MolSearch: Search-based Multi-objective Molecular Generation and Property Optimization

Mengying Sun
sunmeng2@msu.edu
Michigan State University
East Lansing, Michigan, USA

Jing Xing
xingjin1@msu.edu
Michigan State University
Grand Rapids, Michigan, USA

Han Meng
menghan1@msu.edu
Michigan State University
East Lansing, Michigan, USA

Huijun Wang
huijun.wang@agios.com
Agios Pharmaceuticals
Cambridge, Massachusetts, USA

Bin Chen
chenbi12@msu.edu
Michigan State University
Grand Rapids, Michigan, USA

Jiayu Zhou
jiayuz@msu.edu
Michigan State University
East Lansing, Michigan, USA

ABSTRACT

Leveraging computational methods to generate small molecules with desired properties has been an active research area in the drug discovery field. Towards real-world applications, however, efficient generation of molecules that satisfy **multiple** property requirements simultaneously remains a key challenge. In this paper, we tackle this challenge using a search-based approach and propose a simple yet effective framework called MolSearch for multi-objective molecular generation (optimization). We show that given proper design and sufficient domain information, search-based methods can achieve performance comparable or even better than deep learning methods while being computationally efficient. Such efficiency enables massive exploration of chemical space given constrained computational resources. In particular, MolSearch starts with existing molecules and uses a two-stage search strategy to gradually modify them into new ones, based on transformation rules derived systematically and exhaustively from large compound libraries. We evaluate MolSearch in multiple benchmark generation settings and demonstrate its effectiveness and efficiency.

CCS CONCEPTS

• **Applied computing → Bioinformatics**; • **Computing methodologies → Machine learning algorithms**.

KEYWORDS

Molecular Generation and Optimization, Monte Carlo Tree Search, Design Moves

ACM Reference Format:
Mengying Sun, Jing Xing, Han Meng, Huijun Wang, Bin Chen, and Jiayu Zhou. 2022. MolSearch: Search-based Multi-objective Molecular Generation and Property Optimization. In *Proceedings of Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534678.3542676

1 INTRODUCTION

Searching new compounds with desired properties is a routine task in early-stage drug discovery [7]. Common examples include improving the binding activity against one or multiple therapeutic targets while keeping the drug-likeness property; increasing drug solubility while minimizing the change of ADME properties. However, a small modification of chemical structures may lead to an unwanted change of property that even seasoned chemists cannot foresee. Moreover, the virtually infinite chemical space and the diverse properties for consideration impose significant challenges in practice [31]. Advanced machine learning models built upon historical biological and medicinal chemistry data are posed to aid medicinal chemists in designing compounds with multiple objectives efficiently and effectively.

Leveraging computational methods to facilitate and speed up the drug discovery process has always been an active research area [34, 40]. In particular, using deep learning (DL) and reinforcement learning (RL) to generate and optimize molecules has recently received broad attentions [19, 37, 39], which we will summarize in detail later in section 2. Despite the advances, such methods either rely on the quality of latent space obtained by generative models [32], or suffer from high variation, making it hard to train [35]. In reality, DL/RL methods consume large computational resources while the generated molecules could be hard to synthesize. Methods combing multiple objectives often do not work well [13].

In this paper, instead of leveraging DL, we propose a practical search-driven approach based on Monte Carlo tree search (MCTS) to generate molecules. We show that under proper design, search methods can achieve comparable or even better results to DL methods in terms of multi-objective molecular generation and optimization, while being computationally much more efficient. *The efficiency and multi-objective nature allow it to be readily deployed in massive real-world applications such as early-stage drug discovery.*

To design an efficient and effective search framework for practical multi-objective molecular generation and optimization, we need to answer the following questions. *Q1*: where to start; *Q2*: what to search; and *Q3*: how to search. For Q1, prior works using MCTS to generate molecules mostly start with empty or very small molecules [18, 38]. Since most drug-like molecules have 10-40 atoms, the search tree can grow deep and the space grows exponentially with the depth, making the search process less efficient and effective. Some work thus uses pre-trained RNN as a simulator to expand the tree however it requires additional pretraining [38]. Moreover, real optimization projects often have some candidates in place. For Q2, most prior works use atom-wise actions to modify molecules, which makes it hard to improve target property while maintaining drug-likeness and synthesis abilities [39, 43]. Fragment-wise actions tend to work better but the editing rules are mostly heuristic [20, 37]. For Q3, most existing methods combine all the objectives into one score and optimize for that [27, 37]. However, simple aggregation of scores neither fully considers the differences of objective classes nor reflects real optimization scenario.

We seek solutions to Q1-Q3 and propose MolSearch, a simple and practicable search framework for multi-objective molecular generation and optimization. In MolSearch, we start with existing molecules and optimize them towards desired ones (Q1). The modification is based on design moves [3], i.e., transformation rules that are chemically reasonable and derived from large compound libraries (Q2). The property objectives are split into two groups with its rationale explained in detail later. The first group contains all biological properties such as inhibition scores to proteins, and the second group includes non-biological properties such as drug-likeness (QED) and synthetic accessibility (SA). Correspondingly, the entire search process consists of two stages: a HIT-MCTS stage that aims to improve biological properties, followed by a LEAD-MCTS stage that focuses on non-biological properties while keeping biological ones above certain threshold. Each stage contains a multi-objective Monte Carlo search tree where different property objectives are considered separately rather than combined (Q3).

We evaluate MolSearch on benchmark tasks under different generation settings and compare it with various baselines. The results show that MolSearch is on par with or even better than the baselines based on evaluation metrics calculated from success rate, novelty and diversity, within much less running time. In summary, our contributions are as follows:

- MolSearch is among the first that make search-based approaches comparable to DL-based methods in terms of multi-objective molecular generation and optimization.
- MolSearch combines mature components, e.g., tree search, design moves, multi-objective optimization, in a novel way such that the generated molecules not only have desired properties but also achieve a wide range of diversity.
- MolSearch is computationally very efficient and can be easily adopted into any real drug discovery projects without additional knowledge beyond property targets.
- Additional to molecular generation, MolSearch is more tailored for hit-to-lead optimization given the nature of its design, which makes it very general and applicable.

2 RELATED WORK

In general, molecular property optimization comprises three components or less: representation, generative model, and optimization model. The representation of molecules can be simplified molecular-input line-entry system (SMILES) strings, circular fingerprints, and raw graphs, which often corresponds to certain type of generative models. Grouping by each component can be too detailed to capture the big picture, therefore we choose to categorize the related studies based on optimization models.

