Genetic-guided GFlowNets for Sample Efficient Molecular Optimization

Hyeonah Kim[1*] **Minsu Kim**[1] **Sanghyeok Choi**[1] **Jinkyoo Park**[1,2]
[1]Korea Advanced Institute of Science and Technology (KAIST), [2]OMELET

Abstract

The challenge of discovering new molecules with desired properties is crucial in domains like drug discovery and material design. Recent advances in deep learning-based generative methods have shown promise but face the issue of sample efficiency due to the computational expense of evaluating the reward function. This paper proposes a novel algorithm for sample-efficient molecular optimization by distilling a powerful genetic algorithm into deep generative policy using GFlowNets training, the off-policy method for amortized inference. This approach enables the deep generative policy to learn from domain knowledge, which has been explicitly integrated into the genetic algorithm. Our method achieves state-of-the-art performance in the official molecular optimization benchmark, significantly outperforming previous methods. It also demonstrates effectiveness in designing inhibitors against SARS-CoV-2 with substantially fewer reward calls.

1 Introduction

Discovering new molecules is one of the fundamental tasks in the chemical domain, with applications in drug discovery [1] and material design [2]. Particularly, *de novo* molecular design focuses on generating novel molecules with desired properties from scratch. In this context, deep learning-based generative methods have emerged, showing promising results (e.g., [3–7]). However, these methods still face a key challenge: the reward function is computationally expensive (e.g., assessing binding affinity through docking simulations), while the molecule space is combinatorially large.

Sample-efficient molecular optimization is thus crucial for discovering high-reward molecular structures with limited reward calls, especially for real-world applicability. The recently proposed benchmark, Practical Molecular Optimization (PMO) [8], has extensively assessed the sample efficiency of various algorithms, including reinforcement learning [3, 9], active learning [10], variational autoencoders [4, 5], generative flow networks (GFlowNets) [6, 11], and classical optimization methods like Bayesian optimization [12] and genetic algorithms [13, 14]. Interestingly, the PMO benchmark has revealed a shift in algorithm rankings, with classical algorithms often outperforming recently proposed methods such as GFlowNets when the sample efficiency is considered.

Recent investigations [15, 16], including those highlighted by the PMO benchmark [8], indicate that the classical frameworks, especially genetic algorithms (GA), still exhibit competitive performances compared to recently proposed deep learning methods. These studies underscore that GAs effectively navigate the chemical space using domain-specific genetic operators. In contrast, deep learning methods usually do not leverage domain-specific knowledge, relying instead on deep networks to autonomously learn these insights. It can lead to inefficient training processes due to the lack of expert guidance [17]. To address this limitation, Genetic Expert Guided Learning (GEGL) [17] has been introduced, which enhances deep learning by distilling GA-generated samples into a deep generative policy using maximum likelihood estimation. This approach enables the deep generative policy to

*hyeonah_kim@kaist.ac.kr

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-1350

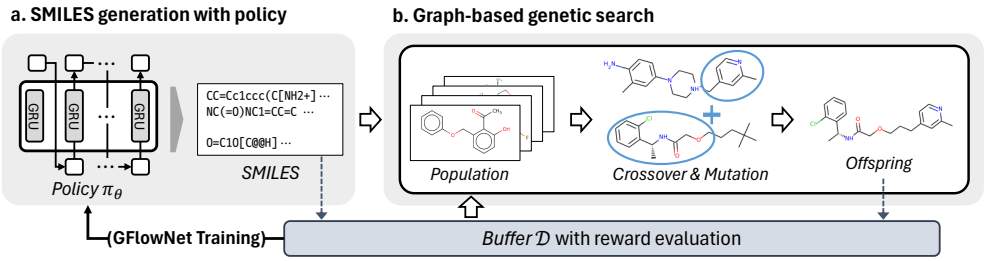


Figure 1: Overview of Genetic GFN. Our generative policy is trained to sample molecules proportional to rewards, and the genetic search refines them to higher-reward samples.

implicitly utilize domain-specific knowledge from the GA as a form of guidance. However, despite its successes, GEGL may face challenges in generalizing to unexplored regions and in learning a peaky distribution from samples, as it maximizes likelihood equally across all high-reward samples without adequately configuring the reward landscape.

To address this, we propose a novel molecular optimization algorithm that integrates domain-specialized genetic algorithms into GFlowNets, an off-policy training method that trains policy to sample proportional to their rewards. As illustrated in Fig. 1, we first generate diverse candidates using the current policy and refine the candidates into higher-reward samples using GAs. Subsequently, we fine-tune the policy with a GFlowNet using the collected samples. Unlike the MLE training, which can be stuck in local modes when the reward landscape is peaky, GFlowNet trains the policy to sample molecules *proportional to rewards*. To enhance sample efficiency, we perform unsupervised pretraining on chemical datasets and regularize the GFlowNets policy with KL-divergence to align generative probability with the dataset distribution, focusing on the compact valid space.

Our contributions can be interpreted from both perspectives of GFlowNets and GAs:

GA increases the exploitation power of GFlowNets. The proposed algorithm incorporates an effective off-policy exploration into GFlowNets based on domain-specialized GA. This approach aligns with recent studies that utilize off-policy explorations to guide toward the high-reward regions [18]. Our key contribution lies in explicitly leveraging domain knowledge about chemical structures and effectively distilling it into GFlowNets, which enable exceptional performance in real-world tasks beyond small-scale molecular generations. This contribution is crucial for the field of GFlowNets, which have struggled with sample-efficient molecular optimization tasks, even when using proxy reward models for active learning [10, 8]; see Section 4.2.

GFlowNets increases population diversity of GA. The proposed algorithm generates diversified samples using GFlowNets, enabling GA to effectively improve samples; our method can be regarded as a new genetic algorithm with deep generative policy-based population resetting. Our GFlowNet policy, parameterized over the whole space, periodically resets the population by diversely sampling individuals proportional to rewards, mitigating premature convergence of GA [19, 20]. Experiments show that ours outperforms recent GAs in the PMO benchmark; see Section 5.1.

Our extensive experiments demonstrate the effectiveness and practical applicability of the proposed method. First, our method achieves the highest total score of 16.213 across 23 oracles in the Practical Molecular Optimization benchmark [8], outperforming all other baselines. Second, we conduct in *in silico* experiments for designing SARS-CoV-2 inhibitors. The proposed method successfully generates inhibitors with ten times fewer reward calls. Moreover, our method effectively balances optimization and diversity, achieving higher scores with increased diversity compared to previously top-ranked methods.

2 Background

2.1 Sample efficient *de novo* molecular optimization

Molecular design is the process of proposing new molecules likely to exhibit desirable outcomes. Compared to traditional virtual screening approaches, which identify suitable molecules from virtual

libraries with a large number of molecules known a priori, de novo approaches seek to generate molecule structures anew. The desired properties can be measured using score functions $\mathcal{O}$, called oracle. Formally, molecular design can be formulated as $\arg\max_{x \in \mathcal{X}} \mathcal{O}(x)$, where x is a molecule, and $\mathcal{X}$ denotes the chemical space which comprises all possible molecules.

The publication of standard benchmarks and datasets has facilitated the assessment of *de novo* design methods (e.g., GuacaMol [21], Therapeutics Data Commons (TDC) [22]). The score functions are designed to consider various properties, such as the presence and absence of substructures, similarity, isomers, structural features, physicochemical properties, biological activity, and binding affinity (i.e., docking score). Notably, the PMO benchmark [8] offers a unified framework that comprehensively evaluates the sample efficiency of a range of molecular design methods.

2.2 Generative flow networks

Generative flow networks (GFlowNets) [23] are introduced as a new class of probabilistic models to sample a discrete compositional object $x \in \mathcal{X}$ from the target distribution, i.e., $P(x) \propto e^{-\mathcal{E}(x)}$. In general, direct sampling from the target distribution is challenging since the partition function $Z = \sum_{x \in \mathcal{X}} e^{-\mathcal{E}(x)}$ is intractable when the sample space is combinatorially large. Hence, GFlowNets sample an object from an unnormalized distribution as a constructive generative process, where discrete *actions* iteratively modify a *state* — a partially constructed object. We define a trajectory as $\tau = (s_0, \ldots, s_T)$, where s_T is a terminal state corresponding to a fully constructed object x.

A GFlowNet models flow F of particles along a directed acyclic graph (DAG). The source and sink nodes of the DAG correspond to the initial state s_0 and terminal states s_T, respectively. The *trajectory flow* $F(\tau)$ is defined as a flow through the trajectory τ, and the *state flow* $F(s)$ is defined as the sum of trajectory flows that include the state s, i.e., $F(s) = \sum_{s \in \tau} F(\tau)$. The *edge flow* $F(s \to s')$ the sum of trajectory flows through the edge from state s to s', i.e., $F(s \to s') = \sum_{(s,s') \in \tau} F(\tau)$.

From the flow function F, we derive two policy distributions. The *forward policy* $P_F(s'|s)$ is the probability of transitioning from state s to its child state s', defined as the edge flow $F(s \to s')$ normalized by the state flow $F(s)$, i.e., $P_F(s'|s) = F(s \to s')/F(s)$. Similarly, the *backward policy* $P_B(s|s')$ is the probability of moving from state s' to its parent state s, defined as $P_B(s|s') = F(s \to s')/F(s')$. Utilizing these forward and backward policies, GFlowNets can derive an optimal sampler $P(s_T) = \prod P_F(s_t|s_{t-1}) = R(s_T)/Z$ if balance conditions (e.g., [6, 24–27]) are satisfied.

Trajectory balance loss [24]. One of the most popular conditions is *trajectory balance (TB)*, which directly parameterizes P_F, P_B, and flow of initial state (i.e., partition function) Z to satisfy the following trajectory balance condition:

$$Z \prod_{t=1}^{n} P_F(s_t|s_{t-1}) = R(s_T) \prod_{t=1}^{n} P_B(s_{t-1}|s_t).$$

Then, this equation is converted into a loss function to be minimized along sampled trajectories, i.e.,

$$\mathcal{L}_{\text{TB}}(\tau;\theta) = \left(\log \frac{Z_\theta \prod_{t=1}^{n} P_F(s_t|s_{t-1};\theta)}{R(x) \prod_{t=1}^{n} P_B(s_{t-1}|s_t;\theta)}\right)^2. \tag{1}$$

In GFlowNet training, employing exploratory behavior policies or replay training is allowed since GFlowNet can be trained in an off-policy manner, which is a key advantage [23, 24, 28, 18].

3 Genetic-guided GFlowNets

This section describes how the desired molecules are discovered with Genetic GFN. We model the generation process of molecules as a string-based constructive process. First, we pretrain the policy to learn the distribution of valid molecules. During the optimization phase, we iteratively generate molecules and update the policy with GFlowNet training using high-reward molecules. Particularly, we introduce graph-based genetic search to refine generated samples.

https://doi.org/10.52202/079017-1350

Algorithm 1 Genetic GFN training with limited reward calls

Set $\pi_\theta \leftarrow \pi_{\text{pre}}$, $\mathcal{D} \leftarrow \emptyset$
while $|\mathcal{D}| \leq$ `numOracle` **do**
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{x}, \mathcal{O}(\boldsymbol{x})\}$, where $\boldsymbol{x} \sim \pi_\theta(\cdot)$ ▷ *SMILES generation with policy*
  Initialize population $\mathcal{D}_{\text{pop}}$ from $\mathcal{D}$ ▷ *Graph-based genetic search*
  for $n = 1$ **to** `numGen` **do**
    $\boldsymbol{x} \leftarrow$ `Crossover`$(\boldsymbol{x}_1, \boldsymbol{x}_2)$, where $(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathcal{D}_{\text{pop}}$
    $\boldsymbol{x}' \leftarrow$ `Mutate`$(\boldsymbol{x})$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{x}', \mathcal{O}(\boldsymbol{x}')\}$, $\mathcal{D}_{\text{off}} \leftarrow \mathcal{D}_{\text{off}} \cup \{\boldsymbol{x}', \mathcal{O}(\boldsymbol{x}')\}$
    $\mathcal{D}_{\text{pop}} \leftarrow$ `Select`$(\mathcal{D}_{\text{pop}} \cup \mathcal{D}_{\text{off}})$
  end for
  for $k = 1$ **to** `numReplay` **do** ▷ *Updating the policy with GFlowNet training*
    Get $\mathcal{B}$ from $\mathcal{D}$ with rank-based sampling (Eq. (4))
    Update θ to minimize $\frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{x}\in\mathcal{B}} \mathcal{L}_{\text{TB}} + \alpha \text{KL}(\pi_\theta(\boldsymbol{x})||\pi_{\text{pre}}(\boldsymbol{x}))$
  end for
end while

3.1 Factorized string-based generative policy and unsupervised pretraining

Building on insights from previous works [3, 17], we employ a string-based representation strategy, simplifying the molecular generation process by reducing it to a one-directional sequence generation. We adopt a sequence generative policy using a string-based assembly strategy, especially the simplified molecular-input line-entry system (SMILES) [29]. Motivated by REINVENT [3], we parameterize the policy using a recurrent neural network architecture [30]. Then, the probability $\pi_\theta(\boldsymbol{x})$ of generating a molecule, can be factorized to $\prod_{t=1}^{n} \pi_\theta(x_t|x_1, \ldots, x_{t-1})$, where $x_1, \ldots, x_n$ are characters of SMILES representation of $\boldsymbol{x}$.

