Efficient Evolutionary Search Over Chemical Space with Large Language Models

Haorui Wang[*,1], **Marta Skreta**[*,2,3], **Cher-Tian Ser**[2], **Wenhao Gao**[4], **Lingkai Kong**[1], **Felix Strieth-Kalthoff**[5], **Chenru Duan**[6], **Yuchen Zhuang**[1], **Yue Yu**[1], **Yanqiao Zhu**[7], **Yuanqi Du**[†,8], **Alán Aspuru-Guzik**[†,2,3], **Kirill Neklyudov**[†,9,10], **Chao Zhang**[†,1]
[1]Georgia Institute of Technology, [2]University of Toronto, [3]Vector Institute,
[4]Massachusetts Institute of Technology, [5]University of Wuppertal, [6]Deep Principle Inc.,
[7]University of California, Los Angeles [8]Cornell University,
[9]Université de Montréal, [10]Mila - Quebec AI Institute
hwang984@gatech.edu, martaskreta@cs.toronto.edu

Abstract

Molecular discovery, when formulated as an optimization problem, presents significant computational challenges because optimization objectives can be non-differentiable. Evolutionary Algorithms (EAs), often used to optimize black-box objectives in molecular discovery, traverse chemical space by performing random mutations and crossovers, leading to a large number of expensive objective evaluations. In this work, we ameliorate this shortcoming by incorporating chemistry-aware Large Language Models (LLMs) into EAs. Namely, we redesign crossover and mutation operations in EAs using LLMs trained on large corpora of chemical information. We perform extensive empirical studies on both commercial and open-source models on multiple tasks involving property optimization, molecular rediscovery, and structure-based drug design, demonstrating that the joint usage of LLMs with EAs yields superior performance over all baseline models across single- and multi-objective settings. We demonstrate that our algorithm improves both the quality of the final solution and convergence speed, thereby reducing the number of required objective evaluations. Our code is available at https://github.com/zoom-wang112358/MOLLEO.

1 Introduction

Molecular discovery is a complex and iterative process involving the design, synthesis, evaluation, and refinement of molecule candidates. This process is often slow and laborious, making it difficult to meet the increasing demand for new molecules in domains such as pharmaceuticals, optoelectronics, and energy storage [71]. One significant challenge is that evaluating molecular properties often requires expensive evaluations (oracles), such as wet-lab experiments, bioassays, and computational simulations [25, 68]. Even approximate computational evaluations require substantial resources [25]. Consequently, the development of efficient algorithms for molecular search, prediction, and generation has gained traction in chemistry to accelerate the discovery process. These advancements in computational techniques, particularly machine learning-driven methods, have facilitated the rapid identification and proposal of promising molecular candidates for real-world experiments [43, 3, 14].

Several current approaches used to generate molecular candidates are based on Evolutionary Algorithms (EAs) [33], which do not require the evaluation of gradients and are thus well-suited for black-box objectives in molecular discovery. However, a major downside is that they generate pro-

[*] Equal first-authorship
[†] Equal senior-authorship
Preprint. Under review.

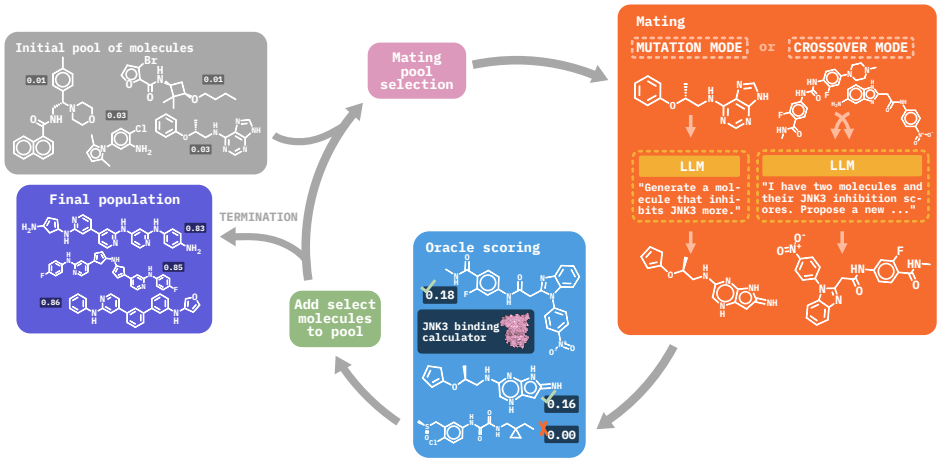


Figure 1: Overview of MOLLEO. Given an initial pool of molecules, mates are selected using default Graph-GA [39] heuristics. LLMs then function as mutation or crossover operators, editing the molecules based on text prompts that describe the target objective(s). The offspring molecules are then evaluated using an oracle, and the best-scoring ones are passed to the next generation. This process is repeated until the maximum number of allowed molecule evaluations is performed.

posals randomly without leveraging task-specific information. Consequently, producing reasonable candidates requires numerous evaluations of the objective function, limiting the practical application of these algorithms. Thus, proposals generated by operators that incorporate task-specific information can help reduce the number of evaluations required to optimize the objective function.

Natural language processing (NLP) has increasingly been utilized to represent molecular structures [9, 64, 57] and extract chemical knowledge from literature [73]. The connection between NLP and molecular systems is facilitated by molecular representations such as the Simplified Molecular Input Line Entry System (SMILES) and Self-Referencing Embedded Strings (SELFIES) [76, 13, 42]. These methods convert 2D molecular graphs into text, allowing molecular structures to be represented in the same modality as their textual descriptions.