The first group optimizes molecules via Bayesian optimization [11, 16, 19, 23]. These methods first learn a latent space of molecules via generative models such as auto-encoders (AEs), then optimize the property by navigating in that latent space, and generates molecules through the decoding process. Most methods in this category only optimize for non-biological properties such as QED and penalized logP [1], and focus on metrics such as validity of generated molecules. They heavily rely on the quality of learned latent spaces, which impose challenges for multi-objective optimization.

Instead of manipulating latent representations, the second category utilizes reinforcement learning (RL) to optimize molecular property. One line of research applies policy gradient to fine-tune generative models, e.g., GAN-based generator [12, 30], GNN-based generator [39], Flow-based generator [25, 33, 41] to generate molecules with better property scores. The other line of work directly learns the value function of molecule states and optimizes for a given property via double Q-learning [43].

Besides RL, the third category uses genetic algorithms (GAs) to generate molecules with desired properties [1, 18, 27]. The generation process of genetic algorithms usually follows mutation and cross-over rules that are predefined from a reference compound library or domain expertise, which are not easy to obtain in general. Some work [27] also combines deep learning, e.g., a discriminator into GA generator to increase the diversity of molecules.

The least explored category aims to optimize molecular property using search methods, e.g., Monte Carlo tree search (MCTS). The earliest work traces back to [18, 38] in which the authors use pre-trained RNNs or genetic mutation rules as the simulator for tree expansion and simulation. [28] proposes atom-based MCTS method without predefined simulator. Again, all the methods focus on single and non-biological properties and are not tailored for multi-objective optimization. Not until recently RationaleRL [20] enables multi-objective molecular generation by first searching property-related fragments using MCTS and then completing the molecular graph using reinforcement learning.

There are also pioneering works that do not fall into any of the categories above, e.g., MARS [37] proposes a Markov sampling process based on molecular fragments and graph neural networks (GNNs) and achieves state-of-the-art performance. Li et al. [24] proposed a conditional graph generative framework for optimizing molecular properties. In summary, we see a trend of utilizing fragment-based actions and directly navigating in the chemical space as opposed to generative models in recent works. Interested readers can refer to [13, 42] for a comprehensive understanding of advances in molecular generation and optimization.

3 METHOD

In this section, we present the proposed framework MolSearch as shown in Figure 1. The entire process consists of two search stages: a HIT-MCTS stage and a LEAD-MCTS stage. HIT-MCTS aims to modify molecules for better biological properties while LEAD-MCTS stage seeks molecules with better non-biological properties. Each stage utilizes a multi-objective Monte Carlo search tree to search for desired molecules.

[1]water-octanol partition coefficient penalized by synthesis accessibility and number of cycles having more than 6 atoms, i.e., PlogP(m)=logP(m)-SA(m)-cycle(m)

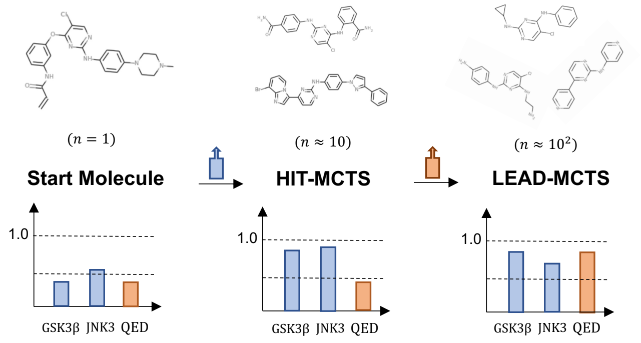


Figure 1: Overall framework of MolSearch. For a given start molecule, it first goes through a HIT-MCTS stage which aims to improve the biological properties, e.g., GSK3β and JNK3, followed by a LEAD-MCTS stage where non-biological properties such as QED are optimized. n refers to number of generated molecules and y-axis reflects the normalized scores.

3.1 Problem Definition

Molecule modification can be mathematically formulated as a Markov decision process (MDP) [5] given that the generated molecule only depends on the molecule being modified. The MDP can be written as $M = (S, A, f, R)$ where S denotes the set of states (molecules), A denotes the set of actions (modifications), $f : S \times A \longrightarrow S$ is the state transition function. For molecule modification, the state transition is deterministic, i.e., $p(s_{t+1}|s_t, a_t) = 1$ for a given state-action pair. That is to say, by taking a modification action, the current molecule reaches the next molecule with that modification with probability 1. $R : S \longrightarrow \mathbb{R}^d$ is the reward received for a given state, where $d > 1$ if multiple reward objectives are considered. The goal is to take the action that maximizes the expected reward, which can be approximated as Eq (1) under repeated simulations [15]:

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{N(s)} \mathbb{I}_i(s,a) z_i, \tag{1}$$

where $N(s)$ denotes the simulation times starting from state s and $N(s, a)$ is the times that action a has been taken from state s. $\mathbb{I}_i(s, a)$ is an indicator function with value 1 if action a is selected from state s at i-th round, 0 otherwise. z_i is the final reward for i-th simulation round starting from state s. A larger value of $Q(s, a)$ indicates higher expected reward by taking action a from state s.

3.2 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) adopts a tree structure to perform simulations and estimate the value of actions. Meanwhile it also uses the previously estimated action values to guide the search process towards higher rewards [8]. The basic MCTS procedure consists of four steps per iteration:

a) *Selection.* Starting from the root node, a best child is recursively selected until a leaf node, i.e., a node that has not been expanded or terminated, is reached.

b) *Expansion.* The selected leaf node is expanded based on a policy until the maximum number of child nodes is reached.

c) *Simulation.* From each child node, recursively generate the next state until termination and get the final reward.

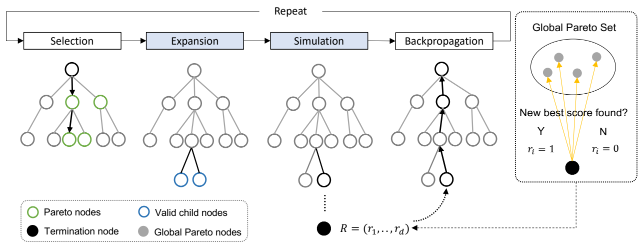


Figure 2: Multi-objective Monte Carlo tree search procedure. Each node represents an intermediate molecule which has a reward vector associated with it. A search iteration consists of selection, expansion, simulation, and backpropagation. For MolSearch, HIT-MCTS and LEAD-MCTS differ in the expansion and simulation policy (blue boxes).

d) *Backpropagation.* The reward is backpropagated along the visited nodes to update their statistics until the root node.
The process is repeated until a certain computational budget is met. The most important step of MCTS is the *selection* step where a criterion needs to be determined to compare different child nodes. The most commonly used criteria is the upper confidence bound (UCB1) [2, 22] in which a child node is selected to maximize:

$$UCB1 = \bar{X}_j + \sqrt{\frac{2\ln n}{n_j}},$$

where $\bar{X}_j$ is the averaged reward obtained so far for node j, n_j denotes times of node j being selected and n is the total times of iteration. The first term $\bar{X}_j$ favors exploitation, i.e., choose the node with greater average performance; while the second term $\sqrt{\frac{2\ln n}{n_j}}$ votes for exploration, i.e., choose nodes that have not been visited so far. UCB1 balances between exploitation and exploration to avoid being trapped in local optimums.