As demonstrated in previous studies, including [3, 17, 8], pretraining is inevitable since training the generative policy from scratch is excessively sample-inefficient. Therefore, our policy is pre-trained to maximize the likelihood of valid molecules on existing chemical datasets $\mathcal{D}_{\text{pre}}$; note that pretraining *does not require oracle information*. Precisely, the policy is pretrained to minimize the following:

$$\mathcal{L}_{\text{pre}}(\boldsymbol{x}) = -\sum_{t=1}^{n} \log \pi_\theta(x_t|x_1, \ldots, x_{t-1}). \tag{2}$$

3.2 GFlowNet training of the generative policy with graph-based genetic search

To generate desirable molecules with limited reward calls, we iteratively generate samples using two distinct strategies (Section 3.2.1) and fine-tune the policy, initialized with π_{pre}, using a GFlowNet by replaying collected samples (Section 3.2.2). The overall procedure is described in Algorithm 1.

3.2.1 Molecule generation strategies in Genetic GFN

We employ two distinct molecule generation strategies, SMILES generation with our training policy and graph-based genetic search. These two strategies are synergized to generate diversified and high-reward samples, efficiently searching the vast chemical space.

SMILES generation with policy. The training policy π_θ generates SMILES sequences. Since the policy is trained using the trajectory balance loss (see the following subsection), it is the same as sampling $\boldsymbol{x}$ from $\prod_{t=1}^{T} P_F(s_t|s_{t-1}) \propto R(s_T = \boldsymbol{x})$, where s_{t-1} is represented by previously collected SMILES token, and $\log R(\boldsymbol{x}) = -\beta\mathcal{O}(\boldsymbol{x})$ with an inverse temperature β.

Graph-based genetic search. To effectively search the higher-reward region, we employ a genetic algorithm that iteratively evolves populations through *crossover*, *mutation*, and *selection*. We adopt the operations of the graph-based genetic algorithm, Graph GA [13], which has proven to effectively search the molecule space with finely designed genetic operations; please refer to the original paper for details. The genetic search is performed as follows:

1. **Initialize a population** $\mathcal{D}_{\text{pop}}$: The initial population is selected from the whole buffer $\mathcal{D}$, consisting of samples from the policy and previous genetic search.
2. **Generate offspring** $\mathcal{D}_{\text{off}}$: A child is generated from randomly chosen two parent molecules by combining the fragments (*crossover*). Then, the child is randomly modified (*mutation*).
3. **Select a new population** $\mathcal{D}'_{\text{pop}}$: Sample from $\mathcal{D}_{\text{pop}} \cup \mathcal{D}_{\text{off}}$, and go back to 2.

One key advantage is that offspring can have a large distance from the parents in the 1D string space, even if the molecule distances are small, which is beneficial to avoid being stuck in local optima; see the experimental results in Table 3b.

3.2.2 Updating the generative policy with GFlowNets training

Using the generated samples, the policy is fine-tuned using the trajectory balance loss. The off-policy property of GFlowNet losses enables the utilization of refined samples from the genetic search. In particular, for better sample efficiency, we employ replay training with a rank-based reweighed buffer.

TB loss with KL-divergence penalty. The generative policy is trained using the trajectory balance loss in Eq. (1). Note that we set P_B to 1 for simplicity since SMILES generations are conducted in one direction. To ensure that the policy does not deviate excessively from the pretrained policy during training, we introduce a Kullback-Leibler (KL) divergence penalty inspired by the works in language model fine-tuning [31, 32]. Thus, our model is updated to minimize the following loss function:

$$\mathcal{L} = \mathcal{L}_{\text{TB}}(\tau;\theta) + \alpha \text{KL}(\pi_\theta(x)||\pi_{\text{pre}}(x)), \tag{3}$$

where π_{pre} denotes the the pre-trained policy. As a result, π_θ is trained to generate desired (by $\mathcal{L}_{\text{TB}}$) and valid (by π_{pre}) molecules. Note that trajectories on which the proposed loss is minimized are sampled from the experience buffer.

Rank-based reweighed experience buffer. The rank-based reweighting biases the samples towards high-reward candidates by assigning greater weight to trajectories with higher ranks, thereby enhancing the focus on more promising solutions [33, 34]. The weight is computed as follows:

$$\frac{(k|\mathcal{D}| + \text{rank}_{\mathcal{O},\mathcal{D}}(\boldsymbol{x}))^{-1}}{\sum_{\boldsymbol{x}\in\mathcal{D}} (k|\mathcal{D}| + \text{rank}_{\mathcal{O},\mathcal{D}}(\boldsymbol{x}))^{-1}}. \tag{4}$$

Here, k is a weight-shifting factor, and $\text{rank}_{\mathcal{O},\mathcal{D}}(\boldsymbol{x})$ is a relative rank of value of $\mathcal{O}(\boldsymbol{x})$ in the dataset $\mathcal{D}$. Note that we also utilize rank-based sampling in the genetic search (steps 1 and 3).

4 Related works

4.1 Genetic algorithms for molecular optimization

Genetic algorithm (GA) is a representative meta-heuristic inspired by the evolutionary process. This subsection focuses on discussing the application of GA in molecular optimization. As one of the seminal works, a graph-based GA (Graph GA) was proposed with sophisticatedly designed operations based on chemical knowledge [13]. Note that our method also adopts Graph GA operations in the genetic search. Various strategies for molecular assembly, not limited to graphs, have been utilized in GA [35, 14, 36]. A recent contribution by [16] introduces an enhanced version of Graph GA. They introduce quantile-uniform sampling to bias the population towards containing higher reward samples while maintaining diversity. Experimental results from Mol GA demonstrate the effectiveness of GAs as strong baselines, achieving state-of-the-art (SOTA) performance in the PMO benchmark.

4.2 GFlowNets for molecular optimization

Generative Flow Networks (GFlowNets or GFN) [23, 6] have drawn significant attention in scientific discovery [37], particularly in molecular optimization and biological sequence design [6, 10, 38, 18, 39, 40, 11, 41]. GFlowNets, which are off-policy variational inference methods [42], are closely related to value-based reinforcement learning within soft Markov Decision Processes (soft MDPs) [43, 44], focusing on learning maximum entropy agents [45, 46]. This allows GFlowNets to generate

https://doi.org/10.52202/079017-1350

Table 1: Mean and standard deviation of AUC top-10 (↑) from five independent runs. We use oracle ID (in lexicographical order) instead of a name for better readability, and the best mean scores are denoted in **bold** for each task. The results of further baselines are provided in Appendix G.2.

ID	Genetic GFN (Ours)	Mol GA [16]	SMILES REINVENT [3]	GEGL [17]	GP BO [12]	Fragment GFN [6]	Fragment GFN-AL [10]
#1	**0.949 ± 0.010**	0.928 ± 0.015	0.881 ± 0.016	0.842 ± 0.019	0.902 ± 0.011	0.382 ± 0.010	0.459 ± 0.028
#2	**0.761 ± 0.019**	0.740 ± 0.055	0.644 ± 0.019	0.626 ± 0.018	0.579 ± 0.035	0.428 ± 0.002	0.437 ± 0.007
#3	**0.802 ± 0.029**	0.629 ± 0.062	0.717 ± 0.027	0.699 ± 0.041	0.746 ± 0.025	0.263 ± 0.009	0.326 ± 0.008
#4	0.733 ± 0.109	0.656 ± 0.013	0.662 ± 0.044	0.656 ± 0.039	0.615 ± 0.009	0.582 ± 0.001	0.587 ± 0.002
#5	**0.974 ± 0.006**	0.950 ± 0.004	0.957 ± 0.007	0.898 ± 0.015	0.941 ± 0.017	0.480 ± 0.075	0.601 ± 0.055
#6	**0.856 ± 0.039**	0.835 ± 0.012	0.781 ± 0.013	0.769 ± 0.009	0.726 ± 0.004	0.689 ± 0.003	0.700 ± 0.005
#7	0.881 ± 0.042	**0.894 ± 0.025**	0.885 ± 0.031	0.816 ± 0.027	0.861 ± 0.027	0.589 ± 0.009	0.666 ± 0.006
#8	**0.969 ± 0.003**	0.926 ± 0.014	0.942 ± 0.012	0.930 ± 0.011	0.883 ± 0.040	0.791 ± 0.024	0.468 ± 0.211
#9	**0.897 ± 0.007**	0.894 ± 0.005	0.838 ± 0.030	0.808 ± 0.007	0.805 ± 0.007	0.576 ± 0.021	0.199 ± 0.199
#10	0.764 ± 0.069	**0.835 ± 0.040**	0.782 ± 0.029	0.580 ± 0.086	0.611 ± 0.080	0.359 ± 0.009	0.442 ± 0.017
#11	**0.379 ± 0.010**	0.329 ± 0.006	0.363 ± 0.011	0.338 ± 0.016	0.298 ± 0.016	0.192 ± 0.003	0.207 ± 0.003
#12	0.294 ± 0.007	0.284 ± 0.035	0.281 ± 0.002	0.274 ± 0.007	**0.296 ± 0.011**	0.174 ± 0.002	0.181 ± 0.002
#13	0.708 ± 0.057	**0.762 ± 0.048**	0.634 ± 0.042	0.599 ± 0.035	0.631 ± 0.093	0.291 ± 0.005	0.332 ± 0.012
#14	**0.860 ± 0.008**	0.853 ± 0.005	0.834 ± 0.010	0.832 ± 0.005	0.788 ± 0.005	0.787 ± 0.002	0.785 ± 0.003
#15	0.595 ± 0.014	**0.610 ± 0.038**	0.535 ± 0.015	0.537 ± 0.015	0.494 ± 0.006	0.423 ± 0.006	0.434 ± 0.006
#16	**0.942 ± 0.000**	0.941 ± 0.001	0.941 ± 0.000	0.941 ± 0.001	0.937 ± 0.002	0.904 ± 0.002	0.917 ± 0.002
#17	0.819 ± 0.018	**0.830 ± 0.010**	0.770 ± 0.005	0.730 ± 0.011	0.741 ± 0.010	0.626 ± 0.005	0.660 ± 0.004
#18	**0.615 ± 0.100**	0.568 ± 0.017	0.551 ± 0.024	0.531 ± 0.010	0.535 ± 0.007	0.461 ± 0.002	0.464 ± 0.003
#19	0.634 ± 0.039	**0.677 ± 0.055**	0.470 ± 0.041	0.402 ± 0.024	0.461 ± 0.057	0.180 ± 0.012	0.217 ± 0.022
#20	**0.583 ± 0.034**	0.544 ± 0.067	0.544 ± 0.026	0.515 ± 0.028	0.544 ± 0.038	0.261 ± 0.004	0.292 ± 0.009
#21	**0.511 ± 0.054**	0.487 ± 0.024	0.458 ± 0.018	0.420 ± 0.031	0.404 ± 0.025	0.183 ± 0.001	0.190 ± 0.002
#22	0.135 ± 0.271	0.000 ± 0.000	**0.182 ± 0.363**	0.119 ± 0.238	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
#23	**0.552 ± 0.033**	0.514 ± 0.033	0.533 ± 0.009	0.492 ± 0.021	0.466 ± 0.025	0.308 ± 0.027	0.353 ± 0.024
Sum	**16.213**	15.686	15.185	14.354	14.264	9.929	9.917

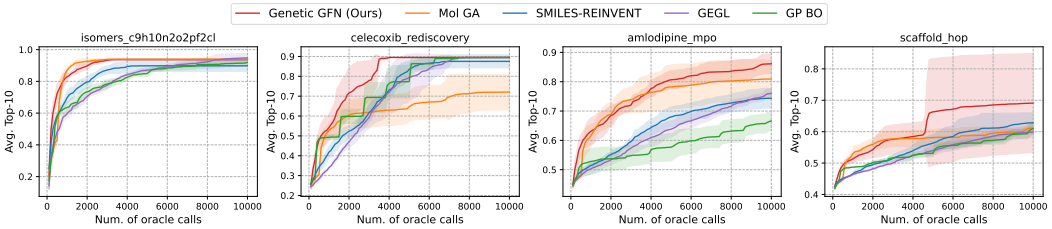


Figure 2: The optimization curve of the average scores of Top-10 over the score function calls. All optimization curves for 23 oracles are provided in Appendix G.2.

diverse candidates for chemical and biological structures. Advances in GFlowNets have included new objective functions [6, 24, 25], improved credit assignments [26, 27], and enhanced off-policy exploration strategies [18, 47–49]. Our approach, an improved off-policy exploration technique for GFlowNets, enhances sample efficiency and scalability for moderate-scale chemical discovery. We present extensive experiments targeting the discovery of larger-scale molecules, with SMILES strings approximately 100 characters long, surpassing the scope of smaller molecule generation tasks commonly addressed in GFlowNets literature [6, 38].

