DEIM Forum 2023 3a-5-2

User-Interactive Molecular Graph Suggestion for Drug Discovery

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 We present a novel molecular graph generation method by auto-completing a privileged scaffold which represents a core graph substructure step-by-step. We propose a generative GNN model thus providing the ability to generate unseen molecular graphs outside the given training set. An edit-aware graph autocompletion paradigm that follows the "substructure-by-substructure" process is designed to complete the scaffold queries in multiple substructure adopt operations and allow meaningful edit operations to show the user's intention. Such operations enable the involvement of user decisions when interacting with a generative user-centered AI system, which differentiates our work from existing single-run generation paradigms. Particularly, a Monte Carlo tree search (MCTS) method is employed to satisfy the property requirements and navigate the search space when complying with users' edit operations. Moreover, we design a top-k ranking function which considers the preferences on popularity and diversity for different applications, such as query compositions for graph database and drug discovery respectively. Such techniques enable human experts to synergistically interact with the generative models grounded on large data.
Key words Graph generation, graph neural networks, 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, which did not attract much attention compared with the success in the graph database community [3, 4, 10, 13, 25, 29, 30]. Instead of exploratory searching given a subgraph query in graph databases and showing matched graphs [29, 30], we prefer to use generative models 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, 8, 9, 21, 23] are proposed to learn the SMILES grammar using RNN architectures and then generate corresponding SMILES strings from the trained models. These methods have limitations to learning the unrelated grammar and thus have low chemical validity from generated SMILES. A recent trend is (2) undirected labeled graph representation [27]. It is more natural to learn the original graph structure by using graph neural networks (GNNs). This representation can easily achieve higher chemical validity. In this work, we adopt the graph-based representation along with GNN models as our generative models. GNN models are employed during the visual graph query composition process to generate completed candidates. A motivating example is shown in Figure 1.(注1)

[Example 1] In Figure 1, a user wants to design a new molecule with a scaffold shown in **user's query**. This

(注1) While we only show an example for the scenario of drug discovery, our autocompletion system can be simply applied to traditional graph query suggestions for graph databases.

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**, with detailed molecule properties (e.g., MW, logP and QED). The user can interact with **rank1** and **rank2**, such as adopt **rank1** as a new query or even erase a part. The new query will be taken as input and sent into the GNNs for further generations. This process repeats until a proper molecular graph is found.

2 Related Work

Scaffold-based Molecular Graph Generation. The idea of growing molecules on scaffolds using DNN models did not receive too much attention except the ScaffoldVAE model [20] and DeepScaffold [18]. 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 single 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 [20] and [18] also proved to be far from practical molecules used in real experimental chemistry.

In this work, we built a system GNNGAC (**GNN**-based **G**raph **A**uto**c**ompletion), to allow users to edit the intermediate graph candidates during the molecule design process in multiple steps, utilizing the edit operations to understand the user's real intention by adjusting the heuristic functions employed in MCTS. We design an interactive substructure-by-substructure adopt process to verify this idea. This process guarantees the involvement of user decisions to interact with a generative user-centered AI system, which differentiates our work from previous studies that generate graphs in a single run [18, 20]. Compared with generating a bunch of cluttered results, involving user decisions can exploit human capability, i.e., domain knowledge or experiences, to generate much more insightful candidates. Also, to make the utmost of graph training data set for efficient training, we design `scaffold-trie` for data augmentation and efficiently train the GNN from the computer memory. A pairwise Tanimoto similarity-based top-k ranking algorithm is also proposed to enhance the practicability. We demonstrate our system using a real-world molecular dataset containing nearly one million graphs. Users can draw various scaffold queries in our prepared Web-based canvas and check the suggestion quality by themselves.

3 Edit-Aware Graph Autocompletion

Scaffold input. 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 can be extracted by utilizing general graph mining algorithms [16], but as for molecular applications, we use HierS [26] to obtain the scaffolds to keep in line with [18].

［Definition 1］（Scaffold input） Given a well-trained generative model $\mathcal{M}$ and a scaffold input 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.

