Grammar-State Aware Beam Search for Enhancing Structural Diversity in LLM Generation

Hantao Hua[1,*], **Jiming Su**[2,*], **Yiping Yao**[2], **Feng Zhu**[2,†]
[1]College of Computer Science and Technology, National University of Defense Technology
[2]College of Systems Engineering, National University of Defense Technology
zhufeng@nudt.edu.cn

Abstract

In many large language model (LLM) applications, generating structurally diverse candidate outputs is crucial for downstream decision-making and reasoning. However, conventional beam search allocates search budget at the token-prefix level, so multiple beam slots may be occupied by lexically different but structurally similar hypotheses. To address these limitations, we propose Grammar-State Aware Beam Search (GSA-Beam). Our method first organizes search at the constraint state level rather than individual prefix level, enabling explicit control over structural diversity. It further incorporates a dynamic state-level beam regulation mechanism that adaptively adjusts beam allocation based on the branching of constraint states, efficiently exploring the feasible structured solution space. Experiments on JSON-schema constrained generation show that GSA-Beam improves explicit state coverage and final structural diversity while preserving schema validity; across several LLM backbones, it reduces reference latency by 22.5% on average compared with standard constrained beam search. Code is available at https://github.com/Paradozile/GSABeam.

1 Introduction

Structured generation under hard validity constraints is a fundamental requirement in many LLM-based applications [Geng *et al.*, 2023]. In applications such as task planning [Ao *et al.*, 2025] and constrained query generation [Luo *et al.*, 2025], the goal is not to identify a single canonical solution, but to explore a space of feasible structures that differ in action composition, execution order, or relational organization. Downstream decision-making [Hong *et al.*, 2025] or evaluation procedures, such as planners, optimizers, or evaluators often operate over this space of alternatives and rely on access to distinct candidates to make informed decisions. In such cases, returning candidates that differ only at the lexical level while sharing the same underlying structure can therefore severely limit the effectiveness of such systems. As a result, the ability to generate structurally diverse yet valid candidates is a critical requirement for LLM-based structured generation.

To generate structurally diverse yet valid candidates, decoding strategies must balance exploration with controllability. Greedy decoding [Gu *et al.*, 2018] commits to a single hypothesis, while stochastic sampling [Kool *et al.*, 2019] introduces randomness that is difficult to regulate in structured settings. Beam search[Steinbiss *et al.*, 1994], by contrast, offers a principled compromise by maintaining multiple high-probability hypotheses in a controlled manner. As a result, beam search has become the standard decoding strategy for multi-candidate generation in practice.

Building on this standard practice, constrained decoding [Ugare *et al.*, 2024] is typically combined with beam search to enforce hard validity requirements while maintaining multiple hypotheses. In this formulation, however, constraints are applied primarily at the token level as feasibility filters, whereas beam expansion and pruning remain governed by prefix-level likelihoods, without explicitly modeling the structure induced by the constraints.

Under grammar- or automaton-based constraints [Dong *et al.*, 2025], this mismatch leads to a many-to-one correspondence between token prefixes and underlying constraint states, so that a large fraction of the beam capacity is devoted to lexically distinct but structurally equivalent partial hypotheses [Chan *et al.*, 2025]. As a consequence, conventional constrained beam search lacks a mechanism for organizing exploration at the structural level, making it difficult to systematically cover the space of feasible structures and to reliably produce structurally diverse candidates required by downstream reasoning and decision-making tasks.

To address these limitations, we propose Grammar-State Aware Beam Search (GSA-Beam), a decoding framework that reorganizes beam search around the underlying grammar state space rather than lexical prefixes. By explicitly organizing beam search around grammar states, GSA-Beam moves beyond using constraints solely for legality checking and instead uses structural state information to guide exploration. This state-centric perspective directly targets the structural redundancy inherent in prefix-level beam search and enables systematic coverage of the feasible structure space, yielding multiple valid candidates that are diverse at the structural level.

Our contributions are summarized as follows:

We introduce a grammar-state–aware beam organization strategy for constrained decoding, which groups partial hypotheses by their corresponding grammar states rather than lexical prefixes, enabling direct control over structural diversity during multi-candidate generation.

We propose a dynamic, state-level beam regulation mechanism that reallocates search budget according to grammar state branching characteristics, reducing redundant exploration while maintaining high decoding efficiency under complex constraints.

We conduct extensive experiments across multiple structured generation tasks and LLM backends, demonstrating that GSA-Beam consistently improves structural diversity while preserving strict validity and achieving lower latency than conventional constrained beam search.

2 Preliminaries

2.1 Constrained Decoding

Constrained decoding enforces a predefined structural specification, such as a context-free grammar, finite-state automaton, or JSON schema, during autoregressive generation [Dong *et al.*, 2025; Chen *et al.*, 2025]. In this paper, we focus on grammar-based hard constraints that can expose a constraint progress state and a set of admissible next tokens. We represent such a constraint mechanism as

$$\mathcal{C} = \langle \mathcal{S}, s_0, \mathcal{V}, \delta, \mathcal{A} \rangle$$

where $\mathcal{S}$ is the set of constraint states, s_0 is the initial state, $\mathcal{V}$ is the token vocabulary, $\delta : \mathcal{S} \times \mathcal{V} \to \mathcal{S} \cup \{\text{error}\}$ is the transition function, and $\mathcal{A}(s) \subseteq \mathcal{V}$ is the set of admissible next tokens at state s.

