GNN-based Molecular Graph Autocompletion

Sheng HU†, Ichigaku TAKIGAWA†,††, and Chuan XIAO†††

† ICReDD, Hokkaido University Kita-ku, Sapporo, Japan
†† RIKEN Center for AIP Seika-cho, Kyoto, Japan
††† Graduate School of Information Science and Technology, Osaka University
1-5, Yamadaoka, Suita, Osaka, Japan
E-mail: †{hu.sheng,takigawa}@icredd.hokudai.ac.jp, ††chuanx@ist.osaka-u.ac.jp

Abstract In this work, we present a novel molecular graph generation method by auto-completing a privileged scaffold which represents a core graph substructure. We propose a generative GNN model thus providing the ability to generate unseen molecular graphs outside the given training set. The GNN is designed with an RNN architecture with convolutional layers. We propose an edit-aware graph autocompletion paradigm that adopts the "two adopts and one edit" process to complete the full graphs in two adopt operations and allow one meaningful edit operation to show the user's intention. We also propose a "scaffold trie" for fast training pair generation or changing training models in real-time. Such techniques are helpful for applications such as drug discovery and molecule optimization.
Key words Graph generation, drug discovery

1 Introduction

The ultimate goal of modern drug discovery is to find the target molecules with desired chemical properties, while the potential chemical space of drug-like compounds is $10^{23} - 10^{60}$. Until recent years, such chemical space exploration was traditionally conducted by expert chemists and pharmacologists, along with huge time and monetary cost being devoted.

Visual graph query composition can assist modern drug discovery. Instead of exploratory searching given a subgraph query in graph databases and showing matched graphs, we prefer to use generative model to grow novel molecules on the given subgraph. In the applications of drug discovery, such a subgraph query is usually called a scaffold (i.e., privileged or bioactive scaffold), and performs as a core structure in the molecule to preserve the preferable bioactivity properties. The generated novel molecular graphs are supergraphs of the scaffold thus being guaranteed to contain the scaffold to reveal the chemical properties. Fixing the scaffold usually dramatically reduces the search space of the desired drug thus saving experts' time and cost.

Due to surprising success of deep neural network (DNN) models these days, two categories of representations used in DNN-based models emerge in the drug discovery domain. 1) simplified molecular input line entry system (SMILES) strings representation. Several early works [1, 5, 7, 13, 15] proposed to learn the SMILES grammar using RNN architectures and then generate corresponding SMILES strings from the trained models. Novel molecular graphs can be obtained by simply transforming SMILES strings. These methods have limitations to learning the unrelated grammar and having low chemical validity from generated SMILES. A recent trend is 2) undirected labeled graph representation. It is more natural to learn the original graph structure by using graph neural networks (GNN). This representation can easily achieve higher chemical validity. In this work, we adopt the graph-based representation along with GNN model as our generative model. We adopt the GNN model during the visual graph query composition process to generate completed candidates. A motivating example is shown in Fig 1.

[Example 1] In Fig. 1, a user wants to design a new molecule with a scaffold shown in **user's query**. This user first adopts a suggested candidate to complete the polygon and then erases a vertex to show the label on this vertex is different with his original intention. Then the system sends the input to the learned GNN and returns two candidates **rank1** and **rank2**.

The idea of growing molecules on scaffolds using DNN models did not receive too much attention except the ScaffoldVAE model [12] proposed by Lim *et al.* and DeepScaffold [10] proposed by Li *et al.*. ScaffoldVAE focused on growing side chains from Bemis-Murcko scaffolds [2], which is a special kind of scaffold that only preserves ring systems. DeepScaffold is similar with ScaffoldVAE, but includes all subscaffolds in their dataset to provide the ability of growing a full molecule from a much smaller subscaffold. However, both above methods generate the final molecule in a sin-

