FragFM: Hierarchical Framework for Efficient Molecule Generation via Fragment-Level Discrete Flow Matching

Anonymous Author(s)
Affiliation
Address
email

Abstract

We introduce FragFM, a novel hierarchical framework via fragment-level discrete flow matching for efficient molecular graph generation. FragFM generates molecules at the fragment level, leveraging a coarse-to-fine autoencoder to reconstruct details at the atom level. Together with a stochastic fragment bag strategy to effectively handle an extensive fragment space, our framework enables more efficient and scalable molecular generation. We demonstrate that our fragment-based approach achieves better property control than the atom-based method and additional flexibility through conditioning the fragment bag. We also propose a Natural Product Generation benchmark (NPGen) to evaluate modern molecular graph generative models' ability to generate natural product-like molecules. Since natural products are biologically prevalidated and differ from typical drug-like molecules, our benchmark provides a more challenging yet meaningful evaluation relevant to drug discovery. We conduct a FragFM comparative study against various models on diverse molecular generation benchmarks, including NPGen, demonstrating superior performance. The results highlight the potential of fragment-based generative modeling for large-scale, property-aware molecular design, paving the way for more efficient exploration of chemical space.

1 Introduction

Deep generative models are achieving remarkable success in modeling complex, structured data, with graph generation being a prominent application area [1, 2]. Among various applications, *de novo* molecular graph generation, which has the potential to accelerate drug and material discovery, is a particularly important. Recently, diffusion- and flow-based graph generative models have demonstrated the ability to generate molecular graphs [3–6].

However, these models that are built on atom-based representation face significant scalability challenges, particularly in generating large and complex molecules [7]. The quadratic growth of edges as the graph size increases results in computational inefficiencies. At the same time, the inherent sparsity of chemical bonds makes accurate edge prediction more complex, often leading to unrealistic molecular structures or invalid connectivity constraints [7, 8]. Moreover, graph neural networks struggle to capture topological features like rings, leading to deviations from chemically valid structures. Although various methods incorporate auxiliary features (e.g., spectral, ring, and valency information) to mitigate these issues, they do not fully resolve the sparsity and scalability bottlenecks [3].

Fragment-based strategies, rooted in long-standing success in traditional drug discovery, offer an alternative [9–11]. By assembling molecules from chemically meaningful substructures, these approaches enable a more efficient exploration of chemical space, preserve global structural coherence,

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

and provide finer control over molecular properties than atom-based methods [12–15]. Diffusion models also adopted the fragment-based approach, showing their potential in improving scalability and property control [16, 17]. However, the existing methods depend on a small fragment library or employ automated fragmentation procedures, leading to restricted chemical diversity and limiting the usage of domain knowledge.

Here, we introduce FragFM, a novel hierarchical framework for molecular graph generation to address these challenges. FragFM first generates a fragment-level graph using discrete flow matching and then reconstructs it into an atom-level graph without information loss. To this end, we develop a novel **stochastic fragment bag strategy** that circumvents reliance on fixed fragment libraries, along with a **coarse-to-fine autoencoder** that ensures direct atom-level reconstruction from the generated fragment-level graph. Consequently, FragFM can efficiently explore the molecular space, avoiding the generation of chemically implausible molecules with an extensive fragment space at moderate computational cost.

We extensively evaluate FragFM on standard molecule generation benchmarks [18, 19], where it consistently outperforms the previous graph generative models in various metrics. FragFM outperforms denoising-based models even with significantly fewer denoising steps. Our fragment-based approach enables more flexible property-guided molecular generation with fragment bag control over standard guidance strategies. Furthermore, we propose a new benchmark, **NPGen**, as the benchmarks mainly focus on small drug-like molecules and the performance of graph generative models has become saturated NPGen opens a new avenue for graph generative models specifically targeting natural products, which is crucial for drug discovery due to their bio-compatibility and structural novelty[20]. Although inherent complexity is present in natural products, FragFM shows particularly strong performance on the new task, highlighting its strength in capturing high-level structural semantics for natural products.

2 Related Works

2.1 Molecular Graph Generative Models

Modern molecular graph generative models can be classified into autoregressive and one-shot generation models. Autoregressive models generate graphs sequentially based on their node, generally an atom or fragment, and corresponding edge representations[12, 21–23]. Despite their performance, these models have an intrinsic issue in learning the permutation of nodes in the graph, which must be invariant for a given graph, often making them highly inefficient. Among one-shot models, there exists a model that directly generates molecular graphs [24]. Also, denoising models have recently become fundamental for generating molecular graphs by iteratively refining noisy graphs into structured molecular representations. Diffusion methods [25, 26], which have been successful in various domains, have been extended to graph structure data [2, 27], demonstrating the advantages of applying diffusion in graph generation. This approach was further extended by incorporating discrete stochastic processes [28], addressing the inherently discrete nature of molecular graphs [3]. The discrete diffusion modeling is reformulated using the continuous-time Markov chain (CTMC) [5, 29, 30], allowing for more flexible and adaptive generative processes. More recently, flow-based models have been explored for generating molecular graphs. Continuous flow matching [31] has been applied to structured data [6], while discrete flow models [32, 33] have been extended to categorical data generation, with recent methods showing that they can also model molecular distributions as diffusion models[4, 34].

2.2 Fragment-Based Molecule Generation

Fragment-based molecular generative models construct new molecules by assembling existing molecular substructures, known as fragments. This strategy enhances chemical validity and synthesizability, facilitating the efficient exploration of novel molecular structures. Several works have employed fragment-based approaches within variational autoencoders (VAEs) by learning to assemble in a chemically meaningful way [35–37]. Also, Jin et al. [12] adopts a stepwise generation approach, constructing a coarse fragment-level graph before refining it into an atom-level molecule through substructure completion. The other strategies construct molecules sequentially assembling fragments, enabling better control over molecular properties during generation [13, 35].

Fragment-based approaches have also been explored in diffusion-based molecular graph generation. Levy and Rector-Brooks [16] proposed a method that utilizes a fixed set of frequently occurring fragments to generate drug-like molecules, ensuring chemical validity but limiting exploration beyond predefined structures. Since enumerating all possible fragment types is infeasible, the method operates solely within a fixed fragment vocabulary. In contrast, Chen et al. [17] proposed an alternative, dataset-dependent fragmentation strategy based on byte-pair encoding, which provides a more flexible molecular representation. However, this approach does not yet integrate chemically meaningful fragmentation methods [38, 39], which are inspired by chemical synthesis and functionality, limiting its ability to leverage domain-specific chemical priors.

3 FragFM Framework

We propose FragFM, a novel hierarchical framework that utilizes discrete flow matching at the fragment-level graph. As shown in the fig. 1, we propose two novel strategies: a coarse-to-fine autoencoder and a stochastic fragment bag strategy. The former compresses atom-level graphs into fragment-level graphs without any information loss using the latent variable, allowing us to utilize discrete flow matching (DFM) in the fragment-level graph. The latter ensures the model handles comprehensive fragment libraries and achieves generalizability to the fragment bag. Starting from fragment graph notation in section 3.1, we elaborate on the details of the conversion between fragment- and atom-level graphs in section 3.2, and then the training and generation procedures for fragment-level graph in sections 3.3 and 3.4..

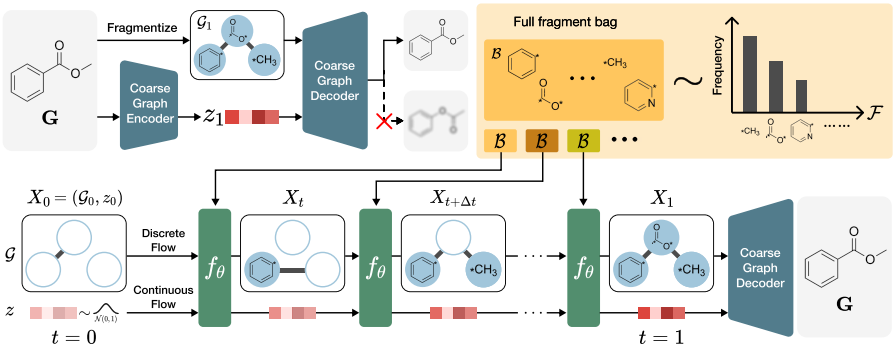


Figure 1: **Overview of FragFM**. FragFM utilizes a hierarchical framework of coarse-to-fine autoencoder (section 3.2) and fragment-level graph flow matching (section 3.3). An input atom-level graph (G) is initially decomposed via fragmentation rule. This is then processed by a coarse-to-fine encoder, which compresses it into a joint representation $X = (\mathcal{G}, z)$ comprising a fragment-level graph $\mathcal{G}$ and a latent vector z designed to capture fine-grained atomistic connectivity information not explicitly present in $\mathcal{G}$. During generation (section 3.4), neural network f_θ selects the most probable fragment from a fragment bag $\mathcal{B}$, which is stochastically sampled subset of the full fragment bag $\mathcal{F}$. FragFM then employs two flow-matching processes: (i) a discrete flow generates the target fragment-level graph $\mathcal{G}_1$ from an initial $\mathcal{G}_0$ (mask and uniform prior for node and edge, respectively), operating with fragments from $\mathcal{B}$; (ii) a continuous flow generates the target latent vector z_1 from a Gaussian prior $\mathcal{N}(0, 1)$ (from an initial z_0). Finally, a coarse-to-fine decoder utilizes the generated pair $(\mathcal{G}_1, z_1)$ to reconstruct the full atom-level molecular graph.

3.1 Fragment Graph Notation

We represent a molecule at the atom level as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is a set of atomic nodes, and $\mathbf{E}$ is the set of edges with associated features (i.e., bond types). The node $\mathbf{v}^{(k)} \in \mathbf{V}$ corresponds to an atom, while $\mathbf{e}^{(kl)} \in \mathbf{E}$ denotes a chemical bond between atoms $\mathbf{v}^{(k)}$ and $\mathbf{v}^{(l)}$. At the fragment level, we introduce a coarse-grained representation of the molecule as a graph $\mathcal{G} = (\mathcal{X}, \mathcal{E})$. Here, each node $\mathbf{x}^{(i)} \in \mathcal{X}$ corresponds to a fragment, and each edge $\varepsilon^{(ij)} \in \mathcal{E}$ represents whether two fragments are connected. Let $\mathcal{F}$ be the set of all possible fragments. Formally, fragment $\mathbf{x}^{(i)}$ is an atom-level sub-graph of $\mathbf{G}$. More specifically, $\{\mathbf{x}^{(i)}\} = \{(\mathbf{V}_i, \mathbf{E}_i)\}$ are mutually disjoint sub-graphs, where

$\mathbf{V}_i \subseteq \mathbf{V}$ and $\mathbf{E}_i \subseteq \mathbf{E}$, with $\mathbf{V}_i \cap \mathbf{V}_j = \varnothing$ for different fragment indices i and j. The coarse-grained edges $\varepsilon^{(ij)} \in \{0, 1\}$ are induced from $\mathbf{E}$, meaning that two fragments $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathcal{X}$ are connected if at least one bond exists between their corresponding atoms, *i.e.*,

$$\varepsilon^{(ij)} \in \mathcal{E} \quad \text{if} \quad \exists\, \mathbf{e}^{(kl)} \in \mathbf{E} \quad \text{s.t} \quad \mathbf{v}^{(k)} \in \mathbf{V}_i, \mathbf{v}^{(l)} \in \mathbf{V}_j. \tag{1}$$

3.2 Molecular Graph Compression by Coarse-to-fine Autoencoder

While a fragment-level graph ($\mathcal{G}$) offers a higher-level abstraction of molecular structures, it also introduces ambiguity in reconstructing atomic connections. Specifically, a single fragment-level connectivity ($\mathcal{E}$) can map to multiple distinct, valid atom-level configurations. To achieve accurate end-to-end molecular generation, it is therefore crucial to preserve atom-level connectivity ($\mathbf{E}$) when forming the fragment-level representation. Drawing on a hierarchical generative framework [40–42], we employ a coarse-to-fine autoencoder. The encoder compresses an atom-level graph ($\mathbf{G}$) into its fragment-level counterpart and, for each input molecule, outputs a single continuous latent vector z that encodes the committed connectivity details. With $\mathcal{G}$ and z the decoder predicts only the atom-level edges linking fragments that are adjacent in $\mathcal{G}$, and we discretize its continuous outputs via the Blossom algorithm [43]. Additional implementation details are given in appendices A.1 and A.2.

$$\textbf{Encoder:} \quad \mathbf{G} \xrightarrow{\text{enc}} (\mathcal{G}, z),$$
$$\textbf{Decoder:} \quad (\mathcal{G}, z) \xrightarrow{\text{dec}} \hat{\mathbf{G}}.$$

3.3 Flow Matching for Coarse Graph

We aim to model the joint distribution over the fragment-level graph and its latent representation, $X := (\mathcal{G}, z)$, through the flow-matching after a continuous-time generative paradigm. Flow matching begins at a known prior at t=0 and follows a learned vector field that continuously transforms this prior into the target data distribution at t=1.

In our coarse graph $\mathcal{G}$, both nodes $\mathbf{x} \in \mathcal{X}$ (fragment types) and edges $\varepsilon \in \mathcal{E}$ (fragment connectivities) are discrete categorical variables, for which we adopt DFM realized by a continuous time Markov chain (CTMC) [32]. The latent vector z is continuous, and thus we treat it with continuous flow matching [31]. This hybrid set-up allows us to evolve the discrete fragment graph and the continuous latent information jointly from a simple prior to the desired data distribution. In this section, we focus on the fragment type generation modeling, and modeling details for the latent vector and edge are described in appendix A.3.

Discrete Flow Matching Following the original DFM formulation [32], we specify the node (fragments type) distribution at t=1 as $p_1(\mathbf{x})$ and define the $\mathbf{x}_1$-conditioned time marginal by a linear interpolation

$$p_{t\,|1}(\mathbf{x}_t \mid \mathbf{x}_1) = t\,\delta(\mathbf{x}_t, \mathbf{x}_1) + (1{-}t)\,p_0(\mathbf{x}_t), \tag{2}$$

where p_0 is a prior and $\delta(\cdot, \cdot)$ is the Kronecker delta. We utilized a masked distribution proposed by Campbell et al. [32] as prior. They proposed a CTMC transition rate that realizes this marginal,

$$R_t(\mathbf{x}, \mathbf{y} \mid \mathbf{x}_1) = \frac{\mathrm{ReLU}\big(\partial_t p_{t\,|1}(\mathbf{y} \mid \mathbf{x}_1) - \partial_t p_{t\,|1}(\mathbf{x} \mid \mathbf{x}_1)\big)}{S\, p_{t\,|1}(\mathbf{x} \mid \mathbf{x}_1)}, \qquad \forall \mathbf{x} \neq \mathbf{y}, \tag{3}$$

with S the number of states for which $p_{t|1}(\mathbf{x} \mid \mathbf{x}_1) > 0$. A brief algebraic manipulation of the Kolmogorov forward equation yields the $\mathbf{x}_1$-unconditional generator

$$R_t(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{\mathbf{x}_1 \sim p_{1|t}(\cdot|\mathbf{x})}\big[R_t(\mathbf{x}, \mathbf{y} \mid \mathbf{x}_1)\big]. \tag{4}$$

from which we can sample trajectories $\{\mathbf{x}_t\}_{t\in[0,1]}$ directly. The only unknown quantity in eq. (4) is the posterior $p_{1|t}(\mathbf{x}_1 \mid \mathbf{x}_t)$, which we approximate with a neural network.

Parameterization and Info-NCE Loss. Because realistic chemical spaces require a very large set of fragment types $|\mathcal{F}|$, assembling the transition matrix in eq. (4) becomes computationally infeasible. We therefore detour the explicit expectation of eq. (4) by Monte-Carlo (MC) sampling, with introducing stochastic fragment bag strategy. Given the current noisy state $\mathbf{X}_t = (\mathcal{G}_t, z_t)$, we

first draw a subset $\mathcal{B} \subset \mathcal{F}$ of size N and then sample a node $\mathbf{x}_1$ within that bag. As a result, the model needs to approximate the in-bag conditional posterior $p(\mathbf{x}_1 \mid \mathbf{X}_t, \mathbf{x}_1 \in \mathcal{B})$ rather than the unconditional one $p_{1|t}(\mathbf{x}_1 \mid \mathbf{X}_t)$.

Following the Info-NCE formulation [44], we construct the fragment bag and parameterize the density ratio $p_{1|t}(\mathbf{x}_1|\mathbf{X}_t)/p_1(\mathbf{x}_1)$ with a neural network f_θ. For each training step we build a bag $\mathcal{B}$ that contains one positive fragment $\mathbf{x}_1^+ \sim p_{1|t}(\mathbf{x}_1|\mathbf{X}_t)$ and $N-1$ negative fragments $\mathbf{x}_1^-$ sampled i.i.d. from the marginal fragments library distribution $p_1(\mathbf{x})$. Applying the Info-NCE formulation, we can write the in-bag posterior as

$$p_{1|t}\big(\mathbf{x} \mid \mathbf{X}_t, \mathcal{B}\big) = \frac{\mathbf{1}_{\mathcal{B}}(\mathbf{x}) p_{1|t}(\mathbf{x} \mid \mathbf{X}_t)/p_1(\mathbf{x})}{\sum_{\mathbf{y}\in\mathcal{B}} p_{1|t}(\mathbf{y} \mid \mathbf{X}_t)/p_1(\mathbf{y})}, \tag{5}$$

where $\mathbf{1}_{\mathcal{B}}$ is an indicator function. We let the $f_\theta(\mathbf{X}_t, \mathbf{x})$ approximate the unknown density ratio $p_{1|t}(\mathbf{x}|\mathbf{X}_t)/p_1(\mathbf{x})$, by optimizing θ with the standard Info-NCE loss:

$$\mathcal{L}(\theta) = -\,\mathbb{E}_{\mathcal{B}}\left[\log \frac{f_\theta(\mathbf{X}_t, \mathbf{x}^+)}{\sum_{\mathbf{y}\in\mathcal{B}} f_\theta(\mathbf{X}_t, \mathbf{y})}\right], \tag{6}$$

which encourages the network to assign higher scores to the positive $\mathbf{x}^+$ within $\mathcal{B}$ while pushing down the negatives. Because the loss computes f_θ for only $\mathbf{x} \in \mathcal{B}$, the computational cost scales with the bag size N rather than the full library $|\mathcal{F}|$, making training process computationally tractable even when the fragment vocabulary is extremely large.

