FRAGFLEX: FRAGMENT-BASED FLEXIBLE MOLECULAR GENERATION FOR EXPLORING VAST CHEMICAL SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Drug discovery requires efficient exploration of an astronomically large chemical space. Molecular graph diffusion models offer a promising approach to this task, but many require graph size to be fixed before generation. Although atom-level insertion and deletion can enable flexible-size generation, jointly modeling molecular size, atom types, and bonds while maintaining chemical validity remains challenging. We introduce FragFlex, a flexible-length discrete diffusion model that performs node insertion and deletion at the fragment level. During denoising, FragFlex jointly learns the size, fragment identities, and inter-fragment connectivity of the fragment graph. A neural attachment-site predictor and valence-aware assembler then convert the generated fragment graph into an atom-level molecule. In large-scale experiments generating 10 million molecules, FragFlex achieves near-perfect chemical validity while maintaining high uniqueness, demonstrating a strong balance between the two.

1 INTRODUCTION

Drug discovery requires exploring an astronomically large chemical space. Estimates of the size of drug-like chemical space vary substantially with the underlying molecular constraints and enumeration strategies, ranging from approximately 10^{33} molecules (Polishchuk et al., 2013) to 10^{60} molecules (Bohacek et al., 1996). Efficiently searching this space is therefore key to reducing the computational and experimental costs of finding promising drug candidates (Schneider et al., 2020; Thomas et al., 2026).

De novo molecular generation offers a promising strategy for exploring chemical space by directly proposing novel molecular structures. Deep generative models can learn molecular distributions from existing compound datasets and generate new molecules. Notably, denoising diffusion models have driven substantial progress across image generation (Ho et al., 2020), discrete text generation (Austin et al., 2021; Lou et al., 2023), and molecular graph generation (Vignac et al., 2022).

However, many existing molecular graph diffusion models keep the number of nodes fixed throughout denoising, requiring molecular size to be specified before generation begins, which has been shown to negatively affect the quality of generated molecules (Dunn & Koes, 2026). Attempts to overcome fixed-length generation are not unique to molecular graphs: masked diffusion models (MDMs) (Sahoo et al., 2024) have been extended to support flexible-length text generation through mask-token insertion (Kim et al., 2026). For molecular graphs, Ninniri et al. proposed GrIDDD (Ninniri et al., 2025), allowing atom nodes to be inserted and deleted during denoising. Despite this flexibility, maintaining high chemical validity remains challenging when molecular graph size is allowed to vary through atom-level insertion and deletion. This difficulty arises because the model must jointly learn molecular size, atom types, and bond assignments while satisfying strict chemical constraints, such as atomic valence.

Fragment-based molecular generation offers a natural way to mitigate this challenge. Instead of constructing every ring and functional group atom by atom, these methods assemble reusable substructures, reducing invalid intermediates (Jin et al., 2018; 2020). Fragment-level diffusion and flow models further improve efficiency and scalability by denoising a coarser graph (Lee et al., 2026). In particular, Levy and Rector-Brooks proposed FragDiffusion (Levy & Rector-Brooks, 2023),

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.

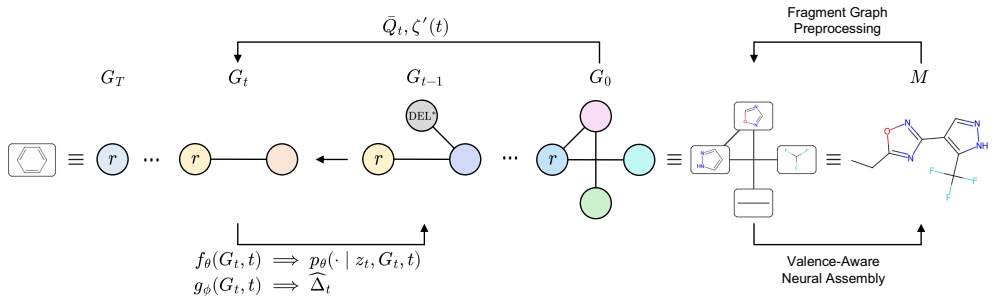

Figure 1: Overview of FragFlex. An atom-level molecule M is preprocessed into a clean fragment graph G_0. Forward diffusion applies categorical noise $\bar{Q}_t$ and structural deletion $\zeta'(t)$ while protecting a root fragment r, yielding the single-fragment terminal state G_T. During sampling, g_ϕ predicts the insertion count $\widehat{\Delta}_t$, and f_θ predicts the fragment types and edge connections of the added DEL* placeholders. At the final denoising step, predicted attachment sites guide the valence-aware neural assembly of the fragment graph into an atom-level molecule. The $\equiv$ symbol indicates representations with the same fragment-level connectivity.

extending discrete graph diffusion from atoms to fragments. Nevertheless, the two capabilities have largely developed separately: flexible-size diffusion typically operates on atoms or generic tokens, whereas representative fragment diffusion models initialize a fragment graph whose size is sampled before denoising.

To bridge this gap, we introduce FragFlex, a flexible-length discrete diffusion model that performs node insertion and deletion at the fragment level. The model jointly learns fragment count, fragment identities, and inter-fragment connectivity while dynamically adjusting the graph size during denoising. This design allows molecular size and composition to co-evolve within a single generative trajectory while retaining the chemical inductive bias of fragment-based generation. Finally, a neural attachment-site predictor and valence-aware assembler convert the generated fragment graph into an atom-level molecule. In large-scale experiments generating 10 million molecules, FragFlex maintains both high uniqueness and chemical validity, showing a favorable trade-off between the two.

Our main contributions are as follows:

- We introduce FragFlex, a discrete diffusion model that enables flexible molecular-size generation through fragment-level node insertion and deletion.
- We develop a neural attachment-site predictor and valence-aware assembler for converting fragment graphs into valid molecules.
- In experiments generating 10 million molecules, FragFlex achieves high chemical validity and a favorable uniqueness–validity trade-off.

2 Related Work

2.1 De Novo Molecular Generation

De novo molecular generation aims to learn distributions over molecular structures for unconditional or property-guided sampling (Pang et al., 2024). Early neural approaches often represented molecules as SMILES strings and applied recurrent models or variational autoencoders (Gómez-Bombarelli et al., 2018; Segler et al., 2018; Kusner et al., 2017). Graph-based methods instead operate directly on atoms and bonds, using variational autoencoders, autoregressive models, and normalizing flows (Simonovsky & Komodakis, 2018; Liu et al., 2018; Shi et al., 2020; Luo et al., 2021).

More recently, score-based and diffusion models have formulated molecular generation as iterative denoising of molecular graphs (Jo et al., 2022; 2023). In particular, discrete diffusion methods such as D3PM (Austin et al., 2021) enable diffusion over categorical variables, providing a natural framework for molecular graphs with discrete atom and bond types. DiGress applies this framework to graph

generation using categorical transition kernels and a permutation-equivariant graph transformer (Vignac et al., 2022).

2.2 Flexible-Length Molecular Generation

Autoregressive models can naturally generate molecules of varying sizes through a stopping decision, whereas many graph diffusion models sample the graph size beforehand and keep it fixed during denoising. DiGress, for instance, samples the graph size from the training-set size distribution before denoising and keeps it fixed throughout generation (Vignac et al., 2022).

