CONDITIONING-AWARE ADAPTIVE NOISING FOR FACTUAL AND EDIT-SENSITIVE GRAPH-TO-SEQUENCE GENERATION

Anonymous authors

Paper under double-blind review

ABSTRACT

Graph-to-sequence (G2S) generation is commonly evaluated by similarity to reference text, but high reference similarity does not imply reliable generation, which additionally requires *factual grounding* in the input graph and *edit sensitivity* to controlled graph edits. Diffusion language models are well suited to these properties because they iteratively revise the output, while the noise schedule controls how much target information is retained during denoising. However, adaptive noising is typically guided by denoising-side signals rather than by support from the conditioning graph. Across WikiOFGraph, GenWiki, and TekGEN, we find that denoising difficulty is only weakly and inconsistently associated with token-level graph support. We therefore introduce `DLM4G`, which separates adaptive noising into two roles: a *global adaptive schedule* uses token-wise denoising difficulty to identify positions that benefit from retaining more signal, while *dynamic conditioning modulation* uses decoder-to-graph cross-attention and conditioning sensitivity—the change in prediction when graph conditioning is removed, to control how strongly that schedule is applied for the current graph during both training and inference. We also introduce *Factual Grounding* (FGT) and *Edit Sensitivity Rate* (ESR) to measure the two reliability properties directly, and validate both against human and semantic assessments. The 63M-parameter `DLM4G` improves BLEU, chrF++, and METEOR over diffusion baselines on all three benchmarks while remaining competitive with 770M and 7-8B models. Within `DLM4G`, conditioning-aware modulation over difficulty-only adaptation yields absolute gains of 4-7 points in FGT@0.5, 6-9 points in ESR@2, 5-7 points in relation-edit ESR, and approximately 3 BLEU points across the three benchmarks.

1 INTRODUCTION

Graph-to-sequence (G2S) generation maps structured graph information to an output sequence, supporting applications such as knowledge verbalization and graph-supported reasoning (Ribeiro et al., 2020; 2021; He et al., 2025). We characterize *reliable G2S generation* through three complementary properties: (1) *generation quality* requires fluent outputs that match reference sequences, and is commonly measured using lexical and semantic similarity metrics such as BLEU, chrF++, METEOR, and BERTScore. However, high similarity to a reference does not necessarily imply factual correctness (Zha et al., 2023). For example, given the input fact ⟨*Marie Curie, birthplace, Warsaw, Kingdom of Poland*⟩, changing only "Warsaw, Kingdom of Poland" to "Paris, France" in an otherwise identical output still gives approximately 0.7 BLEU-4 and 0.8 chrF++ similarity, even though the central fact is wrong. This motivates (2) *factual grounding*: generated information should be supported by the conditioning graph rather than only resemble a reference (Zhang et al., 2023; Zha et al., 2023). A fixed graph-output comparison, however, does not tell us whether the model responds correctly when the graph changes. If the birthplace in the graph is edited from *Warsaw* to *Paris*, a model that ignores the edit and keeps the original output can still receive high overlap with the original reference. This motivates (3) *edit sensitivity*: a local change in the graph should produce the corresponding change in the output while preserving unrelated content.

Diffusion language models provide a natural framework for studying these properties. Unlike standard left-to-right autoregressive decoding, diffusion models iteratively refine output tokens while

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

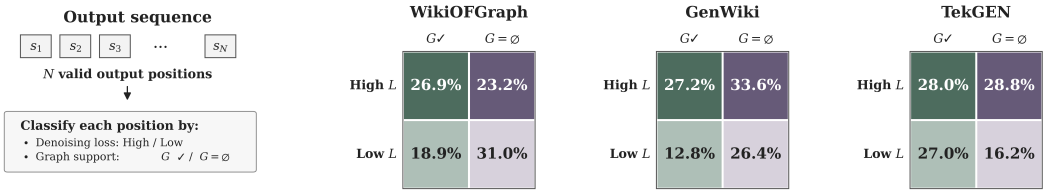


Figure 1: **Denoising difficulty does not reliably identify graph-supported positions.** Results are computed over the full training split of each dataset.

conditioning on the source (Li et al., 2022; Gong et al., 2023a; Sahoo et al., 2024a; Nie et al., 2025). The forward diffusion process also exposes the noise schedule as an explicit control over how much information is preserved during denoising (Ho et al., 2020; Li et al., 2022; Gong et al., 2023a). Recent methods adapt corruption or denoising across token positions using signals such as denoising difficulty, information allocation, output context, confidence, or inter-token dependency (Yuan et al., 2024; Raya et al., 2026; Ye et al., 2025; Cong et al., 2026; Zhou et al., 2026). These signals capture different aspects of denoising, but do not directly capture how strongly each token is supported by the conditioning graph.

For graph-conditioned G2S, we use denoising difficulty as the starting signal for adaptive noising and find that it is not sufficient on its own. A difficult token can be graph-supported or unsupported, while a graph-supported token can also be easy to denoise. Figure 1 shows that the association between denoising difficulty and graph support is weak and dataset-dependent: high-loss positions are more often graph-supported on WikiOFGraph, more often unsupported on GenWiki, and nearly balanced on TekGEN, where the stronger separation appears among low-loss positions. Thus, difficulty identifies positions that are hard to recover, but does not reliably identify which positions are supported by the conditioning graph.

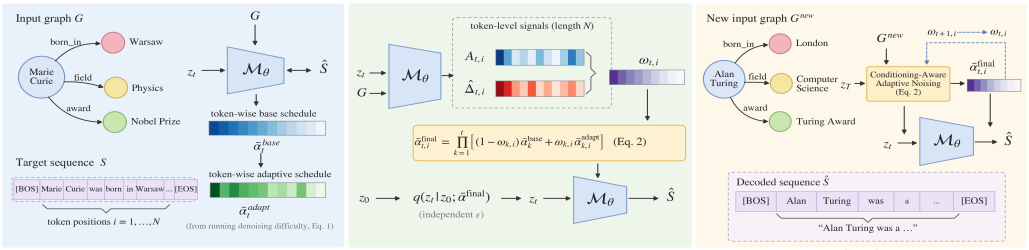
These observations suggest that denoising difficulty and graph conditioning should play different roles in adaptive noising. We therefore introduce **D**iffusion **L**anguage **M**odel **for G**raphs (`DLM4G`), which combines two components. **(1) Global adaptive schedule.** Token-wise denoising difficulty accumulated during training defines an adaptive noise schedule that retains more signal at positions that are generally harder to recover. **(2) Dynamic conditioning modulation.** For the current graph and noisy sequence, decoder-to-graph cross-attention and *conditioning sensitivity*, defined as the change in the denoising prediction when graph conditioning is removed, determine on the fly how strongly the global adaptive schedule is applied at each position. The global schedule captures general denoising difficulty, while the conditioning signals dynamically guide its application during both training and inference.

Contributions. Our contributions are organized around three questions. **(Q1) Can graph-conditioning signals address the limitation of denoising difficulty?** We show that denoising difficulty is only weakly and inconsistently associated with token-level graph support across three datasets, while graph attention and conditioning sensitivity provide complementary signals for conditioning-aware adaptation. **(Q2) How should factual grounding and edit responsiveness be measured?** We introduce *Factual Grounding* (FGT) for static graph grounding and *Edit Sensitivity Rate* (ESR) for controlled graph edits, and validate both against independent semantic and human assessments. **(Q3) Does conditioning-aware adaptive noising improve reliable generation?** Across WikiOFGraph, GenWiki, and TekGEN, conditioning-aware modulation over difficulty-only adaptation yields absolute gains of 4-7 points in FGT@0.5, 6-9 points in ESR@2, 5-7 points in relation-edit ESR, and approximately 3 BLEU points.

2 Background and Related Work

2.1 Related Work

Positioning. G2T generation has progressed from graph-specific models to pretrained Transformer-based models over serialized graphs (Ribeiro et al., 2020; 2021), while factuality methods such as FactSpotter and PlanGTG explicitly assess or improve grounding (Zhang et al., 2023; He et al., 2025). Diffusion-LM and DiffuSeq established continuous text and conditional Seq2Seq diffusion, while



(A) Input Representation (B) Conditioning- Aware Adaptive Noising (C) Inference

Figure 2: DLM4G framework: (A) the input graph conditions the sequence denoiser; (B) token-wise denoising difficulty defines a global adaptive schedule, which is dynamically modulated using graph attention and conditioning sensitivity; (C) the trained DLM4G generates the output sequence conditioned on the input graph.

MDLM and LLaDA established masked diffusion (Li et al., 2022; Gong et al., 2023a; Sahoo et al., 2024a; Nie et al., 2025). Existing adaptive diffusion methods use signals such as denoising loss, context, confidence, or inter-token dependency (Yuan et al., 2024; Ye et al., 2025; Cong et al., 2026; Zhou et al., 2026). Although these methods may operate in conditional generation settings, their adaptation criteria do not directly use the external conditioning input (**Q1**). Existing G2T factuality methods primarily evaluate fixed graph-sequence pairs and do not test responses to controlled graph edits (Zhang et al., 2023; Zha et al., 2023; He et al., 2025) (**Q2**). DLM4G instead combines a global difficulty-based adaptive schedule with dynamic graph-conditioned modulation during both training and inference, and evaluates its effect on factual grounding and edit sensitivity (**Q3**). A detailed method-by-method comparison, including the information domain each adaptation signal is computed from, is given in Table 5.

2.2 Preliminaries: Conditional Diffusion Models

Denoising diffusion models learn a data distribution by gradually corrupting clean representations and learning to reverse this process (Ho et al., 2020). Given a clean representation $\mathbf{z}_0 \in \mathbb{R}^{N \times d}$ and conditioning input $\mathbf{c}$, the forward process samples $\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}$, with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, so that $\bar{\alpha}_t$ controls the retained signal, and hence the signal-to-noise ratio, at timestep t. A conditional denoiser $\mathcal{M}_\theta(\mathbf{z}_t, t, \mathbf{c})$ learns to recover the clean representation; for text diffusion, $\mathbf{z}_0$ is a sequence of token embeddings that are mapped back to discrete tokens after denoising (Li et al., 2022; Gong et al., 2023a). Standard text-diffusion formulations apply the same schedule $\bar{\alpha}_t$ across output positions. In Section 3 we instead allow $\bar{\alpha}_{t,i}$ to vary across token positions, using denoising difficulty together with conditioning information from the graph; the training objective and its full decomposition are given in Section 3.3 and Appendix B.2.

3 The DLM4G Methodology

3.1 Problem Setting: Graph-to-Sequence Generation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ denote an input graph, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is the node set, $\mathcal{E}$ is the edge set, and $\mathbf{X} = \{x_i\}_{i=1}^{n}$ with $x_i \in \mathbb{R}^d$ are node features. Our goal is *graph-to-sequence* (G2S) generation: learning a conditional distribution $p_\theta(\mathbf{S} \mid \mathcal{G})$ over output sequences $\mathbf{S} = (s_1, \ldots, s_N)$, where $s_i \in \mathcal{W}$ and $\mathcal{W}$ is the vocabulary. In practice, variable-length targets are padded/truncated to a maximum length N. We further define $\mathcal{M}_\theta$ as a parameterized conditional denoiser used to model $p_\theta(\mathbf{S} \mid \mathcal{G})$ from paired graph-sequence data.

Knowledge graphs as a special case. A knowledge graph (KG) is a relational graph $\mathcal{G}_{\text{KG}} = (\mathcal{V}, \mathcal{R}, \mathcal{T})$ defined by a set of entities $\mathcal{V}$, relation types $\mathcal{R}$, and directed triples $\mathcal{T} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$. We denote triples as $(h, r, t) \in \mathcal{T}$. In this work, we instantiate our G2S framework on KGs via a standard serialization into a token sequence $\mathbf{G} = \textsc{Serialize}(\mathcal{T})$ (Section 4.1).

3.2 DLM4G: CONDITIONING-AWARE ADAPTIVE NOISING

We embed the target sequence as $\mathbf{z}_0 = g_\Phi(\mathbf{S}) \in \mathbb{R}^{N \times d_z}$. For position i, the forward process is $\mathbf{z}_{t,i} = \sqrt{\bar{\alpha}_{t,i}}\mathbf{z}_{0,i} + \sqrt{1-\bar{\alpha}_{t,i}}\boldsymbol{\epsilon}_i$, where $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The denoiser receives $\bar{\boldsymbol{\alpha}}_t = (\bar{\alpha}_{t,1}, \ldots, \bar{\alpha}_{t,N})$, so that the noise level of every position is explicit. The forward process remains factorized across output positions under the token-wise schedule. Let α_t^{base} denote the baseline per-step coefficient, with $\bar{\alpha}_t^{\text{base}} = \prod_{k=1}^{t} \alpha_k^{\text{base}}$. DLM4G combines a global adaptive (token-wise denoising difficulty) schedule with two conditioning signals (Graph attention & Conditioning sensitivity).

1. Global adaptive schedule. At each training step, we measure the denoising loss at output position i as $\ell_i = \|\mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t^{\text{final}}, \mathbf{G})^{(i)} - \mathbf{z}_{0,i}\|_2^2$. We maintain a running average of this loss over training steps and non-[PAD] occurrences of position i, denoted by $\bar{\ell}_i$. Higher loss indicates that the token at this position is harder to recover and should therefore retain more signal. We map this difficulty to a cumulative signal-retention schedule:

$$\bar{\alpha}_{t,i}^{\text{adapt}} = \bar{\alpha}_t^{\text{base}} + \frac{\bar{\ell}_i - \ell_{\min}}{\ell_{\max} - \ell_{\min} + \varepsilon} \left(\bar{\alpha}_{\max(t-\kappa,0)}^{\text{base}} - \bar{\alpha}_t^{\text{base}} \right). \tag{1}$$

Here $\ell_{\min}$ and $\ell_{\max}$ are computed across valid output positions, and κ controls the maximum schedule shift (default $\kappa = 100$), with $\bar{\alpha}_0^{\text{base}} = 1$. Since the cumulative signal-retention schedule decreases with diffusion time, a larger denoising loss produces a larger $\bar{\alpha}_{t,i}^{\text{adapt}}$ and hence more conservative noising. Whenever a per-step coefficient is required, it is recovered as $\alpha_{t,i}^{\text{adapt}} = \frac{\bar{\alpha}_{t,i}^{\text{adapt}}}{\bar{\alpha}_{t-1,i}^{\text{adapt}}}$, $\bar{\alpha}_{0,i}^{\text{adapt}} = 1$. The global adaptive schedule is updated periodically during training and held fixed between updates.

2. Graph attention.

The global adaptive schedule captures denoising difficulty but does not use information from the current input graph. We therefore use decoder-to-graph cross-attention as the first conditioning signal. Let $a_{t,i,j}^{(h)}$ denote cross-attention from output position i to graph-token position j under head h. Since each cross-attention row is normalized over the graph tokens, we use its peak concentration rather than its total mass: $A_{t,i} = \frac{1}{H}\sum_{h=1}^{H} \max_{j \in \mathcal{I}_\mathbf{G}} a_{t,i,j}^{(h)}, \qquad A_{t,i} \in [0,1]$, where H is the number of attention heads and $\mathcal{I}_\mathbf{G}$ indexes the graph tokens. A larger $A_{t,i}$ indicates stronger concentration of decoder cross-attention on the graph at position i. Unlike the global schedule, $A_{t,i}$ is computed on the fly from the current graph and noisy sequence. However, attention only indicates where the model attends to the graph; it does not measure how much the graph changes the denoising prediction. Our second conditioning signal, *conditioning sensitivity*, directly measures this effect by comparing predictions with and without graph conditioning.

3. Conditioning sensitivity. Graph attention allows the model to use information from $\mathbf{G}$ when updating the representation at each output position. However, the attention score alone does not measure how much this graph information changes the denoising prediction. We therefore compare the prediction with and without graph conditioning, following the conditional-unconditional comparison used in classifier-free guidance (Ho & Salimans, 2021). Both predictions are obtained from the same model $\mathcal{M}_\theta$ and the same noisy state $\mathbf{z}_t$. During training, graph conditioning is randomly dropped, so the same model is trained both with and without $\mathbf{G}$. We define $\Delta_{t,i} = \frac{\left\|\mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t, \mathbf{G})^{(i)} - \mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t, \emptyset)^{(i)}\right\|_2}{\left\|\mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t, \mathbf{G})^{(i)}\right\|_2 + \varepsilon}, \qquad \widehat{\Delta}_{t,i} = \frac{\Delta_{t,i}}{1+\Delta_{t,i}}$. A larger $\widehat{\Delta}_{t,i}$ means that removing the graph changes the prediction more strongly at position i. Unlike classifier-free guidance, this difference is not added to the denoising prediction; it is used only to control the noise schedule. We combine graph attention and conditioning sensitivity as $\omega_{t,i} = \text{StopGrad}\left[A_{t,i} + (1 - A_{t,i})\widehat{\Delta}_{t,i}\right]$, and use this weight to interpolate between the baseline and global adaptive per-step coefficients:

$$\alpha_{t,i}^{\text{final}} = (1-\omega_{t,i})\alpha_t^{\text{base}} + \omega_{t,i}\alpha_{t,i}^{\text{adapt}}, \qquad \bar{\alpha}_{t,i}^{\text{final}} = \prod_{k=1}^{t}\left[(1-\omega_{t,i})\alpha_k^{\text{base}} + \omega_{t,i}\alpha_{k,i}^{\text{adapt}}\right]. \tag{2}$$

Thus, denoising loss defines the global adaptive coefficient $\alpha_{t,i}^{\text{adapt}}$, while $A_{t,i}$ and $\widehat{\Delta}_{t,i}$ determine on the fly how strongly it is applied to the current input. The conditioning weight is treated as fixed with respect to model gradients when constructing the noise schedule.

The global schedule and conditioning signals serve complementary roles. The global schedule identifies positions that are generally harder to denoise, while the conditioning weight controls how strongly this adaptation is applied for the current graph. Positions that are both difficult and strongly graph-conditioned therefore retain more signal, making graph-supported content easier to recover during denoising. This principle is not specific to continuous diffusion and could similarly be implemented through position-specific masking or reveal probabilities in masked diffusion (Appendix A.3).