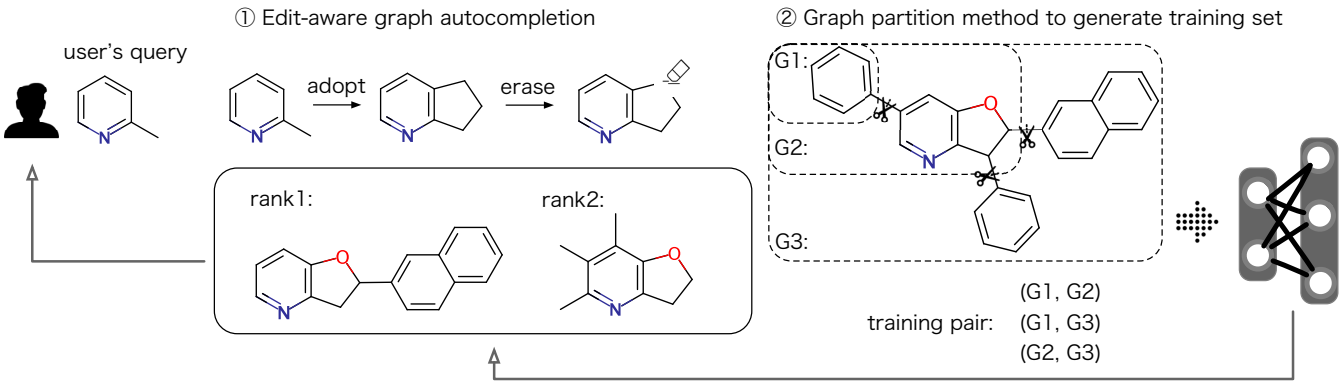


Figure 1 An Example of Generative Graph Autocompletion

gle step without allowing users to edit on the intermediate graph to show their real intentions. This prevents the users, especially for expert chemists, from utilizing their reliable chemical intuitions and experience during the molecule design process. The generated molecules reported in both [12] and [10] also proved to be far from practical molecules used in real experimental chemistry.

In this work, we propose GNNGAC, to allow users to edit the intermediate graph candidates during the molecule design process in multiple steps, utilizing the edit operation to predict the user's real intention to improve effectiveness. We design "two adopts and one edit" process to verify this idea. Also, to make it possible to complete the molecular graph from intermediate graphs of variant sizes in different steps, we design `scaffold-trie` to efficiently train the GNN from the computer memory. A pair-wise Tanimoto similarity-based top-k diversification algorithm is also proposed to enhance the practicability.

To our best knowledge, our work is the first one to focus on generative graph autocompletion. We also introduce a novel interactive way to accelerate the visual graph query composition.

Our main contributions are summarized as follows.

- We develop and combine visual graph composition techniques with graph generation tasks (Section 2 4).
- We propose a "two adopts and one edit" paradigm for generative graph autocompletion. (Section 3).
- We apply a `scaffold-trie` index for effcient training of GNN (Section 3 3).
- We design a top-k diversification algorithm to present the most beneficial suggestions.

2 PRELIMINARIES

In this section, we introduce the basic concepts used in this paper. Then, we provide an overview of the generative graph autocompletion system.

2 1 Scaffold

Scaffolds are generally those subgraphs carrying important characteristics of molecules. The basis scaffolds of a molecule are usually the set of all unique ring systems in the molecule, while a ring system is defined as single/multiple rings sharing an internal bond. The graph representing a molecule itself is called a full molecular graph. Scaffolds of a molecule can be the combinations of graphs appearing in basis scaffolds while such combination should be a subgraph of the full molecular graph, i.e., the largest graph in the scaffolds set is the full molecular graph itself. Scaffolds can be extracted by utilizing general graph mining algorithms [8], but as for molecular applications, we use HierS [17] to obtain the scaffolds to keep in line with [10].

2 2 Scaffold Query

[Definition 1] (Scaffold Query) Given a well-trained generative model $\mathcal{M}$ and a scaffold query q, the candidate graphs generated by $\mathcal{M}$ is a set $\mathcal{M}_q = \{ g \mid q \subseteq g \}$, where $q \subseteq g$ means q is a subgraph of g.

Note that when we refer to Scaffold Query, it both includes the scaffolds extracted from datasets and their corresponding subscaffolds. Also, in this work, we apply different generative models $\mathcal{M}^i$ during different steps of composing incremental queries. The reason is due to the fact pointed out by a recent study [6] that a single GNN might have the shortage on learning short/longest cycle, diameter, or certain motifs that reply on entirely on local neighbor information. Such disadvantages might deteriorate the performance because during different steps we need generate intermediate graphs with variant sizes. Moreover, Definition 1 guarantees that the generated graphs $\{g \mid g \in \mathcal{M}_q\}$ are supergraphs of q.

2 3 Graph Neural Network

We adopt the GNN model used in [10] and [11] for generating our graph candidates. Note that we do not use the single inference process adopted in the original works but will run the inference process multiple times in our multiple

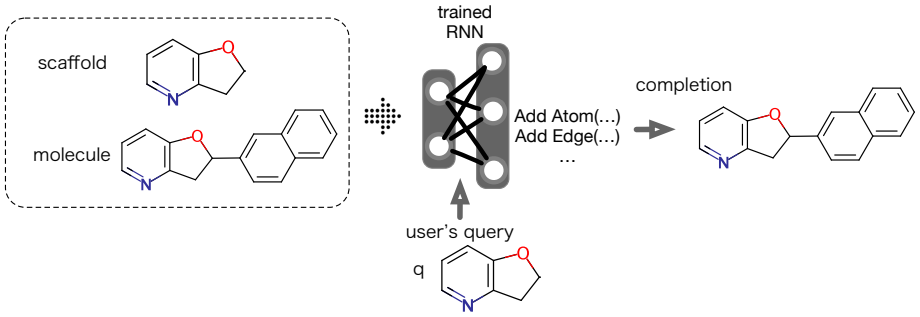


Figure 2 GNN training and generation process

steps autocompletion shown in Section 3.