Recent methods instead allow graph size to change during generation. IFH uses insertion, filling, and halting operations (Cognolato et al., 2024), while GrIDDD deletes nodes during forward diffusion and inserts them during reverse generation (Ninniri et al., 2025). In the sequence domain, FlexMDM enables flexible-length diffusion through mask-token insertion (Kim et al., 2026).

FragFlex applies this size-adaptive principle at the fragment level. Starting from a single root fragment, it predicts fragment insertions and jointly denoises fragment identities and connectivity.

2.3 Fragment-Based Molecular Generation

Fragment-based molecular generative models construct molecules from reusable molecular substructures, or fragments, rather than individual atoms. This coarse representation can reduce the number of generative decisions while incorporating chemically meaningful substructures into the generation process (Voloboev, 2024; Jinsong et al., 2024). JT-VAE generates a junction tree over molecular substructures before assembling the molecular graph (Jin et al., 2018), while HierVAE extends this approach with hierarchical motifs and coarse-to-fine generation (Jin et al., 2020).

Fragment-level denoising methods instead operate directly on coarse fragment graphs. FragDiffusion applies discrete diffusion to BRICS-derived fragment graphs whose size is selected before denoising and reconstructs atom-level molecules using attachment modes stored in a precomputed lookup table (Levy & Rector-Brooks, 2023). FragFM combines fragment-level discrete flow matching with coarse-to-fine atom-level reconstruction (Lee et al., 2026).

FragFlex combines fragment-level generation with size-adaptive diffusion. Unlike prior fragment-level denoising methods, it allows the number of fragments to change during generation while jointly modeling fragment identities and connectivity. It further predicts attachment sites directly and uses a valence-aware assembler to reconstruct the final atom-level molecule.

3 Methods

3.1 Overview and Problem Formulation

Overview. FragFlex generates a molecule in two stages (Figure 1). First, starting from a single root fragment, a discrete diffusion model generates a coarse graph by jointly modeling fragment identities, connectivity, and graph size (Figure 2). Second, a neural attachment-site predictor and a valence-aware assembler convert this fragment graph into an atom-level molecular graph. Our model combines FragDiffusion (Levy & Rector-Brooks, 2023), a discrete diffusion model over fragment graphs, with GrIDDD (Ninniri et al., 2025), an atom-level diffusion model that enables variable graph sizes through node insertions and deletions.

Problem definition. Let $\mathcal{V} = \{F_1, \ldots, F_K\}$ be a fixed vocabulary of BRICS-derived fragment types extracted from the training molecules (Degen et al., 2008). Each F_k specifies an atom-and-bond substructure. Note that a fragment may correspond to either a single heavy atom or a molecular substructure containing multiple heavy atoms. We retain the $K = 100$ most frequent fragment types. Examples of the fragment vocabulary used by FragFlex are provided in Appendix E. We represent an atom-level molecule M by a clean fragment graph $G_0 = (X_0, E_0, S_0)$ with N fragments. The node label $X_{0,i} \in \{1, \ldots, K\}$ identifies the fragment at node i, and $E_{0,ij} \in \{0, 1\}$ indicates whether fragments i and j are connected. For each connected fragment pair (i, j), $S_{0,ij}$ and $S_{0,ji}$ denote the local indices of the attachment atoms in fragments i and j, respectively, with $S_{0,ij} \in \{0, \ldots, A_i - 1\}$, where A_i is the number of atoms in fragment i. Our goal is to jointly

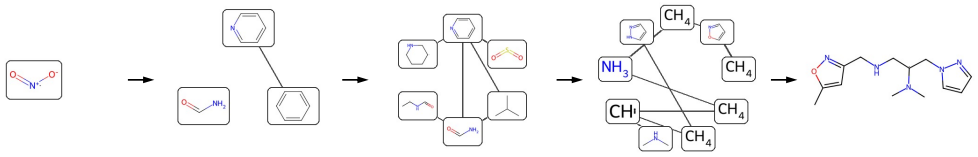

Figure 2: FragFlex sampling procedure. Starting from a single root fragment, the fragment graph progressively grows through node insertion during reverse diffusion while fragment identities and connectivity are denoised. The final arrow denotes the neural assembly step that converts the generated fragment graph into an atom-level molecular graph using predicted attachment sites and valence-aware constraints.

model fragment identities, topology, attachment sites, and graph size. Unlike FragDiffusion (Levy & Rector-Brooks, 2023), which assembles molecules using a fragment-pair lookup table precomputed from attachment frequencies in the training data, FragFlex represents fragment connectivity with binary edges and directly predicts the corresponding attachment atoms.

3.2 Flexible-Length Fragment Diffusion Training

Given a clean fragment graph $G_0 = (X_0, E_0, S_0)$ with node set $V_0 = \{1, \dots, N\}$, we sample $t \sim \mathrm{Unif}\{1, \dots, T\}$ and a root $r \sim \mathrm{Unif}(V_0)$. Each non-root fragment is assigned a deletion time $\tau_i \sim \mathrm{Cat}(\zeta')$, while we set $\tau_r = T + 1$ so that the root is never deleted. Following GrIDDD (Ninniri et al., 2025), $\zeta'(t)$ is a normalized logistic event density. Unlike GrIDDD, which centers insertion and deletion activity around the midpoint of the diffusion process, FragFlex uses $s_{\mathrm{p}} = 0$, concentrating graph growth near the beginning of reverse sampling. Here, s_{p} denotes the reverse step around which fragment insertions are most likely.

At time t, fragments with $\tau_i < t$ are removed and boundary fragments with $\tau_i = t$ are represented by the transient state DEL*; their incident edges are marked identically. The retained fragment and edge labels are corrupted using the empirical-marginal kernel

$$\bar{Q}_t(m) = \bar{\alpha}_t I + (1 - \bar{\alpha}_t)\mathbf{1}m^\top, \tag{1}$$

where m is the corresponding training marginal and $\bar{\alpha}_t$ follows a cosine schedule. At $t = T$, the terminal state therefore contains one protected fragment drawn from the empirical fragment marginal.

An edge-aware graph transformer (Yun et al., 2019) f_θ serves as the main denoiser and predicts clean fragment logits Z^X, binary edge logits Z^E, and directed attachment-site logits Z^S from G_t. Its encoder combines time and root embeddings with the cycle features of DiGress (Vignac et al., 2022). A separate count model g_ϕ receives G_t after removing the DEL* fragments and predicts their number Δ_t. We optimize the joint objective

$$\mathcal{L} = \mathcal{L}_X + \lambda_E \mathcal{L}_E + \lambda_\Delta \mathcal{L}_\Delta + \lambda_S \mathcal{L}_S, \tag{2}$$

where all terms are cross-entropies. These four terms supervise fragment identities, fragment edges, deletion counts, and attachment sites, respectively.

Algorithm 1 summarizes the training procedure. Further details are provided in Appendix A.3.