For a partial hypothesis $y_{1:t}$, the constraint state is updated as $s_t = \delta(s_{t-1}, y_t)$. At decoding step t, invalid tokens are masked before softmax:

$$\tilde{z}_{t,i} = \begin{cases} z_{t,i}, & v_i \in \mathcal{A}(s_{t-1}) \\ -\infty, & \text{otherwise} \end{cases}$$

This logits-level masking is the legality-enforcement mechanism provided by the constrained decoding backend.

2.2 Standard Grammar-Constrained Beam Search

The term constrained beam search has been used in prior work for beam-search variants with explicit constraints, especially lexical or phrasal constraints. In this paper, we use *standard grammar-constrained beam search* to denote a specific operational baseline: standard beam search combined with grammar-based legality masking at each expansion step. At each step, it maintains the top-K partial hypotheses according to accumulated model probability:

$$B_t = \mathrm{TopK}_{y_{1:t}} \, p(y_{1:t} \mid x)$$

Each hypothesis is expanded only with admissible tokens from its current constraint state, but the beam itself is still organized and pruned at the token-prefix level.

This prefix-level organization can be redundant under grammar constraints. Multiple distinct prefixes may expose the same admissible continuation space or correspond to the same constraint progress:

$$s(y_{1:t}^{(i)}) = s(y_{1:t}^{(j)})$$

Equivalently, the beam induces state-specific subsets

$$B_t = \bigcup_{s \in \mathcal{S}} B_t^{(s)}, \qquad B_t^{(s)} = \{ y_{1:t} \in B_t \mid s(y_{1:t}) = s \}$$

Despite this structural equivalence, conventional beam search allocates its fixed beam capacity at the level of individual prefixes. As a consequence, the effective search budget assigned to a constraint state s at step t is given implicitly by the cardinality $|B_t^{(s)}|$, which emerges from prefix-level competition rather than from any explicit state-aware decision. States that admit many high-probability surface realizations may therefore dominate the beam, while other feasible states receive little or no allocation.

This mismatch between structural equivalence under constraints and prefix-level budget allocation can lead to systematic redundancy in the search process. Beam capacity can be consumed by multiple hypotheses that represent identical structural configurations, limiting exploration of alternative constraint states and reducing coverage of the feasible structured solution space.

3 Grammar-State Aware Beam Search

3.1 Overview and Design Rationale

As discussed in Section 2.2, standard grammar-constrained beam search enforces legality through token masking but still organizes and prunes hypotheses at the prefix level. Under hard structural constraints, many lexically different prefixes may correspond to the same constraint progress or expose the same admissible continuation space. This prefix-level competition can allocate multiple beam slots to structurally redundant hypotheses and leave other feasible structural alternatives under-explored.

The core idea behind GSA-Beam is to realign the decoding process with the structure induced by the constraints. Rather than treating constraint states as auxiliary artifacts used solely for legality checking, GSA-Beam elevates grammar states to first-class units in the search process. This shift allows beam organization and budget allocation to account for structural progress, directly addressing the mismatch between prefix-level pruning and state-level constraint enforcement.

Importantly, GSA-Beam does not alter the underlying language model, the constraint specification, the grammar backend, or the token-level scoring mechanism. Partial hypotheses are still generated and scored at the token level. The key difference lies in how these hypotheses are organized and compared during decoding: hypotheses are grouped according to their associated grammar states, and search effort is regulated at the state level rather than being implicitly determined by prefix-level competition.

Based on this design principle, GSA-Beam introduces two components. First, it adopts a state-centric beam organization strategy that partitions active hypotheses by successor grammar-state signatures. Second, it incorporates a dynamic