Schedule update. The running token-wise losses are accumulated throughout training. After the initial warm-up, training proceeds in blocks of K_{up} steps, with the global adaptive coefficients $\bar{\alpha}_{t,i}^{\mathrm{adapt}}$ held fixed within each block. For each training input, $\mathbf{z}_t$ is generated using this blockwise-fixed adaptive schedule. From this state, $A_{t,i}$ and $\widehat{\Delta}_{t,i}$ are computed and combined into $\omega_{t,i}$, which defines the batch-specific $\bar{\boldsymbol{\alpha}}_t^{\mathrm{final}}$ through Eq. 2. The training noisy state is then resampled using $\bar{\boldsymbol{\alpha}}_t^{\mathrm{final}}$ before computing the denoising loss. At the end of each block, the accumulated loss statistics update $\bar{\alpha}_{t,i}^{\mathrm{adapt}}$ for the next block. After training, $\bar{\alpha}^{\mathrm{adapt}}$ is retained, while the conditioning signals are recomputed dynamically during inference.

Algorithm 1: `DLM4G` schedule update

Require: $\mathcal{D}$, $\bar{\boldsymbol{\alpha}}^{\mathrm{base}}$, $\bar{\boldsymbol{\alpha}}^{\mathrm{adapt}}$, $\bar{\boldsymbol{\ell}}$, K_{up}, $\mathcal{M}_\theta$

for each block of K_{up} training steps **do**
  for each training input **do**
    Compute $A_{t,i}$ and $\widehat{\Delta}_{t,i}$
    Compute $\omega_{t,i} = \mathrm{StopGrad}\left[A_{t,i} + (1 - A_{t,i})\widehat{\Delta}_{t,i}\right]$
    Construct $\bar{\alpha}_{t,i}^{\mathrm{final}}$ using Eq. 2
    Train $\mathcal{M}_\theta$ using the final schedule; update $\bar{\boldsymbol{\ell}}$
  end for
  Update $\bar{\alpha}_{t,i}^{\mathrm{adapt}}$ using Eq. 1
end for
return $\bar{\boldsymbol{\alpha}}^{\mathrm{adapt}}$

Complete training-time state construction and re-noising are provided in Appendix C.3, with the full training and inference algorithms in Appendix C.5.

3.3 Training Objective

Our training objective is motivated by the conditional VLB (App B.2). Since the adaptive schedule is constructed using model-derived conditioning signals, we treat the resulting schedule as fixed with respect to gradients and optimize a $\mathbf{z}_0$-prediction objective, rather than treating the adaptive objective as an exact bound on the conditional log-likelihood. Specifically, $\mathcal{M}_\theta$ predicts the clean latent $\mathbf{z}_0$ from $\mathbf{z}_t$. Let $\mathcal{I}(\mathbf{S})$ denote the non-`[PAD]` positions. To map the continuous latent back to discrete tokens, we use a learned rounding distribution $\tilde{p}_\Phi(\mathbf{S} \mid \mathbf{z}_0)$. The resulting end-to-end objective is:

$$\mathcal{L}_{\text{e2e-simple}}(\mathbf{S}) = \mathbb{E}_{q_{\bar{\boldsymbol{\alpha}}^{\mathrm{final}}}(\mathbf{z}_{1:T}|\mathbf{z}_0)}\Bigg[\underbrace{\sum_{t=2}^{T}\sum_{i\in\mathcal{I}(\mathbf{S})}\left\|\mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t^{\mathrm{final}}, \mathbf{G})^{(i)} - \mathbf{z}_{0,i}\right\|_2^2}_{\text{Denoising}} + \underbrace{\sum_{i\in\mathcal{I}(\mathbf{S})}\left\|\mathbf{z}_{0,i} - \mathcal{M}_\theta(\mathbf{z}_1, \bar{\boldsymbol{\alpha}}_1^{\mathrm{final}}, \mathbf{G})^{(i)}\right\|_2^2}_{\text{Consistency}} - \underbrace{\sum_{i\in\mathcal{I}(\mathbf{S})}\log\tilde{p}_\Phi(s_i \mid \mathbf{z}_{0,i})}_{\text{Rounding}}\Bigg]. \tag{3}$$

Here $g_\Phi(\mathbf{S}) \in \mathbb{R}^{N\times d_z}$ denotes the token embedding lookup, i.e., $\mathbf{z}_0 = g_\Phi(\mathbf{S})$ with $(\mathbf{z}_0)_i = \mathbf{e}_{s_i}$. The rounding distribution factorizes over valid output positions, $\tilde{p}_\Phi(\mathbf{S} \mid \mathbf{z}_0) = \prod_{i\in\mathcal{I}(\mathbf{S})}\tilde{p}_\Phi(s_i \mid \mathbf{z}_{0,i})$, where $\tilde{p}_\Phi(s_i \mid \mathbf{z}_{0,i}) = \mathrm{Softmax}(\mathbf{W}\mathbf{z}_{0,i})_{s_i}$ for $\mathbf{W} \in \mathbb{R}^{|\mathcal{W}|\times d_z}$. During training, this term is evaluated using the gold s_i; at inference, we decode from $\tilde{p}_\Phi(\cdot \mid \hat{\mathbf{z}}_0)$. In practice, we estimate the denoising term using a uniformly sampled timestep $t \in \{2, \ldots, T\}$; implementation details are in Appendix B.4.

Inference. After training, the learned cumulative global adaptive schedule $\bar{\boldsymbol{\alpha}}^{\mathrm{adapt}}$ and its derived per-step coefficients $\boldsymbol{\alpha}^{\mathrm{adapt}}$ are retained. We initialize $\omega_{T+1,i} = 1$, so that the first reverse step uses the learned global adaptive schedule. At reverse step t, we use the conditioning weight $\omega_{t+1,i}$ carried from the preceding reverse step. For the current reverse transition, this weight is held fixed across the required schedule prefix, giving $\bar{\alpha}_{r,i}^{\mathrm{final}} = \prod_{k=1}^{r}\left[(1-\omega_{t+1,i})\alpha_k^{\mathrm{base}} + \omega_{t+1,i}\alpha_{k,i}^{\mathrm{adapt}}\right]$, $r \in \{t-1, t\}$, and $\alpha_{t,i}^{\mathrm{final}} = \bar{\alpha}_{t,i}^{\mathrm{final}}/\bar{\alpha}_{t-1,i}^{\mathrm{final}}$. The conditional model then predicts $\hat{\mathbf{z}}_0 = \mathcal{M}_\theta\left(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t^{\mathrm{final}}, \mathbf{G}\right)$. From the same reverse step, $A_{t,i}$ and $\widehat{\Delta}_{t,i}$ are recomputed from $\mathbf{z}_t$ and $\mathbf{G}$ to obtain $\omega_{t,i}$, which is carried to the next reverse step. The model samples $\mathbf{z}_{t-1,i} = \frac{\sqrt{\bar{\alpha}_{t-1,i}^{\mathrm{final}}}\left(1-\alpha_{t,i}^{\mathrm{final}}\right)}{1-\bar{\alpha}_{t,i}^{\mathrm{final}}}\hat{\mathbf{z}}_{0,i} + \frac{\sqrt{\alpha_{t,i}^{\mathrm{final}}}\left(1-\bar{\alpha}_{t-1,i}^{\mathrm{final}}\right)}{1-\bar{\alpha}_{t,i}^{\mathrm{final}}}\mathbf{z}_{t,i} + \sigma_{t,i}\boldsymbol{\epsilon}_i$, where $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\sigma_{t,i}^2 = \frac{1-\bar{\alpha}_{t-1,i}^{\mathrm{final}}}{1-\bar{\alpha}_{t,i}^{\mathrm{final}}}\left(1-\alpha_{t,i}^{\mathrm{final}}\right)$. Thus, $\bar{\boldsymbol{\alpha}}^{\mathrm{adapt}}$ and its derived $\boldsymbol{\alpha}^{\mathrm{adapt}}$ remain fixed after training, while the conditioning weight is updated dynamically as $\omega_{t+1,i} \to \omega_{t,i}$ throughout generation. Further details are provided in Appendix C.4.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Model architecture. DLM4G is an encoder-decoder Transformer conditioned on a serialized KG input. We evaluate two variants differing only in decoder depth: DLM4G-1.o with 6 encoder/6 decoder layers (≈50M parameters) and DLM4G-2.o with 6 encoder/9 decoder layers (≈63M parameters). Both use the `bert-base-uncased` vocabulary (Devlin et al., 2019), with `[HEAD]`, `[REL]`, `[TAIL]`, and `[SEP]` as learned special tokens.

Graph representation. We serialize each relational triple $(h_i, r_{ij}, t_j) \in \mathbf{G}$ as $\langle$`[HEAD]` h_i `[REL]` r_{ij} `[TAIL]` $t_j\rangle$ and concatenate triples using `[SEP]`, preserving the triple order of the released datasets. Full examples are in Appendix D.2.

Datasets and baselines. We evaluate on WikiOFGraph (Kim et al., 2024), GenWiki (Jin et al., 2020), and TekGEN (Agarwal et al., 2021). Baselines cover four families: (i) fine-tuned PLMs, including GPT-2 (Small/Base) (Mager et al., 2020) and T5 (Small/Large) (Ribeiro et al., 2021); (ii) zero-shot LLMs, including GPT-o4-mini, LLaMA-3-8B, Qwen2.5-7B, and DeepSeek-7B; (iii) task-specific G2S systems, including ReGen Dognin et al. (2021), Rule-Based, and Direct-Transfer/Noisy-Supervised Guo et al. (2020); and (iv) diffusion baselines, including DiffuSeq (Gong et al., 2023a), FlowSeq (Hu et al., 2024), and SeqDiffuSeq (Yuan et al., 2024). Dataset statistics, splits, and baseline protocols are in Appendices D.1 and D.3.

Training protocol. We train with $T = 2000$ continuous diffusion timesteps and introduce adaptive noising after a 20K-step warm-up under the base schedule; the cumulative global adaptive schedule $\bar{\alpha}_{t,i}^{\text{adapt}}$ is then updated every 20K steps and held fixed between updates. Details are in App D.5.

Evaluation. We report BLEU (**B**) (Papineni et al., 2002), chrF++ (**CrF++**) (Popović, 2015), and METEOR (**M**) (Banerjee & Lavie, 2005) as conventional generation metrics, and evaluate graph-conditioned factual behavior separately using Factual Grounding (FGT) and Edit Sensitivity Rate (ESR), defined in Section 4.3. MAUVE (**MVE**) and BERTScore-F1 (**B-F1**) are reported in Appendix F.1, and TekGEN factual and edit-sensitive results in Appendix E.5. Results are averaged over three random seeds (App. F.4).

4.2 Q1: CAN CONDITIONING SIGNALS ADDRESS THE LIMITATION OF DENOISING DIFFICULTY?

The global adaptive schedule uses denoising difficulty to determine which positions should retain more signal. However, Figure 1 shows that difficulty alone does not reliably identify positions supported by the input graph. This motivates the conditioning-aware part of DLM4G: the global schedule should be modulated using signals that directly reflect how the current graph influences each output position. We therefore examine the two signals used by our scheduler, graph attention $A_{t,i}$ and conditioning sensitivity $\Delta_{t,i}$. Figure 3(a) shows that decoder-to-graph cross-attention increasingly distinguishes graph-supported from unsupported positions as denoising progresses. AUROC rises from 0.61 to 0.83 on WikiOFGraph, 0.55 to 0.79 on GenWiki, and 0.53 to 0.73 on TekGEN. Thus, $A_{t,i}$ indicates where the denoiser attends more strongly to graph evidence. Figure 3(b) shows that removing graph conditioning changes the prediction more strongly at graph-supported than unsupported positions. As denoising progresses, graph information is progressively integrated into the denoised state; consequently, $\Delta_{t,i}$ decreases over diffusion timesteps because explicitly conditioning on the graph has a smaller additional effect. However, $\Delta_{t,i}$ remains consistently higher for graph-supported positions, indicating that their predictions are more strongly influenced by the input graph. Together with the mismatch identified in Figure 1, these results motivate using $A_{t,i}$ and $\Delta_{t,i}$ to dynamically modulate the global adaptive schedule. Their contribution to generation quality is tested directly through ablations in Q3. Before that, Q2 defines metrics for measuring factual grounding and responsiveness to graph edits.

4.3 Q2: HOW SHOULD FACTUAL AND EDIT-SENSITIVE BEHAVIOR BE MEASURED?

Standard metrics such as BLEU, chrF++, and METEOR primarily measure reference agreement and do not directly quantify grounding to the input graph or responsiveness to controlled graph edits.

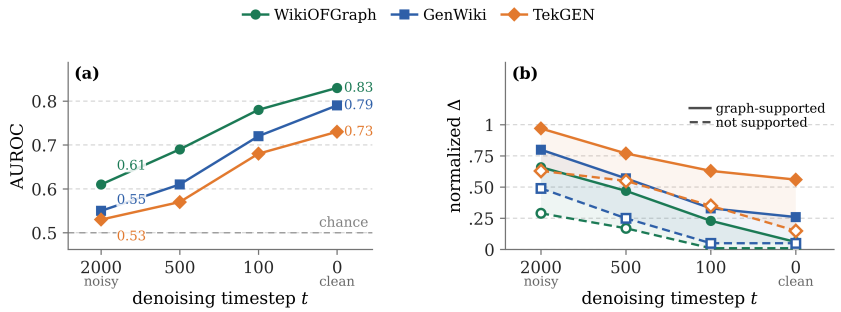


Figure 3: **Graph-conditioning signals distinguish graph-supported positions.** **(a)** Decoder-to-graph cross-attention becomes increasingly discriminative as denoising progresses. **(b)** Conditioning sensitivity Δ is consistently larger for graph-supported than unsupported positions.

We therefore evaluate these two properties separately using **Factual Grounding (FGT)** and **Edit Sensitivity Rate (ESR)**.

Metric definitions. For an input graph $\mathbf{G}$ and generated sequence $\mathbf{S}$, let $\mathcal{U}_{\mathbf{G}}$ denote the units supported by the graph and $\mathcal{U}_{\mathbf{S}}$ the units realized in the generated output. For FGT, these units are entity mentions; for ESR, they also include edited entities, attribute values, and relation labels. We define unsupported generated units as $\mathcal{H}_{\mathbf{S}} = \mathcal{U}_{\mathbf{S}} \setminus \mathcal{U}_{\mathbf{G}}$. FGT measures how well graph-supported units are realized while penalizing unsupported generated units through λ. We report $\lambda \in \{0, 0.5, 1.0\}$ and use $\lambda = 0.5$ by default. For ESR, $(\mathbf{G}, \mathbf{S})$ and $(\mathbf{G}', \mathbf{S}')$ denote the original and edited graph-output pairs. ESR measures whether the changes in the generated output match the controlled changes made to the input graph. We also report entity recall and F1. Entity, attribute, and relation units are extracted and canonicalized using a fixed Qwen2.5-7B-Instruct post-processing pipeline applied identically to all systems; full implementation details and the prompt specification are provided in App E.1.

$$\mathrm{F1} = \frac{2|\mathcal{U}_{\mathbf{G}} \cap \mathcal{U}_{\mathbf{S}}|}{|\mathcal{U}_{\mathbf{G}}| + |\mathcal{U}_{\mathbf{S}}|}, \qquad \mathcal{F}_{\mathrm{GT}}(\mathbf{G}, \mathbf{S}) = \mathrm{F1}\left(1 - \lambda \frac{|\mathcal{H}_{\mathbf{S}}|}{\max(|\mathcal{U}_{\mathbf{S}}|, 1)}\right) \tag{4}$$

$$\Delta\mathcal{G} = (\mathcal{U}_{\mathbf{G}'} \setminus \mathcal{U}_{\mathbf{G}}) \cup (\mathcal{U}_{\mathbf{G}} \setminus \mathcal{U}_{\mathbf{G}'}), \quad \Delta\mathcal{T} = (\mathcal{U}_{\mathbf{S}'} \setminus \mathcal{U}_{\mathbf{S}}) \cup (\mathcal{U}_{\mathbf{S}} \setminus \mathcal{U}_{\mathbf{S}'}), \quad \mathcal{E}_{\mathrm{SR}}(\mathbf{G}, \mathbf{S}; \mathbf{G}', \mathbf{S}') = \frac{|\Delta\mathcal{G} \cap \Delta\mathcal{T}|}{\max(|\Delta\mathcal{G}|, |\Delta\mathcal{T}|, 1)} \tag{5}$$

Controlled perturbations. We evaluate ESR under value and relation edits. Value edits replace entities, years, locations, or attribute values while preserving the surrounding relation; the main evaluation uses $k \in \{1, 2\}$ edits per graph. Relation edits modify one predicate or edge type while retaining the participating entities. Table 1 reports results on WikiOFGraph and GenWiki; $k \in \{3, 4\}$, full perturbation breakdowns, and TekGEN results are provided in Appendices E.3, E.4, and E.5.

Independent metric validation. We test whether FGT and ESR agree with independent assessments; full validation protocols are provided in Appendix E.2.

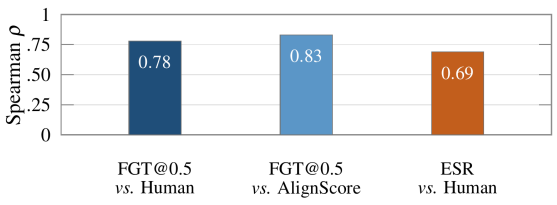


Figure 4: **Independent validation of FGT and ESR.** Spearman correlation (ρ) with independent assessments.

FGT@0.5 correlates strongly with human factual-grounding judgments ($\rho = 0.78$) and AlignScore ($\rho = 0.83$), providing independent support for its grounding assessment. ESR likewise agrees with human edit-sensitivity judgments ($\rho = 0.69$), indicating that higher ESR corresponds to outputs that more accurately reflect controlled changes to the input graph.

Results. Table 1 shows that grounding and edit responsiveness remain imperfect across existing systems. On WikiOFGraph, competitive baselines still produce 1.5-3 unsupported units per sequence,

Table 1: **Factual grounding and edit sensitivity.** Selected metrics report mean$_{\pm \text{std}}$ over three seeds. Lower $|\mathcal{H}_S|$ is better. Extended results are in Appendix E.