3.3 Fragment Graph Sampling

Starting from a single root fragment at $t = T$, the reverse process constructs a fragment graph until $t = 1$. At each reverse step, g_ϕ predicts how many DEL* placeholders to append. These newly inserted nodes are temporary placeholders whose fragment identities and incident edges are subsequently resolved by the denoiser f_θ. The denoiser outputs clean node and edge logits, which parameterize the reverse distribution

$$p_\theta(z_{t-1} \mid z_t, G_t, t) = \sum_{z_0} q(z_{t-1} \mid z_t, z_0) p_\theta(z_0 \mid G_t, t). \tag{3}$$

Algorithm 1 FragFlex training

Require: Dataset of clean fragment graphs $G_0 = (X_0, E_0, S_0)$
repeat
  Sample a minibatch and $t \sim \mathrm{Unif}\{1, \dots, T\}$
  Select a root $r \sim \mathrm{Unif}(V_0)$
  Sample $\tau_i \sim \mathrm{Cat}(\zeta')$ for each $i \neq r$; set $\tau_r = T + 1$
  $V_t \leftarrow \{i \in V_0 : \tau_i \geq t\}$
  $B_t \leftarrow \{i \in V_0 \setminus \{r\} : \tau_i = t\}$
  Remove nodes with $\tau_i < t$ and their incident edges
  Mark nodes in B_t and their incident edges as DEL*
  Corrupt all other retained node and edge labels with $\bar{Q}_t$ to obtain G_t
  Form the count-network input $G_t^- \leftarrow G_t \setminus B_t$
  Set the count target $\Delta_t \leftarrow |B_t|$
  $(Z^X, Z^E, Z^S) \leftarrow f_\theta(G_t, t)$
  $Z^\Delta \leftarrow g_\phi(G_t^-, t)$
  $\mathcal{L}_X \leftarrow \mathrm{CE}(Z^X, X_0)$ over retained nodes
  $\mathcal{L}_E \leftarrow \mathrm{CE}(Z^E, E_0)$ over retained node pairs
  Mask $Z^S_{ij,a}$ for $a \geq A_i$
  $\mathcal{L}_S \leftarrow \mathrm{CE}(Z^S, S_0)$ over directed endpoints of clean fragment edges
  $\mathcal{L}_\Delta \leftarrow \mathrm{CE}(Z^\Delta, \Delta_t)$
  $\mathcal{L} \leftarrow \mathcal{L}_X + \lambda_E \mathcal{L}_E + \lambda_\Delta \mathcal{L}_\Delta + \lambda_S \mathcal{L}_S$
  Update θ and ϕ with AdamW
until N_{epoch} epochs have been completed

Algorithm 2 FragFlex sampling

Require: Trained f_θ and g_ϕ; empirical fragment marginal; $N_{\max}$
Sample one root fragment from the empirical fragment marginal to form G_T
for $t = T, T-1, \dots, 1$ **do**
  Sample $\widehat{\Delta}_t$ from $g_\phi(G_t^-, t)$ and clip to the remaining capacity
  Append $\widehat{\Delta}_t$ nodes and mark them and their incident edges as DEL*
  Predict clean node and edge logits with $f_\theta(G_t, t)$
  if $t = 1$ **then**
    Also compute and retain the attachment-site logits Z^S
  Combine clean predictions with the exact categorical posterior $p_\theta(z_{t-1} \mid z_t, G_t, t)$
  Sample X_{t-1} and the upper triangle of E_{t-1}; symmetrize E_{t-1}
  Connect each newly resolved node that has no neighbor in the pre-insertion graph
  if $t - 1 = 0$ and the fragment graph is disconnected **then**
    Add highest-confidence cross-component edges until the graph is connected
Mask padded site positions and convert the retained Z^S to $\widehat{P}^S$
return $(\widehat{X}_0, \widehat{E}_0, \widehat{P}^S)$

Since attachment sites are needed only for final molecular assembly, we evaluate the site head only at $t = 1$, avoiding unnecessary computation during reverse diffusion. The resulting probabilities $\widehat{P}^S$ are passed to the atom-level assembler.

Algorithm 2 summarizes the reverse sampling procedure. Further details are provided in Appendix A.4.

3.4 Valence-Aware Neural Assembly

After masking infeasible attachment atoms and renormalizing $\widehat{P}^S$ to $\widetilde{P}^S$, the assembler considers the top-k atoms for each generated fragment edge (i, j) and scores candidate pairs (a, b) by

$$\ell_{ij}(a, b) = \log \widetilde{P}^S_{ij}(a) + \log \widetilde{P}^S_{ji}(b). \tag{4}$$

Beam search processes the fragment edges sequentially, adding one candidate single bond at each step and discarding duplicate bonds or candidates that violate valence constraints under RDKit (Landrum et al., 2023). After all fragment edges have been processed, the assembler retains a fixed-size pool

of unique, connected, and sanitized molecules and selects the final product uniformly at random. Uniform selection prevents the assembly procedure from always returning the same highest-scoring realization when several chemically valid assemblies are available. Thus, chemical constraints filter the learned site proposals without requiring an attachment lookup table at inference.

Algorithm 3 summarizes the valence-aware neural assembly procedure. Further details are provided in Appendix A.5.

Algorithm 3 Neural top-k valence-aware assembly

Require: Generated $(\widehat{X}_0, \widehat{E}_0, \widehat{P}^S)$; k, B_{beam}, and $P_{\max}$
Extract the generated fragment-edge set $\mathcal{E}$
for all $(i, j) \in \mathcal{E}$ **do**
    Construct and score the top-k candidate pairs $\mathcal{C}_{ij}$
Order edges by increasing $|\mathcal{C}_{ij}|$
Initialize $\mathcal{B}$ with the disconnected RDKit fragments
for all edges e in the selected order **do**
    Expand $\mathcal{B}$ over $\mathcal{C}_e$, retaining only RDKit-valence-safe branches
    if $\mathcal{E}$ contains a cycle **then**
        Add skip branches with penalty
    Rank by more successfully added bonds, fewer skips, a higher site score, and fewer non-top-1 choices
    Retain the best B_{beam} states in $\mathcal{B}$
    if $\mathcal{B} = \varnothing$ **then**
        return failure
Form $\mathcal{P}$ from the first $P_{\max}$ unique, connected, and sanitized products in beam order
if $\mathcal{P} = \varnothing$ **then**
    return failure
Sample $\widehat{M} \sim \text{Unif}(\mathcal{P})$
return $\widehat{M}$

4 Results

4.1 Experimental Setup

The evaluated models use datasets derived from ZINC (Sterling & Irwin, 2015). For each baseline, we retain the default dataset and model configuration specified in its original implementation. FragFlex follows the preprocessing pipeline of FragDiffusion (Levy & Rector-Brooks, 2023), using BRICS fragmentation and fragment graph construction.

Prior molecular generation studies commonly evaluate models on the order of 10^3–10^4 generated samples. To assess chemical-space exploration at a substantially larger scale, we generate up to 10^7 molecules from each model. All models are evaluated at the same sample sizes using identical metric definitions. Validity is the fraction of generated molecules accepted by RDKit, while uniqueness is computed over canonicalized valid molecules.

We additionally evaluate molecular quality using QED (Bickerton et al., 2012), which measures drug-likeness, and the synthetic accessibility (SA) score (Ertl & Schuffenhauer, 2009), a heuristic measure of synthetic accessibility. QED and SA are computed on 10^5 generated molecules. To further assess retrosynthetic feasibility, we use AiZynthFinder (Saigiridharan et al., 2024), an open-source retrosynthetic planning tool that searches for synthetic routes from target molecules to purchasable precursors.

