

Controllable Text-Attributed Graph Generation via LLM-Based Substructure Modeling

Ruijun Mao^{1,2}, Jiaxuan Chen^{1,2}, Boyan Li^{1,2},
Zixing Zhang^{1,2}, Yuhang Zhang^{1,2}, Fangxin Wang^{1,2,3*}

¹School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen

²Shenzhen Future Network of Intelligence Institute

³The Guangdong Provincial Key Laboratory of Future Networks of Intelligence

Abstract

Text-attributed graphs (TAGs) underpin many NLP applications, but real-world TAGs are often limited in scale, class balance, and diversity, motivating realistic and controllable TAG generation. Existing generators either replace node text with precomputed embeddings, reducing the task to vector-attributed graph generation, or rely on LLM prompting and agent simulation; neither learns a joint distribution over raw node text, edges, and labels. We propose a controllable substructure generator built from a lightweight causal LM. The model generates connected TAG substructures rather than whole graphs: each training substructure is serialized into lossless token sequences from multiple start nodes. Averaging next-token loss over these traversals treats traversal order as a latent variable and optimizes a variational lower bound on the permutation-invariant substructure marginal likelihood. Training combines continual pretraining for the unconditional distribution with control-prefix fine-tuning that conditions generation on target structural and label descriptors. At inference, a single autoregressive pass per prefix emits a parseable substructure, and the same generator supports distribution-aligned synthesis and boundary-aware augmentation through target-side adaptation alone. Across five real-world TAGs, it produces losslessly parseable graphs, matches or exceeds structure-only generators in structural fidelity, and improves cross-graph GNN generalization.¹

1 Introduction

Text-attributed graphs (TAGs), where each node carries natural-language content linked by typed relations, are a fundamental data form in modern NLP: citation networks underpin scholarly retrieval and question answering, product networks

support e-commerce search and recommendation, and knowledge graphs feed retrieval-augmented generation and structured reasoning (He et al., 2024; Liu et al., 2024; Zhao et al., 2023). Yet real-world TAGs are often constrained in size, coverage, and statistical diversity — some are restricted by license or privacy, some are severely imbalanced, and most are static snapshots that cannot be adjusted to test specific hypotheses (Zhang et al., 2025). The ability to *generate* realistic TAGs — plausible new graphs sharing the joint statistical properties of an observed graph — would unblock a range of downstream needs that static datasets cannot meet.

This need spans several directions in graph data mining. GNN augmentation methods improve generalization with synthetic minority-class or structurally diverse samples (Zhou and Gong, 2023; Yu et al., 2025; Zhao et al., 2021; Rong et al., 2020; Verma et al., 2021), yet existing techniques are restricted to perturbing the original adjacency or mixing existing node features, leaving the synthesis of genuinely new text-and-structure samples beyond their reach. Empirical analyses of GNNs increasingly rely on graphs with prescribed structural properties to disentangle model behavior under controlled distribution shifts (Palowitch et al., 2022; Langendonck et al., 2026), yet fixed real-world datasets cannot be adjusted to test hypotheses about homophily (Zhu et al., 2020), density, or label balance. Beyond these, controllable TAG generation could support emerging needs such as counterfactual analysis on graphs (Lucic et al., 2022). What unifies these directions is a common requirement: a generator producing TAGs that are not only realistic but also *controllable* — whose textual content, label distribution, and structural statistics can be specified at generation time.

Despite mounting interest in graph generation, no existing approach meets these dual demands of joint text-structure synthesis and intent-

*Corresponding author: Fangxin Wang,
wangfangxin@cuhk.edu.cn

¹Code is available on GitHub:
github.com/Ridge-yz/taggen

conditioned local control. Prior methods fall into three categories. **Structure-only generators** (You et al., 2018; Liao et al., 2019; Chen et al., 2023; Bergmeister et al., 2024) model the adjacency distribution $p(A)$ and ignore node content entirely, so they cannot serve pipelines where downstream models depend on textual semantics. **Attributed-graph generators** extend to $p(A, X)$ but operate on pre-extracted categorical or numerical node features (Vignac et al., 2023; Jo et al., 2022; Li et al., 2024; Yoon et al., 2023), not raw natural-language text; their outputs are incompatible with text-based pipelines such as BERT-encoded GNNs or LLM agents. **LLM-related approaches** introduce language models but each faces a distinct limitation: LGGM (Wang et al., 2025) uses text only as a coarse domain-level prompt rather than generating per-node text; GAG (Ji et al., 2025) simulates agent interactions without learning an explicit joint distribution; LLM4NG (Yu et al., 2025) generates isolated nodes whose connections are inferred post hoc. None learn the joint distribution $p(A, X_{\text{text}}, Y)$ where X_{text} is unbounded natural-language text correlated with both structure and labels, and their controllability is absent or limited to coarse signals such as class label or domain prompt. To our knowledge, no prior work generates TAGs by *training* a language model to model this joint distribution, as opposed to simulating it with agents or prompting an external model; ours is the first to do so.

The shortcomings of prior LLM-related approaches do not stem from limitations of language models themselves. Pretrained LLMs (OpenAI et al., 2024; Grattafiori et al., 2024) produce coherent long-form text under flexible prompting, and instruction tuning yields strong instruction-following generalization (Wei et al., 2022). Amid growing interest in graph foundation models (Mao et al., 2024), GraphGPT (Zhao et al., 2025) made a key advance: by serializing a graph along an Eulerian path into a lossless token sequence, it showed that a causal LLM can be brought to bear on structured graph data at all, offering a first general bridge across the mismatch between sequential causal attention and combinatorial graph structure—validated, however, only on *discriminative* tasks such as node and graph classification. The barrier to *generation*, then, is not capability but interface, and it takes two forms. **Context length:** serializing a graph of ten thousand nodes with rich textual attributes easily yields sequences

of millions of tokens, exceeding the context window of frontier LLMs; we address this practical constraint by modeling substructures rather than whole graphs. **Permutation invariance:** a graph is invariant under node permutation (Niu et al., 2020), while an LLM generates tokens under a fixed left-to-right order, so any serialization must commit to one of $n!$ orderings, and a model trained naively on one ordering captures only that one—a mismatch with no direct analogue in standard text generation.

We address both by shifting the modeling target from full graphs to *substructures*: each is serialized as an edge-traversal sequence that fits within context, and exposing the model to multiple traversals from different starting nodes optimizes a variational lower bound on $\log p(G_{\text{sub}})$. Continual pretraining on these traversals followed by fine-tuning with control prefixes yields **single-pass controllable substructure generation**: an intent expressed through trained descriptors produces a parseable substructure—text, labels, and local structure—in a single pass. We instantiate this capability in two applications from the *same* trained model: *distribution-aligned generation*, which assembles substructures into a macro-graph matching a reference’s statistics, and *augmentation-oriented generation* (TAG-AUG), which conditions on decision-boundary-aware profiles to improve GNN generalization under disjoint splits. As a first attempt in this regime, our goal is to establish feasibility and competitive performance for the proposed modeling paradigm.

Our contributions are:

- We present, to our knowledge, the first approach to generate text-attributed graphs by *training* a language model, formalizing TAG generation as joint modeling of raw text, structure, and labels.
- We propose a controllable substructure generation framework that responds to generation intents by producing corresponding text-attributed graph substructures. The framework sidesteps the context-length barrier via substructure decomposition and addresses permutation invariance through multi-traversal training with a variational lower bound.
- We instantiate two downstream applications—distribution-aligned generation and boundary-aware augmentation—from one trained model, evidencing that the same intent-responsive sub-

structure generator can be used across heterogeneous objectives.

- We conduct a comprehensive empirical study on five real-world TAGs across citation, web, and e-commerce domains, evaluating structural fidelity and GNN generalization under a disjoint train/test regime.

2 Related Work

Graph generation. Graph generation has progressed from autoregressive models (You et al., 2018; Liao et al., 2019) to adjacency-denoising diffusion. EDGE (Chen et al., 2023) models structure alone. A recent line generates *attributed* graphs: DiGress and GDSS (Vignac et al., 2023; Jo et al., 2022) jointly model node attributes and structure, while GraphMaker (Li et al., 2024) diffuses attributes and edges jointly. Their attributes are categorical or numerical feature *vectors*, whereas our setting includes raw natural-language text. GraphMaker studies whole-graph synthesis under data-sharing or benchmarking constraints (Li et al., 2024); our framework uses the generated TAGs for distribution-aligned synthesis and targeted augmentation.

LLMs on graphs. LLMs have been widely adapted to graph *discriminative* tasks—text-feature enhancement (He et al., 2024; Zhao et al., 2023; Chen et al., 2024b), unified task formulation (Liu et al., 2024), structure-aware sequence encoding (Chen et al., 2024a), Eulerian-path serialization for classification (Zhao et al., 2025), and graph instruction tuning (Tang et al., 2024)—but structured graph *generation* with LLMs remains largely unaddressed, and is mostly explored via prompting rather than training (Yao et al., 2024): GAG (Ji et al., 2025) simulates social-graph growth via multi-agent interaction without learning an explicit joint distribution, and LGGM (Wang et al., 2025) treats text only as a coarse prompt. We borrow from GraphGPT a single ingredient—that an Eulerian traversal losslessly serializes a graph—but otherwise differ entirely: where they perform discriminative inference, we perform generative modeling of the joint text–structure distribution, via a multi-start variational objective, control-prefix conditioning, and two-stage training. To our knowledge this is the first framework for controllable text-attributed graph *generation* with LLMs.

Graph data augmentation. Augmentation methods improve GNN generalization (Ding et al., 2022; Zhao et al., 2022) by editing the adjacency (Rong et al., 2020; Zhao et al., 2021), interpolating node features (Verma et al., 2021), or oversampling rare classes (Li et al., 2023; Zhou and Gong, 2023); LLM4NG (Yu et al., 2025) adds isolated nodes with edges predicted post hoc. All perturb or interpolate existing content, or add nodes without coherent local structure. In contrast, our augmentation injects *newly synthesized* substructures carrying coherent local topology and jointly generated text, conditioned on boundary-aware targets and anchored to the real graph—supplying new structural signal rather than reweighting the existing one.

3 Method

We aim to learn a generator for text-attributed graphs (TAGs) conditioned at inference time on user-specified targets. The core contribution is a *controllable substructure generator* (CSG)—a language model trained to produce local subgraphs (text, labels, and edges jointly) that respond to structural and label intent expressed through trained descriptors in a single autoregressive pass. We formalize the problem and serialization (§3.1–§3.2), show that autoregressive training over Eulerian serializations is variational learning of the substructure distribution (§3.3), describe the two-stage training (§3.4) and resulting single-pass controllability (§3.5), and derive two application instantiations (§3.6).