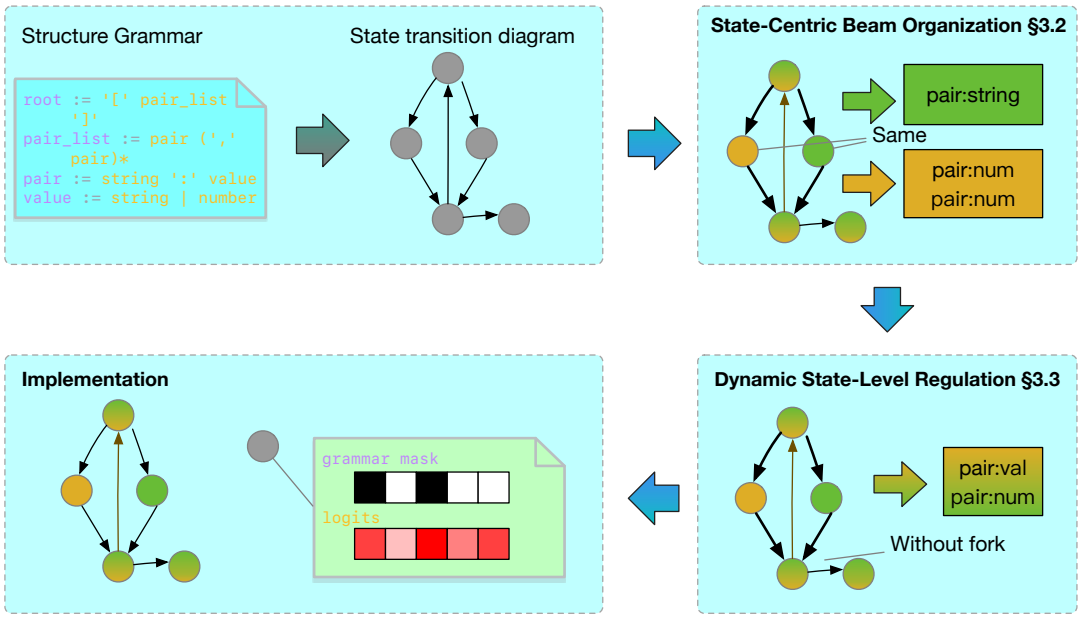

Figure 1: Overview of the Grammar-State Aware Beam Search (GSA-Beam) framework. The algorithm organizes the beam around grammar states rather than token-level prefixes and applies a dynamic state-level regulation strategy. At each decoding step, partial hypotheses are expanded according to valid grammar transitions, grouped by their successor states, and pruned within each state according to the allocated state-level budget.

state-level beam regulation mechanism that adjusts the active beam budget according to model uncertainty and structural branching. Together, these components turn grammar states from passive legality-checking artifacts into active units for organizing exploration.

3.2 State-Centric Beam Organization

GSA-Beam reorganizes the beam around constrain grammar states, rather than lexical prefixes. As discussed in Section 2.2, multiple prefixes may correspond to the same constraint progress or expose the same admissible continuation space. At decoding step t, we partition the active beam into state-specific groups:

$$B_t = \bigcup_{s \in \mathcal{S}_t} B_t^{(s)}, \qquad B_t^{(s)} = \{h \in B_t \mid s(h) = s\} \tag{1}$$

where $\mathcal{S}_t$ denotes the set of active grammar states or state signatures, and $s(h)$ is the state associated with hypothesis h.

Within each group $B_t^{(s)}$, hypotheses are ranked by their accumulated token-level log-probabilities, preserving the original model scoring. Importantly, grouping does not merge hypotheses or discard their prefix histories. Prefixes that reach the same state remain separate candidates and compete for the top-$K_t^{(s)}$ slots within the same state bucket. Thus, state-centric organization changes the unit of beam allocation, not the token-level scoring rule.

Given a total active beam budget K_t, we assign each non-empty state bucket a local quota $K_t^{(s)}$:

$$K_t^{(s)} = \min\left(|B_t^{(s)}|, \left\lfloor \frac{K_t}{|\{s : B_t^{(s)} \neq \emptyset\}|} \right\rfloor\right) \tag{2}$$

Any remaining beam slots caused by rounding are assigned to the highest-scoring retained candidates across active buckets. If the number of active state buckets exceeds the available beam budget, we keep the highest-scoring buckets first. This makes the search budget explicit at the grammar-state level, instead of allowing it to be determined implicitly by prefix-level competition.

In the implementation, the same principle is applied after expansion: successor candidates are grouped by their successor grammar-state signatures, pruned within each state bucket, and then merged to form the next active beam. In the actual pruning step, GSA-Beam applies the same state-centric principle to successor candidates. Let $C_t^{(s)}$ denote the set of candidate hypotheses that reach successor state s after one-step expansion. The retained hypotheses for state s are

$$\widetilde{B}_{t+1}^{(s)} = \mathrm{Top}_{K_t^{(s)}}\left(C_t^{(s)}; \ell\right) \tag{3}$$

where ℓ denotes accumulated log-probability. The next active beam is then formed by merging the retained state-level candidates:

$$B_{t+1} = \mathrm{Top}_{K_t}\left(\bigcup_{s \in \mathcal{S}_{t+1}} \widetilde{B}_{t+1}^{(s)}; \ell\right) \tag{4}$$

Dynamic adjustment of the total active beam budget B is described in Section 3.3.

3.3 Dynamic State-Level Beam Regulation

State-centric beam organization exposes grammar states as explicit units in the search process (Section 3.2). To reduce

redundant expansion in near-deterministic regions while preserving exploration near structural forks, GSA-Beam dynamically adjusts the total beam width K_t at each decoding step based on model uncertainty and structural branching.

At step t, let $\mathcal{S}_t$ denote the set of active grammar states. For each $s \in \mathcal{S}_t$, let $\mathcal{A}(s)$ be its admissible next-token set, and define the union of legal next tokens as

$$V_{\text{legal}} = \bigcup_{s \in \mathcal{S}_t} \mathcal{A}(s)$$

Let $\tilde{p}_i$ denote the next-token probability after masking invalid tokens and renormalizing over V_{legal}. We compute the legal-token entropy as

$$H_t = -\sum_{i \in V_{\text{legal}}} \tilde{p}_i \log \tilde{p}_i \tag{5}$$

We define the dynamic beam width as

$$K_t = \text{clip}\left(\left\lceil n \cdot \frac{H_t + |\mathcal{A}_{\max}|}{H_{\max} + |\mathcal{A}_{\max}|} \right\rceil, K_{\min}, K_{\max} \right) \tag{6}$$

where $|\mathcal{A}_{\max}| = \max_{s \in \mathcal{S}_t} |\mathcal{A}(s)|$, $H_{\max} = \log |V_{\text{legal}}|$, and n is the target number of completed outputs. The clip operation enforces lower and upper budget bounds. Thus, K_t can be smaller, equal to, or larger than n depending on decoding uncertainty and structural branching.

