Received: 29 August 2022 | Revised: 17 January 2023 | Accepted: 10 February 2023

DOI: 10.1002/minf.202200215

RESEARCH ARTICLE

molecular informatics

Fragment-based deep molecular generation using hierarchical chemical graph representation and multi-resolution graph variational autoencoder

Zhenxiang Gao[1] | **Xinyu Wang**[1] | **Blake Blumenfeld Gaines**[1] | **Xuetao Shi**[1,2] | **Jinbo Bi**[1,2] | **Minghu Song**[2]

[1]Department of Computer Science and Engineering, University of Connecticut, Storrs,
and
Current address: Center for Artificial Intelligence in Drug Discovery, School of Medicine, Case Western Reserve University, Cleveland,

[2]Department of Biomedical Engineering, University of Connecticut, Storrs,

Correspondence
Minghu Song, Department of Biomedical Engineering, University of Connecticut, Storrs, CT 06269.
Email: minghu.song@uconn.edu

Funding information
NSF, Grant/Award Number: CCF-1514357, IIS-1718738

Abstract

Graph generative models have recently emerged as an interesting approach to construct molecular structures atom-by-atom or fragment-by-fragment. In this study, we adopt the fragment-based strategy and decompose each input molecule into a set of small chemical fragments. In drug discovery, a few drug molecules are designed by replacing certain chemical substituents with their bioisosteres or alternative chemical moieties. This inspires us to group decomposed fragments into different fragment clusters according to their local structural environment around bond-breaking positions. In this way, an input structure can be transformed into an equivalent three-layer graph, in which individual atoms, decomposed fragments, or obtained fragment clusters act as graph nodes at each corresponding layer. We further implement a prototype model, named multi-resolution graph variational autoencoder (MRGVAE), to learn embeddings of constituted nodes at each layer in a fine-to-coarse order. Our decoder adopts a similar but conversely hierarchical structure. It first predicts the next possible fragment cluster, then samples an exact fragment structure out of the determined fragment cluster, and sequentially attaches it to the preceding chemical moiety. Our proposed approach demonstrates comparatively good performance in molecular evaluation metrics compared with several other graph-based molecular generative models. The introduction of the additional fragment cluster graph layer will hopefully increase the odds of assembling new chemical moieties absent in the original training set and enhance their structural diversity. We hope that our prototyping work will inspire more creative research to explore the possibility of incorporating different kinds of chemical domain knowledge into a similar multi-resolution neural network architecture.

KEYWORDS
deep molecular generation, fragment-based molecular generation, graph-based deep generative model, variational autoencoder

1 | INTRODUCTION

In recent years, deep generative models, such as Variational Autoencoders (VAEs) [1] and Generative Adversarial Networks (GANs) [2], have emerged as promising approaches for automatic molecule design and chemical space exploration. Chemical structures are first expressed as the Simplified Molecular-Input Line-Entry system (SMILES) [3], and then modern machine learning models are built to learn the distribution of large amounts of training compounds and generate novel structures. Gomez-Bombarelli et al [4]. first developed a molecular VAE model to convert discrete SMILES molecular representations into continuous vector representations and generate new chemical structures by performing numerical operations on the latent representation space. Since then, many popular deep learning architectures, including different variations of autoencoders [5], recurrent neural networks (RNNs) [6,7], long short-term memory (LSTM) [6], gated recurrent unit (GRU), [8], generative adversarial networks (GANs) [9,10] and transformers [11–13] have been applied for molecular de novo generation. SMILES can also be described as parse trees. Grammar VAE [14] model was further leveraged to encode to or decode from these parse trees to ensure the structural validity of generated structures.

Several SMILES-based models [15–17] focused on scaffold-constrained or scaffold-retained molecular generation. Josep Arús-Pous et al [15]. introduced a new molecular set preprocessing algorithm to derive an exhaustive set of chemical scaffolds and their respective molecular decorators. Those scaffold-decorator pair data are further leveraged to train one of the first neural translation architectures for selectively decorating diverse input scaffolds with potentially synthesizable fragments. Several other scaffold-constrained molecular generation models have been developed in the literature recently. For example, Langevin et al [16]. built an RNN-based molecular generative model with a modified sampling procedure to perform scaffold-constrained de novo design. Their model also incorporated reinforcement learning to guide the decorative process and optimize molecular properties. Kaitoh et al [17] proposed a new hybrid scaffold-constrained method named EMPIRE that combines the deep learning-based molecular generator with the traditional fragment enumeration method. Fragments generated by the VAE and the enumeration models are merged and subsequently attached to the input scaffold to form new molecules in an arbitrary chemical subspace.

Besides the SMILES notation, a molecular structure can be described by an undirected graph where nodes and edges correspond to specific atoms or associated chemical bonds. Such molecular graph representations carry additional structural information, such as atomic properties and bond distance or angles, as node or edge attributes, which are usually absent in SMILES representation. Various deep learning architectures developed for graph-structured data can be readily applied to chemoinformatics tasks. Li′s and Simonovsky's pioneering works [18,19] show that it is feasible to directly operate VAEs or autoregressive models on the space of molecular graphs. Shi et al. proposed a flow-based autoregressive model termed GraphAF [20] which combines the advantages of both autoregressive and flow-based approaches for molecular graph generation. Other notable examples include MolGAN [21], constrained graph variational autoencoder (CGVAE) [22], Graph convolutional policy networks (GCPN) [23], graph recurrent neural networks (GraphRNN) [24]. While remarkable progress has been made, these deep molecular generative models still face many challenges. More recently, Jin et al. proposed the Junction Tree Variational Autoencoder (JT-VAE) [25] and HierVAE [26] models to improve the reconstruction accuracy and generation efficiency through the assembly of larger structural motifs, instead of individual atoms. For instance, a molecule structure in HierVAE is represented by a hierarchical graph with three distinct layers: a molecular graph layer with atoms as nodes, a motif graph layer with molecular motifs as nodes, and another attachment layer with attachment configurations as nodes. The encoder in HierVAE generates the embedding of each node in the above three graph layers. Its autoregressive decoder component reconstructs the molecule structure by sequentially sampling and attaching the structural motif to the expanding structure. Some real-world data in life science, sociology, and commerce can often be represented as such multi-layer networks [27]. For instance, a multi-layer network exemplified in Figure 1a has been applied in the multi-omics data analysis to model not only interactions taking place within the individual genomic, proteomic, or metabolomic data layer but also that crosstalk between nodes in different layers [28].

Recently, Polishchuk proposed a new computational workflow called chemically reasonable mutations (CReM) [29] for a virtual molecular generation. Like the concept of matched molecular pairs (MMPs) [30], the fundamental idea behind CReM is that fragments connecting to neighbouring structures with identical local structural environments are treated as interchangeable molecular building blocks during molecular assembly. The local structural context of the fragment around its attachment point is specified by a context radius parameter, which is defined as the maximum bond distance

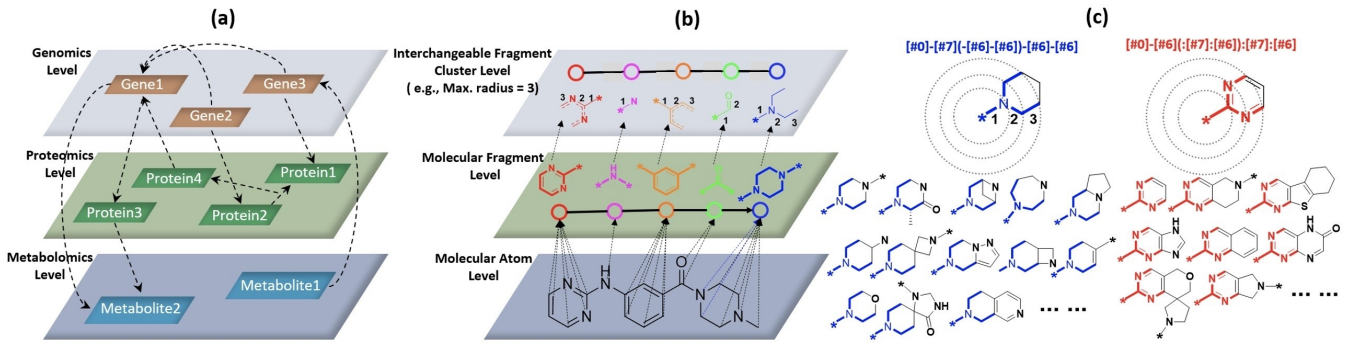

FIGURE 1 The schematic illustration of multi-layer graph or network models for (a) the multi-omics data and (b) molecular fragmentation data. After the molecular fragmentation, decomposed fragments are grouped into different clusters. Each fragment cluster contains structural analogs that share common local structural patterns near attachment points at the given context radius. (c) lists some examples of fragments that share a similar local chemical environment (highlighted in bold) as piperazinyl and pyrimidinyl groups when we set the radius as 3. During the assembly process, fragments will be sampled from the interchangeable fragment cluster and sequentially attached together to form a new structure.