4.2 Large-Scale Molecular Generation

Table 1 reports validity and uniqueness as the number of generated molecules increases from 10^5 to 10^7. We omit novelty from the main comparison because it is nearly saturated at 100% across all evaluated models and therefore provides little discriminative value. Across all models, validity remains largely unaffected by the number of generated samples. In contrast, uniqueness gradually decreases as repeated molecules accumulate at larger sampling scales.

Table 1: Large-scale molecular generation results. Validity and uniqueness are evaluated at 10^5, 10^6, and 10^7 generated samples. $|V \cap U|$ denotes the number of distinct valid molecules obtained from 10^7 generated samples. Best results are shown in bold and second-best results are underlined.

			Validity (%) ↑			Uniqueness (%) ↑			$\lvert V \cap U \rvert$ ↑
Model	Unit	Ins./del.	10^5	10^6	10^7	10^5	10^6	10^7	10^7
DiGress	Atom	No	71.47	71.33	71.36	**99.99**	**99.95**	**99.68**	7,113,016
GrIDDD	Atom	Yes	88.10	88.23	88.22	99.83	99.04	97.10	8,566,049
IFH	Atom	Yes	57.98	57.87	57.92	<u>99.98</u>	<u>99.85</u>	<u>99.40</u>	5,757,629
JT-VAE	Fragment	No	**99.73**	**99.74**	**99.75**	99.25	96.72	89.40	8,916,975
FragDiffusion	Fragment	No	96.25	96.16	96.18	99.81	99.07	96.33	<u>9,265,827</u>
FragFlex	Fragment	Yes	<u>99.67</u>	<u>99.67</u>	<u>99.67</u>	99.94	99.37	95.52	**9,520,600**

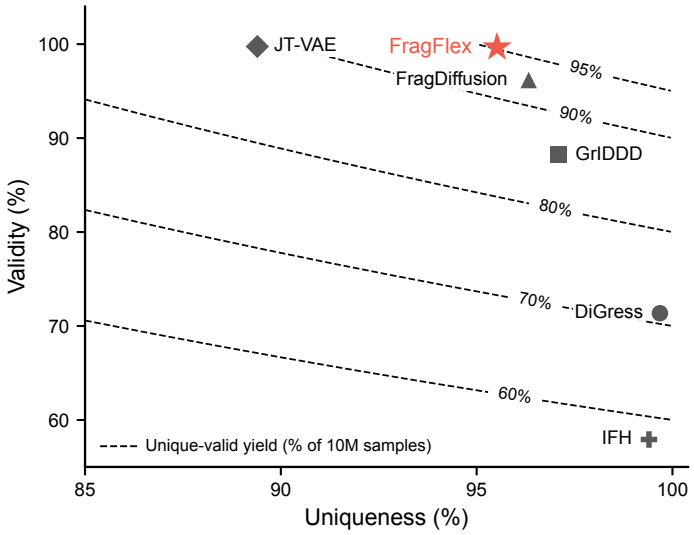

Figure 3: Validity–uniqueness trade-off at 10^7 samples. Dashed contours indicate constant unique-valid yield, computed as the product of validity and uniqueness and expressed as a percentage of all generated samples.

Fragment-based models consistently achieve substantially higher validity than atom-based models, although they generally exhibit lower uniqueness at 10^7 samples. JT-VAE achieves the highest validity at every evaluated scale but also exhibits the largest decline in uniqueness as the sampling scale increases. FragDiffusion shows the lowest validity among the fragment-based models. FragFlex approaches JT-VAE's high validity while retaining uniqueness comparable to FragDiffusion, thereby combining the strengths of the two fragment-based baselines.

Figure 3 highlights the relationship between validity and uniqueness at 10^7 samples. FragFlex maintains 99.67% validity across all three scales while retaining 95.52% uniqueness at 10^7 samples. This balance results in $9{,}520{,}600$ distinct valid molecules, the highest $|V \cap U|$ count among all models. FragFlex lies closest to the upper-right corner and is the only model above the 95% unique-valid yield contour, demonstrating the strongest balance between validity and uniqueness at scale.

4.3 Molecular Quality and Retrosynthetic Feasibility

Table 2 evaluates molecular quality using QED (Bickerton et al., 2012) and SA (Ertl & Schuffenhauer, 2009), together with retrosynthetic feasibility measured by the AiZynthFinder success rate. Overall, fragment-based models exhibit higher AiZynthFinder success rates than atom-based models. FragFlex achieves the highest QED score (0.764), the lowest SA score (3.021), and the highest AiZynthFinder success rate (43.2%) among all evaluated models. Together, these results show that FragFlex combines high generative validity with favorable molecular quality and retrosynthetic feasibility.

Table 2: Molecular quality and retrosynthetic feasibility. QED and SA are evaluated on 10^5 generated molecules, while AiZynthFinder success rate is evaluated on 10^3 generated molecules. Best results are shown in bold and second-best results are underlined.

Model	Unit	Ins./del.	QED ↑ $n = 10^5$	SA ↓ $n = 10^5$	AiZynthFinder (%) ↑ $n = 10^3$
DiGress	Atom	No	0.733	3.702	19.7
GrIDDD	Atom	Yes	0.651	<u>3.413</u>	20.2
IFH	Atom	Yes	0.725	5.367	5.3
JT-VAE	Fragment	No	<u>0.762</u>	3.418	<u>34.3</u>
FragDiffusion	Fragment	No	0.609	6.840	24.9
FragFlex	Fragment	Yes	**0.764**	**3.021**	**43.2**

4.4 Ablation Study

We evaluate three design choices in FragFlex using 10^5 generated molecules. The default FragFlex uniformly selects the final molecule from the valid product pool $\mathcal{P}$ and uses a terminal graph G_T containing a single fragment. FragFlex-Top1 instead selects the highest-ranked product in $\mathcal{P}$. FragFlex-Middle shifts the insertion/deletion schedule from the default $s_{\mathrm{p}} = 0$ to the midpoint, $s_{\mathrm{p}} = T/2$, following the schedule used by GrIDDD. FragFlex-Multi allows the terminal graph G_T to contain multiple fragments, following the variable-size terminal-state formulation used by GrIDDD.

The default FragFlex configuration achieves the highest number of distinct valid molecules. Selecting only the highest-scoring assembly candidate reduces uniqueness, suggesting that uniform selection helps preserve molecular diversity. Moving the insertion and deletion peak to the midpoint slightly improves validity but decreases uniqueness. The multi-fragment terminal state achieves the highest uniqueness but reduces validity. Overall, the default configuration provides the strongest balance between the two metrics.

Table 3: Ablation of the assembly selection rule, structural schedule, and terminal graph at $n = 10^5$. $|V \cap U|$ denotes the number of distinct valid molecules. Best results are shown in bold and second-best results are underlined.

Variant	Modification	Validity (%) ↑	Uniqueness (%) ↑	$\lvert V \cap U \rvert$ ↑
FragFlex-Top1	$\widehat{M} = M^{(\mathrm{Top1})}$	99.66	99.22	98,888
FragFlex-Middle	$s_{\mathrm{p}} = T/2$	**99.76**	99.32	99,076
FragFlex-Multi	Multi-fragment G_T	99.40	**99.97**	<u>99,370</u>
FragFlex	Default	<u>99.67</u>	<u>99.94</u>	**99,619**

5 Conclusion

We introduced FragFlex, a flexible-length molecular diffusion model that performs node insertion and deletion at the fragment level. FragFlex jointly predicts the number, identities, and connectivity of molecular fragments. A neural attachment-site predictor and valence-aware assembler then convert the generated fragment graph into an atom-level molecule. Together, these components allow molecular size to evolve throughout generation while retaining structural information encoded by fragments.