This rule expands the active beam when both legal-token uncertainty and structural branching are high, and contracts the budget in low-entropy or near-deterministic regions. Under loose grammars with many admissible continuations, H_t and $|\mathcal{A}_{\max}|$ tend to be larger, so GSA-Beam preserves more active hypotheses for exploration. Under tight grammars with few legal continuations, the dynamic budget tends to shrink, avoiding excessive allocation to nearly deterministic paths.

Within the dynamically sized beam, successor hypotheses are grouped into state-specific buckets as described in Section 3.2. The resulting state-level budgets are then used to prune candidates within each bucket while preserving the original token-level scoring.

3.4 Algorithm and Implementation Details

Algorithm 1 summarizes GSA-Beam. Each hypothesis stores its token prefix, grammar-state signature, and accumulated log-probability. Completed hypotheses are moved into a separate pool and are no longer expanded; state-level budgets are assigned only to active candidate buckets.

At each decoding step, GSA-Beam expands active hypotheses using admissible tokens from the shared grammar backend. Successor hypotheses are grouped by successor grammar-state signatures, ranked within each bucket by accumulated log-probability, and pruned according to the allocated state budget. This preserves the original token-level scoring while changing the competition unit from individual prefixes to state buckets.

GSA-Beam relies only on admissible-token sets, state signatures, and token-level scores already produced during constrained decoding. It is therefore model-agnostic and requires no modification to the underlying LLM.

Algorithm 1 Grammar-State Aware Beam Search (GSA-Beam)

Input: x, target count n, constraint system $\mathcal{C}$, max length $T_{\max}$
Output: top-n completed outputs

Initialize active beam $B_0 \leftarrow \{\langle \epsilon, s_0, 0 \rangle\}$
Initialize completed pool $C \leftarrow \emptyset$
for $t = 0, 1, \ldots, T_{\max}$ **do**
  Compute active beam width K_t according to Section 3.3
  Initialize empty state-indexed candidate buckets $\{C_t^{(s)}\}$
  for each hypothesis $\langle y_{1:t}, s, \ell \rangle \in B_t$ **do**
    if $\langle y_{1:t}, s, \ell \rangle$ is terminal or accepting **then**
      Add $\langle y_{1:t}, s, \ell \rangle$ to completed pool C
      continue
    end if
    for each admissible token $a \in \mathcal{A}(s)$ **do**
      $y' \leftarrow y_{1:t} \cdot a$
      $s' \leftarrow \delta(s, a)$
      $\ell' \leftarrow \ell + \log p(a \mid x, y_{1:t})$
      Add $\langle y', s', \ell' \rangle$ to bucket $C_t^{(s')}$
    end for
  end for
  Allocate state-level budgets $\{K_t^{(s)}\}$ over non-empty buckets using total budget K_t
  $B_{t+1} \leftarrow \emptyset$
  for each non-empty bucket $C_t^{(s)}$ **do**
    Sort $C_t^{(s)}$ by accumulated log-probability
    Add top-$K_t^{(s)}$ hypotheses from $C_t^{(s)}$ to B_{t+1}
  end for
  if $|C| \geq n$ **then**
    break
  end if
end for
return top-n hypotheses from C by accumulated log-probability

4 Experiments

This section evaluates whether GSA-Beam improves structural coverage under hard grammar constraints while preserving validity and output quality. We report the main results on completed benchmark settings and use partial stress tests only as qualified supporting evidence.

4.1 Experimental Setup

Models and Parameter Scales. To evaluate the robustness of the proposed decoding strategy across practical deployment settings, we conduct experiments using a range of open-source large language models spanning the 7B–32B parameter scale, drawn from the LLaMA [AI@Meta, 2024] [Gerganov, 2023], Qwen [Yang *et al.*, 2025], DeepSeek [DeepSeek-AI, 2025], and Mistral [Jiang *et al.*, 2023]. All models are publicly available off-the-shelf checkpoints and are evaluated in a pure inference setting without any fine-tuning or task-specific adaptation, ensuring that observed dif-

ferences arise solely from the decoding strategy.

Tasks and Constraint Types. We focus on structured generation tasks where outputs must satisfy hard grammatical or schema constraints and where multiple valid solutions naturally exist. We evaluate on JSON-mode-eval and a sampled subset of StructEval:

- **JSON-mode-eval** [Nous Research, 2024], which evaluates whether generated outputs strictly conform to predefined JSON schemas.
- **StructEval-subset** [Cao *et al.*, 2024], for which we sample 250 examples from the original benchmark to test structured-generation behavior under a fixed evaluation budget.