For single-objective MCTS, UCB1 is a scalar and maximization picks the node with the largest value. For multi-objective MCTS, the reward becomes a vector and the comparison is no longer straightforward. Next we formally define each component for multi-objective MCTS under the context of molecular generation.

3.3 Multi-objective Monte Carlo Tree Search

For molecular generation, each node of the tree (e.g., v_j) represents an intermediate molecule. It is associated with a molecule state s_j, number of visits n_j, and a reward vector $\mathbf{X}_j = (x_1, .., x_d) \in \mathbb{R}^d$ where d is the number of objectives. Without loss of generality, we assume that each objective is to be maximized. Before presenting how the reward is calculated, we first introduce the following definitions regarding comparisons between vectors:

Definition 1. Pareto Dominate. Given two points $X = (x_1, .., x_d)$ and $X' = (x'_1, .., x'_d)$, X is said to *dominate* X', i.e., $X \geq X'$ if and only if $x_i \geq x'_i, \forall i = 1, .., d$. X is said to *strictly dominate* X', i.e., $X > X'$ if and only if $X \geq X'$ and $\exists i$ such that $x_i > x'_i$.

Definition 2. Pareto Front. Given a set of vectors $\mathcal{A} \subset \mathbb{R}^d$, the non-dominant set $P_{\mathcal{A}}$ in $\mathcal{A}$ is defined as:

$$P_{\mathcal{A}} = \{X \in \mathcal{A} : \nexists X' \in \mathcal{A}\ s.t.\ X' > X\}$$

The *Pareto front* consists of all non-dominated points [36].

Algorithm 1: UCT algorithm for MO-MCTS.

```
Input: root node $v_0$ with state $s_0$, computation budge $N$,
       maximum number of child $K$, exploration scalar $\lambda$
1  Function SEARCH($v_0$):
2      for $i = 1, .., N$ do
3          $v_l$ = SELECTION($v_0$)            // $v_l = v_{\text{leaf}}$
4          $v_c$ = EXPAND($v_l$)               // $v_c = v_{\text{child}}$
5          $\mathbf{r}_c$ = SIMULATION($v_c$)
6          BACKPROP($v_c, \mathbf{r}_c$)
7      return $v_0'$;
8  Function SELECTION($v$):
9      while $v$ is fully expanded do
10         for $k = 1, .., K$ child node do
11             $\mathbf{U}_k = \frac{\mathbf{X}_k}{n_k} + \lambda\sqrt{\frac{4\ln n + \ln d}{2n_k}}$
12         $V_p$ = ParetoNodeSet($\mathbf{U}_1, .., \mathbf{U}_k$)
13     return Random($V_p$);
```

For a Monte Carlo search tree, we maintain a global pool of all the Pareto molecules found so far. At each simulation round, given a termination state (molecule) with property score $\mathbf{h} = (h_1, .., h_d) \in \mathbb{R}^d$, by comparing it with all Pareto molecules in the global pool, the reward vector $\mathbf{R} = (r_1, .., r_d) \in \mathbb{R}^d$ of this state is defined as:

$$r_i = \frac{1}{N_p}\sum_{l=1}^{N_p} \mathbb{I}[h_i > h_i^l], \qquad \forall i = 1, \ldots, d$$

where N_p is the number of Pareto molecules and h_i^l is the i-th property value of Pareto molecule l. The calculation of reward treats each dimension separately, regardless of their scale difference, which gains an advantage over methods that aggregate all dimensions into one score using predefined weights. We also update the global Pareto pool by adding new Pareto molecules if found and removing invalid ones based on the comparison result. The reward $\mathbf{R}$ will be used for backpropagation with the update formula:

$$n_v \leftarrow n_v + 1, \quad \mathbf{X}_v \leftarrow \mathbf{X}_v + \mathbf{R}, \quad v \leftarrow \text{parent of } v,$$

which concludes the backward part of MCTS.

Next we present the forward part. Starting from the root node, we recursively select the best child to proceed. To determine the best child for a given parent, we calculate the utility for each child:

$$\mathbf{U}_k = \frac{\mathbf{X}_k}{n_k} + \lambda\sqrt{\frac{4\ln n + \ln d}{2n_k}},$$

where $\mathbf{X}_k$ is the average reward obtained so far, n_k and n are the times child node k being visited and the total iterations. d is the reward dimension. Based on Definition 1 and 2, we compute the Pareto node set given statistics of all child nodes. Once the set is computed, we randomly select one child in the set to proceed. Once the selection step is done, we reach a node that has never expanded before. Then we expand the leaf node and start simulations from its children, get reward and backpropagate again. The overall MCTS procedure is illustrated in Figure 2 and Algorithm 1. Due to space limit, we do not present the procedure of expansion and simulation in Algorithm 1 since they are the same as classic single-objective

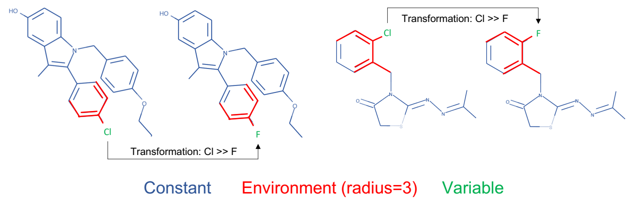


Figure 3: Example of design moves. A transformation is only valid conditional on the existence of certain environments.

count_stat	n	descriptive_stat	rule	env
# fragments	236,827	min	1	1
# environment	55,599	max	20,075	2,480
# rules	1,048,575	median	1	1
# unique rules	672,117	mean	1.78	1.56
Atom Types	C, N, O, Cl, F, P, Br, I, S			
# augment rules	436,532			
# trim rules	443,995			

Table 1: Statistics of rules extracted from ChEMBL on environment radius $r = 3$. # denotes "number of".