3.4 Generation process

In the sampling phase, we require a discretized forward kernel for nodes, edges, and the latent vector: $p_{t+\Delta t|t}(\mathbf{X}_{t+\Delta t} \mid \mathbf{X}_t) = \prod_i p_{t+\Delta t|t}(\mathbf{x}_{t+\Delta t}^{(i)} \mid \mathbf{X}_t) \prod_{ij} p_{t+\Delta t|t}(\varepsilon_{t+\Delta t}^{(ij)} \mid \mathbf{X}_t)\, p_{t+\Delta t|t}(z_{t+\Delta t} \mid \mathbf{X}_t)$. Despite the latent variable being modeled deterministically in the flow-matching framework, we keep the probabilistic notation because the latent trajectory is a limiting case of a diffusion bridge. Similar to Campbell et al. [32], we modeled each transition of nodes and edges as independent. In this section we will focus on the DFM process for nodes, while details of edges and latent vector will be provided in appendix A.3.

In-bag transition kernel One step transition kernel $p_{t+\Delta t|t}$ can be obtained by direct Euler integration of the rate matrix from eq. (4), which entails an expectation of $\mathbf{x}_1$ over the full fragment set $\mathcal{F}$. We here define an in-bag transition kernel, by replacing the expectation by restricting $\mathbf{x}_1$ to a randomly selected subset $\mathcal{B}$.

In the course of the sampling process, it is not possible to access $p_{1|t}$ to place a positive sample in $\mathcal{B}$. Instead, following the conventional Info-NCE approaches, we construct $\mathcal{B}$ by drawing N i.i.d. fragments from the marginal p_1, assuming that $\mathcal{B}$ is independent to the current state. Consequently the $\mathcal{B}$ conditioned $\mathbf{x}_1^{(i)}$-posterior is

$$p^\theta_{1|t,\mathcal{B}}(\mathbf{x}_1^{(i)} \mid \mathbf{X}_t, \mathcal{B}) = \frac{\mathbf{1}_{\mathcal{B}}(\mathbf{x}_1^{(i)}) f_\theta(\mathbf{X}_t, \mathbf{x}_1^{(i)})}{\sum_{\mathbf{y}\in\mathcal{B}} f_\theta(\mathbf{X}_t, \mathbf{y})}.$$

The bag conditioned forward kernel for i-th node is then induced by the rate $R_t(\cdot,\cdot|\mathcal{B})$:

$$\begin{aligned} p^\theta_{t+\Delta t|t}(\mathbf{x}_{t+\Delta t}^{(i)}|\mathbf{X}_t, \mathcal{B}) &:= \mathbb{E}_{\mathbf{x}_1^{(i)}\sim p^\theta_{1|t}(\cdot|\mathbf{X}_t,\mathcal{B})}\left[p_{t+\Delta t|t}(\mathbf{x}_{t+\Delta t}^{(i)}|\mathbf{X}_t, \mathbf{x}_1^{(i)})\right], \\ &= \mathbb{E}_{\mathbf{x}_1^{(i)}\sim p^\theta_{1|t}(\cdot|\mathbf{X}_t,\mathcal{B})}\left[\delta(\mathbf{x}_t^{(i)}, \mathbf{x}_{t+\Delta t}^{(i)}) + R_t(\mathbf{x}_t^{(i)}, \mathbf{x}_{t+\Delta t}^{(i)}|\mathbf{x}_1^{(i)})\Delta t\right], \\ &= \delta(\mathbf{x}_t^{(i)}, \mathbf{x}_{t+\Delta t}^{(i)}) + \underbrace{\mathbb{E}_{\mathbf{x}_1^{(i)}\sim p^\theta_{1|t}(\cdot|\mathbf{X}_t,\mathcal{B})}\left[R_t(\mathbf{x}_t^{(i)}, \mathbf{x}_{t+\Delta t}^{(i)}|\mathbf{x}_1^{(i)})\right]}_{:=R_t(\mathbf{x}_t^{(i)}, \mathbf{x}_{t+\Delta t}^{(i)}|\mathcal{B})}\Delta t. \end{aligned} \tag{7}$$

Strictly speaking, averaging $p_{1|t,\mathcal{B}}$ over $\mathcal{B}$ does not recover the kernel $p_{1|t}$. It nevertheless provides a practical surrogate that converges to $p_{1|t}$ as the bag size N approaches the fragment-pool size $|\mathcal{F}|$

[44]. When the Euler step size Δt is small and the bag size N is moderately large, the discrepancy is negligible while the computational cost remains manageable. In practice, we replace the expectation with a MC estimation, sampling a single bag only once per Euler step.

Conditional generation While generating valid molecules is essential, steering them toward the desired property is crucial for the practical application of molecular generative models. Direct conditioning via classifier-free guidance [45] offers strong control but tightly binds the generative network to specific properties and often requires retraining when new targets are introduced. Instead, we adopt classifier guidance, steering generation at sampling time with an external property predictor. This decouples the generator from any single conditioning signal and allows the predictor to be trained or updated independently [3, 46].

We begin by defining a property c conditional forward kernel as,

$$p_{t+\Delta t|t}(\mathbf{X}_{t+\Delta t}|\mathbf{X}_t, c) \approx \mathbb{E}_{\mathcal{B}|c}[p(\mathbf{X}_{t+\Delta t}|\mathbf{X}_t, \mathcal{B}, c)]. \tag{8}$$

By Bayes' rule, the in-bag transition kernel could be factorized into the unconditional kernel times a guidance ratio:

$$p(\mathbf{X}_{t+\Delta t}|\mathbf{X}_t, \mathcal{B}, c) = \underbrace{p(\mathbf{X}_{t+\Delta t}|\mathbf{X}_t, \mathcal{B})}_{\text{eq. (7)}} \cdot \underbrace{\frac{p(c|\mathbf{X}_{t+\Delta t}, \mathbf{X}_t, \mathcal{B})}{p(c|\mathbf{X}_t, \mathcal{B})}}_{\text{Guidance ratio}}. \tag{9}$$

Following Vignac et al. [3], specifically for fragment type, we approximate the guidance term in fragment bag $\mathcal{B}$ with a noisy property regressor.

To complete the conditional transition kernel in eq. (8), we need to average it on fragments bag conditioned by c. It is achieved by sampling $\mathcal{B}|c$ with re-weighting the selection probabilities of fragments during the bag sampling process, which is controlled by the guidance parameter $\lambda_{\mathcal{B}}$. A detailed description of the conditional generative kernel and construction of the conditional bag $p(\mathcal{B} \mid c)$ is provided in appendix A.4.

4 Natural Product Generation Benchmark

We now introduce NPGen, a benchmark for molecular generative models through the lens of natural products (NPs), to address demands in both the drug discovery and machine learning communities. NPs are chemical compounds biologically synthesized by organisms, *e.g.*, plants and bacteria. They serve as a valuable resource in drug discovery with unique structural features compared to typical synthetic compounds, often exhibiting more complex ring structures, a higher density of heteroatoms, and more oxygen-based functional groups, which contribute to properties like polarity and potent bio-activity [47, 48]. In essence, NPs occupy a biologically meaningful subset of chemical space. Their structures are often similar to endogenous metabolites, making them highly applicable as templates or direct sources for drugs [49, 50]. Thus, a substantial portion of small-molecule drugs are inspired by or derived from NPs, mimicking their structures or functionalities [51]. In this regard, generating novel molecules that reflect the structural characteristics and biological relevance of NPs represents a key direction in modern drug discovery.

Although modern generative models have demonstrated strong performance on standard benchmarks such as MOSES [18] and GuacaMol [19], these benchmarks are insufficient to evaluate cutting-edge models with respect to the unique structural and biological characteristics of NPs, due to limitations in both the source of data and the metrics they offer. Indeed, the molecules in these datasets are predominantly small and structurally simple, and conventional evaluation metrics—Fréchet ChemNet Distance, scaffold overlap, and KL divergence over simple properties—are approaching saturation, limiting their ability to discriminate between state-of-the-art methods. To fill the critical gap between the current state of benchmarks and the rising need for a new benchmark with challenging, domain-specific alternative tasks [52], we introduce NPGen as a benchmark to generate and evaluate natural product-like molecules.

The dataset for NPGen is constructed from the COCONUT database [53, 54], comprising comprehensive NPs with a rigorous filtering process. Beyond standard metrics like Validity, Uniqueness, and Novelty, we focus on NP-specific characteristics for evaluation. To this end, we assess models with Kullback-Leibler (KL) divergence of distributions of NP-likeness scores [55] and the biosynthetic

pathways and structural classes predicted by NP-Classifier [56] between generated molecules against the test set. Furthermore, we select several representative baselines by classifying modern molecular graph generative approaches with two criteria: generation strategy (autoregressive, one-shot), and representation level (atom, fragment). Specifically, we adopt (1) an atom-based autoregressive model (GraphAF[23]), (2) fragment-based autoregressive models (JT-VAE[12], HierVAE[15]), and (3) an atom-based one-shot model (DiGress[3]). More details about the construction method and statistics of the dataset, training, and inference process for these baseline models are provided in appendix D, and their comparative results are presented in section 5.2.

5 Results

5.1 Standard Molecular Generation Benchmarks

We evaluate FragFM on the MOSES [18] and GuacaMol [19] benchmarks, which focus on small drug-like molecule generation, using the same settings as prior work [3]. Molecular graph generative models can be categorized based on their generation strategy (*i.e.*, autoregressive , one-shot) and representation level (*i.e.*, atom, fragment). We compare FragFM against various models with different generation strategies and representation levels. Refer to appendix C for more details about metrics and baseline models.

The table 1 represents the result of the MOSES benchmark. Note that we report the result with the scaffold-splitted test set, following the previous works[3–5, 29] In the MOSES benchmark, diffusion and flow based models typically underperform autoregressive methods on validity and Fréchet ChemNet Distance (FCD)[57]; by contrast, FragFM achieves nearly 100% validity—on par with the best autoregressive models—and attains an FCD of 0.58, outperforming all one-shot models by a large margin. Furthermore, its exceptionally low FCD and strong property-based metrics (MOSES Filters) make it the first one-shot model to surpass both JT-VAE and GraphINVENT. For other metrics, FragFM lags behind novelty compared to the others, while achieving the state-of-the-art performance in Filters and SNN. On the Scaf metric, FragFM underperforms when restricted to training-set fragments, due to the scaffold-split evaluation: if a test molecule's scaffold isn't present in the fragment bag, it cannot be generated. To prove this, we show the results when generating using the test-set fragment bags on tables 6 and 7. With those fragments, FragFM's scaffold score rises markedly, demonstrating its ability to generalize to unseen fragments via the fragment embedding module. We further discuss the results on the GuacaMol benchmark in appendix E.1.

Table 1: **Molecule generation on MOSES benchmark**. We use 25,000 generated molecules for evaluation. The upper part comprises autoregressive methods, while the second part comprises one-shot methods, including diffusion-based and flow-based methods. Results for FragFM are averaged over three independent runs. The best performance is highlighted in **bold**, and the second-best is underlined.

Model	Rep. Level	Valid ↑	Unique ↑	Novel ↑	Filters ↑	FCD ↓	SNN ↑	Scaf ↑
Training set	-	100.0	100.0	-	100.0	0.48	0.59	0.0
GraphINVENT [22]	Atom	96.4	99.8	-	95.0	1.22	0.54	12.7
JT-VAE [12]	Fragment	**100.0**	**100.0**	**99.9**	97.8	1.00	0.53	10.0
DiGress [3]	Atom	85.7	**100.0**	95.0	97.1	1.19	0.52	14.8
DisCo [29]	Atom	88.3	**100.0**	97.7	95.6	1.44	0.50	15.1
Cometh [5]	Atom	90.5	100.0	96.4	97.2	1.44	0.51	15.9
Cometh-PC [5]	Atom	90.5	99.9	92.6	**99.1**	1.27	0.54	**16.0**
DeFoG [4]	Atom	92.8	99.9	92.1	98.9	1.95	0.55	14.4
FragFM (ours)	Fragment	99.8	**100.0**	87.1	**99.1**	**0.58**	**0.56**	10.9

5.2 Natural Product Molecule Generation Benchmark

Next, we compare FragFM in our proposed Natural Product Generation benchmark (NPGen), against the baseline models introduced in section 4.

Table 2 shows the overall benchmark results for molecular graph generative models including FragFM. FragFM demonstrates the best performance on functionality-driven metrics, which are specifically

tailored to assess the generation of natural products in the design of NPGen. This result highlights FragFM's outstanding ability to capture the intricate structural features of natural product molecules, compared to the baseline models. It is worth noting that fragment-based representation generally leads to better performance in functionality-driven metrics than atom-based ones for autoregressive and one-shot generation strategies. This emphasizes the importance of fragments in molecular graph generation, providing a rigorous basis for the success of FragFM. We present visualizations of generated molecules for all baselines and discuss them in appendix F.3.

Table 2: **Molecule generation results on NPGen**. We use 30,000 generated molecules for evaluation. The upper part comprises autoregressive methods, while the second part comprises one-shot methods, including diffusion-based and flow-based methods. The results are averaged over three runs. The best performance is shown in **bold**, and the second-best is underlined.

Model	Rep. Level	Val. ↑	Unique. ↑	Novel ↑	NP Score KL Div. ↓	NP Class KL Div. ↓ Pathway	Superclass	Class	FCD ↓
Training set	-	100.0	100.0	-	0.0006	0.0002	0.0028	0.0094	0.01
GraphAF [23]	Atom	79.1	63.6	95.6	0.8546	0.9713	3.3907	6.6905	25.11
JT-VAE [12]	Fragment	**100.0**	97.2	99.5	0.5437	0.1055	1.2895	2.5645	4.07
HierVAE [15]	Fragment	**100.0**	81.5	97.7	0.3021	0.4230	0.5771	1.4073	8.95
DiGress [3]	Atom	85.4	**99.7**	**99.9**	0.1957	0.0229	0.3370	1.0309	2.05
FragFM (ours)	Fragment	98.0	99.0	95.4	**0.0374**	**0.0196**	**0.1482**	**0.3570**	**1.34**

5.3 Sampling Efficiency

Iterative denoising in stochastic generative models inherently involves a tradeoff between sampling steps and quality; for higher quality, multiple denoising steps are required, resulting in prohibitively slow sampling.

Figure 2 and table 8 shows MOSES benchmark results as we progressively reduce the number of denoising iterations across several diffusion- and flow-based generators. As expected, most models suffer significant validity, filters, and FCD drops at low sampling steps. In contrast, FragFM remains remarkably robust, achieving 95% validity and FCD under 1.0, outperforming atom-based models even when they use far more steps. This robustness arises from our fragment-level discrete flow matching, which reduces the per-step complexity of predicting individual atoms and bonds but operates on meaningful substructures. Additional sampling efficiency and runtime analyses are provided in appendix E.4.

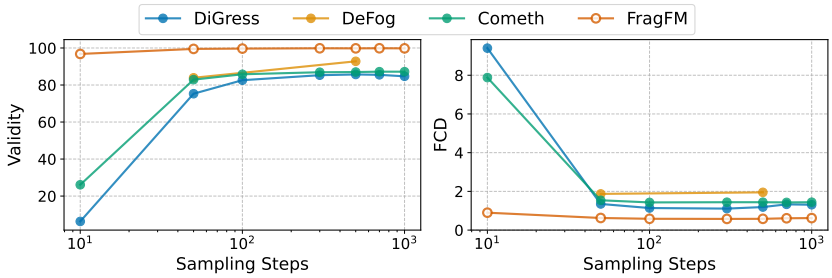


Figure 2: **Analysis of sampling steps across multiple denoising models.** FragFM maintains higher sampling quality than baseline atom-based denoising models as the number of sampling steps is reduced, exhibiting significantly less performance degradation. Additional results are provided in appendix E.4.

5.4 Conditional Generation

To further asses FragFM's conditional generation, we conducted conditional generation experiment using both classifier guidance (section 3.4) and fragment bag reweighting (appendix E.3), and compared against DiGress—an atom-based model with classifier guidance. Intrinsically, as guidance strength increases, generated molecules are driven closer to the target region, reducing mean absolute

error (MAE) but increasing FCD due to distributional shift. We thus investigate the MAE-FCD plot, which shows that curves nearer the lower-left corner (low MAE and FCD) indicate a superior balance of property accuracy and generative fidelity. We evaluate simple properties (logP, number of rings, QED [58]) and more challenging docking scores for FA7, JAK2, PARP1. Additional experimental details are provided in appendix E.5 with an intuitive interpretation of the chemical aspect.

As illustrated by the QED conditioning results (fig. 3), FragFM achieves a significantly more optimal performance curve, exhibiting lower FCD and MAE values across the explored trade-off. This enhanced performance extends to other fundamental molecular properties such as logP and number of rings (see appendix E.5 for more results). A compelling advantage of the fragment-based strategy employed by FragFM is its ability to maintain high molecular validity even under stringent property conditioning. Crucially, FragFM's fragment-based strategy preserves high molecular validity even under aggressive conditioning. In the validity-MAE plots (figs. 11b and 12b), FragFM maintains its validity over 95% despite lower MAE, while the atom-based model often undergoes steep drops while increasing the level of guidance.

Furthermore, we verify that the fragment-based approach possesses additional flexibility and better controlability with fragment bag conditioning by $\lambda_{\mathcal{B}}$. This is particularly evident in the JAK2 docking score conditioning task (fig. 4), where FragFM outperforms its atom-based counterpart even without explicit fragment bag guidance (*i.e.*, $\lambda_{\mathcal{B}} = 0$). Subsequently, increasing $\lambda_{\mathcal{B}}$ consistently shifts FragFM's curve towards a more optimal region, demonstrating that fragment bag reweighting also significantly enhances property control. Our approach thus aligns with and extends the long-standing success of fragment-based paradigms in medicinal chemistry [9, 59] and recent computational strategies [60], by providing a robust framework where the model learns to compose optimal fragments for targeted generation.