	Factual Grounding						Edit Sensitivity Rate		
Model	**Rec.**	**F1**	$\lvert\mathcal{H}_S\rvert \downarrow$	**FGT@0.5**	**FGT@1.0**	**AlignScore**	$k=1$	$k=2$	**Relation**
# WikiOFGraph									
GPT-2 (B)	0.60	0.60	2.95	$0.59_{\pm 0.004}$	0.58	$0.65_{\pm 0.005}$	0.64	$0.55_{\pm 0.006}$	$0.39_{\pm 0.003}$
T5 (S)	0.58	0.64	3.10	$0.56_{\pm 0.003}$	0.48	$0.62_{\pm 0.004}$	0.63	$0.45_{\pm 0.004}$	$0.37_{\pm 0.006}$
T5 (L)	0.81	0.81	1.54	$0.79_{\pm 0.002}$	0.77	$0.83_{\pm 0.004}$	0.66	$0.57_{\pm 0.001}$	$0.56_{\pm 0.003}$
DiffuSeq	0.78	0.83	2.00	$0.79_{\pm 0.005}$	0.75	$0.84_{\pm 0.002}$	0.66	$0.59_{\pm 0.004}$	$0.56_{\pm 0.003}$
SeqDiffuSeq	0.77	0.82	2.01	$0.76_{\pm 0.003}$	0.70	$0.81_{\pm 0.005}$	0.65	$0.58_{\pm 0.011}$	$0.56_{\pm 0.004}$
DLM4G-1.o	0.80	0.84	1.56	$0.80_{\pm 0.006}$	0.76	$0.81_{\pm 0.004}$	0.66	$0.58_{\pm 0.005}$	$0.56_{\pm 0.004}$
DLM4G-2.o	**0.82**	**0.86**	**1.08**	$\mathbf{0.83}_{\pm 0.003}$	**0.80**	$\mathbf{0.86}_{\pm 0.002}$	**0.68**	$\mathbf{0.60}_{\pm 0.005}$	$\mathbf{0.58}_{\pm 0.002}$
# GenWiki									
GPT-2 (B)	0.58	0.59	5.80	$0.48_{\pm 0.003}$	0.38	$0.57_{\pm 0.003}$	0.52	$0.50_{\pm 0.005}$	$0.40_{\pm 0.006}$
T5 (S)	0.60	0.63	4.73	$0.54_{\pm 0.004}$	0.44	$0.62_{\pm 0.003}$	0.50	$0.48_{\pm 0.006}$	$0.39_{\pm 0.004}$
T5 (L)	0.67	0.69	3.56	$0.61_{\pm 0.005}$	0.54	$0.67_{\pm 0.003}$	0.58	$0.54_{\pm 0.006}$	$0.46_{\pm 0.004}$
DiffuSeq	0.70	0.75	3.01	$0.68_{\pm 0.005}$	0.61	$0.73_{\pm 0.003}$	0.59	$0.53_{\pm 0.003}$	$0.45_{\pm 0.003}$
SeqDiffuSeq	0.69	0.73	3.02	$0.66_{\pm 0.007}$	0.59	$0.74_{\pm 0.006}$	0.57	$0.54_{\pm 0.004}$	$0.44_{\pm 0.006}$
DLM4G-1.o	0.72	0.74	3.04	$0.67_{\pm 0.007}$	0.60	$0.76_{\pm 0.005}$	0.57	$0.52_{\pm 0.006}$	$0.46_{\pm 0.005}$
DLM4G-2.o	**0.75**	**0.78**	**2.45**	$\mathbf{0.71}_{\pm 0.004}$	**0.65**	$\mathbf{0.77}_{\pm 0.003}$	**0.59**	$\mathbf{0.55}_{\pm 0.006}$	$\mathbf{0.48}_{\pm 0.004}$

while single-edit ESR remains around 0.63-0.66; GenWiki shows a similar pattern. DLM4G-2.o improves both grounding and edit sensitivity. On WikiOFGraph, it reaches FGT@0.5 of 0.83 with only 1.08 unsupported units, compared with 0.79/1.54 for T5-Large and 0.79/2.00 for DiffuSeq. It also achieves ESR of 0.68, 0.60, and 0.58 for $k = 1$, $k = 2$, and relation edits, respectively. Similar gains hold on GenWiki, where it reaches FGT@0.5 of 0.71 with 2.45 unsupported units and the strongest overall edit-sensitive performance. AlignScore (Zha et al., 2023) independently supports the grounding gains: DLM4G-2.o obtains 0.86/0.77 on WikiOFGraph/GenWiki, compared with 0.84/0.74 for the strongest non-DLM4G baselines. Extended perturbation and TekGEN results are provided in Apps. E.4 and E.5. In **Q3**, we next evaluate overall generation quality and isolate the contributions of the conditioning signals.

4.4 Q3: Does conditioning-aware adaptive noising improve generation quality?

Overall generation quality. Table 2 shows that increasing decoder depth from 6 to 9 layers (50M→63M parameters) improves all nine generation metrics. DLM4G-2.o achieves the best overall result on seven of the nine metrics and ranks second on the remaining two. On WikiOFGraph, T5-Large is slightly stronger on BLEU (0.658 vs. 0.654) and GPT-o4-mini on chrF++ (0.847 vs. 0.844), while DLM4G-2.o achieves the highest METEOR. It obtains the strongest results across all three metrics on both GenWiki and TekGEN. Despite using only 63M parameters, it therefore remains competitive with much larger fine-tuned and zero-shot models across all three benchmarks. Extended BERTScore-F1 and MAUVE results are in App. F.1, with task-specific G2S comparisons in App. F.2.

Table 2: **Main results against fine-tuned and zero-shot baselines.** Best results are shown in bold and second-best results are underlined.

	Fine-tuned PLMs				Zero-shot LLMs				DLM4G variants	
Metric	**GPT-2 (S)**	**GPT-2 (B)**	**T5 (S)**	**T5 (L)**	**LLaMA-3**	**Qwen2.5**	**DeepSeek**	**GPT-o4-mini**	**DLM4G-1.o**	**DLM4G-2.o**
#P	124M	355M	60M	770M	8B	7B	7B	–	50M	63M
# WikiOFGraph										
BLEU	0.166	0.285	0.385	**0.658**	0.622	0.625	0.633	0.648	0.619	0.654
chrF++	0.428	0.572	0.688	0.807	0.801	0.681	0.809	**0.847**	0.823	0.844
METEOR	0.487	0.490	0.471	0.736	0.781	0.743	0.752	0.783	0.688	**0.791**
# GenWiki										
BLEU	0.280	0.312	0.227	0.361	0.461	0.464	0.391	0.464	0.401	**0.469**
chrF++	0.465	0.470	0.495	0.567	0.709	0.697	0.688	0.734	0.663	**0.748**
METEOR	0.435	0.425	0.434	0.447	0.510	0.501	0.533	0.471	0.527	**0.574**
# TekGEN										
BLEU	0.226	0.228	0.189	0.199	0.176	0.182	0.124	0.121	0.247	**0.253**
chrF++	0.358	0.366	0.352	0.370	0.341	0.312	0.345	0.327	0.493	**0.522**
METEOR	0.208	0.211	0.203	0.211	0.251	0.234	0.256	0.277	0.375	**0.414**

Matched-budget comparison. To isolate the effect of training-data scale, we additionally train DLM4G-2.o, T5-Small, and SeqDiffuSeq under the same 100K-pair budget. DLM4G-2.o remains strongest across all three datasets (Table 3); matched-budget FGT, ESR, chrF++, and METEOR results are provided in App F.3.

Table 3: BLEU under matched budget.

Model	**WOF**	**GW**	**TG**
T5-Small	0.385	0.227	0.189
SeqDiffuSeq	0.405	0.247	0.185
DLM4G-2.o	**0.433**	**0.293**	**0.191**

WOF/GW/TG denote WikiOFGraph/GenWiki/TekGEN.

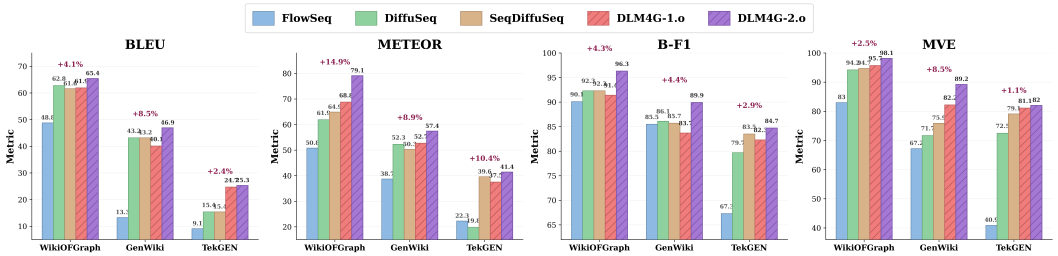


Figure 5: Performance of DLM4G against diffusion baselines.

Diffusion baselines. Finally, we compare DLM4G against prior diffusion-based text generation baselines (Fig. 5). Despite using only 63M parameters, DLM4G-2.o is nearly $1.5\times$ smaller than the strongest 91M-parameter diffusion baselines while consistently outperforming them across datasets and metrics. The gains are particularly strong on GenWiki and TekGEN, including +8.5% BLEU and +8.9% METEOR on GenWiki and +10.4% METEOR on TekGEN. WikiOFGraph likewise shows consistent improvements, including +4.1% BLEU and +14.9% METEOR.

Component attribution. Difficulty-aware adaptation improves over the base schedule across both conventional and factual/edit-sensitive metrics. Adding conditioning information yields further gains, with the sensitivity term Δ providing further improvement over difficulty-only adaptation.

Table 4: Ablation of conditioning-aware adaptive noising; values report mean$_{\pm\text{std}}$ over three random seeds.

Variant	BLEU	FGT@0.5	ESR@1	ESR@2	Relation
Base schedule	$0.60_{\pm 0.003}$	$0.77_{\pm 0.005}$	$0.64_{\pm 0.004}$	$0.53_{\pm 0.001}$	$0.52_{\pm 0.001}$
+ Difficulty (L)†	$0.62_{\pm 0.002}$	$0.79_{\pm 0.002}$	$0.65_{\pm 0.003}$	$0.54_{\pm 0.004}$	$0.53_{\pm 0.003}$
+ Difficulty + Attention ($L + A$)	$0.65_{\pm 0.003}$	$0.80_{\pm 0.001}$	$0.67_{\pm 0.005}$	$0.56_{\pm 0.002}$	$0.53_{\pm 0.003}$
+ Difficulty + Sensitivity ($L + \Delta$)	$0.65_{\pm 0.005}$	$0.81_{\pm 0.003}$	$0.68_{\pm 0.004}$	$0.58_{\pm 0.003}$	$0.56_{\pm 0.002}$
Full ($L + A + \Delta$)	$\mathbf{0.65}_{\pm 0.002}$	$\mathbf{0.83}_{\pm 0.003}$	$\mathbf{0.68}_{\pm 0.002}$	$\mathbf{0.60}_{\pm 0.005}$	$\mathbf{0.58}_{\pm 0.002}$

The full scheduler achieves the strongest overall factual and edit-sensitive performance, reaching FGT@0.5 of 0.83, ESR@2 of 0.60, and relation-edit ESR of 0.58, while BLEU remains at 0.65 across the conditioning-aware variants. Three additional analyses support this behavior. First, the gains depend on where the conditioning weights are applied: randomly shuffling $\omega_{t,i}$ across positions reduces FGT@0.5 to 0.76, while rank reversal reduces it further to 0.72, below the L-only setting (App. G.6). Second, varying the schedule shift shows that performance improves from $\kappa = 0$ to $\kappa = 100$, but provides no further benefit at $\kappa = 200$, suggesting that retaining too much signal can make positions less responsive to the conditioning graph (App. G.5). Third, CFG improves the difficulty-only checkpoint from $0.79/0.54/0.53$ to $0.81/0.56/0.55$ on FGT@0.5/ESR@2/relation ESR, but remains below Full ($0.83/0.60/0.58$), showing that inference-time guidance alone does not recover the same gains (App. G.7). Cross-dataset component attribution and same-checkpoint analyses are provided in Apps. G.3 and G.4.

Accelerated Inference. On WikiOFGraph, 2000-step denoising gives 0.654 BLEU in 89s per batch of 50 on one V100 (DiffuSeq: 0.628 in 317s), while mixed-space decoding reaches 0.468 BLEU in under 5s, matching T5-Small's latency at higher quality. Full quality-latency results are in Appendix H.4.

5 Conclusion and Limitations

We introduced DLM4G, a conditioning-aware diffusion language model for reliable graph-to-sequence generation. Since denoising difficulty does not reliably indicate token-level graph support, DLM4G separates **denoising difficulty** from **graph conditioning**: (1) a global adaptive schedule retains more signal at positions that are harder to recover, while (2) dynamic conditioning modulation uses decoder-to-graph cross-attention and conditioning sensitivity to control how strongly this schedule is applied for the current graph during training and inference. We also introduce FGT and ESR to measure factual grounding and responsiveness to controlled graph edits. Across WikiOFGraph, GenWiki, and TekGEN, conditioning-aware modulation consistently improves grounding, edit sensitivity, and generation quality over difficulty-only adaptation, while the 63M-parameter DLM4G remains competitive with substantially larger models, showing that denoising difficulty and graph conditioning are complementary signals. **Limitations.** We study continuous diffusion over linearized knowledge graphs, which may underuse explicit graph structure; the masked-diffusion extension (App. A.3) is not evaluated. FGT is entity-level and can miss relation errors, which AlignScore and relation-edit ESR partly cover. Decoder-to-graph attention is weakly discriminative at high noise levels.

AI USE STATEMENT

Generative AI tools were used in two roles. First, Qwen2.5-7B-Instruct was used as a fixed evaluation-time post-processing model to extract and canonicalize entity mentions, attribute values, and relation labels for the FGT and ESR metrics, using the same prompt and decoding configuration for all systems. Second, generative AI tools were used to assist with drafting and polishing portions of the manuscript for clarity, readability, and presentation. Generative AI was not used to formulate the research methodology or mathematical claims, design or conduct experiments, implement the proposed method, or interpret experimental results. All AI-assisted work was reviewed by the authors, who take full responsibility for the final content, claims, and conclusions of this work.

REPRODUCIBILITY STATEMENT

We provide an anonymous code repository containing the implementation of DLM4G, including training, evaluation, conditioning-aware schedule construction, and inference code, at `https://anonymous.4open.science/r/DLM4G-G2S-214B/README.md`. The main methodological formulation and training objective are described in Section 3, with complete derivations and training/inference procedures provided in Appendices B-C. Dataset statistics, preprocessing, data splits, baseline protocols, model configurations, and training hyperparameters are documented in Appendix D. Definitions and implementation details for Factual Grounding (FGT), Edit Sensitivity Rate (ESR), controlled graph perturbations, and metric validation are provided in Appendix E. Extended generation metrics, matched-budget experiments, component ablations, and accelerated-inference analyses are reported in Appendices F-H. Together with the released code and configuration files, these details are intended to support reproduction of the reported results.

REFERENCES

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3554–3565, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.278. URL `https://aclanthology.org/2021.naacl-main.278/`.

Hikaru Asano, Tadashi Kozuno, Kuniaki Saito, and Yukino Baba. Where-to-unmask: Ground-truth-guided unmasking order learning for masked diffusion language models, 2026. URL `https://arxiv.org/abs/2602.09501`.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72, Ann Arbor, MI, 2005. Association for Computational Linguistics.

Dat Nguyen Cong, Tung Kieu, and Hoang Thanh-Tung. FastDiSS: Few-step match many-step diffusion language model on sequence-to-sequence generation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17572–17592, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.870. URL `https://aclanthology.org/2026.findings-acl.870/`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423/`.

Pierre Dognin, Inkit Padhi, Igor Melnyk, and Payel Das. ReGen: Reinforcement learning for text and knowledge base generation using pretrained language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1084–1099, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.83. URL https://aclanthology.org/2021.emnlp-main.83/.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=jQj-_rLVXsj.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq-v2: Bridging discrete and continuous text spaces for accelerated seq2seq diffusion models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023b. URL https://openreview.net/forum?id=OLcDbSRjbx.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. CycleGT: Unsupervised graph-to-text and text-to-graph generation via cycle training. In Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina (eds.), *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pp. 77–88, Dublin, Ireland (Virtual), 12 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.webnlg-1.8/.

Jie He, Yijun Yang, Wanqiu Long, Deyi Xiong, Victor Gutierrez Basulto, and Jeff Z. Pan. Evaluating and improving graph to text generation with large language models. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10219–10244, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.513. URL https://aclanthology.org/2025.naacl-long.513/.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Vincent Hu, Di Wu, Yuki Asano, Pascal Mettes, Basura Fernando, Björn Ommer, and Cees Snoek. Flow matching for conditional text generation in a few sampling steps. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 380–392, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-short.33. URL https://aclanthology.org/2024.eacl-short.33/.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. GenWiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 2398–2409, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.217. URL https://aclanthology.org/2020.coling-main.217/.

Daehee Kim, Deokhyung Kang, Sangwon Ryu, and Gary Geunbae Lee. Ontology-free general-domain knowledge graph-to-text generation dataset synthesis using large language model, 2024. URL https://arxiv.org/abs/2409.07088.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. GPT-too: A language-model-first approach for AMR-to-text generation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1846–1852, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.167. URL https://aclanthology.org/2020.acl-main.167/.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025. URL https://arxiv.org/abs/2502.09992.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin (eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina (eds.), *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pp. 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049. URL https://aclanthology.org/W15-3049/.

Gabriel Raya, Bac Nguyen, Georgios Batzolis, Yuhta Takida, Dejan Stancevic, Naoki Murata, Chieh-Hsin Lai, Yuki Mitsufuji, and Luca Ambrogioni. Noise scheduling as information-guided allocation in diffusion training, 2026. URL https://arxiv.org/abs/2602.18647.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. Modeling global and local node contexts for text generation from knowledge graphs. *Transactions of the Association for Computational Linguistics*, 8:589–604, 2020. doi: 10.1162/tacl_a_00332. URL https://aclanthology.org/2020.tacl-1.38/.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. Investigating pretrained language models for graph-to-text generation. In Alexandros Papangelis, Paweł Budzianowski, Bing Liu, Elnaz Nouri, Abhinav Rastogi, and Yun-Nung Chen (eds.), *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pp. 211–227, Online, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.nlp4convai-1.20. URL https://aclanthology.org/2021.nlp4convai-1.20/.

Subham Sekhar Sahoo, Marianne Arriola, Aaron Gokaslan, Edgar Mariano Marroquin, Alexander M Rush, Yair Schiff, Justin T Chiu, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=L4uaAR4ArM.