The StructEval result should therefore be interpreted as a subset evaluation rather than a full benchmark result. Together, these settings allow us to analyze structural redundancy, validity, and diversity under hard structured-output constraints.

Decoding Backend and Implementation. All constrained methods use SGLang [Zheng *et al.*, 2024] with XGrammar [Dong *et al.*, 2025] as the shared legality backend. XGrammar supplies JSON-schema legality masks during decoding. For state-aware grouping and explicitUSR evaluation, we compute deterministic grammar-state signatures with an external state tracker, rather than relying on private XGrammar internal matcher states. Across all compared methods, the model, tokenizer, prompt, schema, legality backend, and token-level scoring are fixed; methods differ only in candidate organization, sampling, allocation, and pruning.

Multi-Candidate Generation Settings. For the main JSON-mode-eval and StructEval-subset experiments, we evaluate target candidate counts $n \in \{4, 16, 64\}$. The multi-model benchmark uses $n \in \{16, 64\}$. Unless otherwise stated, constrained decoding uses a maximum generation length of 2048 new tokens and expansion top-$k = 32$ for beam-style methods. Larger-width settings are reserved for search-efficiency analysis and are not treated as a full candidate-count sweep.

4.2 Baselines and Evaluation Metrics

Baselines. We compare beam-level search strategies under the same XGrammar legality backend. The baselines include standard grammar-constrained beam search, a large-beam variant, Stochastic Beam Search (SBS), constrained sampling, and DBS-lite. DBS-lite is a simplified lexical-diversity baseline and should not be interpreted as a full reproduction of Diverse Beam Search. We compare these methods against GSA-Beam as the main variant.

Evaluation Metrics. We report validity, explicitUSR, Struct-Ratio, Self-BLEU, and reference latency. Validity measures whether an output parses as JSON and satisfies the target schema.

ExplicitUSR is the fraction of unique deterministic JSONSchemaStateTracker state-path signatures among the returned candidates; it measures structural-state coverage and does not use XGrammar internal state identifiers. Struct-Ratio is the fraction of unique JSON structural signatures after ignoring concrete surface values. Self-BLEU measures surface-form similarity and is reported separately, since GSA-Beam targets structural rather than purely lexical diversity. Latency is reported as latency-ref for our SGLang refill/prefix-reuse prototype, not as optimized serving throughput.

4.3 Main Results

Table 1 reports the main benchmark results under the shared SGLang/XGrammar legality backend using Qwen3-8B. Across JSON-mode-eval and the StructEval-subset, GSA-beam achieves the highest schema validity and explicit state-signature coverage among the compared beam-level methods. In particular, its eUSR remains high as the target candidate count increases, indicating that state-aware allocation mitigates the structural collapse caused by prefix-level beam competition.

Figure 2 further evaluates whether the trends in Table 1 generalize beyond a single backbone. We compare GSA-beam directly with Standard Beam on four local 7B/8B models under the same SGLang/XGrammar legality backend. Across all backbones, GSA-beam consistently improves eUSR over the baseline, indicating that state-aware allocation reliably increases explicit grammar-state coverage. The Struct-Ratio gains are smaller in absolute magnitude but remain positive across all four models, suggesting that improved state coverage also translates into more diverse final JSON structures. In addition, GSA-beam achieves lower Lat.-ref in all four comparisons, showing that the structural coverage gain does not come at the cost of higher reference decoding latency.

The Struct-Ratio results further show that broader state-signature coverage often translates into more diverse JSON structures. This effect is especially clear on the StructEval-subset, where GSA-beam improves Struct-Ratio across all candidate counts. On JSON-mode-eval, the absolute Struct-Ratio values are lower at larger n, suggesting that some schemas admit many distinct state trajectories but fewer distinct final JSON skeletons.

Sampling obtains the lowest latency-ref because it does not maintain ranked beam candidates or state-specific buckets. However, this speed comes with substantially lower validity and weaker structural coverage, especially on the StructEval-subset. We therefore treat sampling as a fast stochastic reference rather than a replacement for controlled multi-candidate constrained search. Among beam-style methods, GSA-beam provides the strongest trade-off between validity, eUSR, Struct-Ratio, and reference latency.

We also run small 14B/32B models to verify feasibility on larger backbones. These runs are reported in the released artifacts and are not included in the main aggregate claims because they reached similar conclusions.

4.4 Ablation Studies

We conduct ablations on Qwen3-8B to separate the effects of state-aware grouping and dynamic beam regulation. All variants use the same SGLang/XGrammar legality backend, prompts, schemas, and token-level scoring; they differ only in how the active beam budget is adjusted during decoding. The