3.1 Problem Formulation

Definition 1 (Text-Attributed Graph). A text-attributed graph is a tuple $G = (V, E, X_t, Y)$ with node set V , edge set $E \subseteq V \times V$, textual attributes $X_t = \{x_v\}_{v \in V}$ where each $x_v \in \mathcal{V}^*$ is a token sequence over a vocabulary $\mathcal{V}$, and labels $Y = \{y_v\}_{v \in V}$ with $y_v \in \mathcal{Y}$.

Given a reference TAG G and a conditioning signal c encoding user-specified targets, we learn a generator $p_\theta(A, X_t, Y \mid c)$ that jointly synthesizes adjacency, text, and labels consistent with c . The central difficulty is structural: an autoregressive model factorizes a token sequence $W = (w_1, \dots, w_T)$ as $p_\theta(W) = \prod_t p_\theta(w_t \mid w_{<t})$ over a *total order*, whereas a graph is invariant to node permutation. Reconciling this mismatch is the focus of the method.

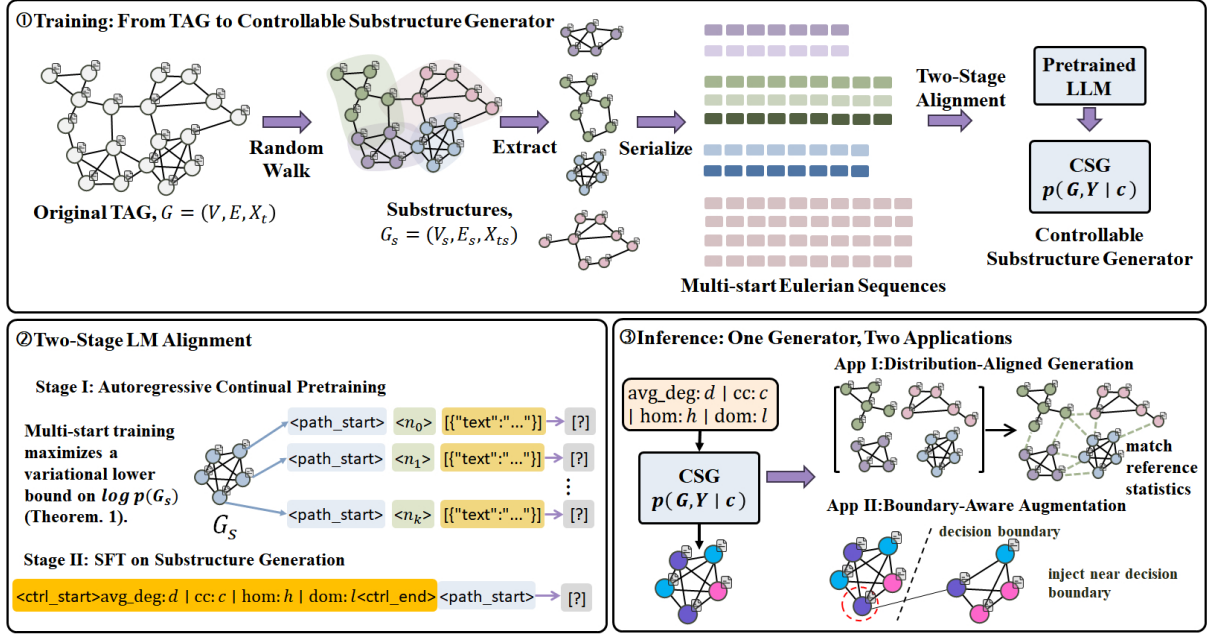


Figure 1: Overview of the method. (①) A lightweight LLM is continually pretrained and supervised-fine-tuned on multi-start Eulerian serializations of sampled substructures, yielding a controllable substructure generator (CSG). (②) Training proceeds in two stages: autoregressive continual pretraining over the N start-node serializations of each substructure, which maximizes a variational lower bound on $\log p(G_s)$ (Theorem 1), followed by control-prefix supervised fine-tuning. (③) At inference, the same generator instantiates two applications by varying only the conditioning targets — distribution-aligned generation and boundary-aware augmentation — with no retraining.

3.2 Substructure-Level Modeling

From full graphs to substructures. Even a moderately sized TAG serializes into a sequence far exceeding a practical context window, so we reduce the modeling target to *substructures*: connected subgraphs $G_s \subset G$ sampled from the reference graph. This scale also makes the variational treatment in §3.3 tractable, permitting a complete enumeration of per-node serializations. We study this design with a *lightweight* LLM, using compact graph statistical descriptors as the conditioning signal and evaluating both descriptor response and strict syntactic validity.

Definition 2 (Lossless Serialization). A serialization scheme is a pair of maps $f : (G_s, \pi) \rightarrow W$ and $f^{-1} : W \rightarrow G_s$ indexed by a serialization choice π , such that $f^{-1}(f(G_s, \pi)) = G_s$ for every admissible π . The scheme is *lossless* when f is a bijection between admissible choices and token sequences, written $G_s \rightleftharpoons W$.

Property 1 (Eulerian Circuit). A connected graph admits an *Eulerian circuit* — a closed walk traversing every edge exactly once — iff every node has even degree. For a graph violating this condition, *Eulerization* adds a minimal multiset of duplicate edges to yield G_s^E with all even degrees, guaranteeing Eulerian circuits exist.

We serialize a substructure by traversing an Eulerian circuit of G_s^E : a choice π fixes a traversal, read off as a token sequence $W = f(G_s, \pi)$. As detailed in §3.4, the duplicate edges added by Eulerization are tagged for removal at decoding, keeping f a lossless bijection $G_s \rightleftharpoons W$. This abstract serialization — one circuit per sequence — is all the following analysis requires. Figure 2 illustrates the interface on a small substructure.

3.3 Autoregressive Training as Variational Learning

A substructure G_s on N nodes is permutation-invariant, so its likelihood marginalizes over the permutation set $\mathcal{A}$, $p(G_s) = \sum_{\pi \in \mathcal{A}} p(G_s, \pi)$ with $|\mathcal{A}| = N!$. This sum is intractable, and a model trained on a single ordering captures only one of its $N!$ terms. Following the ordered-set variational principle of Liao et al. (2019) and related treatments of node ordering as a latent variable (Chen et al., 2021), we optimize a lower bound over a tractable subset of serializations. The enabler is that the Eulerian serialization f of §3.2 is a lossless bijection $G_s \rightleftharpoons W$: each choice π maps to a sequence whose model likelihood $p_\theta(W_\pi)$ equals the joint $p(G_s, \pi)$, so a sum over serializations is a sum of autoregressive likelihoods.

Theorem 1 (Multi-start training is variational

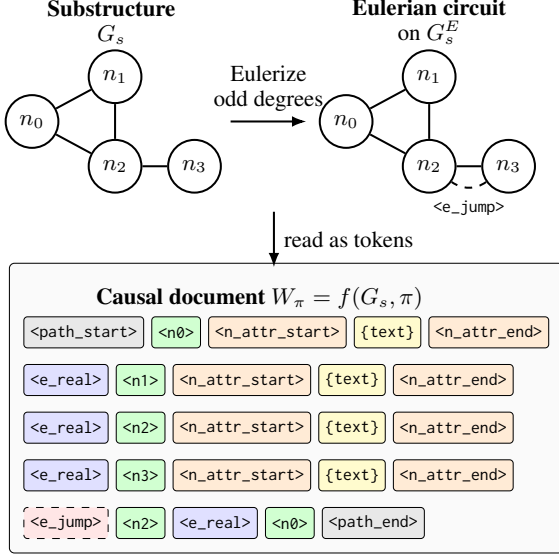


Figure 2: Lossless Eulerian serialization used by the LLM. A connected substructure is Eulerized only to obtain a circuit; duplicate traversal steps are marked as `<e_jump>` and removed at decoding, while real transitions and first-occurrence attributes recover the original text-attributed subgraph.

learning). Let $\tau = \{\pi_1, \dots, \pi_N\}$ be the N Eulerian circuits of G_s , one per start node, with the model trained under the uniform weighting $q_{\pi_i} = 1/N$. Then

$$\log p(G_s) \geq \frac{1}{N} \sum_{i=1}^N \log p_{\theta}(W_{\pi_i}) + \log N. \quad (1)$$

Since $\log N$ is independent of θ , minimizing the next-token cross-entropy averaged over the N Eulerian serializations of G_s exactly maximizes a variational lower bound on $\log p(G_s)$.

The bound follows by introducing a variational distribution q over τ and applying Jensen’s inequality to $\log \sum_{\pi} p(G_s, \pi)$; the optimal $q^* \propto p(G_s, \pi)$ recovers the log-partition bound $\log \sum_{\pi \in \tau} p(G_s, \pi)$, and the uniform q used in training yields Equation (1). The intractable $O(N!)$ marginalization is thus replaced by an $O(N)$ variational set—affordable because we model substructures. In practice, CPT uses uniform q over these circuits; the monotonicity result for adding circuits concerns the tightest attainable bound obtained by optimizing q on each serialization set. Full derivation and the monotonicity result are in Appendix A.

Remark 1. Each start-node circuit is one “view” of the same substructure; training on a single view rewards an arbitrary traversal order, whereas averaging over the N views rewards order-agnostic

structure. The bound thus formalizes a simple intuition—more views give the model a less order-biased target—and turns the choice of “how many traversals to enumerate” into a principled knob rather than a heuristic. We next turn this recipe into a concrete two-stage training procedure.

3.4 Two-Stage LLM Training

Theorem 1 gives a direct recipe: feed the model the N Eulerian serializations of each substructure under uniform weighting. We realize this in two stages—continual pretraining (CPT) for the unconditional distribution and supervised fine-tuning (SFT) for the conditional one—sharing one serialization backbone. The document templates are:

Training document templates.
CPT `<path_start><n0><n_attr_start>Academic paper with title and abstract: Coevolving High Level Representations <n_attr_end><e_real><n1> ... <e_jump><n0><path_end>`
SFT `<ctrl_start>avg_deg:2.38|cc:0.17|homophily:0.12|dom:Rule Learning:0.38<ctrl_end><path_start><n15><n_attr_start>{"label": "Probabilistic Methods", "text": "Academic paper ... revise probabilistic rule bases"}<n_attr_end><e_real><n3> ... <path_end>`

Continual pretraining. CPT targets $p(G_s)$, the quantity lower-bounded in Theorem 1. We sample substructures via random walks, Eulerize each, and emit all N start-node circuits, realizing the uniform- q regime with the full set τ . We enumerate every start node precisely because the bound tightens as τ grows, so next-token training on this corpus directly maximizes the corresponding uniform- q variational lower bound. Each CPT document uses three token roles: node tokens identify distinct nodes, first-occurrence attribute blocks write each node’s text once, and edge tokens mark real versus Eulerization edges. Decoding drops `<e_jump>` transitions and reads content from the first attribute block for each node, recovering G_s exactly (Definition 2); no labels are used at this stage.

