Reinforced Genetic Algorithm for Structure-based Drug Design

Anonymous Author(s)
Affiliation
Address
email

Abstract

Structure-based drug design (SBDD) aims to discover drug candidates by finding molecules (ligands) that bind tightly to a disease-related protein (targets), which is the primary approach to computer-aided drug discovery. Recently, applying deep generative models for three-dimensional (3D) molecular design conditioned on protein pockets to solve SBDD has attracted much attention, but their formulation as probabilistic modeling often leads to unsatisfactory optimization performance. On the other hand, traditional combinatorial optimization methods such as genetic algorithms (GA) have demonstrated state-of-the-art performance in various molecular optimization tasks. However, they do not utilize protein target structure to inform design steps but rely on a random-walk-like exploration, which leads to unstable performance and no knowledge transfer between different tasks despite the similar binding physics. To achieve a more stable and efficient SBDD, we propose Reinforced Genetic Algorithm (`RGA`) that uses neural models to prioritize the profitable design steps and suppress random-walk behavior. The neural models take the 3D structure of the targets and ligands as inputs and are pre-trained using native complex structures to utilize the knowledge of the shared binding physics from different targets and then fine-tuned during optimization. We conduct thorough empirical studies on optimizing binding affinity to various disease targets and show that `RGA` outperforms the baselines in terms of docking scores and is more robust to random initializations. The ablation study also indicates that the training on different targets helps improve the performance by leveraging the shared underlying physics of the binding processes.

1 Introduction

Rapid drug discovery that requires less time and cost is of significant interest in pharmaceutical science. Structure-based drug design (SBDD) [1] that leverages the three-dimensional (3D) structures of the disease-related proteins to design drug candidates is one primary approach to accelerate the drug discovery processes with physical simulation and data-driven modeling. According to the lock and key model [2], the molecules that bind tighter to a disease target are more likely to expose bioactivity against the disease, which has been verified experimentally [3]. As AlphaFold2 has provided accurate predictions to most human proteins [4, 5], SBDD has a tremendous opportunity to discover new drugs for new targets that we cannot model before [6].

SBDD could be formulated as an optimization problem where the objective function is the binding affinity estimated by simulations such as docking [2]. The most widely used design method is virtual screening, which exhaustively investigates every molecule in a library and ranks them. Lyu et al. successfully discovered new chemotypes for AmpC β-lactamase and the D_4 dopamine receptor by studying hundreds of millions of molecules with docking simulation [7]. However, the number of

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

the drug-like molecules is large as estimated to be 10^{60} [1], and it is computationally prohibitive to screen all of the possible molecules. Though machine learning approaches have been developed to accelerate screening [8, 9], it is still challenging to screen large enough chemical space within the foreseeable future.

Instead of naively screening a library, designing drug candidates with generative models has been highlighted as a promising strategy, exemplified by [10, 11]. This class of methods models the problem as the generation of ligands conditioned on the protein pockets. However, as generative models are trained to learn the distribution of known active compounds, they tend to produce molecules similar to training data [12], which discourages finding novel molecules and leads to unsatisfactory optimization performance.

A more straightforward solution is a combinatorial optimization algorithm that searches the implicitly defined discrete chemical space. As shown in multiple standard molecule optimization benchmarks [13, 14], combinatorial optimization methods, especially genetic algorithms (GA) [15, 16], often perform better than deep generative models. The key to superior performance is GA's action definition. Specifically, in each generation (iteration), GA maintains a population of possible candidates (a.k.a. parents) and conducts the crossover between two candidates and mutation from a single candidate to generate new offspring. These two types of actions, crossover and mutation, enable global and local traversal over the chemical space, allowing a thorough exploration and superior optimization performance.

However, most GAs select mutation and crossover operations randomly [15], leading to significant variance between independent runs. Especially in SBDD, when the oracle functions are expensive molecular simulations, it is resource-consuming to ensure stability by running multiple times. Further, most current combinatorial methods are designed for general-purpose molecular optimization and simply use a docking simulation as an oracle. It is challenging to leverage the structure of proteins in these methods, and we need to start from scratch whenever we change a protein target, even though the physics of ligand-protein interaction is shared. Ignoring the shared information across tasks leads to unnecessary exploration steps and, thus, demands for many more oracle calls, which require expensive and unnecessary simulations [17].

To overcome these issues in the GA method, we propose Reinforced Genetic Algorithm (`RGA`), which attempts to reformulate an evolutionary process as a Markov decision process and uses neural networks to make informed decisions and suppress the random-walk behavior. Specifically, we utilize an E(3)-equivariant neural network [18] to choose parents and mutation types based on the 3D structure of the ligands and proteins. The networks are pre-trained with various native complex structures to utilize the knowledge of the shared binding physics between different targets and then fine-tuned with a reinforcement learning algorithm during optimizations. We test `RGA`'s performance with various disease-related targets, including the main protease of SARS-CoV-2.

The main contributions of this paper can be summarized as follows:

- We propose an evolutionary Markov decision process (EMDP) that reformulates an evolutionary process as a Markov decision process, where the state is a population of molecules instead of a single molecule (Section 3.2).
- We show the first successful attempt to use a neural model to guide the crossover and mutation operations in a genetic algorithm to suppress random-walk behavior and explore the chemical space intelligently (Section 3.3).
- We present a structure-based de novo drug design algorithm that outperforms baseline methods consistently through thorough empirical studies on optimizing binding affinity by leveraging the underlying binding physics (Section 4).

2 Related Works

We will discuss the related works on methods of drug design and discuss the advantage of the proposed method over the existing works.

General Molecular Design. Molecular generation methods offer a promising direction for the automated design of molecules with desired pharmaceutical properties such as synthesis accessibility and drug-likeliness. Based on how to generate or search molecules, these approaches can be

categorized into two types, (1) deep generative models (DGMs) imitate the molecular data distribution, including variational autoencoder (VAE) [19, 20], generative adversarial network (GAN) [21, 22], normalizing flow model [23, 24], energy based model [25]; and (2) combinatorial optimization methods directly search over the discrete chemical space, including genetic algorithm (GA) [15, 26, 27], reinforcement learning approaches (RL) [28, 29, 30, 31, 32], Bayesian optimization (BO) [33], Markov chain Monte Carlo (MCMC) [34, 35, 36] and gradient ascent [37, 38].