Recently, the performance of Large Language Models (LLMs) has been investigated in several chemistry-related tasks, such as predicting molecular properties [32, 38], retrieving optimal molecules [43, 61, 80], automating chemistry experiments [8, 7, 81, 12], and generating molecules with target properties [20, 50, 80]. Because LLMs have been trained on large corpora of text that include a wide range of tasks, they demonstrate general-purpose language comprehension as well as knowledge of basic chemistry, making them interesting tools for chemical discovery tasks [77]. However, many LLM-based approaches depend on in-context learning and prompt engineering [32]. This can pose issues when designing molecules with strict numerical objectives, as LLMs may struggle to satisfy precise numerical constraints or optimize for specific numerical targets [2]. Furthermore, methods that solely depend on LLM prompting may produce molecules with lower fitness due to a lack of physical grounding, or they may produce invalid SMILES strings that cannot be decoded into chemical structures [66].

In this work, we propose **Mol**ecular **L**anguage-Enhanced **E**volutionary **O**ptimization (MOLLEO), which incorporates LLMs into EAs to enhance the quality of generated proposals and accelerate the optimization process (see Figure 1). MOLLEO leverages LLMs as genetic operators to produce new proposals through crossover or mutation. To our knowledge, this is the first demonstration of how LLMs can be incorporated into EA frameworks for molecular generation. In this work, we consider three LLMs: GPT-4 [1], BioT5 [58], and MoleculeSTM (MolSTM) [49]. We integrate each LLM into separate crossover and mutation procedures, justifying our design choices through ablation studies. We empirically demonstrate the superior performance of MOLLEO across multiple black-box optimization tasks, including single-objective and multi-objective optimization. For all tasks, including more challenging ones like protein-ligand docking, MOLLEO outperforms the baseline EA and other optimization algorithms based on reinforcement learning (RL) and Bayesian Optimization (BO). To further illustrate how our model can be used in novel molecular discovery settings, we show that MOLLEO can improve on the best existing JNK3 inhibitor molecules in ZINC 250K [67].

2 Related Work

2.1 Molecular Optimization

The molecular design field, encompassing multiple fundamental problems in chemistry, has developed numerous methods. In general, all the existing approaches define the space of possible molecular structures and run a combinatorial search to find the molecule with the target properties. Namely, conventional methods include Monte Carlo Tree Search (MCTS) [79], Reinforcement Learning (RL) [55, 30], and Genetic Algorithms (GA) [39, 22, 54, 21].

Due to existing challenges such as searching through exponentially large chemical space and evaluating expensive objectives [6, 69], conventional algorithms have recently recoursed to machine learning techniques, especially generative modelling [14]. Generative models learn a probability distribution of the observed data which can be later used to propose new molecular structures, thereby concentrating the search space around valid molecular structures. Depending on the type of the data and necessary properties for the search algorithms, different generative models have been considered: autoregressive models (ARs) [59, 24], variational autoencoders (VAEs) [27, 40], flow-based models [52, 65], diffusion models [34, 63].

Despite concentrating the search space around valid molecules by the usage of generative modeling, the optimization of necessary properties can remain infeasible. To narrow down the search space further, one can consider the conditional generative modeling, where the molecular structures are sampled from the conditional distribution having some predefined properties [27, 29, 83, 15, 75]. In this paper, we demonstrate the use of chemistry-aware LLMs as conditional generative models that improve the efficiency of combinatorial search in the molecular space.

2.2 Language Models in Chemistry

LLMs have been widely investigated for their applicability in scientific domains [1, 2], as well as their ability to leverage chemistry tools for chemical discovery and characterization [8, 7]. Several works have benchmarked LLMs such as GPT-4 on chemistry tasks and found that while LLMs can outperform human chemists in some zero-shot question-answering settings, they still struggle with chemical reasoning [53, 32]. Several smaller, open-source models have been trained or fine-tuned specifically on chemistry text [70, 10, 58].

Recently, language models have also been used to guide a given input molecular structure towards specific objective properties; a widely-used term used for this is *molecular editing* [49, 80]. Modifying structures towards specified properties is important so that they can satisfy potentially many required criteria, a requirement in pharmaceutical development where molecules need to be nontoxic and effective against their target (among other things), or in battery design, where molecules need to have a large energy capacity and a long lifespan. In this paper, we focus on molecular optimization to find molecules with desired properties, rather than editing. For interested readers, we provide additional related works about how LLMs have been combined with EAs for code and text generation, as well as benchmarking LLMs in chemical tasks in Appendix A.1..

3 The MOLLEO Framework

3.1 Problem Statement

Black-box optimization. Molecule discovery with a given property can be formulated as an optimization problem

$$m^* = \arg\max_{m \in M} F(m) \tag{1}$$

where m is a molecular structure and M denotes the set of valid molecules constituting the entire chemical space. The objective $F(m) : M \to \mathbb{R}$ is a black-box scalar-valued function that measures a certain molecule property m.

The measurement of chemical properties can involve complicated simulations or *in vivo* experiments, making it impossible to evaluate the gradients of the objective function F. Additionally, we assume

that the main computational expense of the optimization procedure comes from the objective evaluation (oracle call). Therefore, we design algorithms to minimize the number of oracle calls and compare all the algorithms with the same call budget.

Multi-objective black-box optimization. Oftentimes, molecules need to meet multiple, potentially competing objectives simultaneously. Multi-objective optimization aims to find the Pareto-optimal solution, where none of the objectives can be improved without deteriorating any of them [47]. The naive approach to optimize given objectives $\{F_i(\cdot)\}_{i=1}^n$ jointly is to consider an aggregate objective, such as the sum of all individual objectives, i.e.

$$m^* = \arg\max_{m \in M} \sum_i w_i F_i(m)\,, \tag{2}$$

where w_i is the weight of i-th objective, which can be considered a hyperparameter. However, determining the weight of each objective function might be nontrivial [45].

