Zhu JF, Hao ZK, Liu Q *et al.* Towards exploring large molecular space: An efficient chemical genetic algorithm. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 37(6): 1464–1477 Nov. 2022. DOI 10.1007/s11390-021-0970-3

Towards Exploring Large Molecular Space: An Efficient Chemical Genetic Algorithm

Jian-Fu Zhu (朱健甫), Zhong-Kai Hao (郝中楷), Qi Liu* (刘　淇), *Member*, *CCF*, Yu Yin (阴　钰)
Cheng-Qiang Lu (陆承镪), Zhen-Ya Huang (黄振亚), and
En-Hong Chen (陈恩红), *Fellow*, *CCF*, *Senior Member*, *IEEE*

Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology University of Science and Technology of China, Hefei 230026, China

E-mail: {jeffzhu, hzk171805}@mail.ustc.edu.cn; qiliuql@ustc.edu.cn; {yxonic, lunar}@mail.ustc.edu.cn
{huangzhy, cheneh}@ustc.edu.cn

Received September 13, 2020; accepted April 20, 2021.

Abstract Generating molecules with desired properties is an important task in chemistry and pharmacy. An efficient method may have a positive impact on finding drugs to treat diseases like COVID-19. Data mining and artificial intelligence may be good ways to find an efficient method. Recently, both the generative models based on deep learning and the work based on genetic algorithms have made some progress in generating molecules and optimizing the molecule's properties. However, existing methods need to be improved in efficiency and performance. To solve these problems, we propose a method named the Chemical Genetic Algorithm for Large Molecular Space (CALM). Specifically, CALM employs a scalable and efficient molecular representation called molecular matrix. Then, we design corresponding crossover, mutation, and mask operators inspired by domain knowledge and previous studies. We apply our genetic algorithm to several tasks related to molecular property optimization and constraint molecular optimization. The results of these tasks show that our approach outperforms the other state-of-the-art deep learning and genetic algorithm methods, where the z tests performed on the results of several experiments show that our method is more than 99% likely to be significant. At the same time, based on the experimental results, we point out the insufficiency in the experimental evaluation standard which affects the fair evaluation of previous work.

Keywords data mining, molecular generation, genetic algorithm, drug discovery, artificial intelligence

1 Introduction

Drug discovery is one of the most important tasks in the personalized health and the biochemical industry. Indeed, with the improvement of living standards, people's demand for new drugs will continue to grow [1]. For example, the drug design plays an important role in treating diseases like COVID-19 and improving people's medical level. The first step for the drug design is to generate candidates in the drug-like molecular space.

Historically, a traditional paradigm for molecular generation involves four steps [2]: 1) generating or improving a new material concept; 2) synthesizing the material; 3) incorporating the material into a device or system; 4) measuring the desired property. This cycle repeats and improves future discoveries. However, this paradigm needs much expertise and human labor, which costs several years for each step. Therefore, in order to accelerate this paradigm, automatic molecular generation is necessary.

Recently, with the development of data mining and artificial intelligence, given the application of data mining in other fields [3–8], some researchers have attempted to leverage many machine learning methods for molec-

Regular Paper

This work was partially supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000904, the National Natural Science Foundation of China under Grant Nos. 61922073 and U20A20229, and the Youth Innovation Promotion Association of Chinese Academy of Sciences under Grant No. 2014299.

The code of this paper is public available at https://github.com/bigdata-ustc/calm, Mar. 2021.

*Corresponding Author

©Institute of Computing Technology, Chinese Academy of Sciences 2022

ular generation and have made some progress [9–13]. However, due to the complex structure and the large molecular space, the above work did not involve the issue of large molecules. As a matter of fact, there are still many technical and domain challenges for molecular generation. We can summarize the following points. Firstly, some possible molecules in a huge molecular space may contain many atoms and complex structures (e.g., the mol1 in Fig.1 has the higher property score), which should also conform the prior chemistry knowledge. As the complexity increases, the process of molecular generation becomes harder. Secondly, the whole drug-like molecular space contains 10^{23}–10^{60} molecules [14]. Therefore, it is important to generate molecules that can span the whole space. Additionally, this space is not smoothing, where a little change in the molecular structure may result in many differences in the property (e.g., mol2 and mol3 in Fig.1 are similar, while their property scores are much different). Thirdly, the generated molecules should not be affected by existing molecules in the dataset. There may be more valuable molecules in the unknown space. Hence, the model should be able to explore the molecular space outside the dataset.

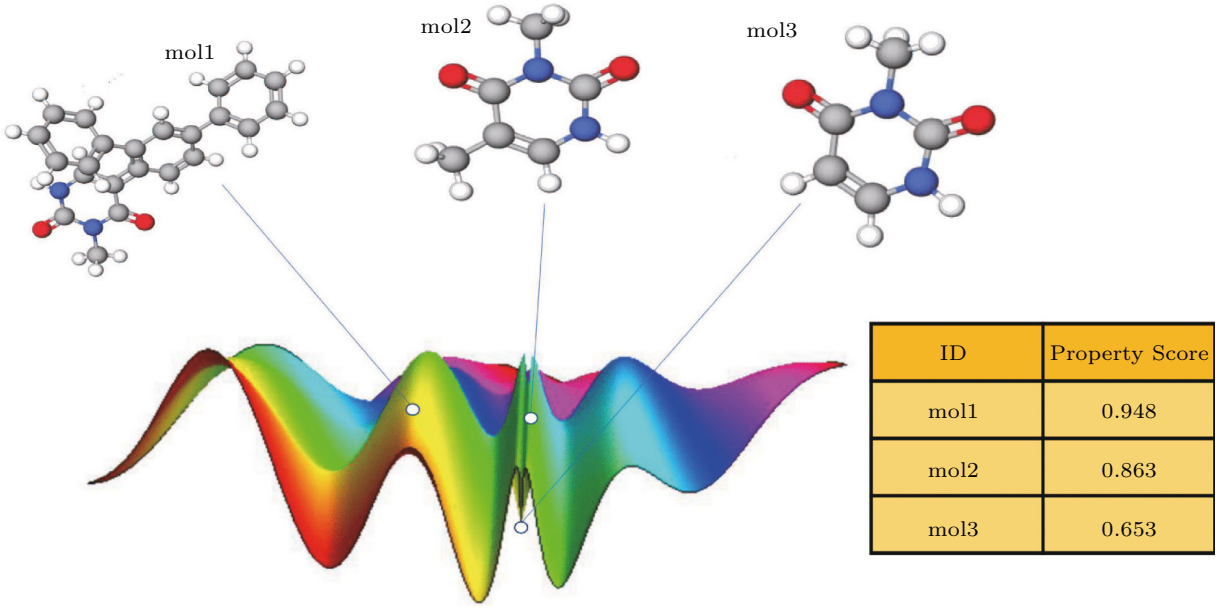
To address the inherent challenges and difficulties mentioned above, in this paper, we propose a chemical genetic algorithm for large molecular space (CALM) for molecular generation. Firstly, we propose a molecular representation named molecular matrix, which is scalable for the molecular size. Furthermore, in order to make the generated molecules conform the laws of chemistry, we design the chemical constraint mask which guarantees the validity of generated molecules even in complex environments. These components pave the way for exploring the large molecular space. Secondly, to deal with the large and non-smoothing molecular space, we design the crossover and the mutation operators respectively. The former can generate offspring molecules that could span the large molecular space. The latter can make a little change to the current molecule, which is suitable for local molecular space exploration. Thirdly, we combine the above components into a genetic algorithm framework and verify that CALM can generate many differences from the existing dataset. The extensive experiments prove that CALM can deal with large molecules and outperform other baselines in molecular generation tasks.

2 Related Work

The related work can be divided into four classes: traditional work, graph generation, deep learning models for molecule generation, and the work based on the genetic algorithm.

