 evaluation; no new head is fitted.

Completed GPU checks. With real checkpoint weights and synthetic drafter inputs, 78 interleaved shared-pool cases passed across 168 captured graphs. Checks cover greedy and positive temperatures, padded batches, short depths, mixed groups, 11-request chunking, support exhaustion, and a missing-graph fallback. Tokens and request RNG states match eager execution, retained proposals survive subsequent replays, and the graph inventory stays fixed. A small GPU engine with controlled target logits passed 13 regression scenarios covering EOS, short outputs, mixed sampling, request RNG isolation, prefix-cache reuse, eviction, and target weight reload, with full KV recovery. A separate Qwen3-4B integration smoke completed 72 timing rows over eight validation prompts, six arms, temperatures zero and one, batches four and eight, and two timing repetitions with 16 output tokens. Source snapshots, graph inventories, repeated outputs/RNG, and KV recovery passed. These are bounded functionality checks, not throughput, task-accuracy, or generalization evidence.

Instrumentation overhead. Twenty builder microbenchmark cases compare the same warmed runner with static-work bookkeeping enabled and disabled: five builders, batches three and eight, and temperatures zero and one. Each uses 20 alternating paired blocks of 25 calls with frozen checkpoint weights and synthetic inputs. Output tokens, proposal snapshots, and request RNG states agree; no build captures a graph. Added builder time ranges from 0.10% to 0.91% across case point estimates. This measures the incremental static-work observer, not the full scheduler/serving instrumentation or end-to-end overhead. A subsequent paired microbenchmark cached the fixed-depth control's constant scores and removed its redundant sorting, reducing builder time by 2.08–3.16% in six synthetic-input cases with identical outputs, proposals, and RNG; this is an execution correction, not an adaptive-policy gain.

I Reproducible Primary-Chain Leaf Construction

Algorithm 1 in the main text specifies one request; batching preserves its proposal order and request-local randomness. It implements the restricted topology in Section 3.4, without a new head fit. Token biases $w_m^\top W_1[v]$ are cached; hidden projections reuse the same backbone pass.

Admission and verification. Assume nonempty proposal support; when $|A_d| = 1$, Z_d is unused. The gate reads only the complete primary chain, backbone features, support, scores, and budgets, never second identities or hidden race information. For $T > 0$, fixed pre-round context, and ideal independent races, the second proposal is

$$\Pr(Z_d = z \mid X_{1:D}, d \in J) = \frac{Q_d(z)\mathbb{1}[z \neq X_d]}{\sum_{v \neq X_d} Q_d(v)}.$$

The verifier retains the original Q_d, normalizes it, and applies the ordered residual updates in Eq. 2; the score is never a proposal. GPU execution may prefetch both tokens, but cannot expose second

draws or correlated noise to any gate or later primary sampler. Static $c = 0.8$ orders supported depths increasingly. At $\theta = 0$, every supported depth within budget is eligible, including zero-score endpoints.

Storage and numerical scope. Pack all primary nodes first, then leaves in depth order. With zero-based primary indices, both depth-d siblings have parent index $d - 2$ (anchor -1). Clone FP32 proposal rows before a shared graph buffer is reused. The native sampler divides in FP64 registers, clamps exponential noise below the FP32 normal minimum, and breaks exact token-score ties by token ID. The ideal continuous-race law above does not assert exact finite-precision sampling. Confidence uses the same frozen weights in FP32, not bitwise BF16 evaluation; static bounds use FP32's representation of 0.8. The geometric score is a surrogate.

J Best-First Implementation Details

Algorithm 2 in the main text uses a pending slot for each eligible parent. Its key bounds the largest remaining score, not the identity of a prefetched sample. After drawing a sibling, remove its token from the proposal support and recompute the parent's key before re-enqueueing. A newly materialized child has its own conditional distribution and pending slot. Depth, sibling, support and remaining-output caps apply to both.

Store the original temperature-scaled proposal once per prepared parent; the verifier derives every ordered without-replacement proposal from that snapshot. The unscaled probabilities used for scoring are distinct from the proposal at $T \neq 1$. Neither calibration nor the threshold changes the verification residual. A node admitted before its draw remains in the tree even when the realized node score falls below the admission threshold.

K Full Validation Diagnostic Figures

Main-text Figure 2 compares greedy calibration with sampled-round prediction. The following figures retain the full allocation screening and greedy reliability diagnostics.

L Validation Work and Separate Phase Profiles

Figure 12 separates work over the complete validation split from a small synchronized timing diagnostic. Both use the final corrected source, Qwen3-4B, $T = 1$, initial batch eight, and 128 output tokens. The native fixed14 arm is the earlier parent-budgeted implementation; fused fixed14 is the optimized same-builder static control. Fused chain7 uses the latter builder with no optional leaves. These execution controls were evaluated on validation and do not replace frozen test comparisons.