The rigorous approach to multi-objective optimization is the introduction of partial order and considering the solutions from the Pareto frontier [26, 18]. In this context, the partial order is defined by comparing all the objectives $\{F_i(\cdot)\}_{i=1}^n$ for the given molecules, i.e., m' surpasses m if every objective evaluated on m' is greater than the same objective evaluated on m (assuming the maximization of objectives). Formally,

$$m' \succeq m \iff \forall i \;\; F_i(m') \geq F_i(m)\,. \tag{3}$$

For the given set of molecules $S = \{m_j\}_{j=1}^m$, the Pareto frontier $P(S)$ is defined as the set of non-dominated solutions. Namely, for every molecule $m \in P(s)$ there is no other molecule in S surpassing m, i.e.

$$P(S) = \{m \in S : \{m' \in S : m' \succeq m\,,\; m' \neq m\} = \varnothing\}\,. \tag{4}$$

When jointly optimizing several objectives, we use the Pareto frontier to select candidates during the evolutionary search and compare algorithms. Namely, assuming that the objectives are bounded (e.g., $F(\cdot) \in [0, 1]$), one can compare two Pareto frontiers by evaluating their hypervolume

$$\text{Volume}(P(S)) = \text{Volume}\left(\cup_{m \in P(s)} H(m)\right)\,, \quad H(m) = \{x \in [0,1]^n : x_i \leq F_i(m)\,, \forall i\}\,, \tag{5}$$

where $H(m)$ is the hyperrectangle associated with the objectives evaluated on molecule m, and $\text{Volume}(\cdot)$ evaluates the Euclidean volume of the input set.

3.2 Evolutionary Algorithms

We build our MOLLEO framework upon the Graph-GA algorithm [39] — an evolutionary algorithm that operates as follows. An initial pool of molecules is randomly selected, and their fitnesses are calculated using a black-box oracle, $F(\cdot)$. Two parents are then sampled with a probability proportional to their fitnesses and combined using a `CROSSOVER` operator to generate an offspring, followed by a random `MUTATION` with probability p_m. This process is repeated `num_crossover` times, and the children are added to the pool of offspring. Finally, the fitnesses of the offspring are measured using $F(\cdot)$ and the offspring are added to the population. For single-objective optimization, the n_c fittest members from the population at a given step are selected to pass on to the next generation. For multi-objective optimization, two strategies are investigated: (1) Objective summation, where the summation of individual objectives is used as a single objective, and the n_c fittest members are retained; and (2) Pareto set selection, where only the Pareto frontier of the current population is kept. This process is repeated until the maximum allowed oracle calls (oracle budget) have been made. This process is outlined in Algorithm 1.

We incorporate chemistry-aware LLMs into the structure of Graph-GA by using them as proposal generators at `CROSSOVER` and `MUTATION` steps. That is, for the `CROSSOVER` step, instead of randomly combining two parent molecules, we generate molecules that maximize the objective fitness function guided by the objective description. For the `MUTATION` step, the operator mutates the fittest members of the current population based on the target description. However, we noticed that LLMs do not always generate candidates with higher fitness than the input molecule (demonstrated in Appendix C.1), and so we constructed a selection pressure to filter edited molecules based on structural similarity

Algorithm 1: MOLLEO Algorithm

Data: the initial pool $\mathbb{M}_0$; the objective F; the population size n_c; the number of offspring n_o.
Result: Optimized molecule population $\mathbb{M}^*$

```
begin
    for m ∈ M_0 do
        Compute F(m);
    for t ∈ [1, oracle_budget] do
        offspring = [];
        for num_crossovers do
            sample m_0, m_1 from M_t proportionally to objective value F(m);
            offspring.append(CROSSOVER(m_0, m_1));
        M_t ← sorted(M_t);
        for i ∈ [1, num_mutations] do
            offspring.append(MUTATION(M_t[i]));
        offspring ← search(offspring)[: n_o]        ▷ smallest Tanimoto distance to M_t[0]
        M_t ← offspring;
        for m ∈ M_t do
            Compute F(m);
        if Task_type == single_objective then
            M_t ← sorted(M_t)[: n_c];
        else
            M_t ← Pareto_Frontier(M_t);
    Return M_t;
```

to the top molecule [54]. That is, we sort the existing population by fitness, apply a mutation to the top population members, and then add them to the pool of offspring. Then, we prune the pool by selecting the n_o most similar offspring to the fittest molecule in the entire pool based on Tanimoto distance. We ablate the impact of this filter in Appendix C.2.

For each LLM, we describe below the details of how we implement the CROSSOVER and MUTATION operators. We empirically studied different combinations of models and hyperparameters (demonstrated in Appendix C.2), and in what follows, we describe the operators that resulted in the best performance.

Graph-GA The baseline algorithm that we build upon and compare against in our experiments.

▷ CROSSOVER: (default Graph-GA crossover): Two parent molecules are sampled with a probability proportional to their fitness. Crossover takes place at a ring position or non-ring position with equal likelihood. Parents are cut at random positions into fragments, and then fragments from both parents are combined. Invalid molecules are filtered out, and a randomly spliced molecule is returned [39].
▷ MUTATION: (default Graph-GA mutation): Random operations such as bond insertion or deletion, atom insertion or deletion, bond order swapping, or atom identity changes are done with predetermined likelihoods [39].