2.1 Traditional Work

The number of drug-like molecules is estimated to be very large [14]. Traditional chemical methods, like high throughput screening (HTS) [15] and high throughput virtual screening (HTVS) [16], narrow the feasible



ID	Property Score
mol1	0.948
mol2	0.863
mol3	0.653

Fig.1. Complex molecular space.

space. These methods have made larger parts of the chemical space accessible to the computational and experimental study. However, these new methods still waste supercomputer and laboratory resources. Therefore, minimizing the number of bad leads generated at the start of the pipeline remains a key priority. In Subsections 2.2–2.4, we will introduce the emerging work based on data mining and artificial intelligence. First, we consider the work about graph generation based on GANs [17] and VAEs [18]. Second, we discuss the deep generative work which is deeply influenced by graph generation. Third, we introduce the work based on genetic algorithms.

2.2 Graph Generation

In somewhat, molecular generation can be viewed as a special case of graph generation. Therefore, it is necessary to introduce the concept of graph generation. The non-Euclid data, i.e., graph data, has its own characteristics. With the great progress in image and text fields made by GANs and VAEs, some work starts exploring the graph generative network, such as [19–21]. However, the above work is limited from learning the single graph or small molecules. In order to adapt to a more complex environment, some researchers provide some other technical perspectives. For example, based on RNN fashion, GraphRNN [22] and GRAN [23] generate a graph serially, where the information of nodes and edges in a graph affects the output of the RNN. The above studies achieve obviously great performance for some complex datasets.

Overall, the graph generation has great prospects, which has emerged as a new method for some applications, such as recommendation, privacy protection and NP-complete problem [24]. Of course, molecular generation is one of them.

2.3 Deep Learning Models for Molecule Generation

This field can be traced back to the molecular autoencoder "CVAE" [25] that employs the SMILES string [26] as data representation and first addresses the generative model for the molecular generation. Since then, other deep learning based work has been greatly improved on this basis. NeVAE [27] first achieves 100% validity of generated molecules through adding a mask to the process of decoding. Junction Tree Variational Autoencoder (JT-VAE [28]) builds a hierarchical process to capture the molecular information by using the tree and graph representation. The constrained graph convolutional policy network [12] provides a novel perspective on molecular generation, which views the generation process as a Markov decision process and combines the graph operation and reinforcement learning [29] perfectly. ALL-SMILES [30] encodes multiple SMILES strings of a single molecule using a set of stacked recurrent neural networks, achieving a promising result. However, deep learning methods tend to learn the distribution underlying the given dataset, which could ignore the valuable molecules outside the dataset.

2.4 Genetic Algorithm for Molecular Design

As for ChemGE [31], it uses the SMILES string as the discrete code of a molecule, which is based on an evolutionary algorithm. Though it does provide a new perspective for molecular generation and achieves relatively good performance compared with some deep learning work, its way of encoding, SMILES string, and corresponding operations for SMILES string limit the power of the genetic algorithm. Then GB-GA [32] emerges as an efficient method to explore the molecular space based on prior chemical knowledge. But molecules generated by GB-GA tend to match a template, which means that GB-GA ignores much molecular space. Finally, GA+D [33] augments the genetic algorithm with the deep neural network, which achieves a competitive performance.

3 Proposed Method

In this section, we first give a formal definition of the problem. Second, we introduce the framework of a basic genetic algorithm [34]. Third, detailed descriptions for the components in CALM as shown in Fig.2 are provided. Finally, we present the full CALM algorithm incorporating each component.

3.1 Problem Definition

We consider that each molecule can be represented by an undirected graph $\mathcal{M}$ with a set of edges $\mathcal{E}$ and nodes $\mathcal{V}$. In the setting of the molecular space, each atom corresponds to a node $v_i \in \mathcal{V}$ associated with an integer indicating the atom type. We further represent each chemical bond as an edge $(v_i, v_j) \in \mathcal{E}$ associated with a bond type $y_{i,j} \in \{1, 2, ..., Y\}$. We also view the whole molecular space as a set $\mathcal{S}$ containing all possible

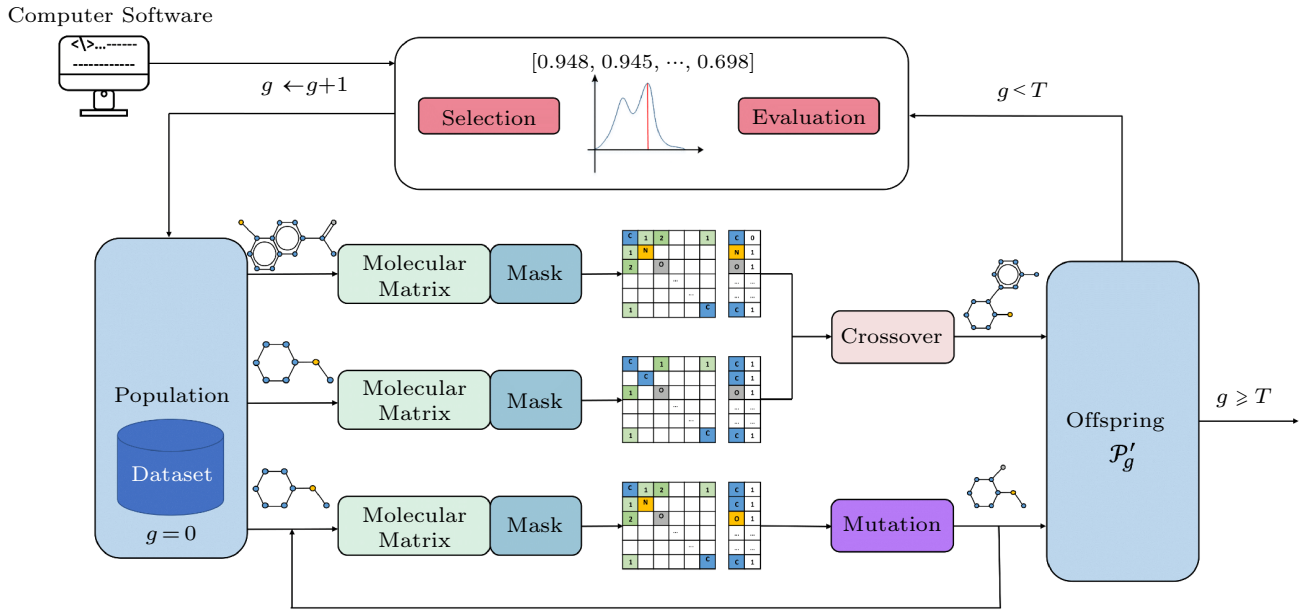


Fig.2. Framework of CALM. C: carbon; N: nitrogen; O: oxygen.

molecules. And there are lots of quantitative properties for molecules, such as penalized logP (partition coefficient), QED (quantitative estimate of druglikeness), which can be denoted as a set $\mathcal{P}$. We view an arbitrary property $p \in \mathcal{P}$ as a function which maps a molecule $\mathcal{M} \in \mathcal{S}$ to a real number, where $p : \mathcal{S} \to \mathbb{R}$. Given a function p, we want to find molecules with the highest value of function p. Thus, our target is to find a set of molecules $\mathcal{S}_p = \{\mathcal{M}^1, \mathcal{M}^2, ...\mathcal{M}^{N_\mathrm{p}}\}$, where $\mathcal{S}_p \subset \mathcal{S}$, with the highest value of property p.

We adopt a genetic algorithm framework for this problem and propose CALM.

3.2 Overall Framework

The core component of a genetic algorithm is the population $\mathcal{P}$ which can be viewed as a set containing a certain number N_p of solutions to a specific problem, where each solution is encoded into a solution representation called gene. A typical procedure of a genetic algorithm is to modify the population iteratively for many generations. The population containing N_p solutions can be defined as follows:

$$\mathcal{P}_g = \{\mathcal{X}^{1,g}, \mathcal{X}^{2,g}, ..., \mathcal{X}^{N_\mathrm{p},g}\},$$

where $\mathcal{X}^{i,g}$ represents the i-th solution in population for generation g and g is a natural number.