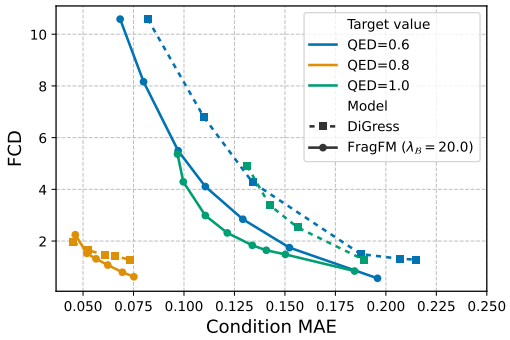


Figure 3: **Property conditioning results for QED**. MAE-FCD curves for FragFM and DiGress under QED conditioning on the MOSES dataset, with different conditioning values color-coded.

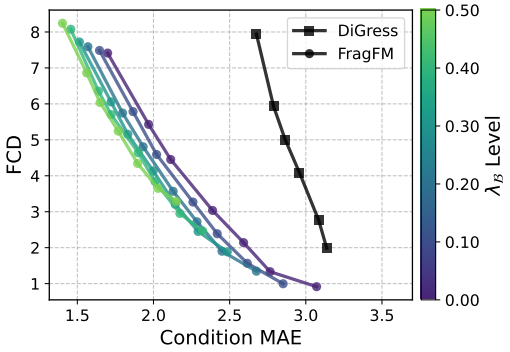


Figure 4: **Effect of $\lambda_{\mathcal{B}}$ in conditioning**. MAE-FCD curves on the ZINC250K dataset under JAK2 docking score conditioning to -11.0 kcal/mol. Different $\lambda_{\mathcal{B}}$ is color-coded.

6 Conclusion

In this paper, we have introduced FragFM, a novel hierarchical framework with fragment-level discrete flow matching followed by lossless reconstruction of the atom-level graph, for efficient molecular graph generation. To this end, we proposed a stochastic fragment bag strategy with a coarse-to-fine autoencoder to circumvent dependency on a limited fragment library cost-effectively. Standing on long-standing fragment-based strategies in chemistry, FragFM showed superior performance on the standard molecular generative benchmarks compared to the previous graph generative models. Additionally, applying classifier guidance at the fragment level and conditioning the fragment bag on the target property enables more precise control over diverse molecular properties. These significant improvements pave the way for a new frontier for fragment-based denoising approaches in molecular graph generation. Finally, to contribute to the growth of the molecular graph-generating domain, we developed a new benchmark for evaluating models of natural products, which is also crucial in drug discovery.

References

[1] Yunhui Jang, Dongwoo Kim, and Sungsoo Ahn. Hierarchical graph generation with k^2-trees. In ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling, 2023.

[2] 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, pages 10362–10383. PMLR, 2022.

[3] Clement 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.

[4] Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. arXiv preprint arXiv:2410.04263, 2024.

[5] Antoine Siraudin, Fragkiskos D Malliaros, and Christopher Morris. Cometh: A continuous-time discrete-state graph diffusion model. arXiv preprint arXiv:2406.06449, 2024.

[6] Floor Eijkelboom, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. arXiv preprint arXiv:2406.04843, 2024.

[7] Yiming Qin, Clement Vignac, and Pascal Frossard. Sparse training of discrete diffusion models for graph generation. arXiv preprint arXiv:2311.02142, 2023.

[8] Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. arXiv preprint arXiv:2305.04111, 2023.

[9] Philip J Hajduk and Jonathan Greer. A decade of fragment-based drug design: strategic advances and lessons learned. Nature reviews Drug discovery, 6(3):211–219, 2007.

[10] Diane Joseph-McCarthy, Arthur J Campbell, Gunther Kern, and Demetri Moustakas. Fragment-based lead discovery and design. Journal of chemical information and modeling, 54(3):693–704, 2014.

[11] Philine Kirsch, Alwin M Hartman, Anna KH Hirsch, and Martin Empting. Concepts and core principles of fragment-based drug design. Molecules, 24(23):4309, 2019.

[12] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In International conference on machine learning, pages 2323–2332. PMLR, 2018.

[13] Seonghwan Seo, Jaechang Lim, and Woo Youn Kim. Molecular generative model via retrosynthetically prepared chemical building block assembly. Advanced Science, 10(8):2206674, 2023.

[14] Leon Hetzel, Johanna Sommer, Bastian Rieck, Fabian Theis, and Stephan Günnemann. Magnet: Motif-agnostic generation of molecules from shapes. arXiv preprint arXiv:2305.19303, 2023.

[15] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In International conference on machine learning, pages 4839–4848. PMLR, 2020.

[16] Daniel Levy and Jarrid Rector-Brooks. Molecular fragment-based diffusion model for drug discovery. In ICLR 2023-Machine Learning for Drug Discovery workshop, 2023.

[17] Zijun Chen, Yu Wang, Liuzhenghao Lv, Hao Li, Zongying Lin, Li Yuan, and Yonghong Tian. Multi-granularity score-based generative framework enables efficient inverse design of complex organics. arXiv preprint arXiv:2409.07912, 2024.

[18] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (moses): a benchmarking platform for molecular generation models. Frontiers in pharmacology, 11:565644, 2020.

[19] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. Journal of chemical information and modeling, 59(3):1096–1108, 2019.

[20] Atanas G Atanasov, Sergey B Zotchev, Verena M Dirsch, and Claudiu T Supuran. Natural products in drug discovery: advances and opportunities. Nature reviews Drug discovery, 20(3):200–216, 2021.

[21] Jaechang Lim, Sang-Yeon Hwang, Seokhyun Moon, Seungsu Kim, and Woo Youn Kim. Scaffold-based molecular design with a graph generative model. Chemical science, 11(4):1153–1164, 2020.

[22] Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph networks for molecular design. Machine Learning: Science and Technology, 2(2):025023, 2021.

[23] Chence Shi*, Minkai Xu*, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=S1esMkHYPr.

[24] Youngchun Kwon, Dongseon Lee, Youn-Suk Choi, Kyoham Shin, and Seokho Kang. Compressed graph representation for scalable molecular graph generation. Journal of Cheminformatics, 12:1–8, 2020.

[25] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33:6840–6851, 2020.

[26] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456, 2020.

[27] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In International Conference on Artificial Intelligence and Statistics, pages 4474–4484. PMLR, 2020.

[28] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. Advances in Neural Information Processing Systems, 34:17981–17993, 2021.

[29] Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. arXiv preprint arXiv:2405.11416, 2024.

[30] Jun Hyeong Kim, Seonghwan Kim, Seokhyun Moon, Hyeongwoo Kim, Jeheon Woo, and Woo Youn Kim. Discrete diffusion schr\" odinger bridge matching for graph transformation. arXiv preprint arXiv:2410.01500, 2024.

[31] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. arXiv preprint arXiv:2210.02747, 2022.

[32] Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. arXiv preprint arXiv:2402.04997, 2024.

[33] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. arXiv preprint arXiv:2407.15595, 2024.

[34] Xiaoyang Hou, Tian Zhu, Milong Ren, Dongbo Bu, Xin Gao, Chunming Zhang, and Shiwei Sun. Improving molecular graph generation with flow matching and optimal transport. arXiv preprint arXiv:2411.05676, 2024.

[35] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In International conference on machine learning, pages 4849–4859. PMLR, 2020.

[36] Xiangzhe Kong, Wenbing Huang, Zhixing Tan, and Yang Liu. Molecule generation by principal subgraph mining and assembling. Advances in Neural Information Processing Systems, 35:2550–2563, 2022.

[37] Krzysztof Maziarz, Henry Jackson-Flux, Pashmina Cameron, Finton Sirockin, Nadine Schneider, Nikolaus Stiefl, Marwin Segler, and Marc Brockschmidt. Learning to extend molecular scaffolds with structural motifs. arXiv preprint arXiv:2103.03864, 2021.

[38] Jorg 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, 2008.

[39] Tairan Liu, Misagh Naderi, Chris Alvin, Supratik Mukhopadhyay, and Michal Brylinski. Break down in order to build up: decomposing small molecules for fragment-based drug design with e molfrag. Journal of chemical information and modeling, 57(4):627–631, 2017.

[40] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. Advances in neural information processing systems, 32, 2019.

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10684–10695, 2022.

[42] Bo Qiang, Yuxuan Song, Minkai Xu, Jingjing Gong, Bowen Gao, Hao Zhou, Wei-Ying Ma, and Yanyan Lan. Coarse-to-fine: a hierarchical diffusion model for molecule generation in 3d. In International Conference on Machine Learning, pages 28277–28299. PMLR, 2023.

[43] Jack Edmonds. Paths, trees, and flowers. Canadian Journal of mathematics, 17:449–467, 1965.

[44] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018.

[45] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598, 2022.

[46] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. Advances in neural information processing systems, 34:8780–8794, 2021.

[47] Miklos Feher and Jonathan M Schmidt. Property distributions: differences between drugs, natural products, and molecules from combinatorial chemistry. Journal of chemical information and computer sciences, 43 (1):218–227, 2003.

[48] Christopher F Stratton, David J Newman, and Derek S Tan. Cheminformatic comparison of approved drugs from natural product versus synthetic origins. Bioorganic & medicinal chemistry letters, 25(21): 4802–4807, 2015.

[49] Asmaa Boufridi and Ronald J Quinn. Harnessing the properties of natural products. Annual review of pharmacology and toxicology, 58(1):451–470, 2018.

[50] Josefin Rosén, Johan Gottfries, Sorel Muresan, Anders Backlund, and Tudor I Oprea. Novel chemical space exploration via natural products. Journal of medicinal chemistry, 52(7):1953–1962, 2009.

[51] Atanas G Atanasov, Sergey B Zotchev, Verena M Dirsch, and Claudiu T Supuran. Natural products in drug discovery: advances and opportunities. Nature reviews Drug discovery, 20(3):200–216, 2021.

[52] Maya Bechler-Speicher, Ben Finkelshtein, Fabrizio Frasca, Luis Müller, Jan Tönshoff, Antoine Siraudin, Viktor Zaverkin, Michael M Bronstein, Mathias Niepert, Bryan Perozzi, et al. Position: Graph learning will lose relevance due to poor benchmarks. arXiv preprint arXiv:2502.14546, 2025.

[53] Maria Sorokina, Peter Merseburger, Kohulan Rajan, Mehmet Aziz Yirik, and Christoph Steinbeck. Coconut online: collection of open natural products database. Journal of Cheminformatics, 13(1):2, 2021.

[54] Venkata Chandrasekhar, Kohulan Rajan, Sri Ram Sagar Kanakam, Nisha Sharma, Viktor Weißenborn, Jonas Schaub, and Christoph Steinbeck. Coconut 2.0: a comprehensive overhaul and curation of the collection of open natural products database. Nucleic Acids Research, 53(D1):D634–D643, 2025.

[55] Peter Ertl, Silvio Roggo, and Ansgar Schuffenhauer. Natural product-likeness score and its application for prioritization of compound libraries. Journal of chemical information and modeling, 48(1):68–74, 2008.

[56] Hyun Woo Kim, Mingxun Wang, Christopher A Leber, Louis-Félix Nothias, Raphael Reher, Kyo Bin Kang, Justin JJ Van Der Hooft, Pieter C Dorrestein, William H Gerwick, and Garrison W Cottrell. Npclassifier: a deep neural network-based structural classification tool for natural products. Journal of Natural Products, 84(11):2795–2807, 2021.

[57] Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Gunter Klambauer. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. Journal of chemical information and modeling, 58(9):1736–1741, 2018.

[58] 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.

[59] Arman A Sadybekov, Anastasiia V Sadybekov, Yongfeng Liu, Christos Iliopoulos-Tsoutsouvas, Xi-Ping Huang, Julie Pickett, Blake Houser, Nilkanth Patel, Ngan K Tran, Fei Tong, et al. Synthon-based ligand discovery in virtual libraries of over 11 billion compounds. Nature, 601(7893):452–459, 2022.

[60] Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Saee Paliwal, Arash Vahdat, and Weili Nie. Molecule generation with fragment retrieval augmentation. arXiv preprint arXiv:2411.12078, 2024.

[61] Linqi Zhou, Aaron Lou, Samar Khanna, and Stefano Ermon. Denoising diffusion bridge models. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=FKksTayvGo.

[62] Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion schrödinger bridge matching. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=qy07OHsJT5.

[63] Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. arXiv preprint arXiv:2406.01572, 2024.

[64] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In International conference on machine learning, pages 1263–1272. PMLR, 2017.

[65] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 24174–24184, 2024.

[66] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826, 2018.

[67] Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In International Conference on Machine Learning, pages 23321–23337. PMLR, 2023.

[68] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. Journal of chemical information and modeling, 50(5):742–754, 2010.

[69] Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. Journal of medicinal chemistry, 39(15):2887–2893, 1996.

[70] Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. Chemical science, 10(12):3567–3572, 2019.

[71] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426, 2018.

[72] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. Journal of chemical information and modeling, 50(5):742–754, 2010.

[73] Soojung Yang, Doyeong Hwang, Seul Lee, Seongok Ryu, and Sung Ju Hwang. Hit and lead discovery with explorative rl and fragment-based molecule generation. Advances in Neural Information Processing Systems, 34:7924–7936, 2021.

[74] David Ryan Koes, Matthew P Baumgartner, and Carlos J Camacho. Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise. Journal of chemical information and modeling, 53 (8):1893–1904, 2013.

[75] Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. Journal of computational chemistry, 31(2): 455–461, 2010.

[76] Joseph B Sweeney. Aziridines: epoxides' ugly cousins? Chemical Society Reviews, 31(5):247–258, 2002.

[77] Barry K Carpenter. Heavy-atom tunneling as the dominant pathway in a solution-phase reaction? bond shift in antiaromatic annulenes. Journal of the American Chemical Society, 105(6):1700–1701, 1983.

[78] Keiichi Nishiyama, Ninoru Sakiyama, Syûzô Seki, Hisanori Horita, Tetsuo Otsubo, and Soichi Misumi. Thermochemical studies on double-and triple-layered [2.2] paracyclophanes. estimation of molecular strain energies. Bulletin of the Chemical Society of Japan, 53(4):869–877, 1980.

A Method Details

A.1 Coarse-to-fine Autoencoder

We adopted a KL-regularized autoencoder for coarse-to-fine graph conversion. First, each molecule is decomposed into fragments using the BRICS decomposition rules [38], producing a fragment-level graph $\mathcal{G}$. During this fragmentation, connectivity information between atoms belonging to different fragments is discarded, *i.e.*, the orientation of an antisymmetric fragment with respect to other fragments cannot be determined by the fragments alone. The fragment-level graph and the missing information, encoded in the latent variable z, are required to reconstruct the original atom-level graph. Formally, the encoding and decoding process is defined as:

$$\mathcal{G} = \text{Fragmentation}(G), \tag{10}$$

$$z \sim q_\theta = \mathcal{N}(\text{Encoder}(G;\theta), \sigma), \tag{11}$$

$$\hat{E} = \text{Decoder}(\mathcal{G}, z; \theta). \tag{12}$$

The decoder reconstructs only those atom-level edges $\hat{E}$ corresponding to the fragment connectivity in the coarse representation. $\hat{E}$ is then used to reconstruct an atom-level graph $\mathbf{G}$ as explained in appendix A.2.

We optimize the autoencoder using a reconstruction loss to ensure that the reconstructed graph faithfully preserves the original molecular structure. Additionally, we introduce a small KL regularization term to the training loss for a latent variable to enforce a well-structured and unscaled latent space:

$$\mathcal{L}_{\text{VAE}}(\theta) = \mathbb{E}_{G\sim p_{\text{data}}}\left[\mathcal{L}_{\text{CE}}\left(E, \hat{E}(\theta)\right) + \beta D_{\text{KL}}\left(q_\theta(z|G) \parallel p(z)\right)\right]. \tag{13}$$

We set a low regularization coefficient of $\beta = 0.0001$ to maintain high-fidelity reconstruction.

A.2 Atom-level Reconstruction of Fragment-level Graph

We utilize the Blossom algorithm [43] to determine the optimal matching in the atom-level connectivity given coarse-to-fine decoder output. The Blossom algorithm is an optimization technique used to find the maximum matching in general graphs by iteratively contracting and expanding odd-length cycles (blossoms) to identify augmenting paths efficiently. We leverage this algorithm in our framework to accurately reconstruct atom-level connectivity from fragment-level graphs, ensuring chemically valid molecular structures. The algorithm takes as input the matching nodes V_m, edges E_m, and edge weights w_{ij}. Once the fragment-level graph and the probabilities of atom-level edges from the coarse-to-fine autoencoder are computed, we define $V_m \subseteq \hat{V}$ as the set of junction atoms in fragment graphs, which are marked as $*$ in fig. 1, and E_m as the set of connections between junction atoms belonging to connected fragments.

Formally, an edge e_{kl} exists in E_m if the corresponding atoms belong to different fragments that are connected in the fragment-level graph, expressed as:

$$e_{kl} \in E_m \quad \text{if} \quad v_k \in \hat{V}_i, \quad v_l \in \hat{V}_j, \quad \text{and} \quad \varepsilon_{ij} \in \mathcal{E}. \tag{14}$$

The edge weights w_{ij} correspond to the predicted log probability of each connection obtained from the coarse-to-fine autoencoder. The Blossom algorithm is then applied to solve the maximum weighted matching problem, formulated as

$$M^* = \text{argmax}_{M\subseteq E_m} \sum_{(i,j)\in M} w_{ij}. \tag{15}$$

Here, M^* represents the optimal set of fragment-level connections that best reconstructs atom-level connectivity, maximizing the joint probability of the autoencoder prediction. Although the algorithm has an $O(N^3)$ complexity for N fragment junctions, its computational cost remains negligible in our case, as the number of fragment junctions is relatively small compared to the total number of atoms in a molecule.