5 Experiments

This section provides extensive experimental results, including experiments on the official sample-efficient molecular optimization benchmark and *in silico* design for SARS-CoV-2 inhibitors. The codes are available at https://github.com/hyeonahkimm/genetic_gfn.

5.1 Sample efficient molecular optimization

In this experiment section, we compare Genetic GFN with various molecular optimization methods from the perspective of sample efficiency. The performance is primarily quantified by the area under the curve (AUC), with the number of score function calls limited to 10K. Note that we rigorously follow the Practical Molecular Optimization (PMO) benchmark [8].

Table 2: Ablation studies. In the GS ablation study (-{GS}), the generative policy solely generates samples, while ϵ-greedy samples from P_F mixed with a uniform distribution. The **bold** text indicates the best value.

	Genetic GFN	Genetic Search		KL-divergence penalty
		- {GS}	- {GS} + {ϵ-greedy}	
AUC Top-1	**16.530 ± 0.198**	16.070 ± 0.290	15.966 ± 0.085	16.251 ± 0.440
AUC Top-10	**16.213 ± 0.173**	15.738 ± 0.274	15.626 ± 0.082	15.928 ± 0.426
AUC Top-100	**15.516 ± 0.127**	15.030 ± 0.322	14.939 ± 0.147	15.188 ± 0.297

5.1.1 Main results in the official benchmark of PMO

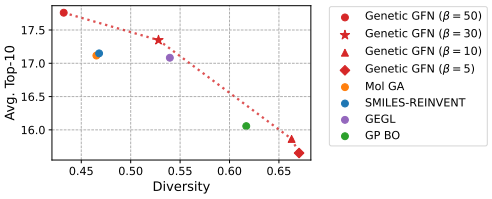
As baselines, we employ Top-8 methods from the PMO benchmark since they recorded the best AUC Top-10 in at least one oracle. The baseline methods include various ranges of algorithms and representation strategies. First, REINVENT [3] is an RL method that tunes the policy with adjusted likelihood. Graph GA [13], STONED [14], SMILES GA [21], and SynNet [36] are genetic algorithms that utilize different assembly strategies; they use fragment-based graphs, SELFIES, SMILES, and synthesis, respectively. Additionally, a hill climbing method (SMILES-LSTM-HC [21]) and Bayesian optimization (GP BO [12]) are included. SMILES-LSTM-HC iteratively generates samples and imitates high-reward samples, while GP BO uses a surrogate model with the Gaussian process (GP) and Graph GA to optimize the GP acquisition functions in the inner loop.

Moreover, we adopt additional methods, Mol GA [16] and GEGL [17]. Mol GA is an advanced version of Graph GA and outperforms other baselines in the PMO benchmark. On the other hand, GEGL is an ablated version of our approach that utilizes imitation learning with a reward-priority queue instead of GFlowNet training with rank-based sampling. For both, we adopt the original implementations[2][3] with hyperparameters searches following the guidelines; see Appendix C.

The main results in Table 1 report the AUC score of Top-10 candidates with independent five runs with different seeds. In addition, Fig. 2 visually presents the Top-10 average score across the computational budget, i.e., the number of oracle calls, providing a concise overview of the results. Due to the lack of space, the best five results are provided; please check Appendix G.2 for the rest of the results. As shown in Table 1, Genetic GFN outperforms the other baselines with a total of 16.213 and attains the highest AUC Top-10 values in 14 out of 23 score functions. The results of diversity and SA score for each oracle are presented in Appendix G.5.

5.1.2 Controllability of the scores-diversity trade-off and ablation studies

Controllability of the scores-diversity trade-off. In the benchmark, we found a pronounced trade-off between attaining high evaluation scores within a limited budget and generating diverse molecular candidates. This section demonstrates the controllability of the score-diversity trade-off through adjustments in the inverse temperature β. Decreasing the inverse temperature gives more diverse candidates. The results in Fig. 3 demonstrate adjustments of β can control the trade-off between score and diversity, achieving Pareto-frontier to other baselines in the benchmark. Notably, Genetic GFN with $\beta = 30$ achieves a higher AUC Top-10 with a greater diversity compared to the SOTA GA method (Mol GA: 15.686 with a diversity of 0.465) and RL method (REINVENT: 15.185 with a diversity of 0.468). Similarly, the weight-shifting factor k in rank-based sampling can control the trade-off; see Appendix G.1.



	$\beta = 1$	$\beta = 5$	$\beta = 10$	$\beta = 30$	$\beta = 50$
AUC-10	11.083	14.597	14.735	15.815	*16.213*
Div.	0.812	0.670	0.663	0.528	*0.432*

Figure 3: Average of Top-10 score and diversity. Note that the fragment-based GFlowNet achieves 10.957 with a diversity of 0.816.

Ablation studies. The ablation studies investigate the essential components of our framework: the genetic search (GS) and the KL-divergence penalty. To assess the effectiveness of the genetic search, we also compare its performance against the exploration strategy used in previous GFlowNet studies

[2] https://github.com/AustinT/mol_ga

[3] https://github.com/sungsoo-ahn/genetic-expert-guided-learning

https://doi.org/10.52202/079017-1350

Table 3: Comparison with GFlowNet variants. Notably, samples from the GS have larger SMILES distances than LS, leading to better sample efficiency. The **bold** text indicates the best value.

(a) Average and standard deviation of AUC scores (↑)

	SMILES		Fragment-based	
	Genetic GFN	LS-GFN [18]	GFN [6]	GFN-AL [10]
AUC Top-1	**16.530 ± 0.198**	15.514 ± 0.269	10.957 ± 0.033	11.032 ± 0.016
AUC Top-10	**16.213 ± 0.173**	15.230 ± 0.026	9.918 ± 0.027	9.928 ± 0.027
AUC Top-100	**15.516 ± 0.127**	14.619 ± 0.027	8.416 ± 0.024	8.064 ± 0.005

(b) Search distances (↑)

(GSK3β)	SMILES	Molecule
Genetic GFN	0.740	0.528
LS-GFN	0.374	0.494
(JNK3)	SMILES	Molecule
Genetic GFN	0.706	0.536
LS-GFN	0.403	0.512

[6, 10]. It samples actions from a GFlowNet sampler mixed with a uniform distribution, similar to ϵ-greedy in RL. The results, shown in Table 2, reveal that the removal of either component results in a decline in performance, underscoring the importance of employing a suitable exploration strategy. Detailed results, including statistical analysis, are provided in Appendix G.4.

5.1.3 Comparisons with GFlowNets variants

We compare Genetic GFN with the graph-based GFlowNet [6], GFlowNet-AL [10], and the local search GFlowNet (LS-GFN) [18] using SMILES representations. LS-GFN utilizes Monte Carlo Markov Chain (MCMC) techniques, incorporating partial backtracking and reconstructing solution trajectories with the training policy as the proposal distribution [18]. The experiments are conducted on the PMO benchmark, and we implement LS-GFN with SMILES by replacing our genetic search with a local search. Note that while the original LS-GFN employs the prepend-append MDP, which does not directly apply to SMILES, we use the same one-directional SMILES generation as ours.

As shown in Table 3a, Genetic GFN outperforms other GFlowNet variants. Notably, generating SMILES is significantly more advantageous than generating graph-based fragments. The performance gap between Genetic GFN and LS-GFN highlights the importance of a proper exploratory policy. To further analyze, we measure the distance between samples before and after searches in GSK3β and JNK3. The normalized Levenshtein distances for SMILES and Tanimoto similarity for molecules are reported in Table 3b. The results show that the local search may be inefficient in effectively searching in moderate-scale chemical spaces because its capabilities heavily depend on the current policy, leading to suboptimal search performance. In contrast, our approach leverages a domain-specialized genetic search within the molecule graph space, working as an effective off-policy exploration — the samples with SMILES representation are used to train the string-based generative policy.

5.1.4 Sample efficient multi-objective molecular optimization

According to Zhu et al. [11], we apply our method to multi-objective tasks: GSK3β+JNK3 and GSK3β+JNK3+QED+SA. Notably, GSK3β and JNK3 are potential targets of Alzheimer's Disease treatments [50]. We use a linear combination of each objective with given coefficients, and the performance is measured by hypervolumes with 1K evaluations. We obtained the results from five independent trials using different seeds. Even though Genetic GFN is not designed for multi-objective molecular optimization, it demonstrates notable performance using proper scalar-valued score functions; please see Appendix E for details.

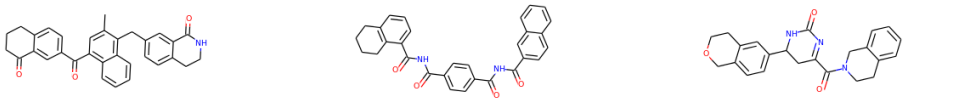
Table 4: Average and standard deviation of hypervolumes (↑) for each task. The baseline results are directly from the HN-GFN paper [11]. The **bold** text indicates the best value.

	GSK3β + JNK3	GSK3β + JNK3 + QED + SA
HierVAE+qParEGO	0.205 ± 0.015	0.186 ± 0.009
HierVAE+qEHVI	0.341 ± 0.072	0.211 ± 0.006
LaMOO	0.279 ± 0.090	0.190 ± 0.069
Graph GA	0.368 ± 0.020	0.335 ± 0.021
MARS	0.418 ± 0.095	0.273 ± 0.020
HN-GFN	0.669 ± 0.061	0.416 ± 0.023
Genetic GFN	**0.718 ± 0.138**	**0.642 ± 0.053**

5.1.5 Further analysis

Active learning with Genetic GFN. Similar to GFlowNet-AL, ours can work as a generative model in multi-round active learning. We compare Genetic GFN-AL with other model-based and active learning methods; please refer to Appendix D.

Genetic GFN with SELFIES representation. Genetic GFN with SELFIES generation achieves the improved sample efficiency to other SELFIES-based methods; see Appendix G.6.

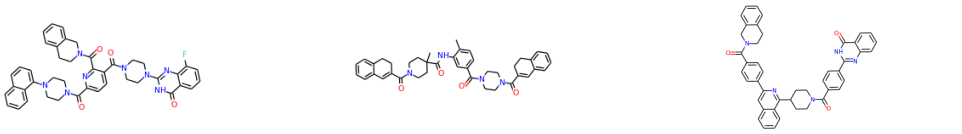


Score: 0.903
(docking: -12.7, QED: 0.383, SA: 2.720)

Score: 0.909
(docking: -12.6, QED: 0.412, SA: 2.251)

Score: 0.917
(docking: -12.0, QED: 0.859, SA: 3.376)

Figure 4: The final candidates for the PLPr_7JIR target with 100 steps.



Score: 0.892
(docking: -13.0, QED: 0.268, SA: 3.226)

Score: 0.908
(docking: -13.5, QED: 0.370, SA: 3.153)

Score: 0.894
(docking: -13.7, QED: 0.197, SA: 2.951)

Figure 5: The final candidates for the RdRp_6YYT target with 100 steps.

Sensitivity analysis. We provide the experimental results by varying the hyperparameters, such as the offspring size and the number of training loops; please refer to Appendix G.8.

5.2 Designing inhibitors against SARS-CoV-2 targets

In this subsection, we conduct drug discovery experiments for Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-Cov-2), known as the novel coronavirus. One desired property is maximizing the binding affinity to the target protein. The binding affinity is measured with a docking score, which is calculated based on the energies of the interaction between the ligand and the receptor. Typically, the computation of docking scores is expensive since it involves predicting the spatial orientation and binding affinity of the molecule in the active site of the target protein. We employ Quick Vina 2 [51] docking software to assess generated molecules.