The basic genetic algorithm runs three steps to update the population iteratively until we are satisfied with the final population. First, at generation g, we generate new solutions to explore the solution space. Generally, there are two ways to produce new solutions: crossover and mutation. For crossover, a genetic algorithm randomly selects a pair of genes from $\mathcal{P}_g$ and combines them to produce an offspring solution. For mutation, a genetic algorithm modifies one of genes in $\mathcal{P}_g$ to generate an offspring solution. Second, after crossover and mutation, the algorithm will get a certain number N_new of new solutions. Now, there is a set of solutions $\mathcal{P}_g{}'$ containing all new solutions and the remaining solutions in the population $\mathcal{P}_g$. We assign an adaptive value to each solution through a fitness function in the population $\mathcal{P}_g$ to estimate the quality of solutions. Third, a genetic algorithm will select N_p solutions in $\mathcal{P}_g{}'$ to form the population $\mathcal{P}_{g+1}$.

To adopt the genetic algorithm framework for our problem, we resolve these three main concerns. First, we design a new encoding method called molecular matrix specifically for molecules, which preserves the whole information of the molecular structure. Second, we propose matrix crossover and matrix mutation for producing offspring molecules based on molecular matrix. Third, we design a chemical constraint mask that can guarantee the validity of generated molecules.

3.3 Components of CALM

In this subsection, we will introduce the following points. First, we give descriptions for a new molecular representation called molecular matrix and its corre-

sponding mask named chemical constraint mask. Second, we introduce population initialization. Third, we present the crossover and mutation operations for offspring generation. Finally, we describe evaluation and selection in CALM.

3.3.1 Molecular Representation

Traditionally, the molecules are represented as SMILES strings, which is widely used in previous work [25, 31, 32]. However, the relationship between atoms and chemical bonds is not described in detail in a SMILES string. Therefore, invalid molecules are often generated after crossover and mutation operators. What is more, small changes in a molecule may correspond to many differences in its corresponding SMILES string, which can lead to difficulties in molecule generation.

The proposed solution of encoding molecules can be described as follows, which is very similar to adjacent matrix. Given a specified molecule $\mathcal{M}$ with N atoms, we can produce the molecular matrix $\boldsymbol{A}$ as follows. We assign f_i to diagonal element A_{ii}, which indicates the type of the atom v_i. And $y_{i,j}$ is assigned to A_{ij}, where $i \neq j$, which indicates the type of the chemical bond. We build an $N \times N$ matrix $\boldsymbol{A}$ with zero initializer, then for each element A_{ij} in matrix $\boldsymbol{A}$:

$$A_{ij} = \begin{cases} f_i, & \text{if } i = j, \\ y_{i,j}, & \text{otherwise.} \end{cases}$$

For example, as shown in Fig.3, the type of atom v_1 is nitrogen, where the value of A_{11} is 7, which is consistent with the position in the periodic table of elements. By analogy, the value of A_{22} is 6, which indicates the type of atom v_2 is carbon.

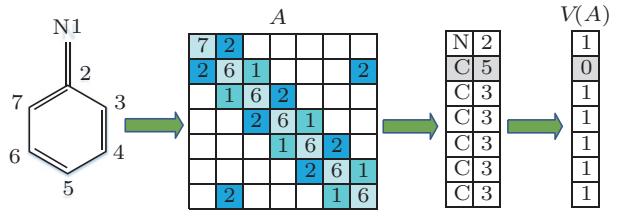


Fig.3. Molecular matrix and chemical constraint mask.

3.3.2 Chemical Constraint Mask

In order to avoid the invalid generated molecules and deal with complex molecular environments, inspired by previous work [35], we design the chemical constraint mask to make the molecules conform the law of prior chemistry knowledge. Given a specified molecular matrix $\boldsymbol{A}$, we can get an indicator vector where the i-th component of the vector corresponds to atom v_i and the value of element indicates whether atom v_i can connect to any other atoms. The i-th component of this vector can be defined as follows:

$$V_i(\boldsymbol{A}) = I(h(A_{ii}) - \sum_{j=1}^{n} A_{ij}), \text{ if } i \neq j.$$

$I(x)$ is an indicator function, which is defined as follows:

$$I(x) = \begin{cases} 0, & \text{if } x \leqslant 0, \\ 1, & \text{otherwise.} \end{cases}$$

Function h determines the maximum valence given a specified atom type. The sum of the i-th row except the diagonal element indicates the total valence of atom v_i. The difference between these two terms indicates whether atom v_i is saturated or not. For example, as shown in Fig.3, A_{22} is 6, which indicates that the type of atom v_2 is carbon. Based on the prior knowledge of chemistry, the maximum valence of carbon is 4. However, the sum of the remaining elements in row 2 indicates atom v_2 has five valences. Therefore, the value of $V_2(\mathcal{M})$ is 0, which indicates that atom v_2 is saturated. By analogy, atom v_1 is unsaturated, which indicates that atom v_1 can connect to the other atoms.

3.3.3 Population Initialization

With molecular matrix representation and chemical constraint mask, we can now present a full description of CALM. To start with, CALM should initialize the population $\mathcal{P}_0$ with randomly selected molecules from a specified dataset, where the dataset provides some information of the chemical space. A dataset can be viewed as a subset of the whole molecular space $\mathcal{S}$, which is denoted as $\mathcal{S}_{\text{sub}}$. Then this process can be formulated as:

$$\mathcal{P}_0 \leftarrow \{\mathcal{M}^{1,0}, \mathcal{M}^{2,0}, ..., \mathcal{M}^{N_{\text{p}},0}\},$$

where $\mathcal{M}^{1,0}, \mathcal{M}^{2,0}, ..., \mathcal{M}^{N_{\text{p}},0}$ are all randomly selected from $\mathcal{S}_{\text{sub}}$.

3.3.4 Matrix Crossover

Matrix crossover is an operator that creates new molecules to explore the chemical space. It receives a pair of molecules from the current population and outputs a new molecule. There are two suboperators of matrix crossover called substructure operator and combination operator.

We first introduce the substructure operator denoted as O_{bfs}, which receives a molecule $\mathcal{M}$ with N atoms as input and outputs a substructure matrix. We

should first determine the width w which decides the size of the substructure, where w is sampled from an arbitrary distribution related to N. Then we randomly select an atom in the molecule as the starting node. We take the Breadth-First Search (BFS) algorithm from the starting node to select w atoms in the molecule. Finally, we output the substructure in the molecule which is represented as a molecular matrix $\boldsymbol{A}'$. This process is described in Algorithm 1.

Algorithm 1. Substructure Operator O_{bfs}

Input: a specified molecular matrix $\boldsymbol{A}$ with N atoms
Output: a substructure matrix of molecule which is denoted as $\boldsymbol{A}'$

Randomly select an atom v_s as the starting atom
Width w sampled from an arbitrary distribution related to N
Initialize a queue Q and a list L
$Q.push(s)$, $L.append(s)$
for $i = 1 \to w$:
  $x \leftarrow Q.pop()$, $L.append(x)$
  for $j = 1 \to N$:
    if $A_{xj} \neq 0$ **and** $j \notin L$:
      $Q.push(j)$
    end
  end
end
Initialize a $w \times w$ matrix $\boldsymbol{A}'$ with a zero initializer
for $i = 1 \to w$:
  for $j = 1 \to w$:
    ${A_{ij}}' \leftarrow A_{L[i]L[j]}$
  end
end
return $\boldsymbol{A}'$

Then we introduce the combination operator denoted as O_{com}. At generation g, CALM first randomly selects a pair of parent molecules from $\mathcal{P}_g$. We can get two substructures by using O_{bfs}. Then combination operator O_{com} combines these two substructures into a new offspring molecular matrix. Additionally, we should keep the connectivity of these two substructures. The connecting process is performed by establishing a single bond between two atoms, where each atom is randomly selected from a different substructure. However, the connecting process can be invalid if one of the randomly selected atoms is saturated for valence. Therefore, we apply the chemical constraint mask to select the unsaturated atoms to avoid the invalid generated molecules. The detail of the combination operator is described in Algorithm 2.