MOLLEO (GPT-4) GPT-4 is a proprietary LLM trained on a web-scale text corpus.

▷ CROSSOVER: Two parent molecules are sampled the same way as in Graph-GA. GPT-4 is then prompted to generate an offspring with the template t_{in} = "I have two molecules and their [target_objective] scores: ($s_{in,0}$, f_0), ($s_{in,1}$, f_1). Propose a new molecule with a higher [target_objective] by making crossover and mutations based on the given molecules." , where $s_{in,x}$ is an input SMILES and f_x is its fitness score. We then obtain an edited SMILES molecule as an output: s_{out} = GPT-4(t_{in}). If s_{out} cannot be decoded to a valid molecule structure, we generate an offspring using the default crossover operation from Graph-GA. We demonstrate the frequency of invalid LLM edits in Appendix C.1.
▷ MUTATION: We use the default Graph-GA mutation.

MOLLEO (BIOT5) BioT5 was developed with a two-phase training process using a baseline T5 model [60]. Initially, the model was trained on molecule-text data (339K samples), SELFIES structures, protein sequences, and general scientific text from multiple sources [58] using language masking as a training objective. Following this, the model was fine-tuned on specific downstream tasks, including text-based molecular generation, where molecules are generated given an input description [17].

▷ CROSSOVER: We use the default Graph-GA crossover.
▷ MUTATION: For the top Y molecules in the entire pool, we mutate them by prompting BioT5 with the template t_{in} = "Definition: You are given a molecule SELFIES. Your job is to generate a SELFIES molecule that [target_objective]. Now complete the following example - Input: <bom>[l_{in}]<eom> Output", where l_{in} is the SELFIES representation of a molecule. We then obtain an edited SELFIES molecule as an output: l_{out} = BioT5(t_{in}). We transform l_{out} back to the SMILES representation and add it to the pool of offspring. Since SELFIES can always be decoded into a molecular structure, there are no issues with BioT5 generating invalid molecules. With X offspring produced from crossover and Y offspring from the editing procedure, we select the top n_c offspring overall. This selection is based on structural similarity determined using Tanimoto distance to the fittest molecule in the entire pool [54].

MOLLEO (MOLSTM) MoleculeSTM was developed by jointly training molecule and text encoders on molecule-text pairs from PubChem using a contrastive loss, which maximizes the embedding similarity of each pair [49]. To enable molecular editing, they implemented a simple adaptor module to align their molecule encoder with the encoder of a pre-trained generative model. This alignment allowed them to utilize the generative model's decoder for structure generation.

▷ CROSSOVER: We use the default Graph-GA crossover.
▷ MUTATION: For the top Y molecules in the entire pool, we edited them by following a single text-conditioned editing step from [49]. Given the MoleculeSTM molecule and text encoders (E_{Mc} and E_{Tc}, respectively), a pre-trained generative model consisting of an encoder E_{Mg} and decoder D_{Mg} [37], and an adaptor module (A_{gc}) to align embeddings from E_{Mc} and E_{Mg}, an input molecule SMILES (s_{in}) is edited towards a text prompt describing the objective by updating the embedding from E_{Mg}. First, the molecule embedding x_0 is obtained from $E_{Mg}(s_{in})$. Then, x_0 is updated using gradient descent for T iterations:

$$x_{t+1} = x_t - \alpha \nabla_{x_t} \mathcal{L}(x_t)\,, \tag{6}$$

where α is the learning rate and $\mathcal{L}(x_t)$ is defined as:

$$\mathcal{L}(x_t) = -\texttt{cosine\_sim}\left(E_{Mc}(A_{gc}(x_t)), E_{Tc}(\texttt{text\_prompt})\right) + \lambda ||x_t - x_0||_2\,. \tag{7}$$

λ controls how much the embedding at iteration t can deviate from the input embedding. Finally, x_T is passed to the decoder D_{Mg} to generate a molecule SMILES s_{out}. We ablate MolSTM hyperparameter selection in Appendix C.4. If s_{out} cannot be decoded into a valid molecule (see Appendix C.1), we edit the next best molecule (so that we have Y offspring after the editing has finished). Similarly to MOLLEO (BIOT5), we combine the X crossover and Y mutated offspring and select the n_c most similar molecules to the top molecule overall to keep.

4 Experiments

4.1 Experimental Setup

We evaluate MOLLEO on 15 total tasks from two molecular generation benchmarks, Practical Molecular Optimization (PMO) [23] and Therapeutics Data Commons (TDC) [35]. Exact task definitions can be found in TDC [1]. We organize the tasks into the following categories:

1. *Structure-based optimization*, which optimizes for molecules based on target structures. It includes isomer generation based on a target molecular formula (isomers_c9h10n2o2pf2cl) and two tasks based on matching or avoiding scaffolds and substructure motifs (deco_hop, scaffold_hop).
2. *Name-based optimization*. These tasks involve finding compounds similar to known drugs (mestranol_similarity, thiothixene_rediscovery) and three multi-property optimization

[1] https://github.com/mims-harvard/TDC/blob/main/tdc/chem_utils/oracle/oracle.py

tasks (MPO) that aim to rediscover drugs (Perindopril, Ranolazine, Sitagliptin) while optimizing for other properties such as hydrophobicity (LogP) and permeability (TPSA). Although these tasks primarily involve rediscovering existing drugs rather than designing new molecules, they demonstrate that LLMs possess basic e. Successfully completing these tasks means that LLMs can make perturbations toward desired molecules when given a chemical optimization goal.

