TRAINING-FREE DYNAMIC VERIFICATION BUDGET FOR PARALLEL SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Parallel speculative decoding reduces drafting latency through one-shot block generation, but its end-to-end efficiency increasingly depends on how target-model verification is budgeted. A fixed verification budget wastes computation on difficult prompts with limited acceptance potential while restricting acceleration on easy ones. We characterize this mismatch through the trade-off between affine verification costs and strict diminishing marginal acceptance gains, yielding a unique interior optimum that varies dynamically with prompt difficulty. To address this, we propose a **T**raining-free **D**ynamic **V**erification **B**udget controller (TDVB) built upon a dual channel architecture. Specifically, a feed-forward Shape module constructs a position-wise acceptance prior from draft confidence margins to capture local spatial uncertainty, while a feedback Scale module estimates global prompt difficulty through an exponential moving average of observed acceptance lengths across rounds. The controller integrates these signals to adapt verification budgets online, effectively converting extra candidate capacity into accepted depth without offline calibration or additional training, while strictly preserving the target model's output distribution. Extensive experiments show that TDVB achieves an average acceptance length of up to 10.18, improves end-to-end speedup by 14%–44% over strong static baselines, and reaches up to $7.40\times$ over autoregressive decoding. Performance profiling reveals that these efficiency gains stem primarily from eliminating redundant verification rounds rather than merely reducing per-round execution overheads. Furthermore, the performance advantages scale robustly with model size, demonstrating that adaptive verification budgeting is essential to translating faster draft generation into sustained end-to-end acceleration and unlocking the true scaling potential of parallel speculative decoding.

1 INTRODUCTION

Autoregressive decoding performs a full forward pass of the target model before generating each token (Vaswani et al., 2017). Speculative decoding mitigates this latency bottleneck by letting a lightweight draft model propose several future tokens, which are then verified in parallel by the target model (Leviathan et al., 2023; Chen et al., 2023; Stern et al., 2018). While traditional speculative approaches rely on sequential draft models such as Medusa (Cai et al., 2024), EAGLE (Li et al., 2024a;b), and Sequoia (Chen et al., 2024), recent advancements in parallel candidate generation, as exemplified by DFlash (Chen et al., 2026), fundamentally transform the cost structure of speculative decoding. An autoregressive draft model pays a linear time cost for every additional candidate, whereas a parallel draft model produces an entire block of candidates in a single forward pass. In such parallel architectures, drafting accounts for less than 20% of the total round latency. The focus of optimization has therefore moved from reducing the drafting cost to allocating the budget in the verification stage, so that end-to-end speedup is maximized. As demonstrated in Figure 1, verification occupies most of each round, and the largest saving comes from reducing the number of target-model interactions rather than making one interaction substantially cheaper. Appendix D gives the full latency decomposition.

However, existing parallel speculative decoding methods rely on a static verification budget, which cannot adapt to the difficulty that varies across generation rounds. On complex reasoning tasks, a fixed and small budget often fails to cover the effective candidate path that the target model eventually accepts. A straightforward remedy is to enlarge the budget upper bound, but this does not

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.

necessarily lead to a better speedup. A larger candidate set does increase the probability of covering the effective output, and hence the absolute acceptance length, yet this marginal gain is strictly concave and decays quickly. At the same time, the verification cost of the target model grows affinely with the budget. Because of this concave gain and affine cost, the end-to-end speedup yields a unique interior optimum with respect to the verification budget. More importantly, the location of this optimum depends strongly on prompt difficulty. As prompt difficulty varies, the budget that achieves the best speedup differs by a factor of several. Consequently, a static budget can be optimal for at most one difficulty level, introducing redundant verification overhead on difficult prompts while prematurely truncating and capping throughput on easy ones. A few recent methods do adjust the number of verified tokens on the fly, but they rely on learned predictors or confidence-scheduled controllers (Shukla, 2026; Cheng et al., 2026) that need offline calibration and become fragile and costly to retune when the target model, the hardware backend, or the data distribution changes.

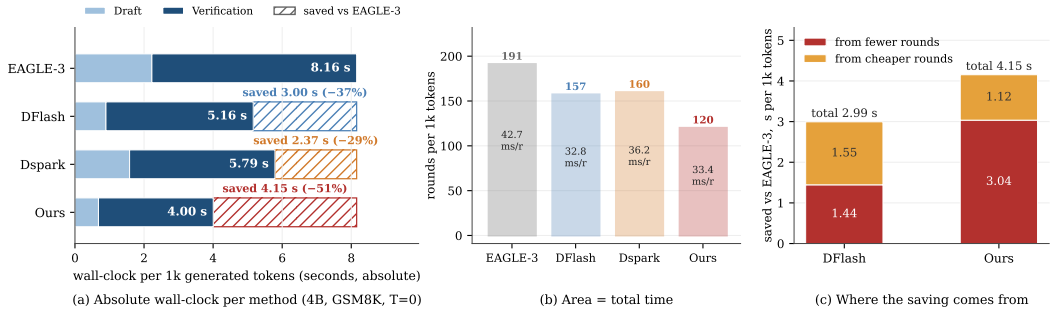

Figure 1: **Mechanism level latency decomposition** (Qwen3-4B on GSM8K at $T = 0$). The profile separates drafting and verification latencies, decomposing the savings over EAGLE-3 into round reduction and per-round cost. The primary efficiency gain stems from eliminating verification rounds rather than reducing per-round execution costs.

To address this issue, we propose TDVB, a training-free dynamic verification budget controller with a feed-forward and feedback dual channel architecture. Appendix A shows the reallocation pattern. The controller adapts to prompt difficulty and reallocates verification computation across decoding rounds. Specifically, the feed-forward channel uses the confidence margins that are native to the draft model to capture the local uncertainty at each lookahead position, and to provide a per-position prior over candidate quotas. The feedback channel aggregates an exponential moving average (EMA) of the target model's historical acceptance length to track a global difficulty level across rounds. The two channels are combined through a quota equation, in which the local shape prior produced by the feed-forward channel is rescaled as a whole by the global difficulty level produced by the feedback channel. Candidate admission and truncation are then carried out under a global hard budget constraint. This mechanism forms a dynamic closed loop that consists of parallel drafting, feed-forward shaping and feedback scaling, quota synthesis and candidate admission, target-model verification, and state update. The controller adds negligible computational overhead and strictly preserves the output distribution of the target model.

In summary, we make the following contributions:

- **Training-free dynamic verification budget.** We introduce TDVB, whose controller forms a real-time closed loop that dynamically synthesizes per-position candidate quotas for each round. The controller adapts to prompt difficulty without offline calibration or additional training, effectively converting extra candidate capacity into accepted depth while strictly preserving the target model's output distribution.
- **A dual channel control architecture.** We decouple the verification budget of speculative decoding into a local spatial allocation and a global scale adjustment. The feed-forward channel builds a spatial prior from the draft's native confidence margin and characterizes the local uncertainty at each lookahead position. The feedback channel tracks global difficulty across rounds through an EMA of the acceptance length measured on the target model.

- **Improved end-to-end acceleration performance.** Experiments show that TDVB consistently improves both the average acceptance length and the end-to-end wall-clock speedup over state-of-the-art parallel speculative decoding baselines. Performance profiling reveals that these efficiency gains stem primarily from eliminating redundant verification rounds rather than merely reducing per-round execution overheads. Furthermore, the performance advantages scale robustly with model size, unlocking the true scaling potential of parallel speculative decoding.

2 Related Work

Draft Generation Architectures. Leviathan et al. (2023) and Chen et al. (2023) introduce the original lossless formulations that verify a draft prefix in parallel and establish the distribution-preserving acceptance rule. Subsequent systems improve the draft side with multiple heads or feature-level predictors, as proposed by Cai et al. (2024) and Li et al. (2024a;b), while Li et al. (2026) study scaling the draft model alongside training-time and test-time trade-offs. Although these methods substantially improve candidate quality, they leave the target-model verification budget statically fixed for a given deployment configuration.

Structural Layouts and Parallelism. To efficiently aggregate and process candidate tokens, Miao et al. (2024) and Chen et al. (2024) construct verification trees and hardware-aware structural layouts. Furthermore, parallel draft generators explored by Chen et al. (2026) significantly reduce the generation cost of producing a massive candidate block. While these approaches make large candidate pools computationally practical, their primary control variables remain the candidate generator architecture or the static tree layout. Consequently, they still typically submit a fixed-size candidate topology to the target model, leaving the verification allocation completely rigid.