Supervised fine-tuning. SFT reuses the same backbone and makes two minimal edits to move from $p(G_s)$ to the conditional $p(G_s, Y | c)$. As shown in the template, each attribute block is upgraded from text to `{label, text}`, and each sequence receives a masked control prefix containing average degree, clustering coefficient, label homophily, and dominant-label proportion. The model thus first learns the unconditional graph grammar, then learns to steer it from a compact prefix, with no change to the traversal format. Two adjustments keep the corpus balanced and bounded: stratified start-node sampling with bin-

ning over (degree, homophily, dominant label) and rare-label targeted walks; and a bounded number of circuits per substructure rather than all N starts. Full corpus statistics are in Appendix F.

3.5 Single-Pass Controllable Substructure Generation

After CPT and SFT, the LLM generates a complete substructure—nodes, text, labels, internal edges—from an intent-conditioned control prefix within the trained descriptor space in a single pass. We call this **single-pass controllable substructure generation**: *single-pass* because the model realizes a control target in one autoregressive pass with no per-target optimization, retraining, or iterative refinement, and *controllable* because generated statistics respond to the requested descriptors (§4). The prefix may moreover specify descriptor values not seen verbatim in training, so a trained generator can respond to new intents within the trained descriptor space without further adaptation. During long-sequence decoding the model may violate the first-occurrence invariant by re-emitting an attribute block for a declared node; we enforce the invariant with a logits processor that masks `<n_attr_start>` when the node is already defined and forces it when new, needing no retraining (Willard and Louf, 2023).

3.6 Application Instantiations

The CSG operates locally, generating one substructure at a time in response to an expressed intent. To examine its value in downstream tasks, we instantiate two applications that derive task-specific conditioning targets and use the generated substructures in task-specific ways. Each dataset has its own CPT+SFT generator, and both applications reuse the corresponding generator $\mathcal{M}$ while differing in target derivation and integration. Full algorithms and hyperparameters are in Appendices H and I.

Distribution-aligned generation. To match a reference G^* , we measure its average degree, clustering coefficient, homophily, and label distribution, and query the generator until a node budget is reached. As each substructure is generated independently, two mechanisms enforce global alignment: a *proportional-integral controller* adjusts each prefix’s targets from the running discrepancy so aggregate degree, clustering, and homophily converge to G^* , and a *deficit-based la-*

bel quota picks the most under-supplied class as each dominant label. Since the independent substructures share no edges, a VGAE (Kipf and Welling, 2016) pretrained on G^* predicts the missing inter-substructure edges from text-conditioned latent embeddings, up to the target degree budget. Thus, the language model supplies packet content and internal topology, while the assembly component supplies global inter-packet connectivity.

Augmentation-oriented generation (TAG-AUG). To augment a training graph G_{tr} , we replace aggregate targets with *boundary-aware* per-class profiles: an oracle GCN (Kipf and Welling, 2017) identifies the correctly-classified, smallest-margin nodes per class as boundary anchors, and their one-hop ego statistics form a target profile, with inverse-frequency quotas favoring minority classes. Querying $\mathcal{M}$ yields candidate packets, of which we keep those that parse losslessly and pass basic validity checks. Each accepted packet is attached by linking its dominant target-label node to the semantically nearest real boundary anchors under text-embedding similarity, injecting new structure near the decision boundary.

Both instantiations consume the same $\mathcal{M}$ and require only target-side adaptation, isolating substructure-level controllability as a reusable capability: a new application is added by specifying target derivation and task-specific integration, with no retraining.

4 Experiments

We evaluate the framework along its intended use: first the substructure generator’s controllability on held-out targets (§4.2), then distribution-aligned generation by structural fidelity (§4.3) and augmentation-oriented generation by GNN generalization (§4.4), and finally the two-stage training ablation (§4.5). We ask whether one controllable text-structure generator can remain competitive with methods specialized to a single task.

4.1 Experimental Setup

Datasets and split. We use five TAGLAS (Feng et al., 2024) graphs spanning citation, web, and e-commerce domains: Cora, CiteSeer, PubMed, WikiCS, and a curated ogbn-Products subset (Hu et al., 2020) (Appendix F). Since a TAG is one large graph and both tasks need train/test graphs, we use a community-aware stratified split:

Dataset	Split	Nodes	Edges	Classes
Cora	Train	1,766	2,258	7
	Test	942	596	7
CiteSeer	Train	2,096	1,793	6
	Test	1,090	503	6
PubMed	Train	12,999	18,641	3
	Test	6,718	5,431	3
Products	Train	35,459	30,853	24
	Test	18,357	8,496	24
WikiCS	Train	7,698	94,180	10
	Test	4,003	24,743	10

Table 1: Per-split statistics under the community-aware stratified split; train and test graphs are node- and edge-disjoint.

METIS-style communities (Karypis and Kumar, 1998) are split within each (label, community) block 0.66/0.34, yielding node- and edge-disjoint train/test graphs that preserve major communities and label proportions. This is harder than transductive evaluation — generation cannot rely on memorized training edges, and augmentation must transfer to a shifted test graph — but is not proposed as a benchmark. Per-split statistics are in Table 1; details are in Appendix C.

Protocol and baselines. All generators produce original-size graphs below 4,000 nodes and are capped at 4,000 otherwise. Structural fidelity uses Gaussian-EMD MMD over degree, clustering, 4-orbit, and spectral statistics of sampled 200-node subgraphs, following You et al. (2018) (Appendix C). We compare against autoregressive GraphRNN (You et al., 2018) and GRAN (Liao et al., 2019), diffusion DiGress (Vignac et al., 2023) and GraphMaker (Li et al., 2024), and the LLM-based GAG (Ji et al., 2025) (comparable only on the datasets in its paper); for augmentation, against DropEdge (Rong et al., 2020), GAUG (Zhao et al., 2021), GraphSHA (Li et al., 2023), and LLM4NG (Yu et al., 2025). For each dataset, we train a separate generator; in the distribution-aligned application, VGAE supplies inter-substructure connectivity after the LM generates the packets. Backbone: Qwen2.5-3B (Qwen et al., 2025); details are in Appendix G. TensorBoard reports approximately 2.2–4.3 h for CPT and 1.2–2.4 h for SFT per dataset on two RTX A6000 GPUs.

4.2 Controllability on Held-Out Targets

We first test the generator directly: prompted with a control prefix it never saw verbatim, does it emit a valid, parseable TAG that responds to the requested targets? For each dataset we sweep

the three structural descriptors over the 20/50/80 percentiles of the SFT control distribution, decoding the raw CPT+SFT model with no downstream correction (Appendix B). Across the raw generations, 83.3% parse under strict rules and 95.3% of those round-trip losslessly, so once a graph parses, its structure and attributes are almost always recovered — evidence that the paradigm aligns causal generation with graph structure at low hallucination. Figure 3 summarizes raw validity and grid-level control response; full Pearson/Spearman/MAE values are in Appendix B, since each dataset uses a separately tuned generator with incomparable descriptor ranges.

Analysis. The generator provides *coarse* local control over trained descriptors. Clustering is the most consistently steerable descriptor (Pearson 0.98 on Products, 0.84 on WikiCS); homophily is strong on WikiCS (0.89) but unstable on several citation and product graphs; average degree shows the right direction but misses extreme regimes — on WikiCS, requested degrees of 16–21 yield substructures of degree 4–6, inflating MAE. The downstream pipeline consequently uses feedback control and VGAE assembly (§3.6) for graph-level targets.

4.3 Structural Fidelity

We assemble controlled substructures into a macro-graph matching a reference and measure MMD against both train and test graphs (Appendix C, Table 5). Despite jointly generating node text, our generator is consistently strong on *global* structure, best on degree MMD on most splits and on spectral MMD on several. On *local* motif statistics the structure-only baselines are sometimes ahead — GRAN on the sparse citation graphs, density-matched diffusion on the tree-like Products subgraph — as expected, since they optimize structure alone while ours also spends capacity on text. Among LLM-based methods, GAG (comparable only on Cora and CiteSeer) is markedly weaker throughout. The dense WikiCS shows the resulting flexibility: ours handles it comfortably, best on three of four metrics per split, where diffusion degrades sharply. Finally, since train and test are disjoint, the right-hand block confirms the generator stays best or near-best on degree and spectral MMD on the *held-out* test graph for most datasets, capturing transferable regularities across the two splits.

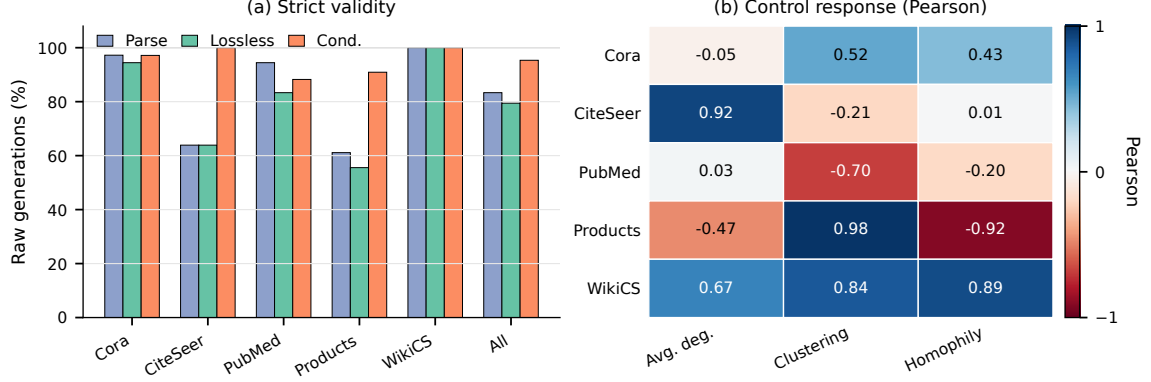


Figure 3: Controllability on held-out targets of the raw CPT+SFT generator. **Left:** raw-generation validity: strict parse rate (Parse), lossless round-trip rate (Lossless), and conditional lossless rate among parseable samples (Cond.). **Right:** Pearson correlation between requested and realized descriptor values over the three percentile grid levels, computed on parseable samples. Strict validity criteria and complete Pearson/Spearman/MAE values are reported in Appendix B.

Packet and assembly comparison. To separate local packet modeling from the subsequent assembly stage, we also compare size-matched BFS-induced samples from packets alone and from the assembled graphs against the same real samples. Here, a *packet* denotes the local substructure emitted by one LM inference. The five-dataset results are reported in Appendix C.4, Table 7. Packet-only samples already provide comparable local statistics, while assembly improves 33 of 40 reported cells (20/20 on train and 13/20 on test). These results indicate that the LM-generated packets capture local structural statistics, while assembly generally improves agreement with the reference at the assembled-graph scale.

