GREEDY MULTI-PATH BLOCK VERIFICATION FOR FASTER DECODING IN SPECULATIVE SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

The goal of L-step speculative decoding is to accelerate autoregressive decoding of a target model by using a cheaper draft model to generate a candidate path of L tokens. Based on a verification algorithm involving target and draft model probabilities, a prefix of the candidate sequence is accepted, and an additional correction token is sampled from a residual distribution to ensure that the final output adheres to the target distribution. While standard speculative decoding uses a verification algorithm which is independent at each token on the path, a recent extension called block verification uses a joint condition involving all sampled on-path probabilities. Block verification (BV) was shown to be optimal over all verification algorithms which use only on-path probabilities, improving on standard speculative decoding. In this work, we first show that block verification is optimal even over verification algorithms that use off-path probabilities, by constructing an information-agnostic linear program (LP). Further, we can extend our LP to the setting where the draft model samples multiple candidate paths, and use it to construct a natural class of multi-path block verification generalizations. While computing the optimal algorithm in this class is not tractable, by considering a stricter class of greedy algorithms, we can formulate an efficient method called greedy multi-path block verification (GBV). Empirically, GBV can improve block efficiency by over 30% and reduce decoding walltimes by over 15% relative to BV. At temperature 0.2 on Llama-3 70B, GBV can improve the end-to-end decoding throughput of SOTA multi-path verification methods by over 15%.

1 INTRODUCTION

Most large language model (LLM) families, such as Qwen (Bai et al., 2023), GPT (Radford et al., 2018; 2019; Brown et al., 2020; OpenAI, 2023), and Llama (Grattafiori et al., 2024) employ autoregressive decoding. When inference is performed with these transformer-based architectures on GPUs, end-to-end latency is dominated by memory bandwidth rather than compute (Fu et al., 2024).

Speculative sampling (Chen et al., 2023; Leviathan et al., 2023) aims to reduce such punitive costs when sampling from a large *target model*. This procedure autoregressively decodes a candidate sequence from a cheaper *draft model*, performs a target model forward pass over the candidate sequence to obtain target distribution values, and then alters the sequence through a randomized *verification algorithm* so that the resulting output matches the target model distribution. As inference is bandwidth-bound, the target forward pass over the candidate sequence has negligible overhead over a standard forward pass. Thus, speculative sampling decodes multiple tokens per target model call and speeds up inference without impacting downstream performance.

In **standard speculative sampling**, the draft model proposes a length L draft block, and the verification algorithm independently accepts or rejects each token based on target and draft model probabilities. The longest prefix without rejections is chosen, and an extra token is sampled from a residual, ensuring that at least one token is generated. While this procedure improves decoding efficiency, it suffers when early tokens have low acceptance. If the first token is usually rejected due to a poor draft, then even if subsequent tokens are always accepted, there is no speedup.

To overcome this early-token bottleneck, recent work on **block verification (BV)** (Sun et al., 2024b) replaces independent token-wise verification with independent prefix-wise verification, and selects the longest accepted prefix. An extra token is still sampled, but the residual distribution is altered

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.

to ensure the output follows the target distribution. This ensures that more tokens can be accepted when earlier tokens are likely rejected. Block verification is 5-8% faster than standard speculative sampling. In fact, it is provably optimal over all verification algorithms that only use target and draft probability distributions along the draft block, including standard speculative sampling.

In this paper, we frame the optimal verification algorithm as a solution to a linear program (LP), and extend the LP to the setting of *multiple* i.i.d. generated draft paths. Our contributions are as follows.

- **Single-Path Information-Agnostic LP.** We develop an LP that encodes feasible speedups from verification algorithms that return a candidate sequence prefix and an additional token, but now have access to the *full joint* target and draft distributions, not just on-path values. We show that block verification is still optimal in this setting. That is, the prefix output requirement is the bottleneck to improving block efficiency, not off-path information.
- **Multi-Path Information-Agnostic LP.** We extend the single-path LP to the setting where the draft model i.i.d. samples $K > 1$ candidate sequences. Unlike $K = 1$, we find that knowledge of off-path target and draft distributions can improve decoding efficiency. We show that the optimal verification algorithm in this setting is to probabilistically select one of the K paths, and run block verification on that single path with a skewed draft.
- **Greedy Approximation Schemes.** The optimal solution to the multi-path LP is an infeasible nonlinear optimization problem over joint target and draft distributions. Thus, we explore a class of greedy approximations: algorithms which globally rank all possible candidate sequences, select the highest-ranking of the K paths, and run block verification on that path. We show these arise from a greedy polymatroid algorithm in the multi-path LP.
- **Greedy Multi-Path Block Verification (GBV).** Many greedy approximation schemes require the full joint target and draft distributions. However, when the global ranking is tree-based, we only require on-path values. We implement a set of tree-based heuristics with theoretical justification, and see achieve wall-clock speedups over 15% relative to BV.

2 Background

We review speculative sampling in the case of one draft path, and then cover multi-path extensions.

2.1 Single-Path

Denote the target model by M_p, and the draft model by M_q, which share a common vocabulary $\mathcal{V}$. We use the sequence notation $a_{1:k} = (a_1, \ldots, a_k)$ for tokens $a_1, \ldots, a_k \in \mathcal{V}$ and the inclusive slicing notation $a_{i:j} = (a_i, \ldots, a_j)$, with the empty sequence convention $a_{i,j} = \emptyset$ for $j < i$. Given a context $\boldsymbol{c}$, the target and draft models induce next-token distribution $p(\cdot|\boldsymbol{c})$ and $q(\cdot|\boldsymbol{c})$, respectively. In autoregressive sampling, these induce next-k-token distributions through conditional factorization:

$$q_k(a_{1:k}|\boldsymbol{c}) = \prod_{i=1}^{k} q(a_i|\boldsymbol{c}, a_{1:i-1}), \quad p_k(a_{1:k}|\boldsymbol{c}) = \prod_{i=1}^{k} p(a_i|\boldsymbol{c}, a_{1:i-1}). \tag{1}$$

In L-step speculative sampling, we autoregressively sample a draft block of L tokens from the draft distribution q_L and call the target model M_p on this block *once*, to obtain target and draft next-token probabilities along the block. Through a randomized rule called the **verification algorithm**, we accept the first $\tau \in \{0, \ldots, L\}$ of these L tokens and sample a correction token, such that this output follows the M_p distribution. The **block efficiency** $\mathbb{E}[\tau + 1]$ is the average decoded tokens per M_p call; when the target and draft models coincide, it achieves its maximum of $L + 1$. When drafting is cheap and L is not large, block efficiency is an accurate indicator of walltime speedup.

Speculative sampling. The original schemes of Chen et al. (2023); Leviathan et al. (2023) first generate a draft block $a_{1:L} \sim q_L(\cdot|\boldsymbol{c})$. They *independently* accept each token a_i with probability $\min(1, p(a_i|a_{1:i-1})/q(a_i|a_{1:i-1}))$, and τ is the maximum value such that $a_{1:\tau}$ consists of only accepted tokens. Essentially, they choose the longest prefix of acceptances. Finally, they sample an additional *correction token* $y \sim p_{\text{res}}^{\text{token}}(\cdot|\boldsymbol{c}, a_{1:i})$ from a residual distribution,

$$p_{\text{res}}^{\text{token}}(\cdot|\boldsymbol{c}, a_{1:i}) \propto \max\{p(\cdot|\boldsymbol{c}, a_{1:i}) - q(\cdot|\boldsymbol{c}, a_{1:i}), 0\}, \tag{2}$$

as long as $\tau < L$. If $\tau = L$, they just sample $y \sim p(\cdot|\boldsymbol{c}, a_{1:L})$ directly from the target. This ensures that the final output $(a_{1:\tau}, y)$ matches the target distribution.

Draft extensions. Some recent works have improved upon block efficiency in speculative sampling by altering the drafting phase, through retrieval or cascading (He et al., 2023; Chen et al., 2024), hierarchical drafting (Sun et al., 2024a), distillation (Zhou et al., 2023; Liu et al., 2023), layer skipping (Zhang et al., 2023; Elhoushi et al., 2024), or multi-token heads (Gloeckle et al., 2024; Samragh et al., 2025)). We instead focus on methods which improve block efficiency by altering the verification algorithm, and these can be integrated with the above works for further gains.

Tree verification. Monte Carlo tree verification from Hu & Huang (2024) provably improves the block efficiency of standard speculative sampling. While the underlying tree structure might suggest this is a multi-path method, we emphasize that it is a single-path algorithm: Section 5 in Hu & Huang (2024) specifies a linear chain of draft tokens. This means it is provably worse than block verification: see the third paragraph of Section 7 in Sun et al. (2024b).

Block verification (BV). To improve block efficiency in the verification stage, Sun et al. (2024b) relax the token-independence assumption for acceptance in standard speculative sampling. They sample the draft block $a_{1:L} \sim q_L(\cdot|\boldsymbol{c})$, and define on-path weights w from $w(\emptyset|\boldsymbol{c}) = 1$:

$$w(a_{1:i}|\boldsymbol{c}) = \min\left\{1, \frac{w(a_{1:i-1}|\boldsymbol{c})p(a_i|a_{1:i-1})}{q(a_i|a_{1:i-1})}\right\}. \tag{3}$$

They independently accept each *prefix* $a_{1:i}$ with probability

$$r(a_{1:i}|\boldsymbol{c}) = \sum_{x\in\mathcal{V}} \max\{p(x|\boldsymbol{c}, a_{1:i}) - q(x|\boldsymbol{c}, a_{1:i}), 0\}, \tag{4}$$

$$h^{\text{block}}(a_{1:i}|\boldsymbol{c}) = \frac{r(a_{1:i}|\boldsymbol{c})}{1 - w(a_{1:i}|\boldsymbol{c}) + r(a_{1:i}|\boldsymbol{c})}, \tag{5}$$

for $i < L$, and $a_{1:L}$ with probability $w(a_{1:L}|\boldsymbol{c})$. They select the longest accepted prefix, of length τ, and sample a correction token $y \sim p^{\text{block}}_{\text{res}}(\cdot|\boldsymbol{c}, a_{1:i})$, a weighted version of $p^{\text{token}}_{\text{res}}$ proportional to the following vector,

$$\max\{w(a_{1:i}|\boldsymbol{c})p(\cdot|\boldsymbol{c}, a_{1:i}) - q(\cdot|\boldsymbol{c}, a_{1:i}), 0\}, \tag{6}$$

if $\tau < L$. If $\tau = L$, they sample $y \sim p(\cdot|\boldsymbol{c}, a_{1:L})$. Like in speculative sampling, the output $(a_{1:\tau}, y)$ follows the target distribution. Sun et al. (2024b) prove that block verification achieves the highest block efficiency among any verification algorithm that only takes in on-path distribution values.

2.2 Multi-Path

There are also recent lines of work that extend L-step speculative sampling to the *multi-path* setting (Sun et al., 2023; Spector & Re, 2023; Cai et al., 2024; Li et al., 2024a). In this setting, $K > 1$ length-L draft blocks are sampled from q_L. The target model M_p is again called *once* on all draft blocks in parallel, to obtain target and draft next-token probabilities along all K paths. Then, using a **multi-path verification algorithm**, a prefix of one of the blocks is accepted, and an additional correction token from a residual distribution is sampled in order to match the target distribution. Block efficiency is defined in the same way as previously. For moderate K and L, leveraging GPU parallelization, there is generally little overhead in performing the *batched* forward pass across K sequences, relative to a forward pass on one sequence (Agrawal et al., 2024; Dao et al., 2022).

We use the concept of a **draft tree** to enumerate all distinct prefixes among the K sampled paths. We will compare our methods against two categories of multi-path verification algorithms which select a node from the draft tree, i.e. a prefix of one of the blocks, in different ways.

OTLP-based algorithms. These perform a top-down traversal of the draft tree. At each step, they compute a feasible solution of an optimal transport linear program (OTLP) to determine the next token and progress to a child node, terminating when they land off the draft tree. SpecTr (Sun et al., 2023), NSS (Miao et al., 2024), and SpecInfer (Miao et al., 2024) are key examples, and recent theoretical OTLP solvers (Khisti et al., 2025; Hu et al., 2025) further improve block efficiency.

Traversal verification. To the best of our knowledge, Traversal Verification in Weng et al. (2025) is the only multi-path algorithm that traverses the draft tree from the bottom-up. Like our algorithm in Section 5, this reduces to BV in the $K = 1$ case. In our experiments, Traversal Verification and our methods outperform OTLP-based methods, but are effective in different (p, q) regimes.

3 Single-Path Information-Agnostic LP

We first formally define the class of single-path verification algorithms. We denote $\mathcal{V}^{\leq k} = \mathcal{V}^0 \cup \mathcal{V}^1 \cup \ldots \cup \mathcal{V}^k$, where $\mathcal{V}^k$ is the set of length-k sequences of tokens in $\mathcal{V}$, and thus $\mathcal{V}^{\leq k}$ is the set of sequences of length $\leq k$. We use L to denote the draft block length. For the remainder of the paper, we use notation from Section 2, omitting the context $\boldsymbol{c}$ and subscripts on p, q when they are clear.

Definition 3.1 (Single-path draft verification algorithm)**.** A **single-path verification algorithm** Φ takes in a sampled draft block $X_{1:L} \sim q_L(\cdot)$, and the full L-step target and draft distributions p_L and q_L, and returns a nonempty sequence in $\mathcal{V}^{\leq L+1}$. It is **valid** under:

- **Prefix-matching:** the algorithm returns a (possibly empty) prefix $X_{1:\tau}$ of the draft block followed by an additional correction token Y, for some $\tau \in \{0, \ldots, L\}$.
- **Target-matching:** for any context $\boldsymbol{c}$, any draft and target models M_p and M_q, and any block length L, when we generate $L - \tau$ additional tokens $Z \sim p_{L-\tau}(\cdot|X_{1:\tau}, Y)$, we have $(X_{1:\tau}, Y, Z) \sim_p p_{L+1}(\cdot)$, where $\sim_p$ denotes equality of distributions.