Adaptive Verification Budgets. To break the rigid constraints of static allocation, a few recent methods explicitly adjust the number of verified tokens on-the-fly based on prompt difficulty or system states. For instance, Shukla (2026) train a lightweight predictor to modulate the speculation length based on KV-cache compression states. Similarly, Cheng et al. (2026) implement a confidence-scheduled controller that dynamically truncates candidate blocks during generation. However, both approaches rely heavily on parameterized components such as auxiliary confidence heads, and they require rigorous offline calibration. These learned mechanisms inevitably become fragile and costly to tune when migrating across different target models, hardware backends, or data distributions. In stark contrast, TDVB achieves dynamic verification without any learned components or offline calibration. By directly tracking runtime acceptance dynamics, it adapts budgets on-the-fly, avoiding the fragility of parameterized controllers and ensuring robust acceleration across diverse workloads.

3 PRELIMINARIES

3.1 The Cost Structure of Parallel Verification

In parallel speculative decoding, at decoding round t the target model verifies a batch $\mathcal{B}_t$ that contains B_t candidates and returns the accepted prefix length τ_t. The expected end-to-end speedup of the system is proportional to the expected acceptance length per unit time, which we write as $S(B_t)$,

$$S(B_t) = \frac{\mathbb{E}[\tau_t]}{C_{\text{draft}} + C_{\text{verify}}(B_t)}. \tag{1}$$

The affine nature of verification cost. Drawing on the memory bound nature of small batch decoding highlighted by the Roofline model (Williams et al., 2009) and memory efficient attention designs such as FlashAttention (Dao et al., 2022), we empirically observe an affine trend in the verification cost. Specifically, we model $C_{\text{verify}}(B_t)$ as being approximately affine with respect to the number of candidates B_t:

$$C_{\text{verify}}(B_t) \approx \kappa B_t + \beta. \tag{2}$$

In this formulation, the fixed overhead β captures the dominant time spent on kernel launches and moving large model weights from HBM to SRAM, while κB_t scales with the FLOPs performed by

the attention and MLP layers. As demonstrated in Appendix C, measurements confirm this empirical perspective, showing that the fixed overhead β accounts for 93% of the verification time at $B_t = 32$ and drops to 62% at $B_t = 256$. This behavior indicates that moderate budget expansion can amortize fixed memory-access overheads.

Concave gain and interior optimum. Although a larger budget amortizes the fixed cost, the verification gain in the numerator, $\mathbb{E}[\tau_t]$, is constrained by the long-tailed nature of the vocabulary distribution and by the saturation of probability-mass coverage. As B_t grows, the probability that the candidate set covers the target output increases, but this process is essentially an exploration of a long-tailed distribution, where the first few high-probability candidates such as Top-1 and Top-2 capture most of the probability mass while the target probability of subsequent candidates decays rapidly. This nonlinearity, which follows from the output distribution of the language model, makes the marginal gain strictly concave and quickly decaying. Appendix C shows that, across $B_t \in [8, 192]$, the marginal acceptance length per additional candidate decreases by a factor of 14.

The combination of a concave gain and an affine verification cost forms a fractional programming problem (Schaible & Ibaraki, 1983). The ratio $S(B_t)$ in Eq. equation 1 therefore has an interior optimum B^*; beyond it, the cost κ of an additional candidate exceeds its probability gain and end-to-end speedup declines. Appendix C provides the corresponding measurements.

3.2 Difficulty Stationarity

The location of the optimal verification budget B^* is neither global nor static; it depends strongly on the underlying per-prompt latent difficulty θ_p. The time-series analysis reported in Appendix C reveals two key statistical properties of this difficulty signal.

Inter-prompt variation. The average acceptance length differs substantially across prompts, for example between highly templated math problems and open-ended conversations, by a factor of up to 6.5. Appendix C reports the full prompt level comparison.

Intra-prompt stationarity. Within a single generation, τ_t fluctuates at a high frequency, but its autocorrelation decays quickly after lag-1 and then falls into the noise band. The autocorrelation analysis in Appendix C quantifies this short memory.

A variance decomposition in Appendix C quantifies this structure: between-prompt variation accounts for 26.6% of the total variance, whereas 73.4% comes from round-level fluctuation within a prompt. This indicates that the long-range memory across rounds is short, and that θ_p is essentially a stationary level contaminated by high-frequency noise, rather than a non-stationary Markov state that evolves over time. Note that θ_p is a latent quantity; it is not observed directly and is estimated online from the acceptance feedback, as described in Section 4.1.

These statistical properties determine the design principle of the controller. Because the optimal budget fluctuates smoothly with prompt difficulty, any static budget suffers from a mismatch between the allocated and the required computation. What the controller needs to perform is not complex trend tracking, but online estimation of a stationary level under noise.

4 Training-Free Dynamic Verification Budget

As discussed above, static verification budgets fail to adapt to the fluctuations of prompt difficulty during generation. Furthermore, calibrating raw draft features at an absolute level is impeded by the extreme sparsity of binary accept or reject feedback per round, which introduces large estimation variance. While existing adaptive approaches attempt to address these challenges, they often trade system efficiency for adaptivity or introduce heavy training complexity.

To overcome these limitations, we propose TDVB, a training-free dynamic verification budget controller. Figure 2 illustrates its execution. The controller partitions computation into independent spatial and scale dimensions, filters high-frequency noise online without offline calibration or additional training, tracks the stationary prompt-level difficulty θ_p, and strictly preserves the output distribution of the target model.

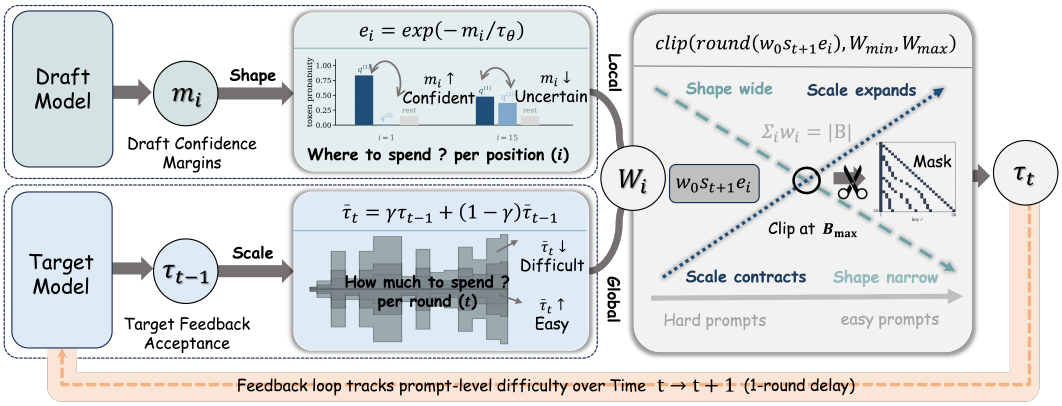

Figure 2: **Overview of TDVB.** The feed-forward ***Shape*** module maps draft confidence margins to per-position weights, and the feedback ***Scale*** module maps the EMA of acceptance lengths to one global multiplier. The two channels respond to prompt difficulty in opposite directions, so their bounded product sets the candidate budget and the resulting quotas assemble a single pool for lossless verification.

Algorithm 1 in Appendix F summarizes one decoding round of the controller. The draft model first produces one forward pass D over K lookahead positions, where the horizon K is the number of positions that the draft block can predict in parallel and therefore equals the maximum number of tokens that a single verification round can accept. The feed-forward channel then turns the confidence margin at each position into a position-wise weight, the feedback channel turns the EMA state into a global scale, and the two are fused into a per-position quota. The target model performs one parallel verification pass, and the EMA state is updated from the observed acceptance length before the KV-cache is advanced.

4.1 Feed-Forward Shape

The candidates produced by the parallel drafting stage are not homogeneous. The draft confidence already carries information about the local prediction uncertainty at each position. During parallel drafting, the draft model performs one forward pass over the current prefix and returns the marginal distributions of K lookahead positions. To distribute relative shares of computation across positions, we need a local uncertainty measure.

Let $q_{t,i}^{(1)}$ and $q_{t,i}^{(2)}$ denote the probabilities of the highest and the second highest candidate token at lookahead position $i \in [1, K]$ in round t. We define the *confidence margin* at this position as

$$m_{t,i} = \log q_{t,i}^{(1)} - \log q_{t,i}^{(2)}. \tag{3}$$

This margin directly reflects how certain the draft model is about the local prediction: the smaller $m_{t,i}$, the closer the top two candidates are in probability, and the less certain the model is.