from the dummy atom where the bond breaking occurs. Those decomposed fragments that share a common local structural context around fragment attachment points are grouped into the same fragment cluster. Inspired by both CReM and HierVAE studies, we implement a multi-resolution graph VAE (MRGVAE) model in this study to incorporate the concept of interchangeable decomposed fragment clusters into the multi-layer graph representation of molecules. As shown in Figure 1b and c, our three-level hierarchical representation of chemical structures now contains a new third layer that is made up of nodes representing relevant interchangeable fragment clusters of decomposed fragments. Including such exchangeable fragments may incorporate extra chemical structural knowledge, e.g., bioisosteres information, into the molecular generation process. This enables the model to generate novel structural moieties absent or rare in the original training set and thus increase the structural diversity of generated structures.

The schematic diagram of our model architecture for learning three-level hierarchical molecular graph representations is illustrated in Figure 2 below. Training compounds are first decomposed into small fragments, and the resulting fragments are further assigned to different clusters according to the local structural context around their attachment points. In this way, each molecule can be represented by three hierarchical graphs in which nodes correspond to constituent atoms, fragments, or relevant clusters of interchangeable fragments. Our encoder consists of three stacked message passing neural networks (MPNNs) [31], each of which reads one of the above three molecular graphs and outputs vector representations of constituent atoms, molecular fragments, and interchangeable fragment clusters in each layer. The molecular embeddings are derived from these node representations and fed to the decoder component. Our decoder component is built based on three multi-layer perceptions (MLPs) and reconstructs a molecule in a fragment-by-fragment manner. It iteratively samples a fragment out of the next likely fragment cluster, determines how that fragment and the previous fragment are connected, and finally attaches them together. More details of our implementation will be described in the method section.

2 | METHOD

2.1 | Molecular fragmentation and the vocabulary of interchangeable fragment clusters

The training compounds in the ChEMBL [32] database are first decomposed into small fragments that comprise rigid rings, branched flexible linkers, terminal small functional groups, etc. These structural motifs will serve as building blocks for later molecular assembly. A few molecular fragmentation approaches, such as RECAP [33], BRICS [34], eMolFrag [35] and MolBLOCKS [36], have been developed previously to decompose molecular structures into a set of reasonable constituent fragments. eMolFrag [35] utilizes the BRICS algorithm to generate fragments comprising larger scaffolds and smaller connecting linkers. Our current prototype models adopt the extended RECAP fragmentation rules from molBLOCKS, but can be extended to other more sophisticated customized rules. In each decomposed fragment, a new dummy atom is added to where the bond breaking occurs to form a pseudo-bond with the attachment point on the fragment.

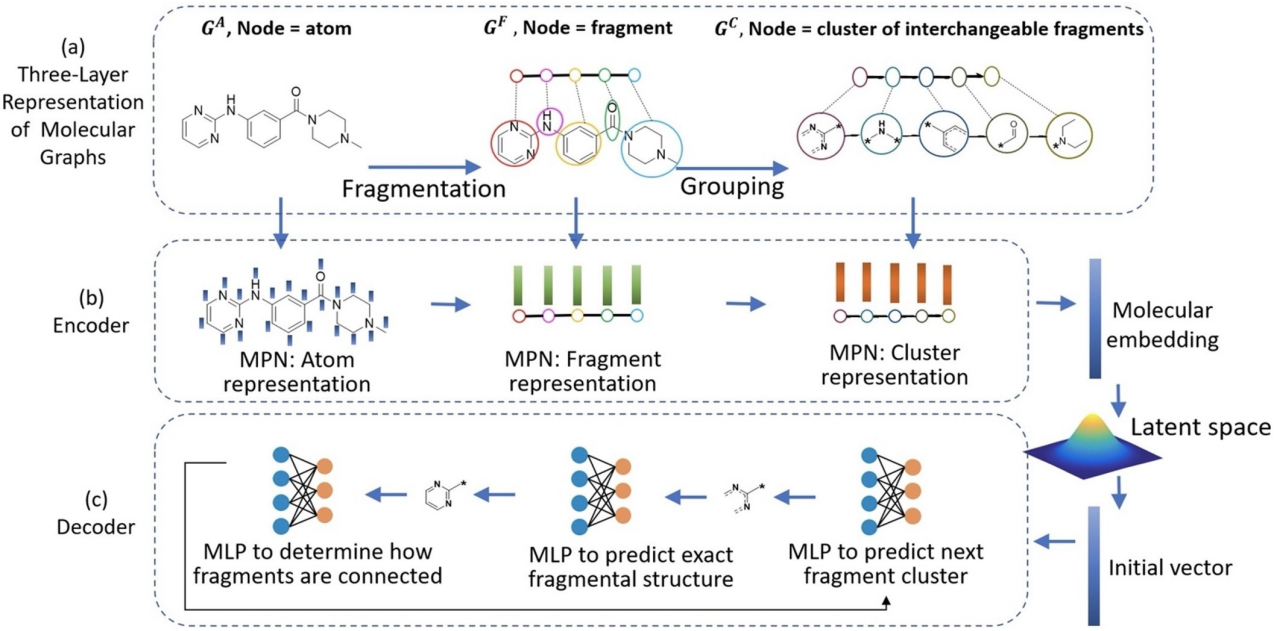

FIGURE 2 The pipeline of our MRGVAE model for deep molecular generation. (a) The illustration of the hierarchical molecular graph. It consists of three levels: atom level, fragment level, and interchangeable fragment level. (b) Our encoder accepts the hierarchical graph as input and runs three MPNNs to obtain embeddings of nodes at each level. We use the output representation of the top level as the whole molecular embedding. The VAE is trained to map the molecular embedding to a normal distribution in the latent space. (c) Our decoder samples an initial vector from latent space and makes three predictions in a coarse-to-fine manner to generate molecules fragment by fragment.

For example, the example molecule in Figure 2a is decomposed into five small fragments. In other words, the original atom-based molecular graph is transformed into a small but equivalent graph that uses these decomposed fragments as nodes. All decomposed fragments are compiled into a list after removing duplicates. Like CReM, those decomposed fragments with identical parent structures but different attachment points, such as ortho-substituted and para-substituted benzenes, will be considered as different fragments.

Collected fragments are further grouped into different interchangeable fragment clusters. CReM follows the MMP [37] algorithm to perform an exhaustive fragmentation of the input molecules and then identify a set of interchangeable fragments with a common neighboring chemical environment under a user-specified radius. A database is created to record SMARTS [38] patterns of neighboring chemical context and canonical SMILES of those interchangeable fragments in the key-value pair format, which enables the search of the next potential interchangeable fragments (values) under a given chemical context around the attachment point (key). In our current MRGVAE model, those decomposed fragments with identical chemical environments around the attachment point under the given context radius are defined as interchangeable fragments and are assigned into the same fragment cluster.

As shown in Figure 1b and c, a radius parameter is set to specify the bond distance from the newly added dummy atom and defines the local chemical environment around the dummy atom or the attachment point. We use the FindAtomEnvironmentOfRadiusN() function in RDKit [39] to extract the local context of fragments under the given attachment point or radius and then encode them with SMARTS [38] strings. Finally, for each decomposed fragment, its chemical context around the attachement point at the specified radius and the fragment structure are stored in a fragment vocabulary or lookup table in a key-value pair data structure. Some examples of interchangeable fragment clusters and their fragment members under the context radius of 3 are illustrated in Figure 1c. A fragment that contains more than one attachment position will be assigned to two or more fragment clusters. In CReM, structures are generated via different fragment enumeration operations, such as mutation, linking, and growing, by looking up the interchangeable fragment key-value pair. On the other hand, our MRGVAE model at first manages to learn how derived interchangeable-fragment clusters are mutually connected. It then builds a probability model to

estimate what would be the next interchangeable fragment cluster. According to the probability distribution of fragments in the same fragment cluster, the model samples a fragment and finally attaches it to the preceding structural moiety. We chose radius 3 as example and listed the vocabularies of interchangeble fragments, fragments and atoms in Appendix.