If we also require that Φ can only take in the *on-path* target and draft distributions $p(\cdot|X_{1:i}), q(\cdot|X_{1:i})$ for $i \in \{0, \ldots, L\}$, then it is called **information-restricted**.

Target-matching means that sampling from M_p autoregressively after verification, until a total of $L+1$ tokens are generated, gives the same result as just sampling $L+1$ tokens from M_p autoregressively without verification. This guarantees that running the verification algorithm and appending its output to the context iteratively maintains the target distribution, even as τ varies: see Lemma 2 in Appendix B of Sun et al. (2024b) for a formal proof.

Prefix-matching forces the verification output to only deviate from the draft block at its last token. This is a strict modeling requirement, not a technical convenience. To bypass prefix-matching while maintaining the target distribution, one must access the next-token target distribution $p(\cdot|\boldsymbol{c})$ for contexts $\boldsymbol{c}$ outside the draft tree. This is not possible without performing another M_p forward pass, which would defeat the goal of speculative decoding: generating multiple tokens in one target call.

Importantly, our definition is *less strict* than in Sun et al. (2024b), because they only consider the class of valid single-path *information-restricted* verification algorithms, and prove that block verification is optimal in this class. Surprisingly, we find that block verification is optimal in the class of *all* valid single-path verification algorithms. To prove this, we utilize node budgets, which represent how much mass a verification algorithm allocates along a path relative to the target distribution.

Definition 3.2. For any single-path verification algorithm Φ, **node budgets** for $a_{1:i} \in \mathcal{V}^{\leq L+1}$ are:

$$D_\Phi(a_{1:i}) = 1 - \sum_{j=1}^{i} \frac{\mathbb{P}(X_{1:\tau} = a_{1:j-1}, Y = a_j)}{p(a_{1:j})}. \tag{7}$$

This can be represented by a function $D_\Phi : \mathcal{V}^{\leq L+1} \to \mathbb{R}$.

Using node budget variables, we define the **single-path information-agnostic LP** in Theorem 3.3, which describes what values of D_Φ a valid single-path verification algorithm Φ can induce. Here, the chain of D_Φ inequalities along paths encode the target-matching constraint, and the pointwise upper bounds on $D_\Phi p$ encode prefix-matching. We defer the proof to Section B, as the special case $K = 1$ of the multi-path LP in Theorem 4.3, which we prove in Section A.

Theorem 3.3. *A single-path verification algorithm Φ is valid if and only if the node budgets $D_\Phi : \mathcal{V}^{\leq L+1} \to \mathbb{R}$ are* feasible *in the following LP:*

$$\max \sum_{a_{1:i} \in \mathcal{V}^{\leq L}} D_\Phi(a_{1:i}) p(a_{1:i}) \tag{8}$$

$$\text{s.t.} \quad 1 = D_\Phi(a_{1:0}) \geq \ldots \geq D_\Phi(a_{1:L}) \geq D_\Phi(a_{1:L+1}) = 0 \quad \forall a_{1:L+1} \in \mathcal{V}^{L+1}, \tag{9}$$

$$D_\Phi(a_{1:i}) p(a_{1:i}) \leq q(a_{1:i}) \quad \forall a_{1:i} \in \mathcal{V}^{\leq L}. \tag{10}$$

The objective value is the block efficiency $\mathbb{E}[\tau + 1]$ for Φ.

Because the feasibility conditions in the single-path LP only involve pointwise and pathwise inequalities, it is not hard to compute the optimal objective value and node budgets for a valid single-path verification algorithm, by using greedy allocation along paths. These node budgets match those derived from block verification, and thus block verification is the optimal valid single-path algorithm.

Theorem 3.4. *Block verification, which is a valid single-path verification algorithm Φ_{BV}, achieves the highest block efficiency of any valid single-path verification algorithm.*

See Section C for a proof of Theorem 3.4. These results show that prefix-matching is the key barrier to improving acceptance: even with access to the full joint target and draft distributions, prefix-matching (pointwise upper bounds on $D_\Phi p$ terms) prevents us from getting better block efficiency (the sum of $D_\Phi p$ terms) than block verification. This motivates us to explore verification algorithms that draft *multiple* candidate paths in the next section. In this setting, the valid prefix output space grows, thereby loosening the prefix-matching requirement and improving block efficiency.

4 Multi-Path Information Agnostic LP

Now, we show that off-path distribution information can improve block efficiency when $K > 1$ draft paths are generated, even with prefix-matching and target-matching requirements. We first define the class of multi-path verification algorithms, and extend Theorem 3.3 to these algorithms.

Definition 4.1 (K-path draft verification algorithm)**.** A K-path verification algorithm Φ takes in K i.i.d. sampled draft blocks $X_{1:L}^{(1)}, \ldots, X_{1:L}^{(K)} \sim q_L(\cdot|\boldsymbol{c})$, and the full L-step distributions p_L and q_L, and returns a nonempty sequence in $\mathcal{V}^{\leq L+1}$. It is **valid** under:

- **Prefix-matching:** the algorithm returns a (possibly empty) prefix $X_{1:\tau}$ of *some* draft block followed by a correction token Y, for some $\tau \in \{0, \ldots, L\}$.
- **Target-matching:** this is the same as in Definition 3.1.

If Φ can only take in the *on-path* target and draft distributions, which are $p(\cdot|X_{1:i}^{(k)}), q(\cdot|X_{1:i}^{(k)})$ for $k \in [K]$ and $i \in \{0, \ldots, L\}$, then it is called **information-restricted**.

To the best of our knowledge, all existing valid multi-path verification algorithms (Section 2.2) are information-restricted. We are the first to consider theoretical efficiency limits in the absence of information-restriction, and encode prefix-matching and target-matching into an LP.

To extend the single-path LP from Theorem 3.3 to this setting, we first define node budgets D_Φ in the same way as Definition 3.2. We also require the notion of an *antichain*, which is a set of variable-length paths in $\mathcal{V}^{\leq L}$ where no path is a prefix of another. Antichains are useful because the mass of an autoregressive distribution over an antichain can be computed by summing its probabilities at all antichain elements, due to the disjointness of autoregressively sampling antichain elements.

Definition 4.2. An **antichain** of $\mathcal{V}^{\leq L}$ is a subset of $\mathcal{V}^{\leq L}$ where no sequence in the antichain is a prefix of another. We denote the set of all antichains in $\mathcal{V}^{\leq L}$ by $\mathcal{A}(\mathcal{V}^{\leq L})$.

Now, to form the **multi-path information-agnostic LP** in Theorem 4.3, we replace the pointwise upper bounds on $D_\Phi p$ by subset-sum upper bounds over antichains. As mentioned at the end of Section 3, this corresponds to altering the prefix output space. The proof uses Hall-type feasibility constraints, and is fairly involved. Due to space constraints, we defer the proof to Section A.

Theorem 4.3. *A K-path verification algorithm Φ is valid if and only if the node budgets $D_\Phi : \mathcal{V}^{\leq L+1} \to \mathbb{R}$ are* feasible *in the following LP:*

$$\max \sum_{a_{1:i} \in \mathcal{V}^{\leq L}} D_\Phi(a_{1:i}) p(a_{1:i}) \tag{11}$$

$$\text{s.t.} \quad 1 = D_\Phi(a_{1:0}) \geq \ldots \geq D_\Phi(a_{1:L}) \geq D_\Phi(a_{1:L+1}) = 0 \quad \forall a_{1:L+1} \in \mathcal{V}^{L+1}, \tag{12}$$

$$1 - \sum_{a_{1:i} \in T} D_\Phi(a_{1:i}) p(a_{1:i}) \geq \left(1 - \sum_{a_{1:i} \in T} q(a_{1:i})\right)^K \quad \forall T \in \mathcal{A}(\mathcal{V}^{\leq L}). \tag{13}$$

The objective value is the block efficiency $\mathbb{E}[\tau + 1]$ for Φ.

This reduces to the single-path LP in the case $K = 1$. However, for $K > 1$, there is a important distinction: the upper bound on sums of $D_\Phi p$ now represent a *submodular* function of T, unlike the *additive (modular)* pointwise upper bounds on $D_\Phi p$ in Theorem 3.3. We now explain how this submodularity naturally arises from a **path selection rule** in valid K-path verification algorithms. In Lemma 4.4 (proof in Section D), we first prove a canonical decomposition of such algorithms into randomized selection of one of the drafted K paths, followed by valid single-path verification.

Lemma 4.4. *A valid K-path verification algorithm has the following equivalent definition. First, given K paths sampled i.i.d. from q, randomly select one through a path selection rule Γ, which can depend on off-path probability values. Say this path follows the distribution q^Γ over $\mathcal{V}^L$. Then, run a valid single-path verification algorithm on this path with target values p and draft values q^Γ.*

We call q^Γ the **skewed draft** induced by Γ. As this is a distribution over $\mathcal{V}^L$, just as for p_L, q_L, we can extend it to a distribution on each $\mathcal{V}^i$ for $0 \leq i \leq L$, and then form an autoregressive distribution:

$$q^\Gamma(a_{1:i}) = \sum_{a_{i+1:L} \in \mathcal{V}^{L-i}} q^\Gamma(a_{1:L}) \;\; \forall a_{1:i} \in \mathcal{V}^i, \quad q^\Gamma(a_{k+1:i} | a_{1:k}) = \frac{q^\Gamma(a_{1:i})}{q^\Gamma(a_{1:k})} \;\; \forall a_{1:i} \in \mathcal{V}^{\leq L}. \tag{14}$$

Not all distributions can be realized as the skewed draft distribution of a path selection rule, given a fixed draft q and path count K. In fact, as in Lemma 4.5, the realizable distributions are those which satisfy a submodular inequality like that in Theorem 4.3. For a proof, see Section E.

Lemma 4.5. *Fix K, a draft q, and a distribution r. Then $r = q^\Gamma$ for some path selection rule Γ over K drafts sampled i.i.d. from q if and only if the following holds over all antichains $T \in \mathcal{A}(\mathcal{V}^{\leq L})$:*

$$\sum_{a_{1:L} \in T} r(a_{1:L}) \leq 1 - \left(1 - \sum_{a_{1:i} \in T} q(a_{1:i})\right)^K. \tag{15}$$

This shows that Lemma 4.4 corresponds to **linearizing** the submodular constraints in Theorem 4.3. Indeed, consider replacing the *submodular* upper bounds in the multi-path LP with the *additive (modular)* upper bounds $\sum_{a_{1:i} \in T} q^\Gamma(a_{1:i})$. By the upper bound in Lemma 4.5, any solution to this **linearized LP** will satisfy the original multi-path LP. In fact, this linearized LP is precisely the single-path LP[1] where the draft distribution is the skewed draft q^Γ rather than the original q. This corresponds to running valid single-path verification with target p and draft q^Γ, as in Lemma 4.4.

For fixed q^Γ, by Theorem 3.4, the optimal single-path algorithm is block verification. This allows us to explicitly describe the optimal multi-path algorithm in Theorem 4.6, proved in Section F.

Theorem 4.6. *The optimal valid multi-path verification algorithm randomly chooses one of the K i.i.d. blocks with Γ and then runs single-path block verification on that path with target values p and draft values q^Γ, for some path selection rule Γ. Furthermore, the optimal choice of Γ solves:*

$$\max_\Gamma \sum_{a_{1:i} \in \mathcal{V}^{\leq L}} \min_{0 \leq k \leq i} p(a_{k+1:i} | a_{1:k}) q^\Gamma(a_{1:k}) \tag{16}$$

$$\text{s.t.} \quad 1 - \sum_{a_{1:L} \in T} q^\Gamma(a_{1:L}) \geq \left(1 - \sum_{a_{1:L} \in T} q(a_{1:L})\right)^K \quad \forall T \subseteq \mathcal{V}^L. \tag{17}$$

[1] See Section B for an explanation of how the sums reduce to pointwise inequalities like in Theorem 3.3.

This optimization problem is intractable, as it requires full joint target and draft distributions. Also, even if the optimal q^Γ can be computed, it can be difficult to efficiently compute a path selection rule Γ that induces this skewed draft distribution. Thus, we turn to approximations. By Lemma 4.7, we can obtain a near-optimal solution by pushing all q^Γ/p ratios to one. The proof is in Section G.

Lemma 4.7. *For a fixed q^Γ, the objective value in Theorem 4.6 is lower bounded by*

$$(L+1)\cdot \min_{a_{1:i}\in\mathcal{V}^{\le L}} \frac{q^\Gamma(a_{1:i})}{p(a_{1:i})} \tag{18}$$

This shows that when $q^\Gamma = p$ is feasible in Theorem 4.6, we can achieve an optimal block efficiency of $L+1$, accepting an entire path. However, finding Γ that pushes q^Γ near p is difficult. Thus, in the next section, we focus on a more tractable class of *greedy* algorithms, where one can both explicitly compute the randomized rule Γ and the resulting skewed draft q^Γ. This characterization is necessary to apply Lemma 4.4, as block verification requires exact draft probability values q^Γ along the selected path, and it is difficult to exactly sample from q^Γ without an explicit randomized rule Γ.

5 Greedy Multi-Path Block Verification

We now design a path selection rule Γ where the skewed draft q^Γ is heuristically close to p and simple to compute with only on-path values. This leads to our algorithm: **greedy multi-path block verification (GBV)**. We first define the class of greedy verification algorithms, which contains GBV.

Definition 5.1. A **greedy multi-path valid verification algorithm** forms a global ranking on paths in $\mathcal{V}^L$, and always selects the highest-ranked drafted path in the path selection rule Γ (Lemma 4.4).

In the context of the multi-path LP, these algorithms can be viewed as outputs of a greedy polymatroid algorithm (Schrijver et al., 2003) on the submodular feasibility constraints in Theorem 4.3, with the algorithm-selected order matching the global ranking. We expand on this in Section H.

