TARGET CORRECTION FOR DIFFUSION SAMPLING AND SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

During LLM inference, control algorithms decide which tokens to resample for diffusion models and which request tokens to verify for speculative decoding. For the sake of efficiency and simplicity, these algorithms do not attempt long-horizon planning to hit a specified target—fidelity for diffusion sampling, or goodput for speculative decoding. This paper presents *target correction*, a technique for noninvasively imbuing simple control algorithms with target awareness. The approach (1) interprets the algorithm as myopically playing a betting game in which wealth multiplies according to random outcomes, (2) devises a target-aware utility function of wealth, (3) derives a target-aware value function from a continuous-time approximation of the game, and (4) uses that value function to modify the original algorithm. Our correction of the Entropy-Bound diffusion sampler in DiffusionGemma achieves 4% lower NFE at comparable or better fidelity. To illustrate how surgical target correction can be, we improve DSpark's goodput robustness under skewed workloads by up to 10% with just a single-line change.

1 INTRODUCTION

In modern inference engines, simple but crucial algorithms control which tokens are processed in each forward pass. These decisions are made with limited resources and information, but substantially affect throughput, latency, and output quality.

- **Speculative decoding.** A draft model proposes future tokens for requests in a batch. A scheduler decides which contiguous tokens from each request, and how many tokens overall, should be verified by the larger target model. Verifying more tokens can potentially improve throughput, but makes the forward pass slower (hurting latency), and runs the risk of wasted computation due to rejections. The scheduler's long-term objective is to maximize throughput while ensuring acceptable latency.
- **Discrete diffusion sampling.** A denoising model predicts the marginal distributions of many tokens in parallel. The sampler chooses which of these predictions should be used in the next round of denoising. Keeping many predictions exploits parallelism to hasten convergence. However, because the marginal distributions don't account for dependencies among the positions, this may reduce output quality. The sampler therefore manages a parallelism–fidelity tradeoff in order to reach a desired fidelity as quickly as possible.

Ideally, these local decisions would account for both the desired end target and the progress already made toward it. Doing so is difficult: the effect of a decision made now on eventual latency or fidelity depends on future model predictions, random outcomes, and subsequent control decisions. Explicitly planning over those future states would introduce substantial complexity into controllers whose main virtues are simplicity and efficiency.

We introduce *target correction*, which makes local control decisions target aware—that is, cognizant of their progress toward a system objective —without complex planning over future states. Our framework interprets the original algorithm as making (economically) myopic decisions: it acts on the value of the opportunity immediately in front of it, without accounting for how previous outcomes have changed the value of taking that opportunity now. This myopia stems from the algorithm (implicitly) adopting a linear utility function of its "wealth" (a quantitative measure of its progress).

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.

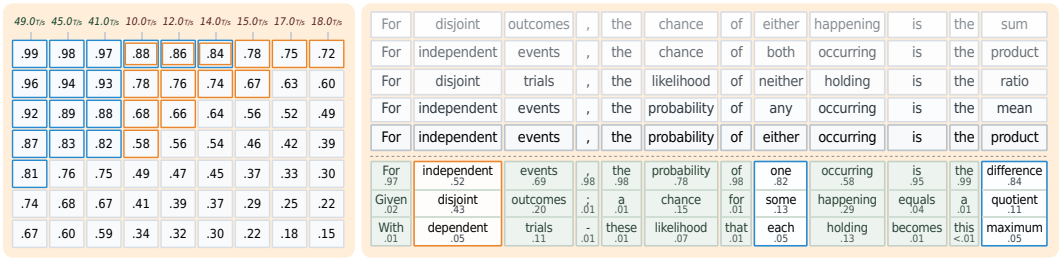


Figure 1: **Left (Speculative Decoding):** Columns are active requests and rows are speculative depths. Each cell shows the probability that the verifier accepts all tokens up to a given length, as predicted by a confidence head. DSpark concentrates verification on high-acceptance, already-fast requests, since that decision maximizes overall throughput. In contrast, DStreak reallocates capacity toward less-served requests, since its utility function targets fairness. **Right (Diffusion Sampling):** Columns are token positions and rows are successive generations produced by a denoising model. The bottom row shows each position's current top-3 marginal. There is a *stable ambiguity* between 'disjoint' and 'independent'; these have persisted as modes at the 2nd position. This is distinct from *churning modes* seen at other positions: for example, there is essentially no historical consistency at the last position. Entropy-Bound (EB) is unable to distinguish these, since it selects which tokens to sample based solely on the entropy of the current marginals. In contrast, Steady Entropy-Bound (SEB) can preserve stable ambiguities.

To correct this, we incorporate an application-specific utility function which accounts for progress towards the system objective. We develop a simple continuous-time model of the algorithm's decisions over time. This model intentionally does *not* account for the full complexity of the algorithm's environment; it is an ansatz in which the long-horizon value of accumulated wealth can be derived. The resulting value function $V(W)$ can then be used to steer local decisions according to current wealth.

1.1 This Paper

Target correction. We develop target correction using a classical multiplicative betting game. Given an application-specific terminal utility, a stationary continuous-time approximation yields a value function $V(W)$ as a solution. This section describes how to obtain such solutions and offers guidance on how to incorporate them into controller decisions.

DStreak adaptive verification. First, we apply target correction to DSpark, an adaptive speculative-verification scheduler (Cheng et al., 2026). For each request, wealth is its realized generation rate, while the terminal utility encodes α-fairness across requests. Target correction changes only how DSpark weights speculative opportunities across requests; its acceptance predictions, prefix structure, global ranking, and hardware-aware verification-budget sweep are unchanged. DStreak incorporates fairness with a single-line modification to the DSpark scheduler.

Discrete diffusion sampling. Next, we apply target correction to the Entropy-Bound sampler used by DiffusionGemma (Ben-Hamu et al., 2025; DiffusionGemma Team, 2026). Entropy-Bound is stateless and therefore cannot distinguish a persistent ambiguity among a few token modes from marginals whose modes churn across denoising passes. Our Steady Entropy-Bound sampler maintains a small amount of per-position state to make this distinction, achieving comparable or better quality with approximately 4% fewer denoiser forward passes.

2 Target Correction

2.1 A Betting Game and a Myopic Bettor

Multiplicative wealth processes are classical in portfolio theory and capital-process formulations of probability (Ville, 1939; Kelly, 1956; Cover, 1991). They provide a simple way to model repeated

decisions whose realized gains and losses affect how much can be committed to future decisions. Target correction models the decisions of an inference controller as bets in the following game. In particular, the existing controller should be equivalent to making bets using the linear value function $V(W) = W$.

> Start from $W_0 > 0$. For rounds $t = 1, \ldots, T$:
>
> **1. Make a bet.** Choose λ_t by solving an optimization involving $V(W)$.
> **2. Observe the outcome.** Observe a random net-return vector Z_t.
> **3. Settle wealth.** Set $W_t = W_{t-1}(1 + \lambda_t^\top Z_t)$.

The existing controller may operate according to this linear-value criterion even if it does not keep track of its current wealth; this is the case for both of our applications. We interpret the use of a linear value function as the cause of the controller's myopia. This interpretation stems from portfolio theory, where repeatedly maximizing expected linear utility can favor strategies that eventually lead to zero wealth, despite positive expected wealth growth (Kelly, 1956; Breiman, 1961).