MCTS and can be found in many places such as [8]. The key component in expansion and simulation step is the policy that used to generate the next state. In MolSearch, within each search tree, expansion and simulation share the same policy to produce actions:

$$A_v = \text{actions}(s_v),$$

for each node v given current state s_v. The possible actions are obtained using transformations we will mention in the next section. Due to the large chemical space, usually there are thousands of possible actions for a given state and not all of them are promising, therefore a subset of actions are selected and served as a candidate pool for both expansion and simulation.

HIT-MCTS vs LEAD-MCTS. The two search stages in MolSearch differ in how the candidates are picked given the original possible actions. In HIT-MCTS, the candidate actions are those yielding states with better property scores as compared to the current **parent** state. In LEAD-MCTS, the candidate actions are those producing states with better property scores than a **constant** threshold.

Theoretical Analysis. The theoretical analysis of multi-objective MCTS has been presented in prior works following concentration inequalities and union bound. Readers can refer to [2, 10, 36].

3.4 Design Moves

A key challenge in MolSearch is the actions to take when searching for new molecules. The modification rules should be chemically reasonable, covering a variety of modification directions, and being large in size in order to successfully navigate in the chemical space. Design moves, proposed in [3], is such an approach. It extracts transformations among molecules based on matched molecular pair (MMP) [17] and outputs a collection of rules that systematically summarize the modification of molecules that exist and chemically valid in the current large compound database such as ChEMBL [26]. The transformation rules contain both atom-wise and fragment-wise modification and for the purpose of simplicity, we refer all of them as *fragments*.

Each rule consists of three major components, a left-hand-side fragment (lhs_frag), an environment, and a right-hand-side fragment (rhs_frag), and can be written as follows:

```
lhs_frag + environment >> rhs_frag
```

An example of design move transformation is shown in Figure 3. Each matched molecular pair has three parts. The constant part denotes the places that remain the same before and after transformation. The variable part denotes the fragment to be replaced. The environment is the most important part in design move which characterizes the context of a transformation. The range of the context is determined by the radius r and contains all the atoms that can be reached from the fragment to be replaced within step size r. Such constraint ensures the transformation is chemically reasonable and the larger the radius r, the more likely the assumption holds true [3]. In Figure 3, we see that even for the same lhs_frag and rhs_frag, due to that environments are different, the transformations are treated as different transformations rules.

We summarized the statistics of all the design move rules extracted from ChEMBL based on radius $r = 3$ in Table 1. We see that it contains more than 1 million transformation rules with more than 600K unique pairs of fragments to be replaced. There are also more than 200K fragments and 50K environments in the total rules. For a transformation rule, the frequency it happens in the database ranges from 1 to 20K, which covers both common and rare transformations. The number of environments for the same rule also ranges from 1 to 2.5K. Given ChEMBL is one of the largest chemical databases, the rules are expected to cover all the possible moves of common molecules of biological interest. Moreover, unlike most prior works which only allow atom or fragment addition, design moves contain modifications that can either increase or decrease the molecular size (436,532 vs 443,995), making it more flexible to find better modification directions.

3.5 Rationale of MolSearch

The last important question regarding MolSearch framework is the two-stage design in which biological properties are first optimized and then followed by optimization of non-biological properties. The reason is two-folded. First, we observe that lower non-biological property (e.g., QED and SA) values are often due to large size or large number of rings of molecules since the fragments are already chemically valid. That is to say, reducing the size of generated molecules can achieve better QED and SA scores in general. However, design move requires valid environment in order to perform modification, the larger the molecules are, the more actions could be found. Therefore, optimizing QED/SA has to come after optimizing biological properties. Second, such design is also inspired by the real-world drug discovery routine that we first find drugs that are biologically active and then optimize them regarding other properties.

Another interesting property of such design is that, in general, molecules from HIT-MCTS stage are quite large, due to that HIT-MCTS modifies molecules into hits by adding property-related fragments repeatedly; However, it is fine because LEAD-MCTS will trim the molecules for a higher QED/SA score by dropping property-unrelated fragments. The entire process will ensure that the final molecules satisfies all the property requirements.

4 EXPERIMENT

We conduct extensive experiments on benchmark tasks following [20, 37] to demonstrate the effectiveness of MolSearch. The results show that search methods can achieve comparable and sometimes superior performance compared to advanced deep learning methods given sufficient information and proper design of the algorithm.

4.1 Experiment Setup

Property Objectives. We consider two biological properties that measure the inhibition of proteins related to Alzheimer disease:

- GSK3β, score of inhibiting glycogen synthase kinase-3β
- JNK3, score of inhibiting c-Jun N-terminal kinase-3

The scores are predicted probabilities of inhibition by pretrained random forest models from [20]. For non-biological properties, we follow [20, 37] and also consider drug-likeness (QED) [6] and synthesis accessibility (SA) [14] scores. The SA score (originally in $[1, 10]$) is reversely normalized to $[0, 1]$. For all scores, the higher the better. The goal is to find compounds that mostly inhibit two essential proteins in Alzheimer's such that their potency is maximized while achieving favorable medicinal chemistry properties.
Multi-objective generation setting. We consider 6 different generation settings as in [20, 37]:

- GSK3β/JNK3: inhibit GSK3β or JNK3 without constraints on QED and SA scores.
- GSK3β+JNK3: jointly inhibit GSK3β and JNK3 without constraints on QED and SA scores.
- GSK3β/JNK3+QED+SA: inhibit GSK3β or JNK3 while being druglike and easy to synthesize.
- GSK3β+JNK3+QED+SA: jointly inhibiting GSK3β and JNK3 while being druglike and easy to synthesize.