2 3. 1 Graph-based representation

[Definition 2] (Molecular graph) A molecular graph g is defined as a triple (V_g, E_g, l_g) where V_g is the set of atoms (vertices) and $E_g \subseteq V_g \times V_g$ is the set of bonds (edges). l_g is a labeling function that map labels to atoms (vertices) and bonds (edges). $l_g(u)$ maps an atom label to vertex u while $l_g(u, v)$ maps an bond type label to an edge between u and v.

When fed into the GNN, we use tensors which are multi-dimensional arrays to represent the corresponding atom ids, atom types and bond types.

2 3. 2 Sequential molecular generative process

The purpose of the GNN is to model the conditional distribution, $p(g \mid q)$, where q is the scaffold query and g is the generated graph. Here, we adopt a sequence-like graph knowledge representation used in [11] which builds a full molecular graph in a sequential fashion $(\langle g_0, t_0\rangle, \langle g_1, t_1\rangle, \cdots, \langle g_i, t_i\rangle)$. Such a sequential molecular generative process is essentially a Markov decision processing thus being learned by a recurrent neural network (RNN). In another word, RNN can learn whether to generate a new atom or bond along with its atom/bond type. In [11], three actions are allowed to build a full molecule: (1) add an atom and connect it with an existing atom. (2) connect an existing atom to the new atom. (3) terminate the generating process. We adopt an updated model reported in [10] which includes a 20-layer dense net and leave the hyper parameters unchanged. The details of parameter settings can be found in [10].

When given a scaffold as input, it then represents the scaffold as a sequence $(\langle q_0, t_0\rangle, \langle q_1, t_1\rangle, \cdots, \langle q_i, t_i\rangle)$. The learned RNN will predict the next operation to take according to the current sequence until the termination operation is invoked. [Example 2] In Fig. 2, the scaffold and molecule training pair is sent into the RNN to train the specific graph generation sequences. After training, when a user's query q is sent into the RNN, it is predicted as a sequence of AddAtom(...), AddEdge(...) and built into a completed molecular graph.

2 4 Visual Graph Query Composition Steps

When a user operates on a visual graph composition interface, there are some fundamental operations he/she can make. For example, edge addition is the most fundamental one. However, in autocompletion scenarios, the most efficient step is adopting a suitable graph suggestion to complete the current query which can accelerate the query composition. Also, we consider that it is natural to allow the user to slightly modify the provided suggestion which produces the need of erasing and replacing a part of the graph.

Therefore, without losing the possibility of extending in the future, in this work, we only consider three types of composition $\mathcal{OP} = \{\mathsf{adopt}, \mathsf{erase}, \mathsf{replace}\}$. Particularly, we call erase and replace as **edit** to allow user perform specific operations on modifying the intermediate resulting graphs. [Example 3] In Fig. 1, the composition steps for user's query q is $\mathsf{adopt}(\mathsf{erase}(\mathsf{adopt}(q)))$.

3 Edit-Aware Graph Autocompletion

The **edit** that user has performed is usually implicit indicators of his/her real intention. Such information becomes hints to reveal the user's real desired molecule or a vague direction of design. By adapting our edit-aware paradigm, the GNN model is able to take advantage of users' ideas, especially for those chemical experts who possess drug design experiences.

edit has two types, erase and replace, however, as replace can be taken as a special case of erase, we mainly focus on erase here. Besides **edit** performed by users, we also consider adopt as the context of erase to show more users' intentions. To be aware of the sequence composed by adopt and erase, we design different training methods.

3 1 Adopt Training

The operation adopt means the user accepts a provided graph suggestion q' and incrementally adds a substructure Δq to current query q. To make sure q' has a strong correlation with the current query q, we need to create the training pair (q, q') and insert it into the training set as shown in