2.2 The terminal utility

To enable target-aware decisions, we define the target according to a terminal utility function $u(W_T)$. The long-horizon goal is to maximize $u(W_T)$, where W_T is the wealth at the stopping time T of the game. This utility function can encode a specific target wealth: for example, $u(W_T) = \min(W_T, W_{\text{max}})$. Alternatively, it can embed parameters which influence how utility grows with wealth. Both of this paper's applications fall in the latter category.

In inference, the total number of rounds T (e.g., forward passes) is typically uncertain. Controllers use information that changes over time, but their rules are usually stationary with respect to time itself. To model this, we draw future opportunities from a stationary local return law and take T to be independent and memoryless. In discrete time, T is geometric with constant hazard $1/\rho$, and the controller does not know it in advance. This parameter ρ often just drops out of subsequent derivations; this will occur in both of this paper's applications.

2.3 Deriving a target-aware V

To derive an (intermediate) value function V from the terminal value u, we approximate the discrete game in continuous time. Under a small-step scaling of the return randomness Z, let μ and Σ denote its local mean and covariance. Then

$$\frac{dW_t}{W_t} = \lambda_t^\top \mu \, dt + \lambda_t^\top \Sigma^{1/2} dB_t. \tag{1}$$

Here B_t is a standard Brownian motion of matching dimension, while μ and Σ are the continuous-time counterparts of the first two moments of Z. Under the same scaling, the geometric terminal clock becomes exponential with hazard $1/\rho$, so $T \sim \text{Exp}(1/\rho)$. Section A in the Appendix derives both limits from the discrete game. In the continuous-time version, the value function is:

$$V(W) = \sup_{\pi} \mathbb{E}[u(W_T) \mid W_0 = W],$$

where π ranges over admissible future betting policies. Under regularity conditions, we may solve for V (e.g., Fleming and Soner, 2006).

Lemma 1 (The value function)**.** *Suppose u is continuous, increasing, and concave, $\Sigma \succ 0$, and the admissible bet set Λ is compact. At any $W > 0$ where the value function V is twice differentiable,*

$$0 = \sup_{\lambda \in \Lambda} \left\{ WV'(W)\lambda^\top \mu + \frac{1}{2} W^2 V''(W) \lambda^\top \Sigma \lambda \right\} + \frac{1}{\rho}\big[u(W) - V(W)\big]. \tag{2}$$

If additionally $V''(W) < 0$ and the unconstrained maximizer lies in the interior of Λ, then

$$V(W) - u(W) = \frac{\rho}{2} \left(\mu^\top \Sigma^{-1} \mu\right) \left(-\frac{V'(W)^2}{V''(W)}\right). \tag{3}$$

A proof of this result is in the Appendix. When applied to prototypical utility functions, this result yields value functions V that are closed form or easily computed.

Algorithm	DStreak	Steady Entropy-Bound (SEB)
Account i	One active generation request.	One token position in a diffusion output block.
New information	Verification probabilities a_{ij}.	Marginal p_i and previous top-k token probabilities.
Wealth W_i	Realized request throughput N_i/T_i.	Historical quality and stability of the i'th marginal.
Bet λ_i	Choose how many speculative tokens to verify.	Accept p_i, or reject and uniformly renoise.
Return	Verified tokens update realized throughput exactly.	Reject returns W_i^{-1}; accept returns R_i.
User target	Fairness target α; $\alpha = 0$ recovers DSpark.	Entropy budget γ, shared with Entropy-Bound.
Utility $u(W)$	$W^{1-\alpha}/(1-\alpha)$ for $\alpha \neq 1$; $\log W$ for $\alpha = 1$.	$\log W$ (Kelly).
Baseline algorithm	Verify B highest a_{ij}.	Accept ℓ lowest-entropy tokens where $C_\ell \leq \gamma$.
Target correction	Verify B highest $V'(W_i)a_{ij}$.	Accept iff $\log W_i + \gamma - C_i$ is at least the churn divergence.

Table 1: Target-correction concepts for DStreak and Steady Entropy-Bound Sampling.

3 DSTREAK: FAIR ADAPTIVE VERIFICATION FOR SPECULATIVE DECODING

In speculative decoding, a small model drafts ℓ output tokens for a larger model to verify. Because multiple tokens can be verified in a single forward pass, speculative decoding is especially effective for memory-bound inference. However, servers become compute bound when they handle many requests; then, verifying highly speculative tokens might waste compute resources that could have been used for other requests. The *adaptive verification* task is to choose, for each request i in a batch of size R, how many of its ℓ draft tokens to verify.

To facilitate this, DSpark (Cheng et al., 2026) uses a confidence model to predict the probabilities a_{ij} that verification will succeed up to token position $j+1$ for request $i \in [R]$. Since a forward pass produces a single "free" next token, the position j is offset by 1. Speculative decoding fails at the first rejected token, so the probabilities decrease in j. At server startup, DSpark tunes an estimate $\widehat{T}(B)$ of how long verifying B tokens takes; this is the dominant time cost in Liu et al. (2026). DSpark then chooses the lengths to maximize the overall tokens per second (aggregate throughput) of verification:

$$\max_{\text{lengths } v_i} \frac{\text{expected number of verified tokens}}{\text{estimated time to verify}} = \max_{v_i \in \mathbb{Z}} \frac{1 + \frac{1}{R}\sum_i \sum_{j=1}^{v_i} a_{ij}}{\widehat{T}(\sum_i v_i)} \quad \text{s.t. } 0 \leq v_i \leq \ell. \tag{4}$$

To solve this optimization, sort all the a_{ij} in descending order and select the first B of them. Choose B to maximize the objective.

Aggregate throughput is an important performance objective, but it doesn't accommodate service-level agreements at the level of individual requests. Imagine a server responding to a small fraction of very easy (high a_{ij}) requests among a larger batch of normal requests; this scenario is depicted in Figure 1. DSpark may preferentially divert resources to this small fraction of requests at the expense of the others. This could potentially decrease goodput, i.e., the throughput originating from requests that individually have satisfactory latency.

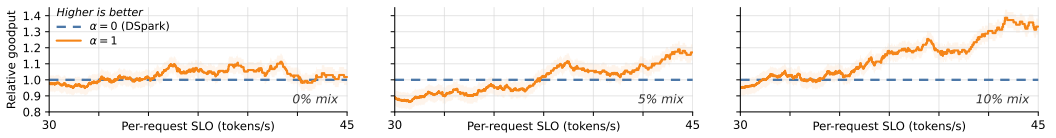
We generalize DSpark to incorporate the following quantities:

- a parameter $\alpha \geq 0$ controlling the fairness target. The case $\alpha = 0$ recovers DSpark. Increasing α strengthens the preference for request-level fairness;
- request-level wealths W_i. These are simply realized, per-request throughputs.

DStreak values every token generated for request i according to its marginal state value $V'(W_i)$. Thus, each optional speculative-token probability a_{ij} is multiplied by $V'(W_i)$, while the mandatory free token produced for each request contributes the corresponding marginal value $V'(W_i)$. The new algorithm, DStreak, is a simple illustration of how target correction can steer a system toward desired behavior with a relatively gentle modification.

3.1 THE ADAPTIVE VERIFICATION GAME

The wealth W_i of request i is its realized throughput N_i/T_i, where N_i is its number of generated tokens and T_i its accumulated decode time. It is easy to express its update in the form of the betting game. The bet is a binary vector $\lambda_i = (1, \lambda_{i1}, \ldots, \lambda_{i\ell})$ which encodes v_i. (The first coordinate



Inputs: probabilities a_{ij}.