Baselines. We compare MolSearch with state-of-the-art methods from each category summarized in section 2: 1) JT-VAE [19], a method uses Bayesian optimization based on hidden representations from a VAE based on molecule fragments. 2) GCPN [39], a method uses policy gradient to finetune a pre-trained molecule generator based on GNN. 3) MolDQN [43], a method directly learns the values of actions for target properties via double Q-learning and generate molecules based on that. 4) GA+D [27], a method utilizes genetic algorithm for molecule generation paired with an adversarial module to increase diversity. 5) RationaleRL [20], a method uses MCTS to find property-related fragments and then completes the graph using RL. 6) MARS [37], a method utilizes Markov sampling based on GNN and molecule fragments.
Evaluation Metrics. We evaluate the generated molecules using metrics similar to prior works [20, 37]: 1) success rate (SR): the proportion of resulted molecules that satisfy all the targeted objectives, i.e., QED ≥ 0.6, SA ≥ 0.67, GSK3$\beta \geq 0.5$, and JNK3 ≥ 0.5. 2) Novelty (Nov): the proportion of resulted molecules that have similarity less than 0.4 compared to the nearest neighbor x_{SNN} in the reference dataset, i.e., Nov $= \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[\text{sim}(x_i, x_{\text{SNN}}) < 0.4]$ where the similarity is calculated as the Tanimoto coefficient [4] between two Morgan fingerprints [29] of molecules. The reference dataset in prior works is training data while in our work, the reference data becomes the start molecules. 3) Diversity (Div): the pair-wise dissimilarity among the generated molecules, i.e., Div

Objectives	GSK3β				JNK3				GSK3β+JNK3			
Method	SR	Nov	Div	PM	SR	Nov	Div	PM	SR	Nov	Div	PM
JT-VAE	0.322	0.118	0.901	0.030	0.235	0.029	0.882	0.006	0.033	0.079	0.883	0.002
GCPN	0.424	0.116	0.904	0.040	0.323	0.044	0.884	0.013	0.035	0.080	0.874	0.002
RationaleRL	0.939	0.457	0.890	0.381	0.880	0.419	0.872	0.321	0.842	0.981	0.831	**0.686**
GA+D	0.85	1.00	0.71	0.60	0.53	0.98	0.73	0.38	0.85	1.00	0.42	0.36
MARS	1.000	0.840	0.718	0.603	0.988	0.889	0.748	**0.657**	0.995	0.753	0.691	0.518
MolDQN-emtpy	0.000	0.038	0.204	0.000	0.000	0.019	0.116	0.000	0.000	0.025	0.126	0.000
MolDQN-nonemtpy	0.341	0.304	0.856	0.089	0.175	0.288	0.857	0.043	0.050	0.421	0.858	0.018
MolSearch	1.000	0.739	0.862	**0.637** ± 0.009	1.000	0.728	0.846	0.616 ± 0.015	1.000	0.787	0.826	0.650 ± 0.009
MolSearch-5000	1.000	0.706	0.850	0.601 ± 0.023	1.000	0.685	0.845	0.579 ± 0.027	1.000	0.756	0.836	0.632 ± 0.030
Ranking				**1st**				2nd				2nd

Table 2: Overall performance of comparison methods on bio-activity objectives. Results of RationaleRL, MolDQN are obtained by running their open source code. Results of JT-VAE, GCPN, GA+D and MARS are taken from [20, 37]. For MolSearch, we repeat the experiments for 10 times and report the mean and standard deviation.

Objectives	GSK3β+QED+SA				JNK3+QED+SA				GSK3β+JNK3+QED+SA			
Method	SR	Nov	Div	PM	SR	Nov	Div	PM	SR	Nov	Div	PM
JT-VAE	0.096	0.958	0.680	0.063	0.218	1.000	0.600	0.131	0.054	1.000	0.277	0.015
GCPN	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
RationaleRL	0.891	0.341	0.891	0.270	0.787	0.190	0.874	0.131	0.750	0.555	0.706	0.294
GA+D	0.89	1.00	0.68	0.61	0.86	1.00	0.50	0.43	0.86	1.00	0.36	0.31
MARS	0.995	0.950	0.719	0.680	0.913	0.948	0.779	**0.674**	0.923	0.824	0.719	0.547
MolDQN-empty	0.000	0.224	0.331	0.000	0.000	0.089	0.245	0.000	0.000	0.046	0.166	0.000
MolDQN-nonempty	0.000	0.431	0.850	0.000	0.000	0.525	0.856	0.000	0.000	0.499	0.857	0.000
MolSearch	1.000	0.821	0.856	**0.702** ± 0.005	1.000	0.783	0.831	0.651 ± 0.009	1.000	0.818	0.811	**0.664** ± 0.007
MolSearch-5000	1.000	0.810	0.869	**0.704** ± 0.009	1.000	0.743	0.843	0.626 ± 0.012	1.000	0.797	0.827	**0.660** ± 0.009
Ranking				**1st**				2nd				**1st**

Table 3: Overall performance of comparison methods on bio-activity plus non-bioactivity objectives. Results of RationaleRL, MolDQN are obtained by running their open source code. Results of JT-VAE, GCPN, GA+D and MARS are taken from [37]. For MolSearch, we repeat the experiments for 10 times and report the mean and standard deviation.

$= \frac{2}{N(N-1)} \sum_{i,j=1;i\neq j}^{N} [1 - \text{sim}(x_i, x_j)]$. 4) PM: the product of SR, Nov and Div metrics, representing the possibility of generated molecules being simultaneously active, novel and diverse [37].

Start Molecules. A critical step in MolSearch is to pick the start molecules. We first download dataset from the Repurposing Hub (https://clue.io/repurposing), which consists of 6,758 FDA-approved and clinical trail drugs. We then cluster all the drugs based on their Tanimoto similarity using Butina algorithm [9] with threshold 0.4, a commonly used cutoff to quantify the structural similarity between molecules. It results in 5,727 small clusters, indicating that most molecules are not similar to each other. We select the centroid of each cluster, i.e., 5,727 dissimilar molecules, as the pre-processed dataset and construct start molecules from it. For benchmark objectives, to avoid making the task easier, we **remove** 1) all successful molecules, i.e., GSK3$\beta \geq 0.5$, JNK3 ≥ 0.5, QED ≥ 0.6, SA ≥ 0.67; 2) top molecules with either GSK3β or JNK3 score larger than 0.8 in the dataset. That is to say, no start molecules has biological score higher than 0.8. We then choose the remaining molecules with GSK3β and JNK3 score no less than 0.3 as the start molecules. Such selection strategy aligns with molecular optimization in reality that starts with molecules having some signals towards the desired property. In total 96 molecules satisfy the starting criteria.

Implementation Details. For MCTS, we set the maximum level of tree depth as 5 and test different values of maximum child nodes $K = [3, 5, 7]$ and the number of simulations $N = [5, 10, 20]$. For design move, we utilize rules derived from environmental radius $r = 3$ and do not impose frequency constraint on the actions, i.e., any action with frequency ≥ 1 will be considered in each modification step. All MolSearch experiments are done on AMD EPYC CPU cores [2]. Baselines requiring deep learning libraries are done on TITAN RTX GPUs with 24GB Memory.

n_child	n_sim	Avg	Median	STD
3	10	0.4h	0.38h	0.19h
5	20	1.02h	0.87h	0.93h

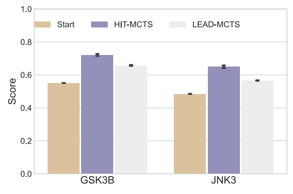
Table 4: Running time per molecule for MolSearch.