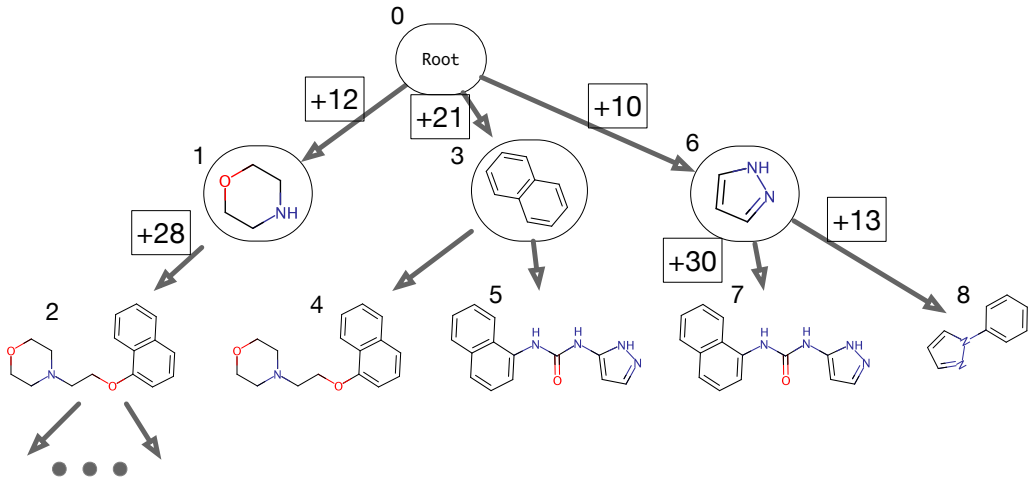


Figure 3 A Scaffold-Trie

Fig. 2. In another word, $q' = q + \Delta q$ and Fig. 2 supposes a simplest case is scaffold $\rightarrow q$ and molecule $\rightarrow q'$. Nevertheless, when a relatively larger molecule requires multiple steps to compose which results in a long composition sequence, more fine-grained training pair enumerations are expected, i.e., q and Δq becomes much smaller and the recursive case $q'' = q' + \Delta q$ must be considered.

For graph structures, such enumerations are impossible because of the combination explosion of subgraphs. Therefore, we choose to store each scaffold in a trie structure according to the super-subgraph relationship in memory and assembly them as training pairs on-the-fly. The advantages of building the `scaffold-trie` is multifold: 1) the trie is memory-resident thus avoiding the I/O overhead which will deteriorate the training time-cost. 2) the granularity of increment size Δq can be controlled flexibly without rebuilding additional indices. 3) a single `scaffold-trie` can generate training pairs for different GNN models' training requirements beginning with different q that used in multi-step graph compositions.

3 2 Increment Size δ

As we might have multiple steps to generate a full molecular graph, we have to set a threshold $\delta = |\Delta q|$ as a granularity constraint of incremental subgraphs that grows from previous intermediate graph q. The value δ can be fixed to a number or can be a variable as a ratio such as 1/2 size of the full molecular graph, which is the case shown in our illustrated scenarios.

3 3 Scaffold-Trie Indexing

The first level nodes in the `scaffold-trie` are the basis scaffolds we extract from the dataset. (See Fig. 3)

3 4 Scaffold-Trie Construction

To build the `scaffold-trie`, we first need to extract all scaffold sets by decomposing each full molecular graph. According to HierS [17], the first scaffold for a molecule is determined by deleting only the chains. Remaining subscaffolds are created by recursively removing a ring system until only basis scaffold remains. All possible ring system combinations are enumerated. After that, we begin to insert the scaffolds as trie nodes according to the supergraph-subgraph relationship. A high level trie node are taken as parent node while their child nodes are supergraphs which contain their parent node as a fragment. As shown on Fig. 3, we also materialize the size of incremental substructure (i.e., Δq) for fast size computation.

Effcient sequence enumeration. There is an efficient way to record the decomposing path by recording the the subscaffolds removing order as a sequence of subscaffold ids. E.g, while the molecule id is 9, and after removing side chains, the remaining subscaffold's id is 5, repeat the recording and the last basis scaffold id is 1, then we reversely output the order as $1 \rightarrow 5 \rightarrow 9$. The time complexity of the index construction is $\mathcal{O}(S)$, where S is the sum of lengths of the decomposing path sequence.

3 5 Traversing Algorithm

Below we show the traversing algorithm for generating a training pair from the dataset in Algorithm 1. Note that Algorithm 1 only shows the training pairs generated for the first adopt operation. For the multi-step generation process that requires multiple adopt operations, simply replace $|parent| < \delta$ with $(|adopt| - 1) \times \delta < |parent| < |adopt| \times \delta$ (also for $|child| < \delta$) will work the same. Moreover, multiple training sets generation only requires for a single full traversal of the `scaffold-trie` thus leading to a time complexity of $\mathcal{O}(|T|)$ where $|T|$ represents the number of nodes in the trie.

The training pair set is then taken as input fed into the GNN model for training purpose.

Algorithm 1: PairGenerator (ST)

Input : ST is a scaffold-trie.
Output : $\{\langle subid, superid\rangle\}$

```
A ← {first level nodes of ST} ;      /* basis scaffolds */
A' ← ∅;
P ← ∅ ;                              /* returned pair set */
while A ≠ ∅ do
    foreach parent in A do
        if |parent| < δ then
            if parent has a child AND |child| − |parent| < δ
              then
                P.insert(⟨parent.id, child.id⟩);
                if parent has a child AND
                  |child| − |parent.ancestor| < δ then
                    P.insert(⟨parent.ancestor.id, child.id⟩);
            if |child| < δ then
                A' ← A' ∪ child;
    A ← A';
return P
```

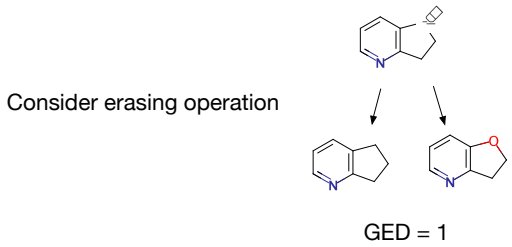


Figure 4 Erase training

3 6 Erase Training

We need to predict which erase edit is tending to happen and generate the corresponding training pair. We generate training pairs that allow some slight errors to happen in the matched common fragment between basis scaffolds. The degree of similarity is measured using Graph Edit Distance (GED). We conduct a similarity join [21] in the scaffold set to find the scaffolds having a GED lower than τ (default set to 1). Then the training is performed by replacing the similar basis scaffolds in scaffolds set with each other.