Given two molecular matrices $\boldsymbol{A}^1, \boldsymbol{A}^2$, the matrix crossover can be defined as follows:

$$\boldsymbol{A}^1_{\text{sub}}, \boldsymbol{A}^2_{\text{sub}} \leftarrow O_{\text{bfs}}(\boldsymbol{A}^1), O_{\text{bfs}}(\boldsymbol{A}^2),$$

$$\boldsymbol{A}_{\text{new}} \leftarrow O_{\text{com}}(\boldsymbol{A}^1_{\text{sub}}, \boldsymbol{A}^2_{\text{sub}}).$$

Algorithm 2. Combining Two Substructures O_{com}

Input: two substructures $\boldsymbol{A}^1$, $\boldsymbol{A}^2$ with N^1, N^2 atoms respectively
Output: a molecular matrix $\boldsymbol{A}'$ corresponding to a new molecule

Determine m, n, where $V_m(\boldsymbol{A}^1) = 1$, $V_n(\boldsymbol{A}^2) = 1$
Build an $(N^1 + N^2) \times (N^1 + N^2)$ matrix $\boldsymbol{A}'$ with zero initializer
for $i = 1 \to N^1$:
  for $j = 1 \to N^1$:
    ${A'}_{ij} \leftarrow A^1_{ij}$
  end
end
for $i = 1 \to N^2$:
  for $j = 1 \to N^2$:
    ${A'}_{i+N^1,j+N^1} \leftarrow A^2_{ij}$
  end
end
${A'}_{m,n+N^1} \leftarrow 1$
${A'}_{n+N^1,m} \leftarrow 1$
return $\boldsymbol{A}'$

For convenience, we denote matrix crossover as O_{cro}. However, in some cases, drastically structural differences are not sufficient. We should explore the local space of a molecule.

3.3.5 *Matrix Mutation*

In order to change the molecular structure directly, we design the matrix mutation. We first introduce the intuition underlying this operation. As shown in Fig.4, the structure of the two molecules is almost the same, while the green dotted line identifies the different atoms. However, the QED values of two molecules are much different. Following the observation mentioned above, matrix mutation should make a little change in the molecular structure for more effective exploration. In this article, we design a responsive operation to deal with this situation. There are three suboperators combined to form the matrix mutation, which are the add-atom operator, add-bond operator, and break-bond operator respectively. A limited combination of these operations can cover most of the local molecular space.

Given a specified molecular matrix $\boldsymbol{A}$ with N atoms, for the add-atom operator, we prepare an atom v_{N+1}. Then we randomly select an unsaturated atom v_i in molecule $\mathcal{M}$, where we establish a single atomic bond between atom v_{N+1} and atom v_i. For the change-bond and break-bond operators, we randomly select an non-zero element A_{ij} in molecular matrix $\boldsymbol{A}$. Then we change the value of A_{ij} and assign 0 to A_{ij} for the change-bond and the break-bond operator respectively.

More detailed description is in Algorithm 3. We denote matrix mutation as O_{mut}.

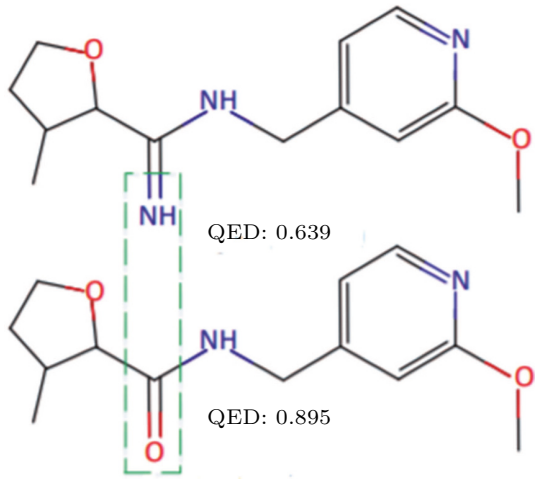


Fig.4. Motivation for mutation operator.

Algorithm 3. Matrix Mutation O_{mut}

Input: a specified molecular matrix $\boldsymbol{A}$ with N atoms
Output: a new molecular matrix $\boldsymbol{A}'$

$\boldsymbol{A}'$ = null
operator randomly selected from add-atom, change-bond, break-bond operator
if *operator* == add-atom operator:
  Initialize $\boldsymbol{A}'$ with an $(N+1)\times(N+1)$ matrix filled with zero
  Determine m, where $\boldsymbol{V}_m(\boldsymbol{A})=1$
  for $i=1\rightarrow N$
    for $j=1\rightarrow N$
      $A'_{ij}\leftarrow A_{ij}$
    end
  end
  $A'_{N+1,m}\leftarrow 1$, $A'_{m,N+1}\leftarrow 1$
else:
  $\boldsymbol{A}'\leftarrow\boldsymbol{A}$
  Determine i,j at random, where $A'_{ij}\neq 0$
  if *operator* == change-bond operator:
    $A'_{ij}\leftarrow x$, $A'_{ji}\leftarrow x$, where $x\in\{1,2,...,Y\}$
  else if *operator* == break-bond operator:
    $A'_{ij}\leftarrow 0$, $A'_{ji}\leftarrow 0$
  end
end
return $\boldsymbol{A}'$

3.3.6 Evaluation and Selection

For generation g, we have the population $\mathcal{P}_g$. After matrix crossover and matrix mutation, CALM generates a certain number N_{new} of offspring molecules, which can be denoted as a set $\mathcal{Q}_g$. Now there is a set $P_g{}' = \mathcal{P}_g \cup \mathcal{Q}_g$ containing all new molecules and remaining molecules in $\mathcal{P}_g$. Therefore, $\mathcal{P}_g{}'$ has $N_{\text{p}}+N_{\text{new}}$ molecules.

To form the next generation, there are two main stages called evaluation and selection. First, the evaluation stage assigns an adaptive value to each molecule in $\mathcal{P}_g{}'$ through a fitness function provided by an external software in this case. Second, the genetic algorithm will select some molecules in $\mathcal{P}_g{}'$ with higher fitness values to form the next generation. The selection scheme in CALM is performed by ranking the fitness values from large to small and eliminating N_{new} molecules with lower fitness values to form the population $\mathcal{P}_{g+1}$. If we view the above process as a function O_{s}, we can get as follows:

$$\mathcal{P}_{g+1}\leftarrow O_{\text{s}}(\mathcal{P}_g{}').$$

3.3.7 Overall Algorithm

At last, we combine the above components together to introduce the complete CALM algorithm in Algorithm 4.