$s_{ij} \leftarrow a_{ij}$
sort all s_{ij}; $S(B) \leftarrow$ sum of top B
$B^\star \leftarrow \arg\max_B [R + S(B)]/\widehat{T}(B)$
verify first $B^\star$ tokens from sorted order

Inputs: probabilities a_{ij}; wealth $W_i = N_i/T_i$; fairness α.

$s_{ij} \leftarrow W_i^{-\alpha} a_{ij}$
sort all s_{ij}; $S(B) \leftarrow$ sum of top B
$B^\star \leftarrow \arg\max_B [\sum_i W_i^{-\alpha} + S(B)]/\widehat{T}(B)$
verify first $B^\star$ tokens from sorted order; update W_i

Figure 2: **Top:** Relative goodput is the rate of requests meeting the per-request throughput SLO, normalized by the DSpark baseline; 1.0 denotes equal performance. $\alpha = 1$ (proportional fairness) tends to preserve more goodput at higher fractions of "easy" requests mixed into the pool. These easy requests have very high predicted acceptance probabilities a_{ij}, which distract the throughput-focused DSpark. **Bottom:** DStreak (right) is a very mild modification of DSpark (left). DStreak serves as a deliberately minimal illustration of target correction.

represents the free token). Let $A_{ij} \in \{0, 1\}$ indicate whether verification survives through position j, so $\mathbb{E}[A_{ij}] = a_{ij}$. Suppose the verification pass, estimated to take time $\widehat{T}(B)$, actually took time Δ. Then, the following update maintains the realized throughput:

$$W_i(1+\lambda_i^\top Z_i) = \frac{N_i + 1 + \sum_j \lambda_{ij} A_{ij}}{T_i + \Delta} \quad \text{where} \quad Z_{i0} = \frac{T_i}{T_i + \Delta}\left(1 + \frac{1}{N_i}\right) - 1 \text{ and } Z_{ij} = \frac{T_i}{T_i + \Delta}\frac{A_{ij}}{N_i}$$

Now, we define the utility-based objective that a bettor maximizes. It simply replaces the average in the numerator of Equation (4) with a weighted average.

$$\max_{v_i \in \mathbb{Z}} \frac{1}{R} \sum_i V'(W_i)\Big(1 + \sum_{j=1}^{v_i} a_{ij}\Big) \Big/ \widehat{T}\Big(\sum_i v_i\Big) \quad \text{s.t. } 0 \le v_i \le \ell. \tag{5}$$

To gain intuition for this objective, inspect the units of each term. W_i is measured in $\frac{\text{tokens}}{\text{second}}$ and $V(W_i)$ in units of utility. So, $V'(W_i)$ has units $\frac{\text{utility}}{\text{tokens/second}} = \frac{\text{utility}\cdot\text{second}}{\text{token}}$. Meanwhile, $1 + \sum_{j=1}^{v_i} a_{ij}$ is the expected number of tokens produced for request i, so $V'(W_i)\big(1 + \sum_{j=1}^{v_i} a_{ij}\big)$ has units of utility $\cdot$ second. Dividing by the predicted duration $\widehat{T}(\sum_i v_i)$ gives $\frac{\text{utility}\cdot\text{second}}{\text{second}} = \text{utility}$ as the objective units. Since the $V'(W_i)a_{ij}$ still descend with j, the sorting-based algorithm remains applicable.

3.2 DSpark uses the linear utility

DSpark is recovered by the linear bettor in our adaptive verification game: $V'(W) = 1$ when $V(W) = W$. (Generalizing myopic algorithms using target correction can be very straightforward.)

3.3 DStreak uses the CRRA utility

DStreak adopts the constant relative risk aversion (CRRA) terminal utility function parameterized by $\alpha \ge 0$. Up to constants, the terminal utility and value functions are the same:

$$u_\alpha(W) \propto V_\alpha(W) \propto \quad \frac{W^{1-\alpha}}{1-\alpha} \text{ if } \alpha \neq 1, \quad \text{else } \log W. \tag{6}$$

Without affecting the algorithm in any way, we normalize these so $V'_\alpha(W) = W^{-\alpha}$ without a leading constant. (See Section B.1 for these details). This CRRA (or isoelastic) family is exactly the α-fair utility family used in network resource allocation (Mo and Walrand, 2000; Borst et al., 2013): $\alpha = 0$ is linear utility, $\alpha = 1$ gives proportional fairness, and increasing α places progressively more weight on lower-throughput requests. When $\alpha = 1$, $V'_1(W_i) = 1/W_i$, so a speculative token is worth more for a request whose realized tokens/second is lower. Increasing α strengthens this preference while leaving DSpark's confidence estimates and profiled verification costs unchanged.

3.4 DSTREAK IMPROVES GOODPUT ROBUSTNESS

We experimentally demonstrate that DStreak can help improve fairness in LLM inference. Our experiment uses SPEED-Bench (Abramovich et al., 2026) — developed to realistically evaluate speculative decoding algorithms — which contains requests of low and high entropy. Into a pool of high entropy requests, we introduce a small fraction (either 5% or 10%) of low-entropy requests which have very high a_{ij}. We measure the impact on goodput: the rate of requests/second that meet a TPOT (time per output token) threshold. This metric balances aggregate throughput against the need to serve individual requests with tolerable latency. We normalize it by the goodput obtained by DSpark at the same TPOT threshold. The batch size is set to 128, which is realistic for production workloads. We test Gemma 4 12B (Gemma Team et al., 2026) on requests of length 2K.

As shown in the top panel of Figure 2, DStreak remains close to DSpark when the pool contains no easy-request mixture, then increasingly improves relative goodput as the easy-request fraction rises. It is important to recognize that DSpark's goodput suffers significantly as increasingly harsh TPOT requirements are imposed, so a value of $\alpha = 1$ may not necessarily be desirable in production. Our next application, to diffusion sampling, achieves larger practical improvements by altering the baseline algorithm by more than just a single line.

4 STEADY ENTROPY-BOUND DIFFUSION SAMPLING

A denoising model takes L tokens and produces marginal distributions $p_1, \ldots, p_L$ in parallel. The controller combines these marginals with its own state to sample L tokens. These tokens are either denoised for another round or returned as output. The controller's key decision is accepting a subset $S \subseteq [L]$ of positions: at accepted $i \in S$ it draws a token using p_i. At $i \notin S$, it draws a uniformly random token from the vocabulary (in DiffusionGemma) or replaces it with a mask token (in masked diffusion).

Choosing a singleton S is akin to autoregressive generation. Taking a larger S exploits parallelism but potentially hurts output fidelity by ignoring interdependencies among the chosen marginal distributions. This loss in output fidelity (as quantified by the KL divergence) can be bounded using the entropies $H_i = -\sum_v p_i(v) \log p_i(v)$ of the marginals. For any S, if p_S is the unknown joint distribution over positions in S, then the fidelity loss (as measured by the KL divergence) is at most $B(S)$:

$$\mathrm{KL}\big(p_S \,\|\, \prod_{i \in S} p_i\big) \le B(S) := \sum_{i \in S} H_i - \max_{i \in S} H_i. \tag{7}$$

Entropy-Bound (EB) sampling (Ben-Hamu et al., 2025) chooses the largest S which keeps the bound B below some fixed threshold γ. Without loss of generality, suppose the positions are ordered from low to high entropy: $H_1 \le \cdots \le H_L$. Then it is optimal to choose the first ℓ positions, for some ℓ. Let $C_i = B(\{1, \ldots, i\}) = \sum_{j<i} H_j$. Entropy-Bound sampling chooses the largest ℓ where $C_\ell \le \gamma$.