Additionally, QED (Quantitative Estimate of Drug-likeness) and SA (Synthetic Accessibility) are considered to quantify the drug-likeness and difficulties of synthesizing. The higher QED, which ranges [0, 1], and the lower SA, which ranges [0, 10], are desired. Therefore, we define the score function $s(x)$ for designing SARS-Cov-2 inhibitors as a linear combination of normalized scores according to the previous work [52]. Following [53] and [52], the target proteins are selected: PLPro_7JIR, a critical enzyme in the life cycle of SARS-CoV-2, and RdRp_6YYT, which is essential for the replication and the transcription of genes.

The experiments are conducted with up to 1000 update steps with 128 batch size [52]. As shown in Table 5, ours achieves the highest Top-100 average scores only with 100 steps, which is 10 times fewer than others. Note that the score is recalculated based on the normalized score function in Eq. (5) using average values in the MolRL-MGPT paper [52]; the full results and a more detailed experimental setup are provided in Appendix F. We also report the best candidates of 100 steps in Fig. 4 and in Fig. 5. The final molecules correspond to the Top-1 score molecules from 3 independent runs.

Table 5: Average Top-100 scores (↑). Ours outperforms baselines with 10 times fewer steps. The **bold** denotes the best scores.

	PLPro	RdRp
JT-VAE	0.272	0.216
GFlowNet	0.326	0.280
Graph GA	0.723	0.786
REINVENT	0.717	0.799
MolRL-MGPT	0.772	0.854
Genetic GFN (100)	0.891	0.873
Genetic GFN (1000)	**0.925**	**0.902**

6 Discussion

This paper introduces a Genetic-guided GFlowNet (Genetic GFN), which integrates a domain-specific genetic algorithm to guide the GFlowNet policy toward higher-reward samples. The method employs off-policy training with a rank-based reweighted buffer, enhancing the policy as a powerful amortized inference sampler for chemical discovery. Extensive experiments demonstrate that Genetic GFN effectively generates desirable molecules within the high-dimensional chemical space, including long chemical structure sequences (e.g., ≥ 100). On the other hand, our approach can be considered as

https://doi.org/10.52202/079017-1350

a novel population reinitialization strategy for genetic algorithms using GFlowNets, which sample diverse objects proportional to rewards.

Limitations and future works. Our method assumes the existence of effective genetic algorithms, which is valid for the molecular design domain. However, designing domain-specific operators for genetic algorithms can be challenging in other fields. One possible future work is to enhance genetic algorithms using the neural policy similar to recent studies [54, 55]. Another direction is to extend our approach to other domains, such as combinatorial optimization. For instance, we could utilize a powerful GA, hybrid genetic search [56], to design a GFlowNet-based solver for routing problems.

Broader Impact. This paper introduces a new generative model, significantly enhancing sample efficiency in molecular optimization. This advance is likely to hold substantial promise for this field, potentially accelerating the development of new therapies and advanced materials. Our research is currently focused on *in-silico* experiments. The potential safety concerns of discovered molecules are further examined in the subsequent processes, such as *in-vitro* experiments and pre-clinical tests.

Acknowledgments and Disclosure of Funding

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00410082). The authors are grateful to Austin Tripp and Xiuyuan Hu for their help with baselines.

References

[1] James P Hughes, Stephen Rees, S Barrett Kalindjian, and Karen L Philpott. Principles of early drug discovery. *British Journal of Pharmacology*, 162(6):1239–1249, 2011.

[2] Alexander W Hains, Ziqi Liang, Michael A Woodhouse, and Brian A Gregg. Molecular semiconductors in organic photovoltaic cells. *Chemical Reviews*, 110(11):6689–6735, 2010.

[3] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9(1):1–14, 2017.

[4] 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*, 4(2):268–276, 2018.

[5] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pages 2323–2332. PMLR, 2018.

[6] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27381–27394, 2021.

[7] Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. In *International Conference on Machine Learning*, pages 18872–18892. PMLR, 2023.

[8] Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in Neural Information Processing Systems*, 35:21342–21357, 2022.

[9] 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.

[10] Moksh Jain, Emmanuel Bengio, Alex Hernández-Garcıa, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with GFlowNets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR, 2022.

[11] Yiheng Zhu, Jialu Wu, Chaowen Hu, Jiahuan Yan, Tingjun Hou, Jian Wu, et al. Sample-efficient multi-objective molecular optimization with GFlowNets. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[13] 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.

[14] AkshatKumar Nigam, Robert Pollice, Mario Krenn, Gabriel dos Passos Gomes, and Alan Aspuru-Guzik. Beyond generative models: superfast traversal, optimization, novelty, exploration and discovery (STONED) algorithm for molecules using SELFIES. *Chemical Science*, 12(20):7079–7090, 2021.

[15] AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alan Aspuru-Guzik. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. In *International Conference on Learning Representations*, 2020.

[16] Austin Tripp and José Miguel Hernández-Lobato. Genetic algorithms are strong baselines for molecule generation. *arXiv preprint arXiv:2310.09267*, 2023.

[17] Sungsoo Ahn, Junsu Kim, Hankook Lee, and Jinwoo Shin. Guiding deep molecular optimization with genetic exploration. In *Advances in Neural Information Processing Systems*, volume 33, pages 12008–12021, 2020.

[18] Minsu Kim, Taeyoung Yun, Emmanuel Bengio, Dinghuai Zhang, Yoshua Bengio, Sungsoo Ahn, and Jinkyoo Park. Local search GFlowNets. In *International Conference on Learning Representations*, 2024.

[19] David B Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, 1994.

[20] Hari Mohan Pandey, Ankit Chaudhary, and Deepti Mehrotra. A comparative review of approaches to prevent premature convergence in GA. *Applied Soft Computing*, 24:1047–1077, 2014.

[21] 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.

[22] Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf H Roohani, Jure Leskovec, Connor W. Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[23] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

[24] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. In *Advances in Neural Information Processing Systems*, volume 35, pages 5955–5967, 2022.

[25] Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pages 23467–23483. PMLR, 2023.

[26] Ling Pan, Nikolay Malkin, Dinghuai Zhang, and Yoshua Bengio. Better training of GFlowNets with local credit and incomplete trajectories. In *International Conference on Machine Learning*, pages 26878–26890. PMLR, 2023.

[27] Hyosoon Jang, Minsu Kim, and Sungsoo Ahn. Learning energy decompositions for partial inference of GFlowNets. In *International Conference on Learning Representations*, 2024.

 https://doi.org/10.52202/079017-1350

[28] Heiko Zimmermann, Fredrik Lindsten, Jan-Willem van de Meent, and Christian A Naesseth. A variational perspective on generative flow networks. *Transactions on Machine Learning Research*, 2023.

[29] David Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

[30] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[31] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

[32] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741, 2023.

[33] Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. In *Advances in Neural Information Processing Systems*, volume 33, pages 11259–11272, 2020.

[34] Minsu Kim, Federico Berto, Sungsoo Ahn, and Jinkyoo Park. Bootstrapped training of score-conditioned generator for offline design of biological sequences. In *Advances in Neural Information Processing Systems*, volume 36, pages 67643–67661, 2023.

[35] Naruki Yoshikawa, Kei Terayama, Masato Sumita, Teruki Homma, Kenta Oono, and Koji Tsuda. Population-based de novo molecule generation, using grammatical evolution. *Chemistry Letters*, 47(11):1431–1434, 2018.

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

[37] Moksh Jain, Tristan Deleu, Jason Hartford, Cheng-Hao Liu, Alex Hernandez-Garcia, and Yoshua Bengio. GFlowNets for AI-driven scientific discovery. *Digital Discovery*, 2(3):557–577, 2023.

[38] Max W Shen, Emmanuel Bengio, Ehsan Hajiramezanali, Andreas Loukas, Kyunghyun Cho, and Tommaso Biancalani. Towards understanding and improving GFlowNet training. In *International Conference on Machine Learning*, pages 30956–30975. PMLR, 2023.

[39] Pouya M Ghari, Alex Tseng, Gökcen Eraslan, Romain Lopez, Tommaso Biancalani, Gabriele Scalia, and Ehsan Hajiramezanali. Generative flow networks assisted biological sequence editing. In *NeurIPS 2023 Generative AI and Biology (GenBio) Workshop*, 2023.

[40] Miruna Cretu, Charles Harris, Julien Roy, Emmanuel Bengio, and Pietro Liò. SynFlowNet: Towards molecule design with guaranteed synthesis pathways. In *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*, 2024.

[41] Hyeonah Kim, Minsu Kim, Sungsoo Ahn, and Jinkyoo Park. Symmetric replay training: Enhancing sample efficiency in deep reinforcement learning for combinatorial optimization. In *Proceedings of the 41st International Conference on Machine Learning*, pages 24110–24136. PMLR, 2024.

[42] Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. GFlowNets and variational inference. In *International Conference on Learning Representations*, 2023.

[43] Sobhan Mohammadpour, Emmanuel Bengio, Emma Frejinger, and Pierre-Luc Bacon. Maximum entropy GFlowNets with soft Q-learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2593–2601. PMLR, 2024.

[44] Tristan Deleu, Padideh Nouri, Nikolay Malkin, Doina Precup, and Yoshua Bengio. Discrete probabilistic inference as control in multi-path environments. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

[45] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[46] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

[47] Jarrid Rector-Brooks, Kanika Madan, Moksh Jain, Maksym Korablyov, Cheng-Hao Liu, Sarath Chandar, Nikolay Malkin, and Yoshua Bengio. Thompson sampling for improved exploration in GFlowNets. In *ICML 2023 Structured Probabilistic Inference & Generative Modeling (SPIGM) Workshop*, 2023.

[48] Minsu Kim, Joohwan Ko, Taeyoung Yun, Dinghuai Zhang, Ling Pan, Woochang Kim, Jinkyoo Park, Emmanuel Bengio, and Yoshua Bengio. Learning to scale logits for temperature-conditional GFlowNets. In *International Conference on Machine Learning*, 2024.

[49] Shuai Guo, Jielei Chu, Lei Zhu, and Tianrui Li. Dynamic backtracking in GFlowNet: Enhancing decision steps with reward-dependent adjustment mechanisms. *arXiv preprint arXiv:2404.05576*, 2024.

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

[51] Amr Alhossary, Stephanus Daniel Handoko, Yuguang Mu, and Chee-Keong Kwoh. Fast, accurate, and reliable molecular docking with QuickVina 2. *Bioinformatics*, 31(13):2214–2216, 2015.

[52] Xiuyuan Hu, Guoqing Liu, Yang Zhao, and Hao Zhang. De novo drug design using reinforcement learning with multiple GPT agents. In *Advances in Neural Information Processing Systems*, volume 36, pages 7405–7418, 2023.

[53] David M Rogers, Rupesh Agarwal, Josh V Vermaas, Micholas Dean Smith, Rajitha T Rajeshwar, Connor Cooper, Ada Sedova, Swen Boehm, Matthew Baker, Jens Glaser, et al. SARS-CoV2 billion-compound docking. *Scientific Data*, 10(1):173, 2023.

[54] Tianfan Fu, Wenhao Gao, Connor Coley, and Jimeng Sun. Reinforced genetic algorithm for structure-based drug design. In *Advances in Neural Information Processing Systems*, volume 35, pages 12325–12338, 2022.

[55] AkshatKumar Nigam, Robert Pollice, and Alán Aspuru-Guzik. Parallel tempered genetic algorithm guided by deep neural networks for inverse molecular design. *Digital Discovery*, 1(4):390–404, 2022.

[56] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.

[57] Joshua Meyers, Benedek Fabian, and Nathan Brown. De novo molecular design and generative models. *Drug discovery today*, 26(11):2707–2715, 2021.

 https://doi.org/10.52202/079017-1350

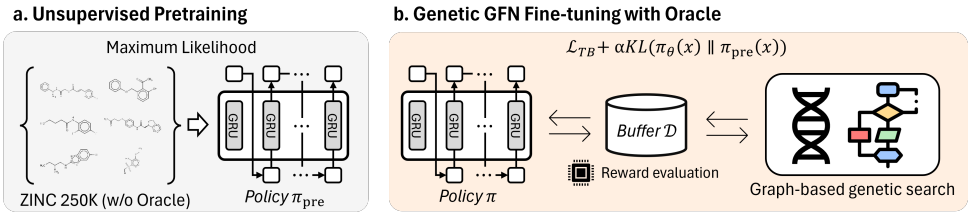


Figure 6: Pretraining and Genetic GFN fine-tuning framework

A Implementation details of Genetic GFN

Genetic GFN is implemented on top of the PMO benchmark source code (MIT license).[4] Mostly, we adopt the REINVENT implementation including the RNN models and experience buffer; the code is included in the benchmark, and the original implementation is also accessible with Apache-2.0 license.[5] See the following subsections for details.