General molecular design algorithms often use general black-box oracle functions, and some are only tested with trivial or self-designed oracles. For example, using penalized octanol-water partition coefficient (LogP) as the oracle function, it grows monotonically with the number of carbons, and thus there exists a trivial policy to optimize LogP. These oracles do not reflect the challenges of real drug discovery, and those algorithms have limited value for pharmaceutical discovery. Recent works are optimizing docking scores to simulate a more realistic discovery scenario [39, 40, 17, 41], same as our work. However, they are still ignoring the information in the given protein structures that could potentially accelerate the design process. However, the extension to leveraging the structural knowledge is nontrivial.

Structure-based Drug Design. Structure-based drug design (SBDD) could utilize the structural information to guide the design of molecules, which are potentially more efficient in drug discovery tasks but poses additional challenges of how to leverage the structures. Since early 1990s, various SBDD algorithms have been proposed, mostly based on combinatorial optimization algorithms such as tree search [42, 43, 44] and evolutionary algorithms [45, 46]. Those methods typically optimize the ligands in the pockets according to a physical model characterizing the binding affinity. For example, RASSE [42] used a force-field-like scoring function [47] to evaluate the partial solutions within a tree search. However, obtaining a fast and accurate model to quantify binding free energy itself is still an unsolved challenge.

Recently, generative modeling of 3D molecules conditioned on protein targets is attracting more attention [10, 11]. Similar to DGMs in general molecular design, those methods learn the atom's compositional and spatial distribution of native structure of protein-ligand complexes with neural models and design new ligands by complete the complex structure given targets. Deep generative models are end-to-end and data-driven thus surpass the necessity of understanding the physics of interaction. However, as the training objective is to learn the distribution of known active compounds, the models tend to produce molecules close to the training set [12], which is undesired in terms of patentability and leads to unsatisfactory optimization performance.

3 Method

In this paper, we focus on structure-based drug design. The goal is to design drug molecules (a.k.a. ligands) that could bind tightly with the disease-related proteins (a.k.a. targets). Given the 3D structures of the target proteins, including binding site information, docking is a popular computational method for assessing the binding affinity, which can be roughly retrieved as the free energy changes during the binding processes. We present a variant of genetic algorithm that is guided by reinforcement learning and a docking oracle. Next, we will first describe the general evolutionary process used in genetic algorithms (Section 3.1); Then we will present how to model this evolutionary process as a Markov decision process (MDP) where RL framework can be constructed (Section 3.2); After that, we describe the detailed implementation of this MDP framework using multiple policy networks (Section 3.3).

3.1 Evolutionary Process

In this section, we introduce the primary setting of the evolutionary processes. With both optimization performance and synthetic accessibility taken into account [48, 14], we follow the action settings in Autogrow 4.0 [16]. It demonstrated superior performance over other GA variants in the empirical validation of structure-based drug design [16], and its mutation actions originated from chemical reactions so that the designed molecules are more likely to be synthesizable. Specifically, an evolutionary processes starts by randomly sampling a *population* of drug candidates from a library. In each *generation* (iteration), it carries out (i) *crossover* between parents selected from the last generation, and (ii) *mutation* on a single child to obtain the offspring pool. An illustration of both

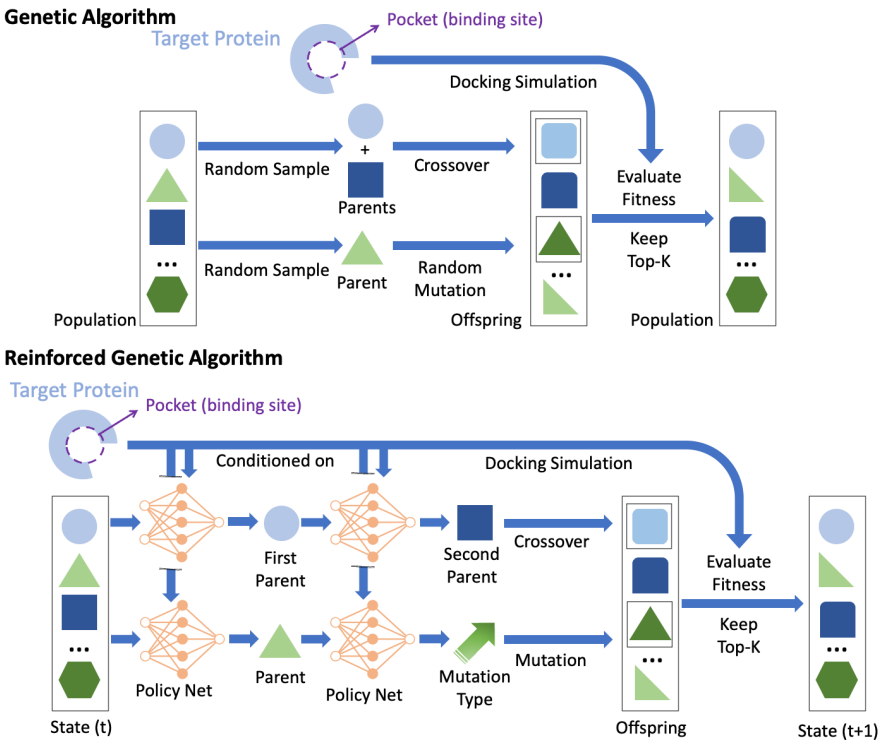


Figure 1: We illustrate one generation (iteration) of GA (top) and RGA pipeline (bottom). Specifically, we train policy networks that take the target and ligand as input to make informed choices on parents and mutation types in RGA.

crossover and mutation operations is available in Appendix. Note that we only adopted the action settings from Autogrow 4.0, without using other tricks such as elitism.

Crossover, also called recombination, combines the structure of two parents to generate new children. Following Autogrow 4.0 [16], we select two parents from the last generation and search for the largest common substructure shared between them. Then we generate two children by randomly switching their decorating moieties, i.e., the side chains attached to the common substructure.

Mutation operates on a single parent molecule and modifies its structure slightly. Following Autogrow 4.0 [16], we adopt transformations based on chemical reactions. Unlike naively defined atom-editing actions, mutation steps based on chemical reactions could ensure all modification is reasonable in reality, leading to a larger probability of designing synthesizable molecules. We included two types of chemical reactions: uni-molecular reactions, which only require one reactant, and bi-molecular reactions, which require two reactants. While uni-molecular reactions could be directly applied to the parent, we sample a purchasable compound to react with the parent when conducting a bi-molecular reaction. In both cases, the parent serves as one reactant, and we use the main product as the child molecule. We use the chemical reactions from [16], which was originally from [46, 49].