EB is based solely on current entropies and maintains no state. As such, it cannot distinguish distributions having *churning modes* from those having historically stable modes. Figure 1 illustrates this dilemma. A historically-stable ambiguity among a few candidate tokens is conceptually very different from random fluctuation over arbitrary multimodal distributions. Preserving stable ambiguities may be important for challenging reasoning tasks. Our new algorithm, Steady Entropy-Bound sampling (SEB), maintains enough state to discriminate churning modes from stable ones.

4.1 THE DIFFUSION SAMPLING GAME

In SEB, the wealth W_i measures the historical quality and stability of the marginal at token position i. In the diffusion sampling game, the binary bet encodes whether the marginal p_i should be accepted (and sampled) or whether it should be rejected (and renoised). Rejection resets the wealth to $W_i = 1$. Accepting randomly updates the wealth according to (1) the potential fidelity loss incurred by sampling $Y_i \sim p_i$, per the entropy bound, and (2) how well p_i matches the previously-sampled marginal p_i^-. The latter can be measured by the likelihood ratio $\frac{p_i^-(Y_i)}{p_i(Y_i)}$. It has mean 1, and its variance is the χ^2-divergence of p_i^- from p_i, so the bet becomes riskier as the current and previous marginals diverge.

Inputs: marginals p_i; budget γ.

compute H_i; sort positions by increasing H_i
$C_i \leftarrow \sum_{j<i} H_j$
$g_i \leftarrow \gamma - C_i$
if $g_i \geq 0$: draw $Y_i \sim p_i$
else: draw $U_i \sim$ uniform

Inputs: p_i; budget γ; wealth W_i; p_i^- on its top-k tokens T_i^-.

compute H_i; sort positions by increasing H_i
$C_i \leftarrow \sum_{j<i} H_j$
$g_i \leftarrow \gamma - C_i + \log W_i - \mathrm{KL}(\mathrm{chop}(p_i, T_i^-) \| \mathrm{chop}(p_i^-, T_i^-))$
if $g_i \geq 0$: draw $Y_i \sim p_i$; store its top-k as (T_i^-, p_i^-)
else: draw $U_i \sim$ uniform; $W_i \leftarrow 1$; $T_i^- \leftarrow \varnothing$

Figure 3: Entropy-Bound (EB) (left) ranks positions by current entropy and accepts according to the shared entropy budget γ. Steady Entropy-Bound (SEB) (right) preserves the same entropy ordering and budget, but augments the decision with persistent wealth and the churn divergence, allowing historically stable marginals to be accepted more readily than churning marginals.

It isn't practical to retain the previous marginal distribution at every position $i \in [L]$, so SEB uses a top-k truncation of those marginals. (We set $k = 32$). That is, SEB retains the probabilities $p_i^-(v)$ of the previously sampled marginal for its top-k tokens T_i^-. This set is empty at the outset, and is also emptied every time the marginal p_i is rejected. With an appropriately-defined truncation operator, the likelihood ratio based on the top-k truncation retains its mean 1:

$$\mathbb{E}_{Y_i \sim p_i}\left[\frac{\mathrm{chop}(p_i^-, T_i^-)(Y_i)}{\mathrm{chop}(p_i, T_i^-)(Y_i)}\right] = 1 \quad \text{where} \quad \mathrm{chop}(q, T_i^-)(y) := \begin{cases} q(y), & y \in T_i^-, \\ 1 - \sum_{v \in T_i^-} q(v), & y \notin T_i^-. \end{cases}$$

With Z_i defined as follows, accepting multiplies wealth by R_i, whereas rejecting makes it 1.

$$Z_i(Y_i) := \begin{pmatrix} W_i^{-1} - 1 \\ R_i(Y_i) - 1 \end{pmatrix} \quad \text{where} \quad R_i(Y_i) := \exp(\gamma - C_i)\frac{\mathrm{chop}(p_i^-, T_i^-)(Y_i)}{\mathrm{chop}(p_i, T_i^-)(Y_i)}.$$

Note that C_i is a weighted sum of log probabilities and γ is a threshold on them. Taking logs, we see $\log R_i$ adds and subtracts terms of the same units. Taking the expectation, we see that accepting grows wealth when the fidelity loss bound and the "churn divergence" are low:

$$\mathbb{E}\ \log R_i(Y_i) = \gamma - C_i - \mathrm{KL}\big(\mathrm{chop}(p_i, T_i^-) \,\big\|\, \mathrm{chop}(p_i^-, T_i^-)\big) \tag{8}$$

The bets are chosen jointly to maximize the value of the next state:

$$\lambda^* = \operatorname*{argmax}_{\lambda_i \in \{\text{accept, reject}\}} \mathbb{E}\sum_{i=1}^{L} V\big(W_i(1 + \lambda_i^\top Z_i(Y_i))\big) \qquad \text{reject} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \text{accept} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

4.2 Entropy-Bound uses the linear utility

Consider a linear bettor which does not maintain wealth: $V(W) = W$ and $W_i = 1$. In choosing to accept or reject, it chooses between keeping the value 1 or instead taking $\mathbb{E}R_i$. The likelihood ratio has mean 1 — its risk is not visible with the linear utility — so the bettor chooses to accept exactly when $e^{\gamma - C_i} \geq 1$, i.e. $C_i \leq \gamma$. Thus, EB is equivalent to the stateless linear bettor.

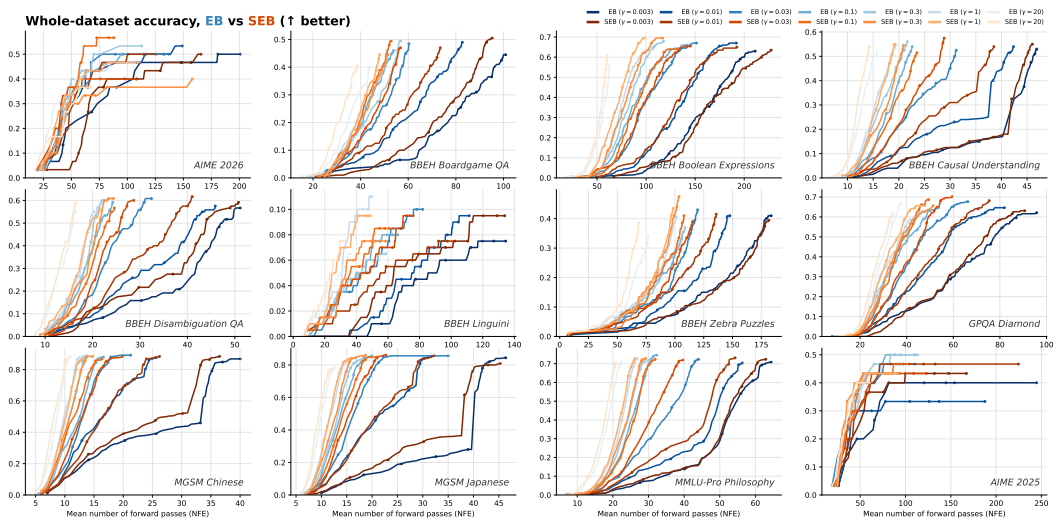
4.3 Steady Entropy-Bound uses the Kelly (logarithmic) utility

SEB adopts the logarithmic terminal utility $u(W) = \log W$, i.e., the Kelly criterion. This utility is classically motivated by maximizing long-run multiplicative wealth growth (Kelly, 1956). Its induced value function is $V(W) = \log W + a$ for an additive constant a; because this constant does not affect the choice between bets, we just normalize $a = 0$. SEB's gain g_i from accepting mirrors Eq. (8):

$$g_i = \log W_i + \gamma - C_i - \mathrm{KL}\big(\mathrm{chop}(p_i, T_i^-) \,\big\|\, \mathrm{chop}(p_i^-, T_i^-)\big)\,. \tag{9}$$

Accordingly, SEB sets $\lambda_i = \text{accept}$ iff $g_i \geq 0$.