If the global ordering of paths is $\mathcal{V}^L = \{P_1, \dots, P_M\}$, then one selects P_m from Γ if and only if all K drafted paths lie in $\{P_1, \dots, P_m\}$, but do not all lie $\{P_1, \dots, P_{m-1}\}$. Thus, we can write:

$$q^\Gamma(P_m) = \left(\sum_{j=1}^{m} q(P_j)\right)^K - \left(\sum_{j=1}^{m-1} q(P_j)\right)^K. \tag{19}$$

Computing this for arbitrary global orderings may require off-path values. However, when paths are ordered by a tree-based rule, this is not an issue. Tree-based rules create a global ranking combining local orderings at all $\mathcal{V}^{\le L}$ prefixes into a lexicographic ordering on length-L paths.

Definition 5.2. A global ranking of paths in $\mathcal{V}^L$ is **tree-based** if the ranking can be obtained as follows. Define an injective function $\pi_{a_{1:i}} : \mathcal{V} \to \mathbb{R}$ at each $a_{1:i} \in \mathcal{V}^{\le L}$, only using the values $p(\cdot|a_{1:i}), q(\cdot|a_{1:i})$. Then, assign to each path $a_{1:L} \in \mathcal{V}^L$ the L-tuple $O(a_{1:L}) = (\pi_{a_{1:i-1}}(a_i))_{i=1}^L$. Finally, rank the paths $a_{1:L} \in \mathcal{V}^L$ in increasing lexicographic order of $O(a_{1:L})$.

Let all $\pi_{a_{1:i}}$ induce the path selection rule Γ_g, with sets $\mathcal{P}_{a_{1:i}}(x) = \{t \in \mathcal{V} : \pi_{a_{1:i}}(t) < \pi_{a_{1:i}}(x)\}$. From Equation (19) and Section I, we can compute each $q^{\Gamma_g}(a_{1:i})$ in $O(|\mathcal{V}|\log|\mathcal{V}|)$ time:

$$q^{\Gamma_g}(a_{1:i}) = \left(q(a_{1:i}) + \sum_{j=0}^{i-1} q(a_{1:j}) \sum_{t\in\mathcal{P}_{a_{1:j}}(a_{j+1})} q(t|a_{1:j})\right)^K - \left(\sum_{j=0}^{i-1} q(a_{1:j}) \sum_{t\in\mathcal{P}_{a_{1:j}}(a_{j+1})} q(t|a_{1:j})\right)^K. \tag{20}$$

Finally, as our path selection rule Γ_g and skewed draft q^{Γ_g} are efficient to compute and only use on-path values, we can use Lemma 4.4 with block verification as our single-path algorithm. This results in **greedy multi-path block verification (GBV)** in Algorithm 1, which allows one to choose arbitrary local orderings π. When $K = 1$, regardless of the choice of π, this is equivalent to block verification with draft q and target p, because Γ_g only has one path to select and thus $q^{\Gamma_g} = q$.

All that remains is to choose the local orderings $\pi_{a_{1:i}}$. We provide some theoretical justification for the heuristic $\pi_{a_{1:i}}(x) = p(x|a_{1:i})/q(x|a_{1:i})$ in Section J, and use this as the default in our experiments. Nevertheless, as we will see in Section 6.1, other simple heuristics can sometimes perform better. In practice, the best heuristic depends significantly on the alignment of p, q.

Table 1: Block efficiency, measured in tokens per target model call, for GBV with OPT 6.7B/350M and OPT 6.7B/125M on GSM8K, HumanEval, and MATH500. Larger numbers are better.

	OPT 6.7B/350M				**OPT 6.7B/125M**		
K	**GSM8K**	**HumanEval**	**MATH500**	K	**GSM8K**	**HumanEval**	**MATH500**
1	3.294	3.157	3.227	1	2.937	2.653	2.814
2	3.827	3.538	3.721	2	3.407	3.023	3.274
3	3.867	3.809	3.856	3	3.526	3.233	3.392
4	**3.884**	**3.863**	**3.896**	4	**3.562**	**3.503**	**3.493**

Table 2: Walltime, measured in ms/token, for GBV with OPT 6.7B/350M and OPT 6.7B/125M on GSM8K, HumanEval, and MATH500. Smaller numbers are better.

	OPT 6.7B/350M				**OPT 6.7B/125M**		
K	**GSM8K**	**HumanEval**	**MATH500**	K	**GSM8K**	**HumanEval**	**MATH500**
1	44.482	46.955	45.851	1	34.479	38.824	36.022
2	39.592	43.373	40.473	2	30.862	35.648	32.166
3	**38.862**	**41.017**	**38.889**	3	**30.109**	**33.627**	**31.163**
4	48.473	50.656	48.440	4	39.831	41.297	40.329

6 EXPERIMENTS

We now test GBV for $K = 1, 2, 3, 4$ paths[2]. Recall $K = 1$ is single-path block verification from Sun et al. (2024b), our baseline. We select a block length $L = 8$ and evaluate single-batch temperature 1.0 sampling with 128 generated tokens from the target-draft model pairs OPT 6.7B/350M and OPT 6.7B/125M (Zhang et al., 2022). We evaluate GSM8K, HumanEval, and MATH500 (Chen et al., 2021; Hendrycks et al., 2021; Cobbe et al., 2021), taking 500 random problems from each test split (HumanEval has 164). We run our experiments on a Paperspace machine with an A100-80GB and Intel Xeon Gold 6342 CPU (12 cores, 2.80 GHz). *Block efficiency* is generated tokens per target model call (higher is better), and *walltime* is average milliseconds per token (lower is better).

Our block efficiencies are in Table 1. In all settings, increasing K improves block efficiency. From $K = 1$ to $K = 4$, there are gains up to 32.04%, with an average of 23.08% across all models and datasets. However, returns diminish: average increases between $K = 1, 2, 3, 4$ are 14.98%, 4.40%, and 2.54%. While GBV greatly improves block efficiency, the most impactful gains are at $K = 2$.

While block efficiency gains are significant, they do not exactly match our walltime results in Table 2, as they are not monotonic in K. There are major improvements from $K = 1$ to $K = 3$, with an average walltime reduction of 13.34% across all settings. For OPT 6.7B/350M, this even reaches 15.19% on MATH500. However, $K = 4$ always performs worse than the baseline $K = 1$, with an average walltime increase of 9.39%. This is due to the increased computation cost of the batched target forward pass on $K = 4$ paths (see Section 2.2), which negates block efficiency gains.

We also analyze gains per dataset. The relative gains from $K = 1$ to $K = 4$, averaged across models, are highest for HumanEval at 27.20%, followed by MATH500 at 22.43% and GSM8K at 19.60%. The best setting $K = 3$ reduces walltime relative to $K = 1$ by 12.65% for GSM8K, 14.34% for MATH500, and 13.02% for HumanEval, so block efficiency and walltime rankings differ.

We further examine the impact of target and draft sizes. Averaged across datasets, the larger draft (350M) has 13.72% higher block efficiency than the smaller draft (125M) at $K = 3$. However, the smaller draft has 20.14% lower average walltime than the larger draft in the same setting. While a larger draft model can improve efficiency, this comes at the cost of higher latency in draft generation.

In practice, we recommend $L = 8, K = 3$. This achieves most of the block efficiency gains of $K = 4$ without incurring significant latency for target model calls. In all settings, $K = 3$ is the fastest method, with $K = 2$ following at an average 10% walltime reduction on block verification.

[2]We release our anonymized code here with implementations of other multi-path methods.

6.1 Other Model Families and Interventions

While $K = 4$ has optimal block efficiency for OPT, this is not always the case. We evaluate Qwen-3 32B/0.6B and Llama-3 70B/8B (Yang et al., 2025; Grattafiori et al., 2024) in Section L. We find that block verification ($K = 1$) always beats GBV, and provide a systems cost breakdown to explain walltime degradation. This shows how to best deploy GBV: $K = 3$ and $K = 1$ are better suited for different families. Nevertheless, we can still improve GBV performance on Qwen-3 in three ways.

Tune output length and heuristics In Section M, we increase the generation length from 128 to 1024, and include local orderings p, q, q/p, and $\min(1, p/q)$ alongside the default p/q rule. We find that GBV with $K > 1$ now performs better than BV for all heuristics except q/p, but p/q is not always the best heuristic. Also, for many heuristics, $K = 4$ achieves the best walltime. In particular, the strongest method at temperature 1.0 is GBV-$\min(1, p/q)$ with $K = 4$, and the best at temperature 0.2 is GBV-p with $K = 4$, each of which reduce BV walltime by around 15%.

GBV-aligned distillation Next, we perform heuristic-aligned distillation. While block verification works well when $p \approx q$, GBV requires $p \approx q^{\Gamma}$. Thus, to improve GBV performance, we perform teacher-forced distillation on Qwen-3 0.6B from Qwen-3 32B trajectories, using the GBV-aligned loss function $\|p - q^{\Gamma}\|_1$. In Section N, we train with local orderings p/q and q/p. Surprisingly, q/p outperforms the more principled ordering p/q. At $K = 3$, the GBV-q/p-aligned draft improves block efficiency by 22.9% and throughput by 23.7% over to the $K = 3$ GBV-q/p baseline.

Lower target temperature Finally, in Section O, we expand our GSM8K experiments from temperature 1.0 to temperatures 0.2, 0.4, 0.6, and 0.8. At low temperature, GBV improves and block verification degrades: at temperature 0.2, GBV with $K = 3$ is over 0.6 tokens/s faster than $K = 1$.

6.2 Draft Length Ablations

In Section P, we increase our $L = 8$ draft length to $L = 16, 24$. We find this yields modest gains in block efficiency, but throughput degrades rapidly. We show this occurs because longer block lengths push more relative work into the drafting bottleneck, and we find the ideal choice of L is around 8.

6.3 Comparison to Multi-Path Methods

Now, we compare GBV to SpecTr, SpecInfer, and Traversal Verification (Sun et al., 2023; Miao et al., 2024; Weng et al., 2025) on Llama-3 70B/8B in Section Q. Traversal beats GBV at temperature 1.0 by 0.9 tokens/s, GBV beat Traversal at temperature 0.2 by 1.5 tokens/s, and other methods lag behind. **GBV with $K = 2$ is 11.5 tokens/s at temperature 0.2, while no other method is over 10 tokens/s in any setting.** Thus, GBV is the most effective multi-path algorithm at low temperatures.

6.4 Integration with EAGLE-3

In Section R, we integrate GBV-p, GBV-p/q, and Traversal into EAGLE-3 (Li et al., 2025), by directly using their drafter. While EAGLE-3 beats GBV and Traversal by around 20 tokens/s at temperature 1.0, GBV-p and GBV-p/q beat EAGLE-3 by 31 and 24 tokens/s at temperature 1.4. Also, GBV-p/q beats Traversal at temperatures 1.1, 1.2, and 1.3. Thus, GBV smooths EAGLE-3 temperature instability. We expect GBV-aligned distillation to further close the temperature 1.0 gap.

7 Conclusion

We developed a single-path information-agnostic linear program (LP) which encodes feasible speedups for valid verification algorithms in speculative sampling, and showed that block verification remains optimal even with access to off-path probabilities. We further extended our LP to the setting where multiple draft paths are generated. While this multi-path LP is not feasible to solve exactly, by approximating it with a class of greedy verification algorithms, we developed a generalization of block verification, called greedy multi-path block verification. This significantly improves decoding efficiency relative to block verification. Future work could further explore the class of greedy verification schemes, or explore alternative approximation to the multi-path LP.

AI USE STATEMENT

In this work, we used generative AI tools to polish writing and figures and search for related work. We have not used generative AI tools to perform any research ideation or execution, to draft sections of the paper, to generate synthetic datasets, or to prove mathematical claims. We have reviewed all AI-assisted work. For example, we checked all LLM-generated related work suggestions through a manual literature survey. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We include full details to reproduce our algorithm in Section 5, and full details to reproduce our experimental results in Section 6.

REFERENCES

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in llm inference with sarathi-serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI '24)*, 2024. URL https://www.usenix.org/system/files/osdi24-agrawal.pdf.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

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.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chang, and Jie Huang. Cascade speculative drafting for even faster llm inference. *Advances in Neural Information Processing Systems*, 37:86226–86242, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Łukasz Kaiser, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems (NeurIPS 2022)*, 2022. URL https://openreview.net/pdf?id=H4DqfPSibmx.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

David Gale. A theorem on flows in networks. In *Classic Papers in Combinatorics*, pp. 259–268. Springer, 1957.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D Lee, and Di He. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Zhengmian Hu and Heng Huang. Accelerated speculative sampling based on tree monte carlo. In *Forty-first International Conference on Machine Learning*, 2024.

Zhengmian Hu, Tong Zheng, Vignesh Viswanathan, Ziyi Chen, Ryan A Rossi, Yihan Wu, Dinesh Manocha, and Heng Huang. Towards optimal multi-draft speculative decoding. *arXiv preprint arXiv:2502.18779*, 2025.

Ashish J Khisti, MohammadReza Ebrahimi, Hassan Dbouk, Arash Behboodi, Roland Memisevic, and Christos Louizos. Multi-draft speculative sampling: Canonical decomposition and theoretical limits. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

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, 2024a.

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

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online speculative decoding. *arXiv preprint arXiv:2310.07177*, 2023.

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 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.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf, 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf, 2019.

Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. Your llm knows the future: Uncovering its multi-token prediction potential. *arXiv preprint arXiv:2507.11851*, 2025.

Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

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. *arXiv preprint arXiv:2210.03057*, 2022.

Benjamin Spector and Chris Re. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*, 2023.

Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. *arXiv preprint arXiv:2404.11912*, 2024a.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36:30222–30242, 2023.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. Block verification accelerates speculative decoding. *arXiv preprint arXiv:2403.10444*, 2024b.

Yepeng Weng, Qiao Hu, Xujie Chen, Li Liu, Dianwen Mei, Huishi Qiu, Jiang Tian, and Zhongchao Shi. Traversal verification for speculative tree decoding. *arXiv preprint arXiv:2505.12398*, 2025.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7655–7671, 2024.

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.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *arXiv preprint arXiv:2309.08168*, 2023.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. *arXiv preprint arXiv:2310.08461*, 2023.