Evolution. At the t-th generation (iteration), given a population of molecules denoted as $\mathcal{S}^{(t)}$, we generate an offspring pool denoted as $\mathcal{Q}^{(t)}$ by applying crossover and mutation operations. Then we filter out the ones with undesirable physical and chemical properties (e.g., poor solubility, high toxicity) in the offspring pool and select the most promising K to form the next generation pool ($\mathcal{S}^{(t+1)}$).

3.2 Evolutionary Markov Decision Process

Next we propose the evolutionary Markov decision process (EMDP) that formulates an evolutionary process of genetic algorithm as a Markov decision process (MDP). The primary purpose is to utilize reinforcement learning algorithms to train networks to inform the decision steps to replace random selections. Taking a generation as a state, Markov property that requires $P(\mathcal{S}^{(t+1)}|\mathcal{S}^{(1)}, \cdots, \mathcal{S}^{(t)}) =$

$P(\mathcal{S}^{(t+1)}|\mathcal{S}^{(t)})$ is naturally satisfied by the evolutionary process described above, where $\mathcal{S}^{(t)}$ denotes the state at the t-th generation, which is the population of ligands. We use X to denote a ligand. We elaborate essential components for Markov decision process as follows, and the EMDP pipeline is illustrated in Figure 1.

State Space. We define the population at the t-step generation, $\mathcal{S}^{(t)}$, in the evolutionary process as the state at the t-step in an EMDP. A state includes population of candidate molecules (i.e., ligand, denoted X) and their 3D poses docked to the target, fully observable to the RL agent. At the beginning of the EMDP, we randomly select a population of candidate molecules and use docking simulation to yield their 3D poses as the initial state.

Action Space. The actions in an EMDP are to conduct the two evolutionary steps: crossover and mutation, in a population. For each evolutionary step, we need two actions to conduct it. Concretely, crossover ($X^{\text{parent 1}}_{\text{crossover}}, X^{\text{parent 2}}_{\text{crossover}} \xrightarrow{\text{crossover}} X^{\text{child 1}}_{\text{crossover}}, X^{\text{child 2}}_{\text{crossover}}$) can be divided to:

1. select the first candidate ligand $X^{\text{parent 1}}_{\text{crossover}}$ from the current state (population $\mathcal{S}^{(t)}$);
2. conditioned on the first selected candidate $X^{\text{parent 1}}_{\text{crossover}}$, select the second candidate ligand $X^{\text{parent 2}}_{\text{crossover}}$ from the remaining candidate ligand set $\mathcal{S}^{(t)} - \{X^{\text{parent 1}}_{\text{crossover}}\}$ and apply crossover (Section 3.1) to them.

Mutation ($X^{\text{parent}}_{\text{mutation}} \xrightarrow{\text{mutated by } \xi} X^{\text{child}}_{\text{mutation}}$) can be divided to:

1. select the candidate ligand $X^{\text{parent}}_{\text{mutated}}$ to be mutated from the current state (population $\mathcal{S}^{(t)}$);
2. conditioned on the selected candidate ligand $X^{\text{parent}}_{\text{mutated}}$, select the reaction ξ from the reaction set $\mathcal{R}$ and apply it to $X^{\text{parent}}_{\text{mutated}}$.

As applying the crossover and mutation steps are deterministic, the actions in an EMDP focus on selecting parents and mutation types. Upon finish the action, we could obtain the offspring pool, $\mathcal{Q}^{(t)}$.

State Transition Dynamics. The state transition in an EMDP is identical to the evolution in an evolutionary process. Once we finish the actions and obtain the offspring pool, $\mathcal{Q}^{(t)} = \{X^{\text{child 1}}, X^{\text{child 2}}, \cdots\}$, we apply molecular quality filters to filter out the ones unlikely to be drug and then select the most promising K to form the parent set for the next generation ($\mathcal{S}^{(t+1)}$).

Reward. We define the reward as the binding affinity change (docking score). The actions leading to stronger binding score would be prioritized. As there is no "episode" concept in an EMDP, we treat every step equally.

3.3 Target-Ligand Policy Network

To utilize molecular structures' translational and rotational invariance, we adopt equivariance neural networks (ENNs) [18] as the target-ligand policy neural networks to select the actions in both mutation and crossover steps. Each ligand has a 3D pose that binds to the target protein, and the complex serves as the input of ENN.

Specifically, we want to model a 3D graph $\mathcal{Y}$, which can be ligand, target, or target-ligand complex. The input feature can be described as $\mathcal{Y} = (\mathcal{A}, \mathcal{Z})$, where $\mathcal{A}$ represents atoms' categories (the vocabulary set $\mathcal{V} = \{H, C, O, N, \cdots\}$) and $\mathcal{Z}$ represents 3D coordinates of the atoms. Suppose $\mathbf{D} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the embedding matrix of all the categories of atoms in a vocabulary set $\mathcal{V}$, is randomly initialized and learnable, d is the hidden dimension in ENN. Each kind of atom corresponds to a row in $\mathbf{D}$. We suppose there are N atoms, and each atom corresponds to a node in the 3D graph. Node embeddings at the l-th layer are denoted as $\mathbf{H}^{(l)} = \{\mathbf{h}_i^{(l)}\}_{i=1}^N$, where $l = 0, 1, \cdots, L$, L is number of layers in ENN. The initial node embedding $\mathbf{h}_i^{(0)} = \mathbf{D}^\top \mathbf{a}_i \in \mathbb{R}^d$ embeds the i-th node, where $\mathbf{a}_i$ is one-hot vector that encode the category of the i-th atom. Coordinate embeddings at the l-th layer are denoted $\mathbf{Z}^{(l)} = \{\mathbf{z}_i^{(l)}\}_{i=1}^N$. The initial coordinate embeddings $\mathbf{Z}^{(0)} = \{\mathbf{z}_i\}_{i=1}^N$ are the real 3D coordinates of all the nodes. The following equation defines the feedforward rules of ENN, for