In EB, γ only controls the current acceptance threshold. In SEB, it also enters every accepted wealth update, so it sets the break-even quality of a trajectory over time. Increasing γ allows more entropic or somewhat less stable trajectories to sustain wealth; decreasing it requires consistently lower entropy and/or churn.



Final mean accuracy +0.50%

0.0%	-3.3%	+6.0%	-2.0%	+0.5%	-2.0%	+2.0%	0.0%
-10.0%	+6.7%	-1.5%	+3.5%	-1.5%	-2.0%	+5.0%	-2.5%
-13.3%	0.0%	-5.0%	-1.5%	+1.5%	+5.5%	-1.5%	+0.5%
-3.3%		+0.5%		+5.0%		+2.5%	
+2.5%	+4.2%	+2.0%	-2.0%	-1.5%	+0.5%	+1.0%	+3.5%
-0.8%	-2.5%	-0.5%	+0.5%	-4.0%	+5.5%	+2.5%	+4.0%
+0.8%	-3.3%	+1.0%	-1.5%	+10.5%	+3.0%	+2.5%	+3.5%
+4.2%		+1.0%		+4.5%		-5.1%	
+1.6%	+0.8%	-3.6%	+0.4%	+1.6%	+2.8%	+3.3%	+13.3%
-0.8%	-1.2%	+0.4%	-0.8%	-0.4%	-2.2%	-3.3%	-3.3%
-0.4%	-2.0%	+2.4%	0.0%	+0.6%	0.0%	-6.7%	-3.3%
-2.0%		-0.8%		-0.4%		+3.3%	

Final mean NFE -3.99%

-17.4%	-16.1%	-5.6%	-11.2%	+7.7%	+0.3%	-1.8%	-9.0%
-20.9%	-8.8%	-5.9%	-1.8%	-4.0%	-6.0%	-7.5%	+4.1%
+40.2%	+1.4%	-9.5%	-5.6%	-0.8%	-8.5%	-3.3%	-0.5%
+8.2%		-12.1%		-3.6%		-9.1%	
-0.7%	-10.6%	-0.5%	-8.1%	-1.1%	-8.0%	-5.4%	-8.0%
-11.4%	+1.4%	-7.5%	-4.6%	-3.5%	+0.4%	-10.1%	-7.9%
+6.0%	0.0%	+0.1%	-2.1%	-4.7%	-2.7%	+5.0%	+4.8%
-0.3%		-7.4%		-11.0%		-9.9%	
-8.8%	+2.3%	-2.3%	+1.6%	-2.4%	-3.6%	-31.7%	+19.7%
-6.5%	-1.6%	-35.1%	+4.5%	-9.9%	-1.4%	+11.6%	-6.0%
+6.4%	-1.7%	+2.4%	-6.1%	-3.3%	-2.2%	-17.3%	+19.3%
-5.2%		-5.5%		-6.6%		+0.4%	

Tokens per forward pass +3.84%

+14.7%	+4.2%	+4.5%	+10.0%	+0.2%	+4.3%	+1.7%	+9.2%
+17.1%	+7.3%	+6.3%	+1.7%	+5.7%	+2.7%	+7.2%	-2.5%
-5.9%	+7.9%	+12.0%	+6.5%	-0.7%	+5.2%	+4.6%	+1.5%
+0.9%		+11.1%		+1.2%		+10.3%	
+2.7%	+9.3%	+1.4%	+7.1%	+2.2%	+6.2%	+2.6%	+5.7%
+9.7%	-1.1%	+6.8%	+4.0%	+3.4%	+1.3%	+9.4%	+3.9%
-3.2%	-0.3%	+1.4%	+1.1%	+4.3%	+5.2%	-1.8%	-1.4%
+2.3%		+6.9%		+12.0%		+8.6%	
+8.6%	0.0%	+6.3%	+1.3%	+1.1%	+4.3%	+16.5%	-10.1%
+5.1%	+3.7%	+25.4%	-4.5%	+10.8%	+1.3%	-9.1%	+2.7%
-5.2%	+0.6%	-3.1%	+3.2%	+2.6%	+0.9%	-1.5%	-8.2%
+4.5%		+5.8%		+6.3%		+2.5%	

Figure 4: **Top:** Comparison of EB and SEB on 12 benchmarks. In each plot, we limit the compute budget (x-axis) and then assess the accuracy achieved at that budget (y axis). Each line represents one run of an algorithm. At each γ, SEB tends to hit a given accuracy faster than EB. **Bottom:** Shared-γ comparisons. The seven cells give the relative SEB change from EB at the seven tested values of γ. In summary, SEB achieves comparable quality as EB while using 4% less compute.

4.4 SEB Improves the Accuracy–Compute Tradeoff

We evaluate EB and SEB with DiffusionGemma on 12 benchmarks. For each benchmark, we run both samplers on the same examples at seven matched values of their shared parameter γ. Thus, each SEB run is directly matched with an EB one. To measure how quickly each method reaches correct solutions, we construct compute–accuracy curves. Suppose answering example i takes c_i forward passes and results in score s_i. We draw the curve by sweeping the compute limit $L > 0$:

$$x(L) = \frac{1}{n}\sum_{i=1}^{n} \min(c_i, L), \qquad y(L) = \frac{1}{n}\sum_{i=1}^{n} s_i \, \mathbf{1}\{c_i \le L\}.$$

An unfinished example therefore consumes L forward passes but remains in the whole-dataset accuracy denominator with score zero. The right endpoint of each curve recovers the ordinary final mean NFE and final accuracy. Figure 4 shows that, at matched γ, SEB generally reaches a given level of accuracy with fewer denoiser evaluations. Over all matched example–γ pairs, SEB improves final mean accuracy by 0.50 percentage points, reduces mean NFE by 3.99%, and increases output tokens processed per forward pass by 3.84% relative to EB.

5 Related Work

Adaptive speculative decoding. Speculative decoding verifies draft-model proposals with a target model while preserving the target distribution (Leviathan et al., 2023; Chen et al., 2023). Many methods adapt the amount or structure of speculation to the current opportunity. SpecDec++ and DISCO learn when to stop drafting; AdaEDL uses current draft entropy; BanditSpec learns configurations

online from realized rewards; SmartSpec adapts speculation length to serving load and speculation accuracy; and tree/candidate methods change which candidates are proposed or verified (Huang et al., 2025; Mamou et al., 2024; Agrawal et al., 2024; Hou et al., 2025; Liu et al., 2024; Cai et al., 2024; Li et al., 2024; Wu et al., 2025). DSpark likewise allocates verification from predicted prefix survival and profiled execution cost (Cheng et al., 2026). These controllers track the current acceptance opportunity, system load, or the performance of candidate configurations. DStreak instead tracks each request's realized generation rate and uses its α-fair marginal value inside DSpark's existing scheduling problem.