3. *Property optimization.* We first consider the trivial property optimization task `QED` [5], which measures the drug-likeness of a molecule based on a set of simple heuristics. We then focus on the three following tasks from PMO, which measure a molecule's activity against the following proteins: DRD2 (Dopamine receptor D2), GSK3β (Glycogen synthase kinase-3 beta), and JNK3 (c-Jun N-terminal kinase-3). For these tasks, molecular inhibition is determined by pre-trained classifiers that take in a SMILES string and output a value $p \in [0, 1]$, where $p \geq 0.5$ predicts that a molecule inhibits protein activity. Finally, we include three protein-ligand docking tasks from TDC [28] (also referred to as structure-based drug design [44]), which are more difficult tasks closer to real-world drug design compared to simple physicochemical properties [11]. The proteins we consider are DRD3 (dopamine receptor D3, PDB ID: 3PBL), EGFR (epidermal growth factor receptor, PDB ID: 2RGP), and Adenosine A2A receptor (PDB ID: 3EML). Molecules are docked against the protein using AutoDock Vina [16], with the output being the docking score of the binding process.

To evaluate our method, we follow [23] and report the area under the curve of top-k average property values versus the number of oracle calls (AUC top-k), which takes into account both the objective values and the computational budget spent. For this study, we set $k = 10$ in order to identify a small, distinct set of top molecular candidates. For the multi-objective optimization, we consider two metrics: top-10 AUC for summing all optimized objectives and the hypervolume of the Pareto frontier (see Equation (5)).

For baselines, we use the top-performing models from the PMO benchmark [23], including REINVENT [56], an RNN that utilizes a reinforcement learning-based policy to guide generation; Graph-GA; and Gaussian process Bayesian optimization (GP BO) [72], where a GP acquisition function is optimized with methods from Graph-GA.

For the choice of hyperparameters, we use the best practices from Graph-GA [39]; we describe these in Appendix B. MOLLEO (MOLSTM) involves additional hyperparameters when doing gradient descent; we investigate their selection in Appendix C.4. Additionally, we investigate design choices for MOLLEO (GPT-4) in Appendix C.5. We use three tasks for model development: `JNK3`, `perindopril_mpo`, and `isomers_c9h10n2o2pf2cl`; the rest are only evaluated during test-time. For each model, we show the prompts we used in Appendix E. We created prompts similar to those demonstrated in the original source code of each model, replacing each template with a task description. We briefly investigate the impact of prompt selection in Appendix C.6. For the initial population of molecules, we randomly sample 120 molecules from ZINC 250K [67]. In all runs, we restrict the budget of oracle calls to $10,000$ but terminate the algorithm early if the average fitness of the top-100 molecules does not increase by 10^{-3} within 5 epochs, as was done in [23]. For the docking experiments, we restrict the budget to 1000 calls due to higher evaluation costs. All experiments are conducted with five random seeds. Computational resources we utilized are described in Appendix A.2.

4.2 Empirical Study

First, we motivate the idea of why incorporating chemistry-aware LLMs in GA pipelines is effective. In Figure 2, we show the fitness distribution of an initial pool of random molecules inhibiting JNK3. We then perform a single round of edits to all molecules in the pool using each LLM and plot the resulting fitness distribution of the edited molecules. We find that the distribution for each LLM shifts to slightly higher fitness values, indicating that LLMs do provide useful modifications. However, the overall objective scores are still low, and so single-step editing is not sufficient. We then show the fitness distributions of the populations as the genetic optimization progresses and find that the fitness increases to higher values on average, given the same number of oracle calls. We show the performance of direct LLM querying versus the optimization procedure for additional tasks in Appendix C.1.

The results of single-objective optimization across 12 tasks in PMO are shown in Table 1, reporting the AUC top-10 for each task and the overall rank of each model. The results indicate that employing

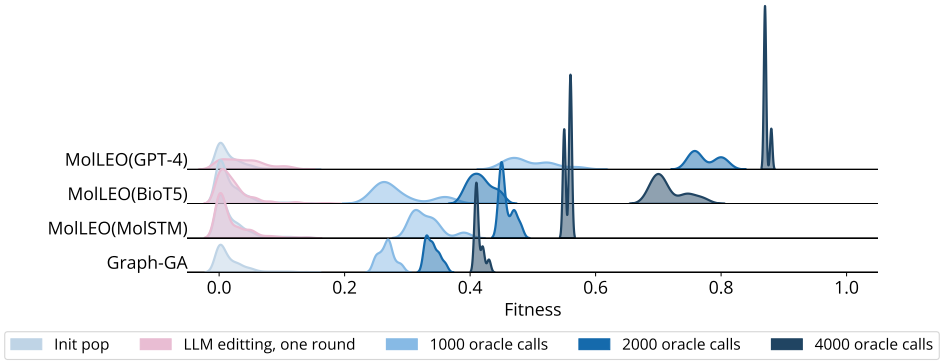


Figure 2: Population fitness over increasing number of iterations for JNK3 inhibition. In the lightest blue, we plot the fitness distribution of the initial molecule pool. We then pass the molecules through a single round of LLM edits (pink curve). Finally, we show the fitness distribution of the top-10 molecules after making 1000, 2000, and 4000 oracle calls.