2.2 | Proposed MRGVAE pipeline for the molecular generation

The overall architecture of our MRGVAE model is illustrated in Figure 2 where G^A, G^F, and G^C represent the molecular graphs at the atom, fragment, and interchangeable fragment cluster levels, respectively. A VAE model generally includes two components: an encoder and a decoder. The encoder in our model consists of three units of MPNN, which are a family of deep learning models widely used for graph learning. These MPNN units are built to learn the above three-level hierarchical graph representations of molecules in a fine-to-coarse manner. Our decoder component reconstructs the molecular structure by iteratively sampling a fragment from the predicted fragment cluster and attaching it to the preceding molecular substructure. In the following paragraphs, we first briefly introduce the concept of generic MPNN and its edge-based variants, then describe how we apply the edge-based MPNN variant to obtain the latent representation of each node in a directed graph. Later we will explain our encoder and decoder components in more detail. Some mathematical notations used in our paper are listed in Table 1.

TABLE 1 Commonly used notations.

Notations	Descriptions
$\boldsymbol{G(V,E)}$	Molecular Graph
$\boldsymbol{V}$	Node set in molecular graph G
$\boldsymbol{E}$	Edge set in molecular graph G
$\boldsymbol{x_w}$	The feature vector of a node w
$\boldsymbol{x_{v,w}}$	The feature vector of an edge (u,v)
$\boldsymbol{m_{u,v}}$	The message of an edge (u,v)
$\boldsymbol{c_u}$	Identifier of interchangeable fragment u
$\boldsymbol{N(u)}$	A set containing the neighbours of node u
$\boldsymbol{x_u}$	The features of node u,
$\boldsymbol{x_{u,v}}$	The feature of edge (u,v)
$\boldsymbol{FCL()}$	A fully connected layer function
$\boldsymbol{MPN_\alpha()}$	MPN function, α is sets of parameters
$\boldsymbol{z^G}$	The representation of molecular graph G

2.2.1 | Edge-based message passing neural networks (MPNNs)

MPNN [31] is a general deep learning framework that operates on graphs with node and edge features. Several MPNN variants [40–44] have been developed in the past few years to learn the embeddings of a graph. MPNNs generally follow three operations: message passing, node updating, and readout. Generic MPNNs directly operate on graph nodes, each of which has a vector hidden representation, or sometimes called the hidden state. Each graph node's hidden state will be updated during the message passing phase by applying a predefined updating function on its current hidden state and an intermediate message aggregated from its neighbouring nodes' hidden states. Afterward, all graph nodes' hidden states are aggregated together in the final readout phase to compute the final vector representation of the whole graph. Rather than operating on graph nodes, some recent MPNNs, such as Directed MPNNs (D-MPNN) [42] or Edge Memory Neural Networks (EMNN) [45] aggregate messages from preceding adjacent edges in a directed graph. In this work, we follow the edge-updated MPNN variant in Jin et al.'s recent HierVAE work [26] to implement the VAE encoder and learn the hierarchical latent representations of molecular graphs.

For a direct graph $G = (V, E)$ where V and E are its node and edge set, $N(w)$ is the set of neighbouring nodes that forward edge message to the node w, x_w is the feature vector of the node w, and $x_{v,w}$ is the feature vector of an edge $(v,w) \in E$. For instance, for a chemical graph with atoms as nodes, the x_w vector consists of several atomic attributes of the atom w, such as its atom identifier in the graph, its atom type, associated formal charge, etc., and $x_{v,w}$ will be another feature vector characterizing the chemical bond between atoms v and w. As shown in Figure 3a, the flow of messages during the message passing phase of edge-based MPNNs is directional. The edge (v,w) in an undirected graph is associated with two edge messages: $m_{v,w}$ from v to w and vice versa. In edge-based MPNNs, each edge message will start as a zero vector, i.e., $m_{v,w}^{(0)} = 0$. Then it will run a predefined number of iterations of message passing operations to update every edge message. For example in Figure 3a, we assume that, at the $(t-1)^{th}$ iteration, two edge messages $m_{u_1,v}^{(t-1)}$ and $m_{u_2,v}^{(t-1)}$ are flowing from preceding nodes, u_1 and u_2, to the central node v, respectively. The next edge message, $m_{v,w}^{t-1}$, from the central node v to the next node w can then be updated to $m_{v,w}^{t}$ by applying the LSTM function illustrated in Figure 3b on three input arguments: the feature vector x_v of node v, the feature vector $x_{v,w}$ of edge (v,w), and the aggregation of preceding edge messages that flow toward the node v, such

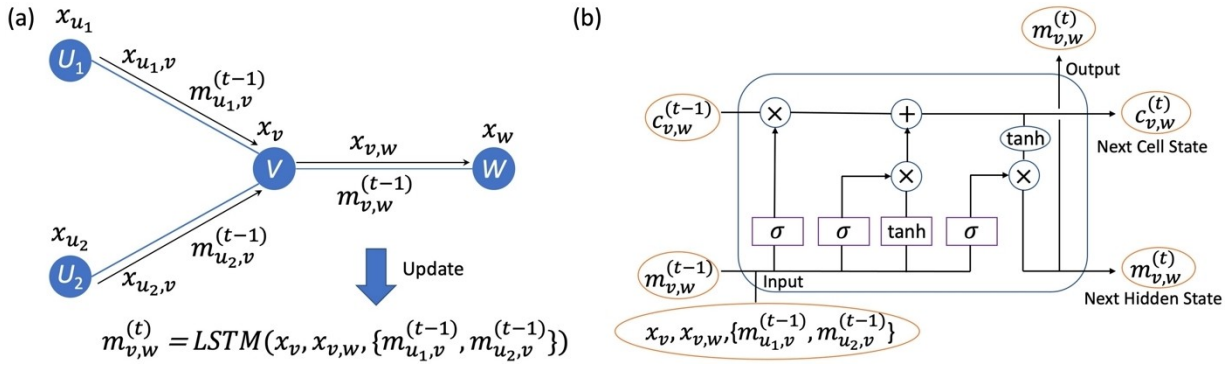

FIGURE 3 Illustration of the message passing in the edge-based MPNNs. (a): Each black arrow represents an edge message. The edge message $m_{v,w}^{t}$ flowing from the node v to w at the t^{th} iteration will be updated according to the preceding edge messages at the previous iteration, $m_{u_1,v}^{(t-1)}$ and $m_{u_2,v}^{(t-1)}$. (b): A LSTM unit is used as the message function to update message from v to w at t^{th} iteration.

as $m_{u_1,v}^{(t-1)}$ and $m_{u_2,v}^{(t-1)}$. After T iterations, a fully connected layer (FCL) function is applied to aggregate all edge messages that flow from preceding neighboring nodes to the node w and then compute the embedding of node w, i.e., $h_w = FCL(x_w, \sum_{v \in N(w)} m_{v,w}^{(T)})$ where x_w is the feature vector of node w.

2.2.2 | Encoder

Our encoder contains three MPNN units, denoted as $MPN_a(\ )$, $MPN_f(\ )$ and $MPN_c(\ )$, which are trained to compute embeddings of atoms, fragments and fragment cluster nodes in three-layer molecular graph. As illustrated in Figure 2a, the MPN_a unit reads a chemical graph G^A with atom nodes and is trained to obtain the embedding of each atom node v in G^A. Each atom has a latent vector as,

$$h_v^A = MPN_a(G^A, e(x_v), e(x_{u,v})) \tag{1}$$

where $e(.)$ is the embedding function, x_v^A is the identifier of atom v and $x_{u,v}$ is the identifier of bond (u, v).

Next, the original graph G^A is transformed to a fragment graph G^F with decomposed fragments as nodes and passed to the $MPN_f(\ )$ unit to compute the embedding of each fragment node v,

$$h_v^F = MPN_f(G^F, S_v^F, e\left(x_{u,v}^F\right)) \tag{2}$$

where $e\left(x_{u,v}^F\right)$ is the embedding vector of the bonding order between two fragment nodes. The fragment feature vector of S_v^F includes not only the fragment identifier but also a weighted sum of atom embedding vectors derived from the previous step, S_v^F is defined as,

$$S_v^F = HLN_f(e(x_v^F),\ H_v^A) \tag{3}$$

where x_v^F is the identifier of the fragment v. The sum of fragment v's atom latent vectors is denoted as H_v^A, $H_v^A = \sum_{i \in v} h_i^A$. HLN_f is one hidden layer network (HLN).