Adaptive diffusion sampling. Diffusion-language-model samplers also adapt parallelism using current confidence or information. Fast-dLLM unmasks tokens above a confidence threshold, adaptive parallel decoding varies simultaneous sampling through an auxiliary-model criterion, and dilated scheduling controls parallel groups with a joint-entropy bound (Wu et al., 2026; Israel et al., 2025; Luxembourg et al., 2026). Other methods use trajectory history: Saber changes unmasking aggressiveness as confidence evolves and backtracks when it deteriorates; RemeDi learns to remask low-confidence tokens; and SchED makes early-exit thresholds depend on diffusion progress (Dong et al., 2025; Huang et al., 2026; Mohamed et al., 2026). These methods adapt to confidence, token quality, or elapsed diffusion progress. However, they are not necessarily applicable to uniform-renoising diffusion, as in DiffusionGemma. SEB instead maintains one wealth state per position that records how consecutive model draws have settled since the last uniform renoise, and uses its state value together with the current marginal to decide whether to sample again or reset. Complementary theory characterizes globally optimal masked-diffusion schedules under distributional knowledge (Chen et al., 2026); Target correction does not solve this scheduling problem.

SLO-aware LLM serving. At the broader serving layer, goodput-oriented systems already make latency objectives explicit. DistServe optimizes capacity subject to TTFT/TPOT requirements; more recent schedulers use SLO slack or latency budgets to adapt batching, scheduling, or cache management (Zhong et al., 2024; Wang and Yang, 2026; Adnan et al., 2026). These systems represent progress toward a service objective, but act on request scheduling and resource management rather than speculative verification itself. DStreak places an analogous target-progress signal inside the token-level controller that decides how much target-model verification each active request receives.

Utility-based online resource allocation. Concave utilities have long been used to translate long-run fairness objectives into local resource-allocation rules in communication networks (Kelly, 1997; Kelly et al., 1998; Mo and Walrand, 2000). Related Lyapunov optimization methods use queue or deficit state to convert long-run stochastic objectives and constraints into simple per-round control decisions (Neely, 2010). Target correction shares the idea that accumulated state should modify a local decision, but starts from an existing inference controller: it constructs a multiplicative wealth process for that controller and uses the value of wealth to correct its original decision rule, without introducing a full network or environment model.

Testing by betting and stochastic control. Testing by betting uses multiplicative wealth to accumulate statistical evidence (Shafer, 2021; Ramdas et al., 2023). Kelly betting chooses exposure to maximize expected log wealth (Kelly, 1956); continuous-time portfolio control relates utility curvature to optimal risk exposure (Merton, 1969; 1971). Target correction uses the same mathematics for a different purpose: wealth is an internal state measuring target-relevant progress, whose value modifies an existing inference controller.

6 Conclusion

This paper shows that some inference-time planning problems can admit small, targeted improvements without redesigning the underlying controller. In our two applications, adding persistent, target-aware state to otherwise local decision rules produced concrete benefits: DStreak improved request-level goodput robustness under skewed workloads, while Steady Entropy-Bound reduced denoising compute at comparable or better fidelity. These modifications do not solve the underlying long-horizon problems, but they show that diagnosing a controller's myopia can suggest simple, effective ways to improve it. Exploring such applications to scheduling, routing, sampling, and other inference-time problems is a promising direction for future work.

AI USE STATEMENT

Generative AI tools assisted with coding (LaTeX for the paper, and Python for the experiments), mathematics (stating lemmas and generating proofs), proofreading, and prospecting the literature for interesting applications of target correction. Though we verified all results, we nonetheless found AI helpful for red-teaming many of our initial claims. We did not find current AI tools particularly helpful for writing our prose. AI was not responsible for coming up with the idea of target correction, refining algorithms, or designing experiments. The authors take complete responsibility for the final manuscript, its experimental results, and its scientific claims.

ETHICS STATEMENT

The authors do not feel this paper poses any substantive ethical concerns.

REPRODUCIBILITY STATEMENT

Appendix B.2 describes the experimental details for speculative decoding. Appendix C.1 does the same for diffusion sampling. Both experiments involved modifying existing inference controllers in vLLM. The controller modifications and experiment code will be fully open-sourced.

REFERENCES

Talor Abramovich, Maor Ashkenazi, Izzy Putterman, Benjamin Chislett, Tiyasa Mitra, Bita Darvish Rouhani, Ran Zilberstein, and Yonatan Geifman. SPEED-bench: A unified and diverse benchmark for speculative decoding. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Rl2uQlCoQX.

Muhammad Adnan, Rohan Mahapatra, Prashant J. Nair, Daniel Berger, Pantea Zardoshti, Rodrigo Fonseca, and Esha Choukse. Cascade: Exploiting SLO-aware latency budget for fair and high goodput LLM inference serving. *arXiv preprint arXiv:2608.06557*, 2026. doi: 10.48550/arXiv.2608.06557. URL https://arxiv.org/abs/2608.06557.

Sudhanshu Agrawal, Wonseok Jeon, and Mingu Lee. AdaEDL: Early draft stopping for speculative decoding of large language models via an entropy-based lower bound on token acceptance probability. In *Proceedings of The 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pages 355–369. PMLR, 2024. URL https://proceedings.mlr.press/v262/agrawal24a.html.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas S. Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *Advances in Neural Information Processing Systems*, volume 38, pages 55981–56007. Curran Associates, Inc., 2025. doi: 10.52202/085713-1874. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/510e22f4a2da5212a64f1591736e2eaf-Abstract-Conference.html.

Sem C. Borst, Neil S. Walton, and Bert Zwart. Network iso-elasticity and weighted α-fairness. *Performance Evaluation*, 70(11):995–1000, 2013. doi: 10.1016/j.peva.2013.07.003. URL https://doi.org/10.1016/j.peva.2013.07.003.

Leo Breiman. Optimal gambling systems for favorable games. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 65–78, Berkeley, California, 1961. University of California Press.

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*, volume 235 of *Proceedings of Machine Learning Research*, pages 5209–5235. PMLR, 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023. doi: 10.48550/arXiv.2302.01318. URL https://arxiv.org/abs/2302.01318.

Sitan Chen, Kevin Cong, and Jerry Li. Optimal inference schedules for masked diffusion models. In *Proceedings of Thirty Ninth Conference on Learning Theory*, volume 336 of *Proceedings of Machine Learning Research*, pages 1279–1311. PMLR, 2026. URL https://proceedings.mlr.press/v336/chen26e.html.

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. doi: 10.48550/arXiv.2607.05147. URL https://arxiv.org/abs/2607.05147.

Thomas M. Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–29, 1991. doi: 10.1111/j.1467-9965.1991.tb00002.x. URL https://doi.org/10.1111/j.1467-9965.1991.tb00002.x.

DiffusionGemma Team. DiffusionGemma technical report. *arXiv preprint arXiv:2608.00146*, 2026. doi: 10.48550/arXiv.2608.00146. URL https://arxiv.org/abs/2608.00146.

Yihong Dong, Zhaoyu Ma, Xue Jiang, Zhiyuan Fan, Jiaru Qian, Yongmin Li, Jianha Xiao, Zhi Jin, Rongyu Cao, Binhua Li, Fei Huang, Yongbin Li, and Ge Li. Saber: An efficient sampling with adaptive acceleration and backtracking enhanced remasking for diffusion language model. *arXiv preprint arXiv:2510.18165*, 2025. doi: 10.48550/arXiv.2510.18165. URL https://arxiv.org/abs/2510.18165.