A.3 Details of Flow Modeling

A.3.1 Flow modeling for edge

Let $\varepsilon^{ij} \in \mathcal{E} := \{0, 1\}$ denote the absence (0) or presence (1) of an edge between the i-th and j-th fragments in the coarse graph. Because the edge state is binary, we adopt a uniform prior, $p_0(\varepsilon) = \frac{1}{2}$.

Following the discrete flow-matching (DFM) recipe, the ε_1-conditioned time marginal is

$$p_{t\,|1}\big(\varepsilon_t \mid \varepsilon_1\big) = t\,\delta\big(\varepsilon_t, \varepsilon_1\big) + (1-t)\,p_0(\varepsilon_t), \qquad t \in [0, 1]. \tag{16}$$

The CTMC rate that realizes this marginal is

$$R_t\big(\varepsilon, \varepsilon' \mid \varepsilon_1\big) = \frac{\mathrm{ReLU}\Big(\partial_t p_{t\,|1}(\varepsilon' \mid \varepsilon_1) - \partial_t p_{t\,|1}(\varepsilon \mid \varepsilon_1)\Big)}{2\,p_{t\,|1}(\varepsilon \mid \varepsilon_1)}, \qquad \forall\, \varepsilon \neq \varepsilon', \tag{17}$$

$$R_t(\varepsilon, \varepsilon') = \mathbb{E}_{\varepsilon_1 \sim p_{1\,|t}(\cdot|\varepsilon)}\big[R_t(\varepsilon, \varepsilon' \mid \varepsilon_1)\big]. \tag{18}$$

The posterior $p_{1\,|t}(\varepsilon_1^{(ij)} \mid \mathbf{X}_t)$ is parameterized by a neural network $\phi_\theta^{\text{edge}}(\mathbf{X}_t)_{ij}$. We trained the model by minimizing the cross-entropy loss:

$$\mathcal{L}_{\text{edge}} = \sum_{ij} \mathbb{E}_{\mathbf{X}_1, \mathbf{X}_t, t}\Big[-\varepsilon_1^{ij} \log \phi_\theta^{\text{edge}}(\mathbf{X}_t)_{ij} - (1-\varepsilon_1^{ij}) \log\Big(1 - \phi_\theta^{\text{edge}}(\mathbf{X}_t)_{ij}\Big)\Big]. \tag{19}$$

In the sampling phase, the neural network replaces the posterior in eq. (18). Because $|\mathcal{E}| = 2$, the expectation above is computed exactly—no Monte-Carlo sampling is required. Thus, the forward kernel for i, j-th edge is as:

$$p^\theta_{t+\Delta t|t}\big(\varepsilon^{(i,j)}_{t+\Delta t} \mid \mathbf{X}_t\big) = \delta\big(\varepsilon^{(i,j)}_t, \varepsilon^{(i,j)}_{t+\Delta t}\big) + R^\phi_t\big(\varepsilon^{(i,j)}_t, \varepsilon^{(i,j)}_{t+\Delta t}\big)\,\Delta t. \tag{20}$$

A.3.2 Flow modeling for latent vector

Let $z \in \mathbb{R}^d$ be the continuous latent vector attached to the fragment-level graph. We model its evolution with conditional flow matching (CFM [31]), which views generation as integrating an ODE whose time-dependent velocity field (VF) is learned from data. Specifically, we linearly change the mean and standard deviation $\mu_t(x) = tz_1$ and $\sigma_t(z) = 1 - t$. The corresponding conditioned target VF is

$$u_t(z_t|z_1) = \frac{z_1 - z_t}{1-t}. \tag{21}$$

Then, the trajectory for a prior sample ($z_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$) and a data sample ($z_1 \sim p_1(z)$) under the target VF, *i.e.*, the solution to $\frac{dz_t}{dt} = u_t(z_t|z_1)$ with z_0 is given by:

$$z_t = (1-t)z_0 + tz_1, \qquad t \in [0, 1]. \tag{22}$$

We fit a neural vector field $v_\theta(\mathbf{X}_t)$ by minimizing the mean-squared error

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{\mathbf{X}_1, \mathbf{X}_t, t}\Big[\big\| v_\theta(\mathbf{X}_t) - \frac{z_1 - z_t}{1-t} \big\|_2^2\Big]. \tag{23}$$

To generate a latent vector, we solve the ODE

$$\frac{\mathrm{d}\hat{z}_t}{\mathrm{d}t} = v_\theta(\mathbf{X}_t), \tag{24}$$

forward from $t = 0$ to $t = 1$ with a deterministic solver. The resulting $\hat{z}_1$ is then fed to the coarse-to-fine decoding network to obtain atom-level graph.

CFM as a Limiting Case of a VE Diffusion Bridge Unlike diffusion models, which first define a reference process and then learn its drift, CFM directly prescribes the time-marginal distribution and optimizes the corresponding velocity fields that "point" toward a fixed data point. This raises a question: How can we treat CFM with the transition kernel $p_{t+\Delta t|t}(z_{t+\Delta t} \mid z_t)$ used in diffusion models?

A diffusion bridge is a reference diffusion process conditioned to hit a fixed end-point. Its SDE is

$$\mathrm{d}z_t = \Big[\mathbf{f}(z_t, t) + g^2(t)\,\nabla_{z_t} \log Q(z_T \mid z_t)\big|_{z_T=y}\Big]\mathrm{d}t \;+\; g(t)\,\mathrm{d}\mathbf{w}_t, \tag{25}$$

where $Q(z_T \mid z_t)$ is the unconditioned transition kernel of the reference process, $\mathbf{f}$ its drift, and g its diffusion coefficients.

If the reference process is the variance-exploding (VE) diffusion $\mathrm{d}z_t = g(t)\,\mathrm{d}\mathbf{w}_t$, Zhou et al. [61] show that (25) reduces to

$$\mathrm{d}z_t = \frac{\mathrm{d}\sigma_t^2/\mathrm{d}t}{\sigma_T^2 - \sigma_t^2}\,(z_T - z_t)\,\mathrm{d}t \;+\; g(t)\,\mathrm{d}\mathbf{w}_t. \tag{26}$$

Setting $T = 1$ and $\sigma_t^2 = c^2 t$ (constant c) gives

$$\mathrm{d}z_t = \frac{z_1 - z_t}{1 - t}\,\mathrm{d}t \;+\; c\,\mathrm{d}\mathbf{w}_t. \tag{27}$$

Taking a limit of $c \to 0$ eliminates the stochastic term and leaves the deterministic drift

$$\frac{\mathrm{d}z_t}{\mathrm{d}t} = \frac{z_1 - z_t}{1 - t}, \tag{28}$$

which is exactly the velocity field optimized by CFM, *i.e.*, the VE diffusion bridge collapses to CFM.

Given a coupling $\pi(z_0, z_1)$, we can form a mixture bridge Π by averaging the pinned-down trajectories over π. According to Proposition 2 of Shi et al. [62], its Markov approximation M satisfies

$$\mathrm{d}z_t = \mathbb{E}_{1|t}\big[\frac{z_1 - z_t}{1 - t}\big]\mathrm{d}t \;+\; c\,\mathrm{d}\mathbf{w}_t, \quad \text{with } M_t = \Pi_t\ \forall t. \tag{29}$$

When $c \to 0$, the drift term above coincides with the averaged velocity field learned by CFM eq. (23), confirming that M recovers the CFM dynamics in the zero-noise limit.

A.3.3 Training Details

Our objective is to learn a generative diffusion on the coarse graph state[1] by combining the node-type Info-NCE loss, the edge binary-cross-entropy loss, and the latent CFM loss into a single training objective.

Sampling a training triple $(\mathbf{X}_1, \mathbf{X}_t, t)$.

1. **Data endpoint.** Sample a atomistic graph $\mathbf{G}_1$ from the molecular dataset, and apply coarse-to-fine encoder to obtain $\mathbf{X}_1 = (\mathcal{G}_1, z_1)$.
2. **Time sampling.** Sample a time $t \in [0, 1]$ from uniform distribution.
3. **Forward noise.** Independently transform each component to its noised counterpart:
   - **Node.** For every fragment indexed with i, sample $\mathbf{x}_t^{(i)} \sim p_{t\,|1}(\cdot \mid \mathbf{x}_1^{(i)})$ with the masked prior.
   - **Edge.** For each pair (i, j), draw $\varepsilon_t^{(ij)} \sim p_{t\,|1}(\cdot \mid \varepsilon_1^{(ij)})$ using eq. (16).
   - **Latent.** Sample $z_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and set $z_t = (1 - t)z_0 + tz_1$ as in eq. (22).
4. **Construct $\mathbf{X}_t$.** Collect the three noised components into $\mathbf{X}_t = (\mathcal{G}_t, z_t)$.

Joint loss.

$$\mathcal{L}_{\text{node}}(\theta; \mathcal{B}) = -\log \frac{f_\theta(\mathbf{X}_t, x_1)}{\sum_{y \in \mathcal{B}} f_\theta(\mathbf{X}_t, y)}, \tag{6}$$

$$\mathcal{L}_{\text{edge}}(\theta) = \sum_{i<j}\Big[-\varepsilon_1^{ij} \log \phi_\theta^{\text{edge}}(\mathbf{X}_t)_{ij} - (1 - \varepsilon_1^{ij})\log\big(1 - \phi_\theta^{\text{edge}}(\mathbf{X}_t)_{ij}\big)\Big], \tag{19}$$

$$\mathcal{L}_{\text{latent}}(\theta) = \big\|\mathbf{v}_\theta^{\text{latent}}(\mathbf{X}_t) - (z_1 - z_0)\big\|_2^2. \tag{23}$$

[1] Recall $\mathbf{X}_t = \big(\mathcal{G}_t, z_t\big)$ with $\mathcal{G}_t = (\{\mathbf{x}_t^{(i)}\}, \{\varepsilon_t^{(ij)}\})$ —the node-type vector $\mathbf{x}_t^{(i)}$, binary edge matrix $\varepsilon_t^{(ij)}$, and latent vector $z_t \in \mathbb{R}^d$.

We minimize the weighted sum

$$\mathcal{L}_{\text{total}}(\theta;\mathcal{B}) \;=\; \mathcal{L}_{\text{node}}(\theta;\mathcal{B}) + \alpha_{\text{edge}}\,\mathcal{L}_{\text{edge}}(\theta) + \alpha_{\text{latent}}\,\mathcal{L}_{\text{latent}}(\theta), \tag{30}$$

with $\alpha_{\text{edge}}, \alpha_{\text{latent}} > 0$. We fix $\alpha_{\text{edge}} = 5.0$ and $\alpha_{\text{latent}} = 1.0$ for all of our experiments.

A.4 Conditional Generation with Fragment Bag

Guidance ratio modeling The guidance ratio in eq. (9) can be written by:

$$\begin{aligned}\frac{p(c|\mathbf{X}_{t+\Delta t}, \mathbf{X}_t, \mathcal{B})}{p(c|\mathbf{X}_t, \mathcal{B})} &= \frac{p(c|\mathbf{X}_{t+\Delta t}, \mathcal{B})}{p(c|\mathbf{X}_t, \mathcal{B})}, \\ &= \frac{p(c|\mathbf{X}_{t+\Delta t})p(\mathcal{B}|c, \mathbf{X}_{t+\Delta t})/p(\mathcal{B}|\mathbf{X}_{t+\Delta t})}{p(c|\mathbf{X}_t)p(\mathcal{B}|c, \mathbf{X}_t)/p(\mathcal{B}|\mathbf{X}_t)}.\end{aligned}$$

The ratio $\frac{p(\mathcal{B}|c,\mathbf{X}_{t+\Delta t})}{p(\mathcal{B}|\mathbf{X}_{t+\Delta t})}\frac{p(\mathcal{B}|\mathbf{X}_t)}{p(\mathcal{B}|c,\mathbf{X}_t)}$ is intractable, yet it involves the difference between two consecutive states $\mathbf{X}_t$ and $\mathbf{X}_{t+\Delta t}$. Because an Euler step is very small, it can be assumed that the diffusion state evolves smoothly: $\mathbf{X}_{t+\Delta t} = \mathbf{X}_t + \mathcal{O}(\Delta t)$. If the bag-sampling distributions $p(\mathcal{B} \mid \mathbf{X})$ and $p(\mathcal{B} \mid c, \mathbf{X})$ vary continuously with $\mathbf{X}$, a first-order Taylor expansion yields

$$p(\mathcal{B} \mid \cdot, \mathbf{X}_{t+\Delta t}) = p(\mathcal{B} \mid \cdot, \mathbf{X}_t) + \mathcal{O}(\Delta t), \tag{31}$$

so the whole ratio is approximately 1, to be

$$\frac{p(c|\mathbf{X}_{t+\Delta t}, \mathbf{X}_t, \mathcal{B})}{p(c|\mathbf{X}_t, \mathcal{B})} \approx \frac{p(c|\mathbf{X}_{t+\Delta t})}{p(c|\mathbf{X}_t)}. \tag{32}$$

Following Vignac et al. [3], Nisonoff et al. [63], we can estimate the ratio via noisy predictor $\hat{c}(\mathbf{X}_t)$ with 1st order Taylor expansion, yielding

$$\begin{aligned}\log \frac{p(c|\mathbf{X}_{t+\Delta t})}{p(c|\mathbf{X}_t)} &\approx \langle \nabla_{\mathbf{X}_t} \log p(c|\mathbf{X}_t), \mathbf{X}_{t+\Delta t} - \mathbf{X}_t \rangle, \\ &\approx \sum_i \langle \nabla_{\mathbf{x}_t^{(i)}} \log p(c|\mathbf{X}_t), \mathbf{x}_{t+\Delta t}^{(i)} \rangle + \sum_{ij} \langle \nabla_{\varepsilon_t^{(ij)}} \log p(c|\mathbf{X}_t), \varepsilon_{t+\Delta t}^{(ij)} \rangle \\ &\quad + \langle \nabla_{z_t} \log p(c|\mathbf{X}_t), z_{t+\Delta t} - z_t \rangle + C.\end{aligned}$$

In practice, we estimate $p(c|\mathbf{X_t})$ by Gaussian modeling with a time conditioned noisy classifier parameterized by ψ, $\mathcal{N}(c; \mu(\mathbf{X}_t, t; \psi), \sigma^2)$. Thus, the guidance term is written as:

$$\begin{aligned}\frac{p(c|\mathbf{X}_{t+\Delta t})}{p(c|\mathbf{X}_t)} \propto \exp(\lambda_{\mathbf{X}} \sum_i \langle \nabla_{\mathbf{x}_t^{(i)}} \|\mu(\mathbf{X}_t, t) - c\|^2, \mathbf{x}_{t+\Delta t}^{(i)} \rangle) \\ \times \exp(\lambda_{\mathbf{X}} \sum_{ij} \langle \nabla_{\varepsilon_t^{(ij)}} \|\mu(\mathbf{X}_t, t) - c\|^2, \varepsilon_{t+\Delta t}^{(ij)} \rangle) \\ \times \exp(\lambda_{\mathbf{X}} \langle \nabla_{z_t} \|\mu(\mathbf{X}_t, t) - c\|^2, z_{t+\Delta t} - z_t \rangle),\end{aligned} \tag{33}$$

where $\lambda_{\mathbf{X}}$ controls the strength of the guidance.

The smoothness assumption we adopt is exactly the one adopted by earlier discrete-guidance methods [3, 63], our derivation remains consistent with the foundations laid out in those works.

Conditional bag sampling To define $\mathcal{B}|c$, we first recall the unconditional case. When no property is specified, a bag $\mathcal{B} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of size N is drawn without replacement from the the fragments vocabulary $\mathcal{F}$ with probability

$$P(\mathcal{B}) = \frac{\prod_{\mathbf{x} \in \mathcal{B}} p(\mathbf{x})}{\sum_{\substack{\mathcal{S} \subset \mathcal{F} \\ |\mathcal{S}| = N}} \prod_{\mathbf{y} \in \mathcal{S}} p(\mathbf{y})}, \tag{34}$$

i.e., bags that contain high-frequency fragments under the marginal distribution $p(\mathbf{x})$ are sampled more often. To steer this toward a desired property c, we replace each $p(\mathbf{x})$ by its conditional counterpart $p(\mathbf{x}|c)$. Viewing a fragment as part of a molecule $\mathbf{X}$, the condition can be written as the expected property value over all molecules that contain that fragment: $\sum_{\mathbf{x}} p(\mathbf{x}|\mathbf{X}, c)p(\mathbf{X}|c)$.

The resulting bag distribution becomes

$$P\big(\mathcal{B} \mid c\big) = \frac{\displaystyle\prod_{\mathbf{x}\in\mathcal{B}} p\big(\mathbf{x} \mid c\big)}{\displaystyle\sum_{\substack{\mathcal{S}\subset\mathcal{F}\\ |\mathcal{S}|=N}} \prod_{\mathbf{y}\in\mathcal{S}} p(\mathbf{y} \mid c)}\,. \tag{35}$$

Applying Bayes' rule and dropping the constant factor $p(c)$ gives

$$p(\mathbf{x}|c) \propto p(\mathbf{x})p(c|\mathbf{x}). \tag{36}$$

In practice, we estimate $p(c|\mathbf{x})$ as a gaussian distribution with a light neural regressor parameterized by ϕ,

$$p_\phi(c \mid \mathbf{x}) \;=\; \mathcal{N}\big(c;\, \mu(\mathbf{x};\phi),\, \sigma^2\big), \quad p_\phi(\mathbf{x} \mid c) \;\propto\; p(\mathbf{x})\, \exp\big(-\lambda_{\mathcal{B}}\, \|\, \mu(\mathbf{x};\phi) - c\|^2\big), \tag{37}$$

where $\mu(\mathbf{x}, \phi)$ is the predicted mean, σ^2 is a fixed variance, and $\lambda_{\mathcal{B}}$ controls the strength of the property-guided bag selection.

A.5 Detailed Balance

The space of valid rate matrices extends beyond the original formulation of eq. (3); thus, alternative constructions can still satisfy the Kolmogorov equation. Campbell et al. [32] show that if a matrix R_t^{DB} fulfils the detialed-balance identity:

$$p_{t|1}(x_t \mid x_1)R_t^{DB}(x_t, y \mid x_1) = p_{t|1}(y \mid x_1)R_t^{DB}(y, x_t \mid x_1),, \tag{38}$$

then,

$$R_t^{\eta} = R_t^{*} + \eta R_t^{DB}, \quad \eta \in \mathbb{R}^{+}, \tag{39}$$

remains a valid CTMC generator. A larger η injects extra stochasticity, opening additional state-transition pathways.