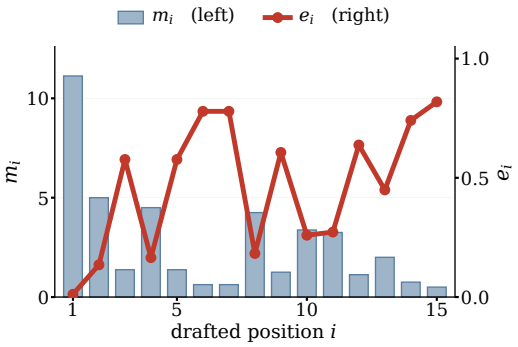

Figure 3: **Feed-forward Shape prior from draft confidence margins.** Blue bars show the margin m_i at draft position i on the left axis, and the red curve shows the induced weight e_i on the right axis. Lower margins indicate greater local uncertainty and receive larger weights without target-model feedback.

The Shape module maps this margin to an position-wise weight $e_{t,i}$ using a fixed parameter free exponential prior:

$$e_{t,i} = \exp\left(-\frac{m_{t,i}}{\tau_\theta}\right), \tag{4}$$

where $\tau_\theta > 0$ is a global smoothing temperature that is kept constant and controls the sharpness of the weighting. As shown in Figure 3, this rule automatically assigns a larger weight to positions where the draft model is highly uncertain, which means small $m_{t,i}$. The weights determine relative emphasis, while the base width and Scale multiplier determine the absolute quota. The Shape channel consumes no feedback data from the target model and therefore introduces no estimation variance.

4.2 Feedback Scale

Because the latent difficulty θ_p is a stationary level wrapped in high-frequency noise, we aggregate the entire feedback history into a single scalar $\bar{\tau}_t$ that requires only $O(1)$ memory.

Let τ_t denote the number of draft tokens that the target model actually accepts in round t. We formalize the objective within the framework of *online convex optimization* (OCO) (Zinkevich, 2003), where at each step the controller predicts a scale surrogate, observes the actual feedback τ_t, and incurs a quadratic tracking loss. Online gradient descent (OGD) on this loss with a constant step size $\gamma \in (0, 1]$ yields exactly the stationary-state update rule.

$$\bar{\tau}_{t+1} = (1 - \gamma)\,\bar{\tau}_t + \gamma\,\tau_t, \tag{5}$$

where $\bar{\tau}_t$ is the smoothed aggregate difficulty state, with $\bar{\tau}_1 \equiv \tau_1$ and γ is a constant step size that controls the trade-off between the weight of the history and the responsiveness of the estimate.

This update rule is mathematically equivalent to an exponential moving average (EMA). Classical results in statistical signal processing show that, when observations are noisy at high frequency while the latent state has no significant low-frequency drift, OGD with a constant step size is a suitable linear filter for estimating a locally stationary level (Brown, 1959). Although the observed τ_t oscillates from round to round, the aggregate state $\bar{\tau}_t$ removes this high frequency noise and tracks prompt level capacity; the corresponding trace is reported in Appendix C.

The stationary state is then converted into a bounded global scale multiplier s_{t+1}:

$$s_{t+1} = \mathrm{clip}\left(\frac{\bar{\tau}_t}{\tau_{\mathrm{ref}}},\, s_{\min},\, s_{\max}\right), \tag{6}$$

where τ_{ref} is a preset reference point that sets the default gain center of the system, and $s_{\min}$ and $s_{\max}$ are lower and upper safety bounds that prevent computation starvation or overload. Because a low acceptance length indicates a difficult prompt, this ratio contracts s_t on hard prompts, where additional candidates rarely survive verification, and expands it on easy prompts, where the draft is already accurate and extra capacity converts directly into accepted depth.

4.3 Budget Synthesis and Bounded Execution

After spatial shaping and global scaling, the local Shape weight $e_{t,i}$ and the global Scale state s_{t+1} are multiplied under hard bounds to produce the candidate quota $W_{t,i}$ for each position:

$$W_{t,i} = \mathrm{clip}(\mathrm{round}(w_0 \cdot s_{t+1} \cdot e_{t,i}),\, W_{\min},\, W_{\max}), \tag{7}$$

where w_0 is a base allocation width, and $W_{\min}$ and $W_{\max}$ are hard lower and upper bounds on the per-position allocation. These two bounds restrict the quota of any single position, so that no position can be starved or dominate the batch.

After per-position clipping, a deterministic global-cap operator enforces $\sum_i W_{t,i} \le B_{\max}$ before candidates are gathered. This explicit bounded controller avoids computation in the tail where marginal verification returns have already decayed. From the resulting quota vector $\mathbf{W}_t = [W_{t,1}, \ldots, W_{t,K}]$, the candidate pool is assembled and flattened before the target sampling step, and verified in a single forward pass under the ancestral visibility mask. The mechanism reallocates verification computation internally with an arithmetic overhead of only $O(K)$, and it leaves the target-model logits, acceptance test, and output distribution unchanged. Appendix H states the distribution-preservation and memory invariants explicitly.

5 EXPERIMENTS

To evaluate the dynamic verification budget comprehensively, we conduct experiments on target models of two scales, Qwen3-4B and Qwen3-8B (Yang et al., 2025). The evaluation covers three reasoning domains that stress the context-handling ability of a model: (1) **mathematical reasoning**, including GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2024), and AIME24 (Mathematical Association of America, 2024); (2) **code generation**, including HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and SWE-bench Lite (Jimenez et al., 2024); and (3) **open multi-turn conversation**, represented by MT-Bench (Zheng et al., 2023).

5.1 EXPERIMENTAL SETUP

Baselines and metrics. We compare TDVB with standard autoregressive decoding (AR) and with state-of-the-art speculative decoding methods, including EAGLE-3 (Li et al., 2026), DFlash (Chen et al., 2026), and DSpark (Cheng et al., 2026). We test under both greedy decoding ($T = 0$) and nucleus sampling ($T = 1$). The main metrics are the **average acceptance length** $\bar{\tau}$ and the **end-to-end wall-clock speedup** relative to autoregressive decoding. Appendix I describes their computation, and Appendix D gives a system-level latency decomposition explaining the round-count reduction.

Table 1: **End-to-end speedup and average acceptance length ($\bar{\tau}$).** Entries are formatted as speedup $\times$ ($\bar{\tau}$). Evaluations span Qwen3-4B/8B models, temperatures $T \in \{0, 1\}$, and block sizes (8 and 16). Bold indicates the best result. Note that DSpark results for block-16 are unavailable due to lack of official support.