At the interchangeable fragment level, the molecular graph is converted to connections between corresponding fragment clusters of decomposed fragments. x_v^C is the identifier of the interchangeable fragment v. Their node feature, S_v^C, contains the information of the identifier of interchangeable fragment $x_v^C$and the embedding of the fragment h_v^F in the cluster learned from the preceding $MPN_f(\ )$ unit. We get S_v^C via one hidden layer network (HLN) as,

$$S_v^C = HLN_f(e(x_v^C)\ ,\ h_v^F) \tag{4}$$

The MPN_c unit output the embedding for each cluster of interchangeable fragments,

$$h_v^C = MPN_c(G^C, S_v^C, x_{u,v}^C) \tag{5}$$

where the edge feature $x_{u,v}^C$ is the binary feature between two nodes.

Once embeddings of fragment clusters are obtained, their average is computed as the final embedding for the molecule h^c. The encoder outputs the mean vector $\mu(h^c)$and standard deviation vectors $\sigma^2(h^c)$ and map them to a standard normal distribution, as

$$z^G = \mu(h^c) + \sigma^2(h^c) \cdot \varepsilon \quad \varepsilon \sim N(0,\ I) \tag{6}$$

2.2.3 | Decoder

The general VAE decoder component aims to reconstruct the original data from its latent space during the encoding process. Our decoder is trained to reconstruct a chemical structure by sequentially sampling new fragments and then assembling them to existing substructures. As schemed in Figure 4, the decoding process starts with a vector z^G sampled from the latent space of molecules. This initial z^G vector is forwarded to the decoder to estimate the probabilities p_c of interchangeable fragment clusters to which the initial fragment moiety belongs to. We sampled the fragment cluster c with the highest probability. The next step is to ascertain the exact fragmental structure because a fragmented cluster often contains several different fragments. The likelihood of every fragment member within this selected cluster is estimated by the decoder. Once an exact fragment is chosen, now we have an initial molecular subgraph $G^{t=1}$ highlighted by the red box in Figure 4b. Besides the initialization procedure, our model can also accept a given fragment input as $G^{t=1}$. This will allow us to perform the fragment-based de novo molecular generation for user-specified scaffolds. This newly generated subgraph will be passed to the trained encoder to obtain corresponding embeddings of constitutuant atoms, fragments, interchangeable fragment clusters as well as the molecular graph embedding, which are all denoted together as $h(G^{t=1})$. After the completion of the first fragment generation cycle, we are going to estimate the next possible fragment cluster $c^{t=1}$, fragment $f^{t=1}$ that will attach to $G^{t=1}$. This reconstruction process will be repeated to sample and assemble fragments sequentially until the latest sampled substructure G^t has no more additional attachment points left or until the generation has reached the predefined maximum iteration of the fragment generation.

Given current sub-hierarchical graph G^t at step t, the decoder continues to contruct a molcule using three multi-layer perceptions (MLPs), MLP_a, MLP_f, and MLP_c, with the softmax activation function in the output layer. These three MLPs are trained to estimate the probability of corresponding chemical moieties, e.g., atom, fragment, and interchangeable fragment cluster, for a given substructure state in the sequence. Specifically, the model predicts the next fragment cluster via MLP_c and softmax activation. This is cast as a classification task over the fragment vocabulary,

$$p_c = softmax(MLP_c(z^G, h(G^t)) + m_{G^t}) \tag{7}$$

where m_{G^t} is a mask which is to prevent the formation of certain undesirable bonds or edges that cannot link to G^tduring the decoding process. We adopt a similar edge-mask strategy as the one introduced in NeVAE [46] for molecular graph generation. We record all atom pairs, such as "C–C", "C–N", "C–O", "N–C", "S–O", which connect each single bond of decomposed molecules and

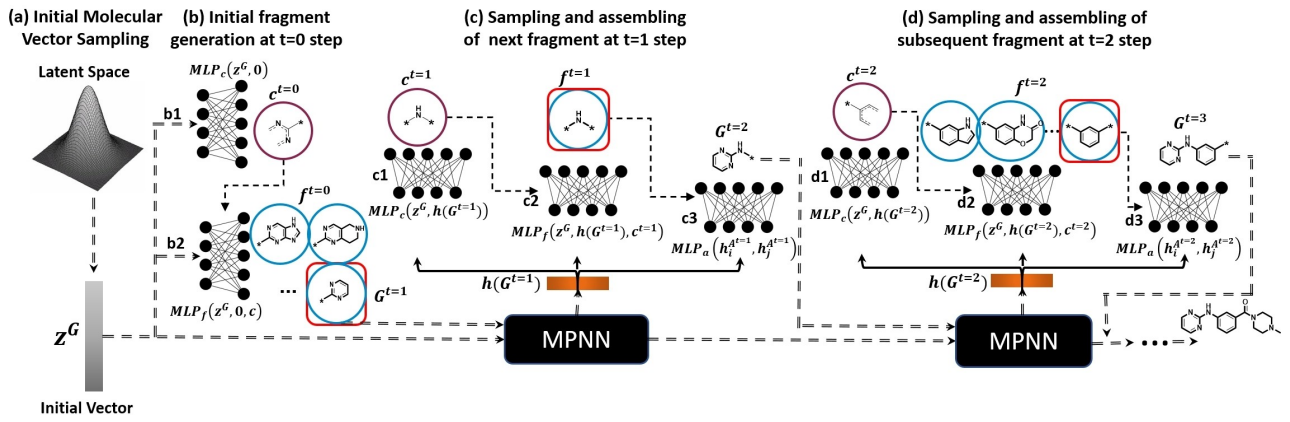

FIGURE 4 Diagram of the decoder in MRGVAE. For simplicity, only two cycles of the fragment assembly are shown here. (a) An initial vector of a molecule, z^G, is sampled from the latent space. (b) The decoder first estimates the probability distribution of interchangeable fragment vocabulary and samples the one with the highest probability as the initial interchangeable fragment cluster. Then it selects a fragment member out of this interchangeable fragment cluster as the initial molecular subgraph $G^{t=1}$ for the subsequent assembling. Alternatively, this initial $G^{t=1}$ subgraph can be specified in the model input if we want to grow structures on a given starting scaffold. (c) $G^{t=1}$ is passed to the encoder to compute its embedding, $h(G^{t=1})$. Similar to the procedure in (b), the decoder predicts what will be the next interchangeable fragment $c^{t=1}$ as well as the exact fragment graph $f^{t=1}$ via MLP_c and MLP_f, respectively. The decoder further estimates how newly sampled $f^{t=1}$ are connected to the previously generated molecular subgraph $G^{t=1}$ via MLP_a, and assembles them together to form a new molecular subgraph $G^{t=2}$. (d) Finally, the decoder iteratively attaches another new fragment to the existing molecular subgraph G^t. The procedure will terminate if no attachment point can be found on the current structure or maximum generation iterations are reached.

use them as prior knowledge to form the edge-mask and guarantee the formation of valid bonds during molecular generation.

Once the new fragment cluster c^t is suggested, the decoder will further select the next exact fragment f^t using MLP_f, the fragment probability distribution is calculated as,

$$p_f = softmax\left(MLP_f(z^G, h(G^t),\ c^t)\right) \tag{8}$$

Next, the decoder needs to decide how the newly sampled fragment f^t will attach to the previous graph G^t, because there may be several potential attaching positions available in G^t. Assume that G^t has multiple attachment positions, the third MLP unit is utilized here to determine the position with attaching atom in f^t, we can compute intermediate atom-pair embedding $l_{i,j} = MLP_a\left(h_i^{A^t},\ h_j^{A^t}\right)$, where $h_i^{A^t}$ and $h_j^{A^t}$ are their node embeddings in graphs G^t and f^t, respectively. This intermediate embedding vector will then be forwarded to a fully connected layer to estimate the connecting probability for each investigated atom pair by

$$p_{i,j} = softmax(z^G \cdot\ l_{i,j})) \tag{9}$$

When sampling a fragment from an interchangeable fragment cluster, this can be done by randomly selecting a fragment or choosing the one with the highest probability. However, fragments with the highest probability may not always be the best fragment candidates. Also, this may lead to the over-sampling of frequent fragments, such as phenyl, methyl, and halogen groups, and assembling a structure containing repetitive fragments. A similar phenomenon often appears in neural text generation tasks. The top-p [47] sampling approaches have recently been introduced to alleviate the repetition of highly frequent words during the text generation. The top-p sampling, also known as nucleus sampling, chooses the smallest set of tokens whose total probability is larger than p to reduce the chances of generating less useful tokens and preserve certain token diversity. In MRGVAE, we implement two sampling approaches, including top-p and random methods, during the decoding process to inspect whether they can further improve the quality of generated structures.