Subham Sekhar Sahoo, Aaron Gokaslan, Christopher De Sa, and Volodymyr Kuleshov. Diffusion models with learned adaptive noise. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=loMa99A4p8.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Qing Wang, Jacob Devasier, and Chengkai Li. What gets unmasked first? trajectory analysis of diffusion models for graph-to-text generation, 2026. URL https://arxiv.org/abs/2605.31564.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. Text diffusion model with encoder-decoder transformers for sequence-to-sequence generation. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 22–39, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.2. URL https://aclanthology.org/2024.naacl-long.2/.

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. AlignScore: Evaluating factual consistency with a unified alignment function. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11328–11348, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.634. URL https://aclanthology.org/2023.acl-long.634/.

Qi Zhan, Yile Wang, and Hui Huang. Assessment of generative named entity recognition in the era of large language models, 2026. URL https://arxiv.org/abs/2601.17898.

Kun Zhang, Oana Balalau, and Ioana Manolescu. FactSpotter: Evaluating the factual faithfulness of graph-to-text generation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10025–10042, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.672. URL https://aclanthology.org/2023.findings-emnlp.672/.

Xueyu Zhou, Yangrong Hu, and Jian Huang. DOS: Dependency-oriented sampler for masked diffusion language models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17404–17419, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.861. URL https://aclanthology.org/2026.findings-acl.861/.

A Extended Related Work and Positioning

This section expands the related work of the main paper. We focus on two questions that are directly relevant to DLM4G: what signal is used to adapt the diffusion process, and whether that signal depends on the external conditioning input. We then distinguish factual grounding from controlled-edit responsiveness and briefly clarify the scope of the continuous-diffusion formulation used in this work.

A.1 Adaptive Diffusion Methods

Existing adaptive diffusion methods use signals such as denoising loss, output context, reconstruction confidence, self-attention, or reveal order to decide how corruption or denoising should vary across positions. These signals are computed from the sample or the current denoising state; MuLAN is sample-adaptive, while methods such as SeqDiffuSeq, Dream, FastDiSS, and DOS use quantities derived from the evolving output or denoising process. They can therefore indicate that a position is difficult, uncertain, or informative, but do not by themselves distinguish whether that position is determined by the conditioning graph or by the evolving output sequence. Conditioning sensitivity is different in this respect: it is defined only in conditional generation, because it measures how much the prediction changes when the conditioning input is removed. DLM4G therefore keeps denoising difficulty as the global training signal and uses graph-conditioned signals $A_{t,i}$ and $\widehat{\Delta}_{t,i}$ to determine how strongly the adaptive schedule is applied to the current input.

Relation to SeqDiffuSeq. SeqDiffuSeq (Yuan et al., 2024) is the closest precursor to the global adaptive component of DLM4G. It uses token-position denoising difficulty to construct an adaptive noise schedule. DLM4G keeps this basic idea, but our Q1 analysis shows that denoising difficulty and

Table 5: **Extended positioning of DLM4G.** Signal domain indicates the information from which the adaptation criterion is computed. Stage denotes training-time adaptation (T), inference-time adaptation (I), or evaluation only (E). F/E denotes evaluation of factual grounding (F) and controlled-edit responsiveness (E).

Method	Setting	Adaptation signal / criterion	Signal domain	Stage	F/E
Graph-to-text grounding					
FactSpotter (Zhang et al., 2023)	G2S	Triple-level factual support	Input × output	E	F/–
PlanGTG (He et al., 2025)	G2S/LLM	Graph planning and attribution	Input × output	T/E	F/–
Continuous adaptive diffusion					
DiffuSeq (Gong et al., 2023a)	Cont. Seq2Seq	Fixed shared noise schedule	Fixed	T	–/–
FlowSeq (Hu et al., 2024)	Cont. Seq2Seq	Fixed flow / noise process	Fixed	T	–/–
SeqDiffuSeq (Yuan et al., 2024)	Cont. Seq2Seq	Token-position denoising loss	Denoising state	T	–/–
MuLAN (Sahoo et al., 2024b)	Cont. diffusion	Learned sample-adaptive noise	Sample	T	–/–
InfoNoise (Raya et al., 2026)	Diffusion	Information rate from denoising loss	Denoising state	T	–/–
FastDiSS (Cong et al., 2026)	Cont. Seq2Seq	Reconstruction confidence	Output prediction	T	–/–
Masked / discrete diffusion					
Dream (CART) (Ye et al., 2025)	MDLM	Output-context informativeness	Output context	T	–/–
DOS (Zhou et al., 2026)	MDLM	Inter-token self-attention	Output state	I	–/–
Where-to-Unmask / LookUM (Asano et al., 2026)	MDLM	Learned / lookahead reveal order	Output state	I	–/–
λ-scaled decoding (Wang et al., 2026)	G2S-MDLM	Structural-token confidence	Conditioning structure + output confidence	I	F/–
DLM4G	**G2S-CDLM**	**Global loss schedule + dynamic A, Δ**	**Input graph × denoising state**	**T/I**	**F/E**

MuLAN adapts its forward process to the sample itself rather than to a separate structured conditioning input. The λ-scaled decoding row refers to the inference-time structural decoding intervention; the same work also introduces Graph-LLaDA with an explicit graph encoder. Its reported graph-faithfulness analysis does not include controlled-edit evaluation. In DLM4G, the global difficulty-based schedule is learned during training, while $A_{t,i}$ and $\widehat{\Delta}_{t,i}$ are recomputed on the fly to modulate it during both training and inference.

token-level factual support are only partially aligned. We therefore treat the difficulty-based schedule as a global schedule and modulate it using information from the current graph. This changes the role of the adaptive schedule from difficulty-only allocation to conditioning-aware allocation, rather than simply changing the parameterization of the same difficulty signal.

Relation to learned adaptive noise. MuLAN (Sahoo et al., 2024b) takes a different approach and learns a sample-adaptive multivariate noise process within a variational formulation. DLM4G instead computes $\omega_{t,i}$ from two model-derived quantities: decoder-to-graph attention and conditioning sensitivity. Directly differentiating through this schedule would create a feedback path in which the model changes the same signals that determine how its inputs are corrupted. We therefore stop gradients through $\omega_{t,i}$ when constructing the schedule. For this reason, our objective is motivated by the conditional VLB, but we do not treat the resulting model-dependent adaptive objective as an exact likelihood bound. The training objective is discussed in Appendix B.

Relation to graph-aware masked diffusion. Recent graph-aware masked-diffusion work (Wang et al., 2026) studies λ-scaled decoding, which changes structural-token confidence during inference, and also introduces Graph-LLaDA with an explicit graph encoder. This is related in motivation but differs in where the intervention is applied: λ-scaled decoding is an inference-time decoding rule, whereas DLM4G uses graph-derived conditioning signals to modulate the noise schedule during both training and inference. DLM4G also does not assign a fixed structural class to an output position; its conditioning weight changes with the current graph, noisy state, and denoising step.

A.2 Graph-Conditioned Factual Generation

Graph-to-sequence generation has progressed from neural graph encoders to pretrained Transformer models over serialized graphs (Ribeiro et al., 2020; 2021). Fine-tuned language models provide strong reference agreement, but reference similarity alone does not determine whether generated information is supported by the input graph. Prior graph-to-text factuality methods therefore evaluate graph support directly, while other graph-conditioned methods use planning or attribution to improve the use of structured input (Zhang et al., 2023; He et al., 2025).

Our evaluation separates two properties. Factual Grounding (FGT) measures whether information realized in a generated sequence is supported by the fixed input graph. Edit Sensitivity Rate (ESR) instead measures whether a controlled change to the graph produces the corresponding change in the output. These properties are related but not equivalent: a generator can produce a well-grounded output for the original graph and still fail to react when a relation or value is changed. This distinction motivates the factual and edit-sensitive evaluation in Appendix E.

A.3 Scope Beyond Continuous Diffusion

DLM4G is instantiated in a continuous text-diffusion model, where the position-specific schedule is represented by $\bar{\alpha}_{t,i}$ and therefore has a direct signal-to-noise interpretation. The same high-level distinction between a global difficulty signal and input-dependent conditioning signals is not specific to continuous diffusion. In a masked diffusion model, an analogous mechanism could act through position-specific masking or reveal probabilities. We do not evaluate that extension here, and the empirical claims in this paper are therefore limited to the continuous formulation used in our experiments.

B Derivation of the Training Objective

The main paper gives the end-to-end objective used by DLM4G. Here we provide the corresponding fixed-schedule diffusion derivation and then explain how the same objective is used with the token-wise schedule in our implementation. The important distinction is that, once the conditioning weight is used to construct the schedule, the resulting schedule is treated as fixed with respect to gradients when the noisy state and loss are evaluated.

B.1 Forward and Reverse Diffusion

For a fixed scalar schedule, the forward Markov chain is

$$q(\mathbf{z}_{1:T} \mid \mathbf{z}_0) = \prod_{t=1}^{T} q(\mathbf{z}_t \mid \mathbf{z}_{t-1}), \tag{6}$$

with Gaussian transitions

$$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t \mid \sqrt{\alpha_t}\mathbf{z}_{t-1}, (1-\alpha_t)\mathbf{I}) \,. \tag{7}$$

Let $\bar{\alpha}_t = \prod_{k=1}^{t} \alpha_k$. The corresponding marginal can be sampled directly as

$$\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{8}$$

Our implementation uses the same *sqrt*-style baseline schedule used in the DiffusionLM family (Li et al., 2022). The per-step coefficients are recovered from the corresponding cumulative schedule as

$$\alpha_t^{\text{base}} = \frac{\bar{\alpha}_t^{\text{base}}}{\bar{\alpha}_{t-1}^{\text{base}}}, \qquad \bar{\alpha}_0^{\text{base}} = 1. \tag{9}$$

DLM4G generalizes Eq. 8 to a schedule that can vary across output positions:

$$\mathbf{z}_{t,i} = \sqrt{\bar{\alpha}_{t,i}}\,\mathbf{z}_{0,i} + \sqrt{1-\bar{\alpha}_{t,i}}\,\boldsymbol{\epsilon}_i, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{10}$$

Conditioned on a fixed token-wise schedule, the forward process remains factorized across output positions. For position i, define

$$\beta_{t,i} = 1 - \alpha_{t,i}, \qquad \bar{\alpha}_{t,i} = \prod_{k=1}^{t} \alpha_{k,i}.$$

The corresponding posterior is

$$q(\mathbf{z}_{t-1,i} \mid \mathbf{z}_{t,i}, \mathbf{z}_{0,i}) = \mathcal{N}\left(\tilde{\boldsymbol{\mu}}_{t,i}, \tilde{\beta}_{t,i}\mathbf{I}\right), \tag{11}$$

where

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_{t,i} &= \frac{\sqrt{\alpha_{t,i}}(1-\bar{\alpha}_{t-1,i})}{1-\bar{\alpha}_{t,i}}\,\mathbf{z}_{t,i} + \frac{\sqrt{\bar{\alpha}_{t-1,i}}\,\beta_{t,i}}{1-\bar{\alpha}_{t,i}}\,\mathbf{z}_{0,i}, \\ \tilde{\beta}_{t,i} &= \frac{1-\bar{\alpha}_{t-1,i}}{1-\bar{\alpha}_{t,i}}\,\beta_{t,i}. \end{aligned} \tag{12}$$

Thus, once the token-wise schedule is fixed, the usual Gaussian reverse-process expressions apply independently at each output position.

B.2 Conditional VLB Motivation

For a fixed forward process, the conditional negative log-likelihood is bounded by the standard diffusion variational objective. Following the usual decomposition (Sohl-Dickstein et al., 2015),

$$\mathcal{L}_{\mathrm{VLB}} = \mathbb{E}_q \left[\mathrm{KL}(q(\mathbf{z}_T \mid \mathbf{z}_0) \,\|\, p(\mathbf{z}_T)) + \sum_{t=2}^{T} \mathrm{KL}(q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{z}_0) \,\|\, p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{G})) - \log p_\theta(\mathbf{z}_0 \mid \mathbf{z}_1, \mathbf{G}) \right]. \tag{13}$$

When the reverse covariance is fixed to the forward posterior covariance, each intermediate KL term reduces to a timestep-weighted squared reconstruction error. As in the standard simplified diffusion objective, we drop the timestep-dependent weights and optimize the corresponding unweighted $\mathbf{z}_0$-prediction loss (Ho et al., 2020):

$$\mathcal{L}_t \propto \sum_{i \in \mathcal{I}(\mathbf{S})} \left\| \mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t, \mathbf{G})^{(i)} - \mathbf{z}_{0,i} \right\|_2^2, \qquad 2 \le t \le T. \tag{14}$$

Here $\mathcal{I}(\mathbf{S})$ denotes the non-`[PAD]` output positions. Equation 13 provides the fixed-schedule motivation for the objective in the main paper. In `DLM4G`, the token-wise schedule also depends on model-derived conditioning signals. We therefore stop gradients through the conditioning weight when constructing the schedule and optimize the resulting $\mathbf{z}_0$-prediction objective. The VLB motivates the form of the loss; we do not treat the model-dependent adaptive objective as an exact bound on the conditional log-likelihood.

B.3 End-to-End $\mathbf{z}_0$-Prediction Objective

The discrete target sequence $\mathbf{S} = (s_1, \ldots, s_N)$ is embedded as $\mathbf{z}_0 = g_\Phi(\mathbf{S})$. In addition to the denoising terms for $t \ge 2$, the model uses a consistency term at $t = 1$ and a discrete rounding term. The rounding distribution factorizes over valid output positions:

$$\tilde{p}_\Phi(\mathbf{S} \mid \mathbf{z}_0) = \prod_{i \in \mathcal{I}(\mathbf{S})} \tilde{p}_\Phi(s_i \mid \mathbf{z}_{0,i}). \tag{15}$$

For each position, $\tilde{p}_\Phi(s_i \mid \mathbf{z}_{0,i}) = \mathrm{Softmax}(\mathbf{W}\mathbf{z}_{0,i})_{s_i}$. For each training example, once $\bar{\boldsymbol{\alpha}}^{\mathrm{final}}$ has been constructed, it is treated as fixed with respect to gradients when the noisy state and loss are evaluated. The construction of $\bar{\boldsymbol{\alpha}}^{\mathrm{final}}$ from the probe state, including the subsequent re-noising step used to obtain the training state, is described in Appendix C.3. Combining the denoising, consistency, and rounding terms gives the objective reported in the main paper:

$$\mathcal{L}_{\text{e2e-simple}}(\mathbf{S}) = \mathbb{E}_{q_{\bar{\boldsymbol{\alpha}}^{\mathrm{final}}}(\mathbf{z}_{1:T} \mid \mathbf{z}_0)} \left[\sum_{t=2}^{T} \sum_{i \in \mathcal{I}(\mathbf{S})} \left\| \mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t^{\mathrm{final}}, \mathbf{G})^{(i)} - \mathbf{z}_{0,i} \right\|_2^2 + \sum_{i \in \mathcal{I}(\mathbf{S})} \left\| \mathbf{z}_{0,i} - \mathcal{M}_\theta(\mathbf{z}_1, \bar{\boldsymbol{\alpha}}_1^{\mathrm{final}}, \mathbf{G})^{(i)} \right\|_2^2 - \sum_{i \in \mathcal{I}(\mathbf{S})} \log \tilde{p}_\Phi(s_i \mid \mathbf{z}_{0,i}) \right]. \tag{16}$$

During training, the rounding term is evaluated using the gold token s_i. During generation, the final clean-latent prediction is mapped to tokens through $\tilde{p}_\Phi(\cdot \mid \widehat{\mathbf{z}}_0)$.

B.4 Practical Single-Timestep Training Estimator

Computing a denoising loss at every timestep for each optimization step is unnecessary. In practice, we optimize the denoising objective by uniformly sampling $t \sim \mathrm{Uniform}\{2, \ldots, T\}$ and evaluating the denoising loss only at the sampled timestep: $\widehat{\mathcal{L}}_{\mathrm{den}} = \sum_{i \in \mathcal{I}(\mathbf{S})} \left\| \mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t^{\mathrm{final}}, \mathbf{G})^{(i)} - \mathbf{z}_{0,i} \right\|_2^2$. The corresponding denoising objective can be written as $\mathcal{L}_{\mathrm{den}} = \mathbb{E}_{t \sim \mathrm{Uniform}\{2,\ldots,T\}} \left[\widehat{\mathcal{L}}_{\mathrm{den}} \right]$. Thus, training uses one uniformly sampled noise level per optimization step rather than explicitly evaluating all $T - 1$ denoising terms. The $t = 1$ consistency term and the discrete rounding term are evaluated separately and combined with the sampled denoising loss using the same convention throughout all experiments. The running difficulty statistic $\bar{\ell}_i$ is updated only from valid non-`[PAD]` occurrences of position i. Per-position denoising losses are recorded before reduction across output positions. Because t is sampled uniformly, $\bar{\ell}_i$ aggregates denoising difficulty across noise levels rather than representing the error at a single fixed timestep. This statistic is used to construct the global adaptive schedule described in Appendix C.1. The $t = 1$ consistency term uses the training-time schedule construction described in Appendix C.3.

C DLM4G SCHEDULER IMPLEMENTATION DETAILS

This section gives the complete schedule construction used during training and inference. The implementation follows the two-component description in the main paper: a blockwise global schedule learned from denoising difficulty and a dynamic conditioning weight computed from the current graph and noisy sequence. The global adaptive schedule is represented by its cumulative signal-retention coefficient $\bar{\alpha}_{t,i}^{\text{adapt}}$; per-step coefficients are recovered from it only when required by the reverse update.

C.1 GLOBAL ADAPTIVE SCHEDULE

For every valid output position i, we maintain the running denoising difficulty $\bar{\ell}_i$. Let $\ell_{\min}$ and $\ell_{\max}$ denote the minimum and maximum running losses across valid output positions. We first normalize the difficulty as $r_i = \frac{\bar{\ell}_i - \ell_{\min}}{\ell_{\max} - \ell_{\min} + \varepsilon}$. The global adaptive schedule is constructed by shifting the cumulative signal-retention schedule according to r_i:

$$\bar{\alpha}_{t,i}^{\text{adapt}} = \bar{\alpha}_t^{\text{base}} + r_i \left(\bar{\alpha}_{\max(t-\kappa,0)}^{\text{base}} - \bar{\alpha}_t^{\text{base}} \right), \tag{17}$$

where $\kappa = 100$ controls the maximum schedule shift and $\bar{\alpha}_0^{\text{base}} = 1$. Because the cumulative signal-retention schedule decreases with diffusion time,

$$\bar{\alpha}_{\max(t-\kappa,0)}^{\text{base}} \geq \bar{\alpha}_t^{\text{base}}, \tag{18}$$

larger denoising difficulty produces larger $\bar{\alpha}_{t,i}^{\text{adapt}}$ at the same nominal timestep. Difficult positions therefore retain more signal and are corrupted more conservatively.