3 7 Generating Molecules with Multiple Steps

Thanks for the flexibility from `scaffold-trie`, we can efficiently generate different training pairs used for various generative models $\mathcal{M}^i$. Each model $\mathcal{M}^i$ corresponds to one adopt operation of composing incremental queries. Particularly, $\mathcal{M}^e$ that trained in Section 3 6 will be used in all erase operations. If we want to design a process with n adopt operations to complete a full molecular graph, we might train a set of models $\{\mathcal{M}^1, \mathcal{M}^2, \cdots, \mathcal{M}^n\}$ and set $\delta = 1/n$. E.g., the composition steps for user's query q might be like adoptn($\cdots$ adopt2(erase(adopt1(q)))). The number of steps n depends on how large the full molecular graph is and how fine-grained the design process is.

3 8 A Showcase: Two Adopts and One Edit

To verify the effectiveness of multi-step generation, we experimentally set the numebr of steps $n = 2$ and allow one erase operation.

4 Top-k Diversifying Results

After running the inference process of our GNN for multiple times, we can obtain our results set R. When $|R|$ is a large number, displaying all the graphs in R will only mess up the interface with similar results. In this section, we propose a method to only choose diversified graphs out of R to form the final result set R' thus $|R'| < |R|$.

We take advantage of the metric proposed in [17], which is called average pairwise Tanimoto (APT). Here, Tanimoto means Tanimoto coefficient which is computed using molecular fingerprint to describe how similar two molecular graphs are. By adopting APT, we sum up the computed Tanimoto coefficients between each pair of graphs appearing in R, and divide the sum by total number of pairs as shown in Equation 1.

$$APT(R) = \frac{1}{|R|(|R|-1)} \Sigma_{i \neq j}^{|R|} \frac{b_{i\&j}}{b_i + b_j - b_{i\&j}} \qquad (1)$$

where $|R|$ represents the size of result set R, $b_i(b_j)$ is the number of bits set to 1 in result i's(j's) fingerprint, $b_{i\&j}$ means the number of bits set to 1 in the the intersection of i and j's fingerprints.

Finding the most diversified set R' out of R means APT(R') should be larger than any other alternative set R'' with the same set size. This process proved to be a NP-hard problem, which is a reduction from the maximum independent problem shown in [19]. When $|R'| = k$, the time complexity of finding the APT(R') is $\mathcal{O}(k \times |R| \times S_{Tanimoto})$, where $|R|$ represents the unique graphs generated and $S_{Tanimoto}$ means the Tanimoto similarity computation.

In Algorithm 2, we iteratively compute the pair between b_i an b_j $(i \neq j)$(Line 3) to find two most diverse graphs to initialize R'.(Line 4) Then we repeatedly find the proper graph i' that can maximize the APT($R' \cup i'$) and put it into R' until the size of R' reaches k.

5 Experiments

5 1 Experiment Setup

The experiments were carried out on a Linux server with an Intel Xeon Silver 4214 2.20GHz Processor and 192GB RAM, running Ubuntu 18.04.5. The training is performed on two Nvidia Quadro RTX 8000 GPUs. The GNN training algorithms were implemented in PyTorch while `scaffold-trie`

Algorithm 2: Diversify (R)

```
R' ← ∅ ;                                   /* final set R' */
foreach ⟨i, j⟩ ∈ R do
    Compute S_Tanimoto(i, j);
Choose the largest S_Tanimoto(i, j) and R' ← R' ∪ i ∪ j;
for n = 1 ··· k − 2 do
    Find an i' to make the APT(R' ∪ i') the largest;
    R' ← R' ∪ i';
return R';
```

Table 1 Training Statistics

Training set	Triaining time	# of graph1	# of graph2
Adopt1	15 hours	102,273	317,478
Adopt2	16 hours	317,478	739,028
Erase	16 hours	87,106	

is a memory-resident index implemented in C++.

We adopt the default parameters used in the RNN-based architectures [10] with a 20-layer dense net. The growth rate, the number of features in each bottleneck layer is set to 24 and 96. Hyperparameter k and α that represents # of samples generated from importance sampling and the degree of uncertainty in route sampling are set to 5 and 0.5, respectively.

Training statistics are shown in Table 1. # of graph1 and # of graph2 represents the number of unique graphs used in $\langle graph1, graph2\rangle$ pairs where $graph1 \subseteq graph2$. In erase training model, we used 87,106 similar pairs of graphs having GED $\tau = 1$.

For δ, our default setting is to set $\delta = 1/2 \times$ (# of atoms + # of edges).

5 2 Dataset

One dataset is used for the molecule completion task.