$i, j = 1, \cdots, N,\ i \neq j,\ l = 0, 1, \cdots, L-1$, we have

$$
\begin{aligned}
&\mathbf{w}_{ij}^{(l+1)} = \text{MLP}_e\Big(\mathbf{h}_i^{(l)} \oplus \mathbf{h}_j^{(l)} \oplus ||\mathbf{z}_i^{(l)} - \mathbf{z}_j^{(l)}||_2^2\Big) \ \in \mathbb{R}^d, \qquad \mathbf{v}_i^{(l+1)} = \sum_{j=1, j\neq i}^{N} \mathbf{w}_{ij}^{(l+1)} \ \in \mathbb{R}^d, \\
&\mathbf{z}_i^{(l+1)} = \mathbf{z}_i^{(l)} + \sum_{j=1, j\neq i}^{N} \Big(\mathbf{z}_i^{(l)} - \mathbf{z}_j^{(l)}\Big)\text{MLP}_x\Big(\mathbf{w}_{ij}^{(l)}\Big) \in \mathbb{R}^3, \quad \mathbf{h}_i^{(l+1)} = \text{MLP}_h\Big(\mathbf{h}_i^{(l)} \oplus \mathbf{v}_i^{(l+1)}\Big) \in \mathbb{R}^d, \\
&\mathbf{h}_{\mathcal{Y}} = \sum_{i=1}^{N} \mathbf{h}_i^{(L)} \ \in \mathbb{R}^d \qquad \Longrightarrow \ \underline{\mathbf{h}_{\mathcal{Y}} = \text{ENN}(\mathcal{Y})}
\end{aligned}
\tag{1}
$$

where $\oplus$ denotes the concatenation of vectors; $\text{MLP}_e(\cdot) : \mathbb{R}^{2d+1} \to \mathbb{R}^d; \text{MLP}_x(\cdot) : \mathbb{R}^d \to \mathbb{R}; \text{MLP}_h(\cdot) : \mathbb{R}^{2d} \to \mathbb{R}^d$ are all two-layer multiple layer perceptrons (MLPs) with Swish activation in the hidden layer [50]. At the l-th layer, $\mathbf{w}_{ij}^{(l)}$ represents the message vector for the edge from node i to node j; $\mathbf{v}_i^{(l)}$ represents the message vector for node i, $\mathbf{z}_i^{(l)}$ is the position embedding for node i; $\mathbf{h}_i^{(l)}$ is the node embedding for node i. $\mathbf{H}^{(L)} = [\mathbf{h}_1^{(L)}, \cdots, \mathbf{h}_N^{(L)}]$ are the node embeddings of the L-th (last) layer. We aggregate them using sum function as readout function to obtain a representation of the 3D graph, denoted $\mathbf{h}_{\mathcal{Y}}$. The whole process is written as $\mathbf{h}_{\mathcal{Y}} = \text{ENN}(\mathcal{Y})$.

Crossover Policy Network. We design two policy networks for two corresponding actions in a crossover, as mentioned in Section 3.2. (1) the first action in crossover operation is to select the first parent ligand $X_{\text{crossover}}^{\text{parent 1}}$ from the population $\mathcal{S}^{(t)}$. Similar to the first action in mutation operation, we obtain a valid probability distribution over all the available ligands based on target-ligand complex as input feature and ENN as the neural network architecture, the selection probability of the ligand $X_{\text{crossover}}^{\text{parent 1}} \in \mathcal{S}^{(t)}$ is $p_{\text{crossover}}^{(1)}(X_{\text{crossover}}^{\text{parent 1}}|\mathcal{S}^{(t)}) = \frac{\exp\left(\text{MLP}(\mathbf{h}_{\mathcal{T}\&X_{\text{crossover}}^{\text{parent 1}}})\right)}{\sum_{X' \in \mathcal{S}^{(t)}} \exp\left(\text{MLP}(\mathbf{h}_{\mathcal{T}\&X'})\right)}$, where $\mathcal{T}$ and X denotes target and ligand (including 3D pose), respectively, $\mathcal{T}\&X$ denotes target-ligand complex. (2) The second action is to select the second parent ligand conditioned on the first parent ligand selected in the first action. Specifically, for ligand in the remaining population set, we concatenate the ENN's embedding of the target, first parent ligand $X_{\text{crossover}}^{\text{parent 1}}$ and the second parent ligand $X_{\text{mutation}}^{\text{child}}$, and feed it into an MLP to estimate a scalar as an unnormalized probability. The unnormalized probabilities for all the ligands in the remaining population set are normalized via softmax function, i.e., $p_{\text{crossover}}^{(2)}(X_{\text{crossover}}^{\text{parent 2}}|X_{\text{crossover}}^{\text{parent 1}}, \mathcal{S}^{(t)}) = \text{Softmax}\big\{\text{MLP}(\mathbf{h}_{\mathcal{T}} \oplus \mathbf{h}_{X_{\text{crossover}}^{\text{parent 1}}} \oplus \mathbf{h}_{X_{\text{crossover}}^{\text{parent 2}}}), \cdots, \big\}_{X_{\text{crossover}}^{\text{parent 2}} \in \mathcal{S}^{(t)} - \{X_{\text{crossover}}^{\text{parent 1}}\}}$. Given two parents ligands, crossover finds the largest substructure that the two parent compounds share and generates a child by combining their decorating moieties. Thus, the generation of child ligands are deterministic, and the probability of the generated ligands $X_{\text{crossover}}^{\text{child}}$ is

$$
\begin{aligned}
&p_{\text{crossover}}(X_{\text{crossover}}^{\text{child 1}}, X_{\text{crossover}}^{\text{child 2}}|\mathcal{S}^{(t)}) = p_{\text{crossover}}(X_{\text{crossover}}^{\text{parent 1}}, X_{\text{crossover}}^{\text{parent 2}}|\mathcal{S}^{(t)}) \\
&= p_{\text{crossover}}^{(1)}(X_{\text{crossover}}^{\text{parent 1}}|\mathcal{S}^{(t)}) \cdot p_{\text{crossover}}^{(2)}(X_{\text{crossover}}^{\text{parent 2}}|X_{\text{crossover}}^{\text{parent 1}}, \mathcal{S}^{(t)}).
\end{aligned}
\tag{2}
$$