Although several designs are possible, we follow Campbell et al. [32]. The only non-zero rates for fragment-type nodes are the transitions between a concrete type x_1 and the mask state $\mathbf{M}$:

$$R_t^{DB}(\mathbf{x}, \mathbf{y}|\mathbf{x_1}) = \delta(\mathbf{x}, \mathbf{x_1})\delta(\mathbf{y}, \mathbf{M}) + \delta(\mathbf{x}, \mathbf{M})\delta(\mathbf{y}, \mathbf{x_1}), \tag{40}$$

where $\mathbf{M}$ denotes the masked type.

For edges, whose states are binary $\varepsilon^{ij} \in \{0, 1\}$, we consider a flip rate η_{edge} and a matching backward rate that satisfies eq. (38) which leads to:

$$R_t^{DB}(\varepsilon, \varepsilon' \mid \varepsilon_1) = \delta(\varepsilon, \varepsilon_1) + \frac{1+t}{1-t}\delta(\varepsilon', \varepsilon_1). \tag{41}$$

We set $(\eta_{\text{node}}, \eta_{\text{edge}}) = (20.0, 20.0)$ for MOSES, and $(10.0, 2.0)$ for GuacaMol and NPGen datasets.

B Parameterization and Hyperparameters

B.1 Coarse-to-Fine Autoencoder

Our coarse-to-fine autoencoder (eq. (10)) compresses the atom-level graph into a single latent vector z and then uses it, together with the fragment-level graph $\mathcal{G}$, to reconstruct all atom–atom connections. The encoder, an MPNN [64], takes G and pools its node features into z. The decoder conditions on $\mathcal{G}$ and z to predict a distribution over every possible atom–atom edge between them for each pair of linked fragments. Internally, it propagates messages along original intra-fragment bonds and across all candidate inter-fragment edges, enabling the recovery of the complete atomistic structure from the coarse abstraction.

B.2 Fragment Embedder, Prediction Model, and Property Discriminator

To parameterize the neural network $f_\theta(\mathbf{X}_t, \mathbf{x})$ in eq. (6), we jointly train two components: a fragment embedder and a Graph Transformer (GT). Figure 5 provides an overview.

The fragment embedder, built on an MPNN backbone [64], maps each fragment to a fixed-dimensional embedding vector. Given a fragment-level graph $\mathbf{X}_t$, we apply this shared embedder to every fragment node, producing a set of local structure embeddings. Multiple GT layers, then process these embeddings to capture inter-fragment interactions and global context. The GT layers were designed with the sample architecture and hyperparameters as prior atom-based diffusion- and flow-based molecule generative models [3–5, 29]. We directly predict the discrete fragment-graph edges $\mathcal{E}_1$ and the continuous latent vector z_1 from the final GT output embeddings. To predict fragment types, we compute the inner product between each candidate fragment type embedding and its corresponding GT embedding to infer the scores of different fragments. We reuse the flow model's architecture for property discrimination: We aggregate both the fragment-level embeddings and the GT's global readout to produce fragment—and molecule-level property predictions.

We train our flow model with AdamW optimizer, using $(\beta_1, \beta_2) = (0.9, 0.999)$, a learning rate of 5×10^{-4}, and gradient-nrom clipping at 4.0. We employ the exponential moving average (EMA) scheme of Karras et al. [65] to stabilize training. Training is performed on a single NVIDIA A100 GPU for 96 h on MOSES and GuacaMol and 144 h on NPGen, and we select the checkpoint with the lowest validation loss.

B.3 Auxiliary Features

Although graph neural networks exhibit inherent expressivity limitations [66], augmenting them with auxiliary features has proven effective at mitigating these shortcomings. For example, Vignac et al. [3] augments each noisy graph with cycle counts, spectral descriptors, and basic molecular properties (e.g., molecular weight, atom valence). More recently, relative random walk probabilities (RRWP) have emerged as a highly expressive yet efficient encoding [67]: by stacking the first K powers of the normalized adjacency matrix $M = D^{-1}A$, RRWP constructs a k-step transition probabilities that capture rich topological information. Accordingly, we integrate RDKit-derived molecular descriptors into the fragment embedder and RRWP features into the graph transformer, enriching the model's ability to capture complex molecular semantics.

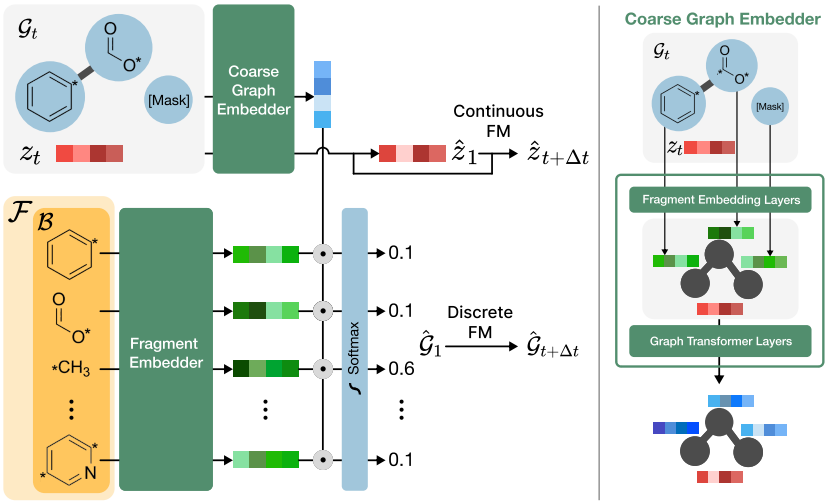


Figure 5: **Overview of the parameterization of the fragment embedder and prediction model, *i.e.*, f_θ in fig. 1**. (left) Each fragment in the fragment bag $\mathcal{B}$ is embedded by the fragment embedder, while each node in the coarse graph is mapped to a fixed-size vector. We compute $f_\theta(\mathbf{X}_t, \mathbf{x})$ by taking the inner product of the two embeddings. (right) The coarse-graph embedder first maps every node to an embedding, producing a coarse graph whose nodes are single vectors; the resulting graph is then passed through the graph-transformer layer.

B.4 Noise Schedule

Selecting an appropriate noise schedule can affect the performance of diffusion- and flow-based models [3–5]. Following Qin et al. [4], we adopt the polydec (polynomially decreasing) time distortion, which skews the initially uniform time distribution so that more steps are spent close to the data manifold. Concretely, a uniformly sampled $u \sim \mathcal{U}[0,1]$ is warped by $f(t) = 2t - t^2$, mapping the endpoints $f(0) = 0,\ f(1) = 1$ while stretching the density toward small t. In our Euler discretisation, this concentrates integration steps where fine-grained denoising is most critical.

C Details for Standard Molecular Generation Benchmarks

C.1 Metrics

We provide details of common metrics in both MOSES[18] and GuacaMol[19] benchmarks.

Common Metrics. These metrics are fundamental for assessing the basic performance of molecular generative models. Note that **V.U.** and **V.U.N.** metrics are multiplied values of each metric, *i.e.*, V.U.N. is computed by multiplying validity, uniqueness and novelty.

- **Validity (Valid):** This metric measures the proportion of generated molecules that are chemically valid according to a set of rules, typically checked using tools like RDKit. A SMILES string is considered valid if it can be successfully parsed and represents a chemically sensible molecule (*e.g.*, correct atom valencies, no impossible structures).
- **Uniqueness (Unique):** This indicates the percentage of unique molecules among valid generated molecules. A high uniqueness score suggests the model is generating diverse structures rather than repeatedly producing the same few molecules.
- **Novelty (Novel):** This metric quantifies the fraction of unique and valid generated molecules that are not present in the training dataset. It assesses the model's ability to generate novel chemical molecules.

MOSES Metrics. The MOSES benchmark focuses on distribution learning. Key metrics beyond the foundational ones include:

- **Filters:** This refers to the percentage of valid, unique, and novel molecules that pass a set of medicinal chemistry filters (PAINS, MCF) and custom rules defined by the MOSES benchmark (*e.g.*, specific ring sizes, element types), which are used in curating dataset of MOSES. This evaluates the drug-likeness or suitability of generated molecules according to predefined structural criteria.
- **Fréchet ChemNet Distance (FCD):** FCD[57] measures the similarity between the distribution of generated molecules and a reference (**test**) dataset based on latent representation of molecules using a pre-trained neural network (ChemNet). A lower FCD indicates that the generated distribution is closer to the reference distribution.
- **Similarity to Nearest Neighbor (SNN):** This metric calculates the average Tanimoto similarity using Morgan fingerprints [68] between each generated molecule and its nearest neighbor in the reference **(test)** dataset. A higher SNN suggests that the generated molecules are similar in structure to known molecules in the target chemical space.
- **Scaffold Similarity (Scaf):** This metric specifically assesses the diversity of molecular scaffolds. It calculates a cosine similarity between the vectors of the occurrence of Bemis–Murcko scaffolds [69] of the molecules in the reference **(test)** dataset and the generated ones. A higher score suggests generated scaffolds are similar to reference scaffolds.

GuacaMol Metrics. GuacaMol provides benchmarks for distribution-learning and goal-directed generation. For its distribution-learning benchmark, which is utilized for our main results, the primary aggregated metrics are:

- **Kullback-Leibler Divergence (KL Div.) Score:** This metric computes the KL divergence between the distributions of several physicochemical and topological properties of the generated molecules and the training set. These individual KL divergences (D_{KL}) are then

combined into a single score, by averaging negative exponential of them (*i.e.*, $\exp(-D_{KL})$) to reflect how well the model reproduces the overall property distributions. Due to the nature of calculation method, a score closer to 1 indicates better similarity.

- **Fréchet ChemNet Distance (FCD) Score:** Similar to the MOSES FCD, GuacaMol also uses an FCD metric to compare the distributions of generated molecules and the **training** set. The only difference is that in GuacaMol, the raw FCD value (where lower is better) is transformed into a score where higher is better.

C.2 Baselines

Next, we briefly introduce baseline strategies that we compared FragFM in the main results. We focused on molecular *graph* generative models, which are categorized by autoregressive and one-shot generation models. Each model uses either atom- or fragment-level representation.

GraphInvent [22] employs a graph neural network (GNN) approach for *de novo* molecular design. It first compute the trajectory of graph decomposition based on atom-level representation, and then trains a GNN to learn action of atom and bond addition on given subgraph of molecule. During inference stage, it builds molecules in atom-wise manner.

JT-VAE [12] or Junction Tree Variational Autoencoder, generates molecular graphs in a two-step process. It first decodes a latent vector into a tree-structured scaffold representing molecular components (like rings and motifs) and then assembles these components into a complete molecular graph, ensuring chemical validity. Since it iteratively decide whether to add node during sampling process, we consider it as autoregressive model despite its use of VAE.

MCTS [70] is a non-deep learning-based strategy that utilizes Monte-Carlo tree search for molecular graph generation. Using atom insertion or addition as action, it sequentially build molecules from a starting molecule.

NAGVAE [24], non-autoregressive Graph Variational Autoencoder, is a VAE-based one-shot graph generation model utilizing compressed graph representation. It reconstructs the molecular graphs from latent vectors, aiming for scalability and capturing global graph structures.

DiGress [3] is the first discrete diffusion model designed for graph generation. It operates by iteratively removing noise from both graph edges and node types, learning a reverse diffusion process to construct whole graphs from a noise distribution.

DisCo [29] is a graph generation model that defines a forward diffusion process with continous-time Markov chain (CTMC). The model learns reverse generative process to denoise both the graph structure and its attributes simultaneously.

Cometh [5] is a continuous-time discrete-state graph diffusion model. Similar to Disco, it formulates graph generation as reversing a CTMC defined on graphs, where the model learns the transition rates of this chain to generate new graph structures.

DeFoG [4] is a generative framework that applies the principles of flow matching directly to discrete graph structures. After training via flow matching strategy, it utilizes CTMC for denoising process to generate graphs.

D Details for Natural Product Generation Benchmark

In this section, we further explain the dataset, baseline, and metrics of the proposed Natural Product Generation benchmark (NPGen).

D.1 Dataset Construction

To construct the NPGen dataset, we utilized the 2024/12/31 version of the COCONUT database [53, 54], which comprises 695,120 natural product-like molecules. Given that the original database contains compounds with transition metals—species that are rarely encountered in typical organic natural products—we applied a filtering procedure to retain only molecules compose exclusively of non-metal atoms: `'B'`, `'C'`, `'N'`, `'O'`, `'F'`, `'Si'`, `'P'`, `'S'`, `'Cl'`, `'As'`, `'Se'`, `'Br'`, `'I'`.

Additionally, to exclude arbitrarily large macromolecules, we retained only those molecules whose heavy-atom counts fell between 2 and 99.

Furthermore, we only consider neutral molecules without salts, filtering charged molecules and molecules containing "." in their SMILES representation. After filtering, a total of 658,566 molecules were retained. The resulting dataset was randomly partitioned into training, validation, and test subsets using an 85:5:15 split under the assumption of i.i.d. sampling, yielding 526,852, 32,928, and 98,786 molecules, respectively.

D.2 Implementation Details for Baselines

As explained in section 4, we selected a set of molecular graph generative models with two aspects, *i.e.*, generation strategy (autoregressive and one-shot) and representation level (atom and fragment). We provide more details on the baseline models.

GraphAF [23] is a flow-based autoregressive model for molecular graph generation that constructs molecules sequentially by adding atoms and their corresponding bonds. We used the authors' official implementation from (`https://github.com/DeepGraphLearning/GraphAF`) with its default settings, extending only the preprocessing and generation steps to include atom types that the original implementation does not support `'B','As','Si','As','Se'`. During generation, the official implementation terminates sampling once 40 atoms are generated for each molecule; we modified this limit to 99 to match the NPGen benchmark's maximum heavy-atom count.

JT-VAE[12] is a fragment-based autoregressive variational autoencoder that generates molecules by building a junction tree of chemically meaningful substructures and then assembling the corresponding atom-level graph [12]. We used the authors' official implementation (`https://github.com/wengong-jin/icml18-jtnn`) with the acceleration module (`fast_molvae`). Because the codebase relies on Python 2 and is incompatible with newer GPU drivers, we performed training and sampling on an NVIDIA GeForce RTX 2080 Ti. We performed a random hyperparameter search over `hidden_dim` and `batch_size`, and report the best results.

HierVAE [15] builds on JT-VAE by introducing a hierarchical latent space and a scaffold-aware message-passing scheme to boost structural diversity and sampling fidelity. We used the authors' official implementation (`https://github.com/wengong-jin/hgraph2graph`), extending only the preprocessing step to include the `'As'` atom type. By default, HierVAE employs a greedy motif-sampling strategy, which prioritizes top-scoring fragments and may bias the output distribution. We observed that this led to artifacts, only generating single carbon chains on the NPGen benchmark. To provide a fair comparison, we report the results of the alternative stochastic-sampling mode (enabled via a single option flag in the official implementation), without modifying the core codebase.

DiGress [3] is an atom-based generative model that employs discrete diffusion. We run the authors' official implementation (`https://github.com/cvignac/DiGress`) with all default hyperparameters, adding the atom types `'B','As'` and their corresponding charges.

D.3 Metrics

As mentioned in the main text, we utilize two methods for distributional metrics: NP-likeness score [55] and NP Classifier [56]. Both strategies are developed by domain experts to effectively analyze the molecule through the lens of a natural product.

NP-likeness score is developed to quantify the similarity of a given molecule to the structural space typically occupied by natural products. Since one of the major differences between natural products and synthetic molecules is structural features such as the number of aromatic rings, stereocenters, and distribution of nitrogen and oxygen atoms, the NP-likeness of a molecule is calculated as the sum of the contributions of its constituent fragments, where each fragment's contribution is based on its frequency in natural product versus synthetic molecule databases. The high value of the NP-likeness score indicates that the probability of a molecule being an NP is high.

NPClassifier is a deep learning-based tool specifically designed to classify NPs. It categorizes molecules at three hierarchical levels—Pathway (7 categories; *e.g.*, Polyketides, Terpenoids), Superclass (70 categories, *e.g.*, Macrolides, Diterpenoids), and Class (672 categories; *e.g.*, Erythromycins, Kaurane diterpenoids)—reflecting the biosynthetic origins, broader chemical and chemo-

taxonomic properties, and specific structural families recognized by the NP research community. This multi-level system, built on an NP-specific ontology and trained on over 73,000 NPs using counted Morgan fingerprints, provides a classification based on knowledge of natural products, including their biosynthetic relationships and structural diversity.

We employed Kullback-Leibler (KL) divergence as a metric for both methods. We compute KL divergence for NP-likeness score and NPClassifier differently, as they are continuous and discrete values, respectively. It is worth noting that NPClassifier often predicts `'Unclassified'`, which indicates a molecule is not included in any classes, along with multiple class results (*e.g.*, `'Peptide alkaloids, Tetramate alkaloids'` in Class). We treat all prediction results as another unique class, since molecules can have multiple structural features.

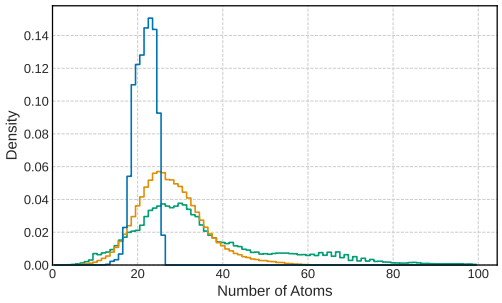
D.4 Dataset Statistics

We analyzed the distributions of several molecular properties to highlight NPGen's distinctions from standard molecular generative benchmarks (MOSES and GuacaMol). These properties fell into two categories: (1) simple molecular descriptors, such as the number of atoms, molecular weight, and number of hydrogen bond acceptors and donors (fig. 6), and (2) functionality-related properties, including NP-likeness scores and NPClassifier prediction results (fig. 7). Consistent with the nature of NPs, which are generally larger and more complex than typical synthetic drug-like molecules, NPGen molecules are, on average, larger in terms of the number of atoms and molecular weight compared to those in MOSES and GuacaMol (figs. 6a and 6b). Furthermore, molecules in NPGen exhibit higher numbers of hydrogen bond acceptors and donors (see figs. 6c and 6d), reflecting another characteristic of NPs.