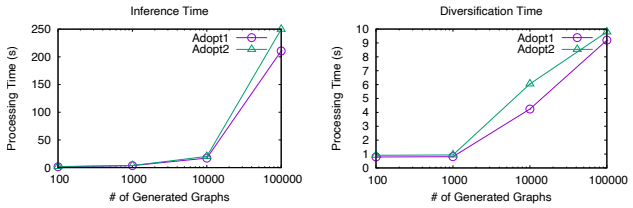
- **ChEMBL** is a molecule dataset containing both structural and bioactivity information for molecules. We remove salt, isotopes and charge neutralization using RDKit. We only kept molecules that containing elements $\{H, C, N, O, F, P, S, Cl, Br, I\}$. To keep in line with [10], we only keep QED larger than 0.5 and used HierS [17] for scaffold extractions. After that, we totally collected 923,786 unique molecules and 348,162 scaffolds.

5 3 Query set

We generated one query set containing 100 basis scaffold graphs defined in Section 2 to keep in line with [20]. Time-cost of inference phase are computed by averaging the sum of the time-cost of the whole query set.

5 4 Time-cost of the Inference Phase

We report the time-cost of the inference phase on average of the whole query set as shown in Fig. 5(a) under the setting of $\delta = 1/2 \times$ |full molecular graph|. An approximate linear growth w.r.t. the # of generated graphs is witnessed for both Adopt1 and Adopt2 models. Erase is not plotted because of the similar performance with Adopt2. Consider the practical scenario, choosing to generate 1,000 graphs for a adopt operation seems ideal as it results in 3.6s and 3.8s for Adopt1 and Adopt2, respectively.



(a) Inference time (b) Diversification time

Figure 5 Time-cost of Adopt1 and Adopt2

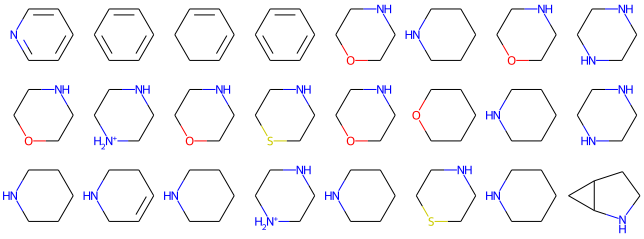


Figure 6 Graph similarity join results with GED = 1

5 5 Time-cost of Diversification

Time-cost of diversification is reported in Fig. 2. Similar with inference time, we observed a nearly linear growth on the diversification time when # of generated graphs is varied. Interestingly, when # of generated graphs equals 100 or 1,000, the time did not grow obviously, this is because a number of duplicate graphs are generated which leads to a smaller difference on diversification time. When # of generated graphs equals 1,000, the time is nearly 1s thus being practical in real-world scenario.

5 6 Time-cost of Traversing Scaffold-Trie

Compared with the inference time and diversification time-cost, traversing `scaffold-trie` seems much more efficient. Constructing the `scaffold-trie` takes less than 10s in our experiment.

Under the default settings, finally, traversing the whole `scaffold-trie` takes nearly 8s in our "two adopts and one edit" system.

5 7 Examples - graph similarity join with GED=1

Fig. 6 shows examples of the resulting pairs after a graph similarity join with GED=1.

5 8 GNNGAC Examples - w/o edit

Fig. 7 shows examples of completed candidates in Adopt1. The left side shows the scaffold query and the right side is the intermediate graphs suggested.

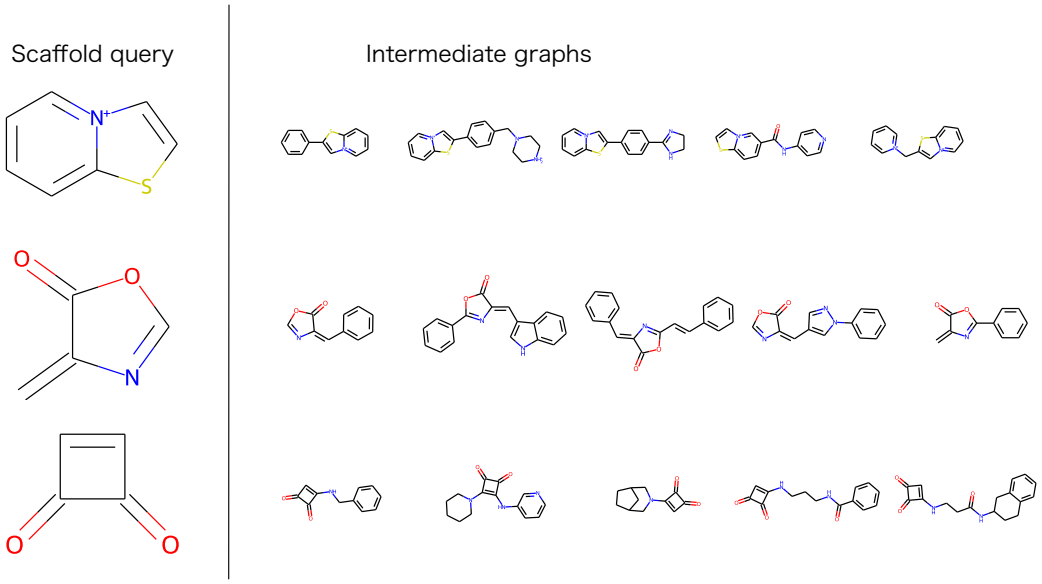


Figure 7 From scaffold query to intermediate graphs

Fig. 8 shows examples of completed candidates in Adopt2. The left side shows the first intermediate graph completed by Adopt1 and the right side is the final completed molecular graphs.