A.1 Network architecture and pretraining

Network architecture. Our policy network is parameterized using a recurrent neural network containing multiple GRU cells [30]. In molecular optimization, RNN-based models with string molecular representations have proven to be successful [3, 17, 52]. In experiments, we employ the same hyperparameters to directly compare with REINVENT, whose input embedding dimension is 128 and hidden dimension is 512 with three layers.

Pretraining. According to the PMO benchmark guidelines [8], the pre-training is conducted on ZINC 250K. The overall framework is illustrated in Fig. 6. Since the network architecture is the same as REINVENT, we adopt the provided pretrained model for REINVENT in the PMO benchmark. This allows the direct comparison of fine-tuning approaches.

A.2 Hyperparameters

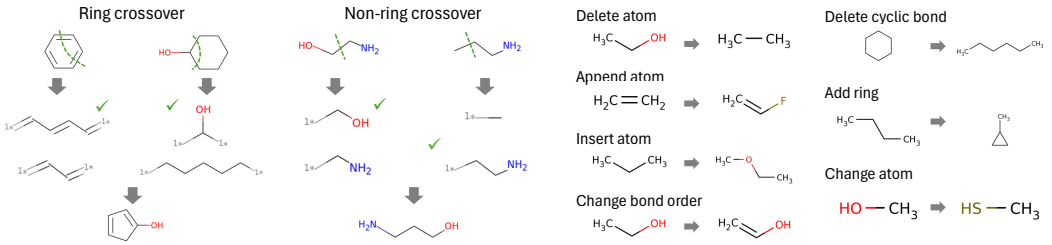
We mostly follow the hyperparmeter setup of REINVENT and GEGL. For instance, the batch size and learning rate are set as 64 and 0.0005 according to REINVENT in the PMO benchmark. On the other hand, the mutation rate and the number of training loops are set to 0.01 and 8 following GEGL. We use 64 samples for the replay training and population size, the same as the batch size without tuning. Lastly, the learning rate of Z, the partition function, is set to 0.1, also without tuning.

In contrast, we have searched several hyperparameters, offspring size, the number of GA generations, and KL-divergence coefficient α. We provide the sensitivity analysis for the offspring size and the number of GA generations in Appendix G.8. Furthermore, we use the inverse temperature $\beta = 10$ and the weight-shifting factor $k = 0.01$, but they can be differently used to control score-diversity trade-off, as explained in Section 5.1.2.

A.3 Computing resource

Throughout the experiments, we utilize a 48-core CPU, Intel(R) Xeon(R) Gold 5317 CPU @ 3.00GHz, and a single GPU. In the PMO benchmark, runtime varies from less than 10 minutes to several hours for 10K evaluations, depending on tasks and algorithms. However, most of the runtime is consumed in evaluating score functions—the motivation for why the sample efficiency matters. On the other hand, in the SARS-CoV-2 inhibitor design tasks, 1000 training steps with a batch size of 128 require more than 1 day for PlPro_7JIR and 2 days for RdRp_6YYT; more than 95% of the time is used to evaluation [52].

[4] https://github.com/wenhao-gao/mol_opt
[5] https://github.com/MolecularAI/Reinvent



(a) Crossover (b) Mutations

Figure 7: Examples of Graph GA operations. These operations are conducted according to predefined SMARTS patterns to ensure molecule validity, such as adherence to valence rules.

B Genetic operations

This section details each operation in our genetic search; see Fig. 7 for illustration. Note that we adopt Graph GA of [13], which has demonstrated its powerful performances and has been adopted by GA-related works like Mol GA [16] and GEGL [17].

B.1 Crossover

A crossover operation is conducted to generate a new candidate (called offspring) by exchanging the genetic information of a pair of selected individuals (parents). This process mimics the crossover of genetic material in biological reproduction. In the context of molecular optimization with graphs, the crossover operation is conducted in two types: 'ring crossover' and 'non-ring crossover with a 50% probability.

These two main crossover operations perform crossover between two parent molecules by cutting and recombining ring substructures. Ring crossover performs a ring cut specifically designed to target ring structures within the molecule. The ring-cut operation cuts the molecule along two different ring patterns, selected randomly. One of the ring patterns checks for a specific arrangement of four consecutive ring atoms, and the other pattern checks for a ring atom connected to two other ring atoms with a single bond. If a suitable ring pattern is found, it cuts the molecule along that pattern, resulting in two fragments. On the other hand, non-ring crossover cuts a single bond, meaning it is not part of a cyclic (ring) structure within the molecule. The obtained fragments from both parents are recombined to create new molecules by applying predefined reaction SMARTS patterns. These operations are repeated for validity to ensure that the resulting molecules meet structural and size constraints.

B.2 Mutation

The mutation is a random change that is introduced to the genetic information of some individuals. This step adds diversity to the population and helps explore new regions of the solution space. In this work, we employ seven different mutation processes and randomly select one of these mutations to modify the offspring molecules slightly. The operations consist of atom and bond deletions, appending new atoms, inserting atoms between existing ones, changing bond orders, deleting cyclic bonds, adding cyclic rings, and altering atom types.

1. Deletion of atom: it selects one of five deletion SMARTS patterns, each representing the removal of a specific number of atoms or bonds. These patterns include the removal of a single atom, a single bond, a bond with two attached atoms, and bonds with multiple attached atoms. The selected pattern is applied to the molecule, deleting the specified atom(s) or bond(s).
2. Appending atom: it introduces a new atom to the molecule. The type of atom (e.g., C, N, O) and the type of bond (single, double, or triple) are chosen based on predefined probabilities. The function then generates a reaction SMARTS pattern to append the selected atom to the molecule, forming a new bond.

https://doi.org/10.52202/079017-1350

3. Inserting atom: it inserts a new atom between two existing atoms in the molecule. Similar to the appending atom, it selects the type of atom and bond based on predefined probabilities and generates a reaction SMARTS pattern to insert the atom.
4. Changing bond order: it randomly selects one of four SMARTS patterns, each representing a change in the bond order between two atoms. These patterns include changing a single bond to a double bond, a double bond to a triple bond, and vice versa.
5. Deletion of cyclic bond: it deletes a bond that is a part of a cyclic structure within the molecule. The SMARTS pattern represents the breaking of a cyclic bond while retaining the atoms connected by the bond.
6. Adding ring: it introduces a new cyclic ring into the molecule by selecting one of four SMARTS patterns, each representing the formation of a specific ring type. These patterns create different types of cyclic structures within the molecule.
7. Changing atom: it randomly selects two types of atoms from a predefined list and generates a SMARTS pattern to change one atom into another. This operation modifies the atom type within the molecule.

C Hyperparameter setup for baseline methods

In this subsection, we present detailed descriptions for hyperparameter tuning of baselines. Except for Mol GA and GEGL, we adopt all the hyperparameters, initial datasets, and pre-trained models provided in the PMO benchmark. For hyperparameters that affect the sample efficiency, such as population size, we have searched for the proper hyperparameters following the guidelines suggested by [8]. In detail, we tune Mol GA and GEGL on the average AUC Top-10, the main performance metric, from 3 independent runs of two oracles, `zaleplon_mpo` and `perindopril_mpo`. The best configurations are used in the main experiments.

Mol GA. As mentioned in Section 5.1, we set the offspring size as 5, the most crucial hyperparameter, according to the original paper [16]. Then, we searched the starting population size in [100, 200, 500, 1000] and the population size [100, 200, 500]. As shown in Fig. 8, we found the best configuration to be 500 and 100 for the starting population size and the population size, respectively.

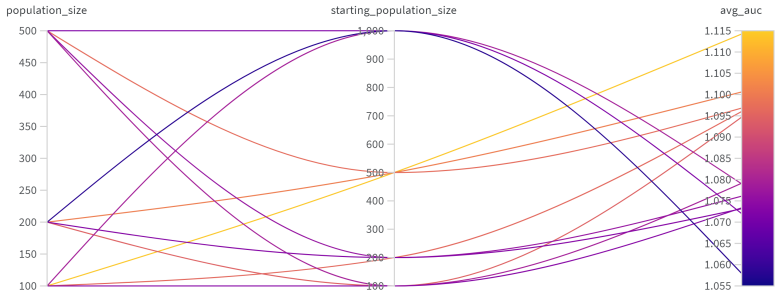


Figure 8: Hyperparameter tuning results for Mol GA

GEGL. In GEGL, the policy sampling size is set as the expert sampling size, and both priority queue sizes (denoted as `num_keep`) are the same as the original implementation. Thus, we searched the expert sampling size in [64, 128, 512] and the priority queue size in [128, 512, 1024]. Originally, they were set as 8192 and 1024, which are improper to the sample efficient setting. For the training batch size, we use 64, which is the same as ours. We use the pretrained policy provided in the original code and adapt the setup of the rest of the hyperparameters, including mutation rate, learning rate, and the number of training loops. Based on the results in Fig. 9, we set the expert sampling size and priority queue size as 128.

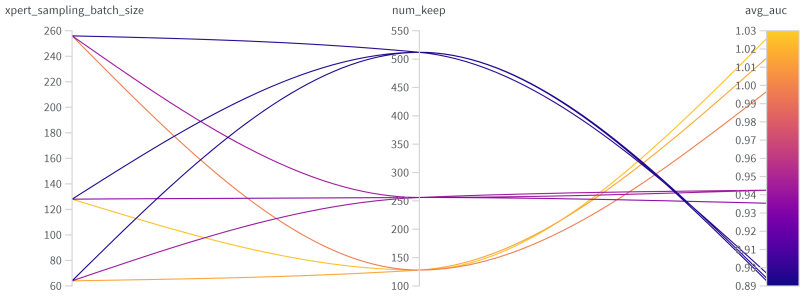


Figure 9: Hyperparameter tuning results for GEGL

https://doi.org/10.52202/079017-1350

D Comparison Genetic GFN-AL with active learning and model-based algorithms

D.1 Implementation of Genetic GFN-AL

We mostly adopt the implementation of GFlowNet-AL, including the training method and the utilization of an acquisition function [10]. The source code is accessible online with an MIT license.[6] Additionally, the number of training proxy and generative models are aligned with the PMO benchmark standards, while the replay training and genetic search hyperparameters are set to those used in our method in Genetic GFN. The pseudo-code is as follows.

Algorithm 2 Multi-round active learning with Genetic GFN

Input: Pretrained policy π_{pre}
Output: Top-K elements of discovered molecule dataset $\mathcal{D}$

Set $\pi_\theta \leftarrow \pi_{\text{pre}}$, $\mathcal{D} \leftarrow \emptyset$
Initialize the proxy model f_ϕ
while $|\mathcal{D}| \leq$ `numOracle` **do**
  ▷ PROXY TRAINING
  for $k = 1$ **to** `numTrainProxy` **do**
    Sample $\boldsymbol{x}$ from π_θ with `GeneticSearch`
    $y^{(i)} \leftarrow \mathcal{O}(\boldsymbol{x}^{(i)})$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=0}^{n}$
    Update ϕ to minimize $\sum_{(\boldsymbol{x},y)\in\mathcal{D}} (f_\phi(\boldsymbol{x}) - y)^2$
  end for
  ▷ GENERATIVE POLICY TRAINING
  $\mathcal{D}_{\text{inner}} \leftarrow \emptyset$
  for $l = 1$ **to** `numTrainPolicy` **do**
    Get $m' = \lceil m(1-\gamma) \rceil$ samples from π_θ with `GeneticSearch`
    Get $m - m'$ samples from $\mathcal{D}$
    $\mathcal{D}_{\text{inner}} \leftarrow \mathcal{D}_{\text{inner}} \cup \{(\boldsymbol{x}^{(i)}, \mathcal{F}(\mu(\boldsymbol{x}^{(i)}), \sigma(\boldsymbol{x}^{(i)}))\}_{i=0}^{m}$ ▷ Acquisition function from [10]
    for $r = 1$ **to** `numReplay` **do**
      Get $\mathcal{B}$ from $\mathcal{D}_{\text{inner}}$ with rank-based sampling
      Update θ to minimize $\frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x}, f_\phi(\boldsymbol{x}))\in\mathcal{B}} \mathcal{L}_{\text{TB}} + \alpha \text{KL}(\pi_\theta(\boldsymbol{x}) || \pi_{\text{pre}}(\boldsymbol{x}))$
    end for
  end for
end while

Hyperparmeters. We use the same hyperparameters for the generative model (i.e., Genetic GFN). Since active learning approaches introduce various hyperparameters, such as the training iterations of the proxy and generative models, not only introduce the proxy model, we tried to keep the setup of GFlowNet-AL in the PMO benchmark (e.g., proxy learning rate). We provide hyperparameters in Table 6; note that our proxy model predicts the score using SMILES rather than fragments, unlike the original GFlowNet-AL. Additionally, we adopt γ, the ratio of offline (oracle-touched) data in the inner loop training, and the acquisition function (related with κ); see the original GFlowNet-AL paper [10].