Definition 1 guarantees that the generated graphs $\{g \mid g \in \mathcal{M}_q\}$ are supergraphs of q.

Autoregressive GNNs. We adopt the GNN model used in [18] and [19] for generating our graph candidates. The entire model architecture is shown in Figure 2. Compared with variational autoencoders (VAEs) or generative adversarial networks (GANs), autoregressive GNNs have unique ability to model edge dependencies, which guarantees the nodes are generated in a sequential way. We adopt a sequence-like graph knowledge representation used in [19] which builds a full molecular graph in a sequential fashion $(\langle g_0, t_0\rangle, \langle g_1, t_1\rangle, \cdots, \langle g_N, t_N\rangle)$, where g_n is a specific graph state (q equals g_0 here) and t_n is an action that transforms $g_n \rightarrow g_{n+1}$. Such a sequential molecular generative process is essentially a Markov decision process thus being modeled as either Markovian or recurrent using a global recurrent neural network (RNN). In other words, the GNNs are used to decide whether to generate a new atom or bond along with its atom/bond type. In [19], three types of actions t_i are allowed to build a full molecule: (1) add an atom and connect it with an existing atom v (with a probability p_v^A), (2) connect an existing atom v to the new atom (with a probability p_v^C), and (3) terminate the generating process (with a probability p^*). According to Figure 2, the MLP layer outputs a tensor with a size $|V| \times (|A|+1) \times |B|$, where V is the atom node set that $v \in V$, A is the atom type set, and B is the bond type set. This tensor is further split into $[\mathbf{p}^A, \mathbf{p}^C] = [tensor^A_{|V|\times|A|\times|B|}, tensor^C_{|V|\times 1\times|B|}]$. On the other side, the MLP* layer outputs a scalar value $p*$ to represent the probability of termination. Eventually, after a softmax computation, $\{\mathbf{p}^A, \mathbf{p}^C, p*\}$ decides which action t_i should be taken when $g_{n+1} \leftarrow t_i(g_n)$.

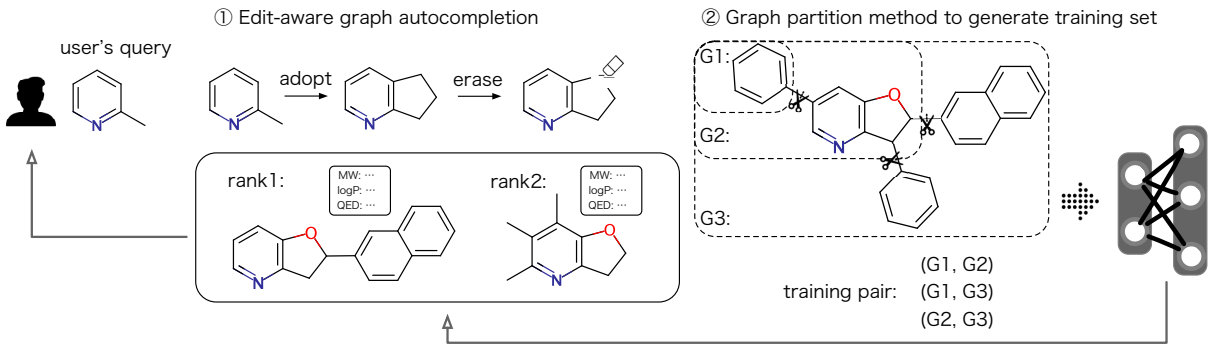

Figure 1 An Example of GNN-based Graph Autocompletion

Query reformulation. The composition process of an intermediate molecular graph can seem like a query reformulation. The user interaction and behavior correlates much with if they are satisfied with the intermediate molecular graphs. For example, if one user adopt the intermediate graph with no hesitation, she probably is very satisfied because the suggestion matched her original intention. Conversely, if she chooses to ignore all the provided suggestions but began to add the incremental part on the current graph, she is unsatisfied and showed implicit negative feedback [31] to all suggestions. If the user first adopt a suggestion but then begins to modify (erase or replace or add) a part of the suggestion, she is also unsatisfied with the provided suggestion. From these interactions, we can learn the user's preference or intent. This is called learning the user's preference online [31]. If we obtain an editing query log, we can model a likelihood function and train such a preference vector of a new user to avoid the cold-start problem.