4.4 GNN Generalization

Switching the conditioning targets to boundary-aware profiles turns the *same* generator into a graph augmenter. We train a two-layer GCN (Kipf and Welling, 2017) on an augmented training graph that contains the real training nodes together with generated substructure nodes and their edges. Model selection uses the real validation split, and the selected model is evaluated on the node- and edge-disjoint real test graph. We repeat the comparison with five random seeds and report changes in accuracy and macro-F1 relative to the same-seed un-augmented graph (Appendix D, Table 8). Figure 4 shows the macro-F1 means. The strict node- and edge-disjoint protocol makes transfer difficult, and the resulting gains are modest and dataset-dependent: our method improves mean Macro-F1 on four of five datasets, with gains of +2.93, +0.57, +4.14, and +1.34

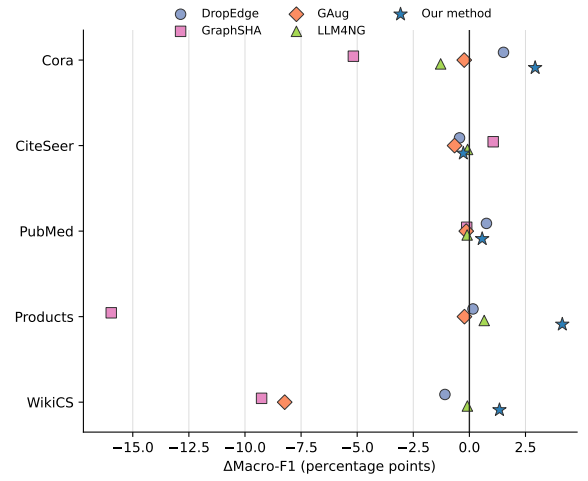


Figure 4: Mean Macro-F1 change over five seeds under the train-augmented / test-on-held-out protocol. Values are percentage-point changes relative to training on the un-augmented graph; the vertical line marks no change. Full accuracy and Macro-F1 mean $\pm$ standard deviation results are in Appendix D.

percentage points on Cora, PubMed, Products, and WikiCS, respectively; CiteSeer shows a slight decrease (-0.27 points).

4.5 Ablation: The Role of Continual Pretraining

Finally we ask whether the two-stage design is necessary, comparing the full model against a *w/o CPT* variant that fine-tunes the base LLM directly on the SFT corpus (Appendix E). SFT alone is already capable, competitive on several MMD columns and occasionally winning, especially on some motif columns. What CPT adds is *stability and generalization*: across most degree and clustering columns, and many 4-orbit columns, the full

Dataset	w/o CPT	Single	Multi
Cora	.0584	.0608	.0390
CiteSeer	.0339	.0360	.0251
PubMed	.2570	.3876	.1629
Products	.0624	.0693	.0519
WikiCS	.1763	.1730	.1390

Table 2: Final SFT held-out next-token loss for the CPT comparison. Lower is better; the best value per row is in **bold**.

Target	CPT source	SFT-only	Cross	Native	SFT-only	Cross
Cora	CiteSeer	.0584	.0556	.0390	72.2/69.4	88.9/88.9
CiteSeer	Cora	.0339	.0308	.0251	80.6/75.0	91.7/75.0

Table 3: Bidirectional cross-dataset CPT transfer. Loss is held-out SFT next-token loss; the last two columns report parse/lossless rates (%).

model is better; on dense WikiCS it improves degree and spectral MMD while the ablation mainly matches local motifs. CPT therefore supports stability of the learned structural distribution across statistics and harder graphs.

Multi-start analysis. To isolate traversal diversity from the general effect of CPT, we match the multi-start corpus size with a single repeated serialization per substructure. All three conditions then use the same target SFT corpus and held-out split. Multi-start gives the lowest final SFT held-out loss on all five datasets (Table 2), directly supporting the benefit of exposing the model to multiple autoregressive views of each substructure and the corresponding variational interpretation. The corresponding aggregate parse/lossless rates over 180 held-out attempts are 83.3/79.4% for multi-start and 80.6/74.4% for single-start.

Cross-dataset CPT transfer. We further test whether CPT learned on one citation TAG transfers to another. The bidirectional Cora–CiteSeer transfer results are shown in Table 3; cross-dataset CPT improves the held-out SFT loss over SFT-only in both directions, while target- native CPT remains strongest.

5 Conclusion

We presented a controllable framework for generating text-attributed graphs by *training* a language model rather than simulating generation with agents or prompting an external one. By modeling substructures with lossless Eulerian serializations and multiple start-node traversals, autoregressive training optimizes a variational lower bound on the permutation-invariant substructure distribution. A two-stage CPT+SFT recipe then

yields a controllable substructure generator that follows unseen prefixes in a single pass and supports both distribution-aligned generation and boundary-aware augmentation through target-side adaptation.

Across five real-world TAGs, a lightweight 3B generator emits valid, losslessly parseable graphs, matches or surpasses specialized structure-only methods on global structural fidelity, and improves GNN generalization in a disjoint train/test regime where perturbation-based augmenters often degrade. The next step is finer control: moving beyond aggregate descriptors toward richer structured and domain-specific targets, including knowledge and retrieval graphs for retrieval-augmented generation (Lewis et al., 2020b) and large-scale recommender graphs.

Limitations

Coarse, aggregate-level control. Our control interface conditions on a small set of aggregate graph descriptors (average degree, clustering, homophily, and dominant-label proportion). As the held-out controllability analysis shows, the generator tracks these targets as a coarse local controller rather than an exact regressor, and some descriptors are followed more reliably than others. Finer-grained or instance-level control—for example specifying motif counts, degree sequences, or per-node constraints—is beyond the current interface and is a natural direction for future work.

Scope of graph types. Our study targets static, undirected, node-attributed graphs with text on nodes, evaluated on five citation, web, and e-commerce datasets. We do not address dynamic or temporal graphs, edge-level text or typed multi-relational edges, or heterogeneous-node graphs, and we do not claim that the observed behavior transfers unchanged to domains with very different topology, such as molecular or knowledge graphs. Extending the serialization and control interface to these settings is left to future work.

Reliance on a separately tuned generator per dataset. To accommodate the differing descriptor ranges and text characteristics of each dataset, we train a separate generator per corpus, and the control-adherence correlations are reported within each dataset rather than across a single unified model. A single generator that transfers across heterogeneous TAGs without per-dataset tuning

would broaden applicability and is an interesting avenue for further study.

Acknowledgments

The work was supported in part by the National Key Research and Development Program of China (Grant No. 2024YFB2907000), the Basic Research Project No. HZQB-KCZY-2021067 of Hetao Shenzhen-HK S&T Cooperation Zone, the Guangdong S&T Programme (Grant No. 2024B0101030002), the National Natural Science Foundation of China (Grant No. 62293482 and Grant No. 62471423), the Shenzhen Science and Technology Program (Grant No. JCYJ20241202124021028 and Grant No. JCYJ20230807114204010), the Guangdong Talents Program (Grant No. 2024TQ08X346), the Shenzhen Outstanding Talents Training Fund 202002, the Young Elite Scientists Sponsorship Program of CAST (Grant No. 2022QNRC001), the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001) and the Shenzhen Key Laboratory of Big Data and Artificial Intelligence (Grant No. SYSPG20241211173853027).

References

- Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. 2024. [Efficient and scalable graph generation through iterative local expansion](#). In *The Twelfth International Conference on Learning Representations*.
- Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. 2024a. LLaGA: Large language and graph assistant. In *International Conference on Machine Learning (ICML)*, pages 7809–7823.
- Xiaohui Chen, Xu Han, Jiajing Hu, Francisco J. R. Ruiz, and Li-Ping Liu. 2021. Order matters: Probabilistic modeling of node sequence for graph generation. In *International Conference on Machine Learning (ICML)*.
- Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. 2023. Efficient and degree-guided graph generation via discrete diffusion modeling. In *Proceedings of the 40th International Conference on Machine Learning*, pages 4585–4610. PMLR.
- Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. 2024b. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. 2022. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2):61–77.
- Jiarui Feng, Hao Liu, Lecheng Kong, Mingfang Zhu, Yixin Chen, and Muhan Zhang. 2024. Taglas: An atlas of text-attributed graph datasets in the era of large graph and language models. *arXiv preprint arXiv:2406.14683*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2024. [Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning](#). In *The Twelfth International Conference on Learning Representations*.
- Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Jiarui Ji, Runlin Lei, Jialing Bi, Zhewei Wei, Xu Chen, Yankai Lin, Xuchen Pan, Yaliang Li, and Bolin Ding. 2025. Llm-based multi-agent systems are scalable graph generative models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1492–1523.
- Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. 2022. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*, pages 10362–10383. PMLR.
- George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392.
- Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

- Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.
- Louis Van Langendonck, Guillermo Bernardez, Nina Miolane, and Pere Barlet-Ros. 2026. [Graphuniverse: Synthetic graph generation for evaluating inductive generalization](#). In *The Fourteenth International Conference on Learning Representations*.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. [BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Mufei Li, Eleonora Kreacic, Vamsi K. Potluru, and Pan Li. 2024. [Graphmaker: Can diffusion models generate large attributed graphs?](#) *Transactions on Machine Learning Research*.
- Wen-Zhi Li, Chang-Dong Wang, Hui Xiong, and Jian-Huang Lai. 2023. Graphsha: Synthesizing harder samples for class-imbalanced node classification. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1328–1340.
- Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. 2019. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32.
- Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2024. [One for all: Towards training one graph model for all classification tasks](#). In *The Twelfth International Conference on Learning Representations*.
- Ana Lucic, Maartje ter Hoeve, Gabriele Tolomei, Maarten de Rijke, and Fabrizio Silvestri. 2022. CF-GNNExplainer: Counterfactual explanations for graph neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. 2024. Position: Graph foundation models are already here. In *International Conference on Machine Learning (ICML)*, pages 34670–34692.
- Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. 2020. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.
- OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. [Gpt-4 technical report](#). *Preprint*, arXiv:2303.08774.
- John Palowitch, Anton Tsitsulin, Brandon Mayer, and Bryan Perozzi. 2022. Graphworld: Fake graphs bring real insights for gnn. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 3691–3701.
- Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. [Droptedge: Towards deep graph convolutional networks on node classification](#). In *International Conference on Learning Representations*.
- Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. GraphGPT: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.
- Vikas Verma, Meng Qu, Kenji Kawaguchi, Alex Lamb, Yoshua Bengio, Juho Kannala, and Jian Tang. 2021. Graphmix: Improved training of gnn for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10024–10032.
- Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. 2023. Digress: Discrete denoising diffusion for graph generation. In *Proceedings of the International Conference on Learning Representations. ICLR*.
- Yu Wang, Ryan A. Rossi, Namyoung Park, Huiyuan Chen, Nesreen K. Ahmed, Puja Trivedi, Franck Dernoncourt, Danai Koutra, and Tyler Derr. 2025. [A large-scale training paradigm for graph generative models](#). In *The Thirteenth International Conference on Learning Representations*.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. [Finetuned language models are zero-shot learners](#). In *International Conference on Learning Representations*.