	Code			Math			Chat	All
Method	HumanEval	MBPP	SWE-Lite	GSM8K	MATH-500	AIME24	MT-Bench	Mean
Qwen3-4B, $T = 0$								
EAGLE-3	2.62× (4.53)	2.27× (4.11)	1.72× (3.05)	3.01× (5.39)	2.89× (5.25)	2.76× (4.92)	1.43× (2.64)	2.39× (4.27)
DFlash (block-8)	3.67× (5.19)	3.44× (4.90)	2.52× (3.57)	4.15× (5.84)	4.19× (5.91)	3.77× (5.44)	2.33× (3.40)	3.44× (4.89)
DSpark (block-8)	3.60× (5.63)	3.52× (5.37)	2.69× (3.95)	4.16× (6.26)	4.15× (6.31)	3.75× (5.87)	2.36× (3.75)	3.46× (5.31)
TDVB (block-8)	**3.99× (6.30)**	**4.02× (6.07)**	**3.21× (4.93)**	**4.67× (6.84)**	**4.40× (6.94)**	**4.36× (6.69)**	**2.88× (4.68)**	**3.93× (6.06)**
DFlash (block-16)	4.75× (6.64)	4.36× (6.17)	2.57× (3.59)	4.62× (6.55)	5.50× (7.72)	5.46× (7.68)	3.11× (4.39)	4.34× (6.11)
TDVB (block-16)	**6.47× (9.06)**	**5.97× (8.55)**	**3.71× (5.32)**	**6.58× (8.73)**	**7.40× (10.06)**	**6.81× (10.00)**	**4.25× (6.22)**	**5.88× (8.28)**
Qwen3-4B, $T = 1$								
EAGLE-3	2.28× (4.13)	2.01× (3.70)	1.49× (2.70)	2.87× (5.12)	2.53× (4.65)	2.14× (3.92)	1.33× (2.47)	2.09× (3.81)
DFlash (block-8)	3.38× (4.77)	3.03× (4.40)	2.17× (3.05)	3.80× (5.41)	3.49× (4.98)	2.85× (4.04)	2.13× (3.08)	2.98× (4.25)
DSpark (block-8)	3.26× (5.40)	3.28× (5.13)	2.45× (3.73)	3.93× (6.01)	3.67× (5.79)	3.01× (4.90)	2.26× (3.58)	3.12× (4.93)
TDVB (block-8)	**3.83× (5.84)**	**3.55× (5.71)**	**2.77× (4.19)**	**4.03× (6.28)**	**4.02× (6.07)**	**3.44× (5.39)**	**2.56× (4.27)**	**3.46× (5.39)**
DFlash (block-16)	4.07× (5.77)	3.70× (5.28)	2.08× (2.95)	3.96× (5.65)	4.58× (6.51)	3.59× (5.08)	2.65× (3.78)	3.52× (5.00)
TDVB (block-16)	**5.91× (8.43)**	**5.42× (7.99)**	**3.24× (4.76)**	**5.83× (8.28)**	**6.17× (9.00)**	**5.10× (7.63)**	**3.87× (5.81)**	**5.08× (7.41)**
Qwen3-8B, $T = 0$								
EAGLE-3	2.47× (4.72)	2.24× (4.25)	1.70× (3.16)	3.00× (5.59)	2.85× (5.40)	2.68× (4.95)	1.56× (2.92)	2.36× (4.43)
DFlash (block-8)	3.65× (5.27)	3.34× (4.82)	2.54× (3.57)	4.22× (5.99)	4.16× (5.96)	3.93× (5.46)	2.38× (3.43)	3.46× (4.93)
DSpark (block-8)	3.62× (5.84)	3.47× (5.41)	2.59× (3.98)	4.10× (6.45)	4.20× (6.44)	4.05× (5.89)	2.36× (3.84)	3.48× (5.41)
TDVB (block-8)	**4.15× (6.39)**	**3.78× (5.95)**	**3.02× (4.74)**	**4.44× (6.83)**	**4.45× (6.90)**	**4.36× (6.65)**	**2.88× (4.50)**	**3.87× (5.99)**
DFlash (block-16)	4.58× (6.48)	4.17× (5.92)	2.53× (3.59)	4.61× (6.54)	5.56× (7.87)	5.16× (7.35)	2.99× (4.30)	4.23× (6.01)
TDVB (block-16)	**5.46× (8.96)**	**5.32× (8.35)**	**3.14× (5.34)**	**5.62× (8.85)**	**6.32× (10.18)**	**5.90× (9.76)**	**3.70× (6.08)**	**5.07× (8.22)**
Qwen3-8B, $T = 1$								
EAGLE-3	2.19× (4.31)	2.08× (3.95)	1.44× (2.78)	2.75× (5.27)	2.41× (4.74)	2.15× (4.06)	1.37× (2.65)	2.06× (3.97)
DFlash (block-8)	3.25× (4.69)	3.06× (4.33)	2.07× (2.96)	3.79× (5.39)	3.42× (4.92)	2.94× (4.16)	2.15× (3.07)	2.95× (4.22)
DSpark (block-8)	3.31× (5.45)	3.34× (5.25)	2.35× (3.62)	3.80× (6.10)	3.58× (5.79)	3.20× (5.06)	2.18× (3.62)	3.11× (4.98)
TDVB (block-8)	**3.51× (5.58)**	**3.42× (5.36)**	**2.40× (3.77)**	**3.90× (6.18)**	**3.69× (5.96)**	**3.24× (5.30)**	**2.48× (4.02)**	**3.23× (5.17)**
DFlash (block-16)	3.72× (5.33)	3.43× (4.94)	2.00× (2.87)	3.80× (5.46)	4.34× (6.24)	3.56× (5.13)	2.47× (3.59)	3.33× (4.79)
TDVB (block-16)	**4.82× (8.03)**	**4.72× (7.72)**	**2.49× (4.40)**	**5.13× (8.38)**	**5.45× (8.92)**	**4.26× (7.36)**	**3.29× (5.58)**	**4.31× (7.20)**

Verification cost and budget ceiling. As analyzed in Section 3.1, submitting too many candidates to the verifier suffers from strongly diminishing marginal returns, with the marginal acceptance gain decreases by a factor of 23.7 over the range we scan. Meanwhile, the fixed overhead β in the affine verification cost $C_{\text{verify}}(B) = \kappa B + \beta$ substantial, accounting for 90% of the cost at $B = 32$ and

61% at $B = 256$. Enlarging the budget blindly does not produce a linear performance gain, and it can reduce the speedup. On GSM8K ($T = 0$), for example, increasing a static budget from $B = 128$ to $B = 256$ yields only a $+1.1\%$ increase in acceptance length, at the cost of an 18% loss in end-to-end speedup. To verify that dynamic budget planning remains better than a saturated static budget under a restricted budget, we therefore set a strict physical budget ceiling of $B_{\max} = 128$ in all main experiments. All experiments are run on NVIDIA A100 GPUs, with a maximum generation length of 2048 tokens.Complete hyperparameter configurations and mask construction details are provided in Appendix B and Appendix J.

5.2 Main Results

As summarized in Table 1, TDVB consistently outperforms EAGLE-3, DFlash, and DSpark across the full evaluation grid, validating the effectiveness of the proposed dual-channel dynamic budget controller. Across all evaluated domains, our approach improves both the average acceptance length $\bar{\tau}$ and the end-to-end wall-clock speedup compared to existing parallel speculative decoding frameworks. On Qwen3-4B at $T = 0$, TDVB achieves an average speedup of $3.93\times$ under block-8 and $5.88\times$ under block-16 across the full task grid, with peak individual speedups reaching up to $7.40\times$ on MATH-500.

The performance margins are particularly pronounced on structurally complex and long-tail reasoning benchmarks such as AIME24, MATH-500, and SWE-bench Lite. While static baselines suffer from premature truncation due to fixed budgets, our feedback-driven controller scales the verification budget dynamically based on prompt difficulty, unlocking substantial throughput gains precisely where traditional methods fall short. Furthermore, these advantages remain remarkably stable under deterministic greedy decoding ($T = 0$), stochastic nucleus sampling ($T = 1$), and when scaling the target model to Qwen3-8B.

5.3 Ablation Study

We compare four variants on Qwen3-4B. Static is the uniform baseline with neither Shape nor Scale; w/o Scale keeps Shape but fixes the global level; w/o Shape keeps Scale but allocates uniformly. Acceptance Calibration (AC) replaces $e_i = \exp(-m_i/\tau_\theta)$ with a rolling empirical $P(\text{accept} \mid m)$ curve.

Table 2: **Ablation and comparison with empirical calibration** (Qwen3-4B, full evaluation set). Entries are formatted as "speedup $\times$ (average acceptance length $\bar{\tau}$)".

	Code			Math			Chat
Method	HumanEval	MBPP	SWE-Lite	GSM8K	MATH-500	AIME24	MT-Bench
Qwen3-4B, $T = 0$							
Static	5.94× (8.60)	5.55× (8.09)	3.35× (4.91)	5.81× (8.34)	6.67× (9.72)	6.63× (9.76)	4.05× (6.00)
w/o Scale	5.89× (8.51)	5.49× (8.03)	3.42× (5.05)	5.75× (8.38)	6.70× (9.78)	6.59× (9.50)	3.88× (5.76)
w/o Shape	5.74× (8.31)	5.42× (7.93)	3.12× (4.55)	5.65× (8.20)	6.61× (9.65)	6.39× (9.33)	3.77× (5.47)
AC	5.57× (8.11)	5.21× (7.61)	3.06× (4.65)	5.53× (8.10)	6.38× (9.41)	5.92× (9.08)	3.66× (5.48)
Qwen3-4B, $T = 1$							
Static	5.38× (8.03)	5.06× (7.60)	2.86× (4.25)	5.27× (7.87)	5.67× (8.55)	4.68× (7.04)	3.70× (5.61)
w/o Scale	5.25× (7.87)	4.92× (7.34)	2.94× (4.44)	5.18× (7.70)	5.80× (8.68)	4.87× (7.33)	3.59× (5.43)
w/o Shape	5.20× (7.76)	4.97× (7.46)	2.62× (3.87)	5.12× (7.63)	5.63× (8.46)	4.56× (6.81)	3.37× (5.01)
AC	4.92× (7.43)	4.72× (7.09)	2.55× (3.96)	5.01× (7.53)	5.43× (8.23)	4.26× (6.80)	3.28× (5.04)

As summarized in Table 2, isolating individual channels reveals distinct performance dynamics. The shape-only variant (*w/o Scale*) generally outperforms the scale-only variant (*w/o Shape*), indicating that position-wise uncertainty modeling is critical for parallel verification. Static sometimes exceeds one single-channel ablation and sometimes falls below it across tasks, which shows that a fixed uniform budget can be competitive at particular difficulty levels but is not consistently reliable. The complete Shape-Scale controller (*TDVB* in Table 1) combines both sources of information and gives the best overall operating point.