Dataset	Method	Val. ↑	eUSR ↑			Struct. ↑			Lat.-ref ↓		
		Avg.	$n=4$	$n=16$	$n=64$	$n=4$	$n=16$	$n=64$	$n=4$	$n=16$	$n=64$
JSON-mode	Standard Beam	0.955	0.907	0.701	0.495	0.275	0.088	0.030	3.69	5.17	12.61
	Large Beam	0.978	0.871	0.693	0.515	0.311	0.109	0.031	4.45	9.72	42.51
	SBS	0.957	0.894	0.699	0.515	0.278	0.097	0.030	3.60	5.11	12.51
	Sampling	0.922	0.485	0.183	0.068	0.235	0.061	0.016	1.61	1.89	4.67
	DBS-lite	0.960	0.912	0.729	0.548	0.275	0.093	0.032	3.54	5.22	12.82
	GSA-beam	**0.982**	**1.000**	**1.000**	**0.783**	**0.384**	**0.122**	**0.035**	**2.66**	**3.94**	**12.03**
StructEval-subset	Standard Beam	0.877	0.719	0.700	0.669	0.530	0.304	0.158	3.24	6.53	20.76
	Large Beam	0.918	0.827	0.729	0.664	0.468	0.312	0.178	4.58	10.06	57.28
	SBS	0.861	0.835	0.744	0.662	0.476	0.281	0.142	3.44	5.72	15.46
	Sampling	0.716	0.713	0.465	0.317	0.398	0.168	0.057	1.96	2.89	4.50
	DBS-lite	0.956	0.928	0.804	0.714	0.542	0.334	0.186	3.17	6.24	17.28
	GSA-beam	**0.978**	**1.000**	**0.996**	**0.925**	**0.678**	**0.381**	**0.329**	**2.74**	**4.04**	**14.76**

Table 1: Main results on JSON-mode-eval and the 250-example StructEval-subset. All results in this table are obtained with Qwen3-8B unless otherwise stated. Val. reports average schema validity over $n \in \{4, 16, 64\}$. eUSR is computed from deterministic JSONSchemaStateTracker state-path signatures; Struct. measures JSON structural diversity after removing concrete values. Lat.-ref is prototype reference latency, not optimized serving throughput. Dashes indicate method/dataset combinations without completed clean artifacts.

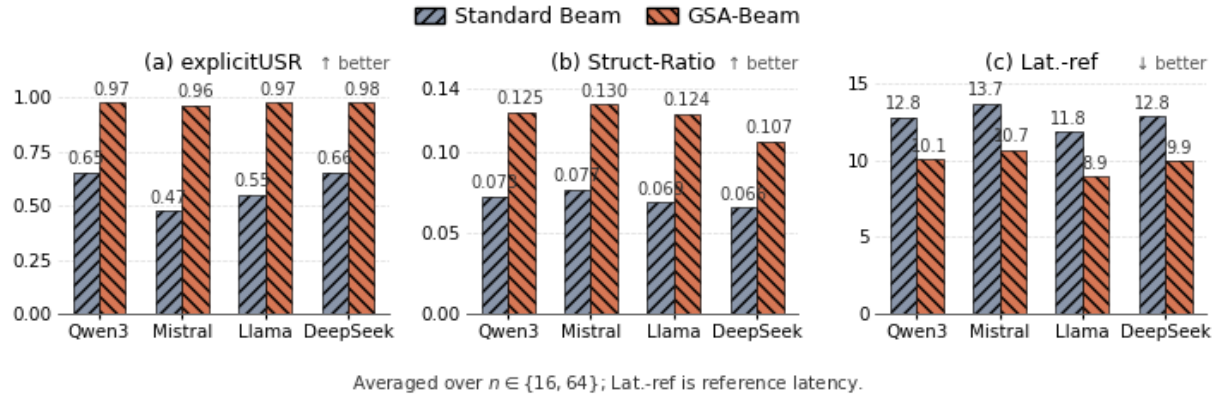

Figure 2: Cross-model comparison between Standard Beam and GSA-beam on four local 7B/8B backbones, averaged over $n \in \{16, 64\}$. eUSR and Struct. are higher-is-better metrics, while Lat.-ref is lower-is-better reference latency.

effect of state-aware grouping itself is reflected by the comparison between Standard Beam and GSA-Beam in Table 1 and Figure 2.

Table 2 shows that the full GSA-Beam variant provides the strongest overall validity–diversity–runtime trade-off. Compared with GSA w/o dynamic, the full method improves validity and eUSR while keeping Struct-Ratio and latency-ref nearly unchanged. This suggests that adaptive regulation stabilizes structural-state coverage rather than merely changing the amount of search.

The two single-signal ablations further clarify the role of the dynamic controller. Removing entropy degrades Struct-Ratio and increases latency-ref, indicating that model uncertainty is useful for deciding when additional exploration is needed. Removing branching leads to the smallest average width and slightly lower latency-ref, but it substantially reduces validity and Struct-Ratio. Thus, lower active width alone is not sufficient: overly aggressive contraction can under-explore structurally meaningful alternatives.

Finally, token-level dynamic control performs competitively but remains slightly worse than the full state-level regulation in Struct-Ratio and latency-ref. This supports the use of grammar-state signals rather than token-level uncertainty alone. Overall, dynamic regulation should be interpreted as an adaptive efficiency mechanism that improves the balance among validity, structural coverage, and reference runtime, while state-aware grouping remains the main source of structural diversity.

4.5 Discussion

The results in Sections 4.3 and 4.4 suggest that structural diversity and decoding cost are not necessarily in direct conflict. Under prefix-level beam competition, many beam slots can be occupied by lexically different but structurally equivalent continuations. GSA-Beam reduces this redundancy by organizing hypotheses around deterministic grammar-state signatures, yielding higher eUSR and Struct-Ratio without simply increasing beam width.