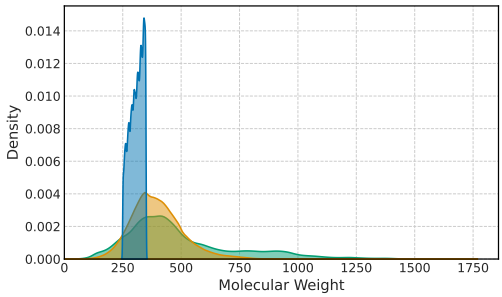
The difference between benchmarks becomes more significant when examining functionality-related properties. NPClassifier predictions for Pathway (fig. 7a) indicate that NPGen molecules span a diverse range of NP categories. In contrast, molecules from MOSES and GuacaMol mostly fall into `'Alkaloids'`, which are non-peptidic nitrogenous organic compounds, or remain unclassified. Focusing on four selected Superclass categories for which NPClassifier had demonstrated high predictive performance (F1 score higher than 0.95 for categories with more than 500 compounds [56]), NPGen shows higher proportions of molecules in these specific categories. Conversely, molecules from the other benchmarks mostly fall into `'Unclassified'`, implying that they are dissimilar to NPs. The NP-likeness score further emphasizes this divergence (fig. 7c). In particular, NPGen's distribution is largely shifted towards higher scores (average: 1.14) compared to MOSES (average: -1.67) and GuacaMol (average: -0.90), where a higher score indicates greater similarity to NPs.

Additionally, we visualize the chemical space of existing benchmarks (MOSES, GuacaMol) and NPGen using UMAP [71] in fig. 8. While MOSES and GuacaMol occupy a largely overlapping region, NPGen extends into distinct areas, indicating coverage of different chemical subspaces.

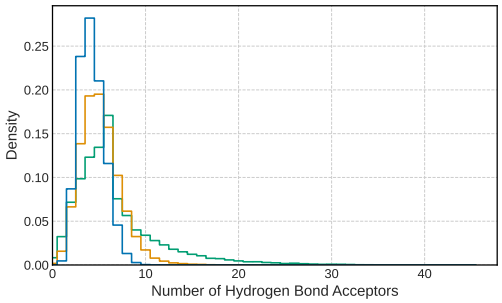
These statistical analyses demonstrate that NPGen has distinct features compared to existing molecular generative benchmarks, proving its suitability to serve as a unique molecular graph generative benchmark targeting NP-like chemical space.



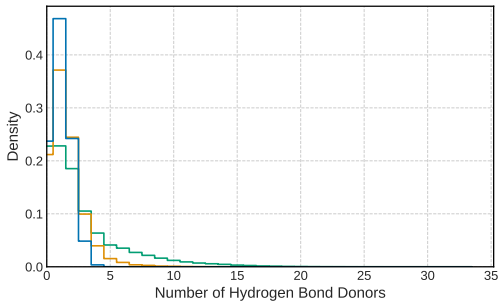
(a) Number of atoms distributions.



(b) Molecular weight distributions.

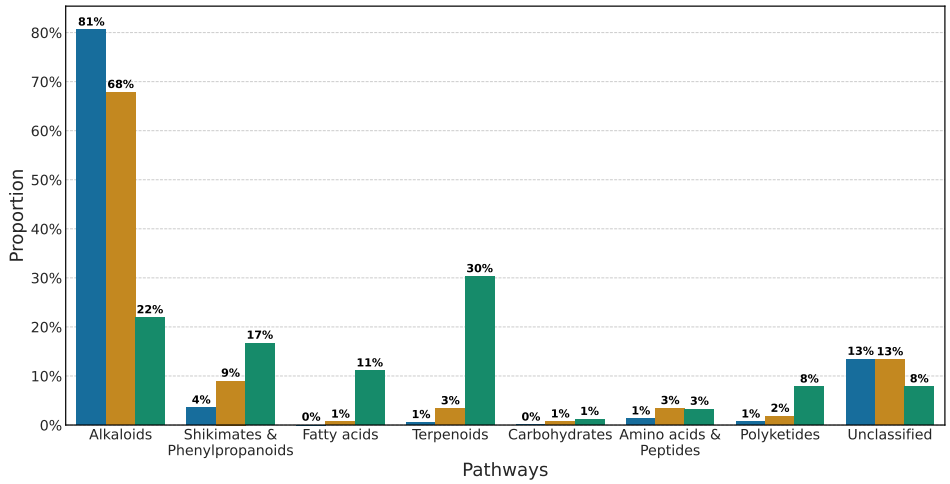


(c) Number of hydrogen bond acceptors distributions.

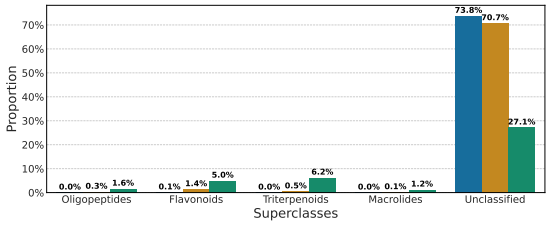


(d) Number of hydrogen bond donors distributions.

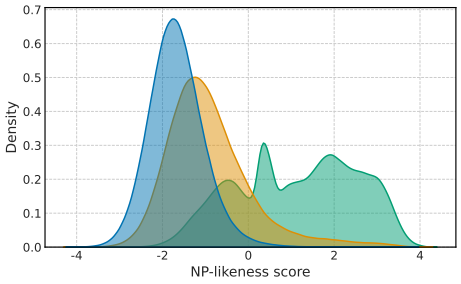
Figure 6: **Comparison of simple molecular property distributions among three benchmarks: MOSES, GuacaMol, and NPGen**. The number of molecules in each dataset is 1,936,962, 1,591,378, and 658,565, respectively.



(a) Proportion of all Pathway, along with `'Unclassified'`. Note that we excluded predicted results with multiple categories, for clarity. Specifically, number of predictions decreased as 1,936,962 to 1,873,287, 1,591,378 to 1,542,118, and 658,565 to 628,840 for MOSES, GuacaMol, and NPGen, respectively.



(b) Proportion of four Superclass, along with `'Unclassified'`. Note that we excluded predicted results with multiple categories, for clarity. Specifically, number of predictions decreased as 1,936,962 to 1,933,854, 1,591,378 to 1,588,486, and 658,565 to 650,013 for MOSES, GuacaMol, and NPGen, respectively.



(c) NP-likeness score distribution.

Figure 7: **Comparison of NP-likeness score and NPClassifier prediction results among three benchmarks: MOSES, GuacaMol, and NPGen**. The number of molecules in each dataset is 1,936,962, 1,591,378, and 658,565, respectively. Note that we also report the ratio of unclassified entities of dataset in figs. 7a and 7b. A statistics of Class prediction results is not included since it has 687 classes and the ratio of each class is too small compared to `'Unclassified'` class.

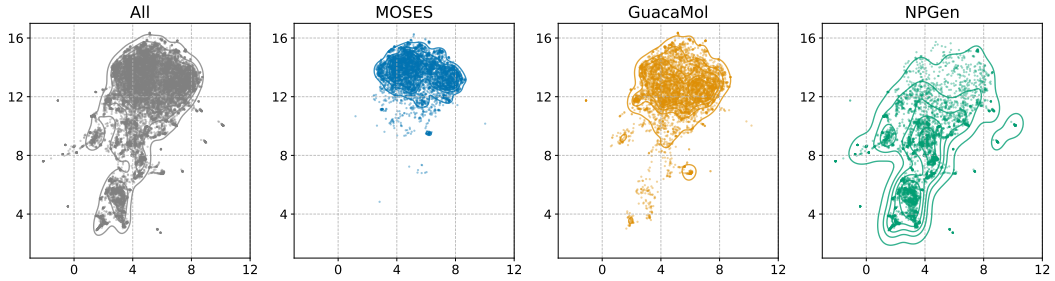


Figure 8: **UMAP visualization of MOSES, GuacaMol, and NPGen datasets**. We randomly selected 5,000 molecules from each train dataset and applied UMAP to their ECFP fingerprints [72] (radius 2, 2048 bits). UMAP was run with its default settings with `n_neighbors=15` and `min_dist=0.1`.

D.5 Benchmark Results with Multiple Runs

We provide the average and standard deviation results from three runs in NPGen for all baselines and FragFM in table 3.

Table 3: **Molecule generation results on NPGen**. We use a total of 30,000 molecules for evaluation. The upper part comprises autoregressive methods, while the second part comprises one-shot methods, including diffusion-based and flow-based methods. The results are averaged over three runs. The best performance is shown in **bold**, and the second-best is underlined. The numbers with ± indicates the standard deviation against each runs.

Model	Val. ↑	Unique. ↑	Novel ↑	NP Score KL Div. ↓	NP Class KL Div. ↓ Pathway	Superclass	Class	FCD ↓
Training set	100.0	100.0	-	0.0006	0.0002	0.0028	0.0094	0.01
GraphAF [23]	79.1±0.1	63.6±0.2	95.6±0.0	0.8546±0.0095	0.9713±0.0055	3.3907±0.0730	6.6905±0.0905	25.11±0.08
JT-VAE [12]	**100.0**±0.0	97.2±0.1	99.5±0.0	0.5437±0.0188	0.1055±0.0019	1.2895±0.1243	2.5645±0.4557	4.07±0.02
HierVAE [15]	**100.0**±0.0	81.5±1.1	97.7±0.0	0.3021±0.0063	0.4230±0.0051	0.5771±0.0121	1.4073±0.0630	8.95±0.06
DiGress [3]	85.4±0.0	**99.7**±0.0	**99.9**±0.0	0.1957±0.0028	0.0229±0.0001	0.3370±0.0042	1.0309±0.0182	2.05±0.01
FragFM (ours)	98.0±0.0	99.0±0.0	95.4±0.1	**0.0374**±0.0001	**0.0196**±0.0008	**0.1482**±0.0026	**0.3570**±0.0006	**1.34**±0.01

E Additional Results

E.1 GuacaMol Benchmark

In the GuacaMol benchmark (table 4), similarly to MOSES, FragFM achieved the best performance among baseline diffusion and flow models in Val. and V.U. metric, with the state-of-the-art FCD score and close second in KL Div. score considering various molecular property distributions. These results emphasize the effectiveness of the fragment-based approach of FragFM in generating valid and chemically meaningful molecules. The visualization results of GuacaMol is shown in appendix F.2.

Table 4: **Molecule generation results on the GuacaMol benchmark**. We use a total of 10,000 generated molecules for evaluation. All baselines except MCTS in this table is one-shot methods. The results for FragFM are averaged over three independent runs. The best performance is shown in **bold**, and the second-best is underlined.

Model	Rep. Level	Val. ↑	V.U. ↑	V.U.N. ↑	KL Div. ↑	FCD ↑
Training set	-	100.0	100.0	-	99.9	92.8
MCTS [70]	Atom	**100.0**	**100.0**	95.4	82.2	1.5
NAGVAE [24]	Atom	92.9	88.7	88.7	38.4	0.9
DiGress [3]	Atom	85.2	85.2	85.1	92.9	68.0
DisCo [29]	Atom	86.6	86.6	86.5	92.6	59.7
Cometh [5]	Atom	94.4	94.4	93.5	94.1	67.4
Cometh-PC [5]	Atom	98.9	98.9	97.6	96.7	72.7
DeFoG [4]	Atom	99.0	99.0	**97.9**	**97.7**	73.8
FragFM (ours)	Fragment	99.7	99.3	95.0	97.4	**85.8**

E.2 Coarse-to-Fine Autoencoder

We measure bond-level and whole-graph reconstruction accuracy to assess the fidelity of the coarse-to-fine autoencoder. As reported in table 5, bond accuracy exceeds 99% on both MOSES and GuacaMol, indicating almost perfect recovery of individual chemical bonds. Graph-level accuracy is similarly high, confirming that the overall connectivity patterns are faithfully preserved. Even in the structurally diverse and larger COCONUT dataset, the autoencoder maintains strong performance, with only a slight drop in accuracy, underscoring its robustness in handling complex molecular topologies.

Table 5: **Coarse-to-fine autoencoder accuracy**.

Benchmarks	Train set accuracy		Test set accuracy	
	Bond	Graph	Bond	Graph
MOSES	99.99%	99.96%	99.99%	99.93%
GuacaMol	99.99%	99.43%	99.98%	99.42%
NPGen	99.98%	97.62%	99.71%	97.43%

E.3 Fragment Bag Generalization

Tables 6 and 7 reports FragFM's performance when sampling with fragment bags derived from the test-set molecules on both MOSES and GuacaMol. Recall that MOSES uses a scaffold-split evaluation—test scaffolds are deliberately excluded from training—so sampling with only training-set fragments limits the model's ability to recover those unseen scaffolds, resulting in a depressed Scaf score. We re-ran the generation using fragment bags drawn from the test split to validate this. When provided with the test-set fragments, FragFM's Scaf score increases dramatically, while all other metrics on MOSES and GuacaMol remain unchanged. This demonstrates that, leveraging its fragment embedding module, FragFM can generalize to novel fragments without compromising validity, uniqueness, or other quality measures.

Table 6: **Molecule generation with unseen fragment bag on MOSES dataset**. We use a total of 25,000 generated molecules for evaluation. The results are averaged over three independent runs.

Model	Rep. Level	Valid ↑	Unique ↑	Novel ↑	Filters ↑	FCD ↓	SNN ↑	Scaf ↑
Training set	-	100.0	100.0	-	100.0	0.48	0.59	0.0
GraphINVENT [22]	Atom	96.4	99.8	-	95.0	1.22	0.54	12.7
JT-VAE [12]	Fragment	100.0	100.0	99.9	97.8	1.00	0.53	10.0
DiGress [3]	Atom	85.7	100.0	95.0	97.1	1.19	0.52	14.8
DisCo [29]	Atom	88.3	100.0	97.7	95.6	1.44	0.50	15.1
Cometh [5]	Atom	90.5	99.9	92.6	99.1	1.27	0.54	16.0
DeFoG [4]	Atom	92.8	99.9	92.1	98.9	1.95	0.55	14.4
FragFM (train fragments)	Fragment	99.8	100.0	87.1	99.1	0.58	0.56	10.9
FragFM (test fragments)	Fragment	99.8	100.0	88.2	98.9	0.44	0.57	24.5

Table 7: **Molecule generation with unseen fragment bag on GuacaMol dataset**. We use a total of 10,000 generated molecules for evaluation. The results are averaged over three independent runs.

Model	Rep. Level	Val. ↑	V.U. ↑	V.U.N. ↑	KL Div. ↑	FCD ↑
Training set	-	100.0	100.0	-	99.9	92.8
MCTS [70]	Atom	100.0	100.0	95.4	82.2	1.5
NAGVAE [24]	Atom	92.9	88.7	88.7	38.4	0.9
DiGress [3]	Atom	85.2	85.2	85.1	92.9	68.0
DisCo [29]	Atom	86.6	86.6	86.5	92.6	59.7
Cometh [5]	Atom	98.9	98.9	97.6	96.7	72.7
DeFoG [4]	Atom	99.0	99.0	97.9	97.7	73.8
FragFM (train fragments)	Fragment	99.7	99.4	95.0	97.4	85.7
FragFM (test fragments)	Fragment	99.8	99.4	97.4	97.6	85.7

E.4 Sampling Efficiency

Diffusion- and flow-based models typically require multiple denoising iterations, resulting in slow sampling. Table 8 shows the performance of MOSES benchmark metrics of FragFM and baseline denoising based models with different denoising steps. For small sampling steps, FragFM outperforms

the baseline models with minimal degradation in metrics, especially at low step counts, by a wide margin. With only 10 sampling steps, FragFM achieves higher validity and a lower FCD than competing models running 500 steps.

We also compare sampling time across different models in table 9. By operating both node- and edge-probability predictions, edge computations scale quadratically with graph size, making them the fastest approach among the compared models. Coupled with its robust performance at far fewer steps, FragFM could be further optimized with substantial speedups over atom-level methods with high generative quality.

Table 8: **Performance of denoising-based graph generative models on the MOSES dataset across different sampling step counts**. All the models are one-shot models. Results for DeFoG and Cometh are taken from their original publications; DiGress (excluding the 500-step setting) were obtained by retraining the model with the differing sampling steps from the official implementation. The best performance is shown in **bold** for each sampling step.