Whenever a per-step coefficient is required, it is recovered from the cumulative schedule as

$$\alpha_{t,i}^{\text{adapt}} = \frac{\bar{\alpha}_{t,i}^{\text{adapt}}}{\bar{\alpha}_{t-1,i}^{\text{adapt}}}, \qquad \bar{\alpha}_{0,i}^{\text{adapt}} = 1. \tag{19}$$

Training begins with a 20K-step warm-up using the base schedule. The first global adaptive schedule is constructed after this warm-up and is subsequently updated every $K_{\text{up}} = 20\text{K}$ optimization steps. The schedule is held fixed between updates. This blockwise construction prevents the running difficulty statistics from continuously changing the global corruption schedule during optimization.

C.2 DYNAMIC CONDITIONING MODULATION

The global schedule captures which output positions are difficult to denoise, but it does not determine how strongly that adaptation should be applied for a particular conditioning graph. We therefore compute two graph-dependent signals from the current noisy state.

For attention head h, let $a_{t,i,j}^{(h)}$ denote decoder-to-graph cross-attention from output position i to serialized graph-token position j. If $\mathcal{I}_G$ indexes the graph tokens and H is the number of attention heads, we compute

$$A_{t,i} = \frac{1}{H} \sum_{h=1}^{H} \max_{j \in \mathcal{I}_G} a_{t,i,j}^{(h)}. \tag{20}$$

Because each cross-attention row is already normalized over graph positions, $A_{t,i}$ measures the peak concentration of decoder-to-graph attention rather than the total attention mass assigned to the graph.

Conditioning sensitivity compares predictions from the same noisy state with and without the conditioning graph:

$$\Delta_{t,i} = \frac{\left\|\mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t, \mathbf{G})^{(i)} - \mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t, \emptyset)^{(i)}\right\|_2}{\left\|\mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t, \mathbf{G})^{(i)}\right\|_2 + \varepsilon}, \qquad \widehat{\Delta}_{t,i} = \frac{\Delta_{t,i}}{1 + \Delta_{t,i}}. \tag{21}$$

Here $\bar{\boldsymbol{\alpha}}_t$ denotes the schedule associated with the current noisy state: the global adaptive schedule for the training probe state and the locally constructed effective schedule during inference.

Graph conditioning is randomly dropped during training so that the same model supports both conditioned and unconditioned evaluations. The prediction difference in Eq. 21 is not added to the denoising prediction; it is used only to construct the schedule.

The two conditioning signals are combined as

$$\omega_{t,i} = \mathrm{StopGrad}\Big[A_{t,i} + (1 - A_{t,i})\widehat{\Delta}_{t,i}\Big]. \tag{22}$$

For a schedule constructed at timestep t, $\omega_{t,i}$ controls how strongly the global adaptive schedule is applied. The corresponding per-step coefficient for prefix index $k \le t$ is

$$\alpha_{k,i}^{\mathrm{final}}(\omega_{t,i}) = (1 - \omega_{t,i})\alpha_k^{\mathrm{base}} + \omega_{t,i}\alpha_{k,i}^{\mathrm{adapt}}. \tag{23}$$

The cumulative coefficient used to construct the noisy state is therefore

$$\bar{\alpha}_{t,i}^{\mathrm{final}}(\omega_{t,i}) = \prod_{k=1}^{t}\Big[(1 - \omega_{t,i})\alpha_k^{\mathrm{base}} + \omega_{t,i}\alpha_{k,i}^{\mathrm{adapt}}\Big]. \tag{24}$$

For a given schedule construction, the same conditioning weight $\omega_{t,i}$ is held fixed across the prefix index k. Thus, $\omega_{t,i} = 0$ recovers the base schedule, whereas $\omega_{t,i} = 1$ applies the complete global adaptive schedule.

C.3 Training-Time State Construction and Re-Noising

The dynamic conditioning weight depends on a noisy state, while the final noisy state used by the training objective should follow the schedule after conditioning modulation. We therefore separate the state used to estimate the conditioning signals from the state used to evaluate the denoising objective.

Within a training block, the computation is

$$\boxed{\bar{\boldsymbol{\alpha}}^{\mathrm{adapt}}} \longrightarrow \boxed{\mathbf{z}_t^{\mathrm{probe}}} \longrightarrow \boxed{A_{t,i}, \widehat{\Delta}_{t,i}} \longrightarrow \boxed{\omega_{t,i}} \longrightarrow \boxed{\bar{\alpha}_{t,i}^{\mathrm{final}}} \longrightarrow \boxed{\mathbf{z}_t^{\mathrm{train}}} \longrightarrow \boxed{\mathcal{L}}. \tag{25}$$

First, the probe state is sampled from the blockwise-fixed global adaptive schedule:

$$\mathbf{z}_{t,i}^{\mathrm{probe}} = \sqrt{\bar{\alpha}_{t,i}^{\mathrm{adapt}}}\,\mathbf{z}_{0,i} + \sqrt{1 - \bar{\alpha}_{t,i}^{\mathrm{adapt}}}\,\boldsymbol{\epsilon}_i^{\mathrm{probe}}, \qquad \boldsymbol{\epsilon}_i^{\mathrm{probe}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{26}$$

We compute $A_{t,i}$ and $\widehat{\Delta}_{t,i}$ from this probe state and use them to obtain $\omega_{t,i}$. Equation 24 then gives the cumulative schedule for the training state.

The state used by the denoising objective is independently re-noised from this final schedule:

$$\mathbf{z}_{t,i}^{\mathrm{train}} = \sqrt{\bar{\alpha}_{t,i}^{\mathrm{final}}}\,\mathbf{z}_{0,i} + \sqrt{1 - \bar{\alpha}_{t,i}^{\mathrm{final}}}\,\boldsymbol{\epsilon}_i^{\mathrm{train}}, \qquad \boldsymbol{\epsilon}_i^{\mathrm{train}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{27}$$

The probe and final Gaussian noises are sampled independently. The denoising loss and the running difficulty statistics are computed from $\mathbf{z}_t^{\mathrm{train}}$. At the end of each 20K-step block, the accumulated position-wise losses update $\bar{\boldsymbol{\alpha}}^{\mathrm{adapt}}$ for the next block.

Table 6: **Scheduler implementation.** Training and inference conventions used in all DLM4G experiments.

Setting	Value
Initial base-schedule warm-up	20K optimization steps
Global schedule update interval K_{up}	20K optimization steps
Maximum cumulative-schedule shift κ	100
Gradient through $\omega_{t,i}$	Stopped
$\omega_{t,i}$ within constructed schedule	Shared across prefix $k \leq t$
Probe/final Gaussian-noise reuse	No (independent draws)
Initial carried weight $\omega_{T+1,i}$	1 (global adaptive schedule)
$t = 1$ conditioning-weight convention	Train: $\omega_{1,i}$ from probe; infer: carried $\omega_{2,i}$
Graph-conditioning dropout probability	0.5

C.4 Inference-Time Schedule Construction

After training, the global adaptive schedule $\bar{\boldsymbol{\alpha}}^{\text{adapt}}$ and its derived per-step coefficients $\boldsymbol{\alpha}^{\text{adapt}}$ are fixed.

Inference differs from training because the current conditioning weight $\omega_{t,i}$ cannot be computed until the noisy state $\mathbf{z}_t$ already exists. We therefore use the conditioning weight carried from the preceding reverse step, denoted $\omega_{t+1,i}$, to construct the schedule for the current transition.

For reverse step t, define the local effective per-step coefficients

$$\alpha_{k,i}^{\text{final}} = (1 - \omega_{t+1,i})\alpha_k^{\text{base}} + \omega_{t+1,i}\alpha_{k,i}^{\text{adapt}}, \qquad k \leq t. \tag{28}$$

The corresponding cumulative coefficients needed for the current transition are

$$\bar{\alpha}_{r,i}^{\text{final}} = \prod_{k=1}^{r} \alpha_{k,i}^{\text{final}}, \qquad r \in \{t-1, t\}. \tag{29}$$

The conditional model prediction is then

$$\widehat{\mathbf{z}}_0 = \mathcal{M}_\theta\left(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t^{\text{final}}, \mathbf{G}\right). \tag{30}$$

The same current state is used to obtain $A_{t,i}$ and $\widehat{\Delta}_{t,i}$, from which we compute $\omega_{t,i}$ using Eq. 22. This newly computed weight is carried to the next reverse transition.

For the current transition, let

$$\alpha_{t,i}^{\text{final}} = \frac{\bar{\alpha}_{t,i}^{\text{final}}}{\bar{\alpha}_{t-1,i}^{\text{final}}}. \tag{31}$$

We use the corresponding posterior-form Gaussian update

$$\mathbf{z}_{t-1,i} = \frac{\sqrt{\bar{\alpha}_{t-1,i}^{\text{final}}}\left(1 - \alpha_{t,i}^{\text{final}}\right)}{1 - \bar{\alpha}_{t,i}^{\text{final}}}\widehat{\mathbf{z}}_{0,i} \tag{32}$$

$$+ \frac{\sqrt{\alpha_{t,i}^{\text{final}}}\left(1 - \bar{\alpha}_{t-1,i}^{\text{final}}\right)}{1 - \bar{\alpha}_{t,i}^{\text{final}}}\mathbf{z}_{t,i} + \sigma_{t,i}\boldsymbol{\epsilon}_i, \tag{33}$$

where

Algorithm 2 DLM4G training with blockwise global adaptation and dynamic conditioning modulation

Require: Dataset $\mathcal{D}$, base cumulative schedule $\bar{\boldsymbol{\alpha}}^{\text{base}}$, denoiser $\mathcal{M}_\theta$, warm-up K_{warm}, update interval K_{up}

Train for $K_{\text{warm}} = 20$K steps using the base schedule and accumulate position-wise denoising losses $\bar{\ell}_i$
Construct $\bar{\alpha}_{t,i}^{\text{adapt}}$ using Eq. 17
Recover $\alpha_{t,i}^{\text{adapt}}$ using Eq. 19
for each block of $K_{\text{up}} = 20$K optimization steps **do**
  for each minibatch $(\mathbf{G}, \mathbf{S})$ **do**
    Sample $t \sim \text{Uniform}\{2, \dots, T\}$
    Sample $\mathbf{z}_t^{\text{probe}}$ using $\bar{\alpha}_{t,i}^{\text{adapt}}$
    Compute $A_{t,i}$ from the conditional probe evaluation
    Compute $\widehat{\Delta}_{t,i}$ from conditioned and unconditioned predictions of the same probe state
    Set

$$\omega_{t,i} = \text{StopGrad}\left[A_{t,i} + (1 - A_{t,i})\widehat{\Delta}_{t,i}\right]$$

    Construct $\bar{\alpha}_{t,i}^{\text{final}}$ using Eq. 24
    Independently re-noise $\mathbf{z}_t^{\text{train}}$ using $\bar{\alpha}_{t,i}^{\text{final}}$
    Optimize the $\mathbf{z}_0$-prediction objective
    Update $\bar{\ell}_i$ using non-`[PAD]` positions
  end for
  Update $\bar{\alpha}_{t,i}^{\text{adapt}}$ from the accumulated $\bar{\ell}_i$
  Recover the corresponding $\alpha_{t,i}^{\text{adapt}}$
end for
return trained parameters θ and final global schedule $\bar{\boldsymbol{\alpha}}^{\text{adapt}}$

$$\sigma_{t,i}^2 = \frac{1 - \bar{\alpha}_{t-1,i}^{\text{final}}}{1 - \bar{\alpha}_{t,i}^{\text{final}}} \left(1 - \alpha_{t,i}^{\text{final}}\right). \tag{34}$$

For a fixed carried $\omega_{t+1,i}$, these coefficients have the same form as the standard Gaussian posterior for the corresponding locally constructed token-wise schedule. Because $\omega_{t,i}$ is recomputed as generation progresses, however, the complete reverse trajectory is state dependent. Equation 33 should therefore be understood as a posterior-form adaptive sampling update rather than the exact reverse posterior of one globally fixed forward diffusion chain.

The use of the carried $\omega_{t+1,i}$ introduces a one-step difference between training and inference. During training, a probe state can be generated first and $\omega_{t,i}$ can then be used to re-noise the same timestep. During generation, $\mathbf{z}_t$ must already exist before $\omega_{t,i}$ can be computed, so the transition at t uses the weight obtained at the preceding reverse step. This carried-weight convention avoids a circular dependency between the current noisy state and the schedule used to construct it.

We initialize

$$\omega_{T+1,i} = 1, \tag{35}$$

so the first reverse step uses the learned global adaptive schedule. At $t = 1$, the reverse update uses the carried $\omega_{2,i}$ and $\bar{\alpha}_{0,i}^{\text{final}} = 1$.

C.5 Complete Training and Inference Algorithms

Algorithm 2 summarizes the blockwise training procedure, including the probe state and final re-noising step. Algorithm 3 gives the corresponding state-dependent reverse-generation procedure.

Algorithm 3 DLM4G reverse generation with dynamic conditioning modulation

Require: Graph $\mathbf{G}$, trained denoiser $\mathcal{M}_\theta$, base schedule $\bar{\boldsymbol{\alpha}}^{\text{base}}$, learned global schedule $\bar{\boldsymbol{\alpha}}^{\text{adapt}}$

Recover $\boldsymbol{\alpha}^{\text{base}}$ and $\boldsymbol{\alpha}^{\text{adapt}}$ from their cumulative schedules
Sample initial noisy state $\mathbf{z}_T$
Set $\omega_{T+1,i} = 1$ for all valid output positions
for $t = T, T-1, \ldots, 1$ **do**
    Construct $\alpha_{k,i}^{\text{final}(t)}$ using carried $\omega_{t+1,i}$
    Construct $\bar{\alpha}_{t,i}^{\text{final}(t)}$ and $\bar{\alpha}_{t-1,i}^{\text{final}(t)}$
    Predict

$$\widehat{\mathbf{z}}_0 = \mathcal{M}_\theta\left(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t^{\text{final}(t)}, \mathbf{G}\right)$$

    Read $A_{t,i}$ from the conditional evaluation
    Compute $\widehat{\Delta}_{t,i}$ using an additional unconditional evaluation of the same state
    Set

$$\omega_{t,i} = \text{StopGrad}\left[A_{t,i} + (1 - A_{t,i})\widehat{\Delta}_{t,i}\right]$$

    Sample $\mathbf{z}_{t-1}$ using the posterior-form adaptive update in Eq. 33
end for
Decode $\widehat{\mathbf{S}}$ from $\tilde{p}_\Phi(\cdot \mid \widehat{\mathbf{z}}_0)$
return $\widehat{\mathbf{S}}$

Table 7: **Dataset statistics and experimental splits for the G2S benchmarks.** Triple statistics report the minimum, maximum, and average number of triples per example. For WikiOFGraph and GenWiki, 100K examples are held out from the released training partition for validation and model selection.

Dataset	# predicates	# entities	# triples (min/max/avg)	Train	Dev	Test
WikiOFGraph	140,733	8.2M	1/17/3.62	5,751,776	100,000	100,000
GenWiki-FINE	287	86.6K	1/10/2.64	581,436	100,000	1,000
TekGEN	50,861	4.3M	1/54/1.73	6,310,061	788,746	796,982

D Experimental Setup and Reproducibility

This section collects the dataset, serialization, baseline, training, metric, and model-selection details used throughout the experiments. We report the training budgets and evaluation settings explicitly because different baseline families follow different standard training regimes.

D.1 Datasets and Splits

WikiOFGraph. We use WikiOFGraph (Kim et al., 2024), which contains approximately 5.85M graph–text pairs extracted from English Wikipedia, where each example pairs a set of RDF-style triples with a textual verbalization. We follow the released train/test partition and hold out 100K examples from the released training set as a validation set for checkpoint and hyperparameter selection. This gives 5,751,776 training examples, 100K validation examples, and the released 100K-example test set.

GenWiki. We use the fine version of GenWiki (Jin et al., 2020). From the released 681,436-example training partition, we hold out 100K examples for validation and model selection, leaving 581,436 examples for training. We use the released 1,000-example test set for final evaluation.

TekGEN. We use TekGEN (Agarwal et al., 2021), which contains aligned Wikidata triple–sentence pairs derived from Wikipedia. We retain the released train, validation, and test partitions, containing 6,310,061, 788,746, and 796,982 examples, respectively.

Matched-budget subset. The full DLM4G models use the training pools in Table 7. We additionally use a controlled 100K-pair setting in which DLM4G-2.o, T5-Small, and SeqDiffuSeq use exactly

the same 100K graph-sequence pairs and the same held-out evaluation inputs. This experiment is reported separately from the full-training-pool results.

D.2 Graph Serialization

Each relational triple (h_i, r_{ij}, t_j) is serialized as

$$\langle \texttt{[HEAD]}\ h_i\ \texttt{[REL]}\ r_{ij}\ \texttt{[TAIL]}\ t_j \rangle, \tag{36}$$

and consecutive triples are separated with `[SEP]`. We preserve the triple order in the released datasets and do not reorder or subsample triples. This keeps the graph representation identical across `DLM4G` and serialized-sequence baselines.

Example. A serialized input can be written as

```
[HEAD] USA [REL] hosted [TAIL] 1994_FIFA_World_Cup [SEP]
[HEAD] USA [REL] capital [TAIL] Washington_D.C. [SEP]
[HEAD] 1994_FIFA_World_Cup [REL] top_scorer [TAIL]
Hristo_Stoichkov
```

with the corresponding target sequence:

> *The United States hosted the 1994 FIFA World Cup; its capital is Washington, D.C., and the tournament's top scorer was Hristo Stoichkov.*

We use linearized graphs because this provides a common encoder–decoder input format for the pretrained language-model and diffusion baselines. Transformer self-attention can model interactions between different triples in the serialized input. We do not assume permutation invariance; the focus of this work is the conditioning-aware noising strategy within this serialized G2S setting.

Table 8: **Tokenized input and target length statistics.** Lengths are measured after tokenization and before truncation. Values report mean $\pm$ standard deviation. Truncation denotes the fraction of examples exceeding the corresponding model limit.