User operations during query reformulation. While users' operations during the query reformulation of textual query settings have been well studied [6, 7, 15, 17, 24], we found that there are few works discussing the similar problem in the graph query settings. 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. Also, we consider that it is natural to allow the user to slightly modify the provided suggestion which produces the need for erasing and replacing a part of the graph. Finally, the most essential operation is adopting, which completes the current query by a suggestion that accelerates the query composition dramatically [30]. Without losing the possibility of extending in the future, we consider four types of composition operation $\mathcal{OP} = \{\mathsf{add}, \mathsf{adopt}, \mathsf{erase}, \mathsf{replace}, \mathsf{rollback}\}$. Particularly, we call add, erase, and replace as **edit** to allow users to perform specific operations on modifying the intermediate resulting graphs. The **edit** that the user has performed is usually implicit indicators of his/her real intention. After defining the operations set that users can make on a graph query reformulation, we find that the operations can have counterparts in the traditional textual query reformulation operations [6]. We show the mappings in Table 1.

text query reformulation					
Add	Delete	Keep	Transform	Others	First Query
graph query reformulation					
add	erase	adopt	replace	rollback	start graph

Table 1 graph query reformulation operations and the counterparts in text query reformulation research

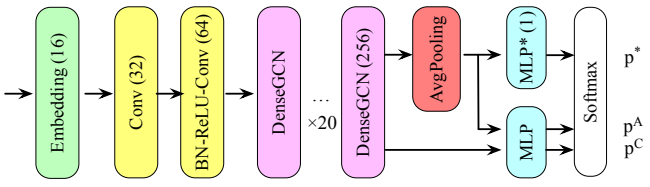

Figure 2 Model Architecture

By adapting our edit-aware paradigm, the MCTS algorithm is able to take advantage of users' ideas, especially for those chemical experts who possess drug design experience.

User preferences and satisfaction. Users' preferences show the search intent in the drug discovery. The users' satisfaction can be reflected in the sequential behavior during the graph query reformulation interactions. There have been some studies focusing on revealing the satisfactions from textual query reformulation interactions [5–7, 11, 12, 17]. Particularly in [7], two main reasons correlating to satisfaction are reported: (1) Users left (clicked or adopted) the query with satisfaction. (2) Being unsatisfied with most results, users were forced to change the query. We can categorize the sequential behaviors into satisfied and unsatisfied class similarly with [17] in Table 2.

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 train-

Case 1	Satisfied	adopt $\rightarrow MCTS$
Case 2	Unsatisfied	add $\rightarrow MCTS$
Case 3	Unsatisfied	adopt $\rightarrow$ erase $\rightarrow MCTS$
Case 4	Unsatisfied	adopt $\rightarrow$ replace $\rightarrow MCTS$
Case 5	Unsatisfied	adopt $\rightarrow$ add $\rightarrow MCTS$

Table 2 Satisfied and Unsatisfied sequential behaviors

ing set. In another word, 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.

Data augmentation. Instead of only using (scaffold $\rightarrow q$, molecule $\rightarrow q'$) pairs for training, we need to conduct data augmentation to enumerate subgraphs to handle cases such as $q'' = q + \Delta q + \cdots + \Delta q'$ For graph structures, such enumerations are impossible because of the combination explosion of subgraphs. Therefore, we decide to only preserve subgraphs with chemical significance (i.e., ring or chain systems). We choose to store each basis scaffold q and their incremental graphs $q'' = q + \Delta q + \cdots + \Delta q'$ 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 for different problem settings such as macromolecules. We set a uniform 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. To efficiently utilize the training set, we set $\delta = 1$ by default, i.e., we put the pair (q, q'') into the training set when $|\Delta q| = |q''| - |q| \geqq 1$.

Moreover, generating the training set 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.

Property optimization and MCTS details. The real-world drug discovery scenario usually places many property requirements for the generated molecular graphs. For example, a higher QED represents better drug-likeness considering the main molecular properties together, which means that the molecular graph is more likely to be synthesized as a useful drug. To improve the effectiveness of produced graphs from the GNN model, we use MCTS to optimize the molecular property such as QED, while considering the user's edit operation at the same time.