At a scale of 10^7 generated molecules, FragFlex produces more than 9.5 million distinct valid molecules, demonstrating a strong balance between validity and uniqueness. Beyond large-scale generation, FragFlex achieves the highest QED score, the lowest SA score, and the highest AiZynthFinder success rate among the evaluated models. Together, these results show that FragFlex combines large-scale generative diversity and validity with favorable molecular quality and retrosynthetic feasibility. Overall, fragment-level structural flexibility provides an effective foundation for large-scale chemical-space exploration.

This flexibility also suggests a promising extension to reinforcement learning, where property-based rewards could balance the exploration of new fragment combinations with the exploitation of promis-

ing molecules. The effectiveness of size-adaptive diffusion models such as GrIDDD in molecular property targeting motivates extending FragFlex to protein-target-conditioned generation. This extension would allow fragment composition and molecular size to adapt to target-specific binding environments. Future work will also investigate property-guided optimization and downstream drug discovery applications.

AI USE DISCLOSURE

In this work, we used generative AI tools to assist with research ideation and execution, retrieval and discovery of related work, drafting portions of the manuscript, and improving the clarity and readability of the writing. All AI-assisted content and analyses were reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021.

Guy W. Bemis and Mark A. Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of Medicinal Chemistry*, 39(15):2887–2893, 1996.

G. Richard Bickerton, Gaia V. Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L. Hopkins. Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4(2):90–98, 2012.

Regine S. Bohacek, Colin McMartin, and Wayne C. Guida. The art and practice of structure-based drug design: A molecular modeling perspective. *Medicinal Research Reviews*, 16(1):3–50, 1996.

Samuel Cognolato, Alessandro Sperduti, and Luciano Serafini. IFH: A diffusion framework for flexible design of graph generative models. In *ECAI 2024: 27th European Conference on Artificial Intelligence*, pp. 3039–3046, 2024.

Jörg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using 'drug-like' chemical fragment spaces. *ChemMedChem*, 3(10):1503–1507, 2008. doi: 10.1002/cmdc.200800178.

Ian Dunn and David R Koes. Flowmol3: flow matching for 3d de novo small-molecule generation. *Digital Discovery*, 5(5):2052–2066, 2026.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1(1):8, 2009.

Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pp. 2323–2332. PMLR, 2018.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International Conference on Machine Learning*, pp. 4839–4848. PMLR, 2020.

Shao Jinsong, Jia Qifeng, Chen Xing, Yajie Hao, and Li Wang. Molecular fragmentation as a crucial step in the AI-based drug development pathway. *Communications Chemistry*, 7(1):20, 2024.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, pp. 10362–10383. PMLR, 2022.

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. *arXiv preprint arXiv:2302.03596*, 2023.

Jaeyeon Kim, Lee Kit, Carles Domingo i Enrich, Yilun Du, Sham Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Albergo. Any-order flexible length masked diffusion. In *International Conference on Learning Representations*, 2026.

Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning*, pp. 1945–1954. PMLR, 2017.

Greg Landrum, Paolo Tosco, Brian Kelley, David Cosgrove, Riccardo Vianello, Eisuke Kawashima, Gareth Jones, Andrew Dalke, Brian Cole, Matt Swain, et al. RDKit/RDKit: 2023_03_1 (q1 2023) release, 2023.

Joongwon Lee, Seonghwan Kim, Seokhyun Moon, Hyunwoo Kim, and Woo Youn Kim. FragFM: Hierarchical framework for efficient molecule generation via fragment-level discrete flow matching. In *International Conference on Learning Representations*, 2026.

Daniel Levy and Jarrid Rector-Brooks. Molecular fragment-based diffusion model for drug discovery. In *ICLR 2023 Workshop on Machine Learning for Drug Discovery*, 2023.

Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. GraphDF: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning*, pp. 7192–7203. PMLR, 2021.

Matteo Ninniri, Marco Podda, and Davide Bacciu. Graph diffusion that can insert and delete. In *Advances in Neural Information Processing Systems*, volume 38, pp. 78375–78401, 2025. doi: 10.52202/085713-2625.

Chao Pang, Jianbo Qiao, Xiangxiang Zeng, Quan Zou, and Leyi Wei. Deep generative models in de novo drug molecule generation. *Journal of Chemical Information and Modeling*, 64(7):2174–2194, 2024.

Pavel G. Polishchuk, Timur I. Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on GDB-17 data. *Journal of Computer-Aided Molecular Design*, 27(8): 675–679, 2013.

Subham S. Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 130136–130184, 2024.

Lakshidaa Saigiridharan, Alan Kai Hassen, Helen Lai, Paula Torren-Peraire, Ola Engkvist, and Samuel Genheden. Aizynthfinder 4.0: developments based on learnings from 3 years of industrial application. *Journal of cheminformatics*, 16(1):57, 2024.

Petra Schneider, W. Patrick Walters, Alleyn T. Plowright, Norman Sieroka, Jennifer Listgarten, Robert A. Goodnow, Jasmin Fisher, Johanna M. Jansen, José S. Duca, Thomas S. Rush, et al. Rethinking drug design in the artificial intelligence era. *Nature Reviews Drug Discovery*, 19(5): 353–364, 2020.

Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, 4(1): 120–131, 2018.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. GraphAF: A flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pp. 412–422. Springer, 2018.

Teague Sterling and John J. Irwin. ZINC 15–ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015.

Morgan Thomas, Raul Astudillo, Yasir S. Raouf, Carles Navarro, Gianni De Fabritiis, and Andreas Bender. Generative virtual screening: From search to generate and back. *Digital Discovery*, 2026.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Sergei Voloboev. A review on fragment-based de novo 2D molecule generation. *arXiv preprint arXiv:2405.05293*, 2024.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. Graph transformer networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

A Detailed Methods

A.1 Notation

Table 4: Notation used in the main text and Appendix A.

Symbol	Meaning
M	Atom-level molecule.
$\mathcal{V} = \{F_1, \ldots, F_K\}$	Fixed vocabulary of K molecular fragments.
$G_0 = (X_0, E_0, S_0)$	Clean fragment graph with fragment labels, binary fragment edges, and directed attachment sites.
G_t	Fragment graph after structural deletion and categorical corruption at timestep t.
$V_0 = \{1, \ldots, N\}$	Node set of the clean fragment graph.
A_i	Number of atoms in the fragment assigned to node i.
r	Uniformly selected root node that is protected from structural deletion.
τ_i	Structural deletion time assigned to non-root node i.
$\zeta'(t)$	Normalized probability mass of a structural event at timestep t.
s_{P}	Reverse step near which fragment additions are most likely.
Δ_t	Number of nodes at the deletion boundary at timestep t.
f_θ	Main denoiser producing Z^X, Z^E, and Z^S.
g_ϕ	Auxiliary model predicting the boundary count Δ_t.
$\widehat{P}_{ij}^S$	Predicted distribution over local attachment atoms in fragment i for its edge to fragment j.
$\mathcal{P}$	Ranked pool of unique valid products returned by the assembler.
$\widehat{M}$	Final atom-level molecule selected from $\mathcal{P}$.