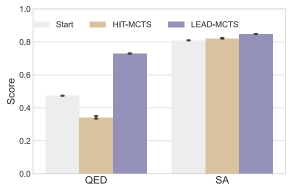
Running Time. In the GSK3β+JNK3+QED+SA setting, RationaleRL takes 6 hours to finetune the model; GA+D takes 300 steps and 4 hours to reach its best performance; MARS takes 10 hours to converge; MolDQN takes 5 and 10 hours to finish for empty and non-empty variants respectively. MolSearch takes on average 0.4-1.0 hours per molecule in both search stages (Table 4). Each molecule only occupies very small amount of memory and computational resources, making MolSearch much more efficient than deep learning methods regardless of computation constraints.

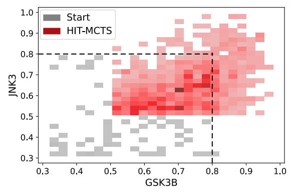
4.2 Benchmark Results

We perform MolSearch, i.e., start molecules $\rightarrow$ HIT MCTS $\rightarrow$ LEAD MCTS 10 times for each generation settings. In each search stage, we

[2]MolSearch code available at https://github.com/illidanlab/MolSearch.


(a) biological property


(b) non-biological property


(c) biological property

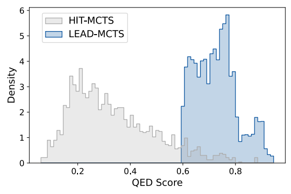

(d) non-biological property

Figure 4: Property dynamics across MolSearch stages. (a)(b): average scores over 10 runs at each stage. (c): distribution of bioactivity scores during Start and HIT-MCTS stage. (d): QED distribution between HIT-MCTS and LEAD-MCTS stage.

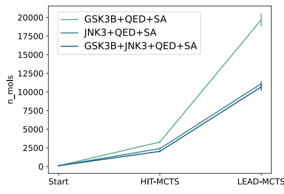
keep track of valid molecules and add them to the final set. Because the number of generated molecules is not controllable in MolSearch, we calculated the metrics for two sets of generated molecules: 1) MolSearch: all the molecules generated by MolSearch; 2) MolSearch-5000: top 5000 molecules generated by MolSearch, ranked by the average score of all properties considered in one setting, to match the number of molecules generated by other baseline methods.

Overall Performance. We summarize all the results in Table 2 and Table 3. MolSearch outperforms all baselines on 3 generation settings and always rank high (1st or 2nd) in terms of PM. Specifically, when considering non-bioactivity objectives, i.e., GSK3β+JNK3+QED+SA, MolSearch significantly outperforms the best baseline by 21.4% on the PM metric. Among all the metrics, MolSearch falls short on the novelty metric since it starts from known molecules and modify them into new ones. However, the novelty still ranks good via the two-stage design of MolSearch such that the generated molecules are not too similar as the original ones. The diversity of molecules generated by MolSearch always ranks high, possibly due to 1) dissimilarity of start molecules, 2) separation of different property objectives and 3) Pareto search on all objective directions.

Start Molecule	GSK3β	JNK3	QED	SA
Empty	0.262	0.083	0.870	0.603
Non-empty	0.334	0.216	0.217	0.586

Table 5: Average scores of generated molecules by MolDQN in GSK3β+JNK3+QED+SA setting.

Moreover, we conduct extensive experiments for the baseline MolDQN because it is the deep learning version of MCTS that tries to learn the values of all the actions and generate molecules that maximize the values. The differences between MolDQN and MolSearch can help verify the motivation and effectiveness of MolSearch. First, MolDQN-empty starts with empty molecules and uses atom-wise actions, and the SR of generated molecules are extreme low ($\approx$ 0.00) in all settings. When we look into the scores of generated molecules, as shown in Table 5, we find the QED and SA score of generated molecules are relatively high while GSK3β and JNK3 scores are very low. This means that QED and


(a) Across stages

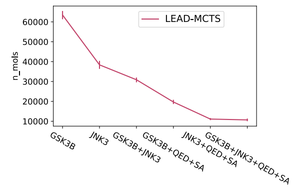
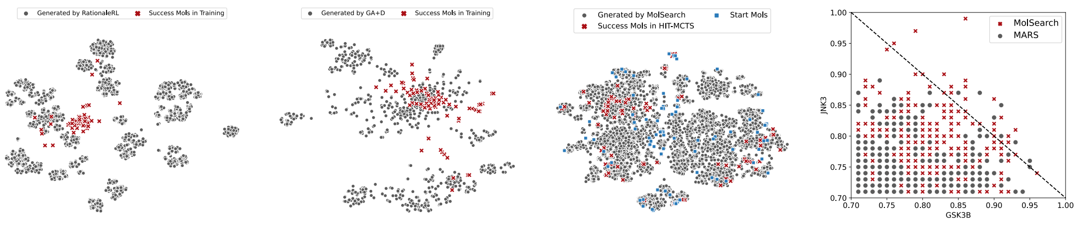

(b) Across objectives.

Figure 5: Number of generated molecules across MolSearch stages and different generation settings (10 runs).

SA are easier to optimize than biological objectives when starting from empty molecules and using atom-wise actions. However, in most real applications, optimizing biological objectives are the major focus before one considers drug-likeness and synthesis abilities. Second, MolDQN-nonempty starts from the same molecules we used in MolSearch, however, the success rates are still low although improved compared to MolDQN-empty. This is due to that MolDQN only allows addition actions thus cannot reduce the size of molecules, making QED and SA drop significantly. Third, the low performances of both MolDQN variants imply that atom-wise actions generally works less effective compared to fragment-based actions for improving biological properties. For MolSearch, the search trees can find desired molecules with relatively small depth and width, therefore it is not necessary to use Deep Q-learning to approximate the action values. All the above observations echo the rationale of MolSearch's design.

MolSearch Dynamics. We next verify if the change of property scores across stages aligns with design motivation of MolSearch. HIT-MCTS aims to improve biological properties and Figure 4a confirms a significant elevation for GSK3β and JNK3 scores. LEAD-MCTS aims to improve non-biological properties and Figure 4b reflects such improvement especially for QED (Figure 4d). Figure 4c demonstrates that, even if we remove all successful molecules and top molecules at start (0.3-0.8 dashed box with grey points), MolSearch is still able to find molecules with both scores larger than 0.8 (red region outside dashed box), demonstrating its power. Figure 5a shows the number of molecules generated in each stage for three settings where both biological and non-biological objectives are considered. We observe an exponential increase from start molecules to the later two stages. GSK3β is easier to optimize as compared to JNK3. Figure 5b shows the number of final molecules generated by MolSearch for all settings. As the number of objectives increases, less valid molecules are found, which is reasonable.