Furthermore, empirical calibration (*AC*) consistently underperforms the fixed exponential prior. Because the empirical curve is updated solely from accepted positions in previous rounds (e.g., typically fewer than $K = 15$ valid observations per round), the estimates become severely data-starved. Splitting the margin range into multiple bins often leaves fewer than one observation per bin on average, causing the fine-grained estimation to degrade rapidly. This performance drop highlights why our analytical exponential prior is significantly more reliable under sparse, noisy online feedback. Appendix K details the full binning sweep and sample-count calculations.

5.4 Effective Search Depth and Scaling Robustness

Adaptive Verification Depth. TDVB changes where computation is spent, not which tokens are accepted. *Effective search depth* is the probability that a round reaches a given draft position. Figure 4 compares survival curves across prompt difficulties. The controller reaches deeper positions in both regimes, raising effective depth by 48% on difficult prompts and by 26% on easy ones. These margins match the improvement in per-round acceptance length, so the gain comes from fuller rounds rather than from verifying more candidates. The two channels act on different quantities. The Scale multiplier rises on easy prompts and contracts on difficult ones, while the Shape prior concentrates the quota on the uncertain positions of a difficult round, and because Shape dominates in our measurements a difficult round still requests more total candidates before the ceiling clips it. Since the pool is assembled before target verification, this placement preserves the target distribution.

Cross-Scale Robustness. Rather than hitting the artificial throughput ceilings of fixed-budget baselines, TDVB systematically converts expanded candidate capacity into accepted depth. This structural advantage compounds with scale. As draft block sizes and target models increase from Qwen3-4B to 8B, our performance gap over the strongest static baseline widens progressively from 14% to over 35%. Appendix E details the complete cross-scale measurements, including a $2.7\times$ throughput amplification under wider parallel configurations.

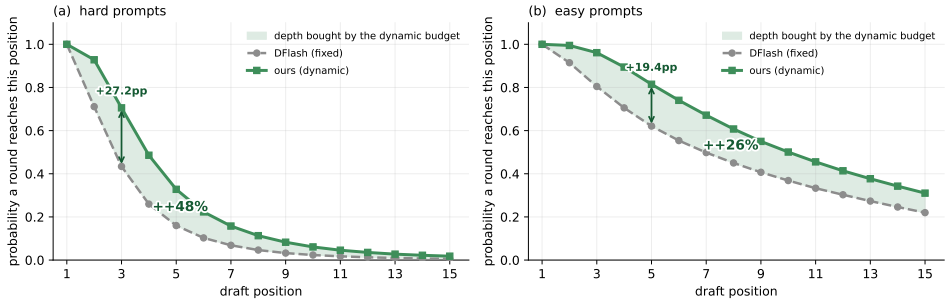

Figure 4: **Position-wise survival curves of verification depth. (a)** On difficult prompts, the Shape prior widens the per-position quotas and TDVB raises the effective depth by 48%. **(b)** On easy prompts, TDVB also remains deeper, by 26%; in both regimes the gain comes from fuller rounds rather than from a larger candidate set.

6 Conclusion

We introduce a training-free dynamic verification budget controller for parallel speculative decoding. By factorizing compute allocation into a local draft-margin Shape prior and an online-optimized global Scale state, our dual-channel architecture dynamically synthesizes per-position quotas in a real-time closed loop. The framework requires no training, offline calibration, or modifications to the target model, thereby strictly preserving its output distribution. Through extensive experiments across diverse benchmarks, model scales, and sampling temperatures, we demonstrate that TDVB fundamentally alters the system-level mechanism of acceleration. The controller systematically converts expanded search capacities into fuller accepted rounds. This eliminates redundant verification interactions and unlocks the true scaling potential of parallel speculative decoding.

AI USE STATEMENT

Generative AI tools were used for language editing, terminology consistency checks, and LaTeX layout review. They were not used to generate experimental data, alter measurements, or determine the paper's scientific claims. The authors reviewed all AI-assisted material and take responsibility for the final text, citations, figures, code, and results.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Robert Goodell Brown. *Statistical Forecasting for Inventory Control*. McGraw-Hill, 1959.

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. *arXiv preprint arXiv: 2401.10774*, 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. *arXiv preprint arXiv:2602.06036*, 2026.

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

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*, 2024.

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, et al. Dspark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, pp. 19274–19286, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *Advances in Neural Information Processing Systems*, 38:136737–136756, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Mathematical Association of America. American invitational mathematics examination (AIME) 2024. American Invitational Mathematics Examination (AIME) 2024, 2024. Accessed February 2024. Available at `https://maa.org/math-competitions/american-invitational-mathematics-examination-aime`.

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

Siegfried Schaible and Toshidide Ibaraki. Fractional programming. volume 12, pp. 325–338. Elsevier, 1983.

Shikhar Shukla. Speckv: Adaptive speculative decoding with compression-aware gamma selection. *arXiv preprint arXiv:2605.02888*, 2026.

Mitchell Stern, Noam Shazeer, Jakob Uszkoreit, and Mohammad Norouzi. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. volume 30, 2017.

Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. volume 36, pp. 46595–46623, 2023.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.

A Dynamic Budget Reallocation Trace

This trace gives a round-level view of the controller. At the cold start the Shape channel provides the only available signal. Once target-model feedback arrives the two channels act on different quantities and can point in opposite directions. The feedback Scale channel sets the single global multiplier s of Eq. equation 6; because it follows the smoothed acceptance length, it contracts s on difficult prompts and raises it on easy ones. The feed-forward Shape channel sets the per-position weights e_i of Eq. equation 4, which are largest exactly at the uncertain positions of a difficult round. The requested budget $\sum_i W_i = w_0\, s \sum_i e_i$ is the product of the two, so whichever channel varies more decides its sign; on our measurements the per-position weights dominate, and a difficult round therefore requests a larger total than an easy one before the global ceiling clips it. Figure 5 shows the resulting allocation before the target model verifies the candidates.

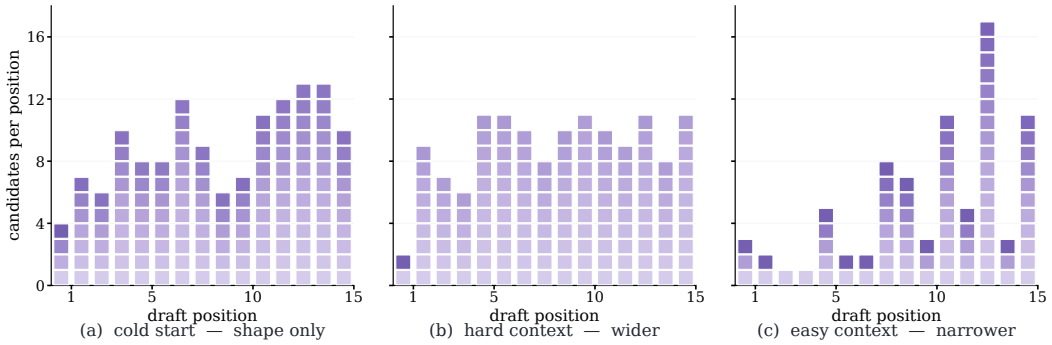

Figure 5: **Round-level reallocation of verification capacity.** Bars in each panel represent the candidates admitted for verification, indexed by draft position. **(a) Cold start.** before the target-model feedback takes effect, only the local draft-confidence prior is active. **(b) Difficult prompt.** the acceptance feedback keeps the global scale bounded, while the small confidence margins make the feed-forward Shape prior request more candidates at each uncertain position; the part beyond the budget ceiling is then truncated. **(c) Easy prompt.** the higher acceptance length raises the scale towards its upper bound, but the large margins shrink the per-position Shape weights, so far fewer candidates are requested in total and computation is saved.

The trace separates the two control timescales. At initialization the quota is determined entirely by the draft-side Shape prior. After the first verification, a low accepted length lowers the next round's Scale multiplier, while a high accepted length raises it. The global cap can clip requested quotas without changing the ordering induced by Shape, so the trace illustrates adaptation of the total budget and its spatial placement separately.