D.2 Experimental results

We compare Genetic GFN-AL with various model-based and active learning methods. Here, GP BO is regarded as a model-based method of Graph GA because GP BO uses Graph when optimizing the GP acquisition function. As shown in Table 7, SMILES Genetic GFN-AL achieves significantly improved performance compared to fragment GFlowNet-AL in the PMO benchmark. It is noteworthy that we further utilize the acquisition function, not directly use the proxy prediction as a reward, and mix the oracle-touched data, as described in the previous section. Therefore, to verify the effectiveness of genetic search in the active learning setup, we implement another baseline, SMILES GFN-AL using

[6] https://github.com/MJ10/BioSeq-GFN-AL

Table 6: Hyperparameters of Genetic GFN-AL

	SMILES Genetic GFN-AL	Fragment GFlowNet-AL (PMO benchmark)
proxy init sample size	480	500
proxy sample size	480	500
proxy training iterations	25	25
proxy training batch size	64	64
proxy hidden dimension	512	64
proxy layers	3	3
generative model training iterations	8×10	100
proxy learning rate	0.00025	0.00025
proxy weight decay	0.000001	0.000001
proxy dropout	0.0	0.0
kappa	0.1	0.0
gamma	0.5	0.0
random action prob	0.0	0.05

ϵ-greedy exploration, similar to our self-ablation studies in Section 5.1.2. The results demonstrate that Genetic GFN is beneficial to enhancing sample efficiency in the active learning setting. As pointed out in the PMO benchmark, though the model-based methods (or active learning methods) are known as more sample efficient, they require careful design to achieve superior performances.

Table 7: Comparing AL and model-based algorithms. The **bold** text indicates the best value.

	Genetic GFN-AL	SMILES GFN-AL with ϵ-greedy	GP BO [12]	Fragments GFlowNet-AL [10]
AUC Top-1	**13.997** $\pm$ 0.337	11.954 $\pm$ 0.190	13.718 $\pm$ 0.080	11.032 $\pm$ 0.016
AUC Top-10	**13.462** $\pm$ 0.303	11.257 $\pm$ 0.171	13.115 $\pm$ 0.074	9.928 $\pm$ 0.027
AUC Top-100	**12.263** $\pm$ 0.218	9.901 $\pm$ 0.237	12.050 $\pm$ 0.082	8.064 $\pm$ 0.005

https://doi.org/10.52202/079017-1350

E Multi-objective sample efficient molecular optimization tasks

This section provides details on multi-objective sample efficient molecular optimization experiments. Even though Genetic GFN targets single (i.e., scalar-valued) objective optimization tasks, ours directly applies to multi-objective tasks using well-defined coefficients. Inspired by the work of Zhu et al. [11], we conduct experiments on two multi-objective tasks: GSK3β + JNK3 and GSK3β + JNK3 + QED + SA. Genetic GFN is implemented on top of the original code (MIT license).[7]

Interestingly, GSK3β and JNK3 are machine-learning-based oracles that estimate inhibiting scores against the Glycogen synthase kinase 3 beta target and the c-Jun N-terminal kinase 3 target. Designing dual inhibitors for both targets can be beneficial to designing treatments for Alzheimer's Disease [50]. We define scalar-valued score functions using the linear combination of oracle functions with given coefficients. The cost coefficients α are set following the HN-GFN paper, i.e., $\alpha = (1, 1)$ for GSK3β + JNK3 and $\alpha = (3, 4, 2, 1)$ for GSK3β + JNK3 + QED + SA. Since the reward scales become larger, we reduce the inverse temperature from 50 to 25.

According to the experiment setup of hypernetwork-based GFlowNets (HN-GFN) [11], we limit the reward calls to 1000. Consequently, the batch size, replay training loops, population, and offspring size are adjusted to half. The performance is measured using hypervolumes with five independent runs. Note that we use the same pretrained model in the main experiment for PMO.

[7] https://github.com/violet-sto/HN-GFN

F Designing of SAS-Cov-2 inhibitors

According to the previous work [52], we define the score function $s(x)$ for designing SARS-Cov-2 inhibitors as a linear combination of normalized scores, i.e.,

$$s(x) = 0.8 \cdot \frac{1}{1 + 10^{0.625(s_{\text{docking}}(x)+10)}} + 0.1 \cdot s_{\text{QED}}(x) + 0.1 \cdot \frac{10 - s_{\text{SA}}(x)}{9}. \tag{5}$$

The target proteins are as follows:

- **PLPro_7JIR**: PLPro (papain-like protease) is a critical enzyme in the life cycle of SARS-CoV-2, which can help in studying the enzyme's function and in designing inhibitors that could potentially disrupt the virus's ability to replicate and evade the immune system. The 7JIR represents a C111S mutant version of PLPro.[8]
- **RdRp_6YYT**: RdRp (RNA-dependent RNA polymerase) is essential for the replication of the genome and the transcription of genes in SARS-CoV-2. The protein structure of RdRp is cataloged in the Protein Data Bank (PDB) under the identification code 6YYT.[9]

Genetic GFN is implemented on top of the implementation of MolRL-MGPT (Molecular design using Reinforcement Learning with Multiple GPT agents).[10] We employ the same hyperparameters with the experiments on the PMO benchmark, except for the weight-shifting factor (we use $k = 0.05$).

In Table 5, the scores of baselines are computed according to the Eq. (5) using the average values in the MolRL-MGPT [52]. We also provide average scores and standard deviation of docking, QED, SA, and diversity in Table 8 and Table 9.

Table 8: The results of Top-100 molecules for PLPr_7JIR target. The docking, QED, and SA of baselines are directly from MolRL-MGPT, and the total score is recalculated according to Eq. (5) using average values.

	Score (↑)	Docking (↓)	QED (↑)	SA (↓)	Diversity (↑)
JT-VAE	0.272	-8.76 ± 0.35	0.795 ± 0.038	2.994 ± 0.140	0.836 ± 0.032
GFlowNet	0.326	-9.11 ± 0.21	0.726 ± 0.015	2.823 ± 0.076	0.825 ± 0.010
Graph GA	0.723	-10.83 ± 0.08	0.380 ± 0.013	3.638 ± 0.162	0.740 ± 0.017
Reinvent	0.717	-10.75 ± 0.05	0.392 ± 0.008	2.649 ± 0.035	0.619 ± 0.023
MolRL-MGPT	0.772	-11.02 ± 0.06	0.386 ± 0.006	2.550 ± 0.047	0.745 ± 0.008
Genetic GFN (Ours)	0.908	-12.86 ± 0.17	0.425 ± 0.092	2.819 ± 0.105	0.592 ± 0.010

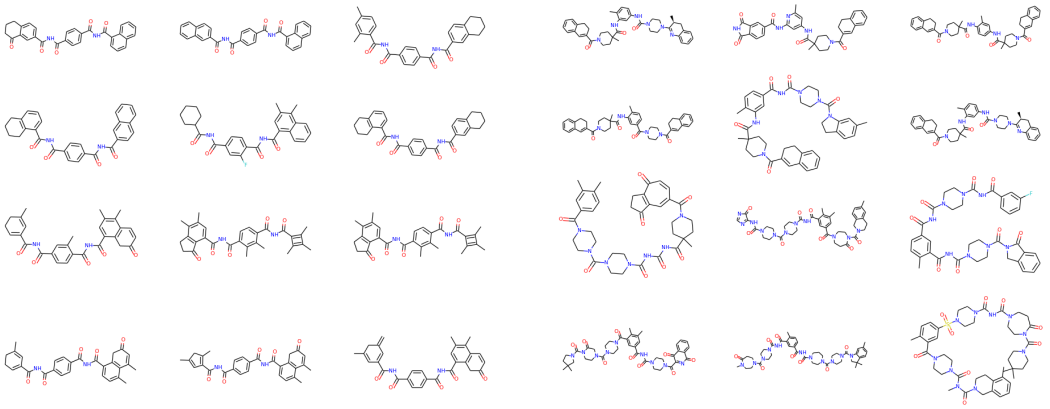
Table 9: The results of Top-100 molecules for RdRp_6YYT target. The docking, QED, and SA of baselines are directly from MolRL-MGPT, and the total score is recalculated according to Eq. (5) using average values.

	Score (↑)	Docking (↓)	QED (↑)	SA (↓)	Diversity (↑)
JT-VAE	0.216	-8.33 ± 0.25	0.719 ± 0.019	2.959 ± 0.094	0.828 ± 0.018
GFlowNet	0.280	-8.89 ± 0.16	0.656 ± 0.033	2.854 ± 0.061	0.770 ± 0.015
GraphGA	0.786	-11.26 ± 0.12	0.262 ± 0.010	3.520 ± 0.049	0.658 ± 0.009
Reinvent	0.799	-11.30 ± 0.04	0.275 ± 0.006	2.917 ± 0.035	0.616 ± 0.021
MolRL-MGPT	0.854	-11.84 ± 0.07	0.278 ± 0.005	2.894 ± 0.072	0.670 ± 0.013
Genetic GFN (Ours)	0.890	-13.26 ± 0.13	0.277 ± 0.076	3.624 ± 0.060	0.708 ± 0.010

We provide additional visual results in Fig. 10. There seems to be a trend of increasing molecular complexity and functional diversity over iterations.

[8] https://www.rcsb.org/structure/7JIR
[9] https://www.rcsb.org/structure/6YYT
[10] Available at https://github.com/HXYfighter/MolRL-MGPT

https://doi.org/10.52202/079017-1350



(a) PLPro_7JIR (b) RdRp_6YYT

Figure 10: Examples of Top3 inhibitors for SARS-CoV-2 over steps (`seed 1`)

G Additional results

G.1 Controllability of the score-diversity using the weight-shifting factor k

As mentioned in Section 5.1.2, the weight-shifting factor k in Eq. (4) also can control the score-diversity trade-off. Increasing k gives more diverse candidates by increasing the probability of high-ranked samples. Similar to adjusting the inverse temperature, the results in Table 10 and Fig. 11 demonstrate that adjusting k with fixed β also can effectively control the score-diversity trade-off.

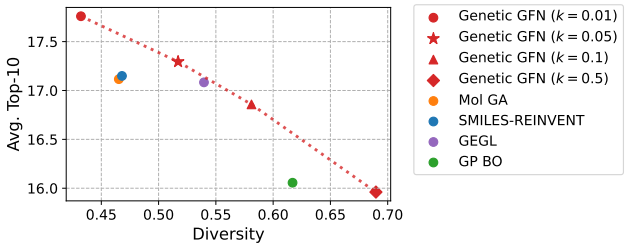


Figure 11: The average score and diversity with adjustments of k.

Table 10: The score-diversity trade-off by varying k with fixed β.

Oracle	$k = 0.1$	$k = 0.05$	$k = 0.01$	$k = 0.005$
AUC Top-1	15.246	15.801	**16.527**	16.323
AUC Top-10	14.652	15.330	**16.213**	16.040
AUC Top-100	13.597	14.453	**15.516**	15.418
Diversity	**0.581**	0.515	0.432	0.444

G.2 Additional results for the PMO benchmark

G.2.1 Statistical analysis

We provided the results of the t-tests of AUC Top-10 with Mol GA (2nd place) and SMIELS REINVENT (3rd place). As shown in Table 11, both p-values are less than 0.05, so Genetic GFN outperforms baselines with statistical significance.

Table 11: The results of t-tests with Mol GA and REINVENT.

	2nd place		3rd place	
	Genetic GFN	Mol GA	Genetic GFN	SMILES REINVENT
Mean	16.213	15.685	16.213	15.185
Variance	0.038	0.033	0.038	0.194
Observations	5	5	5	5
Hypothesized Mean Diff.	0		0	
t Stat	4.426		4.780	
P($T \leq t$) one-tail	**0.001**		**0.002**	
t Critical one-tail	1.860		2.015	
P($T \leq t$) two-tail	**0.002**		**0.005**	
t Critical two-tail	2.306		2.571	

https://doi.org/10.52202/079017-1350

G.2.2 Full results of the PMO benchmark

Table 12: Full results of Table 1.