A.2 Fragment Graph Preprocessing

Each molecule is decomposed into fragments using the preprocessing pipeline of FragDiffusion. In FragDiffusion, each connection between two fragments is represented by a single categorical code called an attachment mode. For an edge (i, j), this code identifies the atom in fragment F_i, the atom in fragment F_j, and the bond type connecting them. For example, one attachment mode may represent "connect atom 3 of F_i to atom 1 of F_j with a single bond." The possible attachment modes and their empirical frequencies are obtained from the training data and stored in a lookup table.

FragFlex uses this lookup table only during preprocessing to convert the attachment-mode labels into neural attachment-site targets. For each fragment edge (i, j), preprocessing recovers the two local atom indices and stores them as $S_{0,ij}$ and $S_{0,ji}$, while replacing the attachment-mode edge with the binary labels

$$E_{0,ij} = E_{0,ji} = 1.$$

Consequently, the model is trained and sampled using binary fragment topology and explicit directed attachment-site targets. Unlike FragDiffusion, FragFlex does not use an attachment-mode lookup table during inference; instead, its neural attachment-site predictor directly proposes the atom-level connection for each generated fragment edge.

A.3 FragFlex Training

The training procedure is summarized in Algorithm 1 in the main text.

Forward corruption. For each clean graph G_0, we sample a timestep $t \sim \mathrm{Unif}\{1, \dots, T\}$ and a root $r \sim \mathrm{Unif}(V_0)$. Each non-root node receives a deletion time $\tau_i \sim \mathrm{Cat}(\zeta')$, while $\tau_r = T + 1$. Define the retained nodes V_t and deletion-boundary nodes B_t as

$$V_t = \{i \in V_0 : \tau_i \geq t\}, \qquad B_t = \{i \in V_0 \setminus \{r\} : \tau_i = t\}. \tag{5}$$

Nodes with $\tau_i < t$ are absent from G_t. Following GrIDDD, nodes in B_t use the transient state DEL*, and every edge incident to a boundary node uses the corresponding transient edge state. All other retained node and edge labels are corrupted with the empirical-marginal kernel,

$$\bar{Q}_t(m) = \bar{\alpha}_t I + (1 - \bar{\alpha}_t)\mathbf{1}m^\top, \tag{6}$$

where m is the empirical fragment or edge marginal and $\bar{\alpha}_t$ is the cumulative cosine schedule. The protected root uses the empirical fragment marginal as its terminal prior. Thus, G_T contains one fragment sampled from the empirical fragment distribution and is not restricted to a single-atom fragment.

The resulting corrupted graph G_t and the current timestep t are passed to the main denoiser f_θ. The denoiser predicts logits for the clean fragment identities, binary fragment-edge labels, and directed attachment sites:

$$(Z^X, Z^E, Z^S) = f_\theta(G_t, t).$$

Here, Z^X, Z^E, and Z^S denote the node, edge, and attachment-site logits, respectively. For a source fragment F_i containing A_i atoms, positions $a \geq A_i$ in $Z^S_{ij,a}$ are masked because they do not correspond to real atoms.

Fragment deletion and insertion schedule. The structural schedule $\zeta'(t)$ determines when each non-root fragment is removed during forward corruption or, equivalently, added during reverse sampling. Because reverse sampling proceeds from $t = T$ to $t = 1$, we define $s_t = T - t$ as the number of reverse steps completed after the initial one-root graph G_T. The parameter s_p specifies the reverse step near which fragment additions are most likely to occur.

We compute the unnormalized timestep weights as

$$u_t = \sigma(x_t)\big(1 - \sigma(x_t)\big), \qquad \zeta'(t) = \frac{u_t}{\sum_{\tau=1}^{T} u_\tau}, \qquad x_t = \frac{s_t - s_\mathrm{p}}{w\,T}. \tag{7}$$

where σ is the sigmoid function and $w\,T$ controls the temporal spread of fragment additions. Each non-root fragment then receives an event time $\tau_i \sim \mathrm{Cat}(\zeta')$. We use $T = 500$, $s_\mathrm{p} = 0$, and $w = 0.05$, concentrating graph growth near the beginning of reverse sampling.

Fragment-count model. At timestep t, the boundary set B_t contains the fragments whose event time is $\tau_i = t$. Its size $\Delta_t = |B_t|$ is the number of fragments that must be added at that step during reverse sampling. The count network g_ϕ is trained to predict this number.

In the corrupted graph G_t, the boundary fragments are represented by DEL* nodes. If these nodes were given directly to g_ϕ, the network could obtain the correct answer simply by counting them. We therefore remove the boundary nodes from the count-network input:

$$G_t^- = G_t \setminus B_t. \tag{8}$$

The count network then predicts Δ_t from G_t^- and the current timestep:

$$Z_t^\Delta = g_\phi(G_t^-, t), \qquad p_\phi^\Delta = \mathrm{softmax}(Z_t^\Delta),$$

where p_ϕ^Δ is the predicted distribution over the possible values of Δ_t. Consequently, g_ϕ must infer how many fragments should be added from the current graph structure and timestep, rather than reading the answer from visible DEL* nodes.

Training objective. We use cross-entropy losses for fragment identity, edge, attachment-site, and fragment-count prediction:

$$\begin{aligned} \mathcal{L}_X &= \mathrm{CE}(Z^X, X_0), & \mathcal{L}_E &= \mathrm{CE}(Z^E, E_0), \\ \mathcal{L}_S &= \mathrm{CE}(Z^S, S_0), & \mathcal{L}_\Delta &= \mathrm{CE}(Z_t^\Delta, \Delta_t). \end{aligned} \tag{9}$$

The first two losses are evaluated over retained nodes and node pairs, respectively. The total objective is

$$\mathcal{L} = \mathcal{L}_X + \lambda_E \mathcal{L}_E + \lambda_\Delta \mathcal{L}_\Delta + \lambda_S \mathcal{L}_S. \tag{10}$$

A.4 Flexible-Length Reverse Sampling

The reverse sampling procedure is summarized in Algorithm 2 in the main text.

Fragment insertion. The count network g_ϕ predicts how many fragment nodes should be added. Note that the current graph has the same form as G_t^-, which is used to train the count network. After sampling the predicted count $\widehat{\Delta}_t$, that many DEL* nodes are appended, up to the maximum graph size $N_{\max}$. These nodes are temporary placeholders. Their fragment identities and edges are determined by the subsequent denoising step.

Categorical denoising. For each node or edge variable z, the main denoiser $f_\theta(G_t, t)$ outputs clean-category logits defining $p_\theta(z_0 \mid G_t, t)$ after softmax. This clean-category distribution is combined with the known forward transition kernel to obtain the reverse distribution

$$p_\theta(z_{t-1} \mid z_t, G_t, t) = \sum_{z_0} q(z_{t-1} \mid z_t, z_0)\, p_\theta(z_0 \mid G_t, t). \tag{11}$$

A new category z_{t-1} is then sampled from this distribution. Before sampling, probability assigned to the transient DEL* state is removed and the remaining probabilities are renormalized.

Connectivity repair. Because fragment edges are sampled independently, the generated fragment graph is not guaranteed to be connected. If a newly inserted node is resolved into a fragment without an edge to the existing graph, the sampler connects it to the existing node with the highest predicted probability of a real edge. After the final reverse step, any remaining components are joined by repeatedly adding the highest-scoring edge between two different components. If the final graph contains C connected components before this repair, exactly $C - 1$ edges are added.