Sampling steps	Model	Rep. Level	Val. ↑	V.U. ↑	V.U.N. ↑	Filters ↑	FCD ↓	SNN ↑	Scaf ↑
-	Training set	-	100.0	100.0	-	100.0	0.48	0.59	0.0
10	DiGress	Atom	6.3	6.3	6.3	66.4	9.40	0.38	7.4
	Cometh	Atom	26.1	26.1	26.0	59.9	7.88	0.36	8.9
	DeFoG	Atom	-	-	-	-	-	-	-
	FragFM	Fragment	**96.8**	**96.6**	**89.4**	**96.8**	**0.92**	**0.52**	**15.3**
50	DiGress	Atom	75.3	75.3	72.3	94.0	1.35	0.51	16.1
	Cometh	Atom	82.9	82.9	80.5	94.6	1.54	0.49	**18.4**
	DeFoG	Atom	83.9	83.8	81.2	96.5	1.87	**0.59**	14.4
	FragFM	Fragment	**99.5**	**99.5**	**89.1**	**98.5**	**0.65**	0.54	11.2
100	DiGress	Atom	82.6	82.6	79.2	95.2	1.14	0.51	15.4
	Cometh	Atom	85.8	85.7	82.9	96.5	1.43	0.50	**17.2**
	DeFoG	Atom	-	-	-	-	-	-	-
	FragFM	Fragment	**99.7**	**99.7**	**88.5**	**98.8**	**0.62**	**0.55**	11.6
300	DiGress	Atom	85.3	85.3	81.1	96.5	1.11	0.52	13.5
	Cometh	Atom	86.9	86.9	83.8	97.1	1.44	0.51	**17.8**
	DeFoG	Atom	-	-	-	-	-	-	-
	FragFM	Fragment	**99.8**	**99.8**	**87.2**	**98.9**	**0.58**	**0.55**	11.6
500	DiGress	Atom	85.7	85.7	81.4	97.1	1.19	0.52	14.8
	DiGress	Atom	84.8	84.8	82.0	94.5	1.37	0.50	14.7
	Cometh	Atom	87.0	86.9	83.8	97.2	1.44	0.51	**15.9**
	DeFoG	Atom	92.8	92.7	85.4	98.9	1.95	0.55	14.4
	FragFM	Fragment	**99.8**	**99.8**	**86.9**	**99.1**	**0.58**	**0.56**	10.9
700	DiGress	Atom	85.5	85.5	82.6	95.0	1.33	0.50	15.3
	Cometh	Atom	87.2	87.1	83.9	97.2	1.43	0.51	**15.9**
	DeFoG	Atom	-	-	-	-	-	-	-
	FragFM	Fragment	**99.9**	**99.9**	**86.9**	**99.1**	**0.61**	**0.56**	10.8
1000	DiGress	Atom	84.7	84.7	81.3	96.1	1.31	0.51	14.5
	Cometh	Atom	87.2	87.2	84.0	97.2	1.44	0.51	**17.3**
	DeFoG	Atom	-	-	-	-	-	-	-
	FragFM	Fragment	**99.8**	**99.8**	**86.6**	**99.1**	**0.62**	**0.56**	12.9

Table 9: **Comparison of sampling time across different datasets and methods**. Experiments were conducted on a single NVIDIA GeForce RTX 3090 GPU and an Intel Xeon Gold 6234 CPU @ 3.30GHz. *Results for DeFoG are taken from the original paper, where experiments were conducted on an NVIDIA A100 GPU.

		Sampling steps	MOSES	GuacaMol	NPGen
Property	Min. nodes	-	8	2	2
	Max. nodes	-	27	88	99
	# Samples	-	25000	10000	30000
Sampling Time (hour)	DiGress	500	3.0	-	36.0
	DeFoG*	500	5.0	7.0	-
	FragFM	500	0.9	1.3	7.0
	FragFM	50	0.2	0.2	0.9

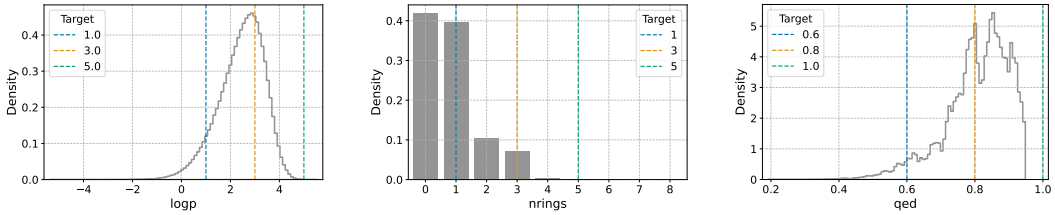
E.5 More Results on Conditional Generation

For simple molecular properties (logP, QED, and number of rings), we perform conditional generation on the MOSES dataset using regressors trained on its training split. The detailed illustration of property distributions and corresponding targets is depicted in fig. 9. For protein-target conditioning, we perform conditioning on the ZINC250K dataset. The targets were selected from the DUD-E$^+$ virtual screening benchmark for our docking score experiments, following Yang et al. [73]. The established reliability of Smina[74], which is a forked version of AutoDock Vina[75], evidenced by its high AUROC for discriminating hits from decoys on DUD-E$^+$, led us to use it as an oracle.

Figure 10 depitcs the Smina docking score distributions for ZINC250K molecules against three targets (fa7, jak2, parp1). Since lower scores correspond to stronger predicted binding, we selected the conditioning value for each protein at the extreme left tail of its distribution (fa7: -10.0 kcal/mol, jak2: -11.0 kcal/mol, parp1: -12.0 kcal/mol), indicated by the vertical dashed lines in fig. 10 to focus generation to the most tightly binding candidiates.

With the perspective of chemistry, the worse FCD and validity with conditions of DiGress shown in appendix E.5 highlights a critical challenge for atom-based approaches: satisfying targeted property constraints while ensuring chemical correctness, *simultaneously*. From a chemical perspective, this distinction can be attributed to the nature of the search space; atom-based approaches explore a vastly larger and less constrained space, where many cases can lead to chemically invalid structures, especially when generation is heavily biased by property objectives. Conversely, FragFM's fragment-based construction inherently operates within a more chemically sound and constrained subspace by assembling pre-validated chemical motifs. These findings collectively emphasize the intrinsic advantages of employing fragments as semantically rich and structurally robust building blocks, particularly for achieving reliable and property-focused molecular generation.

Moreover, the importance of the fragment bag's composition, which is shown in the main text (fig. 4), is intuitive: it defines the accessible chemical space and, consequently, the possible range of achievable molecular properties (*e.g.*, generating acyclic molecules is impossible if the fragment bag exclusively contains ring-based structures, among other structural constraints). Based on $\lambda_{\mathcal{B}}$, FragFM automatically modulates fragment selection probabilities, inducing a drift in the fragment space to generate the chemically valid molecules satisfying the given objective. It enables the model to construct molecules with desired properties even if the initial general-purpose fragment bag is not perfectly tailored to a specific task, making our strategy a powerful and practically manageable tool for fine-grained control.



(a) logP (b) Ring count (c) QED

Figure 9: **Distribution of molecular properties for the MOSES dataset**. Vertical lines denote the conditioning scores applied for each target protein.

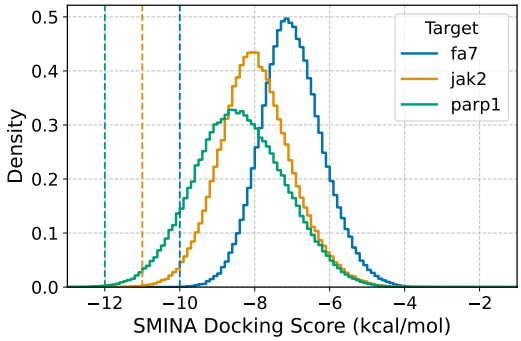
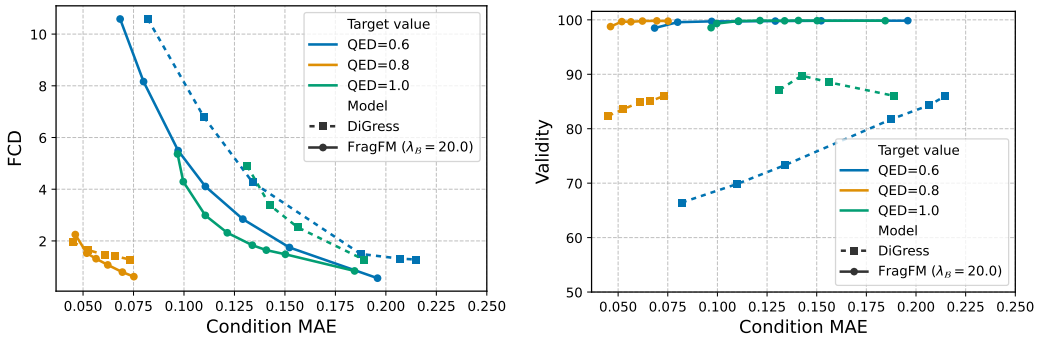
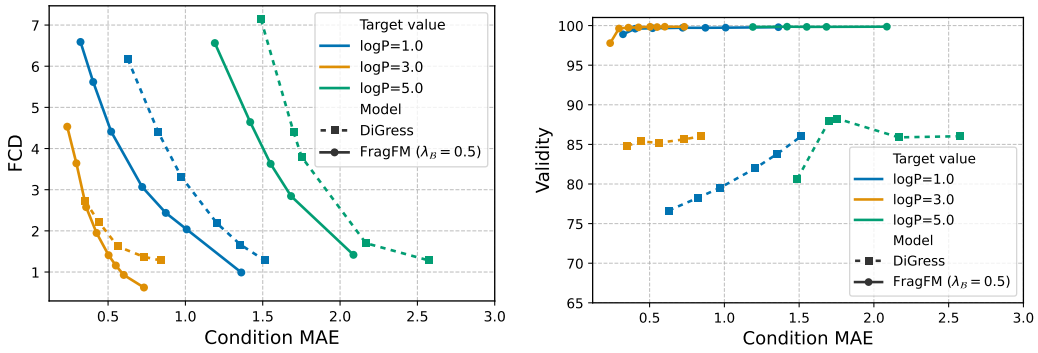


Figure 10: **Distribution of SMINA docking scores for the ZINC250K dataset across different target proteins**. Vertical lines denote the conditioning scores applied for each target protein.



(a) MAE-FCD curves. (b) MAE-validity curves.

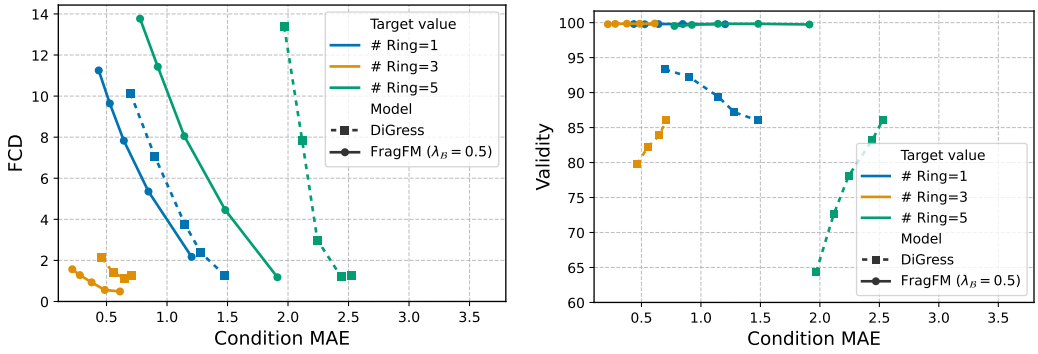
Figure 11: **Conditioning results on QED**. MAE-FCD and MAE-validity curves for FragFM and DiGress under QED conditioning on the MOSES dataset. Different conditioning values are color-coded.



(a) MAE-FCD curves.

(b) MAE-validity curves.

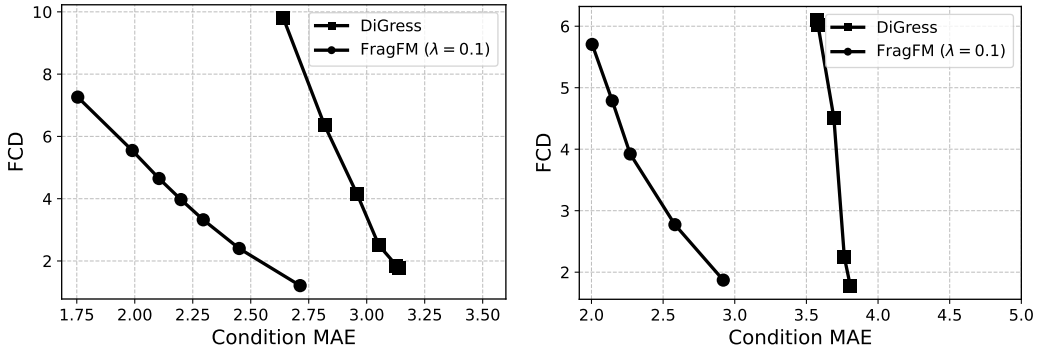
Figure 12: **Conditioning results on logP**. MAE-FCD and MAE-validity curves for FragFM and DiGress under logP conditioning on the MOSES dataset. Different conditioning values are color-coded.



(a) MAE-FCD curves.

(b) MAE-validity curves.

Figure 13: **Conditioning results on number of rings**. MAE-FCD and MAE-validity curves for FragFM and DiGress under number of rings conditioning on the MOSES dataset. Different conditioning values are color-coded.



(a) MAE-FCD curves for FA7 docking score conditioning.

(b) MAE-FCD curves for PARP1 docking score conditioning.

Figure 14: **Conditioning results on FA7 and PARP1**. MAE-FCD curves for FragFM and DiGress under FA7 and PARP1 docking score conditioning on the ZINC250K dataset. Different conditioning values are color-coded.

F Visualization

F.1 Visualization of Fragments

We visualize the top-50 frequent fragments from each dataset (MOSES, GucaMol, and NPGen).

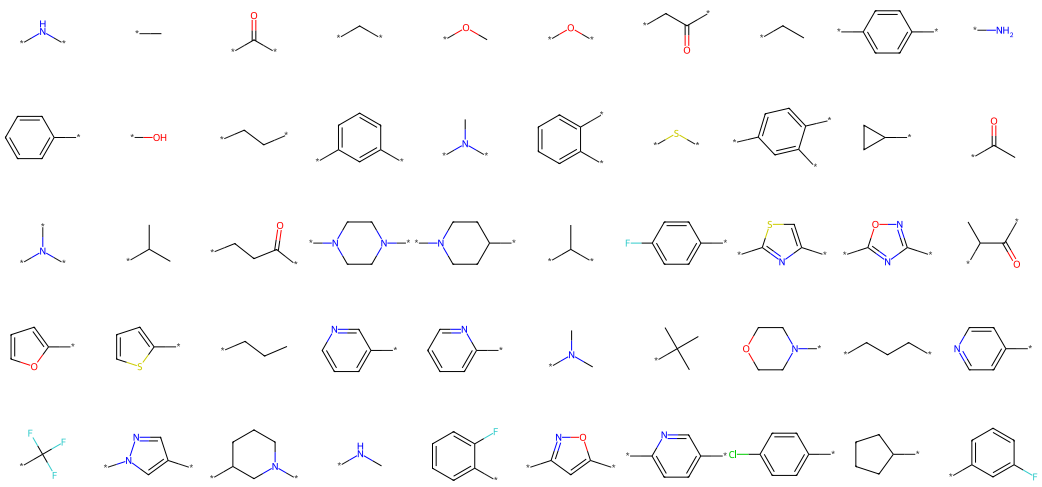


Figure 15: **Top 50 common fragments extracted from the MOSES dataset**. More frequently occurring fragments are positioned toward the top left.

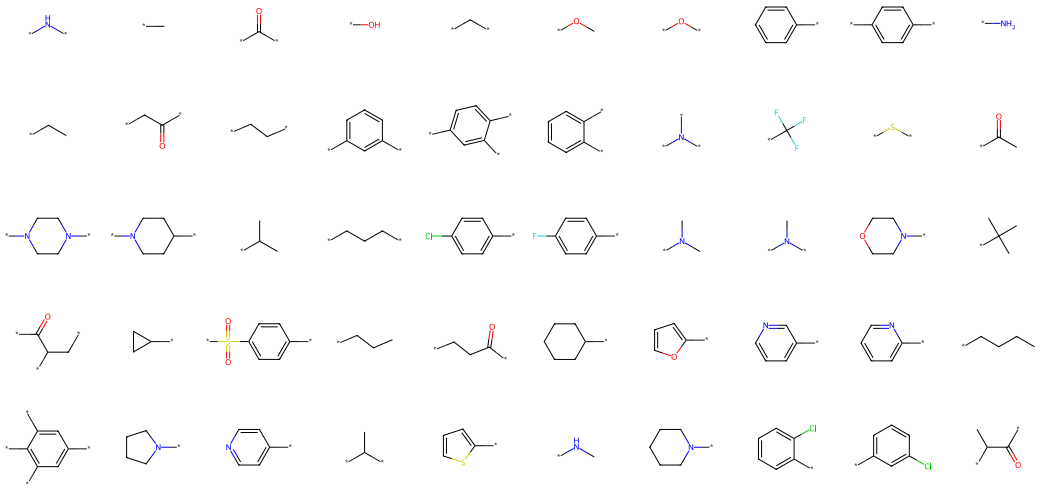


Figure 16: **Top 50 common fragments extracted from the GuacaMol dataset**. More frequently occurring fragments are positioned toward the top left.

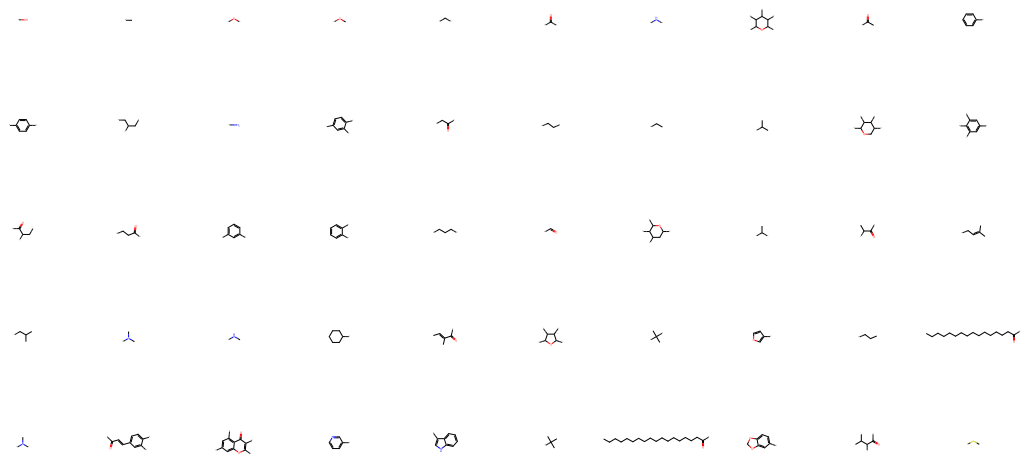


Figure 17: **Top 50 common fragments extracted from the NPGen dataset**. More frequently occurring fragments are positioned toward the top left.

F.2 Visualization of MOSES and GuacaMol Generated Molecules

We visualize samples generated by FragFM on the MOSES and GuacaMol datasets in figs. 18 and 19.