It also shows that diversity metrics capture different behaviors. eUSR measures coverage over grammar-state trajecto-

Variant	Val.↑	eUSR↑	Struct.↑	Lat.↓	Avg. W.
GSA-Beam	**0.948**	0.988	**0.464**	8.766	28.00
w/o dynamic	0.937	0.948	0.463	8.790	26.55
w/o entropy	0.945	0.991	0.440	9.649	19.64
w/o branching	0.917	**0.995**	0.421	**8.739**	13.40
Token dyn.	**0.948**	0.990	0.461	8.983	25.26

Table 2: Ablation study of dynamic beam regulation on Qwen3-8B, averaged over $n \in \{4, 16, 64\}$. All variants use the same SGLang/XGrammar legality backend and token-level scoring. Avg. W. denotes the average active beam-width budget K_t, averaged over decoding steps and clean runs; it is reported as a search-effort indicator rather than an independent quality metric.

ries, whereas Struct-Ratio measures diversity of final JSON structures after removing concrete values. Sampling can be faster because it avoids ranked beam maintenance, but it provides weaker validity and structural coverage in our experiments. Thus, sampling is a useful stochastic reference, but not a replacement for controlled multi-candidate constrained search when many valid and structurally distinct outputs are required.

The ablation study further indicates that state-aware grouping is the main driver of structural-state coverage, while dynamic regulation acts as an adaptive efficiency mechanism. The entropy and branching signals jointly control the active beam-width budget; using only one signal can reduce latency or width, but may also hurt validity or final structural diversity. Therefore, average width should be interpreted as a search-effort indicator rather than an independent quality metric.

Several limitations remain. GSA-Beam requires explicit schemas, grammars, or automata with well-defined state transitions, so its current form is most suitable for hard structured-output constraints. In addition, grammar-state equivalence is structural rather than semantic: prefixes sharing the same state may still differ in downstream meaning or utility. Finally, the benefit of dynamic regulation depends on the grammar regime and may be limited for shallow, highly deterministic, or nearly unconstrained schemas. Extending state-aware allocation to softer constraints and semantic equivalence notions is an important direction for future work.

5 Related Works

5.1 LLM Constrained Generation

Mainstream LLMs predominantly employ decoder-only autoregressive architectures for sequence generation [Vaswani *et al.*, 2017]. To ensure the structural integrity of outputs such as JSON or code, constrained decoding frameworks like Outlines [Willard and Louf, 2023] and XGrammar [Dong *et al.*, 2025] utilize grammar-based schemas to filter the model's vocabulary at each step. This is typically implemented via logit masking, where tokens that violate the formal specification are assigned negative infinity. However, as identified in recent evaluations of constrained decoding frameworks [Chomatek *et al.*, 2025], existing approaches suffer from significant scalability issues in high-throughput environments. Specifically, standard decoders lead to structural redundancy, where multiple lexically distinct prefixes correspond to identical progress in the underlying constraint state. While Syncode [Ugare *et al.*, 2024] and Pre3 [Chen *et al.*, 2025] attempt to optimize mask construction, they still treat each beam candidate independently at the prefix level. This "structural collapsing" causes the search budget to be wasted on minor surface variations of a single structural path, a bottleneck that becomes particularly acute during batch generation.

5.2 Beam Search and Adaptive Regulation

Beam search is a ubiquitous heuristic for balancing generation quality and computational efficiency. While modern inference engines like vLLM [Kwon, 2025] and SGLang [Zheng *et al.*, 2024] optimize this process through PagedAttention [Kwon *et al.*, 2023] and KV cache [Ge *et al.*, 2023] sharing, the selection of beam width remains a static trade-off. Standard beam search is inherently structure-agnostic, allocating its fixed capacity based on cumulative token-level probabilities. Although Diverse Beam Search [Vijayakumar *et al.*, 2016] and Iterative Beam Search [Meister *et al.*, 2021][Wei *et al.*, 2024] introduced penalties to promote lexical variety, they lack the fine-grained awareness of the constraint system's state space required for formal grammars. Furthermore, adaptive strategies such as Adaptive Beam Search often rely on probability thresholds rather than structural branching factors. Consequently, search effort is misallocated: wide beams incur excessive computation in deterministic regions where the grammar allows only a single valid token. Our work addresses this gap by introducing a state-aware regulation mechanism, building on the principles of computation reuse, that dynamically adjusts beam width based on structural forks, thereby facilitating diverse candidate generation without a proportional increase in computational cost.

6 Conclusion and Future Work

We presented Grammar-State Aware Beam Search (GSA-Beam), a decoding framework for structurally diverse multi-candidate generation under hard constraints. By organizing hypotheses around deterministic grammar-state signatures rather than token prefixes, GSA-Beam reduces structural redundancy in constrained beam search while preserving schema validity. Experiments on schema constrained generation show that GSA-Beam improves explicit state coverage and final structural diversity across benchmarks and model backbones, with competitive reference latency.