B Hyperparameter Configurations

For the feed-forward Shape module, we set the base allocation width to $w_0 = 16$, the margin temperature to $\tau_\theta = 2.5$, and the reference point to $\tau_{\text{ref}} = 4$. The per-position quotas are strictly bounded by $W_{\min} = 1$ and $W_{\max} = 32$. For the feedback Scale module, we employ an exponential moving average (EMA) step size of $\gamma = 0.5$ and constrain the scale multiplier strictly within the interval $[0.4, 2.0]$ (i.e., $s_{\min} = 0.4, s_{\max} = 2.0$).

To ensure fair comparison and avoid trivial compute scaling, the global candidate cap is enforced at $B_{\max} = 128$ across all main experiments. All candidates are flattened and verified in a single parallel attention pass utilizing an ancestral visibility mask (the strict mathematical construction of which is detailed in Appendix J).

C Additional Diagnostic Figures

This appendix contains the diagnostic plots that support the cost and stationarity arguments in Sections 3.1 and 3.2.

The first diagnostic tests the cost model behind the interior optimum. Figure 6 shows that the fixed verification overhead is large at small candidate counts, while the marginal acceptance gain falls as the candidate set grows. Their combination produces a finite speedup optimum.

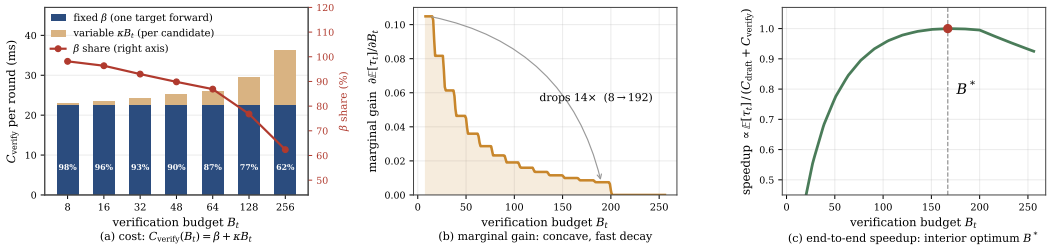

Figure 6: **The verification budget has an interior optimum. (a)** The per-round verification cost decomposes into a large fixed term β (one target forward pass) and a small per-candidate term κB_t. White numbers on the bars denote the share of β in C_{verify}, which decreases from 98% at $B_t = 8$ to 62% at $B_t = 256$ (93% at $B_t = 32$), so that a larger budget amortizes the fixed overhead. **(b)** The marginal gain from enlarging the candidate set, $\partial \mathbb{E}[\tau_t]/\partial B_t$, is strictly concave and decays quickly (a factor of 14 over $B_t \in [8, 192]$). **(c)** With a concave gain and an affine cost, the end-to-end speedup peaks at a finite budget B^*. (Qwen3-4B, $T = 0$, full task grid.)

The second diagnostic examines whether prompt difficulty is stable enough to estimate online. Figure 7 separates variation between prompts from round-level noise and reports the lagged autocorrelation of the acceptance length.

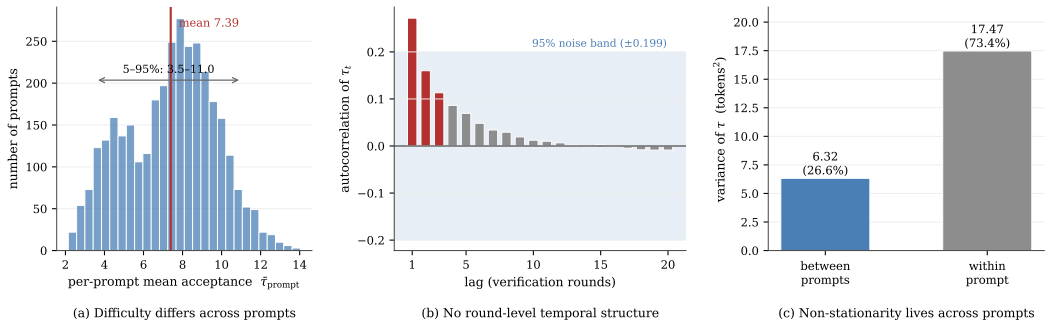

Figure 7: **Statistical properties of the target acceptance length τ_t and its variance decomposition. (a)** Distribution of the per-prompt average acceptance length $\bar{\tau}_{\text{prompt}}$, which shows a $6.5\times$ spread across prompts of different difficulty. **(b)** Lagged autocorrelation of τ_t within a prompt: only lag-1 lies outside the 95% confidence interval (± 0.199), while all remaining lags fall inside the noise band, which indicates a lack of long-range memory across rounds. **(c)** Variance decomposition of τ_t: between-prompt variation (the signal that the controller has to resolve) accounts for 26.6%, whereas within-prompt fluctuation (round-level noise) accounts for 73.4%. (Qwen3-4B, $T = 0$, full task grid.)

Finally, Figure 8 follows the feedback state through a real generation trace. The EMA removes rapid round-to-round fluctuations and keeps the Scale multiplier within its guardrails.

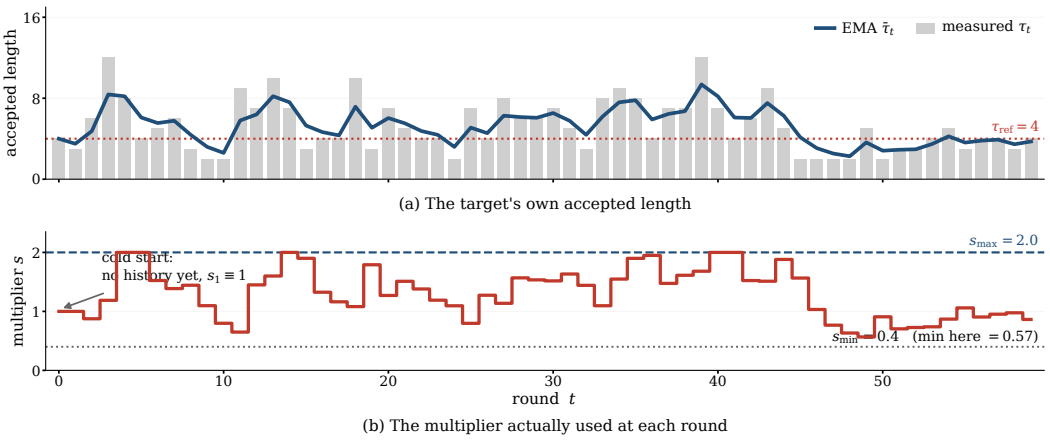

Figure 8: **Behaviour of the feedback Scale controller on a real generation trace. (a)** Gray bars denote the acceptance length τ_t of each round, the dark blue solid line denotes the smoothed EMA state $\bar{\tau}_t$, and the red horizontal dashed line is the reference point τ_{ref}. **(b)** The resulting global scale multiplier, whose lower and upper bounds act as guardrails that limit extreme fluctuations.

D System-level Latency Breakdown

D.1 Sources of Acceleration: Fewer Rounds, Not Cheaper Rounds

To understand where the end-to-end speedup comes from, we perform a fine-grained system-level latency profiling on Qwen3-4B ($T = 0$). There are only two possible sources of acceleration: doing the same work faster, or doing less work in total.

We use Qwen3-4B on GSM8K at $T = 0$ and average over 120 responses per method. Appendix D reports three complementary views of this question. Panel (a) of Figure 1 separates the drafting and verification stages in absolute wall-clock time per 1,000 generated tokens. The standard EAGLE-3 baseline consumes 8.16 s. Thanks to parallelism, DFlash reduces this to 5.16 s (3.00 s saved, -37%), and DSpark to 5.79 s (2.37 s saved, -29%), whereas TDVB reduces it further to 4.00 s (4.15 s saved, -51%).

To separate the effect of the per-round latency from the number of rounds, panel (b) of Figure 1 represents each method as a rectangle. Its width is the per-round verification time (ms/round), its height is the number of verification rounds per 1k tokens (rounds/1k), and its area is the total wall-clock time. EAGLE-3 requires 191 rounds/1k with a per-round time of 42.7 ms/r; DFlash requires 157 rounds/1k with 32.8 ms/r; DSpark requires 160 rounds/1k with 36.2 ms/r; and TDVB achieves 120 rounds/1k with 33.4 ms/r. This decomposition reveals the central mechanism: the "height" of TDVB, that is, the number of rounds, is clearly below that of all baselines, whereas its "width", that is, the per-round verification cost, is almost the same as that of DFlash (only $+1.8\%$). The speedup therefore arises mostly from doing less work, namely from fewer rounds, rather than from making a single round cheaper.