Algorithm 4. Framework of CALM Algorithm

Input: a specified dataset $\mathcal{S}_{\text{sub}}$, maximum iteration limit T, the number of population size N_{p}, the number of layers for matrix mutation N_l and the number of new generated molecules N_{new}
Output: population of solutions $\mathcal{P}_T$

Generation $g=0$
Initialize the population $\mathcal{P}_0$ as follows: $\mathcal{P}_0=(\mathcal{M}^{1,0},\mathcal{M}^{2,0},...,\mathcal{M}^{N_{\text{p}},0})$, where $\mathcal{M}^{1,0},\mathcal{M}^{2,0},...,\mathcal{M}^{N_{\text{p}},0}$ are randomly selected from S_{sub}
while $g<T$ **do**
  $\mathcal{Q}_g=\{\}$
  for $k=1\rightarrow N_{\text{new}}$
    Randomly select *operator* from $O_{\text{cro}}, O_{\text{mut}}$
    if *operator* == O_{cro}:
      $\mathcal{M}^{i,g},\mathcal{M}^{j,g}$ randomly selected from $\mathcal{P}_g$
      $\boldsymbol{A}^{i,g}\leftarrow\mathcal{M}^{i,g}$, $\boldsymbol{A}^{j,g}\leftarrow\mathcal{M}^{j,g}$
      $\boldsymbol{A}^{i,g}_{sub}\leftarrow O_{\text{bfs}}(\boldsymbol{A}^{i,g})$ (Algorithm 1)
      $\boldsymbol{A}^{j,g}_{sub}\leftarrow O_{\text{bfs}}(\boldsymbol{A}^{j,g})$ (Algorithm 1)
      $\boldsymbol{A}_{\text{new}}\leftarrow O_{\text{com}}(\boldsymbol{A}^{i,g},\boldsymbol{A}^{i,g})$ (Algorithm 2)
      $\mathcal{M}_{\text{new}}\leftarrow\boldsymbol{A}_{\text{new}}$
      $\mathcal{Q}_g\leftarrow\mathcal{Q}_g\cup\{\mathcal{M}_{\text{new}}\}$
    else:
      $\mathcal{M}^{i,g}$ randomly selected from $\mathcal{P}_g$
      $\mathcal{M}_{\text{temp}}\leftarrow\mathcal{M}^{i,g}$
      for $l=1\rightarrow N_l$:
        $\boldsymbol{A}_{\text{temp}}\leftarrow\mathcal{M}_{\text{temp}}$
        $\boldsymbol{A}_{\text{new}}\leftarrow O_{\text{mut}}(\boldsymbol{A}_{\text{temp}})$ (Algorithm 3)
        $\mathcal{M}_{\text{temp}}\leftarrow\boldsymbol{A}_{\text{new}}$
        $\mathcal{Q}_g\leftarrow\mathcal{Q}_g\cup\{\mathcal{M}_{\text{temp}}\}$
      end
    end
  end
  $\mathcal{P}_g{}'\leftarrow\mathcal{P}_g\cup\mathcal{Q}_g$
  $\mathcal{P}_{g+1}\leftarrow O_s(\mathcal{P}_g{}')$ (Evaluation and selection)
  $g\leftarrow g+1$
end
return population $\mathcal{P}_T$

4 Experiments

In this section, we give a description for experimental setup and conduct extensive experiments to verify the performance of CALM.

4.1 Experimental Setup

In this subsection, we concern the dataset, parameters setup and baselines used in our experiments.

4.1.1 Dataset

The dataset used in our experiments is the publicly available ZINC dataset [36] which is also widely used in previous work. The ZINC dataset contains 250 000 drug-like commercially available molecules. On average, these molecules contain an average of 23 heavy atoms. We randomly sample molecules from this dataset to initialize the population.

4.1.2 Parameters Setup

The settings of the algorithm mainly have the following four key points. First, for population initialization, we set N_{p} to 50 by default. Second, for matrix crossover and matrix mutation, w is sampled from the Poisson distribution with parameter λ. Given a molecule $\mathcal{M}$ with N atoms, λ is set to $0.5 \times N$. And N_{l} is set to 5. Third, for evaluation and selection, the parameter N_{new} is set to $5 \times N_{\mathrm{p}}$ and the fitness function can be employed from the rdkit software [37], which is the same as the previous work [20, 21, 31–35]. Finally, for the overall framework, the ratio of molecules generated by crossover and mutation is set to 1 : 5. And we mainly concentrate on two important molecular properties: penalized logP and QED, where penalized logP accounts for synthetic accessibility [38] and QED [39] is an indicator of drug-likeness. These two properties are widely used in the previous work [12, 25, 28, 40].

4.1.3 Baselines

We compare CALM with 10 baselines which can be divided into four classes. The first class is the ZINC dataset itself. We will compare the molecules generated by CALM with the molecules in the dataset. The second class is deep learning methods, such as MolDQN [40], JT-VAE [28], GCPN [12], CVAE [25] and ALL-SMILES [30], which are all the representative ones for molecular generation. The third class is the work based on the genetic algorithm framework, which are ChemGE [31], GB-GA [32], GA+D [33]. The last class is traditional search methods, and contains random search.

4.2 Experimental Results

In this subsection, we present the results of extensive experiments.

4.2.1 Constraint Molecular Optimization for Large Molecular Space

In this task, we optimize the penalized logP, which is a logP score that accounts for the ring size and synthetic accessibility, while constraining the degree of deviation $sim(G, G')$ between the original molecule G and the modified molecule G' above a threshold σ. This is a more realistic scenario in the drug discovery, where the development of new drugs usually starts with known molecules [41]. Following the setup in [32], we optimize 800 molecules in the ZINC dataset with low penalized logP. We calculate the molecular similarity based on the Morgan fingerprint of radius 2 using the Tanimoto similarity, which is the same as GB-GM [32]. Given one of the 800 molecules, we use the BFS algorithm to sample 50 substructures of this molecule to initialize the population. We run 800 times to find desired modified molecules for each given molecule. The result is shown in Table 1. The reason why our approach outperforms other work is that CALM can find large molecules with higher penalized logP while meeting the threshold of similarity. The averaged number of atoms in original molecules is 22.5. For $\sigma = 0.4$ and $\sigma = 0.6$, the averaged number of atoms in modified molecules is 91.9 and 63.1 respectively, which shows that CALM has the strong ability of exploring the large molecular space. By applying the z test, we find that the differences between the mean values obtained by CALM and the other algorithms are statistically significant with $|z| \geqslant 14.95$, which indicates that CALM has 99% confidence to be significant. Given the specific computer task for the molecular generation, CALM shows more powerful ability to cover the complex molecular environment.

4.2.2 Comparison of Distribution Underlying Molecules

In this task, we verify that the methods based on GANs and VAEs tend to learn the distribution underlying the dataset. However, the molecules generated by CALM are independent with the dataset and obey a different distribution. First, we introduce the data used in this task. For the ZINC dataset, we just select the 100 molecules with the highest logP property.

Table 1. Comparison on Constraint Molecular Optimization

Method	$\sigma = 0.4$		$\sigma = 0.6$	
	Improvement	Success (%)	Improvement	Success (%)
JT-VAE [28]	0.84±1.45	83.6	0.21±0.71	46.4
GCPN [12]	2.48±1.30	100.0	0.79±0.63	100.0
MMPA [42]	3.29±1.12	-	1.65±1.44	-
DEF [43]	3.41±1.67	85.9	1.55±1.19	72.6
VJTNN [42]	3.55±1.67	-	2.33±1.17	-
GA+D [33]	5.93±1.41	100.0	3.44±1.09	99.8
CALM	21.70±14.87	96.9	11.66±11.88	88.1

For CVAE, we generate 10 000 valid molecules and select 100 molecules with the highest logP property. For CALM, we limit the maximum number of heavy atoms to 30 in a molecule which is similar to that of the molecules in the ZINC dataset. Then we represent the molecules as bit vectors (fingerprints). In this case, we use the standard Morgan fingerprint with the radius of 4. This representation is used for a general analysis of the chemical space and widely used in previous work [31, 32]. Finally, for visualization, we compress these 300 fingerprints that represent 300 molecules in the three sets into two dimensions using PCA. The result shown in Fig. 5 illustrates that CALM can explore the molecular space that is not limited to the initial dataset. However, such unconstrained search often leads to molecules that cannot be synthesized.