Brandon T Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.

Yang Yao, Xin Wang, Zeyang Zhang, Yijian Qin, Ziwei Zhang, Xu Chu, Yuekui Yang, Wenwu Zhu, and Hong Mei. 2024. Exploring the potential of large language models in graph generation. *arXiv preprint arXiv:2403.14358*.

Minji Yoon, Yue Wu, John Palowitch, Bryan Perozzi, and Russ Salakhutdinov. 2023. Graph generative model for benchmarking graph neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, pages 40175–40198. PMLR.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR.

Jianxiang Yu, Yuxiang Ren, Chenghua Gong, Jiaqi Tan, Xiang Li, and Xuechang Zhang. 2025. Leveraging large language models for node generation in few-shot learning on text-attributed graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pages 13087–13095.

Taiyan Zhang, Renchi Yang, Yurui Lai, Mingyu Yan, Xiaochun Ye, and Dongrui Fan. 2025. Leveraging large language models for effective label-free node classification in text-attributed graphs. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2023. [Learning on large-scale text-attributed graphs via variational inference](#). In *The Eleventh International Conference on Learning Representations*.

Qifang Zhao, Weidong Ren, Tianyu Li, Hong Liu, Xingsheng He, and Xiaoxiao Xu. 2025. Graphgpt: Generative pre-trained graph eulerian transformer. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 77630–77656. PMLR.

Tong Zhao, Gang Liu, Stephan Günnemann, and Meng Jiang. 2022. Graph data augmentation for graph machine learning: A survey. *arXiv preprint arXiv:2202.08871*.

Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2021. Data augmentation for graph neural networks. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pages 11015–11023.

Mengting Zhou and Zhiguo Gong. 2023. Graphsr: A data augmentation algorithm for imbalanced node classification. In *Proceedings of the AAAI Conference on artificial intelligence*, volume 37, pages 4954–4962.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

A Proof of Theorem 1

This appendix gives the full derivation behind Theorem 1. We restate the setting, prove the marginalization identity used in the main text, derive the evidence lower bound and its optimal variational distribution in full, specialize it to the uniform distribution that our continual-pretraining objective implements, and finally state a monotonicity property that motivates enumerating all N start-node circuits.

Throughout, G_s is a connected substructure with $N = |V(G_s)|$ nodes; $\mathcal{A}$ is the set of node permutations with $|\mathcal{A}| = N!$; and $f : (G_s, \pi) \mapsto W_\pi$ is the Eulerian serialization map of §3.2, which is a lossless bijection on its admissible domain (Definition 2, Property 1).

A.1 Serialization Decomposition

Proposition 1 (Marginalization over serializations). *Let the serialization choice π be a latent variable with prior $p(\pi)$ supported on a serialization set $\Sigma(G_s) \subseteq \mathcal{A}$. If f is lossless on $\Sigma(G_s)$, then*

$$p(G_s) = \sum_{\pi \in \Sigma(G_s)} p(\pi) p_\theta(W_\pi). \quad (2)$$

Proof. The generative process draws a serialization choice $\pi \sim p(\pi)$ and then emits a token sequence $W \sim p_\theta(\cdot)$. Because f is a bijection between $\Sigma(G_s)$ and the corresponding set of valid sequences, the event “the emitted sequence decodes to G_s ” is the disjoint union

$$\bigsqcup_{\pi \in \Sigma(G_s)} \{W = W_\pi\}.$$

Each admissible π yields exactly one sequence W_π , and distinct choices yield distinct sequences by injectivity, while every valid sequence decodes back to G_s by surjectivity onto the fibre of G_s . Summing the joint probability over this disjoint union gives

$$\begin{aligned} p(G_s) &= \sum_{\pi \in \Sigma(G_s)} p(\pi) p_\theta(W = W_\pi | \pi) \\ &= \sum_{\pi \in \Sigma(G_s)} p(\pi) p_\theta(W_\pi), \end{aligned} \quad (3)$$

where $p_\theta(W_\pi \mid \pi) = p_\theta(W_\pi)$ since W_π is deterministic given π . $\square$

The full permutation marginal is the special case $\Sigma(G_s) = \mathcal{A}$ with the uniform prior $p(\pi) = 1/N!$, absorbed into the joint $p(G_s, \pi)$.

A.2 Evidence Lower Bound

Fix any non-empty serialization set $\tau \subseteq \mathcal{A}$ and let $q_\pi \triangleq q(\pi \mid G_s)$ be a variational distribution with $\text{supp}(q) = \tau$ and $\sum_{\pi \in \tau} q_\pi = 1$. Starting from the marginal and inserting q_π ,

$$\begin{aligned} \log p(G_s) &= \log \sum_{\pi \in \tau} p(G_s, \pi) \\ &= \log \sum_{\pi \in \tau} q_\pi \frac{p(G_s, \pi)}{q_\pi}. \end{aligned} \quad (4)$$

The summand $p(G_s, \pi)/q_\pi$ is averaged under the distribution q , so Equation (4) is the logarithm of an expectation. Since $\log$ is concave, Jensen's inequality gives

$$\begin{aligned} \log p(G_s) &\geq \sum_{\pi \in \tau} q_\pi \log \frac{p(G_s, \pi)}{q_\pi} \\ &\triangleq \mathcal{F}(q). \end{aligned} \quad (5)$$

The gap is exactly the KL divergence between q and the τ -restricted posterior; equivalently $\mathcal{F}(q)$ is the standard ELBO with the serialization choice as the latent variable.

A.3 Optimal Variational Distribution

We maximize $\mathcal{F}(q)$ subject to normalization. The Lagrangian with multiplier λ is

$$\mathcal{L}(q, \lambda) = \sum_{\pi \in \tau} q_\pi \log \frac{p(G_s, \pi)}{q_\pi} + \lambda \left(1 - \sum_{\pi \in \tau} q_\pi \right). \quad (6)$$

Differentiating with respect to a single q_π and using

$$\begin{aligned} \frac{\partial}{\partial q_\pi} \left(q_\pi \log \frac{p(G_s, \pi)}{q_\pi} \right) &= \log p(G_s, \pi) \\ &\quad - \log q_\pi - 1, \end{aligned}$$

we obtain

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial q_\pi} &= \log p(G_s, \pi) - \log q_\pi - 1 - \lambda \\ &= 0. \end{aligned} \quad (7)$$

Solving for q_π gives

$$\begin{aligned} q_\pi &= \exp(\log p(G_s, \pi) - 1 - \lambda) \\ &= \frac{p(G_s, \pi)}{e^{1+\lambda}}. \end{aligned} \quad (8)$$

Imposing $\sum_{\pi \in \tau} q_\pi = 1$ fixes the multiplier:

$$\begin{aligned} e^{1+\lambda} &= \sum_{\pi' \in \tau} p(G_s, \pi') \\ &\triangleq Z. \end{aligned} \quad (9)$$

Thus the optimal variational distribution is the τ -restricted posterior

$$q_\pi^* = \frac{p(G_s, \pi)}{Z}, \quad Z = \sum_{\pi' \in \tau} p(G_s, \pi'). \quad (10)$$

Substituting q_π^* into $\mathcal{F}$ and expanding,

$$\begin{aligned} \mathcal{F}(q^*) &= \sum_{\pi \in \tau} q_\pi^* \log \frac{p(G_s, \pi)}{q_\pi^*} \\ &= \sum_{\pi \in \tau} \frac{p(G_s, \pi)}{Z} \log \frac{p(G_s, \pi)}{p(G_s, \pi)/Z} \\ &= \sum_{\pi \in \tau} \frac{p(G_s, \pi)}{Z} \log Z \\ &= \log Z. \end{aligned} \quad (11)$$

The middle step uses $p(G_s, \pi)/q_\pi^* = Z$ for every $\pi \in \tau$, and the last step uses $\sum_{\pi \in \tau} p(G_s, \pi)/Z = 1$. Hence the tightest bound attainable on τ is

$$\log p(G_s) \geq \log \sum_{\pi \in \tau} p(G_s, \pi), \quad (12)$$

which is the log-partition form summarized in the main text.

A.4 Eulerian Instantiation and the Training Objective

We instantiate τ with the N Eulerian circuits obtained by taking each node of G_s as the traversal origin,

$$\tau = \{\pi_1, \dots, \pi_N\}, \quad |\tau| = N \ll N!,$$

matching the variational set used in Theorem 1. By Property 1 the Eulerization edges are removable, so f is lossless on τ and

$$p(G_s, \pi_i) = p_\theta(W_{\pi_i}),$$

with the standard autoregressive factorization of the token likelihood.

Our continual-pretraining objective weights the N circuits uniformly, i.e., it adopts $q_{\pi_i} = 1/N$ rather than the optimal q^* . Substituting this choice into Equation (5),

$$\begin{aligned}\log p(G_s) &\geq \sum_{i=1}^N \frac{1}{N} \log \frac{p_\theta(W_{\pi_i})}{1/N} \\ &= \frac{1}{N} \sum_{i=1}^N \left(\log p_\theta(W_{\pi_i}) + \log N \right) \\ &= \frac{1}{N} \sum_{i=1}^N \log p_\theta(W_{\pi_i}) + \log N, \quad (13)\end{aligned}$$

recovering Equation (1). Because $\log N$ does not depend on θ , minimizing the token-level cross-entropy averaged over the N circuits,

$$\mathcal{L}_{\text{CE}}(\theta) = -\frac{1}{N} \sum_{i=1}^N \sum_{t=1}^{|W_{\pi_i}|} \log p_\theta \left(w_t^{(\pi_i)} \mid w_{<t}^{(\pi_i)} \right), \quad (14)$$

maximizes the uniform- q variational lower bound in Equation (13). The uniform choice is a deliberate simplification: it avoids the computationally complex repeated updates of the model-dependent q^* , while still optimizing a valid bound. It also treats every start node symmetrically, which is the behavior we want from a permutation-agnostic model.

A.5 Monotonicity in the Serialization Set

Section 3.3 claims that the bound tightens as more circuits are added; we make this precise. The monotonicity is a property of the *tightest attainable* bound on a serialization set, i.e. the log-partition form of Equation (12) obtained at the optimal q^* ,

$$\mathcal{B}(\tau) \triangleq \log \sum_{\pi \in \tau} p(G_s, \pi) = \log \sum_{\pi \in \tau} p_\theta(W_\pi), \quad (15)$$

where the second equality uses the losslessness of f on τ (Property 1), giving $p(G_s, \pi) = p_\theta(W_\pi)$.