Mutation Policy Network. We design two policy networks for two corresponding actions in mutation, as mentioned in Section 3.2. (1) the first action in mutation operation is to select a candidate ligand to be mutated from population $\mathcal{S}^{(t)}$. It models the 3D target-ligand complex to learn if there is improvement space in the current complex. Formally, we obtain a valid probability distribution over all the available ligands based on target-ligand complex as input feature and ENN as neural architecture, the selection probability of the ligand $X_{\text{mutation}}^{\text{parent}} \in \mathcal{S}^{(t)}$ is $p_{\text{mutation}}^{(1)}(X_{\text{mutation}}^{\text{parent}}|\mathcal{S}^{(t)}) = \frac{\exp\left(\text{MLP}(\mathbf{h}_{\mathcal{T}\&X_{\text{mutation}}^{\text{parent}}})\right)}{\sum_{X' \in \mathcal{S}^{(t)}} \exp\left(\text{MLP}(\mathbf{h}_{\mathcal{T}\&X'})\right)}$, where $\mathcal{T}\&X$ denotes target-ligand complex, $\mathbf{h}_{\mathcal{T}\&X} = \text{ENN}(\mathcal{T}\&X)$ represents the ENN's embedding of target-ligand complex. (2) The second action is to select the SMARTS reaction from the reaction set conditioned on the selected ligand in the first action. Specifically, for each reaction, we generate the new ligand $X_{\text{mutation}}^{\text{child}}$, then obtain the embedding of target, first ligand $X_{\text{mutation}}^{\text{parent}}$ and the new ligand $X_{\text{mutation}}^{\text{child}}$ through ENN, concatenate these three embeddings and feed it into a MLP to estimate a scalar as unnormalized probability. The unnormalized probabilities for all the reactions are normalized via softmax func-

tion, i.e., $p^{(2)}_{\text{mutation}}(\xi|X^{\text{parent}}_{\text{mutation}}, \mathcal{S}^{(t)}) = \text{Softmax}\big\{\text{MLP}(\mathbf{h}_{\mathcal{T}} \oplus \mathbf{h}_{X^{\text{parent}}_{\text{mutation}}} \oplus \mathbf{h}_{X^{\text{child}}_{\text{mutation}}}]), \cdots, \big\}_{\xi \in \mathcal{R}}$, where $X^{\text{parent}}_{\text{mutation}} \xrightarrow{\text{mutated by } \xi} X^{\text{child}}_{\text{mutation}}$, $\mathcal{R}$ is the reaction set. The probability of the generated ligand $X^{\text{child}}_{\text{mutation}}$ is

$$p_{\text{mutation}}(X^{\text{child}}_{\text{mutation}}|\mathcal{S}^{(t)}) = p^{(1)}_{\text{mutation}}(X^{\text{parent}}_{\text{mutation}}|\mathcal{S}^{(t)}) \cdot p^{(2)}_{\text{mutation}}(\xi|X^{\text{parent}}_{\text{mutation}}, \mathcal{S}^{(t)}). \tag{3}$$

Policy Gradient. We leverage policy gradient to train the target-ligand policy neural network. Specifically, we consider maximizing the expected reward as objective via REINFORCE [28],

$$\max\ \mathbb{E}_{X \sim p(X|\mathcal{S}^{(t)})}\big[\text{Reward}(X)\big], \tag{4}$$

where $p(X)$ is defined in Equation (2) and (3) for crossover and mutation, respectively. The whole pipeline is illustrated in Figure 1. To provide a warm start and leverage the structural information, we pretrain the ENNs on 3D target-ligand binding affinity prediction task, where the inputs are the target-ligand complexes, and the outputs are their binding affinity.

4 Experiment

In this section, we briefly describe the experimental setup and results. The Appendix includes more details, including software configuration, implementation details, dataset description & processing, hyperparameter tuning, code repository, ablation study, and additional experimental results.

4.1 Experimental Setup

Docking Simulation. We adopt AutoDock Vina [51] to evaluate the binding affinity. The docking score estimated by AutoDock Vina is called Vina score and roughly characterizes the free energy changes of binding processes in kcal/mol. Thus lower Vina score means a stronger binding affinity between the ligand and target. We picked various disease-related proteins, including G-protein coupling receptors (GPCRs) and kinases from DUD-E [52] and the SARS-CoV-2 main protease [53] as targets. Please see the Appendix for more information.

Baselines. The baseline methods cover traditional brute-force search methods (Screening), deep generative models (JTVAE and Gen3D), genetic algorithm (GA+D, graph-GA, Autogrow 4.0), reinforcement learning methods (MolDQN, RationaleRL, REINVENT, GEGL), and MCMC method (MARS). Gen3D and Autogrow 4.0 are structure-based drug design methods, while others are general-purpose molecular design methods. Although methods explicitly utilizing target structures are relatively few, we add general-purpose molecular design methods optimizing the same docking oracle scores as ours, which is a common use case, as baselines [15, 14]. Concretely, **Screening** mimics high throughput screening via sampling from ZINC database randomly; **JTVAE** (Junction Tree Variational Auto-Encoder) [20] uses a Bayesian optimization on the latent space to indirectly optimize molecules; **Gen3D** [10] is an auto-regressive generative model that grows 3D structures atom-wise inside the binding pocket; **GA+D** [26] represents molecule as SELFIES string [54] and uses genetic algorithm enhanced by a discriminator neural network; **Graph-GA** [15] conduct genetic algorithm on molecular graph representation; **Autogrow 4.0** [16] is the state-of-the-art genetic algorithm in structure-based drug design; **MolDQN** (Molecule Deep Q-Network) [30] leverages deep Q-value learning to grow molecules atom-wisely; **RationaleRL** [31] uses rationale (e.g., functional groups or subgraphs) as the building block and a policy gradient method to guide the training of graph neural network-based generator; **REINVENT** [28] represent molecules as SMILES string and uses policy gradient based reinforcement learning methods to guide the training of the RNN generator; **GEGL** (genetic expert-guided learning) [32] uses LSTM guided by reinforcement learning to imitate the GA exploration; **MARS** (Markov Molecule Sampling) [35] leverages Markov chain Monte Carlo sampling (MCMC) with adaptive proposal and annealing scheme to search chemical space. To conduct a fair comparison, we limit the number of oracle calls to 1,000 times for each method. Implementation details of all the baseline methods are available in Appendix.

Dataset: we randomly select molecules from ZINC [55] database (around 250 thousands drug-like molecules) as 0-th generation of the genetic algorithms (`RGA`, Autogrow 4.0, GA+D). ZINC also serves as the training data for pretraining the model in JTVAE, REINVENT, RationaleRL, etc. We adopt CrossDocked2020 [56] dataset that contains around 22 million ligand-protein complexes as the training data for pretraining the policy neural networks, as mentioned in Section 3.3. More descriptions are available in Appendix.