Attachment-site prediction. The attachment-site logits do not participate in the categorical reverse posterior. They are therefore computed only at $t = 1$, when the fragment identities and topology are about to be resolved into G_0. Positions that do not correspond to real atoms are masked, and the remaining logits are normalized to produce the directed attachment-site distributions $\widehat{P}^S$. These distributions are subsequently used by the neural assembler to select atom-level connections.

A.5 Valence-Aware Neural Assembly

The neural assembly procedure is summarized in Algorithm 3 in the main text.

Before top-k selection, the assembler uses RDKit to exclude atoms that cannot accept an external single bond, treating an explicit capping hydrogen as replaceable. Here, $\widehat{P}^S$ denotes the attachment-site distribution predicted by the neural network, and $\widetilde{P}^S$ denotes the distribution obtained by applying this feasibility mask and renormalizing the remaining probabilities.

For each generated fragment edge (i, j), define

$$\mathcal{A}_{i\to j} = \mathrm{TopK}_k(\widetilde{P}^S_{ij}), \qquad \mathcal{A}_{j\to i} = \mathrm{TopK}_k(\widetilde{P}^S_{ji}). \tag{12}$$

The Cartesian product of these sets $\mathcal{C}_{ij} = \mathcal{A}_{i\to j} \times \mathcal{A}_{j\to i}$ contains at most k^2 candidate atom pairs. For a candidate pair $(a_i, a_j) \in \mathcal{C}_{ij}$, the neural score is

$$\ell_{ij}(a_i, a_j) = \log \widetilde{P}^S_{ij,a_i} + \log \widetilde{P}^S_{ji,a_j}. \tag{13}$$

Fragment edges are processed in increasing order of $|\mathcal{C}_{ij}|$, so edges with fewer candidate atom pairs are considered first. For each edge, every beam state is expanded over its candidate pairs. Each expansion replaces an explicit cap hydrogen when needed, adds a single bond, and is retained only if RDKit's strict valence check succeeds.

In a cyclic graph, an invalid predicted edge can be skipped because the remaining edges may still keep the graph connected. Skipping is penalized and is not allowed for trees, where every edge is required for connectivity. Beam states are ranked by the following priorities: more successfully added fragment edges, fewer skipped edges, a higher accumulated neural pair score, and fewer non-top-1 attachment choices. After each fragment edge, only the best B_{beam} states are retained. Thus, k limits the local atom choices for each edge, whereas B_{beam} limits the number of global partial assemblies.

After all edges are processed, beam molecules are examined in the same ranking order. They are fully sanitized, required to be connected, and deduplicated by canonical SMILES. At most $P_{\max}$ unique valid products are collected into a pool $\mathcal{P}$, and the final molecule is sampled as

$$\widehat{M} \sim \mathrm{Unif}(\mathcal{P}), \qquad |\mathcal{P}| \le P_{\max}. \tag{14}$$

Assembly fails if $\mathcal{P} = \varnothing$.

The search is score-based until $\mathcal{P}$ is constructed, while the final product is sampled uniformly from this pool. This prevents the assembler from always selecting the highest-scoring valid product and increases atom-level assembly diversity for the same fragment graph. Because both the local candidate sets and beam states are truncated, assembly may fail even when an unexamined valid assembly exists.

B Additional Results

B.1 Scaffold Diversity and Molecular Quality

We additionally evaluate scaffold-level diversity by counting unique non-empty Bemis–Murcko (BM) scaffolds (Bemis & Murcko, 1996). As shown in Table 5, models generating more unique BM scaffolds tend to produce molecules with less favorable QED and SA profiles, characterized by lower QED and/or higher SA scores. Although FragFlex does not achieve the highest scaffold diversity, it obtains the best QED and SA scores while still generating millions of unique BM scaffolds. These results indicate that FragFlex provides a favorable balance between scaffold diversity and molecular quality rather than maximizing scaffold count alone.

C Implementation Details

Table 6: Hyperparameters and model configuration

Component	Setting
Diffusion	$T = 500$ with a cosine noise schedule $\bar{\alpha}_t$ ($s = 0.008$)
Deletion/insertion schedule	Logistic $\zeta'(t)$ with $s_{\mathrm{p}} = 0$, $w = 0.05$
Terminal prior G_T	One fragment from the empirical fragment marginal
Main denoiser f_θ	7 layers, node width 384, edge width 96, 8 heads
Count model g_ϕ	2 layers, node width 128, edge width 32, 8 heads
Maximum graph size	$N_{\max} = 32$ fragments
Training	200 epochs, 4 GPUs, batch 32 per GPU, dropout 0.05
Optimizer	AdamW, LR 2×10^{-4}, weight decay 10^{-3}
LR schedule	5% warmup from 0.1, cosine decay to 0.05
Assembly	$k = 4$, $B_{\mathrm{beam}} = 512$, $P_{\max} = 16$, $\widehat{M} \sim \mathrm{Unif}(\mathcal{P})$

Table 5: Molecular properties, distinct valid molecules, and unique Bemis–Murcko (BM) scaffolds across generation scales. QED and SA are evaluated at $n = 10^5$. Acyclic molecules are excluded from the BM scaffold counts.

			QED ↑	SA ↓	$\lvert V \cap U \rvert$ molecules ↑			Unique BM scaffolds ↑		
Model	Unit	Ins./del.	$n = 10^5$		10^5	10^6	10^7	10^5	10^6	10^7
DiGress	Atom	No	0.733	3.702	71,458	712,891	7,113,016	55,465	479,402	3,990,775
GrIDDD	Atom	Yes	0.651	3.413	87,949	873,848	8,566,049	**77,285**	**691,875**	**5,889,181**
IFH	Atom	Yes	0.725	5.367	57,964	577,867	5,757,629	57,093	558,945	5,402,044
JT-VAE	Fragment	No	0.762	3.418	98,975	964,773	8,916,975	66,996	504,974	3,404,902
FragDiffusion	Fragment	No	0.609	6.840	96,073	952,672	9,265,827	74,771	618,016	4,734,767
FragFlex	Fragment	Yes	**0.764**	**3.021**	**99,619**	**990,390**	**9,520,600**	67,250	516,485	3,492,520

Training was performed on four NVIDIA L40 GPUs. Training the 14.2M-parameter model for 200 epochs required approximately 10 hours of wall-clock time. Sampling was performed on a single NVIDIA GeForce RTX 2080 Ti GPU, requiring approximately 3 hours to generate 10^5 molecules.