A Proof of Multi-Path Information-Agnostic LP

In this section, we prove the multi-path information-agnostic LP in Theorem 4.3. Our proof begins with a key lemma that dictates when one can sample from one distribution given data from another, with the conditional distribution (often called the transport) to induce the former restricted to a given support. In our context, the restricted support represents the prefix-matching requirement, the distribution from which we are given data is induced by i.i.d. sampling paths from the draft model, and the distribution from which we would like to sample is that of the verification algorithm output.

Lemma A.1 (Bipartite Transport Feasibility). *Let $G = (A \cup B, E \subseteq A \times B)$ be a bipartite graph, with probability distributions $a(\cdot)$ and $b(\cdot)$ over A and B, respectively. Then the following conditions are equivalent, where $N(\cdot)$ denotes neighborhoods:*

1. *There exists a joint distribution $\pi(\cdot,\cdot)$ over $A \times B$ with marginal distributions $a(\cdot)$ and $b(\cdot)$, such that $\pi(x,y) = 0$ for all $(x,y) \notin E$.*
2. *For any $S \subseteq A$, we have $\sum_{x \in S} a(x) \leq \sum_{y \in N(S)} b(y)$.*

Proof. Condition 1 is equivalent to the feasibility of the following LP in variables $\pi_{x,y} \geq 0$:

$$\sum_{x \in A} \pi_{x,y} = b(y) \quad \forall y \in B, \quad \sum_{y \in B} \pi_{x,y} = a(x) \quad \forall x \in A, \quad \pi_{x,y} = 0 \quad \forall (x,y) \notin E. \tag{21}$$

We can incorporate the zero equality condition into the sum equalities to turn this into:

$$\sum_{x \in N(y)} \pi_{x,y} = b(y) \quad \forall y \in B, \quad \sum_{y \in N(x)} \pi_{x,y} = a(x) \quad \forall x \in A. \tag{22}$$

Now, Gale's feasibility theorem for bipartite supply-demand networks (Gale, 1957) implies this LP is feasible in nonnegative variables if and only if

$$\sum_{x \in S} a(x) \leq \sum_{y \in N(S)} b(x) \quad \forall S \subseteq A, \quad \sum_{x \in A} a(x) = \sum_{y \in B} b(x). \tag{23}$$

The equality condition is automatically satisfied, since $a(\cdot), b(\cdot)$ are probability distributions, so both sides become one. Thus, Condition 1 is equivalent to Condition 2. □

Using this, we can prove the following lemma, which dictates what distributions are feasible for a K-path verification algorithm Φ which satisfies prefix-matching. We do not enforce target-matching yet, so this algorithm is not necessarily valid. For any K-path verification algorithm Φ, denote $R_\Phi(\cdot)$ as the distribution of the output of Φ, which is supported over $\mathcal{V}^{\leq L+1} \setminus \mathcal{V}^0$, i.e. all nonempty paths of length $\leq L+1$. We need the nonempty requirement as we must output at least one token. As we prove in Lemma A.2, a collection of subset-sum inequalities over a particular bipartite graph exactly describes the set of R_Φ which can be realized by some Φ satisfying prefix-matching.

Lemma A.2 (Prefix-Matching). *Construct the bipartite graph G with left vertices $V^{\leq L+1} \setminus \mathcal{V}^0$, right vertices $(\mathcal{V}^L)^K$, and edges E consisting of all pairs*

$$\left(a_{1:i}, \left(a_{1:L}^{(1)}, \ldots, a_{1:L}^{(K)}\right)\right) \in \left(\mathcal{V}^{\leq L+1} \setminus \mathcal{V}^0\right) \times \left(\mathcal{V}^L\right)^K, \quad a_{1:i-1} \in \left\{a_{1:i-1}^{(1)}, \ldots, a_{1:i-1}^{(K)}\right\}. \tag{24}$$

The values $R_\Phi(a_{1:i})$ for $a_{1:i} \in \mathcal{V}^{\leq L+1} \setminus \mathcal{V}^0$ are feasible for a K-path verification algorithm Φ that satisfies prefix-matching if and only if they are nonnegative, sum to one, and

$$\sum_{a_{1:i} \in S} R_\Phi(a_{1:i}) \leq \sum_{\left(a_{1:L}^{(1)}, \ldots, a_{1:L}^{(K)}\right) \in N(S)} \prod_{k=1}^{K} q\left(a_{1:L}^{(k)}\right) \quad \forall S \subseteq V^{\leq L+1} \setminus \mathcal{V}^0, \tag{25}$$

where $N(\cdot)$ denotes a neighborhood in G.

Proof. The sum to one equality and nonnegative requirements are equivalent to requiring that R_Φ is a valid probability distribution. Now, given that R_Φ is a valid probability distribution, we observe that the prefix-matching requirement is equivalent to the following: given K i.i.d. sampled paths

$$X_{1:L}^{(1)}, \dots, X_{1:L}^{(K)} \sim q(\cdot), \tag{26}$$

there is a conditional distribution (randomized rule) $\pi(\cdot|X_{1:L}^{(1)}, \dots, X_{1:L}^{(K)})$ over $\mathcal{V}^{\leq L+1} \setminus \mathcal{V}^0$ which is supported only over prefixes of these paths plus an additional token, such that sampling from π results in the distribution R_Φ . Given the construction of the graph G, we can alternatively express the restricted support requirement on $\pi(\cdot|\cdot)$ as $\pi(v_l|v_r) = 0$ for all left vertices v_l and right vertices v_r with $(v_l, v_r) \notin E$. Thus, by turning this conditional distribution into a joint distribution, as R_Φ is a distribution over left vertices and i.i.d. sampling from $q(\cdot)$ is a distribution over right vertices, Lemma A.1 directly applies to show that prefix-matching is equivalent to

$$\sum_{a_{1:i} \in S} R_\Phi(a_{1:i}) \leq \sum_{\left(a_{1:L}^{(1)}, \dots, a_{1:L}^{(K)}\right) \in N(S)} \prod_{k=1}^{K} q\left(a_{1:L}^{(k)}\right) \quad \forall S \subseteq V^{\leq L+1} \setminus \mathcal{V}^0. \tag{27}$$

The product term is the probability of sampling K paths i.i.d from q. This completes the proof. □

Now, we move onto the target-matching requirement. Again, we can establish a necessary and sufficient condition for a K-path verification algorithm Φ to satisfy target-matching terms of the R_Φ values. Now, the condition is path-wise summation requirement.

Lemma A.3 (Target-Matching)**.** *The values $R_\Phi(a_{1:i})$ for $a_{1:i} \in \mathcal{V}^{\leq L+1} \setminus \mathcal{V}^0$ are feasible for a K-path verification algorithm Φ that satisfies prefix-matching if and only if they are nonnegative, sum to one, and*

$$\sum_{i=0}^{L} \frac{R_\Phi(a_{1:i+1})}{p(a_{1:i+1})} = 1 \quad \forall a_{1:L+1} \in \mathcal{V}^{L+1}. \tag{28}$$

Proof. Target-matching asserts that sampling from $p_{L+1}(\cdot)$ is equivalent to sampling the output $(X_{1:\tau}, Y) \sim R_\Phi$ of the verification algorithm and then sampling an additional $L - \tau$ tokens from $Z_{1:L-\tau} \sim p_{L-\tau}(\cdot|X_{1:\tau}, Y)$. For a given path $a_{1:L+1} \in \mathcal{V}^{L+1}$, the probability of sampling this path from the former method is $p_{L+1}(a_{1:L+1})$. The probability of sampling it from the latter method is

$$\mathbb{P}(X_{1:\tau} = a_{1:\tau}, Y = a_{\tau+1}, Z_{1:L-\tau} = a_{\tau+2:L+1}) \tag{29}$$

$$= \sum_{i=0}^{L} \mathbb{P}(\tau = i, X_{1:\tau} = a_{1:\tau}, Y = a_{\tau+1}, Z_{1:L-\tau} = a_{\tau+2:L+1}) \tag{30}$$

$$= \sum_{i=0}^{L} \mathbb{P}(\tau = i, X_{1:\tau} = a_{1:i}, Y = a_{i+1}, Z_{1:L-\tau} = a_{i+2:L+1}) \tag{31}$$

$$= \sum_{i=0}^{L} \mathbb{P}((X_{1:\tau}, Y) = a_{1:i+1}) p(a_{i+2:L+1}|a_{1:i+1}) \tag{32}$$

$$= \sum_{i=0}^{L} R_\Phi(a_{1:i+1}) \cdot \frac{p(a_{1:L+1})}{p(a_{1:i+1})}. \tag{33}$$

Setting these two equal for all paths $a_{1:L+1}$ and dividing out the $p(a_{1:L+1}) = p_{L+1}(a_{1:L+1})$ term gives the desired condition. □

By combining the prefix-matching and target-matching lemmas, we can now write down the multi-path LP, in a form which is slightly more complicated than in Theorem 4.3, and is in terms of variables R_Φ rather than the node budgets D_Φ. We will use the notion of a parent set and minimal set.

Definition A.4. For each $S \subseteq V^{\leq L+1} \setminus \mathcal{V}^0$, the **parent set** $\text{par}(S) \subseteq \mathcal{V}^{\leq L}$ of S consists of all $a_{1:i} \in \mathcal{V}^{\leq L}$ where $a_{1:i+1} \in S$ for some $a_{i+1} \in \mathcal{V}$. For each $S \subseteq V^{\leq L}$, the **minimal set** $\min(S) \subseteq S$ of S consists of all $a_{1:i} \in S$ where no proper prefix of $a_{1:i}$ appears in S, i.e. $a_{1:0}, \dots, a_{1:i-1} \notin S$.

Note that an antichain (Definition 4.2) is precisely a subset $S \subseteq \mathcal{V}^{\leq L}$ where $\min(S) = S$, and any minimal set $\min(S)$ is also an antichain because $\min(\min(S)) = \min(S)$. We will use parent and minimal sets to remove redundant inequalities in the Lemma A.2.

Lemma A.5 (Unsimplified Multi-Path LP)**.** *The values $R_\Phi(a_{1:i})$ for $a_{1:i} \in \mathcal{V}^{\leq L+1} \setminus \mathcal{V}^0$ are feasible for a valid K-path verification algorithm Φ if and only if they are nonnegative, sum to one, and are feasible in*

$$\max \sum_{i=1}^{L+1} \sum_{a_{1:i} \in \mathcal{V}^i} i R_\Phi(a_{1:i}), \tag{34}$$

$$\text{s.t.} \quad \sum_{i=0}^{L} \frac{R_\Phi(a_{1:i+1})}{p(a_{1:i+1})} = 1 \quad \forall a_{1:L+1} \in \mathcal{V}^{L+1}, \tag{35}$$

$$\sum_{a_{1:i} \in T} \sum_{j=i+1}^{L+1} \sum_{a_{i+1:j} \in \mathcal{V}^{j-i}} R_\Phi(a_{1:j}) \leq 1 - \left(1 - \sum_{a_{1:i} \in T} q(a_{1:i})\right)^K \quad \forall T \in \mathcal{A}(\mathcal{V}^{\leq L}). \tag{36}$$

Furthermore, the objective value is precisely the block efficiency $\mathbb{E}[\tau + 1]$ for Φ.

Proof. First, fix a subset $S \subseteq \mathcal{V}^{\leq L+1} \setminus \mathcal{V}^0$. The neighborhood $N(S) \subseteq (\mathcal{V}^L)^K$ induced by the graph G in Lemma A.2 is the set of K-tuples of length-L paths where at least one path has a prefix in $\text{par}(S)$. So, the complement of $N(S)$ consists of all K-tuples of length-L paths where no path has a prefix in $\text{par}(S)$. Now, note that a path has a prefix in $\text{par}(S)$ if and only if it has a prefix in $\min(\text{par}(S))$. Because $\min(\text{par}(S))$ is an antichain (i.e. no two distinct elements in it are a prefix of the same path), disjointness shows that the probability that a length L-path sampled from $q(\cdot)$ has a prefix in $\min(\text{par}(S))$ is $\sum_{a_{1:i} \in \min(\text{par}(S))} q(a_{1:i})$. Using these observations, we simplify:

$$\sum_{\left(a_{1:L}^{(1)}, \ldots, a_{1:L}^{(K)}\right) \in N(S)} \prod_{k=1}^{K} q\left(a_{1:L}^{(k)}\right) = 1 - \sum_{\left(a_{1:L}^{(1)}, \ldots, a_{1:L}^{(K)}\right) \notin N(S)} \prod_{k=1}^{K} q\left(a_{1:L}^{(k)}\right) \tag{37}$$

$$= 1 - \left(1 - \sum_{a_{1:i} \in \min(\text{par}(S))} q(a_{1:i})\right)^K. \tag{38}$$

Thus, the prefix-matching inequalities from Lemma A.2 become:

$$\sum_{a_{1:i} \in S} R_\Phi(a_{1:i}) \leq 1 - \left(1 - \sum_{a_{1:i} \in \min(\text{par}(S))} q(a_{1:i})\right)^K \quad \forall S \subseteq V^{\leq L+1} \setminus \mathcal{V}^0. \tag{39}$$

We now make the key observation that many of these inequalities are redundant. For any $a_{1:i}$ with a proper (i.e. not full) prefix in $\min(\text{par}(S))$, we can add $a_{1:i}$ to S without changing $\min(\text{par}(S))$, keeping the right hand side constant. This also increases the left hand side (since R_Φ is nonnegative). Thus, the strictest of these inequalities are precisely those where S includes all paths with a proper prefix in $\min(\text{par}(S))$. For a fixed $T = \min(\text{par}(S))$, this makes S the set of all $a_{1:j}$ where $a_{1:i} \in T$ for some $i < j$. Since all antichains are achievable as some $\min(\text{par}(S))$, this turns the above collection of inequalities into

$$\sum_{a_{1:i} \in T} \sum_{j=i+1}^{L+1} \sum_{a_{i+1:j} \in \mathcal{V}^{\leq j-i}} R_\Phi(a_{1:j}) \leq 1 - \left(1 - \sum_{a_{1:i} \in T} q(a_{1:i})\right)^K \quad \forall T \in \mathcal{A}(\mathcal{V}^{\leq L}). \tag{40}$$

This covers the prefix-matching. The target-matching requirement remains the same as in Lemma A.3. Thus, all that remains is to show the objective values is the block efficiency. This holds as $R_\Phi(a_{1:i})$ is the probability that Φ outputs $a_{1:i}$, and the total number of tokens generated with this output is i, so summing over all possible outputs $a_{1:i} \in \mathcal{V}^{\leq L+1} \setminus \mathcal{V}^0$ gives the desired result. □

Finally, we can prove the multi-path LP in Theorem 4.3. The only remaining step to derive this is to turn the variables R_Φ into variables D_Φ representing node budgets.