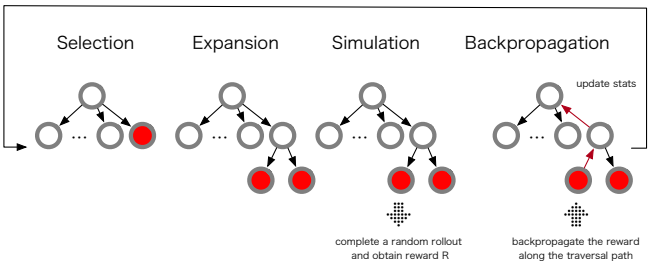

Figure 3 MCTS flow

In MCTS, the root node of the search tree represents one candidate molecular graph from the GNN model. The intermediate node in the search tree is associated with a reward V_i along with the visited times n_i. Additionally, we also attach a flag showing the action types, which will be used to match the user's last edit operation. These statistics will be updated during the MCTS backpropagation. Figure 3 shows the overall flow of all steps.

MCTS usually consists of four steps [14, 22, 28]:

- Selection: select the best node according to a heuristic function.
- Expansion: expand the selected node by autocompleting it with a subgraph building block.
- Simulation: conduct the recursive generation until encountering a leaf node (termination of the molecular graph) and a final reward is gained according to its corresponding QED.
- Backpropagation: backpropagate the reward V on all the visited nodes along the path that has traversed.

Particularly, there are many choices for the subgraph building block. In [22], the work adopted an atom/bond-wise building block which means adding/removing an atom/bond in the Expansion step. We slightly extend the action space in [22] as {Atom addition, Atom removal, Bond addition, Bond removal, Atom replacement, Bond replacement} . We modify the strategy to select the node as:

$$child = \arg\max_i \frac{V_i}{n_i} + c\sqrt{\frac{lnN_i}{n_i}} + edit_{bonus}(edit_{last}) \quad (1)$$

where V_i represents the current value of the node, n_i and N_i are the visit times of node n_i along with its parent N_i, and c is the balance constant of exploration and exploitation. $edit_{bonus}$ will add a particular bonus weight to the child to encourage a more aggressive strategy to select nodes with similar actions as the user's edit operations.

User-interactive MCTS. In our edit-aware paradigm, we design a user-interactive variation of MCTS method to catch

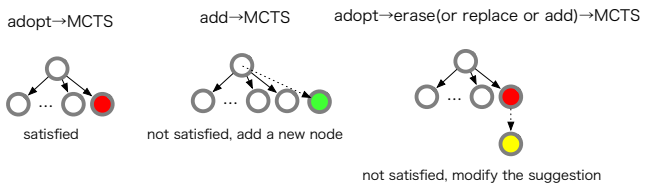

Figure 4 Interactive selection and preferred expansion.

the user's query intents better. Figure 4 shows the differences. There are two strategies in this interactive variation of MCTS.

interactive selection. As shown in Fig. 4, the red node represents the node selected. We allow users to select the arbitrary node despite the score of Equation 1. This helps fix some valuable subgraphs (functional groups) already obtained, which are also being favored by users. When users are not satisfied with all results, they can draw their own scaffold, and then it will be inserted into the search tree as the green node. When users just modify a part of the suggestion, the modified graph will be derived as a new yellow node.

preferred expansion The interaction behavior from the user's edition will be collected through the query logs. Then the expansion step will consider the preferences by learning the preference vectors and return different suggestions according to different users.

Learning the preference vector by satisfactions.

rule set, lhs, rhs, similarity

4 Top-k Diversifying and Ranking Results

4 1 Diversification

After running either GNN inference or MCTS, 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 top-k diversified graphs out of R to form the final result set R' thus $|R'| = k < |R|$.

We take advantage of the metric proposed in [26], 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 2.

$$APT(R) = \frac{1}{|R|(|R|-1)} \sum_{i \neq j}^{|R|} \frac{b_{i\&j}}{b_i + b_j - b_{i\&j}} \qquad (2)$$

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.