Task type	Method objective (↑)	REINVENT	Graph GA	GP BO	MolLEO (MolSTM)	MolLEO (BioT5)	MolLEO (GPT-4)
Property optimization	QED	0.941 ± 0.000	0.940 ± 0.000	0.937 ± 0.000	0.937 ± 0.002	0.937 ± 0.002	**0.948 ± 0.004**
	JNK3	0.783 ± 0.023	0.553 ± 0.136	0.564 ± 0.155	0.643 ± 0.226	0.728 ± 0.079	**0.790 ± 0.027**
	DRD2	0.945 ± 0.007	0.964 ± 0.012	0.923 ± 0.017	0.975 ± 0.003	**0.981 ± 0.002**	0.968 ± 0.012
	GSK3β	0.865 ± 0.043	0.788 ± 0.070	0.851 ± 0.041	**0.898 ± 0.041**	0.889 ± 0.015	0.863 ± 0.047
Name-based optimization	mestranol_similarity	0.618 ± 0.048	0.579 ± 0.022	0.627 ± 0.089	0.596 ± 0.018	0.717 ± 0.104	**0.972 ± 0.009**
	thiothixene_rediscovery	0.534 ± 0.013	0.479 ± 0.025	0.559 ± 0.027	0.508 ± 0.035	0.696 ± 0.081	**0.727 ± 0.052**
	perindopril_mpo	0.537 ± 0.016	0.538 ± 0.009	0.493 ± 0.011	0.554 ± 0.037	**0.738 ± 0.016**	0.600 ± 0.031
	ranolazine_mpo	0.760 ± 0.009	0.728 ± 0.012	0.735 ± 0.013	0.725 ± 0.040	0.749 ± 0.012	**0.769 ± 0.022**
	sitagliptin_mpo	0.021 ± 0.003	0.433 ± 0.075	0.186 ± 0.055	0.548 ± 0.065	0.506 ± 0.100	**0.584 ± 0.067**
Structure-based optimization	isomers_c9h10n2o2pf2cl	0.642 ± 0.054	0.719 ± 0.047	0.469 ± 0.180	0.871 ± 0.039	0.873 ± 0.019	**0.874 ± 0.053**
	deco_hop	0.666 ± 0.044	0.619 ± 0.004	0.629 ± 0.018	0.613 ± 0.016	0.827 ± 0.093	**0.942 ± 0.013**
	scaffold_hop	0.560 ± 0.019	0.517 ± 0.007	0.548 ± 0.019	0.527 ± 0.019	0.559 ± 0.102	**0.971 ± 0.004**
	Total (↑)	7.872	7.857	7.521	8.395	9.202	10.008
	Rank (↓)	4	5	6	3	2	1

Table 1: Top-10 AUC of single-objective tasks. The best model for each task is bolded and the top three are underlined. We also report the sum of all tasks (total) and the rank of each model overall.

any of the three LLMs we tested as genetic operators improves performance over the default Graph-GA and all other baselines. Notably, MolLEO (GPT-4) outperforms all models in 9 out of 12 tasks, demonstrating its utility in molecular generation. MolLEO (BioT5) achieves the second-best results out of all the models tested, obtaining a total score close to that of MolLEO (GPT-4), and has the benefit of being free to use. We observe that MolLEO (BioT5) generally performs better than MolLEO (MolSTM), producing a higher percentage of molecules with improved fitness after editing, as shown in Appendix C.1. For the tasks `deco_hop` and `scaffold_hop`, there is only a small gain for the open-source MolLEO models. We speculate that this is because these models have not been trained on molecular descriptions containing SMARTS patterns. Also, it is unclear how well these models perform with negative matching (e.g., `This molecule` **`does not`** `contain the scaffold [#7]-c1n[c;h1]nc2 [c;h1]c(-[#8])[c;h0][c;h1]c12`). We also were interested to know whether the open-source models were generating molecules that could have been seen during training. We took ZINC20 [36], a database of 1.4 billion compounds that were used to generate the training set for BioT5, and PubChem [41](∼250K molecules), which was used to generate the training set for MoleculeSTM, and checked if the final molecules for the JNK3 task from each model appeared in the respective datasets. We found that this was not the case; there was no overlap between the generated molecules and the datasets.

We demonstrate empirically that MolLEO algorithms consistently converge faster than all the considered baselines, i.e., for any given budget of oracle calls, MolLEO achieves better objective values (see Appendix C.3). This is important when considering how these models can translate to real-world experiments to reduce the number of experiments needed to find ideal candidates.

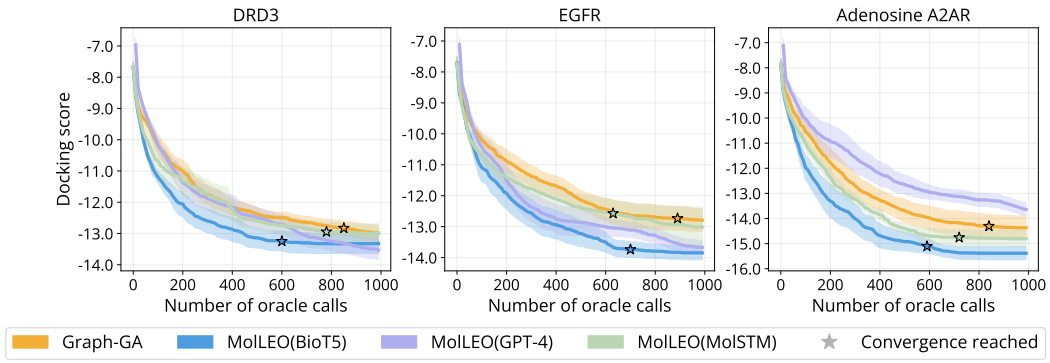


Figure 3: Average docking score of top-10 molecules when docked against DRD3, EGFR, or Adenosine A2A receptor proteins. Lower docking scores are better. For each model, we show the convergence point (the moment of stabilization of the population scores) with a star, if the model converges before 1000 oracle calls have been made. Here, the model is considered to have converged if the mean score of the top 100 molecules does not increase by at least 1e-3 within 5 epochs.