Proof. By the definition of D_Φ in Definition 3.2, we see that

$$D_\Phi(a_{1:i}) = 1 - \sum_{j=1}^{i} \frac{R_\Phi(a_{1:i})}{p(a_{1:i})} \quad \forall a_{1:i} \in \mathcal{V}^{\leq L+1}. \tag{41}$$

The target-matching condition simply implies that each $D_\Phi(a_{1:L+1}) = 0$. Now, observe that

$$R_\Phi(a_{1:i}) = D_\Phi(a_{1:i-1})p(a_{1:i}) - D_\Phi(a_{1:i})p(a_{1:i}) \quad \forall a_{1:i} \in \mathcal{V}^{\leq L}. \tag{42}$$

for all. Thus, incorporating the nonnegativity constraint on R_Φ, we get

$$1 = D_\Phi(a_{1:0}) \geq D_\Phi(a_{1:1}) \geq \ldots \geq D_\Phi(a_{1:L}) \geq D_\Phi(a_{1:L+1}) = 0 \quad \forall a_{1:L+1} \in \mathcal{V}^{L+1}. \tag{43}$$

Next, observe that for any $a_{1:i} \in \mathcal{V}^{\leq L}$, we have the following telescoping identity:

$$\sum_{j=i+1}^{L+1} \sum_{a_{i+1:j} \in \mathcal{V}^{j-i}} R_\Phi(a_{1:j}) \tag{44}$$

$$= \sum_{j=i+1}^{L+1} \sum_{a_{i+1:j} \in \mathcal{V}^{j-i}} D_\Phi(a_{1:j-1})p(a_{1:j}) - \sum_{j=i+1}^{L+1} \sum_{a_{i+1:j} \in \mathcal{V}^{j-i}} D_\Phi(a_{1:j})p(a_{1:j}) \tag{45}$$

$$= \sum_{j=i}^{L} \sum_{a_{i+1:j+1} \in \mathcal{V}^{j+1-i}} D_\Phi(a_{1:j})p(a_{1:j+1}) - \sum_{j=i+1}^{L+1} \sum_{a_{i+1:j} \in \mathcal{V}^{j-i}} D_\Phi(a_{1:j})p(a_{1:j}) \tag{46}$$

$$= D_\Phi(a_{1:i})p(a_{1:i}) + \sum_{j=i+1}^{L} \sum_{a_{i+1:j+1} \in \mathcal{V}^{j+1-i}} D_\Phi(a_{1:j})p(a_{1:j+1}) \tag{47}$$

$$- \sum_{j=i+1}^{L} \sum_{a_{i+1:j} \in \mathcal{V}^{j-i}} D_\Phi(a_{1:j})p(a_{1:j}) - 0 \tag{48}$$

$$= D_\Phi(a_{1:i})p(a_{1:i}) + \sum_{j=i+1}^{L} D_\Phi(a_{1:j}) \left(\sum_{a_{i+1:j+1} \in \mathcal{V}^{j+1-i}} p(a_{1:j+1}) - \sum_{a_{i+1:j} \in \mathcal{V}^{j-i}} p(a_{1:j}) \right) \tag{49}$$

$$= D_\Phi(a_{1:i})p(a_{1:i}) + \sum_{j=i+1}^{L} D_\Phi(a_{1:j}) \left(p(a_{1:i}) - p(a_{1:i}) \right) = D_\Phi(a_{1:i})p(a_{1:i}). \tag{50}$$

This means the condition that all R_Φ sum to one is automatically satisfied by applying this identity for $i = 0$. This also reduces the antichain condition in the unsimplified multi-path LP to

$$\sum_{a_{1:i} \in T} D_\Phi(a_{1:i})p(a_{1:i}) \leq 1 - \left(1 - \sum_{a_{1:i} \in T} q(a_{1:i}) \right)^K \quad \forall T \in \mathcal{A}(\mathcal{V}^{\leq L}). \tag{51}$$

This covers all conditions, so all that remains is to show that the objective reduces to the desired expression. This holds by interchanging the order of summation, and using the telescoping identity:

$$\sum_{i=1}^{L+1} \sum_{a_{1:i} \in \mathcal{V}^i} iR_\Phi(a_{1:i}) = \sum_{i=1}^{L+1} \sum_{j=0}^{i-1} \sum_{a_{1:i} \in \mathcal{V}^i} R_\Phi(a_{1:i}) \tag{52}$$

$$= \sum_{j=0}^{L} \sum_{i=j+1}^{L+1} \sum_{a_{1:i} \in \mathcal{V}^i} R_\Phi(a_{1:i}) \tag{53}$$

$$= \sum_{j=0}^{L} \sum_{i=j+1}^{L+1} \sum_{a_{1:j} \in \mathcal{V}^j} \sum_{a_{j+1:i} \in \mathcal{V}^{j-i}} R_\Phi(a_{1:i}) \tag{54}$$

$$= \sum_{j=0}^{L} \sum_{a_{1:j} \in \mathcal{V}^j} \sum_{i=j+1}^{L+1} \sum_{a_{j+1:i} \in \mathcal{V}^{j-i}} R_\Phi(a_{1:i}) = \sum_{a_{1:j} \in \mathcal{V}^{\leq L}} D_\Phi(a_{1:j})p(a_{1:j}). \tag{55}$$

This completes the reduction to the desired form for the multi-path LP. □

B Proof of Single-Path Information-Agnostic LP

Using the multi-path in Theorem 4.3, it is easy to prove the single-path LP in Theorem 3.3 by reducing it to the case $K = 1$. This works because a valid single-path verification algorithm is precisely the same as a valid K-path verification algorithm in the case $K = 1$.

Proof. It suffices to show that when $K = 1$, the multi-path information-agnostic LP from Theorem 4.3 reduces to the single-path LP. Indeed, the antichain condition becomes

$$\sum_{a_{1:i} \in T} D_\Phi(a_{1:i}) p(a_{1:i}) \leq \sum_{a_{1:i} \in T} q(a_{1:i}) \quad \forall T \in \mathcal{A}(\mathcal{V}^{\leq L}). \tag{56}$$

This forces the pointwise conditions $D_\Phi(a_{1:i})p(a_{1:i}) \leq q(a_{1:i})$, since by taking the singleton antichains $T = \{a_{1:i}\}$. Furthermore, these pointwise conditions imply the antichain conditions through summation. All other conditions and the objective function remain the same. Hence, we obtain the desired single-path LP. □

C Proof of Block Verification Optimality

Using the single-path LP, we can prove that block verification is optimal in the class of valid single-path verification algorithms (Theorem 3.4). We will use results from Sun et al. (2024b). The idea is to show that $D_{\Phi_{BV}}$ for the single-path block verification algorithm Φ_{BV} are tight in the single-path LP.

Proof. From Lemma 4 in Appendix B of Sun et al. (2024b), we have for each path $a_{1:i} \in \mathcal{V}^{\leq L}$ that

$$\mathbb{P}(\tau \geq i | X_{1:i} = a_{1:i}) = w(a_{1:i}) \tag{57}$$

under Φ_{BV}, where weights w are defined in Equation (3). Now, because Φ_{BV} is a valid verification algorithm, we can use the telescoping identity from the proof of Theorem 4.3 at the end of Appendix A (with $R_{\Phi_{BV}}$ defined similarly as the distribution of the output of Φ_{BV}) to show

$$\begin{aligned}
D_{\Phi_{BV}}(a_{1:i})p(a_{1:i}) &= \sum_{j=i+1}^{L+1} \sum_{a_{i+1:j} \in \mathcal{V}^{j-i}} R_{\Phi_{BV}}(a_{1:j}) && (58)\\
&= \sum_{j=i+1}^{L+1} \sum_{a_{i+1:j} \in \mathcal{V}^{j-i}} \mathbb{P}(\tau = j-1, X_{1:j-1} = a_{1:j-1}, Y = a_j) && (59)\\
&= \sum_{j=i+1}^{L+1} \mathbb{P}(\tau = j-1, X_{1:i} = a_{1:i}) = \mathbb{P}(\tau \geq i, X_{1:i} = a_{1:i}) && (60)\\
&= \mathbb{P}(\tau \geq i | X_{1:i} = a_{1:i}) \mathbb{P}(X_{1:i} = a_{1:i}) = w(a_{1:i}) q(a_{1:i}). && (61)
\end{aligned}$$

Thus, to show block verification is optimal, it suffices to show that for any valid single-path verification algorithm Φ and path $a_{1:i} \in \mathcal{V}^{\leq L}$, we have $D_\Phi(a_{1:i})p(a_{1:i}) \leq w(a_{1:i})q(a_{1:i})$, as then the objective value for Φ (the block efficiency, i.e. the sum of these $D_\Phi p$ terms) cannot exceed the objective value for Φ_{BV} (the sum of the $D_{\Phi_{BV}} p$ terms) in the single-path LP. The proof is by induction on i. For the base case $i = 0$, this is clear as $w(a_{1:0})q(a_{1:0}) = 1 = D_\Phi(a_{1:0})p(a_{1:0})$. Now, suppose this statement holds for i. To prove it for $i + 1$, we use the single-path conditions to obtain

$$\begin{aligned}
D_\Phi(a_{1:i+1})p(a_{1:i+1}) &\leq D_\Phi(a_{1:i})p(a_{1:i+1}) && (62)\\
&= D_\Phi(a_{1:i})p(a_{1:i})p(a_{i+1}|a_{1:i}) && (63)\\
&\leq w(a_{1:i})q(a_{1:i})p(a_{i+1}|a_{1:i}) && (64)\\
&= q(a_{1:i+1}) \cdot \frac{p(a_{i+1}|a_{1:i})}{q(a_{i+1}|a_{1:i})} \cdot w(a_{1:i}) && (65)\\
&\leq q(a_{1:i+1}) \cdot \min\left\{1, \frac{p(a_{i+1}|a_{1:i})}{q(a_{i+1}|a_{1:i})} \cdot w(a_{1:i})\right\} = q(a_{1:i+1}) w(a_{1:i+1}). && (66)
\end{aligned}$$

The last inequality holds by using the pointwise $D_\Phi p \leq q$ bound from the single-path LP. This completes the induction, and thus the proof that block verification is optimal.

□

D Proof of Path Selection Decomposition

In this section, we prove Lemma 4.4, which states that any valid K-path verification algorithm can be decomposed into a randomized path selection rule which chooses one of the K paths, and then a valid single-path verification algorithm that operates on an skewed draft distribution.

Proof. Let Φ denote the valid verification algorithm. This takes in K paths $P_1, \dots, P_K$ i.i.d. sampled from $q(\cdot)$, and samples a nonempty path in $\mathcal{V}^{\leq L+1}$ using a conditional distribution $\pi(\cdot|P_1, \dots, P_K)$, which can depend on off-path target and draft distribution values. This is constrained to only output $(X_{1:\tau}, Y)$ where $X_{1:\tau}$ is a prefix of one of $P_1, \dots, P_K$ and $Y \in \mathcal{V}$. Thus, we can conditionally factorize π into a prefix selection rule π_1 on prefixes of $P_1, \dots, P_K$ conditioned over $P_1, \dots, P_K$, and an additional token rule π_2 on $\mathcal{V}$ conditioned over $P_1, \dots, P_K, X_{1:\tau}$:

$$\pi(X_{1:\tau}, Y|P_1, \dots, P_K) = \pi_1(X_{1:\tau}|P_1, \dots, P_K)\pi_2(Y|P_1, \dots, P_K, X_{1,\tau}). \tag{67}$$

For each prefix $a_{1:i}$ of some path in $P_1, \dots, P_K$, denote its multiplicity $m(a_{1:i}) \geq 1$ as the number of paths that contain it as a prefix. For example, $m(\emptyset) = K$. Then, we define a path selection rule π_3 on $[K]$ conditioned over $P_1, \dots, P_K$, as:

$$\pi_3(k_0|P_1, \dots, P_K) = \sum_{i=0}^{L} \pi_1((P_{k_0})_{1:i}|P_1, \dots, P_K)m((P_{k_0})_{1:i})^{-1}. \tag{68}$$

To see this is a valid probability distribution, observe that for each distinct $a_{1:i}$ appearing as a prefix of one of $P_1, \dots, P_K$, when we sum the above formula over all $k_0 \in [K]$, we have exactly $m(a_{1:i})$ terms in the sum above with $(P_{k_0})_{1:i} = a_{1:i}$, each of which are $\pi_1(a_{1:i}|P_1, \dots, P_K)m(a_{1:i})^{-1}$. Thus, the terms containing $a_{1:i}$ cancel to $\pi_1(a_{1:i}|P_1, \dots, P_K)$, and summing this over all *distinct* $a_{1:i}$ gives a result of 1, because π_1 is a valid distribution. Also, define a prefix selection rule π_4 on prefixes of P_{k_0} conditioned over $P_1, \dots, P_K, k_0 \in [K]$:

$$\pi_4((P_{k_0})_{1:i}|P_1, \dots, P_K, k_0) = \frac{\pi_1((P_{k_0})_{1:i}|P_1, \dots, P_K)m((P_{k_0})_{1:i})^{-1}}{\sum_{i=0}^{L} \pi_1((P_k)_{1:i}|P_1, \dots, P_K)m((P_k)_{1:i})^{-1}}, \tag{69}$$

which is also a valid probability distribution. We see that when sampling $k_0 \sim \pi_3(\cdot|P_1, \dots, P_k)$ and then $X_{1:\tau} \sim \pi_4(\cdot|P_1, \dots, P_K, k_0)$, we have

$$\begin{aligned}
&\mathbb{P}(X_{1:\tau} = a_{1:i}|P_1, \dots, P_K) && (70)\\
&= \sum_{k_0 \in [K]} \pi_3(k_0|P_1, \dots, P_k)\pi_4(a_{1:i}|P_1, \dots, P_K, k_0) && (71)\\
&= \sum_{k_0 \in [K], (P_{k_0})_{1:i} = a_{1:i}} \pi_3(k_0|P_1, \dots, P_k)\pi_4(a_{1:i}|P_1, \dots, P_K, k_0) && (72)\\
&= \sum_{k_0 \in [K], (P_{k_0})_{1:i} = a_{1:i}} \pi_1(a_{1:i}|P_1, \dots, P_K)m(a_{1:i})^{-1} && (73)\\
&= m(a_{1:i})\pi_1(a_{1:i}|P_1, \dots, P_K)m(a_{1:i})^{-1} = \pi_1(a_{1:i}|P_1, \dots, P_K). && (74)
\end{aligned}$$