Wendell H. Fleming and H. Mete Soner. *Controlled Markov Processes and Viscosity Solutions*, volume 25 of *Stochastic Modelling and Applied Probability*. Springer, New York, 2 edition, 2006. doi: 10.1007/0-387-31071-1. URL https://doi.org/10.1007/0-387-31071-1.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Yunlong Hou, Fengzhuo Zhang, Cunxiao Du, Xuan Zhang, Jiachun Pan, Tianyu Pang, Chao Du, Vincent Y. F. Tan, and Zhuoran Yang. BanditSpec: Adaptive speculative decoding via bandit algorithms. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 24045–24079. PMLR, 2025. URL https://proceedings.mlr.press/v267/hou25h.html.

Kaixuan Huang, Xudong Guo, and Mengdi Wang. SpecDec++: Boosting speculative decoding via adaptive candidate lengths. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Y131N9fUbU.

Zemin Huang, Yuhang Wang, Zhiyang Chen, and Guo-Jun Qi. Don't settle too early: Self-reflective remasking for diffusion language models. In *International Conference on Learning Representations*, volume 2026, pages 56998–57026, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5d8bc24ef15b7a808b6dc403c01583e2-Abstract-Conference.html.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion LLMs via adaptive parallel decoding. In *Advances in Neural Information Processing Systems*, volume 38, pages 52870–52888. Curran Associates, Inc., 2025. doi: 10.52202/085713-1763. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4c7d31b2da05a17f15101db5b37a1e24-Abstract-Conference.html.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K. Jain, Virginia Aglietti, Disha Jindal, Peter Chen, Nishanth Dikkala, Gladys Tyen, Xin Liu, Uri Shalit, Silvia Chiappa, Kate Olszewska, Yi Tay, Vinh Q. Tran, Quoc V. Le, and Orhan Firat. BIG-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26473–26501, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1285. URL https://aclanthology.org/2025.acl-long.1285/.

F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3): 237–252, 1998. doi: 10.1057/palgrave.jors.2600523.

Frank Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8(1):33–37, 1997. doi: 10.1002/ett.4460080106.

John L. Kelly, Jr. A new interpretation of information rate. *The Bell System Technical Journal*, 35(4): 917–926, July 1956. doi: 10.1002/j.1538-7305.1956.tb03809.x. URL https://doi.org/10.1002/j.1538-7305.1956.tb03809.x.

Harold J. Kushner and Paul Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time*, volume 24 of *Stochastic Modelling and Applied Probability*. Springer, New York, 2 edition, 2001. doi: 10.1007/978-1-4613-0007-6.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626. Association for Computing Machinery, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

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

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

Xiaoxuan Liu, Cade Daniel, Langxiang Hu, Woosuk Kwon, Zhuohan Li, Xiangxi Mo, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Optimizing speculative decoding for serving large language models using goodput. *arXiv preprint arXiv:2406.14066*, 2024. doi: 10.48550/arXiv.2406.14066. URL https://arxiv.org/abs/2406.14066.

Xiaoxuan Liu, Jiaxiang Yu, Jongseok Park, Ion Stoica, and Alvin Cheung. Speculative decoding: Performance or illusion? In A. Chowdhery and Z. Jia, editors, *Proceedings of Machine Learning and Systems*, volume 8, pages 1719–1741. MLSys, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/file/554e056fe2b6d9fd27ffcd3367ae1267-Paper-Conference.pdf.

Omer Luxembourg, Haim H. Permuter, and Eliya Nachmani. Plan for speed: Dilated scheduling for masked diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=DbzAsMRNhy.

Jonathan Mamou, Oren Pereg, Daniel Korat, Moshe Berchansky, Nadav Timor, Moshe Wasserblat, and Roy Schwartz. Dynamic speculation lookahead accelerates speculative decoding of large language models. In *Proceedings of The 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pages 456–467. PMLR, 2024. URL https://proceedings.mlr.press/v262/mamou24a.html.

Mathematical Association of America. MAA invitational competitions: American invitational mathematics examination, 2026. URL https://maa.org/maa-invitational-competitions/.

Robert C. Merton. Lifetime portfolio selection under uncertainty: The continuous-time case. *The Review of Economics and Statistics*, 51(3):247–257, 1969. doi: 10.2307/1926560. URL https://doi.org/10.2307/1926560.

Robert C. Merton. Optimum consumption and portfolio rules in a continuous-time model. *Journal of Economic Theory*, 3(4):373–413, 1971. doi: 10.1016/0022-0531(71)90038-X. URL https://doi.org/10.1016/0022-0531(71)90038-X.

Jeonghoon Mo and Jean C. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, 2000. doi: 10.1109/90.879343. URL https://doi.org/10.1109/90.879343.

Amr Mohamed, Yang Zhang, Michalis Vazirgiannis, and Guokan Shang. Fast-decoding diffusion language models via progress-aware confidence schedules. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 35793–35807, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1782. URL https://aclanthology.org/2026.findings-acl.1782/.

Michael J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Synthesis Lectures on Communication Networks. Morgan & Claypool Publishers, 2010. ISBN 9781608454556. doi: 10.2200/S00271ED1V01Y201006CNT007.

Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023. doi: 10.1214/23-STS894. URL https://doi.org/10.1214/23-STS894.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Glenn Shafer. Testing by betting: A strategy for statistical and scientific communication. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 184(2):407–431, 2021. doi: 10.1111/rssa.12647. URL https://doi.org/10.1111/rssa.12647.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=fR3wGCk-IXp.

Jean Ville. *Étude critique de la notion de collectif*. PhD thesis, Faculté des sciences de Paris, 1939. URL https://www.numdam.org/item/THESE_1939__218__1_0/.

Qipeng Wang and Zhendong Yang. Taming request imbalance: SLO-aware scheduling for disaggregated LLM inference. *arXiv preprint arXiv:2605.02329*, 2026. doi: 10.48550/arXiv.2605.02329. URL https://arxiv.org/abs/2605.02329.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pages 95266–95290, 2024. doi: 10.52202/079017-3018.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3Z3Is6hnOT.

Zhaoxuan Wu, Zijian Zhou, Arun Verma, Alok Prakash, Daniela Rus, and Bryan Kian Hsiang Low. TETRIS: Optimal draft token selection for batch speculative decoding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33329–33345, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1598. URL https://aclanthology.org/2025.acl-long.1598/.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 193–210, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin.

A Target Correction Technical Details

The proofs in this section are standard consequences of diffusion approximation and stochastic dynamic programming; we include them for completeness and to fix notation (Kushner and Dupuis, 2001; Fleming and Soner, 2006).

A.1 Discrete-to-continuous approximation

The continuous-time model in Section 2 is a local approximation to the discrete betting game. Let one discrete round correspond to a time interval of length δ, and write its return as $Z_t^{(\delta)}$. Suppose that, conditionally on the current state,

$$\mathbb{E}[Z_t^{(\delta)}] = \mu\,\delta + o(\delta), \qquad \mathrm{Cov}(Z_t^{(\delta)}) = \Sigma\,\delta + o(\delta),$$

with the usual Lindeberg condition ruling out macroscopic jumps. Because the bet set Λ is compact, the relative wealth increment $\lambda_t^\top Z_t^{(\delta)}$ therefore has conditional drift $\lambda_t^\top \mu\,\delta + o(\delta)$ and conditional variance $\lambda_t^\top \Sigma \lambda_t\,\delta + o(\delta)$. Under the standard diffusion approximation, the rescaled discrete wealth process converges to

$$\frac{dW_t}{W_t} = \lambda_t^\top \mu\,dt + \lambda_t^\top \Sigma^{1/2} dB_t,$$

which is Eq. (1). Conversely, an Euler–Maruyama step of this diffusion has the same drift and covariance to first order in δ. Thus the continuous model preserves precisely the local first- and second-order return information used in the HJB calculation.