Table 1: The summarized performance of different methods. The mean and standard deviation across targets are reported. Arrows (↑, ↓) indicate the direction of better performance. For each metric, the best method is underlined and the top-3 methods are bolded. RGA-pretrain and RGA-KT are two variants of RGA that without pretraining and without training on different target proteins, respectively.

Method	TOP-100↓	TOP-10↓	TOP-1↓	Nov↑	Div↑	QED↑	SA↓
screening	-9.351±0.643	-10.433±0.563	-11.400±0.630	0.0±0.0%	0.858±0.005	0.678±0.022	2.689±0.077
MARS	-7.758±0.612	-8.875±0.711	-9.257±0.791	100.0±0.0%	**0.877±0.001**	**0.709±0.008**	**2.450±0.034**
MolDQN	-6.287±0.396	-7.043±0.487	-7.501±0.402	100.0±0.0%	**0.877±0.009**	0.170±0.024	5.833±0.182
GEGL	-9.064±0.920	-9.91±0.990	-10.45±1.040	100.0±0.0%	0.853±0.003	0.643±0.014	2.99±0.054
REINVENT	-10.181±0.441	-11.234±0.632	-12.010±0.833	100.0±0.0%	0.857±0.011	0.445±0.058	2.596±0.116
RationaleRL	-9.233±0.920	-10.834±0.856	-11.642±1.102	100.0±0.0%	0.717±0.025	0.315±0.023	2.919±0.126
JTVAE	-9.291±0.702	-10.242±0.839	-10.963±1.133	98.0±0.027%	0.867±0.001	0.593±0.035	3.222±0.136
Gen3D	-8.686±0.450	-9.285±0.584	-9.832±0.324	100.0±0.0%	**0.870±0.006**	0.701±0.016	3.450±0.120
GA+D	-7.487±0.757	-8.305±0.803	-8.760±0.796	99.2±0.011%	0.834±0.035	0.405±0.024	5.024±0.164
Graph-GA	**-10.848±0.860**	**-11.702±0.930**	**-12.302±1.010**	100.0±0.0%	0.811±0.037	0.456±0.067	3.503±0.367
Autogrow 4.0	**-11.371±0.398**	**-12.213±0.623**	**-12.474±0.839**	100.0±0.0%	0.852±0.011	**0.748±0.022**	**2.497±0.049**
RGA (ours)	**-11.867±0.170**	**-12.564±0.287**	**-12.869±0.473**	100.0±0.0%	0.857±0.020	**0.742±0.036**	**2.473±0.048**
RGA - pretrain	-11.443±0.219	-12.424±0.386	-12.435±0.654	100.0±0.0%	0.854±0.035	0.750±0.034	2.494±0.043
RGA - KT	-11.434±0.169	-12.437±0.354	-12.502±0.603	100.0±0.0%	0.853±0.028	0.738±0.034	2.501±0.050

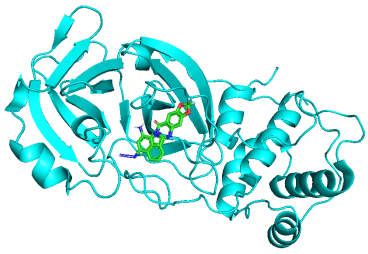
Metrics. The selection of evaluation metrics follows recent works in molecule optimization [20, 26, 31, 35] and structured based drug design [16, 10, 14]. For each method, we select top-100 molecules with the best docking scores for evaluation and consider the following metrics: **TOP-1/10/100** (average docking score of top-1/10/100 molecules): docking score directly measures the binding affinity between the ligand and target and is the most informative metric in structure-based drug design; **Novelty (Nov)** (% of the generated molecules that are not in training set); **Diversity (Div)** (average pairwise Tanimoto distance between the Morgan fingerprints); We also evaluate some simple pharmaceutical properties, including quantitative drug-likeness (**QED**) and synthetic accessibility (**SA**). QED score indicates drug-likeliness ranging from 0 to 1 (higher the better). SA score ranges from 1 to 10 (lower the better).

4.2 Results

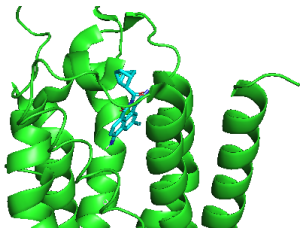
Stronger Optimization Performance. We summarized the main results of the structure-based drug design in Table 1. We evaluate all the methods on all targets and report each metric's mean and standard deviations across all targets. Our result shows RGA achieves the best performance in TOP-100/10/1 scores among all methods we compared. Compared to Autogrow 4.0, RGA's better performance in docking score demonstrates that the policy networks contribute positively to the chemical space navigation and eventually help discover more potent binding molecules. On the other hand, including longer-range navigation steps enabled by crossover leads to superior performance than other RL methods (REINVENT, MolDQN, GEGL and RationaleRL) that only focus on local modifications. In addition, we also observed competitive structure quality measured by QED (> 0.7) and SA_Score (< 2.5) in Autogrow 4.0 and RGA without involving them as optimization objectives, thanks to the mutation steps originating from chemical reactions. We visualize two designed ligands with optimal affinity for closer inspection in Figure 2(a) and 2(b), and find both ligands bind tightly with the targets.

Suppressed Random-Walk Behavior. Especially in SBDD, when the oracle functions are expensive molecular simulations, robustness to random seeds is essential for improving the worst-case performance of algorithms. One of the major issues in traditional GAs is that they have a significant variance between multiple independent runs as they randomly select parents for crossover and mutation types. To examine this behavior, we run five independent runs for RGA, Autogrow 4.0 and graph-GA (three best baselines, all are GA methods) on all targets and plot the standard deviations between runs in Figure 2(c) and 2(d). With policy networks guiding the action steps, we observed that the random-walk behavior in Autogrow 4.0 was suppressed in RGA, indicated by the smaller variance. Especially in the later learning phase (after 500 oracle calls), the policy networks are fine-tuned and guide the search more intelligently. This advantage leads to improved worst-case performance and a higher probability of successfully identifying bioactive drug candidates with constrained resources.