On the full validation split, the surviving-edge best-first setting $\theta = 0.05$ increases τ by 16.75% over chain7 while reducing throughput by 8.59%; executed parent rows per emitted token increase by a factor of 1.963. Confidence $\theta = 0.05$ retains 28.87% fewer nodes than fused fixed14, while prepared and executed parent work change by $+0.22\%$ and $+0.27\%$, respectively. Its throughput changes by -0.44%. These descriptive comparisons do not establish equivalence between the static and adaptive policies.

In the separate eight-prompt profile, confidence retains 1,704 versus fixed14's 2,485 nodes, but target verification takes 220.47 versus 219.75 ms, across 31 versus 30 calls. This subset does not show a proportional target-time saving. It does not isolate graph padding, batch shape, or any individual kernel as its cause. These phase times describe only the eight-prompt profile, not the complete validation split.

On this validation split, fused fixed14, confidence $\theta = 0.05$ and chain7 produce 1,763.3, 1,755.6 and 1,607.1 tokens/s, respectively. These descriptive rates are not the frozen test estimates.

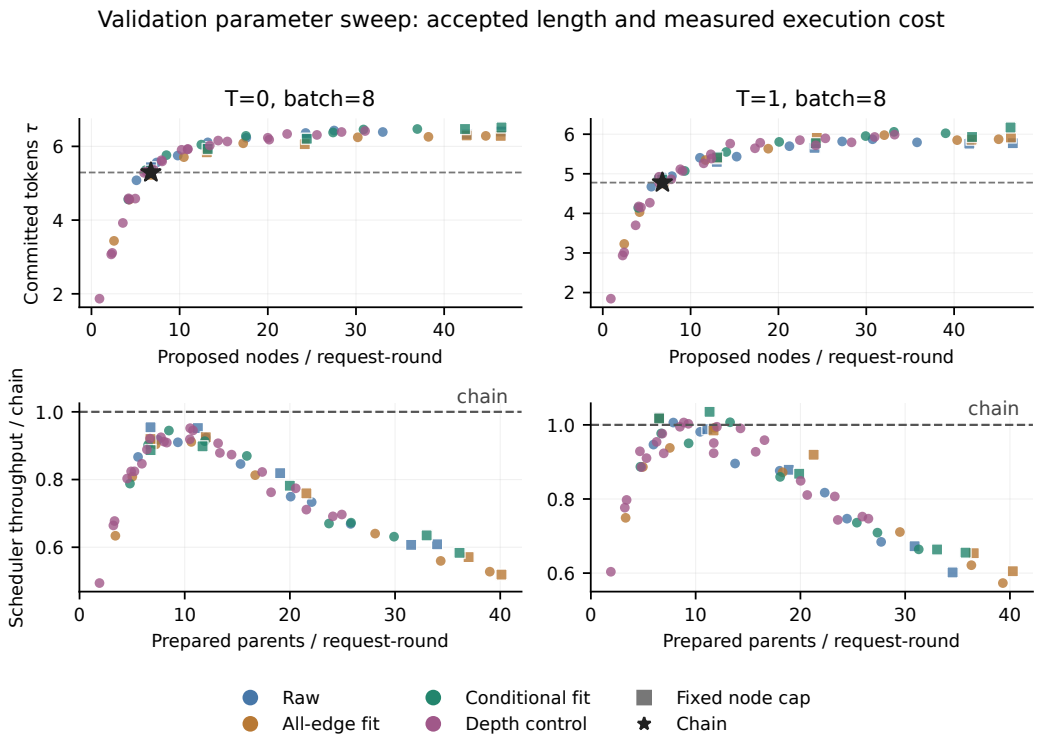

Figure 10: **Validation screening, not final test performance.** Qwen3-4B, batch 8, 24 prompts and 128 output tokens per prompt. All settings from the declared raw/all-edge/conditional/depth-score grid are retained. Fixed-cap markers here use best-first construction; separately evaluated static topologies are additional controls. Top: longer accepted paths show diminishing returns as node count grows. Bottom: extra prepared-parent work can erase those returns in measured scheduler throughput. Each point pools three complete input batches, one seed set and one timing repetition; the screening stage nominates configurations for a larger validation rerun.