		Task 1: QED (↑), JNK3 (↑), SAscore (↓)		Task 2: QED (↑), GSK3β (↑), SAscore (↓)		Task 3: QED (↑), JNK3 (↑), SAscore (↓),GSK3β (↓), DRD2 (↓)	
Aggregate objective	Model	Sum	Hypervolume	Sum	Hypervolume	Sum	Hypervolume
Sum	Graph-GA	1.967 ± 0.088	0.713 ± 0.083	2.186 ± 0.069	0.719 ± 0.055	3.856 ± 0.075	0.162 ± 0.048
	MolLEO (MolSTM)	2.177 ± 0.178	0.625 ± 0.162	2.349 ± 0.132	0.303 ± 0.024	**4.040 ± 0.097**	0.474 ± 0.193
	MolLEO (BioT5)	1.946 ± 0.222	0.592 ± 0.199	2.306 ± 0.120	0.693 ± 0.093	3.904 ± 0.092	0.266 ± 0.201
	MolLEO (GPT-4)	**2.367 ± 0.044**	**0.752 ± 0.085**	**2.543 ± 0.014**	**0.832 ± 0.024**	4.017 ± 0.048	**0.606 ± 0.086**
PO	Graph-GA	2.120 ± 0.159	0.603 ± 0.082	2.339 ± 0.139	0.640 ± 0.034	4.051 ± 0.155	0.606 ± 0.052
	MolLEO (MolSTM)	2.234 ± 0.246	0.472 ± 0.248	2.340 ± 0.254	0.202 ± 0.054	3.989 ± 0.145	0.381 ± 0.204
	MolLEO (BioT5)	2.325 ± 0.164	0.630 ± 0.120	2.299 ± 0.203	0.645 ± 0.127	3.946 ± 0.115	0.367 ± 0.177
	MolLEO (GPT-4)	**2.482 ± 0.057**	**0.727 ± 0.038**	**2.631 ± 0.023**	**0.820 ± 0.024**	**4.212 ± 0.034**	**0.696 ± 0.029**

Table 2: Summation and hypervolume scores of multi-objective tasks. We report the results for two aggregation methods: Summation (Sum) and Pareto optimality (PO). The best model for each task is bolded.

In Figure 3, we present results for more challenging protein-ligand docking tasks, which better approximate real-world molecular generation scenarios compared to those in Table 1. We plot the average docking scores of the top-10 best molecules for MolLEO and Graph-GA against the number of oracle calls. We observe that nearly all LLMs in MolLEO generate molecules with lower (better) docking scores than the baseline model for all three proteins, and they converge faster to the optimal set. Among the three LLMs, MolLEO (BioT5) achieves the best performance. Surprisingly, MolLEO (GPT-4) performs worse than Graph-GA in the Adenosine A2A receptor docking task. In practice, better docking scores and faster convergence rates could result in requiring fewer bioassays to screen molecules, making the process both more cost- and time-effective. We visualize the top-10 molecules found by MolLEO in `EGFR docking` and `deco_hop` tasks in Appendix D.2.

In Table 2, we show the results of our multi-objective optimization for three tasks. Tasks 1 and 2 are inspired by goals in drug discovery and aim for simultaneous optimization of three objectives: maximizing a molecule's QED, minimizing its synthetic accessibility (SA) score (meaning that it is easier to synthesize), and maximizing its binding score to either JNK3 (Task 1) or GSK3β (Task 2). Task 3 is more challenging as it targets five objectives simultaneously: maximizing QED and JNK3 binding, as well as minimizing GSK3β binding, DRD2 binding, and SAScore. We find that MolLEO (GPT-4) consistently outperforms the baseline Graph-GA in all three tasks in terms of hypervolume and summation. In Figure 4, we visualize the Pareto optimal set (in objective space) for MolLEO and Graph-GA for Tasks 1 and 2. In Table 2, we see that the performance of open-source LLMs degrades when introducing multiple objectives into the prompt. We speculate that this performance drop may come from their inability to capture large, information-dense contexts. We also analyze the structural diversity and objective diversity of the Pareto optimal set in Appendix D.1.

Model	JNK3 Top-10 AUC
Initial fitness	0.373±0.079
Graph-GA	0.787±0.035
MOLLEO (MOLSTM)	0.815±0.048
MOLLEO (BIOT5)	0.799±0.036
MOLLEO (GPT-4)	**0.844±0.052**

Table 3: Initializing MOLLEO with the best molecules from ZINC 250K [67]. The results of three different LLMs in MOLLEO and Graph-GA are compared. For all molecules in ZINC 250K, we run the JNK3 oracle and select the top 120 molecule pool. We run MOLLEO initializing from this pool of molecules and optimizing JNK3. We report the top-10 AUC on the output of MOLLEO. See the description of the models in the text.