	Serialized graph			Target sequence		
Dataset	**Length**	**Input cap**	**Trunc.**	**Length**	**Output cap**	**Trunc.**
WikiOFGraph	55.6 ± 6.0	128	$< 1\%$	51.0 ± 6.0	64	$\approx 1.5\%$
GenWiki	48.3 ± 5.0	128	$< 1\%$	44.0 ± 5.0	64	$< 0.1\%$
TekGEN	49.0 ± 7.0	128	$< 1\%$	44.5 ± 7.0	64	$\approx 0.3\%$

Length handling. Serialized graphs are short relative to the 128-token encoder limit across all three benchmarks, with mean lengths between approximately 48 and 56 tokens. Target sequences are typically 4–5 tokens shorter and are padded or truncated to a maximum of 64 tokens. Input truncation affects fewer than 1% of examples on each dataset, while target truncation is approximately 1.5% on WikiOFGraph and below 1% on GenWiki and TekGEN.

D.3 Baseline Protocols

We group baselines into four families. Because these families use different training regimes, we report their training-data budgets explicitly.

Fine-tuned PLMs. We compare with GPT-2 Small/Base (Mager et al., 2020) and T5 Small/Large (Ribeiro et al., 2021). These models use a fixed 100K-pair supervised training set for each benchmark and are evaluated on the same held-out inputs used for `DLM4G`. Since the full `DLM4G` runs use larger training pools, the PLM rows in the full-data comparison should be read as reference results rather than as a training-budget-matched comparison. The controlled 100K-pair comparison is reported separately in Appendix F.3.

Zero-shot LLMs. We evaluate GPT-o4-mini, LLaMA-3-8B, Qwen2.5-7B, and DeepSeek-7B without task-specific fine-tuning or in-context training examples. Each model receives the same serialized

graph and the prompt in Appendix D.4. These models are included as zero-shot reference points; we do not compare against separately fine-tuned 7–8B LLMs.

Task-specific G2S systems. We include ReGen on TekGEN and Rule-Based, Ontology-Free, Direct-Transfer, and Noisy-Supervised systems on WikiOFGraph/GenWiki where compatible benchmark results are available. These rows use the corresponding reported benchmark protocol and are not treated as retrained controlled baselines. We do not include CycleGT$_{\text{Base}}$ in the controlled comparison because its reported data split differs from the split used here. Available task-specific comparisons are reported in Appendix F.2.

Diffusion baselines. We compare with DiffuSeq (Gong et al., 2023a), FlowSeq (Hu et al., 2024), and SeqDiffuSeq (Yuan et al., 2024). Reproduced diffusion baselines use the full training partition, the same graph serialization, held-out evaluation inputs, maximum output length, and metric pipeline as `DLM4G`. DiffuSeq and SeqDiffuSeq use DDPM decoding, while FlowSeq uses its standard single-step flow decoding.

Table 9: **Training and decoding settings for reproduced baselines.** Fine-tuned PLMs use the 100K-pair supervised setting, while reproduced diffusion baselines use the full training partitions.

Model	Train pairs	Optimizer	LR	Batch	Budget	Selection	Decoding	Max out.
GPT-2 Small	100K	AdamW	5×10^{-5}	32	3 epochs	Dev BLEU	Greedy	64
GPT-2 Base	100K	AdamW	5×10^{-5}	32	3 epochs	Dev BLEU	Greedy	64
T5 Small	100K	AdamW	5×10^{-5}	32	3 epochs	Dev BLEU	Greedy	64
T5 Large	100K	AdamW	5×10^{-5}	32	3 epochs	Dev BLEU	Greedy	64
DiffuSeq	Full train split	AdamW	1×10^{-4}	128	200K steps	Dev BLEU	DDPM	64
FlowSeq	Full train split	AdamW	1×10^{-4}	128	200K steps	Dev BLEU	1-step flow	64
SeqDiffuSeq	Full train split	AdamW	1×10^{-4}	128	200K steps	Dev BLEU	DDPM	64

D.4 Zero-Shot Prompting

The zero-shot prompt is deliberately short so that the models are evaluated on graph verbalization rather than prompt engineering. The prompt contains the task instruction, an explicit requirement not to introduce unsupported information, and the serialized graph.

Zero-shot prompt template.
Convert the following knowledge graph into a fluent English verbalization. Use only information supported by the graph and keep the output concise.
`<serialized graph>`

No task-specific demonstrations are included. The same prompt template is used across zero-shot models.

Table 10: **Zero-shot decoding settings.** All models use deterministic decoding with the same maximum output length and a single generation per input.

Model	Temperature	Top-p	Sampling	Max new tokens	Repetitions
GPT-o4-mini	0.0	1.0	No	64	1
LLaMA-3-8B	0.0	1.0	No	64	1
Qwen2.5-7B	0.0	1.0	No	64	1
DeepSeek-7B	0.0	1.0	No	64	1

For locally evaluated models, sampling is disabled (`do_sample=False`); temperature and top-p are therefore inactive during decoding. The same 64-token output limit is used for all zero-shot models.

Table 11: **Training and scheduler hyperparameters for DLM4G.**

Category	Hyperparameter	Value
Model		
Architecture	DLM4G-1.o	6 encoder / 6 decoder layers (≈50M)
Architecture	DLM4G-2.o	6 encoder / 9 decoder layers (≈63M)
Activation	Transformer nonlinearity	GeLU
Tokenizer	Vocabulary	bert-base-uncased + [HEAD], [REL], [TAIL], [SEP]
Sequence length	Maximum encoder input length	128 tokens
Sequence length	Maximum output length	64 tokens
Diffusion and scheduler		
Diffusion	Timesteps T	2000
Diffusion	Base schedule	sqrt-style continuous schedule
Scheduler	Initial schedule warm-up	20K optimization steps
Scheduler	Update interval K_{up}	20K optimization steps
Scheduler	Maximum shift κ	100
Scheduler	Conditioning weight	$A_{t,i} + (1 - A_{t,i})\widehat{\Delta}_{t,i}$
Scheduler	Gradient through $\omega_{t,i}$	StopGrad
Conditioning	Graph-drop probability	0.5
Conditioning	Ablation variants	Same $p_{\mathrm{drop}} = 0.5$ for Base/L/L+A/L+Δ/Full
Optimization		
Optimizer	Optimizer	AdamW
Optimization	Peak learning rate	1×10^{-4}
Optimization	Learning-rate warm-up	10K steps
Optimization	Learning-rate decay	Linear decay to zero
Optimization	Gradient clipping	Global norm $= 1.0$
Regularization	Transformer dropout	0.2
Regularization	Label smoothing	None
Training	Batch size	128
Training	Maximum optimization steps	200K per dataset
Evaluation	Independent training seeds	3
Evaluation	Seed values	$\{128, 156, 208\}$

Table 12: **Evaluation protocol.** A fixed held-out evaluation subset is used for every compared model and is kept unchanged across independent training seeds.

Dataset	Eval. examples	Selection	Checkpoint metric	Seeds
WikiOFGraph	1,000	Fixed held-out subset	Dev BLEU	3
GenWiki	1,000	Fixed held-out subset	Dev BLEU	3
TekGEN	1,000	Fixed held-out subset	Dev BLEU	3

D.5 Training and Optimization

Table 11 reports the configuration used for all DLM4G training runs. The 20K scheduler warm-up is separate from the 10K learning-rate warm-up.

D.6 Model Selection and Evaluation Protocol

DLM4G-1.o and DLM4G-2.o are trained for at most 200K optimization steps. For all reproduced trained models, checkpoints are selected using development-set BLEU rather than taking the final optimization step by default.

All reported generation and reliability metrics are computed on the same fixed 1,000-example held-out set for each dataset. Using identical evaluation examples makes comparisons between models paired at the example level. Across the three training seeds, the evaluation inputs remain fixed while parameter initialization, data order, optimization stochasticity, and DLM4G schedule evolution may vary.

Table 13: **Metric implementations used for all systems.** All model outputs are evaluated through the same implementation and post-processing pipeline.

Metric	Implementation	Configuration
BLEU	SacreBLEU	`tok=13a`, mixed case, exponential smoothing
chrF++	SacreBLEU	char-order $= 6$, word-order $= 2$, $\beta = 2$
METEOR	NLTK	Standard English METEOR with stemming and WordNet
BERTScore-F1	`bert-score`	`roberta-large`, F1
MAUVE	`mauve-text`	`gpt2-large` features
FGT	Our implementation	Appendix E.1
ESR	Our implementation	Appendix E.1

The reported standard deviations therefore summarize variation across independent training runs on the fixed evaluation set; they do not represent uncertainty from resampling the evaluation examples.

Checkpoint selection. For each reproduced trained model, we retain the checkpoint with the highest development-set BLEU and evaluate that checkpoint on the fixed held-out set. The same selection rule is used across `DLM4G`, fine-tuned PLMs, and reproduced diffusion baselines.

Training-budget interpretation. The full `DLM4G` and reproduced diffusion runs use the full training partitions, whereas the fine-tuned GPT-2 and T5 reference models use 100K training pairs. We therefore do not treat the full-data `DLM4G` versus 100K-pair PLM comparison as a matched-budget experiment. The separate 100K-pair comparison in Appendix F.3 controls the amount of supervised training data across T5-Small, SeqDiffuSeq, and `DLM4G`-2.o.

Standard generation metrics. We report BLEU, chrF++, and METEOR for reference agreement, together with BERTScore-F1 and MAUVE for semantic similarity and distributional quality. Every model is evaluated using the same decoded prediction and reference files after a common text-normalization step.

For BLEU, we use the SacreBLEU configuration `case:mixed|tok:13a|smooth:exp`. chrF++ uses character order 6, word order 2, and $\beta = 2$. METEOR uses the standard NLTK English implementation. BERTScore-F1 uses `roberta-large`, and MAUVE uses `gpt2-large` representations.

Text normalization. Before metric computation, model-specific special tokens are removed and all predictions are converted to plain text. For BERT-tokenized diffusion outputs, WordPiece continuations are merged during decoding. We remove padding and sequence-control tokens, normalize repeated whitespace, strip leading and trailing whitespace, and preserve the original text case. The same normalized prediction and reference strings are then passed to BLEU, chrF++, METEOR, BERTScore, and MAUVE. No model-specific metric preprocessing is applied.

Reliability metrics and statistical reporting. FGT and ESR are computed separately as graph-conditioned reliability metrics; their canonicalization and edit protocols are specified in Appendix E.1. Results from trained models are averaged over three independent training runs. The main paper reports standard deviations for the headline reliability metrics, while complete three-seed statistics for the conventional generation metrics are reported in Appendix F.4.

Reproducibility. Code, preprocessing scripts, evaluation scripts, and experiment configurations are available in the anonymous repository. The scheduler conventions used by the reported models are summarized in Table 6, and the corresponding training hyperparameters are given in Table 11.

E Extended Factual Grounding and Edit-Sensitivity Analysis

The main paper introduces FGT for grounding to a fixed graph and ESR for responsiveness to controlled graph edits. This section gives the implementation of the two metrics, reports their independent validation, and provides extended results for larger edit budgets and additional controls.

E.1 Metric Implementation and Canonicalization

For an input graph G and generated sequence S, let U_G denote graph-supported units and U_S the corresponding units realized in the output. For FGT, the units are entity mentions. For ESR, they additionally include canonicalized edited entities, attribute values, and relation labels. Unsupported output units are

$$H_S = U_S \setminus U_G.$$

Unit extraction and canonicalization. We use Qwen2.5-7B-Instruct as a fixed post-processing model to extract and canonicalize entity mentions, attribute values, and relation labels from graph inputs and generated outputs. Recent evaluations report strong structured entity-recognition performance for Qwen2.5-7B, including 93.73 F1 on CoNLL-2003 (Zhan et al., 2026). The same prompt and decoding configuration are applied to outputs from all systems, so FGT and ESR do not use model-specific post-processing. The resulting canonicalized units define U_G and U_S throughout the evaluation. The exact prompt template and evaluation implementation are provided in the anonymous code repository.

The entity-level overlap term and FGT are

$$F_1 = \frac{2|U_G \cap U_S|}{|U_G| + |U_S|}, \tag{37}$$

$$\mathrm{FGT}(G, S) = F_1 \left(1 - \lambda \frac{|H_S|}{\max(|U_S|, 1)}\right). \tag{38}$$

We report $\lambda \in \{0, 0.5, 1.0\}$ and use $\lambda = 0.5$ as the default. The additional unsupported-unit penalty is reported separately from the underlying entity F_1 so that the strength assigned to unsupported generated content can be varied explicitly through λ. FGT should therefore be interpreted as an entity-level grounding metric; broader semantic support is evaluated independently using AlignScore and the human validation in Appendix E.2.

For ESR, let (G, S) and (G', S') denote the original and edited graph-sequence pairs. We define

$$\Delta_G = (U_{G'} \setminus U_G) \cup (U_G \setminus U_{G'}), \tag{39}$$

$$\Delta_T = (U_{S'} \setminus U_S) \cup (U_S \setminus U_{S'}), \tag{40}$$

and

$$\mathrm{ESR} = \frac{|\Delta_G \cap \Delta_T|}{\max(|\Delta_G|, |\Delta_T|, 1)}. \tag{41}$$

The same extraction and canonicalization procedure is used for the original and edited graph–output pairs, so ESR measures whether changes in the graph are reflected in the generated output under a common evaluation pipeline. The controlled perturbation protocol and additional edit-sensitive analyses are described in Appendices E.3–E.4.

E.2 Human and Semantic Validation

We perform a lightweight validation on 500 held-out examples. For factual grounding, human evaluators judge whether the information in the generated sequence is supported by the input graph. For edit sensitivity, they compare the original and edited graph-output pairs and judge whether the intended change is reflected in the output while unrelated content is preserved. AlignScore provides an additional semantic grounding assessment that is independent of the FGT construction.

We report Spearman correlation between FGT or ESR and the corresponding independent assessment. The purpose of this analysis is to check whether the automatic metrics agree with independent judgments, rather than to introduce a separate human-evaluation benchmark.

The correlations indicate that FGT agrees with both human grounding judgments and an external semantic metric, while ESR agrees with human judgments of whether controlled graph edits are reflected in the output. FGT and ESR remain complementary: FGT evaluates grounding to a fixed graph, whereas ESR evaluates response to a controlled graph intervention.

Table 14: **Independent validation of FGT and ESR on 500 held-out examples.**

Metric	Independent assessment	Spearman ρ
FGT@0.5	Human factual-grounding judgment	**0.78**
FGT@0.5	AlignScore	**0.83**
ESR	Human edit-sensitivity judgment	**0.69**

Controlled-edit example.
Original: `[HEAD] RoboTech [REL] located_in [TAIL] Boston`
Edited: `[HEAD] RoboTech [REL] located_in [TAIL] Silicon_Valley`
All other graph triples remain unchanged.

Figure 6: Schematic example of the controlled graph edits used for ESR.

E.3 Controlled Perturbation Construction

We evaluate two edit types. *Value edits* replace an entity, year, location, or attribute value while keeping the surrounding relation fixed. *Relation edits* change the predicate or edge type while retaining the participating entities. The main paper reports one or two value edits per graph and one relation edit. The extended evaluation additionally considers $k \in \{3, 4\}$ simultaneous value edits.

A controlled edit changes only the selected graph unit, while the remaining triples are kept fixed. The paired output comparison therefore tests whether the edited fact changes while unrelated generated content remains stable.

E.4 Multi-Edit and Relation-Edit Results

Table 15 extends the main ESR results to larger edit budgets. We report value-edit ESR for $k \in \{1, 2, 3, 4\}$ together with relation-edit ESR on WikiOFGraph and GenWiki.

Relation edits are generally harder than single value edits because they change the predicate semantics while preserving the participating entities. Increasing the number of simultaneous value edits also reduces edit sensitivity, although the degradation is gradual across both datasets.

E.5 TekGEN Results

The main paper reports factual and edit-sensitive results for WikiOFGraph and GenWiki and reserves TekGEN for the appendix. Table 16 reports the corresponding FGT, AlignScore, multi-edit ESR, and relation-edit ESR results.

E.6 Paired-Noise Evaluation and No-Edit Sampling Control

Diffusion sampling can produce different outputs for the same conditioning graph when independent sampling randomness is used. We therefore measure a no-edit sampling control on WikiOFGraph using `DLM4G-2.0` and separate this source of variation from the controlled-edit evaluation.

For the no-edit control, we generate two outputs from the same graph $\mathbf{G}$ using independent sampling randomness. We define the output-change rate as the fraction of pairs for which the canonicalized output-unit sets differ, $\Delta\mathbf{T} \neq \varnothing$.

For controlled graph edits, the original and edited graphs use exactly the same sampling randomness throughout reverse denoising. Pairing the random streams removes variation caused only by stochastic sampling, so differences between the paired outputs can be attributed more directly to the graph intervention.

As an implementation sanity check, applying the paired protocol to an identical graph pair $(\mathbf{G}, \mathbf{G})$ must produce identical trajectories when the model is evaluated deterministically and the complete random stream is shared. We therefore use this identity case to verify the paired-sampling implementation rather than treating it as an empirical stochastic floor.

Table 15: **Extended edit-sensitivity results.** Values are means over the evaluation runs; standard deviations are shown as bold maroon subscripts.