	Genetic GFN (Ours)	Mol GA	SMILES REINVENT	GEGL	GP BO	SELFIES REINVENT	Graph GA
#1	**0.949 ± 0.010**	0.928 ± 0.015	0.881 ± 0.016	0.842 ± 0.019	0.902 ± 0.011	0.867 ± 0.025	0.859 ± 0.013
#2	**0.761 ± 0.019**	0.740 ± 0.055	0.644 ± 0.019	0.626 ± 0.018	0.579 ± 0.035	0.621 ± 0.015	0.657 ± 0.022
#3	**0.802 ± 0.029**	0.629 ± 0.062	0.717 ± 0.027	0.699 ± 0.041	0.746 ± 0.025	0.588 ± 0.062	0.593 ± 0.092
#4	0.733 ± 0.109	0.656 ± 0.013	0.662 ± 0.044	0.656 ± 0.039	0.615 ± 0.009	0.638 ± 0.016	0.602 ± 0.012
#5	**0.974 ± 0.006**	0.950 ± 0.004	0.957 ± 0.007	0.898 ± 0.015	0.941 ± 0.017	0.953 ± 0.009	0.973 ± 0.001
#6	**0.856 ± 0.039**	0.835 ± 0.012	0.781 ± 0.013	0.769 ± 0.009	0.726 ± 0.004	0.740 ± 0.012	0.762 ± 0.014
#7	0.881 ± 0.042	**0.894 ± 0.025**	0.885 ± 0.031	0.816 ± 0.027	0.861 ± 0.027	0.821 ± 0.041	0.817 ± 0.057
#8	**0.969 ± 0.003**	0.926 ± 0.014	0.942 ± 0.012	0.930 ± 0.011	0.883 ± 0.040	0.873 ± 0.041	0.949 ± 0.020
#9	**0.897 ± 0.007**	0.894 ± 0.005	0.838 ± 0.030	0.808 ± 0.007	0.805 ± 0.007	0.844 ± 0.016	0.839 ± 0.042
#10	0.764 ± 0.069	**0.835 ± 0.040**	0.782 ± 0.029	0.580 ± 0.086	0.611 ± 0.080	0.624 ± 0.048	0.652 ± 0.106
#11	**0.379 ± 0.010**	0.329 ± 0.006	0.363 ± 0.011	0.338 ± 0.016	0.298 ± 0.016	0.353 ± 0.006	0.285 ± 0.012
#12	0.294 ± 0.007	0.284 ± 0.035	0.281 ± 0.002	0.274 ± 0.007	**0.296 ± 0.011**	0.252 ± 0.010	0.255 ± 0.019
#13	0.708 ± 0.057	**0.762 ± 0.048**	0.634 ± 0.042	0.599 ± 0.035	0.631 ± 0.093	0.589 ± 0.040	0.571 ± 0.036
#14	**0.860 ± 0.008**	0.853 ± 0.005	0.834 ± 0.010	0.832 ± 0.005	0.788 ± 0.005	0.819 ± 0.005	0.813 ± 0.006
#15	0.595 ± 0.014	**0.610 ± 0.038**	0.535 ± 0.015	0.537 ± 0.015	0.494 ± 0.006	0.533 ± 0.024	0.514 ± 0.025
#16	**0.942 ± 0.000**	0.941 ± 0.001	0.941 ± 0.000	0.941 ± 0.001	0.937 ± 0.002	0.940 ± 0.000	0.937 ± 0.001
#17	0.819 ± 0.018	**0.830 ± 0.010**	0.770 ± 0.005	0.730 ± 0.011	0.741 ± 0.010	0.736 ± 0.008	0.718 ± 0.017
#18	**0.615 ± 0.100**	0.568 ± 0.017	0.551 ± 0.024	0.531 ± 0.010	0.535 ± 0.007	0.521 ± 0.014	0.513 ± 0.026
#19	0.634 ± 0.039	**0.677 ± 0.055**	0.470 ± 0.041	0.402 ± 0.024	0.461 ± 0.057	0.492 ± 0.055	0.498 ± 0.048
#20	**0.583 ± 0.034**	0.544 ± 0.067	0.544 ± 0.026	0.515 ± 0.028	0.544 ± 0.038	0.497 ± 0.043	0.483 ± 0.034
#21	**0.511 ± 0.054**	0.487 ± 0.024	0.458 ± 0.018	0.420 ± 0.031	0.404 ± 0.025	0.342 ± 0.022	0.373 ± 0.013
#22	0.135 ± 0.271	0.000 ± 0.000	**0.182 ± 0.363**	0.119 ± 0.238	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
#23	**0.552 ± 0.033**	0.514 ± 0.033	0.533 ± 0.009	0.492 ± 0.021	0.466 ± 0.025	0.509 ± 0.009	0.468 ± 0.025
Sum	**16.213**	15.686	15.185	14.354	14.264	14.152	14.131
Div.	0.432	0.465	0.468	0.540	0.617	0.555	0.661

Table 13: Full results of Table 1 (continued).

	SMILES LSTM-HC	STONED	SynNet	SMEILS GA	Fragment GFN	Fragment GFN-AL
#1	0.731 ± 0.008	0.765 ± 0.048	0.568 ± 0.033	0.649 ± 0.079	0.382 ± 0.010	0.459 ± 0.028
#2	0.598 ± 0.021	0.608 ± 0.020	0.566 ± 0.006	0.520 ± 0.017	0.428 ± 0.002	0.437 ± 0.007
#3	0.552 ± 0.014	0.378 ± 0.043	0.439 ± 0.035	0.361 ± 0.038	0.263 ± 0.009	0.326 ± 0.008
#4	**0.837 ± 0.018**	0.612 ± 0.007	0.635 ± 0.043	0.612 ± 0.005	0.582 ± 0.001	0.587 ± 0.002
#5	0.941 ± 0.005	0.935 ± 0.014	0.970 ± 0.006	0.958 ± 0.015	0.480 ± 0.075	0.601 ± 0.055
#6	0.733 ± 0.002	0.791 ± 0.014	0.750 ± 0.016	0.705 ± 0.025	0.689 ± 0.003	0.700 ± 0.005
#7	0.846 ± 0.019	0.666 ± 0.022	0.713 ± 0.057	0.714 ± 0.038	0.589 ± 0.009	0.666 ± 0.006
#8	0.830 ± 0.019	0.930 ± 0.012	0.862 ± 0.004	0.821 ± 0.070	0.791 ± 0.024	0.468 ± 0.211
#9	0.693 ± 0.018	**0.897 ± 0.032**	0.657 ± 0.030	0.853 ± 0.049	0.576 ± 0.021	0.199 ± 0.199
#10	0.670 ± 0.014	0.509 ± 0.065	0.574 ± 0.103	0.353 ± 0.061	0.359 ± 0.009	0.442 ± 0.017
#11	0.263 ± 0.007	0.264 ± 0.032	0.236 ± 0.015	0.187 ± 0.029	0.192 ± 0.003	0.207 ± 0.003
#12	0.249 ± 0.003	0.254 ± 0.024	0.241 ± 0.007	0.178 ± 0.009	0.174 ± 0.002	0.181 ± 0.002
#13	0.553 ± 0.040	0.620 ± 0.098	0.402 ± 0.017	0.419 ± 0.028	0.291 ± 0.005	0.332 ± 0.012
#14	0.801 ± 0.002	0.829 ± 0.012	0.793 ± 0.008	0.820 ± 0.021	0.787 ± 0.002	0.785 ± 0.003
#15	0.491 ± 0.006	0.484 ± 0.016	0.541 ± 0.021	0.442 ± 0.016	0.423 ± 0.006	0.434 ± 0.006
#16	0.939 ± 0.001	**0.942 ± 0.000**	0.941 ± 0.001	0.941 ± 0.002	0.904 ± 0.002	0.917 ± 0.002
#17	0.728 ± 0.005	0.764 ± 0.023	0.749 ± 0.009	0.723 ± 0.023	0.626 ± 0.005	0.660 ± 0.004
#18	0.529 ± 0.004	0.515 ± 0.025	0.506 ± 0.012	0.507 ± 0.008	0.461 ± 0.002	0.464 ± 0.003
#19	0.309 ± 0.015	0.600 ± 0.103	0.297 ± 0.033	0.449 ± 0.068	0.180 ± 0.012	0.217 ± 0.022
#20	0.440 ± 0.014	0.375 ± 0.029	0.397 ± 0.012	0.310 ± 0.019	0.261 ± 0.004	0.292 ± 0.009
#21	0.369 ± 0.015	0.309 ± 0.030	0.280 ± 0.006	0.262 ± 0.019	0.183 ± 0.001	0.190 ± 0.002
#22	0.011 ± 0.021	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
#23	0.470 ± 0.004	0.484 ± 0.015	0.493 ± 0.014	0.470 ± 0.029	0.308 ± 0.027	0.353 ± 0.024
Sum	13.583	13.531	12.610	12.254	9.929	9.917
Div.	0.686	0.498	0.728	0.596	0.816	0.846

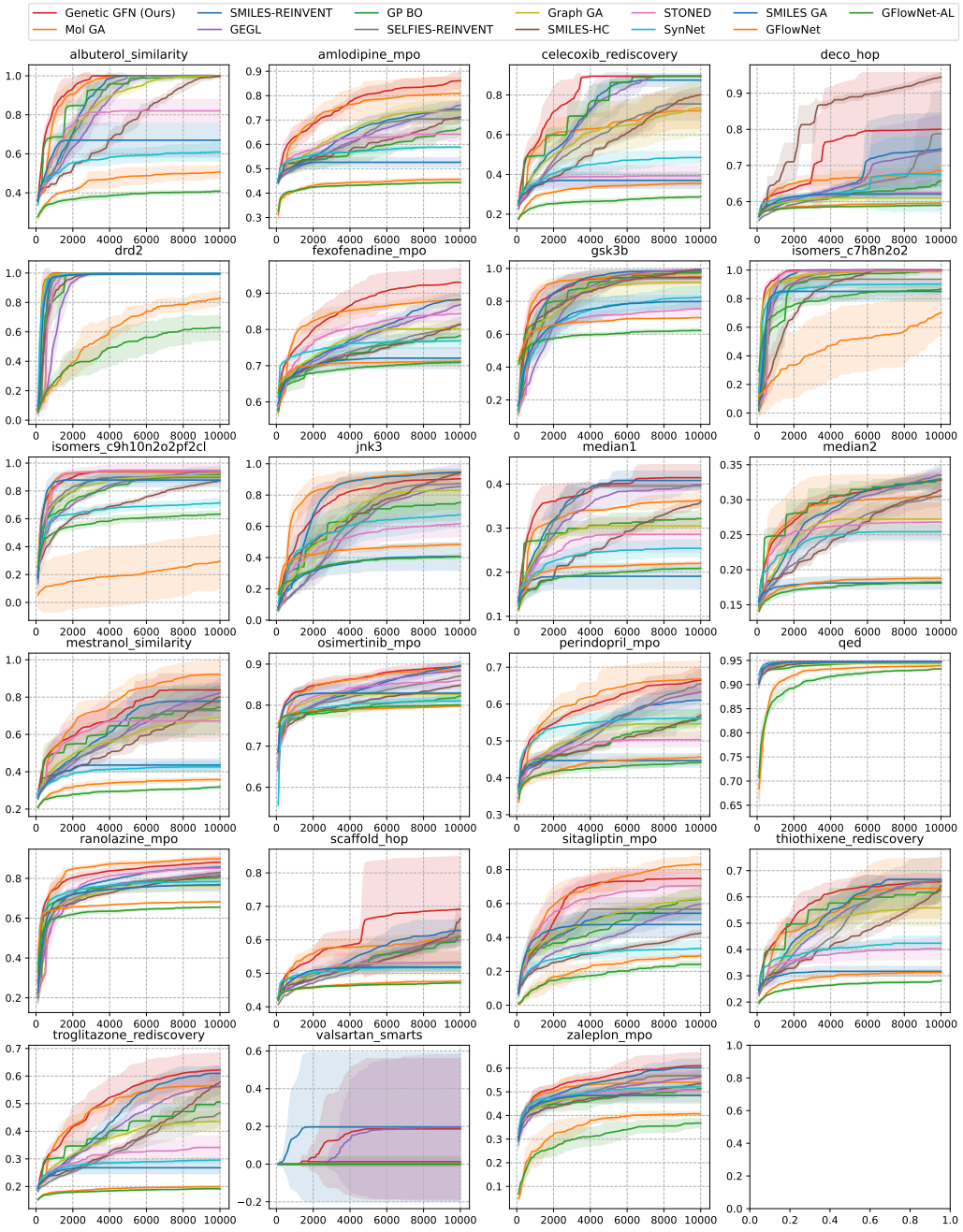


Figure 12: The optimization curves for 23 oracle score functions.

https://doi.org/10.52202/079017-1350

G.2.3 Ranks with various metrics

According to the PMO benchmark, we also provide the rank of each method with various metrics. The results in Table 14 show that Genetic GFN achieves the first place in total, not only in the AUC Top-10.