Proposition 2 (Monotonicity). *Let $\tau \subseteq \tau' \subseteq \mathcal{A}$ be two non-empty serialization sets on which f is lossless. Then*

$$\mathcal{B}(\tau) \leq \mathcal{B}(\tau') \leq \log p(G_s). \quad (16)$$

In particular, enumerating additional start-node circuits never loosens the attainable bound, and the bound is sandwiched below the true log-likelihood for every τ .

Proof. Each token-sequence likelihood is non-negative, $p_\theta(W_\pi) \geq 0$, so adding the terms indexed by $\tau' \setminus \tau$ to the sum in Equation (15) cannot decrease it: $\sum_{\pi \in \tau} p_\theta(W_\pi) \leq \sum_{\pi \in \tau'} p_\theta(W_\pi)$. Monotonicity of $\log$ gives the left inequality of Equation (16). For the right inequality, apply Proposition 1 with $\Sigma(G_s) = \mathcal{A}$ and the full permutation marginal: since $\tau' \subseteq \mathcal{A}$ and all terms are non-negative, $\sum_{\pi \in \tau'} p_\theta(W_\pi) \leq \sum_{\pi \in \mathcal{A}} p(G_s, \pi) = p(G_s)$, and applying $\log$ closes the chain. $\square$

Why enumerate all N start nodes. Proposition 2 justifies the continual-pretraining design of §3.4: among the affordable $O(N)$ serialization sets, the one containing all N start-node circuits is maximal, hence yields the tightest attainable bound $\mathcal{B}(\tau)$. Two remarks delimit the claim. First, monotonicity is stated for the optimal- q bound $\mathcal{B}(\tau)$, not for the specific uniform- q training objective of Equation (13): the latter carries an explicit $+\log N$ entropy term whose argument changes with $|\tau|$, so it is the *reachable* bound, rather than the realized value at a fixed weighting, that improves monotonically. Second, the gap $\log p(G_s) - \mathcal{B}(\tau)$ vanishes only as $\tau \uparrow \mathcal{A}$; modeling substructures keeps N small enough that enumerating all N start-node circuits is tractable while still moving $\mathcal{B}(\tau)$ toward $\log p(G_s)$. This is the precise sense in which the bound “provably tightens as more circuits are added.”

B Held-Out Controllability: Full Details

B.1 Protocol

This probe uses the final CPT+SFT model for each dataset. For each dataset and each descriptor in $\{\text{avg_deg}, \text{cc}, \text{homophily}\}$, we request the 20/50/80 percentiles of that descriptor under the SFT control-prefix distribution, while fixing the other two descriptors at their dataset medians. The dominant label and dominant-label proportion are also fixed to the modal dominant label and median dominant proportion from the SFT corpus. If a formatted prefix occurs verbatim in the SFT corpus, the swept value is nudged by 0.01 when possible, so the tested control string is not a memorized training prefix.

We run 4 stochastic generations per grid point, giving $5 \times 3 \times 3 \times 4 = 180$ raw LLM generations. Decoding uses the same dataset-specific maximum token budget as distribution-aligned generation, with temperature 0.9 and top- $p = 0.9$. This

probe tests the capability of the LLM substructure generator.

B.2 Strict Parse and Lossless Criteria

Parse rate is the fraction of raw generations that can be converted into a graph topology under strict grammar checks. A generation counts as parseable only if it terminates with `<path_end>`, contains valid node tokens, has at least three distinct nodes, and has exactly one `<e_real>` or `<e_jump>` token between each consecutive pair of node tokens. Only `<e_real>` tokens become graph edges; jump tokens are used to preserve the Eulerian path grammar.

Lossless rate uses all raw generations as the denominator. A generation is lossless only if it is parseable, every first occurrence of a node has a valid attribute dictionary with label and text fields, repeated nodes do not re-emit conflicting attributes, there is no extra junk inside structural segments, and canonical serialize–parse round-tripping preserves the path nodes, edge tokens, and attributes. **Conditional lossless rate** is the same lossless count divided by the number of parseable generations. Thus a dataset can have a low parse rate but 100% conditional lossless rate if all successfully parsed generations round-trip exactly; CiteSeer in Table 4 is such a case. The dominant failure mode in the raw probe is a malformed edge sequence, usually a missing edge token after a long attribute block.

Dataset	Parse	Lossless	Cond.
Cora	97.2	94.4	97.1
CiteSeer	63.9	63.9	100.0
PubMed	94.4	83.3	88.2
Products	61.1	55.6	90.9
WikiCS	100.0	100.0	100.0
All	83.3	79.4	95.3

Table 4: Raw generation validity per dataset (%). *Parse*: parses into a graph under strict rules; *Lossless*: parses and round-trips losslessly; *Cond.*: lossless rate among parseable samples only.

B.3 Full Control-Adherence Sweep

For the summary in Figure 3, we use only parseable samples. For each dataset, descriptor, and requested percentile, we average the realized descriptor over the 4 repeats. Pearson and Spearman correlations are then computed over the three requested grid levels, and MAE is the mean absolute difference between the requested grid value

and the corresponding mean realized value. We use grid-level averages rather than all individual samples so that a single high-variance decoding repeat does not dominate the correlation. The raw per-generation records are kept in the supplementary artifacts.

C Dataset Split and Structural Fidelity Results

C.1 Split Rationale and Configuration

We use a community-aware stratified split because a purely random split leaks local neighborhoods across train and test, while a pure community split can introduce large label shift. We first partition the graph with METIS-style sparse partitioning, then split each (label, community) block into train and test nodes using a 0.66/0.34 ratio. The resulting train and test graphs are induced subgraphs over disjoint node sets. The split seed is 42. The number of communities is 5 for Cora, Products, and WikiCS; 3 for CiteSeer; and 10 for PubMed. For GNN training on the real train graph, a stratified 20% validation split is carved from the training nodes.

C.2 Per-Class Label Distribution

The split is stratified inside each label, so every class present in the processed graph is represented in both train and test whenever the class has enough nodes. This matters for both tasks: the distribution-aligned generator receives label quotas from the reference split, and the augmentation evaluation must test on labels that were visible during training without reusing their nodes or edges. We therefore report node and edge counts in Table 1 and use the same split object for all MMD, augmentation, and ablation runs.

C.3 MMD Sampling Protocol

For each generated graph and reference split we evaluate a many-vs-many MMD. In each repeat, we draw connected BFS-induced subgraphs of 200 nodes from both graphs, relabel them compactly, and compute MMD with the Gaussian-EMD kernel used in GraphRNN-style evaluation. Cora and CiteSeer use 20 real and 20 generated samples per repeat; PubMed, Products, and WikiCS use 50 real and 20 generated samples. We average over 5 sampling repeats and report the mean. The four reported statistics are degree distribution, clustering

Dataset	Type	Method	Generated graph vs train				Generated graph vs test			
			Deg.	Clus.	Orb.	Spec.	Deg.	Clus.	Orb.	Spec.
Cora	Autoreg.	GraphRNN	0.0057	1.5064	1.0987	0.0918	0.0281	1.8072	1.6995	0.1366
		GRAN	0.0189	0.6779	0.2383	0.0836	0.0675	1.0784	0.7898	0.1391
	Diffusion	DiGress	0.1055	1.5352	1.0051	0.2122	0.1582	1.5525	1.7366	0.2810
		GraphMaker	0.0262	1.5078	0.7455	0.0356	0.0089	1.7159	1.6675	0.0675
	LLM	GAG	0.3500	0.8400	1.2100	0.4100	—	—	—	—
		Ours	0.0055	0.2030	0.9481	0.0603	0.0540	0.8686	1.3219	0.1468
CiteSeer	Autoreg.	GraphRNN	0.0306	1.0337	0.9275	0.0612	0.6053	0.3204	0.7845	0.6995
		GRAN	0.0522	0.5285	0.3573	0.1618	0.5992	1.1515	1.0530	0.7218
	Diffusion	DiGress	0.2119	1.1514	1.3033	0.2256	0.7109	1.9776	2.0000	0.7491
		GraphMaker	0.0292	1.3118	0.8608	0.0543	0.6123	0.0016	0.2578	0.7003
	LLM	GAG	0.1600	0.1900	1.0200	0.3200	—	—	—	—
		Ours	0.0395	0.4837	0.2330	0.0659	0.0320	0.4441	0.1362	0.1040
PubMed	Autoreg.	GraphRNN	0.0686	0.9221	0.9770	0.1458	0.0766	0.9355	0.9829	0.1401
		GRAN	0.1762	0.2430	0.1016	0.0558	0.2926	0.4397	0.1410	0.1083
	Diffusion	DiGress	0.3068	1.1451	1.0516	0.2864	0.3400	1.3180	1.0895	0.3659
		GraphMaker	0.0263	0.8747	0.7780	0.0313	0.0125	0.9592	0.7417	0.0182
	LLM	GAG	—	—	—	—	—	—	—	—
		Ours	0.0323	0.3904	0.2495	0.0485	0.0633	0.6740	0.7377	0.1098
Products	Autoreg.	GraphRNN	0.1188	0.1401	0.1854	0.1590	0.7387	0.5243	0.8299	0.7220
		GRAN	0.1263	0.9246	1.0402	0.2213	0.6974	1.0705	1.0500	0.7387
	Diffusion	DiGress	0.0866	0.0798	0.0325	0.1894	0.5874	0.2347	0.3122	0.5546
		GraphMaker	0.0130	0.1428	0.0503	0.0534	0.7399	0.0039	0.6393	0.7366
	LLM	GAG	—	—	—	—	—	—	—	—
		Ours	0.0467	1.0259	1.0480	0.0596	0.0567	1.2775	0.4149	0.0739
WikiCS	Autoreg.	GraphRNN	0.1399	0.0889	0.0705	0.0773	0.2300	0.0968	0.0711	0.1119
		GRAN	0.1047	0.0975	0.0704	0.0641	0.2289	0.1121	0.0711	0.1295
	Diffusion	DiGress	0.4695	1.0232	1.0204	0.1149	0.5759	1.0325	1.0211	0.1690
		GraphMaker	0.2694	0.7432	0.2025	0.2291	0.1963	0.8166	0.2995	0.1675
	LLM	GAG	—	—	—	—	—	—	—	—
		Ours	0.0774	0.0722	0.0700	0.0722	0.0615	0.0760	0.1822	0.0573

Table 5: Structural fidelity by MMD ($\downarrow$) against train and test splits. Best per column is in **bold**; “—” marks unavailable GAG runs.

coefficients, 4-node orbit counts, and Laplacian spectral features.