2.3 | Model training and implementation

After the molecular fragmentation, we apply a depth-first search algorithm to traverse the graph of fragments and the graph of fragment clusters and convert them into a sequence of graph nodes and edges in a machine-readable format. The objective of our VAE training is to minimize the following negative evidence lower bound (ELBO) calculated over the entire training set of molecules:

$$\begin{aligned} L_{VAE}(\phi, \theta, G) = -E_{z\sim Q}[\log P_\theta(G|z^G)] + \\ \lambda_{KL} D_{KL}\left[Q_\phi\left(z^G|G\right)||P(z^G)\right] \end{aligned} \tag{10}$$

The first term in the above loss function is the reconstruction error or expected negative loglikelihood, which urges the decoder to learn how to reconstruct the molecular graph. The generative probabilistic distribution $P_\theta(G|z^G)$ of molecular graphs conditioned on a latent vector z^G consists of three components parameterized by θ: p_c is the probability of a fragment cluster connecting to the preceding molecular subgraph G, p_f is the probability of an exact fragment structure connecting to the preceding molecular subgraph G, and finally $p_{i,j}$ is the connection probability between a particular atom i in the newly sampled fragment and another atom j in preceding graph G. One of the decoder's main objectives is to minimize the total cross-entropy loss between the estimated p_c, p_f, and $p_{u,k}$ distributions and ground-truth distributions from the training data. The second term is a regularizer, Kullback-Leibler divergence, which quantifies how much information will be lost if the prior Gaussian distribution $P\left(z^G\right)$ is used to approximate the encoder's posterior distribution $Q_\phi\left(z^G|G\right)$. The minimization of the KL divergence ensures that $Q_\phi\left(z^G|G\right)$ is similar to $P\left(z^G\right)$. A regularization hyperparameter, λ_{KL}, with the default value of 0.1 in our experiments, controls the balance between the above two loss terms in the objective function.

Our deep molecular generative models are built on PyTorch 1.1.0 [48] and trained with the Adam algorithm. In this study, the size of the mini-batch is set to 32. The dimension of both hidden layers and the latent vectors are all set to 250. The VAE network is trained for ten

epochs with a learning rate of 0.001. RDKit [39] (2020.03.3 version) is utilized for some data preprocessing and post hoc analyses, such as analyses of the distribution of molecular properties. Other required python libraries, such as networkx 2.4 for the graph process and data storage, are listed in the GitHub readme file. All deep learning models are trained on one Nvidia Tesla V-100 card and 32 CPU cores using CUDA 10.

2.4 | Model training and implementation

ChEMBL version 28 [32] is used and preprocessed as the molecular training set in this study. We first remove some irrelevant components or structures, e.g., saltions, solvents, SMILES containing isotopes, duplicate compounds, etc. Since the main focus of this study is to check how well those deep learning-based generative models perform on the traditional small molecular drug-like chemical space. Therefore, those extra large, greesy or flexible compounds are removed according to some structural descriptors, such as molecular weight, number of hydrogen bond donors/acceptors, the ring size, etc. We also check the remaining molecules using some empirical SMARTS [38]-based structural filters to further remove those toxic or over-reactive structures. The preprocessing python codes and employed SMARTS rules are described in Data and Software Availability section. All the above data preprocesses result in nearly 1 M remaining structures, which are decomposed into around ten thousand unique fragments. Some statistics of the resulted training set and decomposed fragments are summaried in Appendix Figure S1.

Besides the JT_VAE and HierVAE models mentioned earlier, we also include several other molecular graph generative models as baselines in this paper. NAT GraphVAE [49] is a graph variational autoencoder for molecular graph generation in a non-autoregressive manner. $NAGVAE_{compress}$ [50] is another non-autoregressive graph VAE model, in which six prevalent substructural patterns in real-world molecules are converted to graph edges and attributes by the graph compression method. We also include SMILES LSTM [6] for the comparative study.

Several metrics are employed to evaluate the performances of various deep molecular generative models. Some are directly adopted from the MOSES benchmark [51]. They include internal diversity ($IntDiv_p$), structural validity, uniqueness, and novelty. In addition to the distance-based diversity, we also include the coverage-based diversity measurement here. Given a predefined functional group (FG) or ring set (RS), they measure how well these reference fragments or rings are covered in the newly generated molecular set. Such coverage-based methods have been used as the surrogate of chemical diversity analysis in some recent chemoinformatics literature [52,53].

3 | RESULTS AND DISCUSSIONS

In this section, we carry out a series of experiments to (1). compare MRGVAE's performance with other graph- or SMILES-based baseline models over some molecular benchmark metrics; (2). explore the influence of the context radius parameter on the physicochemical distribution profile of structures generated by MRGVAE; (3). investigate how different sampling techniques during the decoding process affect physicochemical profiles of generated structures.

3.1 | Molecular metric evaluation of structures generated by the MRGVAE model and other baseline models

This section will assess the quality of structures generated by MRGVAE and how well models learn to generate molecules that are similar to the training set. We first use the MRGVAE model to assemble 50,000 structures without specifying any starting fragments. One hundred examples of generated structures are demonstrated in Figure 5 below. In this initial experiment, radius and top-p parameters are set as 3 and 0.99, respectively. Since the model performance will vary with different choices of sampling methods and input parameters, such as the fragment context radius, we also investigate the effect of varying sampling approaches and context radii on the model performance. In this study, we also try two different sampling methods, random and nucleus sampling, to find the subsequent fragments during the molecular decoding process. More analyses about the influence of different context radii or sampling methods will be discussed in sections 3.2 and 3.3.

Results of different molecular generative models are summarized in Table 2. Our MRGVAE models demonstrate competitive performance compared to other baseline models. As shown in the table, our MRGVAE model with a context radius of 2 and random sampling yields comparable or higher scores in terms of validity, uniqueness, novelty, diversity, functional group coverage, and system ring coverage. The improvement of the structural diversity and novelty by our three-level VAE model is partially related to the introduction of additional