Thus, sampling from π_3 and then π_4 is equivalent to sampling from π_1, so sampling from π is equivalent to sampling in the order π_3, π_4, π_2. Now, let π_5 be the distribution formed by sampling from π_4 and then π_2. Then sampling from π is equivalent to sampling in the order π_3, π_5, where π_3 is a path selection rule conditioned on $P_1, \dots, P_K$, and π_5 returns $(X_{1:\tau}, Y)$ conditioned only the selected path and $P_1, \dots, P_K$. Now, denote q^{π_3} as the true distribution of the path select by π_3. Then the output of Φ follows the distribution induced by sampling a *single* path from q^{π_3}, and sampling from π_5 conditioned only on that single path (the conditioning over $P_1, \dots, P_K$ will cancel). To meet target-matching and prefix-matching, π_5 must correspond to a valid single-path verification algorithm with draft distribution q^{π_3} and target distribution p. This completes the proof of the decomposition into a path selection rule π_3 and a valid single-path verification algorithm with draft values being the true distribution of the π_3-selected path. □

E Proof of Skewed Draft Distribution Feasibility

Here, we prove Lemma 4.5, which describes when a distribution can be realized as a skewed draft distribution q^Γ from a given draft q and path count K. The proof is a direct application of Lemma A.1.

Proof. Define the bipartite graph G with left vertices $(\mathcal{V}^L)^K$, right vertices $\mathcal{V}^L$, and edges E consisting of pairs $((P_1, \dots, P_K), P_k)$ for all $P_1, \dots, P_K \in \mathcal{V}^L$ and $k \in [K]$. Our distribution over left vertices is i.i.d. sampling from q, and our distribution over right vertices is r. We would like to find when there is a joint distribution on $(\mathcal{V}^L)^K \times \mathcal{V}^L$ supported on E, whose marginals are the left and right vertex distributions. Lemma A.1 shows such a distribution exists if and only if

$$\sum_{P \in S} r(P) \le \sum_{(P_1,\dots,P_K) \in N(S)} \prod_{k=1}^{K} q(P_k) \quad \forall S \subseteq \mathcal{V}^L. \tag{75}$$

Here, $N(\cdot)$ denotes neighborhoods in G. Thus, $N(S)$ consists of all $(P_1, \dots, P_K)$ where some $P_k \in S$. This means the complement of $N(S)$ is precisely $(\mathcal{V}^L \setminus S)^K$, so the above becomes

$$\sum_{P \in S} r(P) \le 1 - \left(\sum_{P \in \mathcal{V}^L \setminus S} q(P) \right)^K \quad \forall S \subseteq \mathcal{V}^L. \tag{76}$$

We are almost at our desired condition, except we need to show this holds for all antichains $S \in \mathcal{A}(\mathcal{V}^L)$. Indeed, for an antichain S, consider the set $P_S \subseteq \mathcal{V}^L$ of length-L paths with a prefix in the antichain. No path in P_S can have two distinct elements in S as a prefix. Thus, the mass of q over S is the same as the mass of q over P_S:

$$\sum_{a_{1:i} \in S} q(a_{1:i}) = \sum_{a_{1:i} \in S} \sum_{a_{i+1:L} \in \mathcal{V}^{L-i}} q(a_{1:L}) = \sum_{a_{1:L} \in P_S} q(a_{1:L}) \tag{77}$$

The same holds for r, because it is also a distribution over $\mathcal{V}^L$. Thus, the antichain conditions reduce to the conditions for $S \subseteq \mathcal{V}^L$ above, as desired. □

F Proof of Optimal Multi-Path Algorithm Description

Now, we prove Theorem 4.6, which describes the optimal multi-path valid verification algorithm as a solution to a nonlinear optimization problem.

Proof. The first part of the theorem is a direct consequence of the fact that block verification is the optimal single-path valid verification algorithm (Theorem 3.4), and any valid multi-path verification algorithm can be decomposed into path selection and valid single-path verification (Lemma 4.4). Now, we prove the second part of the theorem, using this description. Recall from the proof of Lemma 4.5 in Appendix E that a distribution q^Γ can be realized from a given draft q and path count K with a path selection rule if and only if

$$q^\Gamma(a_{1:L}) \le 1 - \left(1 - \sum_{a_{1:L} \in T} q(a_{1:L})\right)^K \quad \forall T \subseteq \mathcal{V}^L. \tag{78}$$

This covers the feasibility condition, so all that remains is to show the objective function in the theorem equals the block efficiency for block verification run on q^Γ. By using the expression for $D_{\Phi_{BV}} p$ in the proof of Theorem 3.4 in Appendix C, the objective value in the single-path LP (Theorem 3.3) simplifies to

$$\sum_{a_{1:i} \in \mathcal{V}^{\le L}} D_{\Phi_{BV}}(a_{1:i}) p(a_{1:i}) = \sum_{a_{1:i} \in \mathcal{V}^{\le L}} w(a_{1:i}) q^\Gamma(a_{1:i}), \tag{79}$$

where the weights w are defined as in Equation (3), but with q replaced by q^Γ. Thus, all that remains is to show that

$$w(a_{1:i}) q^\Gamma(a_{1:i}) = \min_{0 \le k \le i} p(a_{k+1:i} | a_{1:k}) q^\Gamma(a_{1:k}). \tag{80}$$

The proof is by induction on i. For $i = 0$, this is clear as both sides are one. Now, suppose this statement holds for i. To show it is true for $i + 1$, observe that

$$w(a_{1:i+1})q^\Gamma(a_{1:i+1}) = \min\left\{1, \frac{p(a_{i+1}|a_{1:i})}{q^\Gamma(a_{i+1}|a_{1:i})} w(a_{1:i})\right\} q^\Gamma(a_{1:i+1}) \tag{81}$$

$$= \min\left\{q^\Gamma(a_{1:i+1}), p(a_{i+1}|a_{1:i})q^\Gamma(a_{1:i})w(a_{1:i})\right\} \tag{82}$$

$$= \min\left\{q^\Gamma(a_{1:i+1}), p(a_{i+1}|a_{1:i}) \cdot \min_{0\le k\le i} p(a_{k+1:i}|a_{1:k})q^\Gamma(a_{1:k})\right\} \tag{83}$$

$$= \min\left\{q^\Gamma(a_{1:i+1}), \min_{0\le k\le i} p(a_{k+1:i+1}|a_{1:k})q^\Gamma(a_{1:k})\right\} \tag{84}$$

$$= \min_{0\le k\le i+1} p(a_{k+1:i+1}|a_{1:k})q^\Gamma(a_{1:k}). \tag{85}$$

This completes the proof. □

G Proof of Optimal Multi-Path Lower Bound

We now prove Lemma 4.7, which lower bounds the objective value in Theorem 4.6.

Proof. Denote the minimum value of q^Γ/p as

$$\epsilon = \min_{a_{1:i}\in\mathcal{V}^{\le L}} \frac{q^\Gamma(a_{1:i})}{p(a_{1:i})}. \tag{86}$$

Then for each $a_{1:i} \in \mathcal{V}^{\le L}$ and $0 \le k \le i$, we have the lower bound

$$p(a_{k+1:i}|a_{1:k})q^\Gamma(a_{1:k}) = \frac{q^\Gamma(a_{1:k})}{p(a_{1:k})} \cdot p(a_{1:i}) \ge \epsilon \cdot p(a_{1:i}). \tag{87}$$

Thus, we have the desired lower bound on the objective value:

$$\sum_{a_{1:i}\in\mathcal{V}^{\le L}} \min_{0\le k\le i} p(a_{k+1:i}|a_{1:k})q^\Gamma(a_{1:k}) \ge \sum_{a_{1:i}\in\mathcal{V}^{\le L}} \epsilon \cdot p(a_{1:i}) = (L+1)\epsilon. \tag{88}$$

□

H Greedy Polymatroid Connection

Here, we describe the connection between greedy multi-path valid verification algorithms (Definition 5.1) and the greedy polymatrid algorithm. We start with Theorem 4.6, but only consider the feasibility condition:

$$q^\Gamma(a_{1:L}) \le 1 - \left(1 - \sum_{a_{1:L}\in T} q(a_{1:L})\right)^K \quad \forall T \subseteq \mathcal{V}^L. \tag{89}$$

As previously mentioned, the right hand side is a submodular function $\psi(T)$ in T. Note that the inequality must also be an equality at $T = \mathcal{V}^L$ to ensure that q^Γ is a probability distribution. Thus, to find *some* feasible q^Γ satisfying these submodular constraints, we can use the greedy polymatroid algorithm from Schrijver et al. (2003), which aims to maximize the sum of q^Γ terms given the above constraints. First, we fix any ordering $\{P_1, \ldots, P_M\}$ on $\mathcal{V}^L$. Then, a feasible solution is

$$q^\Gamma(P_i) = \psi(\{P_1, \ldots, P_i\}) - \psi(\{P_1, \ldots, P_{i-1}\}). \tag{90}$$

This is precisely what Equation (19) reduces to when the path selection rule Γ is induced by the *reversed* global ordering $\{P_M, \ldots, P_1\}$. Thus, greedy multi-path valid verification algorithms are simply outputs of the greedy polymatroid algorithm for the multi-path LP feasibility conditions.

I Efficient Computation of q^Γ

We can compute each $q^{\Gamma_g}(a_{1:i})$ with Equation (19), where $\preceq, \prec$ mean precedence and strict precedence in the global path ordering, i.e. $b_{1:i} \prec a_{1:i}$

$$q^\Gamma(a_{1:i}) = \left(\sum_{b_{1:i} \preceq a_{1:i}} q(b_{1:i})\right)^K - \left(\sum_{b_{1:i} \preceq a_{1:i}} q(b_{1:i})\right)^K \quad \forall a_{1:i} \in \mathcal{V}^i. \tag{91}$$

However, this is not efficient to compute in its direct form, as it involves a sum over up to $|\mathcal{V}|^L$ paths. Thus, we reduce it to Equation (92) below. This holds by the definition of the lexicographic ordering, the second sum is the q-mass of all paths $b_{1:L}$ where $b_{1:j} = a_{1:j}$ and $\pi_{a_{1:j}}(b_{j+1}) < \pi_{a_{1:j}}(a_{j+1})$ for some $0 \le j \le i-1$, and the first sum also contains the q-mass of all paths $b_{1:L}$ with $b_{1:i} = a_{1:i}$:

$$q^{\Gamma_g}(a_{1:i}) = \left(q(a_{1:i}) + \sum_{j=0}^{i-1} q(a_{1:j}) \sum_{t \in \mathcal{P}_{a_{1:j}}(a_{j+1})} q(t|a_{1:j})\right)^K - \left(\sum_{j=0}^{i-1} q(a_{1:j}) \sum_{t \in \mathcal{P}_{a_{1:j}}(a_{j+1})} q(t|a_{1:j})\right)^K. \tag{92}$$

In fact, given a fixed $a_{1:i-1}$, we can efficiently compute all $q^{\Gamma_g}(a_{1:i})$ for $a_i \in \mathcal{V}$. Using the above expression, these $|\mathcal{V}|$ quantities have the exact same terms except at $q(a_{1:i})$ and the $j = i-1$ term in the summation. We can compute the former over all $a_i \in \mathcal{V}$ in $O(|\mathcal{V}|)$ time by multiplying the conditional distribution $q(\cdot|a_{1:i-1})$ by $q(a_{1:i-1})$. The latter is a sum of $q(t)$ over all $t \in \mathcal{V}$ with $\pi_{a_{1:i-1}}(t) < \pi_{a_{1:i-1}}(a_i)$, so we can use a cumulative sum over the ordering on $\mathcal{V}$ induced by $\pi_{a_{1:i-1}}$. From these $|\mathcal{V}|$ values, we can compute conditional probabilities $q^{\Gamma_g}(\cdot|a_{1:i})$ using Equation (14). In our actual GBV algorithm, we only need to compute the distributions for prefixes $a_{1:i}$ of a single path, so there are $L+1$ such computations, leading to the overall time estimate $O(|\mathcal{V}| \log |\mathcal{V}|)$.

J Motivating the Ratio Ordering

We now theoretically motivate the local ordering chosen at the end of Appendix 5,

$$\pi_{a_{1:i}}(x) = \frac{p(x|a_{1:i})}{q(x|a_{1:i})}. \tag{93}$$

From Equation (19), and the convexity of $X \mapsto X^K$, we see $q^\Gamma(P_m)/q(P_m)$ increases along the global ranking $m = 1, \dots, M$. Hence, to make $q^\Gamma \approx p$, which yields near-optimal block efficiency as remarked at the end of Appendix 4, we would also like p/q to increase along the global ranking. Since enforcing this requires the full joint p_L and q_L, we further relax this to a tree-based ranking by making each local ordering follow p/q in increasing order. When $L = 1$, this global-to-local relaxation is not required, so approximation quality can be higher.