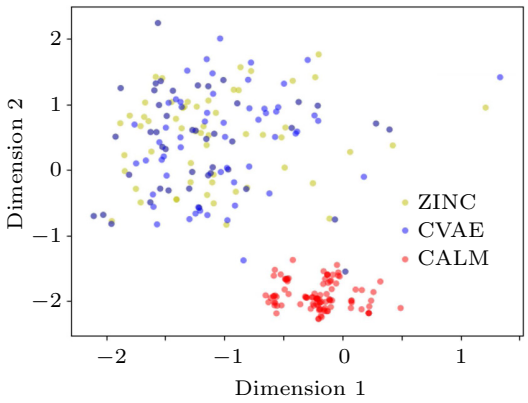


Fig.5. Comparison of top 100 molecules with the highest penalized logP after PCA. The molecular size is around 30.

4.2.3 *Molecular Property Optimization*

In this task, we focus on generating molecules with the highest possible penalized logP and QED scores. Penalized logP is a logP score that accounts for the ring size and the synthetic accessibility. Penalized logP has an unbounded range, while the QED has a range of [0, 1] by definition, and thus directly comparing the percentage improvement of QED may not be meaningful. This task is widely used to verify the performance of molecular generation methods. However, there exists insufficiency in the evaluation criteria in this task. The penalized logP is severely affected by the number of atoms. Normally, the larger the number of atoms, the higher the value of penalized logP. Therefore, it is trivial to compare the unbounded score of penalized logP without limiting the number of atoms.

In order to compare our work with the previous work fairly, we divide the previous work into two categories. One is the work that explicitly limits the number of atoms, and the other is the work that does not limit the number of atoms. For the former, we compare the efficiency between these tasks and CALM in detail. For the latter, we list the related data for reference only. In fact, we call on subsequent studies to give a detailed description of the number of atoms used in the work.

For the work that clearly limits the number of atoms, we take the experiment with a limited size of generated molecules, where the SMILES string corresponding to each molecule is restricted with a maximum length of 81 characters. Following the same experimental setup in [28], the experimental result is shown in Table 2. We can draw the following conclusions. CALM can generate competitive results in a short time. By applying the z test, we find that the differences between the mean values obtained by CALM and the other algorithms are statistically significant with $|z| \geqslant 4.17$.

Table 2. Maximum Penalized logP Scores Averaged over 10 Runs

Method	Max logP	Number of Molecules	CPU Time (s)
ChemGE	4.9±0.5	5 000	7 200
	5.6±0.5	20 000	28 800
GB-GM(80%)	3.4±0.6	1 000	90
	4.3±0.6	5 000	540
GB-GA(50%)	6.8±0.7	1 000	30
GB-GA(1%)	7.4±0.9	1 000	30
CALM	10.1±0.7	6 500	30

Note: The above data is referenced from [31] and [32].

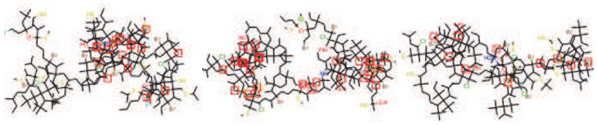
For the work with an unconstrained number of atoms, the result is presented in Table 3. It is mean-

Table 3. Comparison of Top-3 Property Scores of Generated Molecules Found by Each Model

Method	Penalized logP				QED			
	1st	2nd	3rd	Validity (%)	1st	2nd	3rd	Validity (%)
ZINC [36]	4.52	4.30	4.23	100.0	0.948	0.948	0.948	100.0
JT-VAE [28]	5.30	4.93	4.49	100.0	0.925	0.911	0.910	100.0
GCPN [12]	7.98	7.85	7.80	100.0	0.948	0.947	0.946	100.0
CVAE [25]	4.52	4.23	4.22	10.3	0.948	0.948	0.948	10.3
MolDQN [40]	11.84	11.84	11.82	100.0	0.948	0.944	0.943	100.0
GB-GA [32]	12.85	12.85	12.80	99.6	-	-	-	-
ChemGE [31]	6.37	6.24	6.22	27.3	0.947	0.947	0.947	16.3
ALL-SMILES [30]	42.46	42.42	41.54	-	0.948	0.948	0.948	-
CALM	63.75	61.48	60.51	100.0	0.948	0.948	0.948	100.0

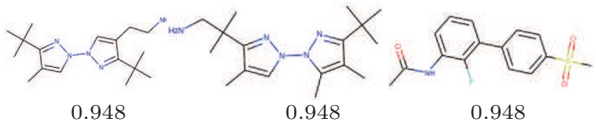
Note: The numerical results of ZINC, GCPN, JT-VAE, MolDQN, ALL-SMILES are referenced from corresponding papers. The remaining results are obtained by our experiments.

ingless to compare the scores without the limited number of atoms, because the number of atoms can greatly affect the result. The molecules with the top highest penalized logP score generated by CALM contain hundreds of atoms, which are very difficult to be synthesized in practice because of stability and synthesis. Therefore, we call on subsequent articles to explain the conditions and environment when conducting this experiment. And the molecules with the highest penalized logP scores and QED are shown in Fig.6 and Fig.7 respectively.



63.75 61.48 60.51

Fig.6. Molecules with top-3 penalized logP scores.



0.948 0.948 0.948

Fig.7. Molecules with top-3 QED scores.

4.2.4 *Accurate Property Target Task*

In this experiment, we take the accurate property target task to verify the performance of CALM. Molecular weight is a basic property for a molecule. We provide several target values of molecular weight. We make the experimental methods generate molecules whose molecular weights are close to these targets as much as possible. We select 50 results that are the closest to the target and calculate their mean and variance. The target values of the molecular weight are set to 150, 200 respectively, which is the same as [9]. We select ChmeGE, random search and ZINC as baselines. The random search is initialized with molecules in the ZINC dataset. And we run each method for 10 minutes under the same computing resources except for the ZINC dataset.

As shown in Table 4, the performance of CALM is the best obviously. The target of the molecular weight with a value of 150 deserves more attention, where a small number of molecules are around this target in the ZINC dataset.

Table 4. Comparison of Results of Accurate Property Task by Each Model

Model	Molecular Weight			
	150		200	
	Mean	STD	Mean	STD
ZINC [36]	150.36	4.50	200.04	2.52
ChemGE [31]	171.31	10.58	200.09	1.18
Random search	203.51	14.12	206.75	12.68
CALM	150.10	9.09	200.02	1.26

Note: STD: standard deviation.

4.2.5 *Effect of the Parameters*

In the following, we will discuss the influence of parameters of CALM.

First, we show the effect of the mutation operator under different population sizes. As mentioned above, if we have no limit on the number of atoms, the logP results will not be comparable. In this case, we set the upper limit of the number of atoms to 70. The search space is limited; therefore the results are comparable. As shown in Fig. 8, we run CALM for 30 minutes to find the highest penalized logP. The solid lines represent different population sizes with molecular mutation. With the same population size, the solid line is higher than the same color dashed line, which indicates that

the molecular mutation does contribute to the whole algorithm. And the result also shows that if the population size is larger than 250, the time consuming of a single iteration will be much greater than that when the population is 50 or 100, which will have a negative effect.

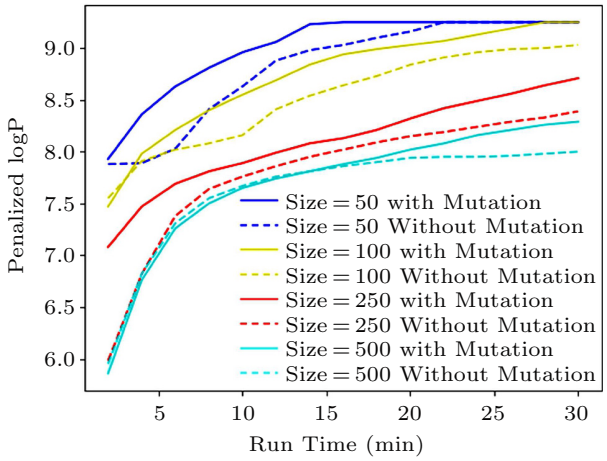


Fig.8. Effect of the mutation operator under different population sizes.