Visualization. We compare the molecules generated under setting GSK3β+JNK3+QED+SA by different methods using t-SNE plots shown in Figure 6 (a)-(c). The red crosses are the molecules that satisfy all the requirements in reference (training) dataset, while grey dots are molecules generated by each method. For MolSearch, there are no successful molecules in the start (reference) dataset, instead we plot the successful ones in HIT-MCTS stage. The start molecules of MolSearch are also plotted for reference (Figure 6c). We observe that baseline methods such as GA+D and RationaleRL generate molecules with large clusters, indicating relatively low diversity. The molecules generated by MolSearch evenly span the entire embedding space and also cover some novel regions compared to start molecules. MARS is very similar to MolSearch whose



(a) RationaleRL **(b) GA+D** **(c) MolSearch** **(d) MolSearch vs MARS**

Figure 6: t-SNE visualization of generated molecules and positive molecules in the reference (training) dataset.

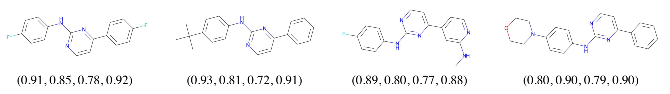


Figure 7: Sample molecules generated by MolSearch in the GSK3β+JNK3+QED+SA setting with associated scores.

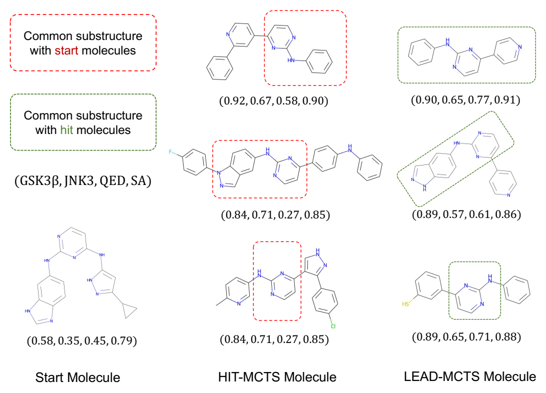


Figure 8: MolSearch path for GSK3β + JNK3 + QED + SA.

generated molecules enjoy both diversity and novelty, therefore we seek other comparison between MARS and MolSearch. As shown in Figure 6d, MolSearch is able to find more dominant molecules in terms of biological properties as compared to MARS (5 runs). We visualize the structure of several molecules generated by MolSearch with high property scores in Figure 7.

Figure 8 shows an example trajectory of MolSearch under the generation setting GSK3β + JNK3 + QED + SA. The property scores for the start molecule are relatively low. After HIT-MCTS stage, the generated molecules obtain higher GSK3β and JNK3 score by replacing certain substructures of the original molecule while also keeping certain original substructures. As we also can see, the QED score for HIT molecules are extremely low due to their large size. After LEAD-MCTS stage, the QED scores of the final molecules are elevated by dropping fragments that are less property related. The scaffold of the final molecules is not simply the substructure of start molecules but rather a combination of fragments from start molecules and new fragments from transformation rules. Also, the replacement is not completed in one round because the added fragments are relatively large, indicating the states are reached by multiple search steps instead of one.

Sensitivity Analysis. Table 6 shows the overall performance of MolSearch under different combination of hyper-parameters for

Setting	GSK3β+JNK3				GSK3β+JNK3+QED+SA			
K, N	SR	Nov	Div	PM	SR	Nov	Div	PM
3, 5	1.00	0.72	0.83	0.60	1.00	0.77	0.82	0.63
3, 10	1.00	0.78	0.83	0.65	1.00	0.82	0.81	0.67
3, 20	1.00	0.77	0.83	0.64	1.00	0.80	0.81	0.65
5, 5	1.00	0.76	0.83	0.63	1.00	0.79	0.82	0.65
5, 10	1.00	0.77	0.83	0.64	1.00	0.81	0.81	0.66
5, 20	**1.00**	**0.80**	**0.83**	**0.66**	1.00	0.82	0.81	0.67
7, 5	1.00	0.76	0.83	0.63	1.00	0.79	0.81	0.64
7, 10	1.00	0.78	0.83	0.65	**1.00**	**0.84**	**0.81**	**0.68**
7, 20	1.00	0.80	0.83	0.66	1.00	0.82	0.81	0.67

Table 6: Performance of MolSearch under different hyper-parameters for two generation settings.

Setting	GSK3β+JNK3			GSK3β+JNK3+QED+SA		
N/K	3	5	7	3	5	7
5	9,373	14,776	18,077	3,543	5,463	6,773
10	13,960	21,982	28,659	5,499	7,772	10,295
20	16,085	29,912	43,778	6,233	10,406	13,884

Table 7: Number of generated molecules by MolSearch under different hyper-parameters for two generation settings.

two generation settings. Table 7 shows the number of valid molecules corresponding to Table 6. We observe that the performance is not very different regarding different hyper-parameters, but rather the number of generated molecules are highly affected by these hyper-parameters. Because the maximum number of child nodes and simulations rounds actually increases the search range such that more molecules can be found along the way.

4.3 Discussion

The extensive experiments of MolSearch demonstrated that given proper design and sufficient information, search-based method is also able to find molecules that satisfy multiple property requirements simultaneously with performance comparable to advanced methods using deep learning and reinforcement learning, while being much more time efficient. For MolSearch, the benefits comes from several aspects. For example, the two-stage design increases the novelty of generated molecules; Treating different objectives separately improves the diversity of the generated molecules; Fragment-based actions and starting from existing molecules maintain the synthesis abilities and drug-likeness of generated molecules. Additional to properties in benchmark tasks, MolSearch can be easily adopted into real drug discovery projects targeting other objectives. For example, replacing GSK3β and JNK3 scoring models with COVID related predictors [21] may lead to the identification of novel and synthesizable compounds. Properties other than QED/SA, such as solubility and ADMET properties can also be included to search for more promising candidates.