Dataset	Comm.	Train/Test	Real/Gen.	Repeats
Cora	5	0.66/0.34	20/20	5
CiteSeer	3	0.66/0.34	20/20	5
PubMed	10	0.66/0.34	50/20	5
Products	5	0.66/0.34	50/20	5
WikiCS	5	0.66/0.34	50/20	5

Table 6: Split and sampled-MMD settings. “Real/Gen.” gives the number of 200-node sampled subgraphs drawn per repeat from the reference and generated graph respectively.

C.4 Packet-Only and Assembled-Graph Comparison

This same-scale comparison isolates the local structure generated by the LM from the subsequent assembly stage. For every generated packet, we draw size-matched BFS-induced subgraphs from the real split and the assembled graph. The same real samples are reused over five repeats. Each MMD cell in Table 7 reports *packet-only / assembled*; lower is better. The packet count varies with the generated node budget and packet sizes and is reported only to specify the evaluated sample collection.

D GNN Generalization: Full Results

All entries use the fixed train/test split and the same five GNN training seeds.

E Ablation: Continual Pretraining

The *w/o CPT* variant starts from the same base LLM after expanding the tokenizer with graph-serialization tokens, then trains directly on the SFT corpus. It does not initialize from any continual-pretrained checkpoint. Generation, VGAE assembly, split protocol, MMD evaluation, and decoding budgets are otherwise kept identical to the full CPT+SFT pipeline.

E.1 Multi-Start Controlled Comparison

We separate traversal diversity from corpus size by retaining one fixed serialization per substructure in the single-start condition and repeating it to the size of its corresponding multi-start group. Single- and multi-start CPT therefore use the same substructures, line count, total token count, and optimizer steps. All three conditions in Table 2 then use the same target SFT corpus and held-out split. The reported quantity is final SFT held-out next-token loss on this common split. Multi-start obtains the lowest loss on all five datasets. Across the 180 held-out generation attempts, aggregate parse/lossless rates are 80.6/74.4% for single-start and 83.3/79.4% for multi-start.

E.2 Bidirectional Citation-Domain Transfer

Table 3 tests transfer between Cora and CiteSeer, the two citation TAGs in our evaluation. Cross-CPT first performs CPT on the source dataset and

Dataset	Split	Packets	Degree	Clustering	Orbit	Spectral
Cora	train	122	.0390/.0199	.0998/.0753	.0307/.0183	.0450/.0345
Cora	test	66	.1238/.0959	.1321/.1027	.0800/.0673	.0914/.0748
CiteSeer	train	265	.0428/.0186	.0706/.0646	.0050/.0034	.0360/.0352
CiteSeer	test	132	.0535/.0459	.1351/.0566	.0107/.0029	.0549/.0479
PubMed	train	329	.0816/.0183	.3219/.1499	.0402/.0195	.0949/.0353
PubMed	test	326	.0599/.0663	.2353/.2967	.0328/.0250	.0719/.0758
Products	train	402	.0460/.0364	.1658/.1428	.0023/.0020	.0444/.0322
Products	test	400	.0650/.0762	.2000/.2490	.0094/.0118	.0503/.0655
WikiCS	train	298	.1015/.0298	.1050/.0992	.1302/.0768	.0600/.0310
WikiCS	test	313	.1242/.0399	.1123/.0972	.1477/.0858	.0852/.0276

Table 7: Same-scale structural MMD for LM-generated packets and assembled graphs against real BFS-induced samples. Each metric is *packet-only* / *assembled*; lower is better.

Dataset	Method	ΔAcc	$\Delta\text{Macro-F1}$
Cora	DropEdge	$+1.72 \pm 0.80$	$+1.52 \pm 1.44$
	GraphSHA	-3.23 ± 2.61	-5.17 ± 2.86
	GAug	$+0.11 \pm 1.00$	-0.23 ± 1.24
	LLM4NG	-1.04 ± 0.92	-1.28 ± 1.49
	Ours	$+1.76 \pm 1.06$	$+2.93 \pm 1.13$
CiteSeer	DropEdge	$+0.61 \pm 0.36$	-0.44 ± 1.08
	GraphSHA	$+0.33 \pm 0.75$	$+1.06 \pm 0.98$
	GAug	$+0.55 \pm 1.00$	-0.66 ± 1.04
	LLM4NG	$+0.17 \pm 0.64$	-0.08 ± 0.67
	Ours	$+0.17 \pm 0.52$	-0.27 ± 0.68
PubMed	DropEdge	$+0.82 \pm 0.57$	$+0.76 \pm 0.54$
	GraphSHA	-0.15 ± 0.39	-0.12 ± 0.39
	GAug	-0.18 ± 0.66	-0.14 ± 0.60
	LLM4NG	-0.11 ± 0.88	-0.10 ± 0.85
	Ours	$+0.60 \pm 0.45$	$+0.57 \pm 0.41$

Dataset	Method	ΔAcc	$\Delta\text{Macro-F1}$
Products	DropEdge	$+0.25 \pm 0.38$	$+0.16 \pm 0.64$
	GraphSHA	-11.25 ± 0.70	-15.96 ± 0.52
	GAug	-0.11 ± 0.50	-0.22 ± 0.75
	LLM4NG	$+0.09 \pm 0.44$	$+0.66 \pm 0.42$
	Ours	$+1.20 \pm 0.44$	$+4.14 \pm 0.67$
WikiCS	DropEdge	$+0.38 \pm 0.48$	-1.09 ± 0.97
	GraphSHA	-6.30 ± 0.31	-9.26 ± 0.85
	GAug	-2.88 ± 0.62	-8.23 ± 1.02
	LLM4NG	-0.12 ± 0.36	-0.09 ± 0.75
	Ours	$+1.06 \pm 0.59$	$+1.34 \pm 0.50$

Table 8: GNN generalization under the train-augmented / test-on-held-out protocol (GCN). Each entry is the mean $\pm$ standard deviation over five seeds for the change relative to the same-seed un-augmented train graph, in percentage points; higher is better. Best mean per dataset and metric is in **bold**.

Dataset	Variant	Generated graph vs train				Generated graph vs test			
		Deg.	Clus.	Orb.	Spec.	Deg.	Clus.	Orb.	Spec.
Cora	Full (CPT+SFT)	0.0055	0.2030	0.9481	0.0603	0.0540	0.8686	1.3219	0.1468
	w/o CPT	0.0062	0.2052	0.9532	0.0663	0.0573	0.9927	1.3225	0.1308
CiteSeer	Full (CPT+SFT)	0.0395	0.4837	0.2330	0.0659	0.0320	0.4441	0.1362	0.1040
	w/o CPT	0.0416	0.5791	0.6180	0.0518	0.0411	0.6611	0.1111	0.1111
PubMed	Full (CPT+SFT)	0.0323	0.3904	0.2495	0.0485	0.0633	0.6740	0.7377	0.1098
	w/o CPT	0.0413	0.2177	0.2530	0.0580	0.0689	0.5386	1.1400	0.1203
Products	Full (CPT+SFT)	0.0467	1.0259	1.0480	0.0596	0.0567	1.2775	0.4149	0.0739
	w/o CPT	0.0469	1.3102	1.0692	0.0545	0.0663	1.2808	0.0071	0.0861
WikiCS	Full (CPT+SFT)	0.0774	0.0722	0.0700	0.0722	0.0615	0.0760	0.1822	0.0573
	w/o CPT	0.1145	0.0727	0.0700	0.0949	0.0816	0.0754	0.0700	0.0634

Table 9: Effect of continual pretraining across all datasets. MMD ($\downarrow$) on both train and test splits. *w/o CPT* fine-tunes the base LLM directly on the control-prefixed corpus, skipping continual pretraining; all else fixed. Best per column in **bold**.

then SFT and evaluation on the target; SFT-only omits CPT, and Native-CPT uses the target’s own CPT checkpoint. Target SFT data and settings are shared within each row. Cross-CPT lowers held-out loss relative to SFT-only in both directions and increases parse rate by 16.7 and 11.1 percentage points, while Native-CPT remains the lowest-loss initialization. This result indicates transfer of serialization grammar and substructure priors between related citation TAGs. A unified model across heterogeneous TAG domains would additionally require shared domain/style representations, composable label schemas, cross-graph descriptor nor-

malization, and a balanced mixed training corpus.

F Corpus Construction Details

F.1 Substructure Sampling

For each dataset we sample local substructures from the processed source TAG by short random walks, then take the induced subgraph over the visited nodes. The walk-size distribution is dataset-specific: sparse citation graphs use smaller packets, while WikiCS uses larger packets to expose the model to its denser local neighborhoods. We keep the sampled subgraph, not the walk trace it-

Dataset	Source and text processing	Comm.	Packet sizes	SFT balancing budget	Sequence guard
Cora	cora_node; head-tail text cap 128	5	10–48	6K walks; bin cap 25; 350 seq./class	drop over-length paths
CiteSeer	citeseer; head-tail text cap 256	3	8–26	6K walks; bin cap 25; 350 seq./class	keep 1.2K–6.5K-token paths
PubMed	pubmed_node; head-tail text cap 320	10	8–20	8K walks; bin cap 40; 900 seq./class	max path length 6.6K tokens
Products	products; remove long-text outliers, stratified 30K sub-sample, LCC, min-class filter; text cap 160	5	10–28	20K walks; bin cap 50; 120 seq./class	keep 1.2K–6.6K-token paths
WikiCS	wikics; BART summary cache (1024-token input, 60–140-token output, 2 beams); text cap 192	5	14–52	8K walks; bin cap 30; 250 seq./class	drop over-length paths

Table 10: Corpus construction and preprocessing settings. CPT uses the same packet-size distribution with 100 sampling rounds per dataset; the SFT budget controls the conditional corpus for single-pass controllable generation.

self, so repeated visits only affect which nodes enter the packet; edges are always the real induced edges before Eulerization. The exact size probabilities and sampling budgets are fixed in the released configuration.

F.2 Multi-Start Serialization and Corpus Balancing

CPT and SFT use the same lossless path grammar but differ in how aggressively they enumerate serializations. CPT is meant to learn the unconditional graph grammar, so it enumerates the available start-node circuits of each sampled substructure whenever the sequence-length budget permits. SFT instead uses a bounded set of start nodes per substructure: the objective is no longer to enlarge the variational set indefinitely, but to expose a broad and balanced set of control prefixes. To reduce majority-label collapse, SFT uses stratified start-node sampling, bins substructures by degree, homophily, and dominant label, caps overpopulated bins, and adds targeted walks around rare labels. Control fields are computed after sampling, so the prefix always describes the serialized substructure rather than a pre-specified template.