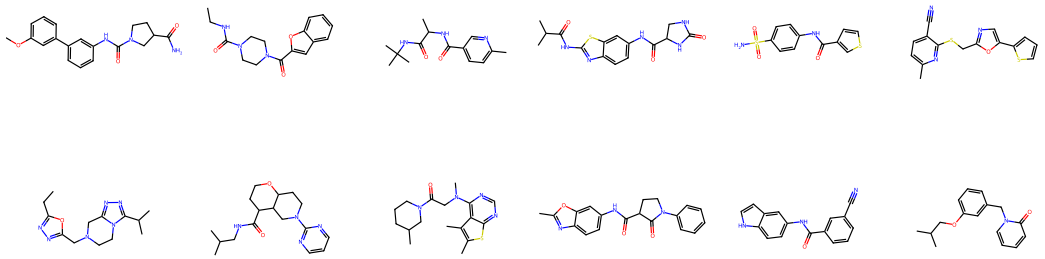


Figure 18: **Molecules generated by FragFM on the MOSES benchmark**. Molecules were randomly selected for visualization.

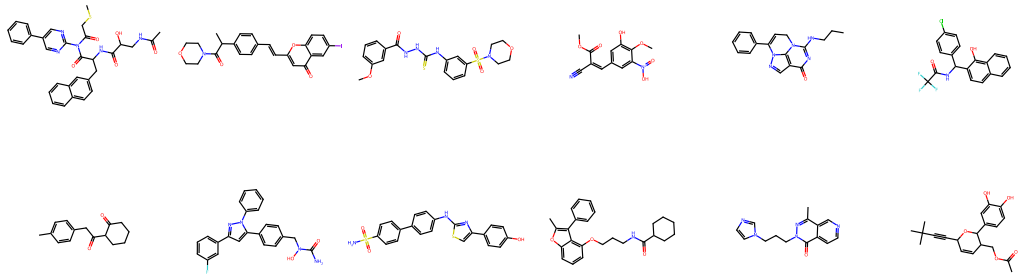


Figure 19: **Molecules generated by FragFM on the GuacaMol benchmark**. We randomly select molecules for visualization.

F.3 Visualization and Discussion of Generated Molecules on NPGen

For the NPGen task, we show generated molecules from FragFM alongside baseline models (GraphAF, JT-VAE, HierVAE, and Digress) in figs. 20 to 24. Although all visualized molecules are formally valid in terms of valency, atom-based generative models often introduce chemically implausible motifs-such as aziridine or eposide rings fused directly to aromatic systems, inducing severe angle strain [76]; anti-aromatic rings with $4n$ π-electrons (violating Hückel's rule), resulting in high electronic instability [77]; and bonds between nonadjacent atoms in a ring system, causing extreme geometric distortion [78]. Fragment-based autoregressive models largely avoid these issues, yet they, too, exhibit limitations: JT-VAE tends to generate only small, homogeneous ring systems, while HierVAE is strongly biased toward long aliphatic chains and simple linear scaffolds. Consequently, these approaches show a distinct distribution of molecules from the trained dataset, matching the benchmark results in table 2.

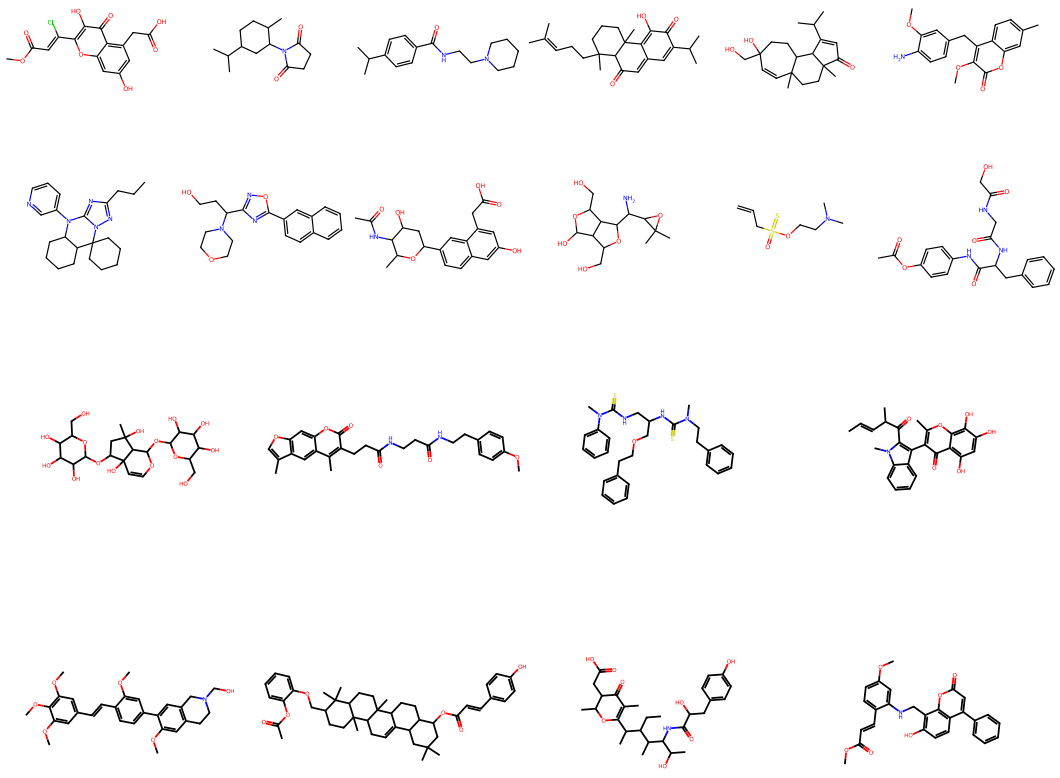


Figure 20: **Valid molecules generated by FragFM on NPGen**. The top two rows show molecules with up to 30 heavy atoms, while the bottom two rows show molecules with 31-60 heavy atoms. Molecules were randomly selected for visualization.

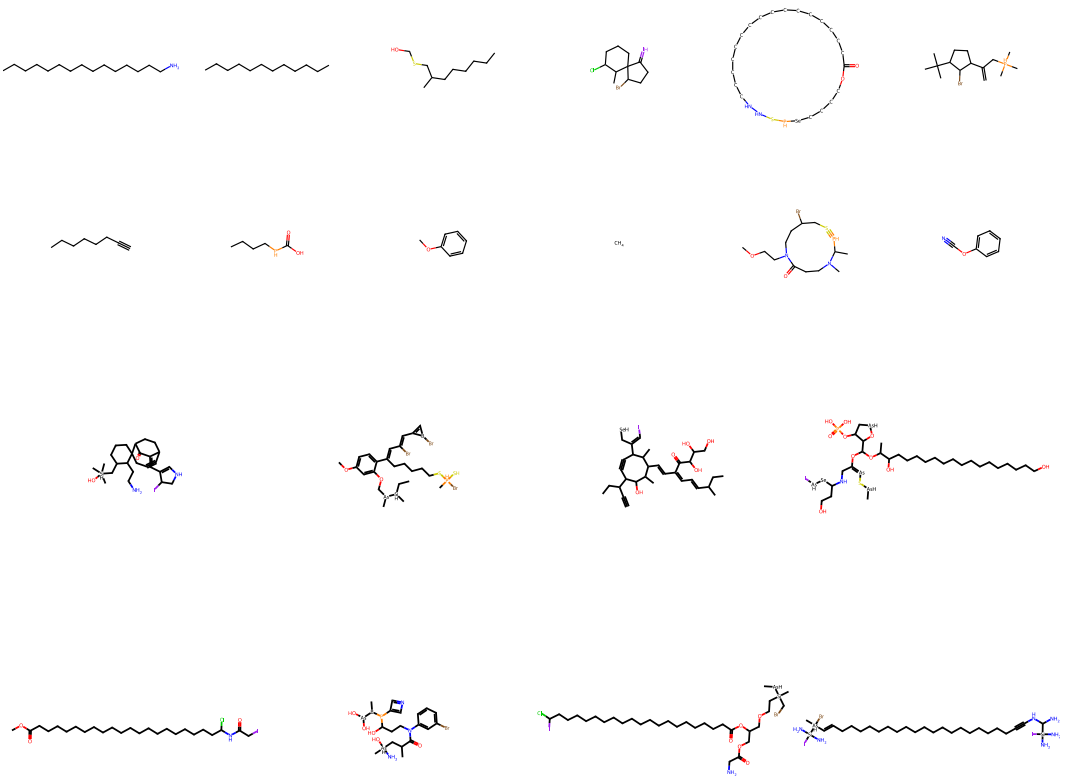


Figure 21: **Valid molecules generated by GraphAF on NPGen**. The top two rows show molecules with up to 30 heavy atoms, while the bottom two rows show molecules with 31-60 heavy atoms. Molecules were randomly selected for visualization.

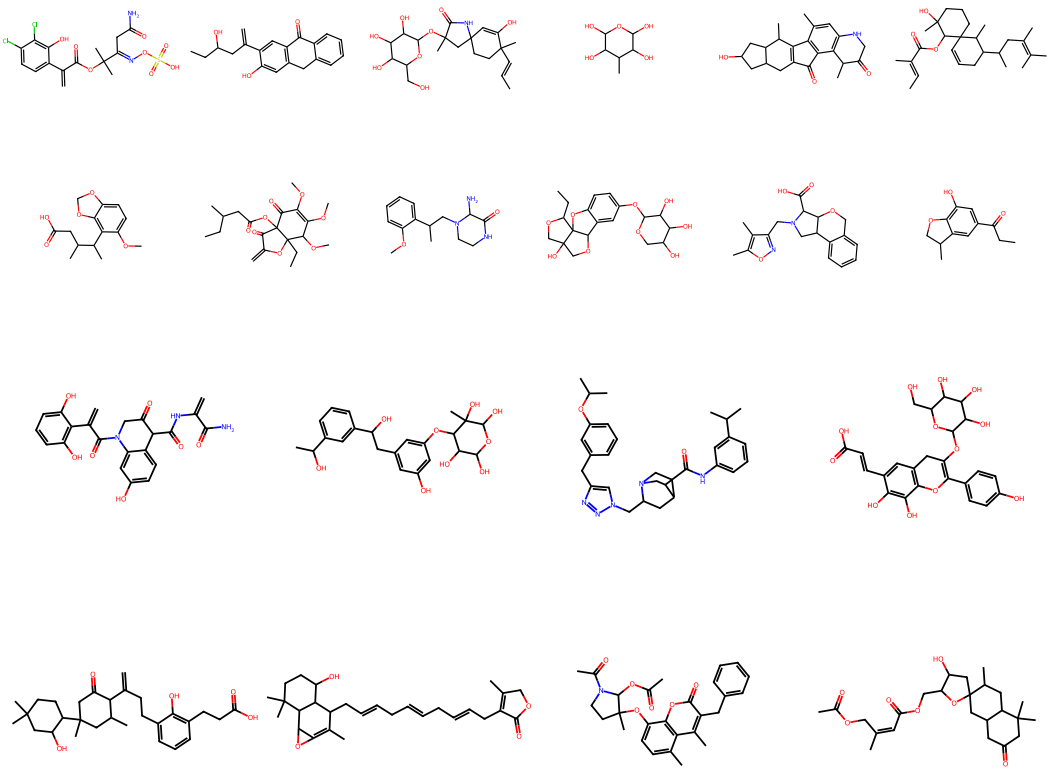


Figure 22: **Valid molecules generated by JT-VAE on NPGen**. The top two rows show molecules with up to 30 heavy atoms, while the bottom two rows show molecules with 31-60 heavy atoms. Molecules were randomly selected for visualization.

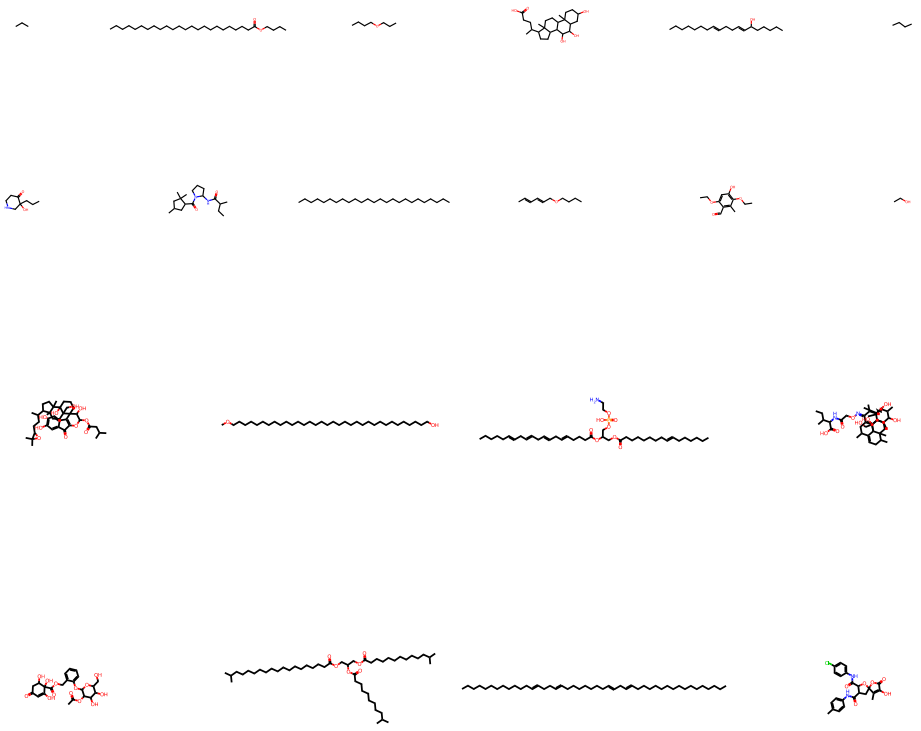


Figure 23: **Valid molecules generated by HierVAE on NPGen**. The top two rows show molecules with up to 30 heavy atoms, while the bottom two rows show molecules with 31-60 heavy atoms. Molecules were randomly selected for visualization.

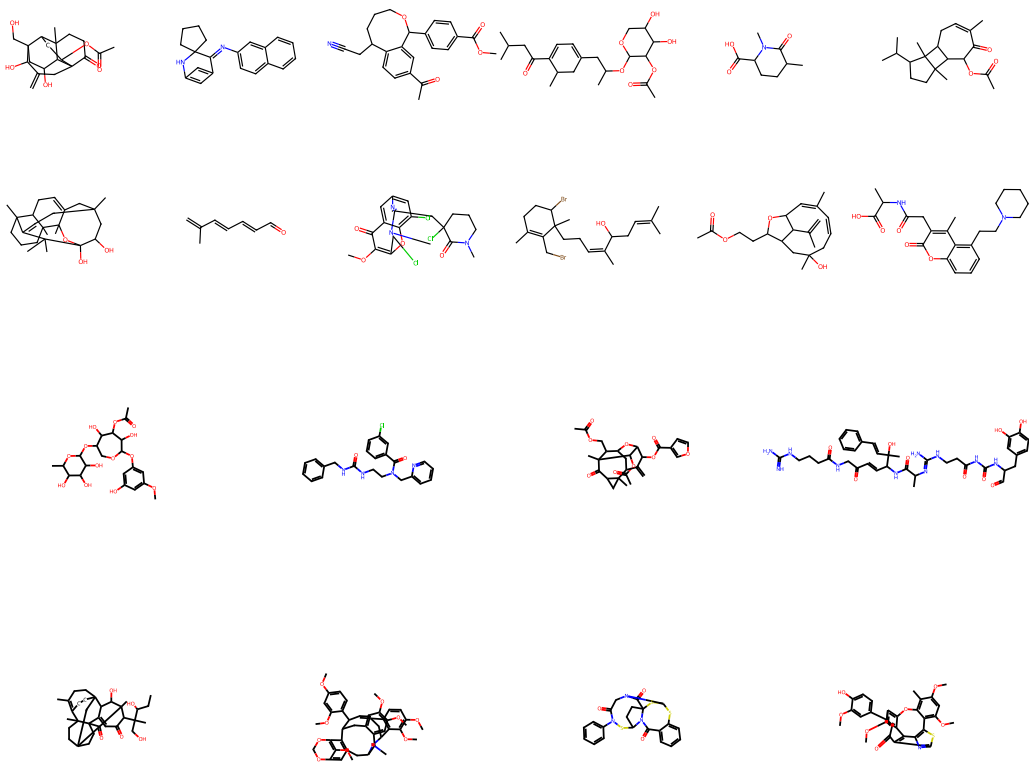


Figure 24: **Valid molecules generated by Digress on NPGen**. The top two rows show molecules with up to 30 heavy atoms, while the bottom two rows show molecules with 31-60 heavy atoms. Molecules were randomly selected for visualization.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide full information about the claims and contributions in section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We point out assumptions and approximations through out the methods section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: We do not include theorems or propositions.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper reports all datasets, models, training hyperparameters, noise schedules, guidance settings, and evaluation protocols in enough detail to let an independent reader re-implement the experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All source code and data-download scripts are included in the supplementary material, along with complete instructions and the raw experimental outputs needed to reproduce the reported results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The paper details the data splits, all hyperparameters, their selection criteria, and optimizer settings, providing sufficient information to understand and replicate the reported results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: For NPGen, our proposed benchmark, we included all averaged results with standard deviations. For the rest of benchmarks (MOSES, GuacaMol), we only report three-run averaged results for our method since the previous methods did not report their values.

Guidelines:

- The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes. We report the training time and sampling computational resources in appendix B.2

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have checked NeurIPS Code of Ethics. Also, we anonymized the information in the paper and codes.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: We propose a benchmark for natural product generation, which we expect to positively impact all machine learning and drug discovery society. Since our method and benchmark do not possess any potential of negative social impacts, we did not include it in the paper.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our method do not contain any kinds of risk of misuse, as it does not related to pretrained language models, image generators, or scrapped datasets.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: For dataset and codes we used, we properly cited following their licenses.

Guidelines:

- The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We introduce a new benchmark in section 4 and all data, code for all experiments and evaluations will be provided in zipped format as a supplementary data.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We did not conduct any kinds of Crowdsourcing experiments. All experiments are done with ourselves.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: We did not conduct any kinds of Crowdsourcing experiments. All experiments are done with ourselves.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We did not use LLMs to any theoretical foundations or modeling of our methods.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.