MolSearch also has its own limitations. First, the bioactivity scores drop in LEAD-MCTS compared to HIT-MCTS although they are still significantly higher than start molecules (Figure 4a). It is because the child nodes only need to maintain bioactivity score above 0.5 threshold in LEAD-MCTS in exchange of higher non-bioacitvity scores. It is possible to improve the situation by setting more strict constraint during LEAD-MCTS. Second, the evaluation metrics are calculated based on unique molecules found in the search process, however, we do observe the molecules generated in LEAD-MCTS often contain many duplicates and thus causes redundancy. Third, for objectives that has relatively clear structural requirement, e.g., binding to a specific protein target, MolSearch is able to find desired molecules. However, if the objective is not sensitive to structure changes, i.e., regulation effects of multiple genes, then MolSearch, or any other related methods works less effectively. Last but not least, the scoring models are not perfect in reality since they also come form machine learning models, which may affect the quality of generation results.

ACKNOWLEDGMENT

This research is funded in part by National Science Foundation IIS-1749940 (JZ), Office of Naval Research N00014-20-1-2382 (JZ), National Institute of Health R01GM134307 (JZ, BC). This work is also supported via computational resources and services provided by the Institute for Cyber-Enabled Research (ICER) at MSU.

REFERENCES

[1] Sungsoo Ahn, Junsu Kim, Hankook Lee, and Jinwoo Shin. 2020. Guiding deep molecular optimization with genetic exploration. *arXiv preprint arXiv:2007.04897* (2020).

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2 (2002), 235–256.

[3] Mahendra Awale, Jérôme Hert, Laura Guasch, Sereina Riniker, and Christian Kramer. 2021. The Playbooks of Medicinal Chemistry Design Moves. *Journal of Chemical Information and Modeling* 61, 2 (2021), 729–742.

[4] Dávid Bajusz, Anita Rácz, and Károly Héberger. 2015. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics* 7, 1 (2015), 1–13.

[5] Richard Bellman. 1957. A Markovian decision process. *Journal of mathematics and mechanics* 6, 5 (1957), 679–684.

[6] G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. 2012. Quantifying the chemical beauty of drugs. *Nature chemistry* 4, 2 (2012), 90–98.

[7] Benjamin E Blass. 2015. *Basic principles of drug discovery and development.* Elsevier.

[8] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[9] Darko Butina. 1999. Unsupervised data base clustering based on daylight's fingerprint and Tanimoto similarity: A fast and automated way to cluster small and large data sets. *Journal of Chemical Information and Computer Sciences* 39, 4 (1999), 747–750.

[10] Weizhe Chen and Lantao Liu. 2021. Pareto monte carlo tree search for multi-objective informative planning. *arXiv preprint arXiv:2111.01825* (2021).

[11] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. 2018. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786* (2018).

[12] Nicola De Cao and Thomas Kipf. 2018. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973* (2018).

[13] Daniel C Elton, Zois Boukouvalas, Mark D Fuge, and Peter W Chung. 2019. Deep learning for molecular design—a review of the state of the art. *Molecular Systems Design & Engineering* 4, 4 (2019), 828–849.

[14] Peter Ertl and Ansgar Schuffenhauer. 2009. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics* 1, 1 (2009), 1–11.

[15] Sylvain Gelly and David Silver. 2011. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence* 175, 11 (2011), 1856–1875.

[16] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science* 4, 2 (2018), 268–276.

[17] Jameed Hussain and Ceara Rea. 2010. Computationally efficient algorithm to identify matched molecular pairs (MMPs) in large data sets. *Journal of chemical information and modeling* 50, 3 (2010), 339–348.

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

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

[20] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2020. Multi-objective molecule generation using interpretable substructures. In *International Conference on Machine Learning*. PMLR, 4849–4859.

[21] Govinda B Kc, Giovanni Bocci, Srijan Verma, Md Mahmudulla Hassan, Jayme Holmes, Jeremy J Yang, Suman Sirimulla, and Tudor I Oprea. 2021. A machine learning platform to estimate anti-SARS-CoV-2 activities. *Nature Machine Intelligence* 3, 6 (2021), 527–535.

[22] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*. Springer, 282–293.

[23] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar variational autoencoder. In *International Conference on Machine Learning*. PMLR, 1945–1954.

[24] Yibo Li, Liangren Zhang, and Zhenming Liu. 2018. Multi-objective de novo drug design with conditional graph generative model. *Journal of cheminformatics* 10, 1 (2018), 1–24.

[25] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. 2021. GraphDF: A discrete flow model for molecular graph generation. *arXiv preprint arXiv:2102.01189* (2021).

[26] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, et al. 2019. ChEMBL: towards direct deposition of bioassay data. *Nucleic acids research* 47, D1 (2019), D930–D940.

[27] AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alán Aspuru-Guzik. 2019. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. *arXiv preprint arXiv:1909.11655* (2019).

[28] Anand A Rajasekar, Karthik Raman, and Balaraman Ravindran. 2020. Goal directed molecule generation using monte carlo tree search. *arXiv preprint arXiv:2010.16399* (2020).

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

[30] Benjamin Sanchez-Lengeling, Carlos Outeiral, Gabriel L Guimaraes, and Alan Aspuru-Guzik. 2017. Optimizing distributions over molecular space. An objective-reinforced generative adversarial network for inverse-design chemistry (ORGANIC). (2017).

[31] Gisbert Schneider and Uli Fechner. 2005. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery* 4, 8 (2005), 649–663.

[32] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2015), 148–175.

[33] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. 2020. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382* (2020).

[34] Gregory Sliwoski, Sandeepkumar Kothiwale, Jens Meiler, and Edward W Lowe. 2014. Computational methods in drug discovery. *Pharmacological reviews* 66, 1 (2014), 334–395.

[35] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

[36] Weijia Wang and Michele Sebag. 2012. Multi-objective monte-carlo tree search. In *Asian conference on machine learning*. PMLR, 507–522.

[37] Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. 2021. Mars: Markov molecular sampling for multi-objective drug discovery. *arXiv preprint arXiv:2103.10432* (2021).

[38] Xiufeng Yang, Jinzhe Zhang, Kazuki Yoshizoe, Kei Terayama, and Koji Tsuda. 2017. ChemTS: an efficient python library for de novo molecular generation. *Science and technology of advanced materials* 18, 1 (2017), 972–976.

[39] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. 2018. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473* (2018).

[40] Wenbo Yu and Alexander D MacKerell. 2017. Computer-aided drug design methods. In *Antibiotics*. Springer, 85–106.

[41] Chengxi Zang and Fei Wang. 2020. MoFlow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 617–626.

[42] Yi Zhang. 2021. An In-depth Summary of Recent Artificial Intelligence Applications in Drug Design. *arXiv preprint arXiv:2110.05478* (2021).

[43] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. 2019. Optimization of molecules via deep reinforcement learning. *Scientific reports* 9, 1 (2019), 1–10.