Model	$k = 1$	$k = 2$	$k = 3$	$k = 4$	Relation
WikiOFGraph					
GPT-2 (B)	0.64	$0.55_{\pm 0.006}$	$0.50_{\pm 0.008}$	$0.46_{\pm 0.006}$	$0.39_{\pm 0.003}$
T5 (S)	0.63	$0.45_{\pm 0.004}$	$0.42_{\pm 0.006}$	$0.40_{\pm 0.005}$	$0.37_{\pm 0.006}$
T5 (L)	0.66	$0.57_{\pm 0.001}$	$0.54_{\pm 0.005}$	$0.52_{\pm 0.007}$	$0.56_{\pm 0.003}$
DiffuSeq	0.66	$0.59_{\pm 0.004}$	$0.56_{\pm 0.006}$	$0.53_{\pm 0.008}$	$0.56_{\pm 0.003}$
SeqDiffuSeq	0.65	$0.58_{\pm 0.011}$	$0.55_{\pm 0.007}$	$0.52_{\pm 0.009}$	$0.56_{\pm 0.004}$
DLM4G-1.o	0.66	$0.58_{\pm 0.005}$	$0.56_{\pm 0.004}$	$0.54_{\pm 0.006}$	$0.56_{\pm 0.004}$
DLM4G-2.o	**0.68**	$\mathbf{0.60}_{\pm 0.005}$	$\mathbf{0.58}_{\pm 0.005}$	$\mathbf{0.56}_{\pm 0.007}$	$\mathbf{0.58}_{\pm 0.002}$
GenWiki					
GPT-2 (B)	0.52	$0.50_{\pm 0.005}$	$0.46_{\pm 0.011}$	$0.42_{\pm 0.007}$	$0.40_{\pm 0.006}$
T5 (S)	0.50	$0.48_{\pm 0.006}$	$0.45_{\pm 0.004}$	$0.44_{\pm 0.006}$	$0.39_{\pm 0.004}$
T5 (L)	0.58	$0.54_{\pm 0.006}$	$0.50_{\pm 0.009}$	$0.51_{\pm 0.004}$	$0.46_{\pm 0.004}$
DiffuSeq	0.59	$0.53_{\pm 0.003}$	$0.50_{\pm 0.006}$	$0.48_{\pm 0.011}$	$0.45_{\pm 0.003}$
SeqDiffuSeq	0.57	$0.54_{\pm 0.004}$	$0.50_{\pm 0.013}$	$0.47_{\pm 0.008}$	$0.44_{\pm 0.006}$
DLM4G-1.o	0.57	$0.52_{\pm 0.006}$	$0.50_{\pm 0.002}$	$0.49_{\pm 0.004}$	$0.46_{\pm 0.005}$
DLM4G-2.o	**0.59**	$\mathbf{0.55}_{\pm 0.006}$	$\mathbf{0.53}_{\pm 0.007}$	$\mathbf{0.53}_{\pm 0.015}$	$\mathbf{0.48}_{\pm 0.004}$

Table 16: **Factual grounding and edit sensitivity on TekGEN.** Values report mean$_{\pm \text{std}}$ over three runs. FGT evaluates grounding on the original graph, while ESR evaluates controlled value and relation edits.

Model	FGT@0.5	AlignScore	$k = 1$	$k = 2$	$k = 3$	$k = 4$	Relation
GPT-2 (B)	$0.46_{\pm 0.005}$	$0.55_{\pm 0.004}$	$0.50_{\pm 0.006}$	$0.47_{\pm 0.006}$	$0.44_{\pm 0.009}$	$0.41_{\pm 0.007}$	$0.38_{\pm 0.005}$
T5 (S)	$0.51_{\pm 0.004}$	$0.59_{\pm 0.005}$	$0.48_{\pm 0.005}$	$0.46_{\pm 0.007}$	$0.43_{\pm 0.005}$	$0.42_{\pm 0.006}$	$0.37_{\pm 0.004}$
T5 (L)	$0.59_{\pm 0.005}$	$0.65_{\pm 0.004}$	$0.56_{\pm 0.006}$	$0.52_{\pm 0.005}$	$0.49_{\pm 0.008}$	$0.48_{\pm 0.006}$	$0.44_{\pm 0.005}$
DiffuSeq	$0.65_{\pm 0.006}$	$0.70_{\pm 0.003}$	$0.57_{\pm 0.004}$	$0.51_{\pm 0.005}$	$0.48_{\pm 0.007}$	$0.46_{\pm 0.010}$	$0.43_{\pm 0.004}$
SeqDiffuSeq	$0.63_{\pm 0.005}$	$0.71_{\pm 0.006}$	$0.55_{\pm 0.005}$	$0.52_{\pm 0.006}$	$0.48_{\pm 0.011}$	$0.45_{\pm 0.007}$	$0.42_{\pm 0.005}$
DLM4G-1.o	$0.65_{\pm 0.006}$	$0.73_{\pm 0.005}$	$0.55_{\pm 0.006}$	$0.50_{\pm 0.005}$	$0.48_{\pm 0.004}$	$0.47_{\pm 0.006}$	$0.44_{\pm 0.005}$
DLM4G-2.o	$\mathbf{0.69}_{\pm 0.004}$	$\mathbf{0.75}_{\pm 0.003}$	$\mathbf{0.56}_{\pm 0.005}$	$\mathbf{0.52}_{\pm 0.006}$	$\mathbf{0.50}_{\pm 0.006}$	$\mathbf{0.49}_{\pm 0.009}$	$\mathbf{0.46}_{\pm 0.004}$

ESR is not reported for the no-edit control because no graph intervention is present. Under independent sampling, the same graph produces a different canonicalized output in 17% of pairs, showing that output differences alone cannot be interpreted as edit responsiveness.

In the ESR evaluation, the original and edited graphs instead share the same sampling randomness. Under this paired protocol, controlled value and relation edits produce output changes in 0.79 and 0.72 of the evaluated pairs, respectively, together with ESR scores of 0.68 and 0.58. The paired design therefore reduces sampling variation when measuring whether a controlled graph change is reflected in the generated output.

E.7 Target-Edit Success and Collateral Changes

ESR measures whether graph edits are reflected in the generated output, but a local edit should also preserve information that is unrelated to the intervention. We therefore analyze DLM4G-2.o on WikiOFGraph using three complementary quantities.

Target-edit realization is the fraction of edited graph units correctly realized in the edited output. Unchanged-unit preservation is the fraction of graph-supported units unaffected by the intervention

Table 17: **No-edit sampling control and paired-edit evaluation for DLM4G-2.o on WikiOFGraph.** The no-edit row uses independent sampling randomness, whereas controlled edits use the same randomness for the original and edited graph.

Graph pair	Sampling noise	Output-change rate	ESR
$\mathbf{G}, \mathbf{G}$	Independent	0.17	N/A
Value edit ($k = 1$)	Paired	0.79	0.68
Relation edit	Paired	0.72	0.58

that remain unchanged in the paired output. Collateral change rate measures the fraction of changed output units that are not attributable to the intended graph edit.

Table 18: **Target-edit realization and preservation of unrelated content for DLM4G-2.o on WikiOFGraph.**

Setting	Target-edit realization	Unchanged-unit preservation	Collateral change rate
Value edit, $k = 1$	0.76	0.94	0.12
Value edit, $k = 2$	0.70	0.91	0.17
Relation edit	0.65	0.92	0.18

Most intended edits are realized while the majority of unrelated graph-supported content remains unchanged. As the number of value edits increases, target realization decreases and collateral changes increase. Relation edits are also more difficult than single value edits, consistent with the ESR results in Table 15.

F Extended Results and Baseline Comparisons

This section collects generation-quality results omitted from the main paper for space, including semantic metrics, task-specific G2S comparisons, the matched 100K-pair experiment, and complete three-seed statistics for conventional generation metrics.

F.1 Complete Generation-Quality Metrics

In addition to BLEU, chrF++, and METEOR, we report BERTScore-F1 (B-F1) and MAUVE (MVE) to provide complementary semantic and distributional measures of generation quality across all three benchmarks.

F.2 Task-Specific G2S Baselines

We report the compatible task-specific comparisons for GenWiki and TekGEN here. The comparison is useful because these systems are trained specifically for graph verbalization rather than evaluated only in a zero-shot setting.

On GenWiki, DLM4G-2.o improves the conventional metrics over the listed specialized systems and reaches the strongest BERTScore-F1 among the models in this table. On TekGEN, it improves all five reported metrics over Rule-Based and ReGen variants. These comparisons complement the fine-tuned and zero-shot baselines in the main paper.

F.3 Matched 100K-Pair Comparison

To separate model performance from differences in training-data scale, we compare T5-Small, SeqDiffuSeq, and DLM4G-2.o under a matched 100K-pair training budget. All three models are trained using the same number of graph-sequence pairs and evaluated on the same held-out inputs.

Table 19: **Extended generation-quality metrics.** B, CrF++, M, B-F1, and MVE denote BLEU, chrF++, METEOR, BERTScore-F1, and MAUVE, respectively. T5-Large is the fine-tuned PLM reference and GPT-o4-mini is the zero-shot LLM reference.

Dataset	Model	B	CrF++	M	B-F1	MVE
WikiOFGraph						
	T5 (L)	**0.658**	0.807	0.736	0.926	0.980
	GPT-o4-mini	0.648	**0.847**	0.783	0.960	**0.983**
	DLM4G-1.o	0.619	0.823	0.688	0.945	0.966
	DLM4G-2.o	0.654	0.844	**0.791**	**0.963**	0.981
GenWiki						
	T5 (L)	0.361	0.567	0.447	0.812	0.852
	GPT-o4-mini	0.464	0.734	0.471	0.865	0.811
	DLM4G-1.o	0.401	0.663	0.527	0.857	0.841
	DLM4G-2.o	**0.469**	**0.748**	**0.574**	**0.899**	**0.892**
TekGEN						
	T5 (L)	0.199	0.370	0.211	0.789	0.803
	GPT-o4-mini	0.121	0.327	0.277	0.652	0.751
	DLM4G-1.o	0.247	0.493	0.375	0.795	0.781
	DLM4G-2.o	**0.253**	**0.522**	**0.414**	**0.847**	**0.820**

Table 20: **Comparison with task-specific G2S baselines.** B-F1 and MVE denote BERTScore-F1 and MAUVE. DLM4G rows use the final values reported in the main paper.

GenWiki

Model	B	CrF++	M	B-F1	MVE
Rule-Based	0.219	0.360	0.397	0.679	0.822
Direct-Transfer	0.234	0.483	0.332	0.808	0.801
Noisy-Sup.	0.384	0.623	0.414	0.878	**0.901**
DLM4G-1.o	0.401	0.663	0.527	0.857	0.841
DLM4G-2.o	**0.469**	**0.748**	**0.574**	**0.899**	0.892

TekGEN

Model	B	CrF++	M	B-F1	MVE
Rule-Based	0.189	0.309	0.301	0.509	0.672
ReGen-SCST	0.219	0.385	0.223	0.698	0.719
ReGen-CE	0.199	0.372	0.214	0.612	0.701
DLM4G-1.o	0.247	0.493	0.375	0.795	0.781
DLM4G-2.o	**0.253**	**0.522**	**0.414**	**0.847**	**0.820**

F.4 Complete Seed Statistics

For trained models, we report mean performance and standard deviation across three independent training runs. The same evaluation inputs are used across seeds so that the reported variation reflects training variability rather than changes in the evaluation set.

G Conditioning-Signal Analysis and Ablations

This section supports the mechanism tested in Q1 and the component attribution in Q3. We first summarize the behavior of the two conditioning signals and then separate independently trained component variants from same-checkpoint inference interventions.

Table 21: **Matched 100K-pair training-budget comparison.** All models use the same 100K graph-sequence pairs and the same held-out evaluation inputs.

Dataset	Model	BLEU	CrF++	M	FGT@0.5	ESR@2	Relation
WikiOFGraph							
	T5-Small	0.385	0.688	0.471	0.56	0.45	0.37
	SeqDiffuSeq	0.405	0.701	0.500	0.60	0.48	0.43
	DLM4G-2.o	**0.433**	**0.713**	**0.537**	**0.64**	**0.52**	**0.47**
GenWiki							
	T5-Small	0.227	0.495	0.434	0.54	0.48	0.39
	SeqDiffuSeq	0.247	0.510	0.465	0.57	0.50	0.41
	DLM4G-2.o	**0.293**	**0.545**	**0.507**	**0.61**	**0.52**	**0.44**
TekGEN							
	T5-Small	0.189	0.352	0.203	0.51	0.46	0.37
	SeqDiffuSeq	0.185	0.361	0.208	0.52	0.47	0.38
	DLM4G-2.o	**0.191**	**0.397**	**0.237**	**0.56**	**0.49**	**0.41**

Table 22: Complete three-seed generation statistics.

Model	WikiOFGraph			GenWiki			TekGEN		
	B	CrF++	M	B	CrF++	M	B	CrF++	M
GPT-2 (S)	$0.166_{\pm 0.003}$	$0.428_{\pm 0.001}$	$0.487_{\pm 0.005}$	$0.280_{\pm 0.004}$	$0.465_{\pm 0.002}$	$0.435_{\pm 0.005}$	$0.226_{\pm 0.006}$	$0.358_{\pm 0.003}$	$0.208_{\pm 0.003}$
GPT-2 (B)	$0.285_{\pm 0.001}$	$0.572_{\pm 0.003}$	$0.490_{\pm 0.005}$	$0.312_{\pm 0.003}$	$0.470_{\pm 0.005}$	$0.425_{\pm 0.004}$	$0.228_{\pm 0.002}$	$0.366_{\pm 0.004}$	$0.211_{\pm 0.004}$
T5 (S)	$0.385_{\pm 0.003}$	$0.688_{\pm 0.005}$	$0.471_{\pm 0.003}$	$0.227_{\pm 0.004}$	$0.495_{\pm 0.001}$	$0.434_{\pm 0.001}$	$0.189_{\pm 0.003}$	$0.352_{\pm 0.002}$	$0.203_{\pm 0.003}$
T5 (L)	$0.658_{\pm 0.002}$	$0.807_{\pm 0.003}$	$0.736_{\pm 0.001}$	$0.361_{\pm 0.002}$	$0.567_{\pm 0.003}$	$0.447_{\pm 0.005}$	$0.199_{\pm 0.001}$	$0.370_{\pm 0.004}$	$0.211_{\pm 0.003}$
DLM4G-1.o	$0.619_{\pm 0.005}$	$0.823_{\pm 0.007}$	$0.688_{\pm 0.006}$	$0.401_{\pm 0.006}$	$0.663_{\pm 0.005}$	$0.527_{\pm 0.008}$	$0.247_{\pm 0.005}$	$0.493_{\pm 0.007}$	$0.375_{\pm 0.006}$
DLM4G-2.o	$\mathbf{0.654}_{\pm 0.004}$	$\mathbf{0.844}_{\pm 0.005}$	$\mathbf{0.791}_{\pm 0.006}$	$\mathbf{0.469}_{\pm 0.006}$	$\mathbf{0.748}_{\pm 0.006}$	$\mathbf{0.574}_{\pm 0.005}$	$\mathbf{0.253}_{\pm 0.006}$	$\mathbf{0.522}_{\pm 0.006}$	$\mathbf{0.414}_{\pm 0.007}$

G.1 Decoder-to-Graph Attention A

The main paper shows that decoder-to-graph cross-attention becomes more discriminative as reverse denoising proceeds. Table 23 reports the endpoint AUROC values shown in Figure 3(a). The trend is consistent across all three datasets: the attention signal is weak when the state is highly noisy and becomes more informative as the output structure is recovered.

G.2 Conditioning Sensitivity Δ

Conditioning sensitivity asks a different question from attention. Attention measures where the decoder interacts with graph tokens, while Δ measures how much removing the graph changes the denoising prediction. Figure 3(b) in the main paper shows that Δ is consistently larger on graph-supported than unsupported positions across WikiOFGraph, GenWiki, and TekGEN. This separation remains visible across denoising time.

For supported positions, the conditioned and unconditioned predictions differ more because the graph supplies information needed for recovery. For unsupported positions, the prediction is increasingly determined by the denoised sequence itself. This is why A and Δ are used together rather than treating either one as a complete measure of conditioning.

G.3 Cross-Dataset Component Attribution

The component attribution in the main paper uses separately trained variants on WikiOFGraph. To test whether the same ordering holds outside that dataset, we repeat the same training decomposition on GenWiki and TekGEN to test whether the component ordering generalizes beyond WikiOFGraph.

Table 23: **Decoder-to-graph attention across reverse denoising.** Endpoint AUROC values are taken from Q1 in the main paper.

Dataset	Early reverse step	Late reverse step
WikiOFGraph	0.61	**0.83**
GenWiki	0.55	**0.79**
TekGEN	0.53	**0.73**

This is different from the same-checkpoint intervention in Appendix G.4: each row here corresponds to a model trained with the indicated schedule components.

Table 24: **Cross-dataset component attribution.** Each row corresponds to a separately trained schedule variant. Standard deviations are shown as bold maroon subscripts.

Variant	WikiOFGraph				GenWiki				TekGEN			
	B	FGT	ESR@2	Rel.	B	FGT	ESR@2	Rel.	B	FGT	ESR@2	Rel.
Base	$0.60_{\pm 0.003}$	$0.77_{\pm 0.005}$	$0.53_{\pm 0.001}$	$0.52_{\pm 0.001}$	$0.414_{\pm 0.006}$	$0.62_{\pm 0.008}$	$0.44_{\pm 0.007}$	$0.40_{\pm 0.005}$	$0.205_{\pm 0.010}$	$0.58_{\pm 0.012}$	$0.40_{\pm 0.009}$	$0.36_{\pm 0.011}$
$+L$	$0.62_{\pm 0.002}$	$0.79_{\pm 0.002}$	$0.54_{\pm 0.004}$	$0.53_{\pm 0.003}$	$0.438_{\pm 0.004}$	$0.65_{\pm 0.007}$	$0.47_{\pm 0.009}$	$0.42_{\pm 0.006}$	$0.224_{\pm 0.007}$	$0.62_{\pm 0.010}$	$0.43_{\pm 0.013}$	$0.39_{\pm 0.008}$
$+L + A$	$0.65_{\pm 0.003}$	$0.80_{\pm 0.001}$	$0.56_{\pm 0.002}$	$0.53_{\pm 0.003}$	$0.462_{\pm 0.008}$	$0.68_{\pm 0.005}$	$0.50_{\pm 0.006}$	$0.43_{\pm 0.010}$	$0.245_{\pm 0.012}$	$0.64_{\pm 0.008}$	$0.47_{\pm 0.010}$	$0.40_{\pm 0.014}$
$+L + \Delta$	$0.65_{\pm 0.005}$	$0.81_{\pm 0.003}$	$0.58_{\pm 0.003}$	$0.56_{\pm 0.002}$	$0.451_{\pm 0.006}$	$0.69_{\pm 0.009}$	$0.53_{\pm 0.005}$	$0.46_{\pm 0.007}$	$0.232_{\pm 0.009}$	$0.66_{\pm 0.013}$	$0.49_{\pm 0.011}$	$0.44_{\pm 0.007}$
Full $L + A + \Delta$	$\mathbf{0.65}_{\pm 0.002}$	$\mathbf{0.83}_{\pm 0.003}$	$\mathbf{0.60}_{\pm 0.005}$	$\mathbf{0.58}_{\pm 0.002}$	$\mathbf{0.469}_{\pm 0.005}$	$\mathbf{0.71}_{\pm 0.006}$	$\mathbf{0.55}_{\pm 0.007}$	$\mathbf{0.48}_{\pm 0.004}$	$\mathbf{0.253}_{\pm 0.008}$	$\mathbf{0.69}_{\pm 0.010}$	$\mathbf{0.52}_{\pm 0.012}$	$\mathbf{0.46}_{\pm 0.009}$

G.4 Same-Checkpoint Inference-Time Signal Intervention

The training ablation compares separately trained variants that use the same schedule components during training and inference. Here we instead start from the Full checkpoint trained with all three components, $L + A + \Delta$, and modify only the schedule used during inference. The model parameters therefore remain fixed in every row.