5 9 GNNGAC Examples - with edit

Fig. 9 shows examples that predicted by in Erase. The left side shows intermediate graph after erase operation and the right side is the predicted graphs. Note that the after erase operation is performed, the inference phase can take advantage of the idle time of GUI to send the predicted graphs into inference model in advance for parallel processing. This is inspired by the idea in [4] that shows the latency offered by the GUI and user operations can be utilized to reduce the waiting time.

6 Related Work

Graph Query Autocompletion. The problem of graph query autocompletion (GQAC) originates from the domain of query formulation on graph databases. Bhowmick *et al.* [3] first proposed to construct the visual graph query interface in a data-driven way. This means to first conduct graph mining on graph databases and extract frequent graph features, then construct the icons and toolkit on the interface based on extracted frequent graph features to improve the user experience.

Yi *et al.* [19] proposed the first GAC system, AutoG, to index frequent graph features and how they combine together using a network index. After that, they improved the ranking method of AutoG and proposed GFocus [20], which can utilize focus patterns from users to improve the effectiveness.

Molecular Graph Generation. We refer readers to a survey [18] for a clear taxonomy over different graph-based molecular generative models designed for molecular graph generation tasks.

Scaffold-based Molecular Graph Generation. Modern drug discovery usually starts from specific core structures with high drug-likeness, which are called scaffolds, to ensure preferable bioactivity properties are retained. Existing works have focused on extracting, organizing and visualizing scaffolds such as HierS [17], SCONP [9], ST [14] and SN [16]. However, scaffold-based molecule design did not receive too much attention in the past even though it can obviously incorporate the chemists' intuition and improve the generative performance in the early stage. Lim *et al.* [12] proposed a graph generative model that accepts a molecular scaffold as input and extends it by sequentially adding atoms and bonds. Their generative model consists of a typical VAE structure. The generated molecules are guaranteed to preserve the scaffold with desired chemical properties. DeepScaffold [10] proposed by Li *et al.* came up with a similar idea that growing a full molecule from a scaffold, while this work includes subgraphs of scaffolds in the training set thus being able to generate a molecule from a subgraph of the scaffold. Moreover, DeepScaffold was able to attach atom labels to a simplified cyclic skeleton graph from a molecule that does not contain any atom labels.

Acknowledgment

This work is partly supported by the start-up grant of WPI-ICReDD and JSPS KAKENHI Grant Number JP19K11979.

References

[1] J. Arús-Pous, A. Patronov, E. J. Bjerrum, C. Tyrchan, J.-L. Reymond, H. Chen, and O. Engkvist. Smiles-based deep generative scaffold decorator for de-novo drug design. *Jour-*