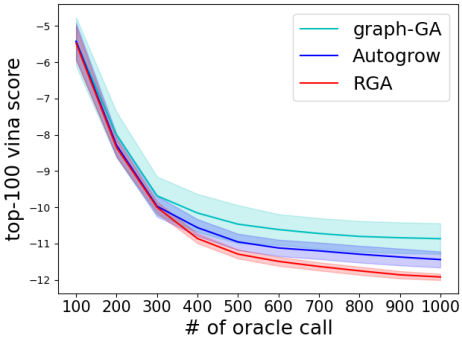
Knowledge Transfer Between Protein Targets. To verify if RGA benefited from learning the shared physics of ligand-target interaction, we conducted an ablation study whose results are in the last two rows of Table 1. Specifically, we compare RGA with two variants: (1) RGA-pretrain that does not pretrain the policy network with all native complex structures in the CrossDocked2020; (2) RGA-KT



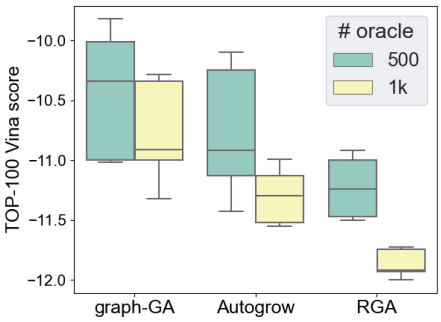
(a) Example of 7l11



(b) Example of 3eml



(c) TOP-100 vs # oracle call



(d) Score Bar

Figure 2: (a) and (b): Example of ligand poses (generated by RGA) and binding sites of target structures. Example of 7l11: the PDB ID of target is 7l11, which is SARS-COV-2(2019-NCOV) main protease. Example of 3eml: the PDB ID is 3eml, which is a human A2A Adenosine receptor. (c) and (d): studies of suppressed random-walk behavior. (c) reports TOP-100 docking score as a function of oracle calls. The results are the means and standard deviations of 5 independent runs. (d) shows the bars of TOP-100 docking score for various independent runs.

(knowledge transfer) that fine-tune the networks with data of individual target independently. We find that both strategies positively contribute to RGA on TOP-100/10/1 docking score. These results demonstrate the policy networks successfully learn the shared physics of ligand-target interactions and leverage the knowledge to improve their performance.

5 Conclusion

In this paper, we propose Reinforced Genetic Algorithm (RGA) to tackle the structure-based drug design problem. RGA reformulate the evolutionary process in genetic algorithms as a Markov decision process called evolutionary Markov decision process (EMDP) so that the searching processes could benefit from trained neural models. Specifically, we train policy networks to choose the parents to crossover and mutate instead of randomly sampling them. Further, we also leverage the common physics of the ligand-target interaction and adopt a knowledge-transfer strategy that uses data from other targets to train the networks. Through empirical study, we show that RGA has strong and robust optimization performance, consistently outperforming baseline methods in terms of docking score.

Though we adopted mutations originating from chemical reactions and the structural quality metrics seem good, we need to emphasize that the designed molecules from RGA do not guarantee synthesizability [48], as the crossover operations may break inheriting synthesizability. Directly working on synthetic pathways could solve the problem [27, 57], but the extension is not trivial. As for future direction, we expect to theoretically analyze the EMDP formulation and the performance of RGA. We also expect to generalize RGA to other combinatorial optimization scenarios, such as symbolic laws discovery [58], quantum circuits design [59]), etc.

References

[1] Regine S Bohacek, Colin McMartin, and Wayne C Guida. The art and practice of structure-based drug design: a molecular modeling perspective. *Medicinal research reviews*, 16(1):3–50, 1996.

[2] Ashutosh Tripathi and Vytas A Bankaitis. Molecular docking: From lock and key to combination lock. *Journal of molecular medicine and clinical applications*, 2(1), 2017.

[3] Assaf Alon, Jiankun Lyu, Joao M Braz, Tia A Tummino, Veronica Craik, Matthew J O'Meara, Chase M Webb, Dmytro S Radchenko, Yurii S Moroz, Xi-Ping Huang, et al. Structures of the $\sigma 2$ receptor enable docking for bioactive ligand discovery. *Nature*, 600(7890):759–764, 2021.

[4] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[5] Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, et al. Alphafold protein structure database: Massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic acids research*, 50(D1):D439–D444, 2022.

[6] Feng Ren, Xiao Ding, Min Zheng, Mikhail Korzinkin, Xin Cai, Wei Zhu, Alexey Mantsyzov, Alex Aliper, Vladimir Aladinskiy, Zhongying Cao, et al. Alphafold accelerates artificial intelligence powered drug discovery: Efficient discovery of a novel cyclin-dependent kinase 20 (cdk20) small molecule inhibitor. *arXiv preprint arXiv:2201.09647*, 2022.

[7] Jiankun Lyu, Sheng Wang, Trent E Balius, Isha Singh, Anat Levit, Yurii S Moroz, Matthew J O'Meara, Tao Che, Enkhjargal Algaa, Kateryna Tolmachova, et al. Ultra-large library docking for discovering new chemotypes. *Nature*, 566(7743):224–229, 2019.

[8] David E Graff, Eugene I Shakhnovich, and Connor W Coley. Accelerating high-throughput virtual screening through molecular pool-based active learning. *Chemical science*, 12(22):7866–7881, 2021.

[9] Francesco Gentile, Jean Charle Yaacoub, James Gleave, Michael Fernandez, Anh-Tien Ton, Fuqiang Ban, Abraham Stern, and Artem Cherkasov. Artificial intelligence–enabled virtual screening of ultra-large chemical libraries with deep docking. *Nature Protocols*, pages 1–26, 2022.

[10] Shitong Luo, Jiaqi Guan, Jianzhu Ma, and Jian Peng. A 3d generative model for structure-based drug design. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[11] Yibo Li, Jianfeng Pei, and Luhua Lai. Structure-based de novo drug design using 3d deep generative models. *Chemical science*, 12(41):13664–13675, 2021.

[12] W Patrick Walters and Mark Murcko. Assessing the impact of generative ai on medicinal chemistry. *Nature biotechnology*, 38(2):143–145, 2020.

[13] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. GuacaMol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

[14] Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: machine learning datasets and tasks for therapeutics. *NeurIPS Track Datasets and Benchmarks*, 2021.

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

[16] Jacob O Spiegel and Jacob D Durrant. Autogrow4: an open-source genetic algorithm for de novo drug design and lead optimization. *Journal of cheminformatics*, 12(1):1–16, 2020.

[17] Austin Tripp, Gregor NC Simm, and José Miguel Hernández-Lobato. A fresh look at de novo molecular design benchmarks. In *NeurIPS 2021 AI for Science Workshop*, 2021.

[18] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. *Proceedings of the 38th International Conference on Machine Learning, ICML*, 139:9323–9332, 2021.