For the L-only intervention, we use the learned global adaptive schedule directly, corresponding to $\omega_{t,i} = 1$. For $L + A$, we use $\omega_{t,i} = A_{t,i}$; for $L + \Delta$, we use $\omega_{t,i} = \widehat{\Delta}_{t,i}$; and Full inference uses

$$\omega_{t,i} = A_{t,i} + (1 - A_{t,i})\widehat{\Delta}_{t,i}.$$

All other sampling settings are unchanged. Thus, the first three rows introduce an inference-time mismatch relative to the schedule used to train the Full checkpoint, whereas the final row retains the matched Full configuration.

Table 25: **Same-checkpoint inference-time signal intervention on WikiOFGraph.** All rows use checkpoints trained with the Full $L + A + \Delta$ schedule; only the conditioning signals used during inference are changed.

Inference signals	BLEU	FGT@0.5	ESR@2	Relation
L only	$0.59_{\pm 0.012}$	$0.74_{\pm 0.015}$	$0.49_{\pm 0.013}$	$0.47_{\pm 0.017}$
$L + A$	$0.62_{\pm 0.009}$	$0.78_{\pm 0.012}$	$0.53_{\pm 0.011}$	$0.50_{\pm 0.014}$
$L + \Delta$	$0.61_{\pm 0.011}$	$0.80_{\pm 0.010}$	$0.56_{\pm 0.014}$	$0.54_{\pm 0.012}$
Full $L + A + \Delta$	$\mathbf{0.65}_{\pm 0.002}$	$\mathbf{0.83}_{\pm 0.003}$	$\mathbf{0.60}_{\pm 0.005}$	$\mathbf{0.58}_{\pm 0.002}$

Removing conditioning signals only at inference produces a larger degradation than the corresponding matched train-inference variants in the training ablation. This is expected because the Full checkpoint was optimized using the complete conditioning-aware schedule. The $L + A$ intervention retains more generation quality, whereas $L + \Delta$ retains more of the edit-sensitive behavior. Using both signals restores the matched Full configuration and gives the strongest overall result.

G.5 Sensitivity to the Maximum Schedule Shift κ

The adaptive schedule controls the maximum amount of additional signal retention through κ. When $\kappa = 0$, $\bar{\alpha}_{t,i}^{\text{adapt}} = \bar{\alpha}_t^{\text{base}}$, recovering the base schedule. Increasing κ allows difficult positions to retain signal corresponding to progressively earlier points in the base schedule. We evaluate

$\kappa \in \{0, 50, 100, 200\}$ on WikiOFGraph while keeping the remaining training configuration fixed. This tests both whether adaptive signal retention is useful and whether the method is sensitive to its strength. The default setting used throughout the paper is $\kappa = 100$.

Table 26: **Sensitivity to the maximum schedule shift κ on WikiOFGraph.** $\kappa = 0$ recovers the base schedule and $\kappa = 100$ is the default setting. Values report mean$_{\pm \text{std}}$ over three seeds.

κ	BLEU	FGT@0.5	ESR@2	Relation
0	$0.60_{\pm 0.003}$	$0.77_{\pm 0.005}$	$0.53_{\pm 0.001}$	$0.52_{\pm 0.001}$
50	$0.63_{\pm 0.003}$	$0.81_{\pm 0.004}$	$0.57_{\pm 0.003}$	$0.55_{\pm 0.003}$
100	$\mathbf{0.65}_{\pm 0.002}$	$\mathbf{0.83}_{\pm 0.003}$	$\mathbf{0.60}_{\pm 0.005}$	$\mathbf{0.58}_{\pm 0.002}$
200	$0.65_{\pm 0.003}$	$0.82_{\pm 0.004}$	$0.59_{\pm 0.004}$	$0.57_{\pm 0.003}$

Performance improves consistently from $\kappa = 0$ to $\kappa = 50$ and $\kappa = 100$, showing that retaining additional signal at difficult positions is beneficial. The gains saturate at $\kappa = 100$: increasing the shift to $\kappa = 200$ leaves BLEU unchanged at 0.65 but slightly reduces FGT@0.5, ESR@2, and relation-edit ESR. This behavior suggests that moderate signal retention helps difficult positions remain recoverable during denoising, allowing graph evidence to guide their reconstruction. Excessive retention, however, can make the current noisy state too informative, reducing the need to revise a position using the conditioning graph. This creates greater persistence of existing token content and can reduce responsiveness to graph edits. The results therefore indicate a useful intermediate regime: adaptive retention improves over the base schedule, while increasing it beyond $\kappa = 100$ provides no further benefit.

G.6 Does the Conditioning Gate Need to Track the Graph?

The previous experiment shows that changing the inference schedule of the Full checkpoint can itself reduce performance. We therefore compare two position-wise perturbations of $\omega_{t,i}$ against the same Full-trained checkpoint. Both controls preserve the set of conditioning weights computed for the current graph and noisy state, but alter how these weights are assigned to output positions.

Shuffled. We randomly permute $\omega_{t,i}$ across valid output positions before constructing the reverse schedule. This preserves the distribution, mean, and range of the conditioning weights while breaking their correspondence with the positions for which they were computed.

Rank reversal. We sort the valid output positions according to their conditioning weights and reverse their assignment: the largest weight is assigned to the position with the smallest original weight, the second largest to the second smallest, and so on. Unlike random shuffling, this systematically reverses the position-wise ordering of the gate while preserving exactly the same set of $\omega_{t,i}$ values.

As in Appendix G.4, model parameters remain fixed and all interventions are applied only during inference.

Table 27: **Inference-time conditioning-weight controls on WikiOFGraph.** All rows use the same Full-trained checkpoint. L only removes the position-dependent conditioning gate; shuffled $\omega_{t,i}$ randomly reassigns the same weights across positions; rank reversal systematically reverses their position-wise ordering; and Full retains the original graph-dependent assignment.

Inference setting	BLEU	FGT@0.5	ESR@2	Relation
L only	$0.59_{\pm 0.012}$	$0.74_{\pm 0.015}$	$0.49_{\pm 0.013}$	$0.47_{\pm 0.017}$
Shuffled $\omega_{t,i}$	$0.62_{\pm 0.009}$	$0.76_{\pm 0.014}$	$0.51_{\pm 0.009}$	$0.48_{\pm 0.011}$
Rank-reversed $\omega_{t,i}$	$0.60_{\pm 0.005}$	$0.72_{\pm 0.011}$	$0.46_{\pm 0.009}$	$0.44_{\pm 0.006}$
Full $\omega_{t,i}$	$\mathbf{0.65}_{\pm 0.004}$	$\mathbf{0.83}_{\pm 0.003}$	$\mathbf{0.60}_{\pm 0.005}$	$\mathbf{0.58}_{\pm 0.002}$

The two perturbations show that the position-wise assignment of the conditioning weights matters. Random shuffling preserves the marginal distribution of $\omega_{t,i}$ but reduces FGT@0.5 from 0.83 to 0.76, ESR@2 from 0.60 to 0.51, and relation-edit ESR from 0.58 to 0.48. Rank reversal causes a larger degradation, reducing these metrics to 0.72, 0.46, and 0.44, respectively. Thus, preserving the

set of conditioning weights alone is insufficient: performance is strongest when the weights remain associated with the positions selected by the graph-conditioning signals, and systematically reversing this association further degrades grounding and edit sensitivity.

Because shuffled and rank-reversed inference both modify the schedule used by a Full-trained checkpoint, these controls should not be interpreted as isolating a training-time causal effect. Instead, they provide inference-time evidence that both the presence and the position-wise assignment of the graph-dependent conditioning weights are important for the behavior of the scheduler.

G.7 Classifier-Free Guidance

Conditioning sensitivity uses the difference between conditioned and unconditioned predictions to modulate the token-wise noising schedule. A natural control is classifier-free guidance (CFG), which uses the same two predictions to modify the denoising prediction directly.

For consistency with the sensitivity definition, we write

$$\widehat{\mathbf{z}}_0^{\text{cond}} = \mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t, \mathbf{G}), \tag{42}$$

$$\widehat{\mathbf{z}}_0^{\text{uncond}} = \mathcal{M}_\theta(\mathbf{z}_t, \bar{\boldsymbol{\alpha}}_t, \emptyset), \tag{43}$$

and apply CFG as

$$\widehat{\mathbf{z}}_0^{\text{CFG}} = \widehat{\mathbf{z}}_0^{\text{uncond}} + s\left(\widehat{\mathbf{z}}_0^{\text{cond}} - \widehat{\mathbf{z}}_0^{\text{uncond}}\right), \tag{44}$$

where s is a scalar guidance scale shared across output positions. At $s = 1$, Eq. 44 reduces to the ordinary conditional prediction.

We apply CFG to the separately trained difficulty-only (L) checkpoint and sweep $s \in \{1.0, 1.5, 2.0, 3.0, 5.0\}$. The guidance scale is selected using FGT@0.5 on the 100K-example validation set held out from the WikiOFGraph training partition. Both CFG and the conditioning-sensitivity term Δ require conditioned and unconditioned model evaluations at each reverse step, giving them the same denoiser-evaluation count per reverse step.

Table 28: **Classifier-free guidance sweep on WikiOFGraph.** CFG is applied to the separately trained difficulty-only checkpoint. The guidance scale is selected using FGT@0.5 on the 100K-example validation set. Full DLM4G is shown for reference.

Setting	Scale s	BLEU	FGT@0.5	ESR@2	Relation
Difficulty only (L)	1.0	$0.620_{\pm 0.004}$	$0.790_{\pm 0.005}$	$0.540_{\pm 0.006}$	$0.530_{\pm 0.004}$
L+CFG	**1.5**	$0.620_{\pm 0.005}$	$0.810_{\pm 0.004}$	$0.560_{\pm 0.005}$	$0.550_{\pm 0.006}$
L+CFG	2.0	$0.612_{\pm 0.006}$	$0.808_{\pm 0.006}$	$0.568_{\pm 0.007}$	$0.556_{\pm 0.005}$
L+CFG	3.0	$0.598_{\pm 0.009}$	$0.801_{\pm 0.008}$	$0.565_{\pm 0.009}$	$0.548_{\pm 0.008}$
L+CFG	5.0	$0.561_{\pm 0.012}$	$0.772_{\pm 0.011}$	$0.543_{\pm 0.010}$	$0.521_{\pm 0.013}$
Full $L + A + \Delta$	N/A	$\mathbf{0.650}_{\pm 0.002}$	$\mathbf{0.830}_{\pm 0.003}$	$\mathbf{0.600}_{\pm 0.005}$	$\mathbf{0.580}_{\pm 0.002}$

Moderate CFG improves grounding and edit sensitivity over the difficulty-only checkpoint, while stronger guidance begins to reduce generation quality. Across the evaluated guidance scales, CFG does not recover the performance of Full DLM4G. Because CFG is applied to the separately trained difficulty-only checkpoint whereas Full DLM4G is trained with conditioning-aware schedule modulation, this comparison does not isolate the effect of prediction guidance versus schedule control. Instead, it shows that direct CFG applied to the difficulty-only checkpoint does not recover the gains of the Full conditioning-aware training and inference procedure.

H Computational Cost and Accelerated Inference

This section separates the computational cost of the conditioning signals from the quality–latency trade-off obtained by reducing the number of reverse steps. Unless stated otherwise, inference timing uses a batch size of 50 on a single NVIDIA V100 with the same maximum output length across compared settings.

Table 29: **Component-wise computational cost on WikiOFGraph.** Training cost counts model evaluations required to construct the training state and compute the denoising objective. Inference NFE is reported per reverse step. Batch-50 time corresponds to 2000-step inference on one NVIDIA V100.

Variant	Train evals / step	Inference NFE / step	Batch-50 time
Base	1	1	$\sim$ 45s
$+L$	1	1	$\sim$ 45s
$+L + A$	2	1	$\sim$ 46s
$+L + \Delta$	3	2	$\sim$ 88s
Full $L + A + \Delta$	3	2	89s

H.1 Training-Time Overhead

The global difficulty-based schedule L introduces little additional per-step computation because its statistics are accumulated during training and the schedule is updated only once every 20K optimization steps.

The conditioning signals require additional probe evaluations before the final training state is constructed. For $L + A$, one conditional probe evaluation is used to obtain decoder-to-graph attention. For $L + \Delta$, both conditioned and unconditioned probe predictions are required. In the Full model, A is read from the same conditioned probe evaluation already used to compute Δ, so adding A on top of $L + \Delta$ does not require another model evaluation.

After the schedule is constructed, the training state is re-noised and one additional model evaluation is used for the denoising objective. Base and L therefore require one model evaluation per optimization step, $L + A$ requires two, and both $L + \Delta$ and Full require three.

H.2 Inference-Time Component Cost

At inference, the global adaptive schedule is fixed. Attention A is obtained from the conditional denoising pass and therefore does not require an additional model evaluation. Conditioning sensitivity Δ requires one additional unconditional prediction at each reverse step. Consequently, Base, L, and $L + A$ require one model evaluation per reverse step, whereas $L + \Delta$ and Full require two.

The dominant inference-time overhead therefore comes from Δ. Attention adds no additional model evaluation because it is read from the conditional forward pass already required for denoising. The approximately twofold difference between the one- and two-NFE settings is also consistent with the observed Full-model latency.

H.3 Compute-Matched CFG Comparison

Classifier-free guidance and conditioning sensitivity both require conditioned and unconditioned predictions at each reverse step. They therefore have the same model-evaluation count of two evaluations per step. The distinction is how the two predictions are used. CFG applies a single scalar guidance scale to the denoising prediction, whereas `DLM4G` uses their difference to construct a position-dependent schedule through $\Delta_{t,i}$.

The guidance-scale sweep in Table 28 shows that moderate CFG improves over difficulty-only adaptation, but no tested guidance scale recovers the Full scheduler on the grounding and edit-sensitivity metrics. This provides a compute-matched inference comparison showing that CFG applied to the difficulty-only checkpoint does not recover the performance of the Full training-and-inference procedure.

H.4 Accelerated Sampling

We evaluate the standard ancestral reverse process together with reduced-step DDIM, DPM-Solver++, and mixed-space decoding. These accelerated settings modify only inference; the trained model parameters and the global adaptive schedule remain unchanged.

Table 30: **Inference efficiency on WikiOFGraph.** Time is end-to-end wall-clock inference time for a batch of 50 on one NVIDIA V100. Speedup is measured relative to 2000-step DiffuSeq.

Model	Inference method	Params	Steps	Time	Speedup	BLEU
Autoregressive baselines						
GPT-2 (B)	AR decoding	355M	64	<5s	–	0.285
T5 (S)	AR decoding	60M	64	<5s	–	0.385
Diffusion baselines						
DiffuSeq	DDPM	91M	2000	317s	1.0×	0.628
SeqDiffuSeq	DDPM	50M	2000	89s	3.6×	0.616
DLM4G-2.o full denoising						
DLM4G-2.o	Ancestral (DDPM)	63M	2000	89s	3.6×	**0.654**
DLM4G-2.o reduced-step DDIM						
DLM4G-2.o	DDIM	63M	1000	45s	7.0×	0.551
DLM4G-2.o	DDIM	63M	500	23s	13.8×	0.365
DLM4G-2.o	DDIM	63M	100	5s	63.4×	0.312
DLM4G-2.o accelerated decoding						
DLM4G-2.o	DPM-Solver++	63M	50	<5s	> 63×	0.418
DLM4G-2.o	DPM-Solver++	63M	10	<5s	> 63×	0.405
DLM4G-2.o	Mixed-space	63M	50	<5s	> 63×	**0.468**
DLM4G-2.o	Mixed-space	63M	10	<5s	> 63×	0.434

Reduced-step sampling. The Full 2000-step result uses the stochastic ancestral update described in Appendix C.4. For reduced-step DDIM, we select a smaller set of reverse timesteps and use deterministic transitions between the retained states.

We additionally evaluate DPM-Solver++ as an empirical accelerated sampler. At each retained model-evaluation state, the conditioning signals are recomputed from the current noisy representation and the corresponding local effective schedule is constructed as in Appendix C.4. The solver then advances between the selected reverse-time states using substantially fewer model evaluations.

Standard DPM-Solver++ is derived for a diffusion process with a prescribed noise schedule. In DLM4G, the effective schedule is additionally token- and state-dependent through $\omega_{t,i}$. We therefore use DPM-Solver++ here as an empirical acceleration procedure and do not rely on its fixed-schedule ODE guarantees for the complete adaptive trajectory.

Mixed-space decoding. Following the mixed-space decoding strategy of DiffuSeq-v2 (Gong et al., 2023b), continuous reverse updates are interleaved with projection through the discrete vocabulary space. At selected reverse steps, the predicted clean representation is mapped to token representations and re-embedded before the next continuous reverse update. This periodically anchors the sparse reverse trajectory to valid token representations when many intermediate diffusion steps are skipped. The conditioning-aware schedule itself is unchanged; only the representation used between sparse reverse updates is modified.

Table 30 reports the resulting quality–latency trade-off on WikiOFGraph. Full 2000-step ancestral sampling reaches 0.654 BLEU in 89s per batch of 50. Reduced-step DDIM substantially lowers latency but degrades under aggressive step reduction. DPM-Solver++ provides a stronger low-step operating point, while mixed-space decoding gives the highest BLEU among the accelerated configurations.

All times are end-to-end measurements under the same hardware, batch size, and maximum output length. Cross-system wall-clock comparisons also reflect implementation differences between the corresponding codebases and therefore should not be interpreted solely from parameter count or NFE. The NFE analysis in Table 29 is most directly comparable across the DLM4G variants, which share the same implementation.

Within DLM4G, the longer DDIM Song et al. (2021) runs scale approximately with the number of reverse steps. At very small step counts, fixed graph encoding, input/output processing, vocabulary projection and rounding, and framework overhead become a larger fraction of end-to-end latency. Consequently, the sub-5s configurations do not scale linearly with reverse-step count.

Full ancestral denoising provides the highest generation quality. Among the accelerated settings, mixed-space decoding gives the strongest result, reaching 0.468 BLEU in under 5s with 50 reverse steps. These accelerated procedures modify the reverse-time inference procedure but do not modify the trained model or the learned global adaptive schedule.