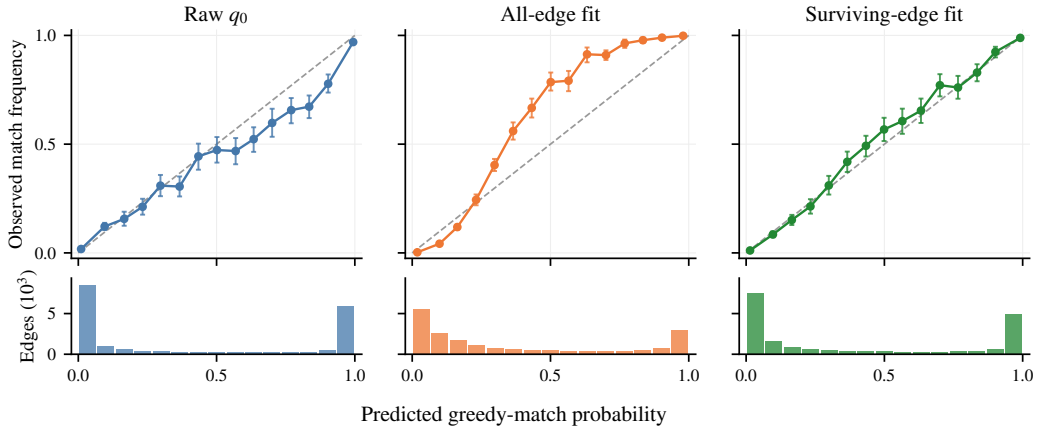

Figure 11: **Completed greedy calibration diagnostic.** Qwen3-4B, $T = 0$: the same 19,144 surviving-context edges from 72 validation prompts for all predictors. Top: reliability in 15 equal-width bins; bars show 95% prompt-cluster intervals. Bottom: bin counts. Models are fit only on 72 separate calibration prompts. These are target-argmax labels, not sampled-acceptance events or a throughput measurement.

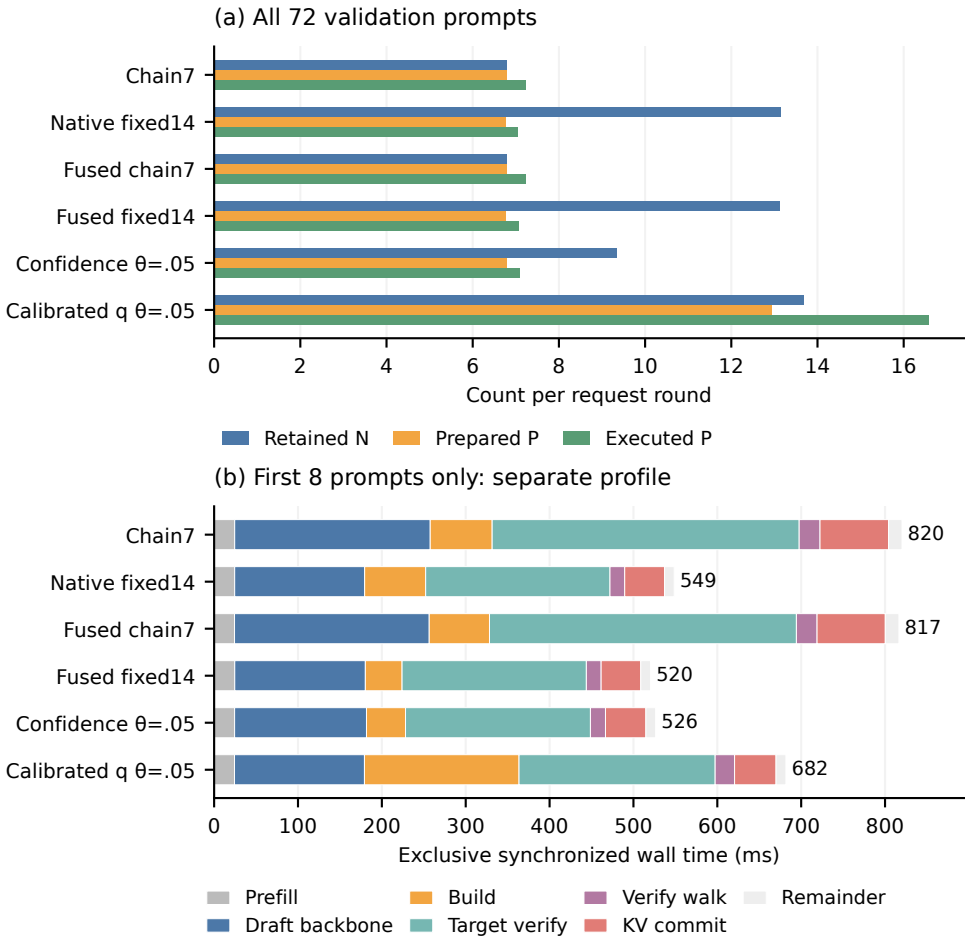

Figure 12: **Retained nodes and executed work differ.** (a) All 72 validation prompts, two seed sets and two timing repetitions: mean non-root nodes, prepared parent rows and executed parent rows per request-round. Executed rows include padding. (b) Separate synchronized profiles of only the first eight mixed-task prompts and one seed. The segments are exclusive measured phase times; the remainder is timed scheduler work outside these phases. Profiles reproduce their matched normal-run tokens, RNG states and work counters, but synchronization changes runtime, so profile times do not enter throughput estimates. Neither panel is a held-out result or a per-task wall-time attribution.