Panel (c) of Figure 1 further decomposes the time saved relative to EAGLE-3 into a part that comes from fewer rounds and a part that comes from cheaper rounds. Of the 2.99 s saved by DFlash, 1.44 s comes from fewer rounds and 1.55 s from cheaper rounds. Of the 4.15 s saved by TDVB, 3.04 s (about 73%) comes from fewer rounds and only 1.12 s from cheaper rounds. These three pieces of evidence agree that the leverage of the dynamic budget controller lies in increasing the effective acceptance length τ_t of each round, which in turn reduces the number of expensive interaction rounds.

The same system trace also confirms the premise on which this paper relies: the drafting stage is no longer the system bottleneck (it accounts for only 17.0% to 17.4% in DFlash and in TDVB), whereas the verification stage, including tree assembly and commit within the same parallel propa-

gation, accounts for 72.7% (EAGLE-3 and DSpark) to 83.0% (TDVB) of the per-round wall-clock time. By tilting the verification capacity towards positions with a high acceptance probability, the dynamic controller turns the same underlying verifier into fewer but fuller effective rounds: the average acceptance length τ per round rises from 6.53 for DFlash to 8.50 for TDVB (+30.1%). In this way, the dynamic budget trades a marginal per-round overhead (+1.8% in time) for a 23.7% reduction in the total number of rounds, which translates into a large end-to-end gain.

E Cross-scale Results

The main text summarizes the cross-scale trend, while Figure 9 reports the complete measurements. The upper row uses a block-8 candidate budget and compares Qwen3-4B with Qwen3-8B under both temperatures. The lower row repeats the comparison with block-16 candidates. The annotations report the gain over the strongest fixed-budget baseline at the same scale. Reading the rows horizontally isolates model scaling, whereas comparing the rows vertically isolates the effect of enlarging the candidate pool.

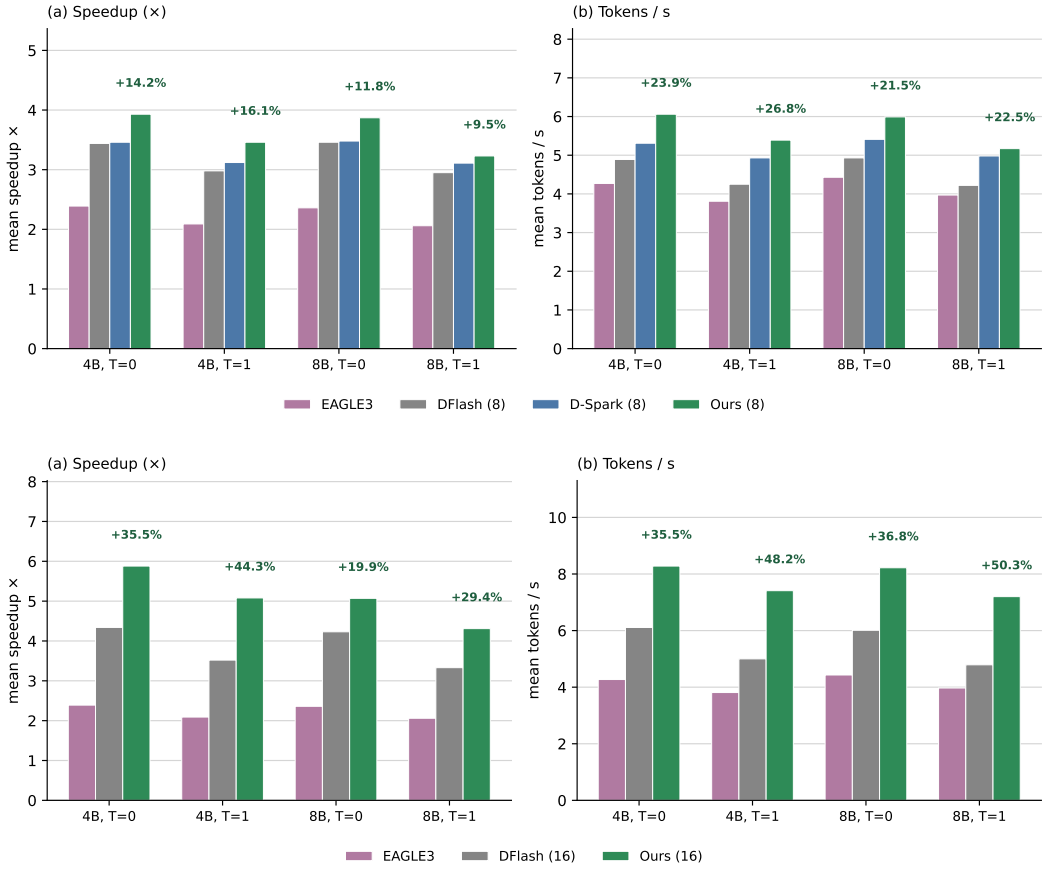

Figure 9: **Cross-scale results for block-8 and block-16 candidate budgets.** Top: block-8; bottom: block-16. Each panel reports average end-to-end speedup relative to autoregressive decoding and average generation throughput over the seven benchmarks. Results cover Qwen3-4B and Qwen3-8B at $T = 0$ and $T = 1$. TDVB consistently outperforms the baselines shown in each panel; annotations report the relative gain over the strongest fixed-budget DFlash baseline at the same scale.

F PER-ROUND CONTROLLER ALGORITHM

This appendix gives the complete per-round pseudocode for the training-free dynamic verification budget controller. The algorithm has one draft forward pass, $O(K)$ quota arithmetic, one candidate-gathering step, and one target-model verification pass per round. The only persistent adaptive state is the scalar $\bar{\tau}_t$; all bounds are fixed hyperparameters. Since the quota is finalized before target verification and the commit operation accepts the same prefix as standard speculative sampling, the controller changes the amount and placement of computation without changing the sampled distribution. The algorithm makes the information flow explicit: Shape computes unnormalized position-wise weights from the draft margins, Scale converts the previous EMA state into a global multiplier, and their bounded fusion produces the per-position quotas. The global cap is enforced before candidate gathering and target-model verification. The acceptance feedback is used only after verification to update the EMA for the next round; it does not alter the target-model logits or the lossless acceptance rule.

Algorithm 1 Training-free dynamic verification budget controller (one decoding round)

Require: target model p, draft model q, base width w_0, reference τ_{ref}, step γ, scale bounds $(s_{\min}, s_{\max})$, position bounds $(W_{\min}, W_{\max})$, global cap $B_{\max}$, margin temperature τ_θ, horizon K

State: $\bar{\tau} \leftarrow \varnothing$ ▷ EMA state, unset at $t = 1$

$D \leftarrow \text{DraftForward}(q, x_{<t})$ ▷ one non-causal pass

for $i = 1$ **to** K **do**

  $(q^{(1)}, q^{(2)}) \leftarrow$ top-2 probabilities of $D[i]$

  $m_i \leftarrow \log q^{(1)} - \log q^{(2)}$ ▷ confidence margin

  $e_i \leftarrow \exp(-m_i/\tau_\theta)$ ▷ position-wise weight (Shape)

end for

if $\bar{\tau} = \varnothing$ **then**

  $s \leftarrow 1.0$ ▷ cold start

else

  $s \leftarrow \text{clip}(\bar{\tau}/\tau_{\text{ref}},\ s_{\min},\ s_{\max})$ ▷ Scale

end if

for $i = 1$ **to** K **do**

  $W_i \leftarrow \text{clip}(\text{round}(w_0 \cdot s \cdot e_i),\ W_{\min},\ W_{\max})$

end for

$W \leftarrow \text{GlobalCap}(W, B_{\max})$ ▷ $\sum_i W_i \le B_{\max}$

$B \leftarrow \text{GatherTopCandidates}(D, W)$ ▷ $|B| = \sum_i W_i$

$M \leftarrow \text{AncestralVisibilityMask}(B)$ ▷ causal mask within B

$\tau \leftarrow \text{TargetVerify}(p, B, M)$ ▷ one parallel pass

if $\bar{\tau} = \varnothing$ **then**

  $\bar{\tau} \leftarrow \tau$ ▷ initialize EMA

else

  $\bar{\tau} \leftarrow \gamma \cdot \tau + (1 - \gamma) \cdot \bar{\tau}$ ▷ EMA update

end if

$x_{<t+\tau} \leftarrow \text{Commit}(B, \tau)$ ▷ advance KV-cache

return $x_{<t+\tau}, \bar{\tau}$

The algorithm has one draft forward pass, $O(K)$ quota arithmetic, one candidate-gathering step, and one target-model verification pass per round. The only persistent adaptive state is the scalar $\bar{\tau}_t$; all bounds are fixed hyperparameters. Since the quota is finalized before target verification and the commit operation accepts the same prefix as standard speculative sampling, the controller changes the amount and placement of computation without changing the sampled distribution.