Our results highlight a broader principle: in hard-constrained generation, the unit of search should align with the unit of constraint enforcement. Prefix-level competition can waste beam capacity on structurally equivalent continuations, whereas state-aware allocation directs search toward distinct feasible trajectories.

For future work, we plan to extend state-aware decoding beyond explicit schemas and grammars, including softer constraints and semantic notions of equivalence. This would allow structural diversification to be applied in settings where constraints are partially specified or not directly represented as automata.

Ethical Statement

There are no ethical issues.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No.62503491.

Contribution Statement

Hantao Hua and Jiming Su contributed equally to this work. Feng Zhu is the corresponding author.

References

[AI@Meta, 2024] AI@Meta. Llama 3 model card. 2024.

[Ao *et al.*, 2025] Jicong Ao, Fan Wu, Yansong Wu, Abdalla Swiki, and Sami Haddadin. Llm-as-bt-planner: Leveraging llms for behavior tree generation in robot task planning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1233–1239. IEEE, 2025.

[Cao *et al.*, 2024] Boxi Cao, Mengjie Ren, Hongyu Lin, Xianpei Han, Feng Zhang, Junfeng Zhan, and Le Sun. StructEval: Deepen and broaden large language model assessment via structured evaluation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5300–5318, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[Chan *et al.*, 2025] Brian J. Chan, Jui-Hung Cheng, Mao Xun Huang, Chao-Ting Chen, and Hen-Hsen Huang. Efficient beam search for large language models using trie-based decoding. *arXiv preprint arXiv:2502.00085*, 2025.

[Chen *et al.*, 2025] Junyi Chen, Shihao Bai, Zaijun Wang, Siyu Wu, Chuheng Du, Hailong Yang, Ruihao Gong, Shengzhong Liu, Fan Wu, and Guihai Chen. Pre3: Enabling deterministic pushdown automata for faster structured llm generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11253–11267, 2025.

[Chomatek *et al.*, 2025] Łukasz Chomatek, Jakub Papuga, Przemyslaw Nowak, and Aneta Poniszewska-Marańda. Decoding ci/cd practices in open-source projects with llm insights. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pages 1638–1644, 2025.

[DeepSeek-AI, 2025] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[Dong *et al.*, 2025] Yixin Dong, Charlie F Ruan, Yaxing Cai, Ziyi Xu, Yilong Zhao, Ruihang Lai, and Tianqi Chen. Xgrammar: Flexible and efficient structured generation engine for large language models. *Proceedings of Machine Learning and Systems*, 7, 2025.

[Ge *et al.*, 2023] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.

[Geng *et al.*, 2023] Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured nlp tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 10932–10952. Association for Computational Linguistics, 2023.

[Gerganov, 2023] Georgi Gerganov. llama.cpp: A portable high-performance inference engine for llama models. GitHub repository, 2023. Open-source project, MIT License.

[Gu *et al.*, 2018] Jiatao Gu, Daniel Jiwoong Im, and Victor OK Li. Neural machine translation with gumbel-greedy decoding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Hong *et al.*, 2025] Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. Next-generation database interfaces: A survey of llm-based text-to-sql. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[Jiang *et al.*, 2023] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[Kool *et al.*, 2019] Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International conference on machine learning*, pages 3499–3508. PMLR, 2019.

[Kwon *et al.*, 2023] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

[Kwon, 2025] Woosuk Kwon. *vLLM: An Efficient Inference Engine for Large Language Models*. PhD thesis, University of California, Berkeley, 2025.

[Luo *et al.*, 2025] Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Yuan-Fang Li, Chen Gong, and Shirui Pan. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 41540–41565. PMLR, 13–19 Jul 2025.

[Meister *et al.*, 2021] Clara Meister, Martina Forster, and Ryan Cotterell. Determinantal beam search. *arXiv preprint arXiv:2106.07400*, 2021.

[Nous Research, 2024] Nous Research. Json-mode-eval. https://huggingface.co/datasets/NousResearch/json-mode-eval, 2024. Hugging Face dataset; accessed 2025-10-07.

[Steinbiss *et al.*, 1994] Volker Steinbiss, Bach-Hiep Tran, and Hermann Ney. Improvements in beam search. In *ICSLP*, volume 94, pages 2143–2146, 1994.

[Ugare *et al.*, 2024] Suhas Ugare, Tarun Suresh, Hanjun Kang, Faisal Ahmed, Yelong Qi, Anusha Kannan, Yuntian Cao, Kathryn Mazaitis, and Graham Neubig. Syncode: Llm generation with grammar augmentation. *Transactions on Machine Learning Research*, 2024. TMLR.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Vijayakumar *et al.*, 2016] Ashwin K. Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. In *Proceedings of the 2016 AAAI Conference on Artificial Intelligence*, 2016.

[Wei *et al.*, 2024] Chengwei Wei, Kee Kiat Koo, Amir Tavanaei, and Karim Bouyarmane. Confidence-aware substructure beam search (cabs): Mitigating hallucination in structured data generation with large language models. *arXiv preprint arXiv:2406.00069*, 2024.

[Willard and Louf, 2023] Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models, 2023.

[Yang *et al.*, 2025] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.

[Zheng *et al.*, 2024] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 62557–62583. Curran Associates, Inc., 2024.