[19] 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. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 2018.

[20] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *ICML*, 2018.

[21] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

[22] Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs, 2018.

[23] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. GraphAF: a flow-based autoregressive model for molecular graph generation. In *ICLR*, 2020.

[24] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. GraphDF: A discrete flow model for molecular graph generation. *Proceedings of the 38th International Conference on Machine Learning, ICML*, 139:7192–7203, 2021.

[25] Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. Graphebm: Molecular graph generation with energy-based models. *arXiv preprint arXiv:2102.00546*, 2021.

[26] AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alán Aspuru-Guzik. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. In *ICLR*, 2020.

[27] Wenhao Gao, Rocío Mercado, and Connor W Coley. Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design. *International Conference on Learning Representations*, 2022.

[28] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 2017.

[29] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *NIPS*, 2018.

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

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

[32] Sungsoo Ahn, Junsu Kim, Hankook Lee, and Jinwoo Shin. Guiding deep molecular optimization with genetic exploration. *Advances in neural information processing systems*, 33:12008–12021, 2020.

[33] Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. ChemBO: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, pages 3393–3403. PMLR, 2020.

[34] Tianfan Fu, Cao Xiao, Xinhao Li, Lucas M Glass, and Jimeng Sun. MIMOSA: Multi-constraint molecule sampling for molecule optimization. *AAAI*, 2020.

[35] Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. MARS: Markov molecular sampling for multi-objective drug discovery. In *ICLR*, 2021.

[36] Yoshua Bengio, Tristan Deleu, Edward J. Hu, Salem Lahlou, Mo Tiwari, and Emmanuel Bengio. GFlowNet foundations. *CoRR*, abs/2111.09266, 2021.

[37] Tianfan Fu, Wenhao Gao, Cao Xiao, Jacob Yasonik, Connor W Coley, and Jimeng Sun. Differentiable scaffolding tree for molecular optimization. *International Conference on Learning Representations*, 2022.

[38] Cynthia Shen, Mario Krenn, Sagi Eppel, and Alan Aspuru-Guzik. Deep molecular dreaming: Inverse machine learning for de-novo molecular design and interpretability with surjective representations. *Machine Learning: Science and Technology*, 2021.

[39] Tobiasz Cieplinski, Tomasz Danel, Sabina Podlewska, and Stanislaw Jastrzebski. We should at least be able to design molecules that dock well. *arXiv:2006.16955*, 2020.

[40] Casper Steinmann and Jan H Jensen. Using a genetic algorithm to find molecules with good docking scores. *PeerJ Physical Chemistry*, 3:e18, 2021.

[41] Soojung Yang, Doyeong Hwang, Seul Lee, Seongok Ryu, and Sung Ju Hwang. Hit and lead discovery with explorative rl and fragment-based molecule generation. *Advances in Neural Information Processing Systems*, 34, 2021.

[42] Zhaowen Luo, Renxiao Wang, and Luhua Lai. Rasse: a new method for structure-based drug design. *Journal of chemical information and computer sciences*, 36(6):1187–1194, 1996.

[43] Valerie Gillet, A Peter Johnson, Pauline Mata, Sandor Sike, and Philip Williams. Sprout: a program for structure generation. *Journal of computer-aided molecular design*, 7(2):127–153, 1993.

[44] David A Pearlman and Mark A Murcko. Concepts: New dynamic algorithm for de novo drug suggestion. *Journal of computational chemistry*, 14(10):1184–1193, 1993.

[45] Dominique Douguet, Etienne Thoreau, and Gérard Grassy. A genetic algorithm for the automated generation of small organic molecules: drug design using an evolutionary algorithm. *Journal of computer-aided molecular design*, 14(5):449–466, 2000.

[46] Jacob D Durrant, Steffen Lindert, and J Andrew McCammon. Autogrow 3.0: an improved algorithm for chemically tractable, semi-automated protein inhibitor design. *Journal of Molecular Graphics and Modelling*, 44:104–112, 2013.

[47] Renxiao Wang, Liang Liu, Luhua Lai, and Youqi Tang. Score: A new empirical method for estimating the binding affinity of a protein-ligand complex. *Molecular modeling annual*, 4(12):379–394, 1998.

[48] Wenhao Gao and Connor W Coley. The synthesizability of molecules proposed by generative models. *Journal of chemical information and modeling*, 60(12):5714–5723, 2020.

[49] Markus Hartenfeller, Martin Eberle, Peter Meier, Cristina Nieto-Oberhuber, Karl-Heinz Altmann, Gisbert Schneider, Edgar Jacoby, and Steffen Renner. A collection of robust organic synthesis reactions for in silico molecule design. *Journal of chemical information and modeling*, 51(12):3093–3098, 2011.

[50] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[51] Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

[52] Michael M Mysinger, Michael Carchia, John J Irwin, and Brian K Shoichet. Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *Journal of medicinal chemistry*, 55(14):6582–6594, 2012.

[53] Chun-Hui Zhang, Elizabeth A Stone, Maya Deshmukh, Joseph A Ippolito, Mohammad M Ghahremanpour, Julian Tirado-Rives, Krasimir A Spasov, Shuo Zhang, Yuka Takeo, Shalley N Kudalkar, et al. Potent noncovalent inhibitors of the main protease of sars-cov-2 from molecular sculpting of the drug perampanel guided by free energy perturbation calculations. *ACS central science*, 7(3):467–475, 2021.

[54] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

[55] Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

[56] Paul G Francoeur, Tomohide Masuda, Jocelyn Sunseri, Andrew Jia, Richard B Iovanisci, Ian Snyder, and David R Koes. Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of Chemical Information and Modeling*, 60(9):4200–4215, 2020.

[57] John Bradshaw, Brooks Paige, Matt J Kusner, Marwin Segler, and José Miguel Hernández-Lobato. Barking up the right tree: an approach to search over molecule synthesis dags. *Advances in Neural Information Processing Systems*, 33:6852–6866, 2020.

[58] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[59] Yuxuan Du, Tao Huang, Shan You, Min-Hsiu Hsieh, and Dacheng Tao. Quantum circuit architecture search: error mitigation and trainability enhancement for variational quantum solvers. *arXiv preprint arXiv:2010.10217*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the supplementary materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See the supplementary materials.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the supplementary materials.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See the supplementary materials.
   (b) Did you mention the license of the assets? [Yes] See the supplementary materials.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See the supplementary materials.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]