Second, we show the effect of the layer of the mutation operator N_l, which affects the scope of a single mutation operation. As shown in Fig.9, we run CALM with different execution times of the mutation operator for searching the molecules with the highest QED. The larger the size, the better the effect. We set N_l to 5 by default in terms of the trade-off between performance and computing resources.

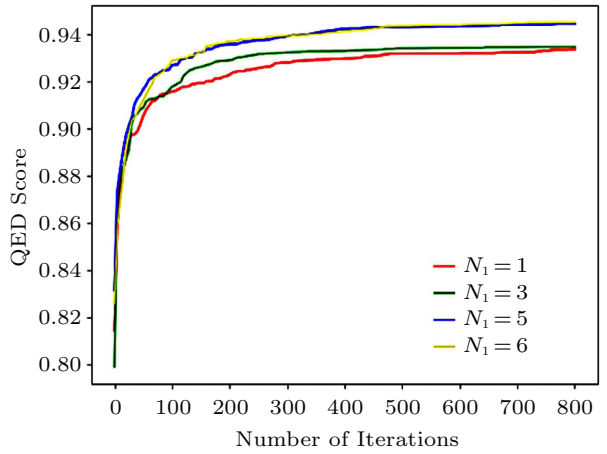


Fig.9. Effect of different execution times of mutation operator.

4.2.6 *Algorithm Analysis and Explanation*

In the following, we will explain and analyze the advantages, disadvantages and characteristics of CALM. As shown above, the performance of CALM is relatively better than the other baselines'. The main reason is that the algorithm's crossover and mutation operations are relatively simple and flexible. The generated multiple molecules will not have a similar paradigm; therefore the molecular space can be explored more efficiently.

In theory, the computational complexity of the mutation operator and crossover operator is $O(n^2)$, where n is the number of the atoms in the molecule. This complexity is acceptable under normal circumstances. However, our design still has some flaws. Assuming that a molecule contains n atoms, the time complexity for adding or reducing an atom is $O(n)$. Given that the above operations will be performed in large numbers, this is unacceptable especially when n is large. Therefore, proposing a more effective molecular representation is still a challenging task.

Another issue to explore is the indication of the number of simulation executions of the stochastic processes. However, it depends on specific tasks. Therefore, it is very difficult to give a suitable guidance. But in fact, there are $O(N_{\text{new}} \times n \times n)$ stochastic choices in one iteration, where N_{new} is the number of new generated molecules and n is the average number of atoms in a molecule. However, the number of new molecules that can be generated in each iteration depends on the population size. In practice, we have tested our algorithm in our laptops. CALM can produce 13 000 new molecules per minute in the constraint logP task. This data can be used to guess how many stochastic processes will be generated in the actual task.

5 Conclusions

In this paper, we introduced an efficient molecular generation method called CALM. We performed z tests on the results of the molecular generation tasks, which showed that CALM is of significance with more than 99% confidence. At the same time, CALM is efficient in time. CALM can achieve the above performance in a short time. Additionally, we call on the scholars in the future to pay more attention to the insufficiency in experimental evaluation standard mentioned in this work.

Acknowledgement The authors would like to thank the valuable comments from the reviewers and those important corrections from Dr. Jan H. Jenson.

References

[1] DiMasi J A, Grabowski H G, Hansen R W. Innovation in the pharmaceutical industry: New estimates of R&D costs. *Journal of Health Economics*, 2016, 47: 20-33. DOI: 10.1016/j.jhealeco.2016.01.012.

[2] Sanchez-Lengeling B, Aspuru-Guzik A. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 2018, 361(6400): 360-365. DOI: 10.1126/science.aat2663.

[3] Broadbelt L J, Stark S M, Klein M T. Computer generated pyrolysis modeling: On-the-fly generation of species, reactions, and rates. *Industrial and Engineering Chemistry Research*, 1994, 33(4): 790-799. DOI: 10.1021/ie00028a003.

[4] Devlin J, Chang M W, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv.: 1810.04805, 2018. https://arxiv.org/abs/1810.04805, Nov. 2022.

[5] Girshick R. Fast R-CNN. In *Proc. the 15th IEEE International Conference on Computer Vision*, December 2015, pp.1440-1448. DOI: 10.1109/ICCV.2015.169.

[6] He K M, Gkioxari G, Dollár P, Girshick R, Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020, 42(2): 386-397. DOI: 10.1109/TPAMI.2018.2844175.

[7] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998, 86(11): 2278-2324. DOI: 10.1109/5.726791.

[8] Peters J, Schaal S. Policy gradient methods for robotics. In *Proc. the 19th IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2006, pp.2219-2225. DOI: 10.1109/IROS.2006.282564.

[9] Liu Q, Allamanis M, Brockschmidt M, Gaunt A L. Constrained graph variational autoencoders for molecule design. In *Proc. the 32nd International Conference on Neural Information Processing Systems*, Dec. 2018, pp.7806-7815.

[10] Schütt K T, Arbabzadah F, Chmiela S, Müller K R, Tkatchenko A. Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, 2017, 8: 13890. DOI: 10.1038/ncomms13890.

[11] Lu C Q, Liu Q, Wang C, Huang Z Y, Lin P Z, He L X. Molecular property prediction: A multilevel quantum interactions modeling perspective. In *Proc. the 33rd AAAI Conference on Artificial Intelligence*, Jul. 2019, pp.1052-1060. DOI: 10.1609/aaai.v33i01.33011052.

[12] You J X, Liu B W, Ying R, Pande V, Leskovec J. Graph convolutional policy network for goal-directed molecular graph generation. In *Proc. the 32nd International Conference on Neural Information Processing Systems*, Dec. 2018, pp.6412-6422.

[13] Hao Z K, Lu C Q, Huang Z Y, Wang H, Hu Z Y, Liu Q, Chen E H, Lee C. ASGN: An active semi-supervised graph neural network for molecular property prediction. In *Proc. the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2020, pp.731-752. DOI: 10.1145/3394486.3403117.

[14] Polishchuk P G, Madzhidov T I, Varnek A. Estimation of the size of drug-like chemical space based on GDB-17 data. *Journal of Computer Aided Molecular Design*, 2013, 27(8): 675-679. DOI: 10.1007/s10822-013-9672-4.

[15] Macarron R, Banks M N, Bojanic D, Burns D J, Cirovic D A, Garyantes T, Green D V S, Hertzberg R P, Janzen W P, Paslay J W, Schopfer U, Sittampalam G S. Impact of high-throughput screening in biomedical research. *Nature Reviews Drug Discovery*, 2011, 10(3): 188-195. DOI: 10.1038/nrd3368.

[16] Pyzer-Knapp E O, Suh C, Gómez-Bombarelli R, Aguilera-Iparraguirre J, Aspuru-Guzik A. What is high-throughput virtual screening? A perspective from organic materials discovery. *Annual Review of Materials Research*, 2015, 45: 195-216. DOI: 10.1146/annurev-matsci-070214-020823.

[17] Goodfellow I J, PougetAbadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In *Proc. the 27th International Conference on Neural Information Processing Systems*, December 2014, pp.2672-2680.

[18] Kingma D P, Welling M. Auto-encoding variational bayes. arXiv: 1312.6114, 2013. https://arxiv.org/abs/1312.6114, Nov. 2022.

[19] Kipf T N, Welling M. Variational graph auto-encoders. arXiv: 1611.07308, 2011. https://arxiv.org/abs/1611.07308, Nov. 2022.

[20] Grover A, Zweig A, Ermon S. Graphite: Iterative generative modeling of graphs. In *Proc. the 36th International Conference on Machine Learning*, May 2019, pp.2434-2444.