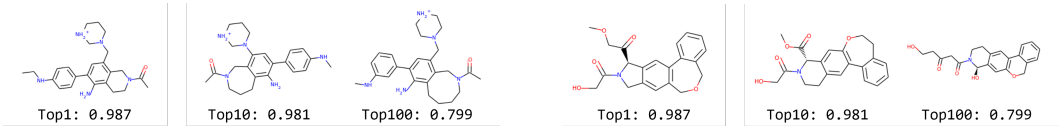
Table 14: The ranks of 10 methods based on various performance metrics

Method	Category	AUC			Average score			Mean
		Top-1	Top-10	Top-100	Top-1	Top-10	Top-100	
Genetic GFN (Ours)	Off-policy	**1**	**1**	**1**	**1**	**1**	**1**	**1.00**
Mol GA [16]	GA	2	2	2	5	3	2	2.67
SMILES-REINVENT [3]	On-policy	3	3	3	4	2	3	3.00
GEGL [17]	Off-policy	4	4	6	3	4	4	4.17
SELFIES-REINVENT [3]	On-policy	7	6	4	6	6	5	5.67
GP BO [12]	Active Learning	6	5	5	7	7	6	6.00
SMILES-LSTM-HC [21]	Off-policy	5	8	10	2	5	7	6.17
Graph GA [13]	GA	8	7	7	8	8	8	7.67
STONED [14]	GA	9	9	8	9	9	9	8.83
SynNet [36]	GA	10	10	11	10	10	11	10.33
SMILES GA [21]	GA	11	11	9	11	11	10	10.50
GFlowNet [6]	Off-policy	13	13	12	12	12	12	12.33
GFlowNet-AL [10]	Active Learning	12	12	13	13	13	13	12.67

G.3 Further studies on `valsartan_smarts (#22)`

Notably, we have observed that only a few methods achieve non-zero scores on `valsartan_smarts (#22)` in Table 1. The valsartan SMARTS targets molecules containing a SMARTS pattern related to valsartan while being characterized by physicochemical properties corresponding to the sitagliptin molecule [57]. It measures the arithmetic means of several scores, including (1) binary score about whether it contains a certain SMARTS structure, (2) LogP, (3) TPSA, and (4) Bertz score. Since we utilize a TDC oracle function for evaluations, we provide our empirical observations here.

Due to the binary score (1 if the certain SMARTS pattern is included), many tries terminate with 0. Especially with a limited number of oracle calls, generating molecules containing a certain substructure is notoriously hard. Other literature shows that other methods achieve high scores with more oracle calls [52]. With 10K calls, even REINVENT and Genetic only succeed in finding non-zero score molecules once out of five independent runs. Another observation is that methods (REINVENT, Genetic GFN, and GEGL) achieving non-zero scores all generate SMILES with RNN-based models. Thus, we have a conjecture that SMILES generation is effective in generating a certain SMARTS pattern. We provide examples of generated molecules with non-zero `valsartan_smarts` scores. Note that the other four seeds failed. Each run generates similar molecules (see Top1,10,100 samples in Fig. 13 in the additional material), but the samples between the two runs (REINVENT and Genetic GFN) have different structures (the molecule distance between Top1 samples is 0.854).



(a) REINVENT (`seed 0`) (b) Genetic GFN (`seed 4`)

Figure 13: Examples of molecules with non-zero scores on `valsartan_smarts (#22)`

G.4 Additional results for ablation studies

G.4.1 Statistical analysis

We provided the results of the t-tests of ablation studies. As shown in Table 15, the p-values for ablating genetic search are less than 0.05, so our genetic search is a statistically significant component.

Table 15: The results of t-tests of ablation studies.

	Genetic Search				KL-divergence penalty	
	Genetic GFN	- {GS}	Genetic GFN	- {GS} + {ϵ-greedy}	Genetic GFN	- {KL}
Mean	16.213	15.738	16.213	15.627	16.213	15.928
Variance	0.038	0.094	0.038	0.008	0.038	0.227
Observations	5	5	5	5	5	5
Hypothesized Mean Diff.	0		0		0	
t Stat	2.931		6.109		1.236	
P($T \leq t$) one-tail	**0.011**		**0.000**		**0.136**	
t Critical one-tail	1.895		1.943		2.015	
P($T \leq t$) two-tail	**0.021**		**0.001**		**0.271**	
t Critical two-tail	2.365		2.447		2.571	

G.5 Results for diversity and synthesizability

We report the diversity and synthetic accessibility (SA) score of Top-100 molecules on each oracle.

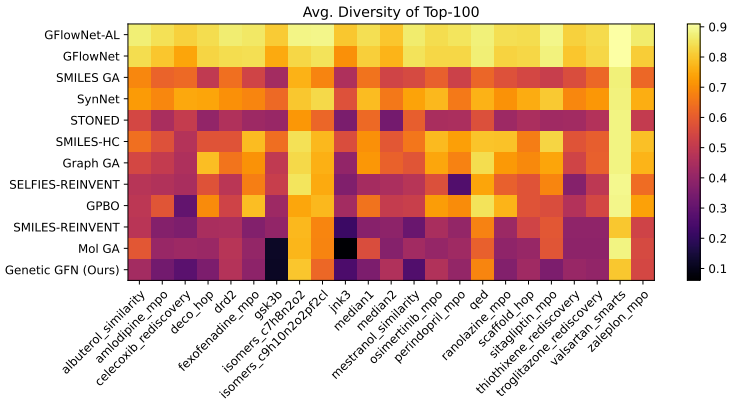


Figure 14: The average diversity of Top-100 molecules (↑)

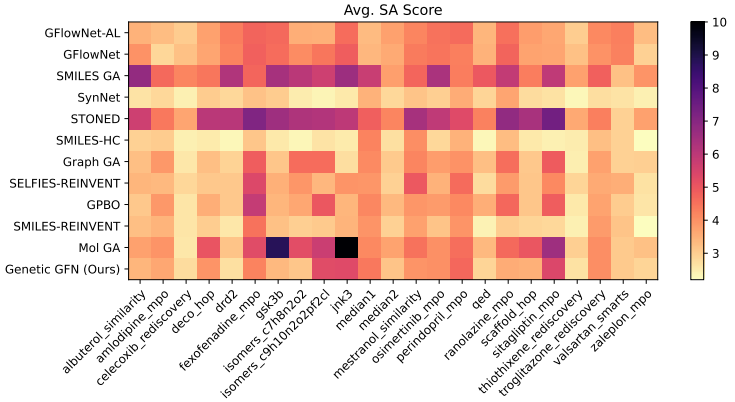


Figure 15: The average SA score of Top-100 molecules (↓)

https://doi.org/10.52202/079017-1350

G.6 Genetic GFN with SELFIES generation

As our method employs a string-based sequence generation model, it is applicable to SELFIES representation. Following the same procedure and setup described in Section 3 and Section 5.1. The results in Table 16 demonstrate that Genetic GFN significantly outperforms SELFIES-REINVENT by achieving 14.986 compared to 14.152, not only the other high-ranked method, GP BO (14.264). Notably, our AUC Top-10 is higher than that of SELFIES-REINVENT in 19 oracles.

Table 16: Performance of Genetic GFN with SELFIES generations.

Oracle	SELFIES REINVENT	SELFIES Genetic GFN
albuterol_similarity	0.867 ± 0.025	0.918 ± 0.031
amlodipine_mpo	0.621 ± 0.015	0.711 ± 0.029
celecoxib_rediscovery	0.588 ± 0.062	0.578 ± 0.049
deco_hop	0.638 ± 0.016	0.631 ± 0.020
drd2	0.953 ± 0.009	0.971 ± 0.005
fexofenadine_mpo	0.740 ± 0.012	0.797 ± 0.012
gsk3b	0.821 ± 0.041	0.900 ± 0.042
isomers_c7h8n2o2	0.873 ± 0.041	0.952 ± 0.017
isomers_c9h10n2o2pf2cl	0.844 ± 0.016	0.879 ± 0.031
jnk3	0.624 ± 0.048	0.675 ± 0.140
median1	0.353 ± 0.006	0.351 ± 0.034
median2	0.252 ± 0.010	0.263 ± 0.014
mestranol_similarity	0.589 ± 0.040	0.680 ± 0.076
osimertinib_mpo	0.819 ± 0.005	0.849 ± 0.008
perindopril_mpo	0.533 ± 0.024	0.551 ± 0.015
qed	0.940 ± 0.000	0.942 ± 0.000
ranolazine_mpo	0.736 ± 0.008	0.785 ± 0.013
scaffold_hop	0.521 ± 0.014	0.531 ± 0.020
sitagliptin_mpo	0.492 ± 0.055	0.590 ± 0.018
thiothixene_rediscovery	0.497 ± 0.043	0.527 ± 0.036
troglitazone_rediscovery	0.342 ± 0.022	0.387 ± 0.087
valsartan_smarts	0.000 ± 0.000	0.000 ± 0.000
zaleplon_mpo	0.509 ± 0.009	0.518 ± 0.016
Sum	14.152	14.986
Diversity	0.555	0.528

G.7 Genetic GFN with string-based genetic search

We also additionally provide experiments that incorporate STONED (GA with SELFIES)[14] as an exploration strategy to guide GFN training instead of Graph GA. Note that STONED only utilizes mutations since designing valid crossover with string representation is challenging.

Table 17: Results with different genetic search algorithms

	Genetic GFN	Genetic GFN with STONED
AUC Top-1	16.527 ± 0.043	15.806 ± 0.037
AUC Top-10	16.213 ± 0.042	15.439 ± 0.037
AUC Top-100	15.516 ± 0.041	14.870 ± 0.036

G.8 Experiments with varying hyperparameters

In this subsection, we verify the robustness of Genetic GFN for differing hyperparameter setups. We conduct experiments by varying the number of GA generation (refining loops), offspring size, and the number of training inner loops. The results show that our results are robust to each hyperparameter setup by achieving similar or better performance compared to Mol GA in all tested configurations.

Table 18: Ablation studies for the number of GA generation. '$\times 0$' stands for the results of TB loss training without GA explorations.

Oracle	$\times 0$	$\times 1$	$\times 2$	$\times 3$
AUC Top-1	16.070	15.968	**16.527**	16.040
AUC Top-10	15.738	15.615	**16.213**	15.735
AUC Top-100	15.030	14.909	**15.516**	15.074
Diversity	**0.479**	0.470	0.432	0.440

Table 19: Results by varying the offspring size.

Oracle	4	8	16	32
AUC Top-1	16.174	**16.527**	15.984	15.963
AUC Top-10	15.846	**16.213**	15.669	15.621
AUC Top-100	15.175	**15.516**	14.977	14.858
Diversity	**0.458**	0.432	0.452	0.437

Table 20: Results by varying the number of training inner loops.

Oracle	$\times 4$	$\times 8$	$\times 16$
AUC Top-1	16.125	**16.527**	16.049
AUC Top-10	15.768	**16.213**	15.717
AUC Top-100	15.175	**15.516**	14.977
Diversity	0.423	0.432	**0.511**

https://doi.org/10.52202/079017-1350

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our main research claim, which is consistent with the experimental results.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include our limitations and future works in the discussion section.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide implementation details, including pseudo-codes.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the anonymized link of our implementation. Also, the paper uses the public dataset and benchmark; we provide accessible links in the supplemental material.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all details regarding experiments in the appendix and the anonymized codes.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The experimental results include standard deviation with independent multiple runs. In addition, statistical tests are conducted for the main results and self-ablation studies.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

Justification: We provide the computer resources in the appendix (due to lack of spaces).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We have checked the paper according to the NeurIPS Code of Ethics; our study does not include human participation and privacy-related data.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impact of this study at the end of the paper.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our study is conducted using the public dataset, and all experimental tasks are from previous publications.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper clearly states the source and license of the original code, data, and models in the appedix.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Our experiments are conducted using the already published datasets and benchmarks.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our study does not involve crowdsourcing nor human subjects.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our study does not involve crowdsourcing nor human subjects.

https://doi.org/10.52202/079017-1350