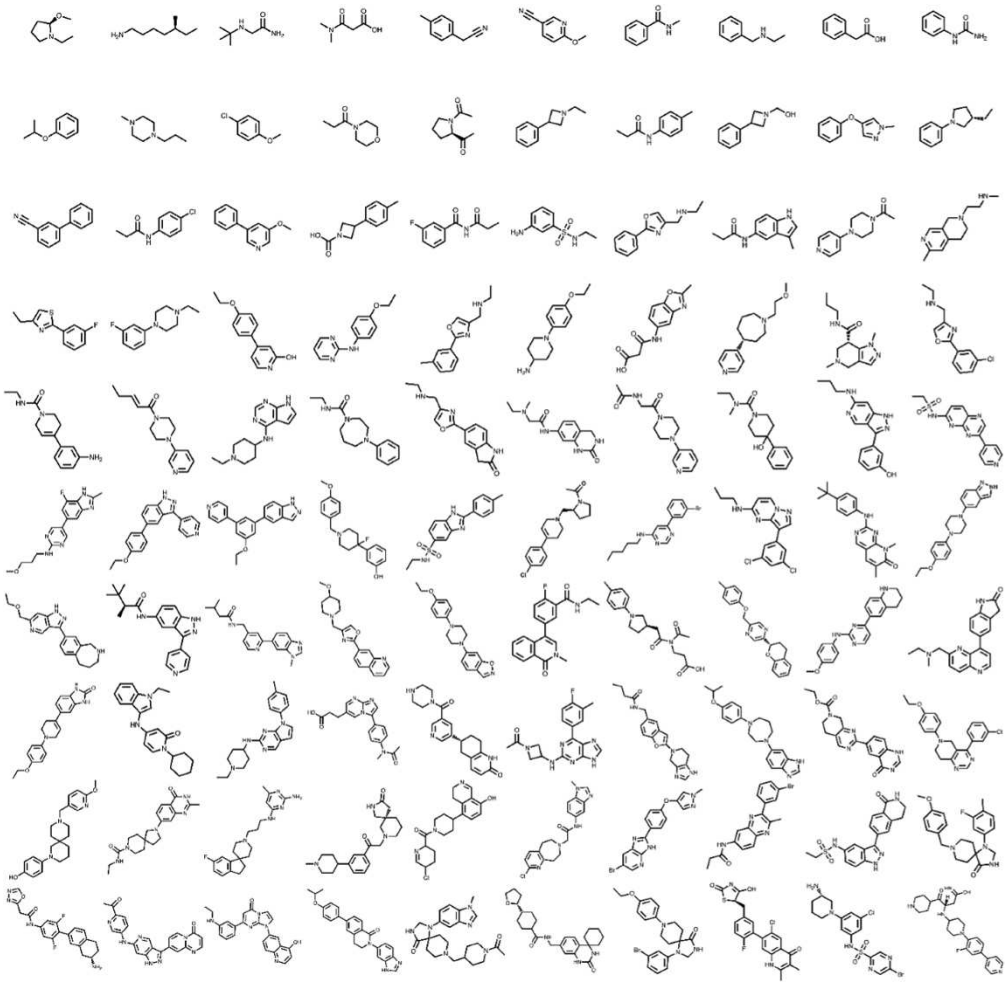

FIGURE 5 A hundred examples randomly sampled out of 50,000 generated structures generated by the MRGVAE model (the radius parameter and the top-p values are set as 3 and 0.99, respectively.).

TABLE 2 Generation performance comparison of baseline and proposed models.

Method	Validity	Uniqueness	Novelty	IntDiv	FG Coverage	RS Coverage
SMILES LSTM	0.945	0.996	0.959	0.872	0.508	0.365
NAT GraphVAE	0.831	0.866	1.000	0.823	0.159	0.025
$NAGVAE_{compress}$	0.932	0.885	1.000	0.827	0.139	0.033
JT-VAE	1.000	0.997	0.995	0.867	0.476	0.301
HierVAE	1.000	0.952	0.976	0.886	0.492	0.263
MRGVAE with Radius 2 (Top-p = 0.99)	1.000	0.955	0.995	0.887	0.419	0.464
MRGVAE with Radius 2 (Random)	1.000	0.995	1.000	0.888	0.594	0.649
MRGVAE with Radius 3 (Top-p = 0.99)	1.000	0.945	0.994	0.885	0.338	0.314
MRGVAE with Radius 3 (Random)	1.000	0.963	0.995	0.886	0.402	0.475

fragment candidates from the same interchangeable fragment cluster, which may result in novel structural moieties absent in the training set. However, we also observe a much larger distribution divergence of calculated chemical properties when random sampling is adopted in the decoder. Such property distribution divergence can be lowered by exploring top-p methods, as discussed in section 3.3.

3.2 | The influence of the context radius on the molecular generation

To study the potential effect of different fragment context radii on the molecular generation, we try five different context radii ranging from 1 to 5 and build five corresponding models for each radius using the same ChEMBL dataset. In the previous section, we sampled the fragment with the highest probability in the fragment vocabulary as the initial fragment. However, different initial fragments would also affect the size and molecular properties of generated structures. To alleviate the data variation caused by different initial fragments, we extract 200 structurally diverse druglike fragments or scaffolds from the fragment vocabulary. Structures of fifty out of these two hundred scaffolds are listed as initial fragment examples in Appendix Figure S2. To simplify the follow-up analysis, we only extract those particular fragments containing only one attachment point and exclude those tiny fragments, e.g., the methylene spacer or ether linkage, whose fragment radius from the attachment point is less than four. Each of these 200 fragments will serve as a new starting point for assembling 200 different molecules. Thirty examples of quinazolinone derivatives generated by our MRGVAE model are shown in Figure 6.

We also compute scores of structural uniqueness, novelty, and diversity for each set of structures derived from one of the 200 scaffolds and draw the boxplots in Figure 7. Average uniqueness and diversity scores of generated molecules from 200 different scaffolds substantially dropped when models were constructed using a larger context radius to define interchangeable fragments. Increasing the context radius usually leads to fewer fragments in the same interchangeable cluster, which will decrease the structural diversity and cause the model to generate more structures identical to the original training compounds.

However, when we adopt smaller context radii, we observe a larger divergence of the physicochemical property distributions between generated and training compounds. The distribution of four calculated molecular properties, lipophilicity (LogP), molecular weight (MW), chemical synthetic bulky accessibility score (SA), and quantitative estimation of drug-likeness (QED), for structures generated under different radii are shown in Figure 8. When we adopt a small context radius, there is less structural restriction for the selection of the next

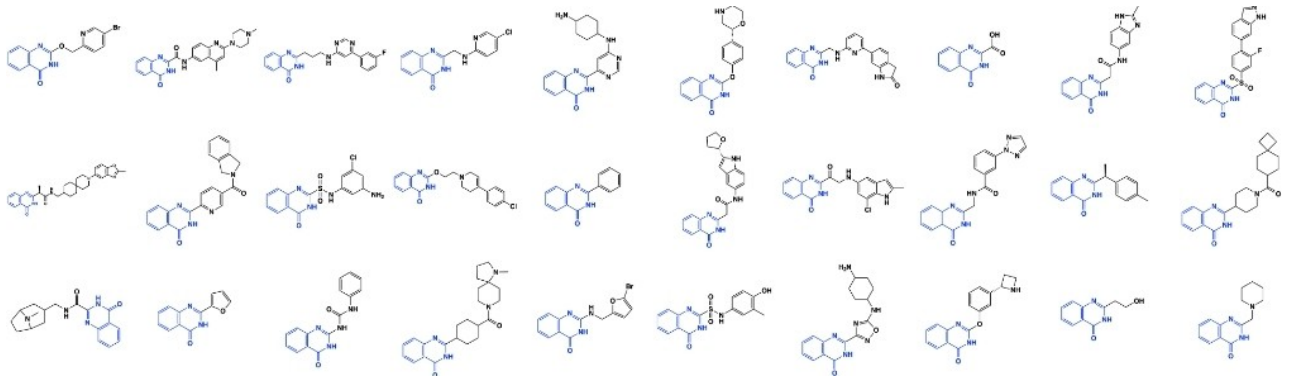

FIGURE 6 Thirty examples of quinazolinone derivatives generated by the MRGVAE model (with the radius parameter and the top-p value of 3 and 0.99).

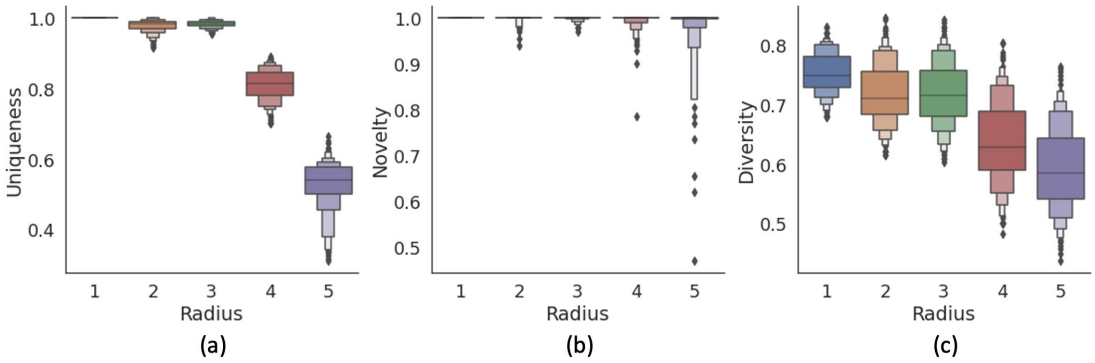

FIGURE 7 Boxplots of uniqueness, novelty, and diversity scores obtained from 200 scaffold derivatives under different context radii ranging from 1 to 5.