G PER-POSITION ACCEPTANCE RATE

This appendix isolates the position-wise consequence of the budget decision. Figure 10 plots the survival function $P(\tau_t \geq k+1)$, so a higher curve means that candidate paths remain available deeper into the draft block. All four methods use the same prompts and the same block-7 engine. TDVB maintains the highest survival probability across positions, which explains why its average accepted length is larger even when its per-round verification cost is comparable to DFlash.

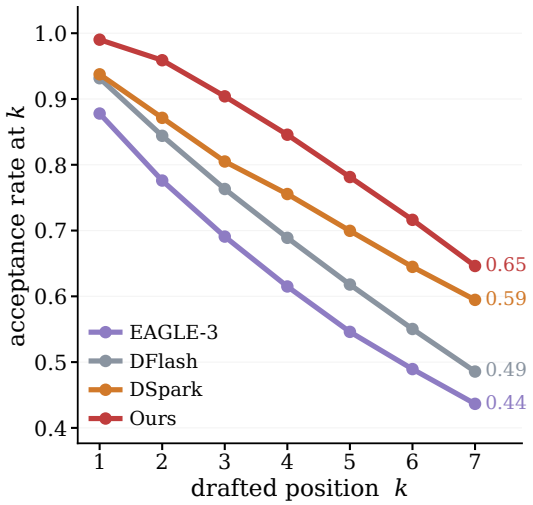

Figure 10: **Per-position acceptance rate.** On the block-7 engine, all four methods use the same prompts. Curves show the survival function $P(\tau_t \geq k+1)$ over draft position k. TDVB is highest at every position ($0.99 \rightarrow 0.65$), yielding an expected 5.84 accepted tokens per round versus 5.31, 4.88, and 4.38 for DSpark, DFlash, and EAGLE-3 ($+10\%$, $+20\%$, and $+33\%$).

H IMPLEMENTATION INVARIANTS AND DISTRIBUTION PRESERVATION

The controller changes only which draft candidates are admitted to the verifier. It does not alter the target-model logits, the acceptance test, or the rule that commits the accepted prefix. At round t, all candidate quotas are computed before target verification from the draft margins and the previous EMA state. The target model then evaluates the assembled candidate pool under the ancestral visibility mask, and exactly the accepted prefix is committed. Consequently, conditioning on the same draft randomness and target-model randomness, the committed sequence is identical to standard speculative decoding; the controller changes latency and acceptance length but not the target distribution.

The per-round controller state consists of the scalar EMA $\bar{\tau}_t$ and the fixed bounds $(s_{\min}, s_{\max}, W_{\min}, W_{\max})$. Its memory overhead is therefore $O(1)$ beyond the candidate pool and mask. The quota computation is $O(K)$, while mask construction and the measured latency of the actual implementation are reported in Appendix J.

I SAMPLING CONFIGURATIONS AND METRICS

Sampling configurations. We report results under two decoding regimes. The deterministic regime uses greedy decoding with temperature $T = 0$. The stochastic regime uses nucleus (top-p) sampling with temperature $T = 1$; unless stated otherwise, we keep the default nucleus threshold of the serving framework and do not apply additional top-k truncation. Generation is stopped either when the model emits an end-of-sequence token or when the maximum generation length of 2048 tokens is reached.

End-to-end wall-clock speedup. For each method we measure the total wall-clock latency needed to generate the same output for the same prompt, with a batch size of one on a single NVIDIA A100 GPU. The end-to-end wall-clock speedup relative to autoregressive decoding is defined as

$$\text{Speedup} = \frac{T_{\text{AR}}}{T_{\text{method}}}, \tag{8}$$

where T_{AR} and T_{method} are the total latencies of autoregressive decoding and of the method under test, respectively. Latency is averaged over the responses collected for each benchmark, and the throughput reported in Figure 9 is the reciprocal of the per-token latency averaged in the same way. All timings exclude model loading and warm-up iterations.

Average acceptance length. The average acceptance length $\bar{\tau}$ is the mean number of draft tokens that the target model accepts per verification round, computed over all rounds of a response and then averaged over the responses of a benchmark. A larger $\bar{\tau}$ means that each verification round produces more output tokens, and it is the quantity that the dynamic budget controller directly targets.

J Construction of the Ancestral Visibility Mask

Inside the flattened batch of size $|B| = \sum_i W_i$, a candidate token j must attend only to the prefix of the sequence that it actually continues. Concretely, each candidate carries an *ancestor identifier*, namely the position index i at which it was drafted, and a *parent identifier* within the candidate list of that position. The mask is then

$$\mathbf{M}_{j\ell} = \mathbb{1}[\ell \text{ is an ancestor of } j, \text{ or } \ell = j]\,. \tag{9}$$

We state the complexity of the implementation that we actually use, rather than an idealized one. The mask is built by copying the parent row of each node. Its time cost is $O(L^2)$, and it stores a dense boolean array of size $O(L^2)$, where L is the number of tree nodes. At $L = 128$ this is 16 KB, and it is measured at 0.0019 s per round. This is negligible in absolute terms.

The visible set of a node is the *ancestor prefix* $[0, \text{parent}(v)]$ together with the diagonal, not the full prefix $[0, v]$, because the nodes between the parent and the node itself belong to sibling subtrees and must be excluded. Flattening the tree in topological order turns the visible set into a contiguous ancestor prefix, which is why the construction reduces to a single row copy per node; the resulting sparsity pattern.

K Details of the Acceptance Calibration Baseline

What AC is. Acceptance Calibration (AC) is an alternative baseline to the TDVB exponential prior $e_i = \exp(-m_i/\tau_\theta)$. Instead of assuming that the acceptance probability follows a fixed exponential shape, AC sets the number of candidates to keep at each margin position, that is, the candidate width, directly from the empirical curve $P(\text{accept} \mid m)$ estimated by rolling statistics.

Method.

1. The margin range $[0, C_{\max}]$ is divided uniformly into M bins.
2. After each decoding round, an exponential moving average (EMA) maintains the empirical acceptance-rate estimate of each bin: $\text{acc}_i \leftarrow \gamma \cdot \text{acc}_i + (1-\gamma) \cdot$ (fraction of accepted positions in this bin in this round), initialized with $\text{acc}_i = 0.5$ as an uninformative prior.
3. The controller decides the candidate width from the current empirical acceptance rate of the bin, instead of from $\exp(-m_i/\tau_\theta)$.

Parameters. The calibration interval is bounded by $C_{\max} = 10.0$ (margins outside $[0, 10]$ are clipped), the EMA smoothing coefficient of the per-bin acceptance-rate estimate is $\gamma = 0.9$, the binning granularity is $M \in \{2, 5, 10, 20, 40\}$, and the number of feedback observations per round is $K = 15$, which equals the number of draft positions and determines the location of the sample-complexity wall. For reference, $M = 0$ corresponds to no calibration (base), and a scalar prior with $M \approx 1$ corresponds to the scalar form used by TDVB.

M	Samples per bin per round (K/M)	Speedup ($\times$)	τ
0 (base, no calibration)	—	5.793	8.377
1 (TDVB, scalar)	15.00	**5.918**	**8.530**
2	7.500	5.857	8.530
5	3.000	5.697	8.308
10	1.500	5.576	8.118
20	0.750	5.535	8.101
40	0.375	5.596	8.157

Table 3: **Effect of the binning granularity of the Acceptance Calibration baseline.** GSM8K, Qwen3-4B, $T = 0$, $n = 64$. A coarse prior is better than a fine-grained empirical estimate.

Results. As shown in Table 3, the speedup is best near the scalar setting ($M \approx 1$, which is TDVB) and decreases monotonically as the binning becomes finer. At $M = 20$ it is only $5.535\times$, which is about $0.38\times$ below the scalar variant. The underlying reason is a *sample-complexity wall*: each round provides only $K = 15$ observations, so once $M > K$ each bin receives fewer than one sample per round, and the EMA estimate degenerates into noise. An "empirical binning" therefore performs worse than a parametric exponential prior. This also explains why the AC variant is consistently the weakest entry in the ablation Table 2.