Algorithm 1: Top-k (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 util(R' ∪ i') the largest;
    R' ← R' ∪ i';
return R';
```

4 2 Top-k Ranking

Along with diversification, we also need to rank the results according to the substructure popularity to assist the users in easily adding the most possible subgraphs. Here, we adapt a statistic $\mathsf{sel}_\Delta(q')$ used in [30] which represents the number of supergraphs of the increments $(\Delta q')$ in the database (training set) D. This can ensure that novel molecular graphs generated will not have low ranks (which are not contained in the database). Differently, we further extend the $\Delta q'$ as $\Delta q' \cup q_\Delta$, where q_Δ represents the smallest subgraph in q which connects $\Delta q'$ to avoid meaningless increments. Note that $\mathsf{sel}_\Delta(q')$ can be efficiently computed offline using graph indexing techniques proposed in [29].

The final ranking function (util) can be written as below:

$$\mathsf{util}(R') = \frac{w}{k} \sum_{q' \in R'} \mathsf{sel}_\Delta(q') + (1-w) APT(R') \qquad (3)$$

where a weight parameter $w \in [0, 1]$ is added to balance the popularity and diversity scores.

The value of w also has influences on different applications. For example, for traditional graph query suggestions on database search, a larger w is expected to add more frequent subgraph fragments to formulate the query quickly. On the contrary, drug discovery requires a smaller w to improve the diversity for generating novel molecules.

Finding the most optimal set R' out of R means $\mathsf{util}(R')$ 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 set problem shown in [29]. Similar with [29], we use a greedy algorithm (Algorithm 1) to obtain the optimal solution. 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.

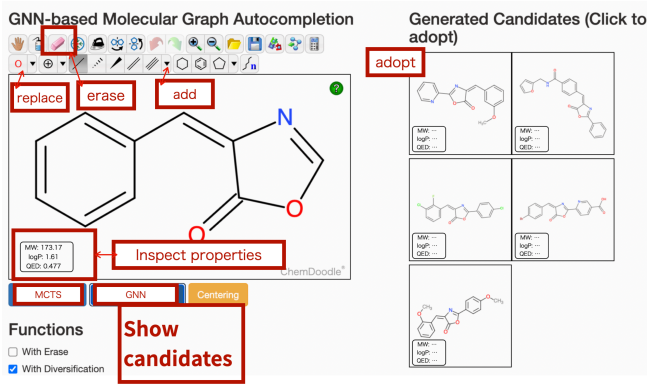

Figure 5 A Screenshot

5 System Architecture

Front end. The front end is a Web-based graphical interface for drawing queries and navigating through graph candidates. The editor is a canvas where users can input the scaffold queries using various canned patterns or drawing edge-by-edge. The user can click on the candidate to adopt the completion and add it to the working canvas.

Back end. Five modules are included in the back end. (1) GNN module, (2) MCTS module, (3) Graph generation module, (4) Top-k ranking module, and (5) Query logging module.

A demo interface. To verify the idea of multi-step graph generations, we prepared a demonstration system. We set up the system as shown in Figure 5. In Figure 5, button MCTS means generating graphs by MCTS, and GNN will trigger generations with a GNN model. After checking the checkbox With Erase, the erase toolkit will collect the erased information to adjust the heuristic functions in MCTS. The operations of add and replace will also change the behavior of MCTS as erase does. In this demo, we aim to show that GNNGAC gives graph suggestions (right panels in Figure 5) in an interactive way. The users can take the reference of the chemical properties (MW, logP, QED) for judgement in drug discovery.

Acknowledgment

This work is partly supported by JSPS KAKENHI Grant Number JP21K17745, JP21K12041, JP20H00323, JP20H00605, JP20H05962, JP17H06099, JP18H04093, JP19K11979 and JST CREST Grant Number JPMJCR18K.

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. *Journal 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] F. Cai and M. de Rijke. Selectively Personalizing Query Auto-Completion. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 993–996, Pisa Italy, July 2016. ACM. 24 citations (Semantic Scholar/DOI) [2022-10-26].

[6] J. Chen, Y. Liu, J. Mao, F. Zhang, T. Sakai, W. Ma, M. Zhang, and S. Ma. Incorporating Query Reformulating Behavior into Web Search Evaluation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 171–180, Virtual Event Queensland Australia, Oct. 2021. ACM. 3 citations (Semantic Scholar/DOI) [2022-11-01].