The stopping rule has the same limit. If the probability of termination during one discrete interval is $q_\delta = \delta/\rho + o(\delta)$ and T_δ is the resulting geometric number of rounds, then for any $s \geq 0$,

$$\Pr(\delta T_\delta > s) = (1 - q_\delta)^{\lfloor s/\delta \rfloor} \longrightarrow e^{-s/\rho}.$$

Hence the rescaled geometric clock converges to an independent exponential clock with hazard $1/\rho$. The continuous-time problem should therefore be viewed as a stationary local approximation to the original betting game, rather than as a model of the full future inference trajectory.

A.2 Proof of the value-function lemma

Proof of Lemma 1. Fix $W > 0$ at which V is twice differentiable. The memorylessness of the exponential stopping time gives the dynamic-programming identity, for any sufficiently small $h > 0$,

$$V(W) = \sup_\pi \mathbb{E}\left[\int_0^h \frac{1}{\rho} e^{-t/\rho} u(W_t)\,dt + e^{-h/\rho} V(W_h) \,\middle|\, W_0 = W\right].$$

Over an infinitesimal interval, only the current bet matters. For a fixed $\lambda \in \Lambda$, the diffusion satisfies $\mathbb{E}[W_h - W] = W\lambda^\top \mu\,h + o(h)$ and $\mathbb{E}[(W_h - W)^2] = W^2 \lambda^\top \Sigma \lambda\,h + o(h)$. A second-order expansion of V at W therefore gives

$$\mathbb{E}[V(W_h)] = V(W) + h\left[WV'(W)\lambda^\top \mu + \frac{1}{2} W^2 V''(W) \lambda^\top \Sigma \lambda\right] + o(h).$$

Meanwhile, continuity of u and $W_t \to W$ as $t \downarrow 0$ give $\mathbb{E}[\int_0^h \rho^{-1} e^{-t/\rho} u(W_t)\,dt] = h\rho^{-1} u(W) + o(h)$, while $e^{-h/\rho} = 1 - h/\rho + o(h)$. Substituting these expansions into the dynamic-programming identity, subtracting $V(W)$, dividing by h, and letting $h \downarrow 0$ yields

$$0 = \sup_{\lambda \in \Lambda} \left\{ WV'(W)\lambda^\top \mu + \frac{1}{2} W^2 V''(W) \lambda^\top \Sigma \lambda \right\} + \frac{1}{\rho}\big[u(W) - V(W)\big],$$

which is Eq. (2).

Now additionally suppose that $V''(W) < 0$. Since $\Sigma \succ 0$, the Hamiltonian is then strictly concave in λ. Its unconstrained maximizer is obtained from the first-order condition $WV'(W)\mu + W^2V''(W)\Sigma\lambda = 0$, giving $\lambda^*(W) = -V'(W)[WV''(W)]^{-1}\Sigma^{-1}\mu$. If this point lies in the interior of Λ, it is also the constrained maximizer. Substituting it into the Hamiltonian gives

$$\sup_{\lambda\in\Lambda}\left\{WV'(W)\lambda^\top\mu + \frac{1}{2}W^2V''(W)\lambda^\top\Sigma\lambda\right\} = -\frac{1}{2}\frac{V'(W)^2}{V''(W)}\mu^\top\Sigma^{-1}\mu.$$

Substitution into Eq. (2) and rearrangement therefore give

$$V(W) - u(W) = \frac{\rho}{2}\left(\mu^\top\Sigma^{-1}\mu\right)\left(-\frac{V'(W)^2}{V''(W)}\right),$$

which is Eq. (3). □

B DSTREAK TECHNICAL DETAILS

B.1 CRRA NORMALIZATION

Let $\chi = \mu^\top\Sigma^{-1}\mu$. For $\alpha > 0$ and $\alpha \neq 1$, take $V_\alpha(W) = W^{1-\alpha}/(1-\alpha)$, so $V'_\alpha(W) = W^{-\alpha}$ and $-V'_\alpha(W)^2/V''_\alpha(W) = W^{1-\alpha}/\alpha$. Assuming the interior condition of Lemma 1 holds, Eq. (3) is satisfied by $u_\alpha(W) = c_\alpha V_\alpha(W)$ with

$$c_\alpha = 1 - \frac{\rho(1-\alpha)}{2\alpha}\chi.$$

When $c_\alpha > 0$, this is a positive rescaling of CRRA utility, so the induced value can be normalized so that $V'_\alpha(W) = W^{-\alpha}$.

For $\alpha = 1$, take $u_1(W) = \log W$ and write $V_1(W) = \log W + C$. Since $-V'_1(W)^2/V''_1(W) = 1$, Eq. (3) gives $C = \rho\chi/2$, and hence $V'_1(W) = 1/W$. The endpoint $\alpha = 0$ is handled separately by the linear normalization $V_0(W) = W$, which recovers DSpark.

B.2 EXPERIMENT DETAILS

Data and mixtures. We constructed each workload from 512 requests drawn exclusively from the high-entropy SPEED-Bench 2K split. To create controlled scheduling pressure, we deterministically selected 0%, 5% (26 requests), or 10% (51 requests) of low-entropy prompts. The selected sets were hash-ranked and nested, so the 5% set was contained in the 10% set. This intervention affected only the allocator's predictions; verifier probabilities and actual token acceptance were unchanged. Runs were repeated and averaged 3 times.

Model and inference setup. We used Gemma 4 12B with the published DSpark drafter in vLLM 0.29.0. Inference used a batch size of 128 and a client concurrency of 512. Each condition used 512 warm-up and 512 measured requests, generated up to 2,048 output tokens at temperature zero with prefix caching disabled. The reported comparison uses $\alpha = 0$ (DSpark-equivalent allocation) and $\alpha = 1$ (proportional fairness), with per-request throughput measured by the server-side DStreak controller.

C STEADY ENTROPY-BOUND TECHNICAL DETAILS

C.1 EXPERIMENT DETAILS

Benchmarks. We evaluate 12 complete benchmark slices (2,377 prompts): AIME I/II 2025 and 2026 (Mathematical Association of America, 2026), GPQA Diamond (Rein et al., 2024), Chinese and Japanese MGSM (Shi et al., 2023), MMLU-Pro Philosophy (Wang et al., 2024), and six BBEH tasks—Boardgame QA, Boolean Expressions, Causal Understanding, Disambiguation QA, Linguini, and Zebra Puzzles (Kazemi et al., 2025). We use each task's native exact-answer or multiple-choice scorer.

Implementation. SEB modifies only the Python DiffusionGemma sampler in vLLM v0.29.0 (Kwon et al., 2023). We serve the dynamic-FP8 `RedHatAI/diffusiongemma-26B-A4B-it-FP8-dynamic` checkpoint on four NVIDIA L40S GPUs as four replicas with tensor parallelism one. We retain the standard DiffusionGemma generation settings, including its 256-token canvas and 48-pass cap (DiffusionGemma Team, 2026).