[21] Simonovsky M, Komodakis N. GraphVAE: Towards generation of small graphs using variational autoencoders. In *Proc. the 27th International Conference on Artificial Neural Networks*, Oct. 2018, pp.412-422.

[22] You J X, Ying R, Ren X, Hamilton W L, Leskovec J. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proc. the 35th International Conference on Machine Learning*, Jul. 2018, pp.5694-5703.

[23] Liao R J, Li Y J, Song Y, Wang S L, Hamilton W L, Duvenaud D, Urtasun R, Zemel R. Efficient graph generation with graph recurrent attention networks. arXiv: 1910.00760, 2019. https://arxiv.org/abs/1910.00760, Oct. 2019.

[24] You J X, Wu H Z, Barrett C, Ramanujan R, Leskovec J. G2SAT: Learning to generate SAT formulas. In *Proc. the 32nd International Conference on Neural Information Processing Systems*, Dec. 2019, pp.10552-10563.

[25] Gómez-Bombarelli R, Wei J N, Duvenaud D, Hernández-Lobato J M, Sánchez-Lengeling B, Sheberla D, Aguilera-Iparraguirre J, Hirzel T D, Adams R P, Aspuru-Guzik A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 2018, 4(2): 268-276. DOI: 10.1021/acscentsci.7b00572.

[26] Weininger D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling*, 1988, 28(1): 31-36. DOI: 10.1021/ci00057a005.

[27] Samanta B, De A, Jana G, Chattaraj P K, Ganguly N, Rodriguez M G. NeVAE: A deep generative model for molecular graphs. In *Proc. the 33rd AAAI Conference on Artificial Intelligence*, Jul. 2019, pp.1110-1117. DOI: 10.1609/aaai.v33i01.33011110.

[28] Jin W G, Barzilay R, Jaakkola T S. Junction tree variational autoencoder for molecular graph generation. In *Proc. the 35th International Conference on Machine Learning*, Jul. 2018, pp. 2328-2337.

[29] Sutton R S, Barto A G. Reinforcement Learning: An Introduction. MIT Press, 2018.

[30] Alperstein Z, Cherkasov A, Rolfe J T. All SMILES variational autoencoder. 1905.13343, 2019. https://arxiv.org/abs/1905.13343, Nov. 2022.

[31] Yoshikawa N, Terayama K, Sumita M, Homma T, Oono K, Tsuda K. Population-based de novo molecule generation, using grammatical evolution. *Chemistry Letters*, 2018, 47(11): 1431-1434. DOI: 10.1246/cl.180665.

[32] Jensen J H. A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space. *Chemical Science*, 2019, 10(12): 3567-3572. DOI: 10.1039/C8SC05372C.

[33] Nigam A, Friederich P, Krenn M, Aspuru-Guzik A. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. In *Proc. the 8th International Conference on Learning Representations*, April 2020, pp.250-256.

[34] Banzhaf W, Nordin P, Keller R E, Francone F D. Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Application. Morgan Kaufmann Publishers, 1998.

[35] Kim Y, Kim W Y. Universal structure conversion method for organic molecules: From atomic connectivity to three-dimensional geometry. *Bulletin of the Korean Chemical Society*, 2015, 36(7): 1769-1777. DOI: 10.1002/bkcs.10334.

[36] Irwin J J, Sterling T, Mysinger M M, Bolstad E S, Coleman R G. ZINC: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 2012, 52(7): 1757-1768. DOI: 10.1021/ci3001277.

[37] Coley C W, Green W H, Jensen K F. RDChiral: An RDKit wrapper for handling stereochemistry in retrosynthetic template extraction and application. *Journal of Chemical Information and Modeling*, 2019, 59(6): 2529-2537. DOI: 10.1021/acs.jcim.9b00286.

[38] Ertl P, Schuffenhauer A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 2009, 1: Article No. 8. DOI: 10.1186/1758-2946-1-8.

[39] Bickerton G R, Paolini G V, Besnard J, Muresan S, Hopkins A L. Quantifying the chemical beauty of drugs. *Nature Chemistry*, 2012, 4(2): 90-98. DOI: 10.1038/nchem.1243.

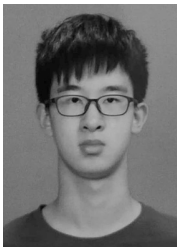
[40] Zhou Z P, Kearnes S, Li L, Zare R N, Riley P. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 2019, 9(1): 10752. DOI: 10.1038/s41598-019-47148-x.

[41] Bleicher K H, Böhm H J, Müller K, Alanine A I. Hit and lead generation: Beyond high-throughput screening. *Nature Reviews Drug Discovery*, 2003, 2(5): 369-378. DOI: 10.1038/nrd1086.

[42] Jin W G, Yang K, Barzilay R, Jaakkola T. Learning multimodal graph-to-graph translation for molecular optimization. arXiv: 1812.01070, 2018. https://arxiv.org/abs/1812.01070, Nov. 2022.

[43] Assouel R, Ahmed M, Segler M H, Saffari A, Bengio Y. DEFactor: Differentiable edge factorization-based probabilistic graph generation. arXiv: 1811.09766, 2018. https://arxiv.org/abs/1811.09766, Nov. 2022.

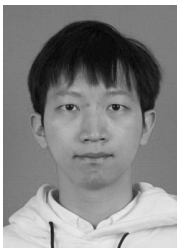
Jian-Fu Zhu received his Bachelor's degree in communication engineering from Hunan University, Changsha, in 2018. He is currently a postgraduate student, School of Computer Science and Technology, University of Science and Technology of China, Hefei. His research interests mainly include graph neural networks, molecular property prediction and molecular generation.



Zhong-Kai Hao is an undergraduate student, School of Mathematics, University of Science of Technology of China, Hefei. He has been awarded Yang Yuanqing Education Fund Scholarship. His interests mainly contain graph neural networks, molecular property prediction and active learning.

Qi Liu received his Ph.D. degree in computer science from University of Science and Technology of China, Hefei, in 2013. He is a professor of School of Computer Science and Technology, University of Science and Technology of China, Hefei. He has been awarded ICDM'11 Best Research Paper Award, KDD'18 (Research Track) Best Student Paper Award, KSEM'13 Best Paper Award, Alibaba Dharma Academy Green Orange Award, and so on. His general area of research is data mining and knowledge discovery.

Yu Yin received his Bachelor's degree in computer science from University of Science and Technology of China, Hefei, in 2017. He is a Ph.D. student, School of Computer Science and Technology, University of Science and Technology of China, Hefei. He has a wide range of research interests including reinforcement learning, natural language processing, and data mining.



Cheng-Qiang Lu received his Bachelor's degree in computer science from Nanjing University of Aeronautics and Astronautics, Nanjing, in 2017, and his Master's degree in computer science from University of Science and Technology of China, Hefei, in 2020. He is a senior algorithm researcher employed by Alibaba Group now. His research mainly focuses on the application of graph neural networks.

Zhen-Ya Huang received his Ph.D. degree in computer science from University of Science and Technology of China, Hefei, in 2020. He is an associate researcher of School of Computer Science and Technology, University of Science and Technology of China, Hefei, now. Dr. Huang has been awarded Excellent Award of President Scholarship of Chinese Academy of Sciences, the 6th Huayu Fund Scholarship for Graduate Students and so on. The application of data mining for education and finance is his main research interest.

En-Hong Chen received his Ph.D. degree in computer science from University of Science and Technology of China, Hefei, in 1996. Chen is a professor, an executive dean of School of Data Science and the vice dean of School of Computer Science and Technology of University of Science and Technology of China, Hefei. He is a fellow of CCF, and a senior member of IEEE. Chen won the Best Application Paper Award of KDD'08 and Best Research Paper Award of ICDM'11. His current research interests are data mining and machine learning, especially social network analysis and recommender systems.