D Code Availability

Our code is available at `https://anonymous.4open.science/r/FragFlex-2C20/`

E FRAGMENT VISUALIZATION

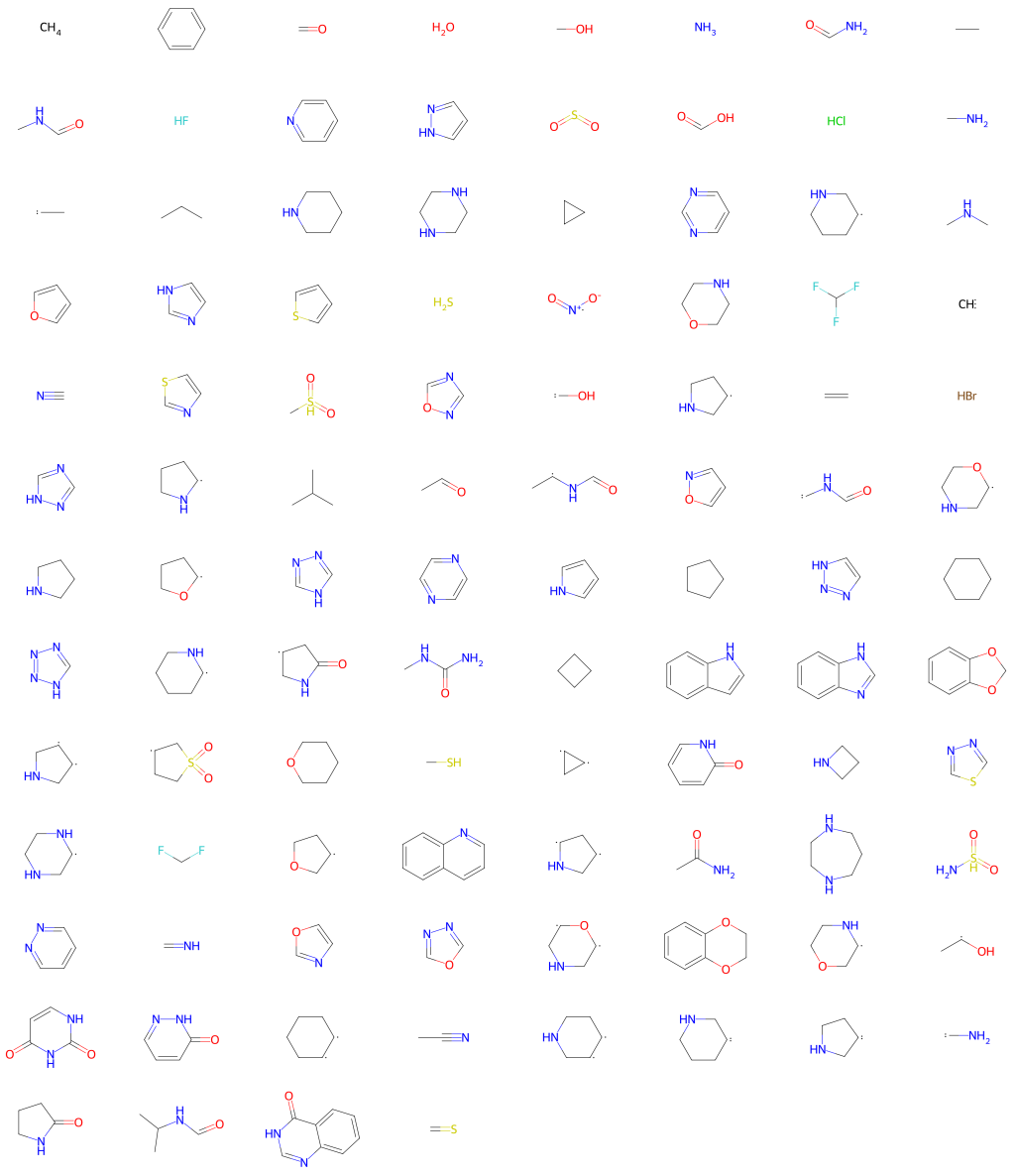

Figure 4: The 100 fragment types used by FragFlex, arranged in descending order of frequency in the ZINC training set.

F GENERATED MOLECULES

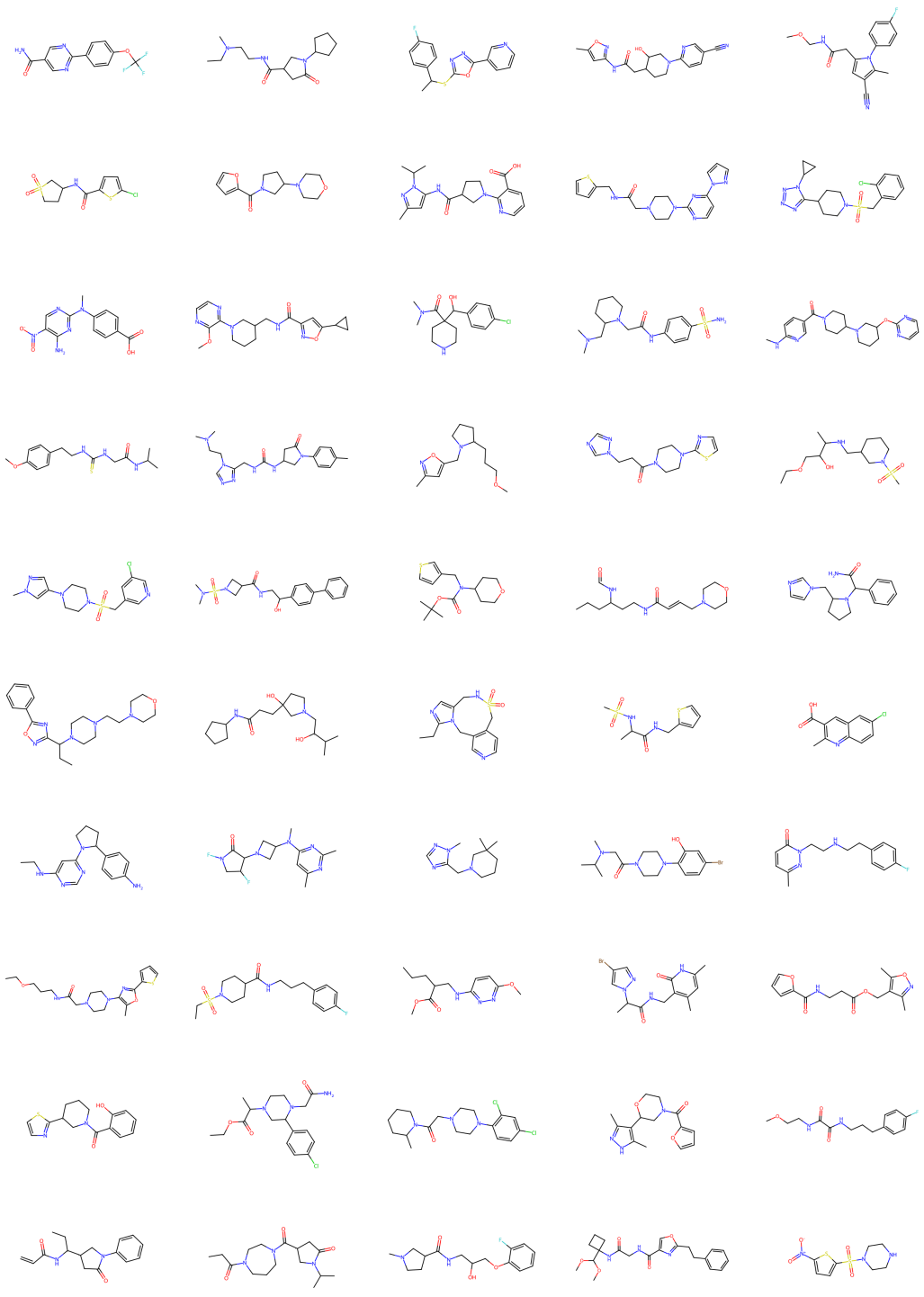

Figure 5: 50 random samples generated by FragFlex.