[7] J. Chen, J. Mao, Y. Liu, F. Zhang, M. Zhang, and S. Ma. Towards a Better Understanding of Query Reformulation Behavior in Web Search. In *Proceedings of the Web Conference 2021*, pages 743–755, Ljubljana Slovenia, Apr. 2021. ACM. 14 citations (Semantic Scholar/DOI) [2022-11-01].

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

[9] 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.

[10] W. Han, J. Lee, M. Pham, and J. X. Yu. igraph: A framework for comparisons of disk-based graph indexing techniques. *Proc. VLDB Endow.*, 3(1):449–459, 2010.

[11] A. Hassan, X. Shi, N. Craswell, and B. Ramsey. Beyond clicks: query reformulation as a predictor of search satisfaction. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, CIKM '13, pages 2019–2028, New York, NY, USA, Oct. 2013. Association for Computing Machinery. 112 citations (Semantic Scholar/DOI) [2022-11-02].

[12] K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *Proceedings of the 20th ACM international conference on Information and knowledge management - CIKM '11*, page 249, Glasgow, Scotland, UK, 2011. ACM Press. 129 citations (Semantic Scholar/DOI) [2022-10-26].

[13] K. Huang, H. Chua, S. S. Bhowmick, B. Choi, and S. Zhou. CATAPULT: data-driven selection of canned patterns for efficient visual graph query formulation. In P. A. Boncz, S. Manegold, A. Ailamaki, A. Deshpande, and T. Kraska, editors, *ACM SIGMOD 2019*, pages 900–917, 2019.

[14] J. 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.

[15] J.-Y. Jiang, Y.-Y. Ke, P.-Y. Chien, and P.-J. Cheng. Learning user reformulation behavior for query auto-completion. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 445–454, Gold Coast Queensland Australia, July 2014. ACM. 97 citations (Semantic Scholar/DOI) [2022-10-26].

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

[17] L. Li, H. Deng, A. Dong, Y. Chang, H. Zha, and R. Baeza-Yates. Analyzing User's Sequential Behavior in Query Auto-

Completion via Markov Processes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 123–132, Santiago Chile, Aug. 2015. ACM. 28 citations (Semantic Scholar/DOI) [2022-10-26].

[18] 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.

[19] 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.

[20] 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.

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

[22] A. A. Rajasekar, K. Raman, and B. Ravindran. Goal directed molecule generation using Monte Carlo Tree Search, Dec. 2020.

[23] 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.

[24] L. Tamine, J. L. Melgarejo, and K. Pinel-Sauvagnat. What Can Task Teach Us About Query Reformulations? In J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins, editors, *Advances in Information Retrieval*, volume 12035, pages 636–650. Springer International Publishing, Cham, 2020. Series Title: Lecture Notes in Computer Science.

[25] C. Wang, M. Xie, S. S. Bhowmick, B. Choi, X. Xiao, and S. Zhou. Ferrari: an efficient framework for visual exploratory subgraph search in graph databases. *The VLDBJ*, pages 1–26, 2020.

[26] 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.

[27] 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.

[28] X. Yang, J. Zhang, K. Yoshizoe, K. Terayama, and K. Tsuda. ChemTS: an efficient python library for *de novo* molecular generation. *Science and Technology of Advanced Materials*, 18(1):972–976, Dec. 2017.

[29] 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.

[30] 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.

[31] A. Zhang, A. Goyal, W. Kong, H. Deng, A. Dong, Y. Chang, C. A. Gunter, and J. Han. adaQAC: Adaptive Query Auto-Completion via Implicit Negative Feedback. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 143–152, Santiago Chile, Aug. 2015. ACM. 43 citations (Semantic Scholar/DOI) [2022-10-26].