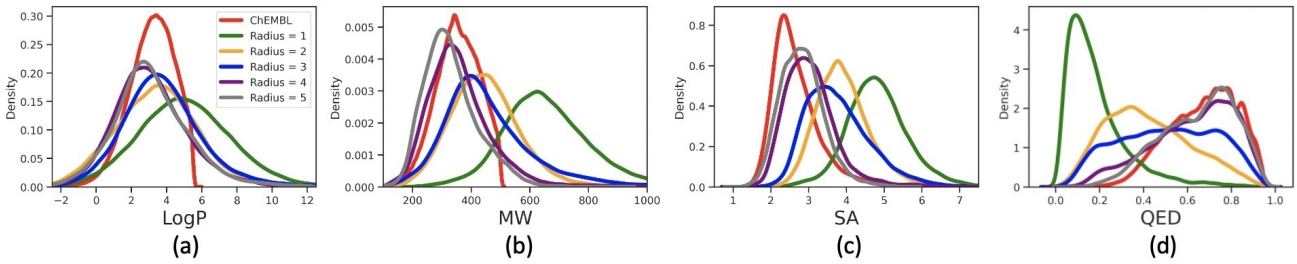

FIGURE 8 Density Plots of chemical properties for the ChEMBL training set (highlighted in red) and 40,000 structures derived from 200 different scaffolds with radii ranging from 1 to 5. (a) The distribution of lipophilicity (LogP). (b) The distribution of Molecular Weight (MW), (c) The distribution of chemical synthetic accessibility score (SA). (d) The distribution of drug-likeness (QED).

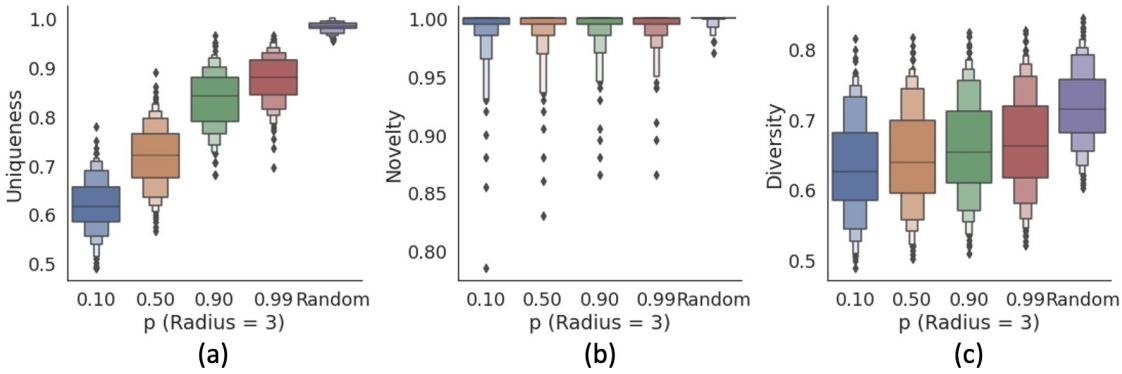

FIGURE 9 Effect of top-p sampling methods on the uniqueness, novelty, and diversity scores of generated structures.

fragment, which increases the possibility of attaching more fragments. Thus, we will observe a significant rightward shift of the distributions of LogP and molecular weight under a smaller context radius. On the other hand, such distribution discrepancies will be alleviated when increasing the context radius.

3.3 | The effect of various sampling parameter setting on chemical property profiles of generated structures

Inspired by several recent studies [47] showing that the quality of neural text generation can be improved by adopting top-p and random sampling method, we implement these two methods in our decoder to check whether they can improve the quality of generated structures. The effect of various sampling techniques along with context radii on uniqueness, novelty, and diversity scores of generated molecules are illustrated in Figure S3 of the Appendix. To save the space, here we only show results when the context radius is set to 3 because the other radii follow a similar general trend. Boxplots of uniqueness, novelty, and diversity scores produced by different parameters p of top-p sampling method are demonstrated in Figure 9. For both sampling methods, when we increase the values of p, more less-frequent fragments within the same fragment cluster will be sampled. This will result in the monotonic improvement of uniqueness and diversity scores. In addition, because the top-p sampling method keeps only fragments with either the highest probability or cumulative probability larger than the cutoff, the fragment candidate pool for top-p sampling will be much smaller and restricted than in the random sampling case. Therefore, the top-p method will yield lower uniqueness and diversity scores than the random method.

As shown in Figure 10, We also check how LogP, MW, SA, and QED distributions shift under different sampling methods or parameters. The context radius of 3 is chosen as the example here again, while more property distributions under various radii parameters and sampling methods are included in Appendix Figure S4. Similar to the effect of the context radius parameter, the distributions of these molecular properties start to shift right when we increase p values. When p is small, it will be more equivalent to selecting the fragment with the highest probability learned from the ChEMBL training set. Therefore, under this scenario, the trained generative model generates a new set of molecules with more similar property distribution profiles as the training ChEMBL data. When we increase values of p, as

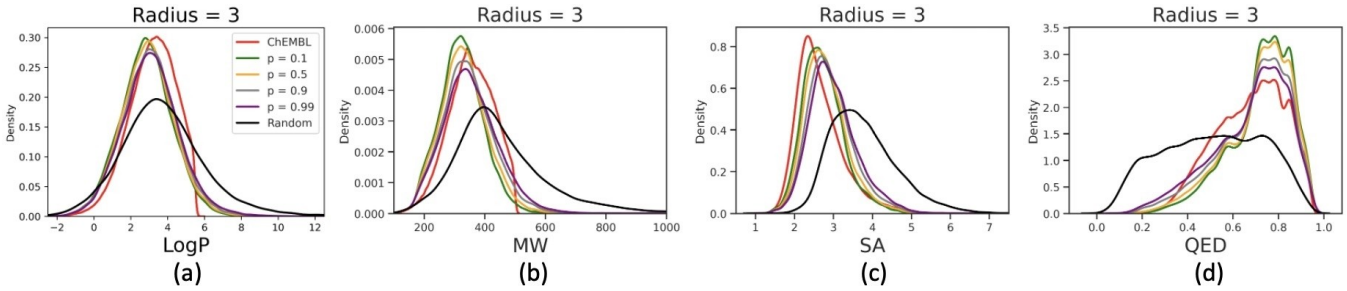

FIGURE 10 Kernel density plots of molecular property distributions when using different sampling techniques.

shown in Figure 9, diversity and novelty scores of generated structures will increase. But as discussed earlier, a concomitant effect is that the distribution-learning capability is weakened when the novelty scores of generated structure increase. As shown in Figure 10, in contrast to top-p method, the random sampling method produces a molecular set with a much higher calculated logP, molecular size, and poor synthetic accessibility score.

4 | CONCLUSIONS

In this paper, we present a new graph-based variational autoencoder model for the virtual molecular generation. Unlike other motif-based molecular graph generative models, we further group decomposed fragments into different interchangeable fragment clusters according to their local structural environment so that the chemical structure can be represented by a new molecular graph, which uses these interchangeable fragment clusters as graph nodes. Our new VAE model is constructed to learn such three-layer hierarchical graph representations of chemical structures in a fine-to-coarse order. Atoms, decomposed fragments, and related fragment clusters act as graph nodes at each corresponding graph layer. Its decoder component is designed to iteratively select a fragment out of a predicted fragment cluster vocabulary and then attach it to the preceding substructure. Our proposed prototype approach demonstrates comparative performance in terms of several molecular evaluation metrics when compared with several other graph- and SMILES-based generative molecular models. Moreover, the interchangeable fragment graph layer may enable us to indirectly inject additional chemical structural knowledge, e.g., bioisosteres information, into the molecular generation process and increase the odds of assembling novel chemical moieties absent in the original training set.

ACKNOWLEDGMENTS

The implementation of graph-based VAE and related undergraduate *research* were supported in part by NSF grants, CCF-1514357 and IIS-1718738.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

Our graph-based MRGVAE models are implemented in Python 3.6.10. Its python source codes and the readme file are available on our public GitHub repository: https://github.uconn.edu/mldrugdiscovery/MRGVAE. Other three graph-based molecular generative models (JT-VAE, HierVAE, and CGVAE) are downloaded from their published repositories: https://github.com/wengong-jin/icml18-jtnn, https://github.com/wengong-jin/hgraph2graph, and https://github.com/microsoft/constrained-graph-variational-autoencoder. We adopt their default model parameters when applying these three graph-based molecular generative models to our own training set. SMILES strings of the preprocessed ChEMBL training data as well as 200 structurally diverse scaffolds utilized in this study for the scaffold-based molecular generation are available under the data directory of our MRGVAE GitHub repository.