K Full GBV Algorithm

Here, we present the full GBV algorithm. Note that while we use $\pi_{a_{1:i}}(\cdot) = p(\cdot|a_{1:i})/q(\cdot|a_{1:i})$ in our experiments, this algorithm allows one to define arbitrary local orderings, and it will still work.

L Other Model Families and Datasets

Here, we expand our original evaluations for OPT-6.7B/350M/125M on GSM8K, HumanEval, and MATH500. We broaden our model family coverage to include pairs Qwen-3 32B/0.6B and Llama-3 70B/8B, which allows us to test the efficacy of our approach on much larger target models. Due to memory constraints, the Llama-3 experiments were run on a Paperspace machine with 4xA6000 and an Intel Xeon Gold 5315Y CPU (32 cores, 3.20 GHz). We further expand the Qwen-3 dataset coverage to include 500 prompts from MGSM and ToolBench (Shi et al., 2022; Qin et al., 2023), to measure performance on diverse, non-academic tasks. Our block efficiency results for $L = 8$ and temperature 1.0 target sampling are shown in Table 3, and walltime numbers are in Table 4.

Interestingly, for Qwen-3, the single-path block verification baseline, i.e. $K = 1$, is most effective. On GSM8K, block efficiency is 4.32 tokens/M_p-call at $K = 1$, but decreases to 3.91, 3.57, and

Algorithm 1 Greedy Multi-Path Block Verification

Require: draft paths $X[1]_{1:L}, \ldots, X[K]_{1:L}$, target probabilities $p(\cdot|X[k]_{1:i})$, draft probabilities $q(\cdot|X[k]_{1:i})$, local orderings $\pi_{X[k]_{1:i-1}}(\cdot)$.

for $k = 1, \ldots, K$ **do**
  $O_k \leftarrow (\pi_{X[k]_{1:i-1}}(X[k]_i))_{i=1}^{L}$
end for
$k_0 \leftarrow \arg\max_k O_k$ under lexicographic ordering
for $i = 1, \ldots, L$ **do**
  Compute $q^{\Gamma_g}(\cdot|X[k_0]_{1:i-1})$ using $\{p(\cdot|X[k]_{1:i-1}), q(\cdot|X[k]_{1:i-1})\}_{k=1}^{K}$ as in Appendix I
end for
return BV(path $X[k_0]_{1:L}$; target probs $p(\cdot|X[k_0]_{1:i-1})$; draft probs $q^{\Gamma_g}(\cdot|X[k_0]_{1:i-1})$)

3.43 for $K = 2, 3, 4$ respectively, while walltime increases from 79.6 to 92.8, 100.5, and 106.1 ms/token. Similar trends hold across HumanEval, MATH500, MGSM, and ToolBench. Likewise, for Llama-3, $K = 1$ again appears to be the optimal setting. Nevertheless, unlike Qwen-3, there can be meaningful gains at higher K. On GSM8K, the walltime decreases by nearly 10% from $K = 3$ to $K = 4$.

Table 3: We compute the block efficiency (tokens/call) of GBV, for target-draft model pairs Qwen-3 32B/0.6B and Llama-3 70B/8B on up to 500 prompts from each of the datasets GSM8K, HumanEval, and MATH500, MGSM, and ToolBench (only the first two for Llama-3). Larger numbers are better. In almost all settings, efficiency decreases as K increases from 1 (block verification baseline) to 4.

Model pair	**Dataset**	**Block efficiency** (tokens/M_p-call)			
		$K = 1$	$K = 2$	$K = 3$	$K = 4$
Qwen-3 32B/0.6B	GSM8K	**4.322**	3.910	3.571	3.429
	HumanEval	**4.204**	3.423	3.331	2.960
	MATH500	**4.342**	3.917	3.693	3.411
	MGSM	**4.329**	3.973	3.468	3.445
	ToolBench	**3.090**	2.706	2.791	2.681
Llama-3 70B/8B	GSM8K	**4.943**	4.430	4.271	4.625
	HumanEval	**5.993**	4.806	4.473	4.370

Table 4: We compute the walltimes (ms/token) of GBV, for target-draft model pairs Qwen-3 32B/0.6B and Llama-3 70B/8B on up to 500 prompts from each of the datasets GSM8K, HumanEval, and MATH500, MGSM, and ToolBench (only the first two for Llama-3). Smaller numbers are better. In almost all settings, walltime increases as K increases from 1 (block verification baseline) to 4.

Model pair	**Dataset**	**Walltime** (ms/token)			
		$K = 1$	$K = 2$	$K = 3$	$K = 4$
Qwen-3 32B/0.6B	GSM8K	**79.638**	92.823	100.547	106.123
	HumanEval	**84.166**	106.051	109.215	124.554
	MATH500	**81.517**	91.133	97.880	106.575
	MGSM	**80.992**	90.312	102.296	105.387
	ToolBench	**113.563**	132.477	128.405	133.973
Llama-3 70B/8B	GSM8K	**114.123**	133.188	136.272	123.545
	HumanEval	**96.063**	124.540	131.889	135.186

In our Llama-3 70B/8B experiments, we profiled draft tree size and saw that it grows roughly linearly with K, at around $9, 16, 24, 29$ for $K = 1, 2, 3, 4$. Across the K sweep, the fraction of walltime spent in draft and target passes remains nearly constant (around 50% draft and 40% target), and the peak target and draft KV-cache footprints remain in a narrow band (target is around 80 MB on GSM8K and 110 MB on HumanEval, with the draft being around 40 MB for both).

This shows that increasing K scales FLOPs and KV traffic in proportion to the larger tree, rather than changing the draft bottleneck. Even if block efficiency is high, end-to-end walltime degrades with K.

For Qwen-3 32B/0.6B, we performed the same profiling with similar results. Again, draft tree size grows approximately linearly with K. The peak KV-cache footprint also stays in a narrow band as K increases, with the target ranging from roughly 60 to 120 MB across datasets and the draft from around 30 to 50 MB. Further, draft and target pass times also remain nearly constant relative to the walltime. However, compared to Llama-3, the key difference is that 70% of time is spent in drafting and only around 20% is in the batched target forward pass, with only small variations across K.

The target forward pass dominates the cost for Llama-3 significantly more than for Qwen-3. We also observe worse walltimes for the same block efficiencies. For example, Qwen-3 on ToolBench sees essentially the same walltimes for $K = 1$ and $K = 2$ as Llama-3 on GSM8K, even though Llama-3 has significantly higher block efficiencies than Qwen-3 (around 1.9 and 1.7 higher tokens/M_p-call for $K = 1$ and $K = 2$). Differences in hardware setups may additionally explain and influence these relative target and draft costs.

M Tuning Generation Length and Heuristics

We alter the generation length and local orderings in our Qwen-3 experiments from Appendix L. In Table 5, we compute block efficiencies for 1024-token generations on the same MATH500 as before; previous experiments were with 128-token generation. We evaluate various local ordering rules: p/q is what we used in our default experiments, but $p, q, \min(1, p/q), q/p$ are also valid rules. In particular, we choose p, q to see if simple target-based skewing or draft-based skewing works as well as our theoretically motivated p/q heuristic. Also, $\min(1, p/q)$ is a natural ordering choice as the acceptance rate for individual tokens in standard speculative sampling. Finally, we choose q/p to test if ordering by p/q pushes the skewed draft in the right direction. We test temperature 1.0 and 0.2, and $K = 1, \ldots, 4$. Note that for each temperature, all heuristics have the same entry for $K = 1$, as GBV with any heuristic reduces to block verification for $K = 1$.

We find that the original trend, where higher K reduced block efficiency, now flips: there are meaningful gains up to $K = 4$ for the original heuristic p/q, as well as for other heuristics such as $\min(1, p/q)$ at temperature $t = 1.0$ and p at temperature $t = 0.2$. In fact, at this increased generation length, GBV beats the BV baseline for all heuristics except q/p, with maximum block efficiency gains around 20%. The fact that q/p performs worst corroborates our heuristic justification for the p/q ordering: flipping the order significantly degrades block efficiency and walltime.

Table 5: Block efficiencies for Qwen-3 32B/0.6B on MATH500 with max-1024-token generations.

Heuristic	**Temp. 1.0**				**Temp. 0.2**			
	$K=1$	$K=2$	$K=3$	$K=4$	$K=1$	$K=2$	$K=3$	$K=4$
p/q	4.861	5.785	5.833	**5.939**	5.293	5.493	5.958	**6.197**
$\min(1, p/q)$	4.861	5.634	5.854	**5.965**	5.293	5.576	5.893	**6.207**
p	**4.861**	5.408	5.679	5.615	5.293	5.406	5.942	**6.218**
q	4.861	**5.065**	4.945	4.644	5.293	5.164	5.398	**5.499**
q/p	**4.861**	4.148	3.804	3.570	**5.293**	3.988	3.648	3.383

The walltime numbers in Table 6 show a similar trend. From these numbers, it is apparent that ordering by p/q roughly follows the right direction: ordering by the reciprocal ratio q/p significantly decreases block efficiency and increases walltime. Nevertheless, p/q is no longer the best heuristic:

Table 6: Walltimes for Qwen-3 32B/0.6B on MATH500 with max-1024-token generations.

Heuristic	**Temp. 1.0**				**Temp. 0.2**			
	$K=1$	$K=2$	$K=3$	$K=4$	$K=1$	$K=2$	$K=3$	$K=4$
p/q	61.281	59.385	**58.565**	58.621	57.266	63.418	58.416	**56.437**
$\min(1, p/q)$	61.281	54.279	52.429	**52.013**	57.266	55.096	51.841	**50.214**
p	61.281	55.198	**53.240**	53.903	57.266	56.037	50.628	**48.349**
q	61.281	**59.662**	61.024	64.810	57.266	58.017	56.031	**55.016**
q/p	**61.281**	72.371	77.994	82.896	**57.266**	75.652	82.950	90.121

the variant $\min(1, p/q)$, based on the Naive OTLP acceptance rate, is best at temperature $t = 1.0$, while simply ordering by p is best at temperature $t = 0.2$.

Finally, we note that the p/q rule can perform worse than alternative heuristic rules like $\min(1, p/q)$ and p. For example, $\min(1, p/q)$ is the best at temperature 1.0 for Qwen, and p is the best for Qwen at temperature 0.2. In fact, both of these heuristics achieve their best walltime at the maximum paths $K = 4$ in these settings, so increasing K consistently helps them. Each heuristic also performs quite well in the others' best temperatures setting. This suggests that by altering heuristics, model-specific deficits can be overcome.

N GBV-Aligned Distillation

We now perform GBV-aligned to improve our Qwen results. We fine-tune the draft Qwen-3 0.6B on 7,469 GSM8K-train trajectories generated by Qwen-3 32B (temperature 1.0, generation length 1024). We train for 3 epochs with batch size 1, gradient accumulation 4, learning rate 2×10^{-5}, and held-out val fraction 0.1. The training objective is the L_1 distance between the target distribution and the corresponding skewed draft q^{Γ} at the configured K, with the local ordering at each tree node given by either p/q or q/p. All inference runs use $L = 8$, temperature 1.0, a generation length of 128, and matched-direction GBV, i.e. we evaluate on the same K and local ordering that we trained on. Table 7 shows our results on a held-out 100-prompt GSM8K subset.

Table 7: We compute the block efficiency (tokens/M_p-call), walltime (ms/token), and throughput (tokens/s) for GBV-distilled draft models on GSM8K, with matched-direction GBV inference.

	$K=1$			$K=2$			$K=3$		
Draft model	**BE**	**WT**	**TP**	**BE**	**WT**	**TP**	**BE**	**WT**	**TP**
GBV-q/p	5.355	65.140	15.354	4.112	87.512	11.427	3.567	100.564	9.944
distilled	5.373	65.193	15.342	**4.354**	**82.171**	**12.170**	**4.385**	**81.291**	**12.301**
GBV-p/q	5.676	62.308	16.049	**4.774**	**76.339**	**13.099**	**4.072**	**90.421**	**11.059**
distilled	5.576	62.699	15.949	4.292	84.985	11.767	3.661	100.086	9.991

We see that distillation on GBV-q/p improves the untuned baseline at every $K > 1$, with the $K = 3$ gap being the largest: block efficiency rises from 3.567 to 4.385 (+22.9%) and throughput rises from 9.94 to 12.30 tokens/s. At $K = 1$, the skewed draft q^{Γ} reverts to q, so both training directions reduce to standard target-draft L_1 distillation and land within noise of the baseline. We also see that distillation on GBV-p/q fails to improve at any $K > 1$, underperforming compared to the untuned baseline at $K = 2$ and $K = 3$ by around 10% in block efficiency.

We hypothesize that this inversion arises from how the skewed draft allocates mass: q^{Γ} concentrates mass in the tail of the global ranking. Under the default p/q ordering, that tail consists of high-p/q tokens, so the alignment objective pushes the draft to spread mass onto tokens it was previously underpredicting; under the inverted q/p order, the tail consists of low-p/q tokens that the draft over-predicts, so the objective reinforces the draft's existing tendencies. We confirm this directly by measuring the average per-token softmax entropy of the trained drafts at $K = 3$ on the first 30

held-out GSM8K prompts: the untuned Qwen-3 0.6B baseline averages 1.92 nats, the p/q-aligned draft model averages 2.74 nats ($+43\%$, flatter), and the q/p-aligned draft model averages 0.92 nats (-52%, much sharper). Empirically, the latter sharpening behavior produces a more decoding-efficient draft. The L_1 objective minimizes pointwise distance to a feasibility upper bound on per-token acceptance, but our experiments suggest this is not equivalent maximizing the realized token acceptance.

O Temperature Ablations

To measure the impact of temperature on walltime speedups in GBV, we evaluate GBV on Qwen-3 32B/0.6B for GSM8K (500 prompts) with temperatures 0.2, 0.4, 0.6, 0.8, also including the temperature 1.0 run from Table 3. Our results are shown in Table 8.