F.3 Data Preprocessing

All node texts are normalized through the same tokenizer used for LLM training, and long texts are shortened with a head-tail truncation rule to preserve the beginning of the document plus a small amount of tail context. Because WikiCS entries are long, we summarize them with facebook/bart-large-cnn (Lewis et al., 2020a) before corpus construction (1024-token input cap, 60–140 output tokens, 2 beams). For ogbn-Products we remove extreme text-length outliers, take a stratified 30K-node subset, keep its largest connected component, and filter very small classes. These preprocessing steps are shared by

training, generation, VGAE assembly, and evaluation.

F.4 Sequence-Length Control

Training examples that exceed the dataset-specific maximum context are dropped rather than truncated inside a serialization, because cutting a path can destroy the parseable graph grammar. The same principle is used at inference time: each dataset keeps its generation-time maximum token budget, and the held-out controllability probe refuses to lower that budget. This makes invalid half-path outputs a property of the model rather than an artifact of an artificially short decoding cap.

G Model Input–Output Interface

Base LM and optimization. All generators start from Qwen2.5-3B (Qwen et al., 2025). We add the serialization and control symbols in §3.4 as special tokens, including path delimiters, edge markers, attribute delimiters, control delimiters, and local node-index tokens $\langle n0 \rangle$ – $\langle n200 \rangle$; the token embeddings are resized before training. We use full-parameter fine-tuning, not LoRA, in bfloat16 with DeepSpeed ZeRO-2. CPT runs for 5 epochs and SFT for 3 epochs with AdamW, learning rate $2 \cdot 10^{-5}$, weight decay 0.01, warmup ratio 0.03, per-device batch size 1, gradient accumulation 8 or 16 depending on the sequence budget, early stopping on held-out next-token loss, and seed 42. Maximum sequence lengths are dataset-specific, ranging from 5120 to 6656 tokens. Each CPT/SFT run used two NVIDIA RTX A6000 GPUs; TensorBoard logs report per-dataset wall-clock training times of about 2.2–4.3 h for CPT and 1.2–2.4 h for SFT. All source benchmarks come from public TAGLAS/OGB resources (MIT-licensed repositories), and Qwen2.5-3B is released under the Qwen Research License for research use.

The generator is always used as a causal continuation model. CPT is therefore not an encoder–decoder “input–output” task: the model is trained to continue full Eulerian-path documents under the standard next-token objective. SFT keeps the same continuation format but prepends a control header as left context. At inference time we provide the control header and `<path_start>`; the continuation is then parsed into a graph object.

CPT. The sequence is an unconditional path document from `<path_start>` to `<path_end>`. There is no control header and no source–target split; every token in the serialized path is a next-token target.

SFT. The same path grammar is kept, but a control header is prepended as left context. Header tokens condition the continuation and are masked from the loss; graph tokens remain standard next-token targets.

Held-out controllability probe. The prompt encodes a generation intent through structural descriptors, followed by `<path_start>`. We decode one stochastic continuation under the dataset-specific token budget and send it directly to the strict parser.

Applications. Distribution-aligned generation and TAG-AUG use the same decoder; the former uses a header whose structural-descriptor targets are adjusted by PI from the current generation state, while the latter uses a header constructed from the boundary profile. Their outer loops may retry, filter, or reject packets, but the LLM still only produces a serialized continuation.

H Distribution-Aligned Generation: Full Algorithm and Settings

H.1 Full Algorithm

Here d^* , c^* , and h^* denote the target average degree, clustering coefficient, and homophily measured from the reference graph G^* , respectively. The label distribution is denoted by $\mathcal{P}_\ell$.

The generation loop has two safeguards that are easy to miss in the main text. First, failed parses are not inserted into the graph; the loop simply spends another attempt under the current feedback state. Second, the assembly step only scores cross-substructure pairs, so the LLM controls local intra-packet structure while the VGAE supplies global connectivity.

Algorithm 1 Distribution-Aligned TAG Generation

Require: SFT generator $\mathcal{M}$, VGAE $\mathcal{G}$, reference graph G^* , node budget N
Ensure: Generated TAG G_{gen}

- 1: Measure target (d^*, c^*, h^*) and label distribution $\mathcal{P}_\ell$ from G^*
- 2: Initialize label quotas, generated counts, and PI integrals for degree, clustering, and homophily
- 3: $\mathcal{S} \leftarrow \emptyset$
- 4: **while** generated nodes $< N$ **do**
- 5: Select the label with the largest remaining quota
- 6: Refresh running degree every step and clustering/homophily every r accepted packets
- 7: Update PI errors and clip requested (d, c, h) to valid dataset-specific ranges
- 8: Use SFT generator $\mathcal{M}$ to autoregressively decode one packet from the `avg_deg/cc/homophily/dom` prefix
- 9: Strictly parse it; on failure, record the failure and continue
- 10: Add the parsed packet to $\mathcal{S}$ and update node, edge, and label counts
- 11: **end while**
- 12: Merge accepted packets into a disconnected graph G_0
- 13: Encode node texts and intra-packet edges with $\mathcal{G}$
- 14: Optionally add VGAE edges for largest-component repair
- 15: Add cross-packet VGAE edges until the edge budget implied by d^* is reached
- 16: **return** the graph assembled from accepted packets and VGAE cross-packet edges, with generated texts, labels, and edges

H.2 Hyperparameters

Default decoding uses temperature 0.9 and top- $p = 0.9$. The PI controller uses proportional gain 0.3, integral gain 0.1, and integral clipping at 3.0; target ranges for degree, clustering, homophily, dominant-label proportion, token budget, and optional connectivity repair are dataset-specific. VGAE assembly uses BERT (Devlin et al., 2019) text embeddings, a two-layer variational GCN encoder with 512 hidden dimensions and 256 latent dimensions, dropout 0.2, Adam learning rate 0.01, and early stopping on validation link prediction. All values used to reproduce the tables are specified in the configuration file rather than repeated here.

I Boundary-Aware Augmentation: Full Algorithm and Settings

I.1 Full Algorithm

The augmentation graph keeps edge types: real training edges, packet internal edges, and real-to-packet anchor edges. During GNN evaluation, train and validation nodes come from the augmented training graph, while the final test set is always the held-out real test graph.

I.2 Hyperparameters

The oracle is a GCN with 256 hidden dimensions, dropout 0.5, learning rate 0.005, weight decay $5 \cdot 10^{-4}$, and early stopping on the real-train validation split. The default boundary fraction is 0.25, profile blend weight is 0.5, minimum packet size is 3, maximum unknown-label ratio is 0.2, and minimum non-empty-text ratio is 0.8. Dataset-specific

Stage	Content
Requested control	<ctrl_start>avg_deg:2.63 cc:0.19 homophily:0.65 dom:Rule Learning:0.53<ctrl_end><path_start>
Model output	<n15><n_attr_start>{"label":"Theory","text":"Academic paper ... dataset decomposition approach ..."}<n_attr_end> <e_real><n17><n_attr_start>{"label":"Theory","text":"Academic paper ... Pattern Theoretic Learning ..."}<n_attr_end> <e_real><n15><e_real><n17><e_real><n10> <n_attr_start>{"label":"Rule Learning","text":"Academic paper ... rule induction and refinement ..."}<n_attr_end> ... <e_real><n0><path_end>
Parse result	Strict parsing yields a TAG with $ V = 12$ and $ E = 15$. Only <e_real> transitions are retained as edges; node labels and texts are read from first-occurrence attribute blocks, and later occurrences such as <n15> and <n17> are treated as references rather than new nodes.
Response analysis	The substructure has avg. degree 2.50, clustering coefficient 0.00, and homophily 0.67. Its dominant generated labels include Rule Learning nodes (n0, n2, n3, n4, n5, n6) together with Theory nodes introduced by the model. Representative recovered edges include (0, 5), (0, 12), (2, 15), (3, 4), (3, 6), (4, 5), (5, 6), (5, 15), (6, 8), and (6, 12).

Table 11: Cora generation case under a held-out control prefix. Long paper texts are abridged; structural tokens, parse result, and measurements follow the released raw artifact.

Dataset	N	Max tok.	r	Repair	Deg. clip	Clus. clip	Hom. clip
Cora	900	5632	3	train/test	[1.90, 3.94]	split-specific	[0.36, 0.93]
CiteSeer	1090	5800	3	train	[1.80, 3.85]	[0.05, 0.25]	[0.25, 0.85]
PubMed	8000	6700	3	train	[2.27, 4.35]	[0.02, 0.15]	[0.33, 0.91]
Products	5000	4400	5	train	[1.13, 3.15]	[0.00, 0.05]	[0.19, 0.76]
WikiCS	5000	4800	3	train/test	[12.0, 60.0]	[0.30, 0.65]	[0.45, 0.85]

Table 12: Key distribution-aligned generation settings. Clips are base dataset-level ranges; some split-specific runs narrow degree/clustering clips and enable connectivity repair.

Algorithm 2 Boundary-Aware TAG Augmentation

Require: Training TAG G_{tr} , SFT generator $\mathcal{M}$, augmentation budget B
Ensure: Augmented training graph $\tilde{G}_{tr}$
1: Train an oracle GNN and compute top-1/top-2 margins
2: **for** each class ℓ **do**
3: Select correctly classified low-margin boundary anchors
4: Compute class and boundary-anchor ego profiles
5: Use the boundary profile, fall back to the class profile if anchors are scarce, or blend when variance is high
6: Assign a class quota using inverse-frequency smoothing
7: **end for**
8: **for** each class ℓ until its quota is met or attempts are exhausted **do**
9: Use SFT generator $\mathcal{M}$ to autoregressively decode packets from the class-profile control prefix
10: Parse the packet and compute validity, target-label ratio, text quality, and homophily
11: Reject packets that are too small, lack target-label nodes, contain too many unknown labels or empty texts, miss the target-label ratio, or leave the homophily band
12: Keep accepted packets and update the class quota
13: **end for**
14: Choose one generated target-label anchor per packet
15: Attach it to the nearest real boundary anchor(s) in BERT embedding space, with an optional reuse penalty
16: **return** the augmented training graph formed by the real training graph plus accepted packet nodes, internal packet edges, and anchor edges

Dataset	B	β	Boundary frac.	Class ratio	Dom. prop.	Hom. band	Min target ratio
Cora	250	1.5	0.25	[0.03, 0.35]	0.68	0.25	0.30
CiteSeer	450	0.5	0.20	[0.10, 0.20]	0.74	0.32	0.20
PubMed	2000	0.8	0.25	[0.25, 0.42]	0.70	0.25	0.25
Products	1200	0.3	0.20	[0.025, 0.055]	0.82	0.60	0.03
WikiCS	1200	0.9	0.25	[0.02, 0.20]	0.62	0.30	0.15

Table 13: Key TAG-AUG generation settings; B counts target-label quota.

budgets, quota exponents, class-ratio caps, homophily bands, target-label ratio thresholds, and anchor settings are set in the released configuration because they depend strongly on the class imbalance and local density of each graph.