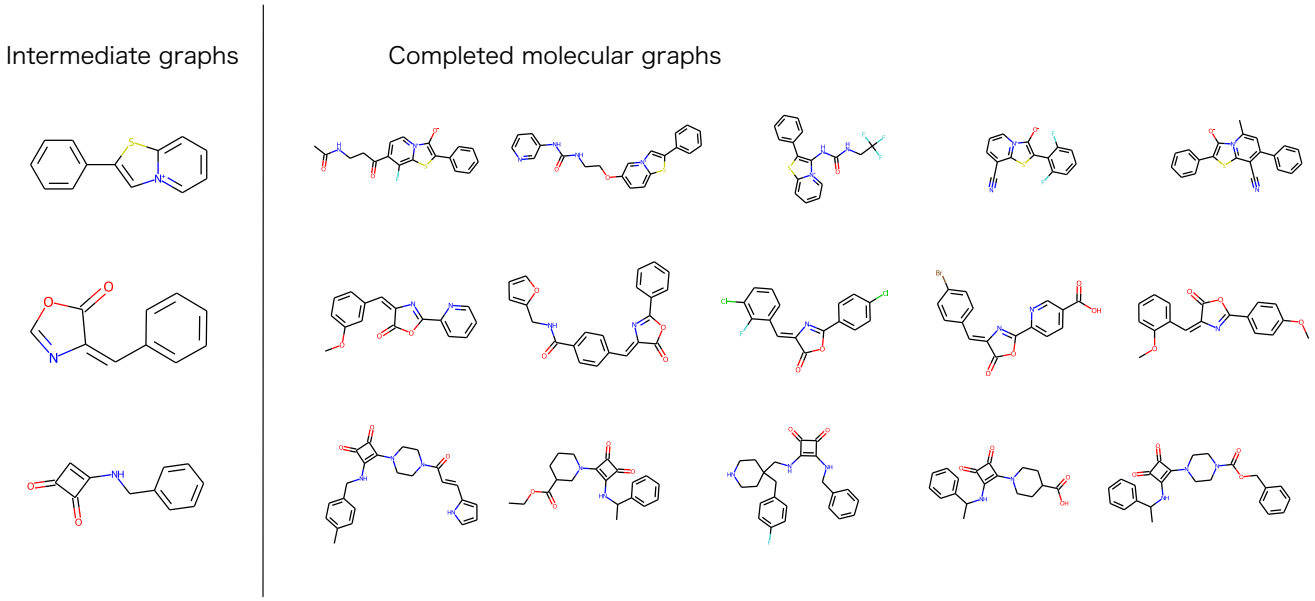


Figure 8 From intermediate graphs to completed molecular graphs

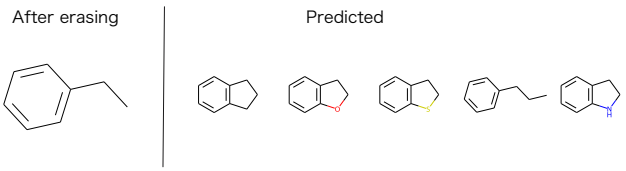


Figure 9 Predicting the erase position and label

nal of Cheminformatics, 12(1):1–18, 2020.

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

[3] S. S. Bhowmick, B. Choi, and C. Dyreson. Data-driven visual graph query interface construction and maintenance: Challenges and opportunities. *PVLDB*, 9(12):984–992, Aug. 2016.

[4] S. S. Bhowmick, B. Choi, and C. Li. Graph querying meets hci: State of the art and future directions. In *ACM SIGMOD 2017*, page 1731–1736, New York, NY, USA, 2017.

[5] H. Dai, Y. Tian, B. Dai, S. Skiena, and L. Song. Syntax-directed variational autoencoder for structured data. *ICLR 2018*, 2018.

[6] V. Garg, S. Jegelka, and T. Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430. PMLR, 2020.

[7] 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, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[8] W. Jin, R. Barzilay, and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation. *ICML 2018*, 2018.

[9] M. A. Koch, A. Schuffenhauer, M. Scheck, S. Wetzel, M. Casaulta, A. Odermatt, P. Ertl, and H. Waldmann. Charting biologically relevant chemical space: a structural classification of natural products (sconp). *Proceedings of the National Academy of Sciences*, 102(48):17272–17277, 2005.

[10] Y. Li, J. Hu, Y. Wang, J. Zhou, L. Zhang, and Z. Liu. Deepscaffold: a comprehensive tool for scaffold-based de novo drug discovery using deep learning. *Journal of Chemical Information and Modeling*, 60(1):77–91, 2019.

[11] Y. Li, L. Zhang, and Z. Liu. Multi-objective de novo drug design with conditional graph generative model. *Journal of cheminformatics*, 10(1):33, 2018.

[12] J. Lim, S.-Y. Hwang, S. Moon, S. Kim, and W. Y. Kim. Scaffold-based molecular design with a graph generative model. *Chemical Science*, 11(4):1153–1164, 2020.

[13] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

[14] A. Schuffenhauer, P. Ertl, S. Roggo, S. Wetzel, M. A. Koch, and H. Waldmann. The scaffold tree- visualization of the scaffold universe by hierarchical scaffold classification. *Journal of chemical information and modeling*, 47(1):47–58, 2007.

[15] M. H. Segler, T. Kogej, C. Tyrchan, and M. P. Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131, 2018.

[16] T. Varin, A. Schuffenhauer, P. Ertl, and S. Renner. Mining for bioactive scaffolds with scaffold networks: improved compound set enrichment from primary screening data. *Journal of chemical information and modeling*, 51(7):1528–1538, 2011.

[17] S. J. Wilkens, J. Janes, and A. I. Su. Hiers: hierarchical scaffold clustering using topological chemical graphs. *Journal of medicinal chemistry*, 48(9):3182–3193, 2005.

[18] X. Xia, J. Hu, Y. Wang, L. Zhang, and Z. Liu. Graph-based generative models for de novo drug design. *Drug Discovery Today: Technologies*, 2020.

[19] P. Yi, B. Choi, S. S. Bhowmick, and J. Xu. Autog: a visual query autocompletion framework for graph databases. *The VLDBJ*, 26(3):347–372, 2017.

[20] P. YI, B. Choi, Z. Zhang, S. S. Bhowmick, and J. Xu. Gfocus: User focus-based graph query autocompletion. *IEEE TKDE*, pages 1–1, 2020.

[21] X. Zhao, C. Xiao, X. Lin, W. Wang, and Y. Ishikawa. Efficient processing of graph similarity queries with edit distance constraints. *The VLDB Journal*, 22(6):727–752, 2013.