Importantly, we find that GBV performs better at low temperature settings, which directly opposes block verification trends. Whereas standard block verification (the $K = 1$ rows) walltimes increase from under 80 ms/token to over 90 ms/token from temperatures 0.2 to 1.0, GBV with $K = 3$ sees its walltimes decrease from over 100 ms/token at temperature 1.0 to under 86 ms/token for lower temperatures (0.2, 0.4, 0.6), with its highest throughput at temperature 0.6. These results also show that temperature 1.0 sampling achieves over 10% lower throughput than all other GBV $K = 3$ settings, so our results in Appendix 6 actually reflect GBV in its worst-performing setting.

Even as we noted in Appendix L and Table 3 that standard BV may outperform GBV for Qwen-3 32B/0.6B, this finding no longer holds at low temperatures. For example, GBV with $K = 3$ achieves a throughput of around 5.6% higher tokens/s than BV when the target temperature is 0.2. Even at slightly higher temperatures of 0.4 and 0.6, GBV achieves a higher block efficiency than BV, and achieves similar throughput. Thus, even for model pairs where GBV with $K > 1$ does not outperform BV at temperature 1.0 sampling, it may still be preferable over BV at low temperatures.

Table 8: We evaluate how temperature impacts the block efficiency, throughput, and walltime for GBV with $K = 1, 3$ when run on Qwen-3 32B/0.6B with GSM8K (500 prompts) and $L = 8$. We use temperature 0.2, 0.4, 0.6, 0.8, and 1.0 sampling from the target model. While BV ($K = 1$) performs best at high temperatures (0.8, 1.0), GBV ($K = 3$) works better at low temperatures (0.2, 0.4, 0.6).

K	M_p **Temp.**	**Block Efficiency** (tokens/M_p-call)	**Throughput** (tokens/s)	**Walltime** (ms/token)
1	0.2	3.867	11.034	90.626
1	0.4	4.313	12.407	80.598
1	0.6	4.308	12.368	80.851
1	0.8	**4.430**	**12.761**	**78.362**
1	1.0	4.322	12.557	79.638
3	0.2	4.219	11.661	85.759
3	0.4	4.333	11.911	83.954
3	0.6	**4.492**	**12.149**	**82.311**
3	0.8	4.161	11.341	88.173
3	1.0	3.571	9.946	100.547

P Draft Length Ablations

Here, we provide ablation experiments on L, the draft block length. We extend our $L = 8$ results for Qwen-3 32B/0.6B on GSM8K with temperature 1.0 target sampling to larger draft lengths: $L = 16$ and $L = 24$. Our results are shown in Table 9.

As L increases, block efficiency improves only modestly, while throughput degrades rapidly. For example, at $K = 1$, the block efficiency increases from 4.32 for $L = 8$ to 5.67 and 5.79 for $L = 16$ and $L = 24$, but the walltime grows from roughly 80 ms/token to 103 ms/token and 140 ms/token.

The effect is worse for $K \geq 2$, where moving from $L = 8$ to $L = 24$ more than doubles the walltime.

The draft and target costs also shift substantially: the draft time, as a percentage of the end-to-end walltime, grows from about 71% at $L = 8$ to 80% and 86% at $L = 16, 24$, while the target share drops to as low as 10%. Coupled with our analysis of the draft bottleneck in Appendix L, this indicates longer blocks increase aggregate cost by pushing more work into the draft bottleneck.

Table 9: We evaluate GBV with $1 \leq K \leq 4$ for Qwen-3 32B/0.6B temperature 1.0 sampling on GSM8K (500 prompts), over varying draft block lengths $L = 8, 16, 24$. We report block efficiency, walltime, and draft and target forward pass runtimes as percentages of end-to-end walltime. We find that around $L = 8$ is an ideal length for avoiding more pronounced draft bottlenecks.

L	K	**Block Efficiency** (tokens/M_p-call)	**Walltime** (ms/token)	**Draft Cost** (%walltime)	**Target Cost** (%walltime)
8	**1**	**4.322**	**79.638**	**71.5**	**21.3**
	2	3.910	92.823	71.3	21.0
	3	3.571	100.547	71.1	21.1
	4	3.429	106.123	70.8	21.3
16	**1**	**5.667**	**102.970**	**81.7**	**13.2**
	2	4.016	149.775	81.0	13.3
	3	3.915	155.406	80.5	13.6
	4	3.742	163.200	80.1	13.9
24	**1**	**5.794**	**140.411**	**86.0**	**9.9**
	2	4.275	196.937	85.0	10.2
	3	3.945	218.631	84.3	10.8
	4	3.628	240.142	83.4	11.5

Q Multi-Path Comparison

To empirically compare GBV to other multi-path methods, we implement six other speculative decoding algorithms in our code: standard speculative decoding (Leviathan et al., 2023; Chen et al., 2023), block verification (Sun et al., 2024b), NSS (Miao et al., 2024), SpecInfer (Miao et al., 2024), SpecTr (Sun et al., 2023), and Traversal Verification (Weng et al., 2025). SpecInfer provably improves NSS and block verification provably improves standard speculative decoding, so we only benchmark GBV against SpecInfer, SpecTr, and block verification.

We run comparison experiments for Llama-3 70B/8B on GSM8K (the same 500 prompts as previous experiments) for $L = 8$. We test $K = 2, 3$ as well as temperature $0.2, 1.0$ sampling from the target model. Our block efficiency, throughput, and walltime numbers are shown in Table 10. Due to poor preliminary decoding rates and Llama-3 70B memory footprint, we do not run SpecTr and SpecInfer, the worst-performing temperature 1.0 algorithms, at temperature 0.2.

For temperature 1.0 sampling, GBV and Traversal Verification perform significantly better than SpecTr and SpecInfer. For both $K = 2, 3$, GBV achieves over 1.1 token/s higher throughput than SpecTr and over 1.6 token/s higher than SpecInfer. While Traversal Verification does outperform GBV in these settings, coming out on top, the incremental gains are less significant: GBV achieves around 0.6 fewer token/s for $K = 2$ and 0.9 fewer token/s for $K = 3$.

On the other hand, at lower temperatures like 0.2, the trend flips: GBV now performs better than Traversal Verification. While throughput and block efficiency gains are negligible for $K = 3$ (only around 0.1 more tokens/M_p-call and 0.2 higher token/s), the benefits for $K = 2$ are significant: GBV achieves with over 1.6 higher token/s than Traversal Verification, and over 1.1 higher tokens/M_p-call. In fact, GBV at temperature 0.2 with $K = 2$ achieves the highest throughput by far out of all tested settings: no other method for Llama-3 70B/8B achieves over 10 tokens/s throughput. This suggests that GBV outperforms Traversal Verification for low temperatures and sharper p distributions.

Table 10: We compare GBV, SpecTr, SpecInfer, and Traversal for Llama-3 70B/8B on GSM8K (500 prompts). We report block efficiency and throughput (higher is better), and walltime (lower is better), for $K = 2, 3$ and temperature $0.2, 1.0$ target sampling. While Traversal outperforms other methods for temperature 1.0, GBV achieves over 1.5 tokens/s higher throughput at temperature 0.2.

M_p **Temp.**	K	**Method**	**Block Efficiency** (tokens/M_p-call)	**Throughput** (tokens/s)	**Walltime** (ms/token)
1.0	2	GBV	4.430	7.508	133.188
		SpecTr	3.716	6.355	157.365
		SpecInfer	3.259	5.603	178.480
		Traversal	**4.760**	**8.114**	**123.245**
1.0	3	GBV	4.271	7.338	136.272
		SpecTr	3.540	6.211	161.002
		SpecInfer	3.239	5.686	175.879
		Traversal	**4.779**	**8.260**	**121.065**
0.2	2	**GBV**	**6.730**	**11.415**	**87.602**
		Traversal	5.650	9.773	102.319
0.2	3	**GBV**	**5.663**	**9.775**	**102.299**
		Traversal	5.763	9.962	100.379

R INTEGRATION WITH EAGLE-3

GBV relies on the assumption that the draft tree is the union of i.i.d. sampled paths from the draft model, so it cannot be directly combined with deterministic drafting methods like EAGLE-2 (Li et al., 2024a). While EAGLE-2 dynamically uses draft confidence estimates to expand the tree, the tokens selected at each stage are chosen deterministically to be top-k candidates, not randomly; thus, GBV will not necessarily preserve the target distribution if run naively on EAGLE-2 trees.

EAGLE-3 (Li et al., 2025) uses the same context-aware dynamic tree, but its main contribution is to use direct token prediction and replace top-layer features with multi-layer features when training its draft model. While the EAGLE-2 construction is not compatible with GBV, this kind of draft model training is orthogonal to our work and can be combined with GBV, similar to how we performed GBV-aligned training in Appendix N. Such training also does not increase the drafting-side work, since that latency is largely a fixed function of the number K and length L of i.i.d. draft paths.

We leave training to future work, and take the simplest approach. For GBV with local orderings p and p/q, and Traversal Verification, we perform integration into EAGLE-3 by directly using the EAGLE-3 draft model to sample i.i.d. paths, unlike the deterministic formation in EAGLE-3, and then applying GBV or Traversal for verification. We use Vicuna1.3-13B as our target model, since it has an official EAGLE-3 draft; also, as a smaller model, it allows us to test the efficacy of GBV in higher TPS regimes. We evaluate on SpecBench (Xia et al., 2024). For GBV/Traversal, we choose $K = 3, L = 6$ after tuning. For EAGLE-3, their default tree expansion parameters are used, as these are already highly optimized for Vicuna. Even with this naive integration approach, we find in Table 11 that there are substantial benefits to combining the two at higher sampling temperatures.

Table 11: Block efficiency, measured in tokens per target-model call, for EAGLE-3 and EAGLE-3 combined with GBV/Traversal on SpecBench using Vicuna1.3-13B as the target model.

Temperature	**EAGLE-3**	**EAGLE-3 GBV-p**	**EAGLE-3 GBV-p/q**	**EAGLE-3 Traversal**
1.0	**5.53**	3.91	3.95	3.82
1.1	5.08	3.82	**3.87**	3.63
1.2	3.52	**3.69**	3.64	3.16
1.3	1.85	3.18	**3.23**	2.73
1.4	1.31	2.73	**2.76**	2.61

Table 12: Throughput, measured in tokens/s, for EAGLE-3 and EAGLE-3 combined with GBV/Traversal on SpecBench using Vicuna1.3-13B as the target model.

Temperature	EAGLE-3	EAGLE-3 GBV-p	EAGLE-3 GBV-p/q	EAGLE-3 Traversal
1.0	**90.81**	69.43	69.50	73.37
1.1	**81.99**	66.98	68.91	68.04
1.2	57.93	**63.56**	62.93	58.17
1.3	32.89	52.21	**60.38**	47.74
1.4	20.67	45.05	**52.06**	44.90

While EAGLE-3 is better at temperatures 1.0 and 1.1, its performance rapidly degrades at high temperatures. At temperature 1.3, it is roughly 30 tokens/s slower than GBV-p/q and 20 tokens/s slower than GBV-p, and it is nearly unusable at temperature 1.4 with 20 tokens/s throughput. We also note that GBV-p/q is significantly better than GBV-p, and especially Traversal, at achieving high throughput at high temperatures. This is interesting because our original results show that Traversal is better than GBV at high temperatures; however, for an EAGLE-3 draft, this no longer holds.

EAGLE-3 degrades at high temperatures because it uses NSS verification. This method does not use draft probabilities: it samples from the target distribution until it lands off the draft tree. The block efficiency of EAGLE-3 is thus purely a function of target mass on the draft tree. Since the EAGLE-3 draft model was trained to construct a sparse draft tree aligned with mostly high-probability target tokens, it is likely that the target mass at nodes decreases as temperature grows. This leads to EAGLE-3's massive block-efficiency drop from temperature 1.0 to 1.4, which is over 4 tokens/call. This is a limitation of the entire EAGLE family, since EAGLE-1/2/3 use deterministic draft trees, where only NSS is a valid verification method. We observed similar degradation for EAGLE runs.

GBV and Traversal degrade EAGLE-3 performance at temperatures 1.0 and 1.1 because EAGLE-3 was trained using direct top-k token prediction rather than feature regression. This means that draft probability tensors may not match up well with target probability tensors, especially deep in the draft tree. GBV and Traversal work best when p/q tensors are decently aligned with respect to L_1 distance, which may not be true for the trained EAGLE-3 draft model at temperatures 1.0 and 1.1 in our setup.

We offer two promising directions for future work to mitigate the above. First, one could abandon EAGLE-3 in favor of an EAGLE (Li et al., 2024b) draft. While EAGLE performs feature regression and avoids the aforementioned direct-token-prediction issue, it causes diverging p/q deeper in the tree. Training an L-lookahead EAGLE draft could fix this by directly aligning target and draft probabilities up to depth L. Second, one could fuse the multi-layer fusion approach in EAGLE-3 with our GBV-aligned training in Appendix N to retrain an EAGLE-3 draft model from scratch.

We also profile GBV-p/q for temperature 1.0 EAGLE-3 runs above, the slowest relative to the EAGLE-3 baseline, using Nsight Systems and Compute on an A100 GPU. The target model dominates end-to-end decoding at 71.8%, followed by state updates and drafting at 16.8%, and GBV verification at 11.4%. The draft bottleneck is reduced compared to previous experiments as the EAGLE-3 draft is small (0.33B) and depth is smaller. At the kernel level, GEMM kernels in the target pass take 63.4% of GPU time at 36.9% SM utilization, while non-GEMM kernels for verification and updates take 36.6% of GPU time at 8.3% SM utilization. The gather and reduce operations for verification are cheap at 0.4% and 0.2% SM utilization, bottlenecked by memory latency rather than throughput. Since verification is only around 10% of walltime, GBV is already highly practical without custom kernels; optimizations yield modest end-to-end speedups, and the dominant cost is the target pass.

We also believe our EAGLE-3 block efficiency results provide strong evidence that GBV can be implemented effectively in inference engines, and even in multi-batch decoding. As EAGLE-3 reports 40% throughput gains with batch size 64 in SGLang, with a much larger tree size (over 40 nodes) than we use for GBV in our experiments above (around 20 nodes), and GBV achieves higher block efficiency than EAGLE-3 across many temperatures, it likely would fare well in such settings.