We wrote in-house Python scripts, StructCleansing, to handle the preprocessing of ChEMBL dataset, which is available under https://github.uconn.edu/mldrugdiscovery/ ChemStructClean.git. It is implemented based on two publically available curation pipelines of chemical structures: ChEMBL_Structure_Pipeline [54] and rd_filters [55]. Users can customize our ChemStructClean pipeline to exclude certain classes of compounds by adding or modifying particular SMARTS filters or molecular descriptor criteria. Molecular descriptor filters are specified in the "rules_one_step.json" file, while SMARTS rules filers are listed within the "alert_collection.csv" file. Both of these files can be found in our code repository.

ORCID

Zhenxiang Gao http://orcid.org/0000-0002-2223-5092

REFERENCES

1. D. P. Kingma, M. Welling, *ICLR* **2014**.
2. I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, *NIPS* **2014**, 2672–2680.
3. D. Weininger, *J. Chem. Inf. Comput.* **1988**, *28* (1), 31–36.
4. R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, A. Aspuru-Guzik, *ACS Cent. Sci.* **2018**, *4* (2), 268–276.
5. A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper, A. Zhavoronkov, *Mol. Pharm.* **2017**, *14* (9), 3098–3104.
6. M. H. S. Segler, T. Kogej, C. Tyrchan, M. P. Waller, *ACS Cent. Sci.* **2018**, *4* (1), 120–131.
7. M. Olivecrona, T. Blaschke, O. Engkvist, H. Chen, *J. Cheminformatics* **2017**, *9* (1), 1–14.
8. J. Arús-Pous, T. Blaschke, S. Ulander, J. L. Reymond, H. Chen, O. Engkvist, *J. Cheminformatics* **2019**, *11* (1), 1–14.
9. G. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias, A. Aspuru-Guzik, *arXiv.* **2017**.
10. O. Prykhodko, S. V. Johansson, P. C. Kotsias, J. Arús-Pous, E. J. Bjerrum, O. Engkvist, H. Chen, *J. Cheminformatics* **2019**, *11* (1), 1–13.
11. Y. Yang, S. Zheng, S. Su, C. Zhao, J. Xu, H. Chen, *Chem. Sci.* **2020**, *11* (31), 8312–8322.
12. V. Bagal, R. Aggarwal, P. K. Vinod, U. D. Priyakumar, *J. Chem. Inf. Model.* **2021**, *62*(9), 2064–2076.
13. J. Wang, C. Y. Hsieh, M. Wang, X. Wang, Z. Wu, D. Jiang, et al., *Nat. Mach. Intell.* **2021**, *3*(10), 914–922.
14. M. J. Kusner, B. Paige, J. M. Hernández-Lobato, *ICML* **2017**, 1945–1954.
15. J. Arús-Pous, A. Patronov, E. J. Bjerrum, C. Tyrchan, J. L. Reymond, H. Chen, O. Engkvist, *J. Cheminformatics* **2020**, *12* (1), 1–18.
16. M. Langevin, H. Minoux, M. Levesque, M. Bianciotto, *J. Chem. Inf. Model.* **2020**, *60*(12), 5637–5646.
17. K. Kaitoh, Y. Yamanishi, *J. Chem. Inf. Model.* **2022**, *62*(9), 2212–2225.
18. Y. Li, O. Vinyals, C. Dyer, R. Pascanu, P. Battaglia, *arXiv.* **2018**.
19. M. Simonovsky, N. Komodakis, *ICANN* **2018**, 412–422.
20. C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, J. Tang, *ICLR* **2020**.
21. N. De Cao, T. Kipf, *arXiv.* **2018**.
22. Q. Liu, M. Allamanis, M. Brockschmidt, A. L. Gaunt, *NIPS* **2018**, 7795–7804.
23. J. You, B. Liu, R. Ying, V. Pande, J. Leskovec, *NIPS* **2018**, 6410–6421.
24. M. Popova, M. Shvets, J. Oliva, O. Isayev, *arXiv.* **2019**.
25. W. Jin, R. Barzilay, T. Jaakkola, *RSC Drug Discov. Ser* **2021**, 228–249.
26. W. Jin, R. Barzilay, T. Jaakkola, *arXiv.* **2020**.
27. M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, M. A. Porter, *J. Complex Netw.* **2014**, *2* (3), 203–271.
28. M. Gosak, R. Markovič, J. Dolenšek, M. Slak Rupnik, M. Marhl, A. Stožer, M. Perc, *Phys. Life Rev.* **2018**, *24*, 118–135.
29. P. Polishchuk, *J. Cheminformatics* **2020**, *12* (1), 1–18.
30. P. W. Kenny, J. Sadowski, *Chemoinform. Drug Discov.* **2005**; *23*, 271–285.
31. J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, *ICML* **2017**, 2053–2070.
32. A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, McGlinchey, et al., *Nucleic Acids Res.* **2012**, *40* (D1), D1100–D1107.
33. X. Q. Lewell, D. B. Judd, S. P. Watson, M. M. Hann, *J. Chem. Inf. Comput.* **1998**, *38* (3), 511–522.
34. J. Degen, C. Wegscheid-Gerlach, A. Zaliani, M. Rarey, *ChemMedChem* **2008**, *3* (10), 1503–1507.
35. T. Liu, M. Naderi, C. Alvin, S. Mukhopadhyay, M. Brylinski, *J. Chem. Inf. Model.* **2017**, *57* (4), 627–631.
36. D. Ghersi, M. Singh, *Bioinform.* **2014**, *30* (14), 2081–2083.
37. J. Hussain, C. Rea, *J. Chem. Inf. Model.* **2010**, *50* (3), 339–348.
38. Daylight CIS Ltd. Daylight Chemical Information Systems Tutorials http://www.daylight.com/.
39. G. Landrum. http://www.rdkit.org **2020**.
40. B. Tang, S. T. Kramer, M. Fang, Y. Qiu, Z. Wu, D. Xu, *J. Cheminformatics* **2020**, *12* (1), 1–9.
41. M. Withnall, E. Lindelöf, O. Engkvist, H. Chen, *J. Cheminformatics* **2020**, *12* (1), 1–18.
42. J. Klicpera, J. Groß, S. Günnemann, *arXiv.* **2020**.
43. Y. Song, S. Zheng, Z. Niu, Z. H. Fu, Y. Lu, Y. Yang, *IJCAI* **2020**, 2831–2838.
44. K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, A. Palmer, V. Settels, T. Jaakkola, K. Jensen, R. Barzilay, *J. Chem. Inf. Model.* **2019**, *59* (8), 3370–3388.
45. M. Withnall, E. Lindelöf, O. Engkvist, H. Chen, *ICANN* **2019**, *11731 LNCS*, 752–757.
46. B. Samanta, A. De, G. Jana, P. K. Chattaraj, N. Ganguly, M. G. Rodriguez, *JMLR.* **2020**, *21*(1), 4556–4588. .
47. A. Holtzman, J. Buys, L. Du, M. Forbes, Y. Choi, *ICLR.* **2020**.
48. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, et al., *NIPS.* **2019**.
49. Y. Kwon, J. Yoo, Y. S. Choi, W. J. Son, D. Lee, S. Kang, *J. Cheminformatics* **2019**, *11*(1), 1–10.
50. Y. Kwon, D. Lee, Y. S. Choi, K. Shin, S. Kang, *J. Cheminformatics* **2020**, *12* (1), 1–8.
51. D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, et al., *Front. Pharmacol.* **2020**, *11*.
52. J. Zhang, R. Mercado, O. Engkvist, H. Chen, *J. Chem. Inf. Model.* **2021**, *61*(6), 2572–2581.
53. T. Blaschke, O. Engkvist, J. Bajorath, H. Chen, *J. cheminformatics* **2020**, *12*(1), 1–17.
54. A. P. Bento, A. Hersey, E. Félix, G. Landrum, A. Gaulton, F. Atkinson, et al., *J. Cheminformatics* **2020**, *12*(1), 1–16.
55. P. Walters, https://github.com/PatWalters/rd_filters. **2018**.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article. Additional figures and vocabularies mentioned in the manuscript are listed there.

How to cite this article: Z. Gao, X. Wang, B. Blumenfeld Gaines, X. Shi, J. Bi, M. Song, *Molecular Informatics* **2023**, *42*, e202200215. https://doi.org/10.1002/minf.202200215