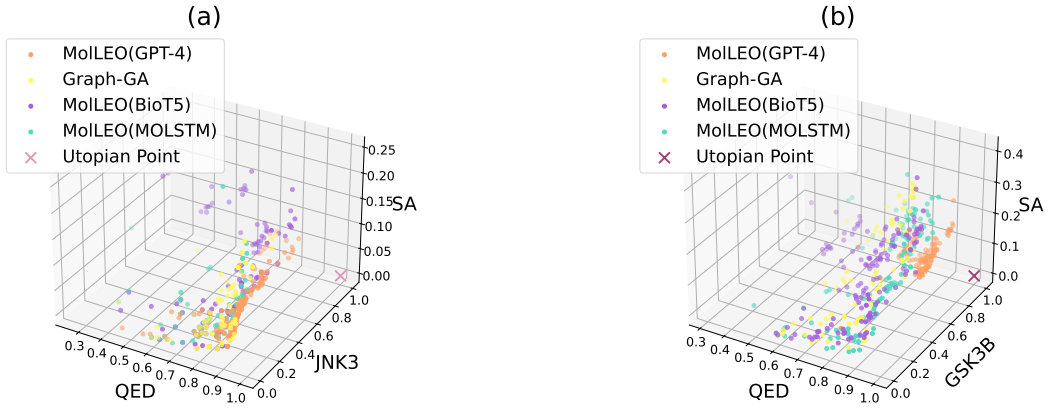


Figure 4: Pareto frontier visualizations for Graph-GA and MOLLEO on the following multi-objective tasks: (a) Task 1 (min SAscore, max JNK3 binding, max QED) and (b) Task 2 (min SAscore, max GSK3β binding, max QED). The utopian point corresponds to the maximum (best) possible values across all objectives. SAscores are rescaled to $[0, 1]$.

Given that the goal of EAs is to improve upon the properties of an initial pool of molecules and discover new molecules, we showcase these abilities by generating a set of molecules with higher objective values than the best known molecules from ZINC 250K [67]. That is, we initialize the molecular pool with the best molecules from ZINC 250K and run the optimization with MOLLEO and Graph-GA. We report the top-10 AUC on the JNK3 task in Table 3 and find that MOLLEO algorithms are consistently able to outperform the baseline model and improve upon the best values found in the existing dataset. We also briefly investigate the use of retrieval augmented search in Appendix C.5 and find that incorporating information from existing databases is helpful; we leave further investigations on this to future work.

5 Conclusion, Takeaway and Future Work

Herein, we propose MOLLEO: the first demonstration of incorporating LLMs into evolutionary algorithms for molecular discovery. We show that chemistry-aware LLMs can serve as informed proposal generators, resulting in superior optimization performance across multiple molecular optimization benchmarks, including protein-ligand docking. Furthermore, we show that both open-source and commercial versions of MOLLEO can be used in scenarios that involve numerous objective evaluations and can generate higher-ranked candidates with fewer evaluation calls compared to baseline models. Because the structural perturbations of MOLLEO are more effective than random perturbations in a genetic algorithm, it will become more feasible to deploy oracles that are computationally more expensive but more accurate in representing the target property, generating candidates that show greater promise for real-life applications. This is an important consideration due to the high experimental costs of testing candidates.

Molecular discovery and design is a rich field with numerous practical applications, many of which extend beyond the current study's scope but remain relevant to the proposed framework. Integrating LLMs into evolutionary algorithms offers versatility through plain text specifications, suggesting

that the MOLLEO framework can be applied to scenarios such as drug discovery, expensive *in silico* simulations, and the design of materials or large biomolecules. Future work will aim to further improve the quality of proposed candidates, both in terms of their objective values and the speed with which they are found. As LLMs continue to advance, we anticipate that the performance of the MOLLEO framework will also continue to improve, making MOLLEO a promising tool for applications in generative chemistry.

6 Acknowledgements

M.S. thanks Ella Rajaonson for feedback on protein-ligand docking and Austin Cheng for discussions on molecule generation. Y.D. thanks Delia Qu for helpful discussions on evolutionary algorithms. A. A.-G. thanks Dr. Anders G. Frøseth for his generous support. A. A.-G. also acknowledges the generous support of Natural Resources Canada and the Canada 150 Research Chairs program.

This work was supported in part by NSF IIS-2008334, IIS-2106961, CAREER IIS-2144338, ONR MURI N00014-17-1-2656, and computing resources from Microsoft Azure. Resources used in preparing this research were also provided, in part, by the Vector Institute and the Acceleration Consortium.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Microsoft Research AI4Science and Microsoft Azure Quantum. The impact of large language models on scientific discovery: a preliminary study using gpt-4. *arXiv preprint arXiv:2311.07361*, 2023.

[3] Kenneth Atz, Francesca Grisoni, and Gisbert Schneider. Geometric deep learning on molecular representations. *Nature Machine Intelligence*, 3(12):1023–1032, 2021.

[4] Nikhil Behari, Edwin Zhang, Yunfan Zhao, Aparna Taneja, Dheeraj Nagaraj, and Milind Tambe. A decision-language model (dlm) for dynamic restless multi-armed bandit tasks in public health. *arXiv preprint arXiv:2402.14807*, 2024.

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

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

[7] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

[8] Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

[9] Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. Chemberta: Large-scale self-supervised pretraining for molecular property prediction.

[10] Dimitrios Christofidellis, Giorgio Giannone, Jannis Born, Ole Winther, Teodoro Laino, and Matteo Manica. Unifying molecular and textual representations via multi-task language modelling. In *International Conference on Machine Learning*, pages 6140–6157. PMLR, 2023.

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

[12] Kourosh Darvish, Marta Skreta, Yuchi Zhao, Naruki Yoshikawa, Sagnik Som, Miroslav Bogdanovic, Yang Cao, Han Hao, Haoping Xu, Alán Aspuru-Guzik, et al. Organa: A robotic assistant for automated chemistry experimentation and characterization. *arXiv preprint arXiv:2401.06949*, 2024.

[13] Inc. Daylight Chemical Information Systems. Smarts-a language for describing molecular patterns, 2007.

[14] Yuanqi Du, Arian R. Jamasb, Jeff Guo, Tianfan Fu, Charles Harris, Yingheng Wang, Chenru Duan, Pietro Liò, Philippe Schwaller, and Tom L. Blundell. Machine learning-aided generative molecular design. *Nature Machine Intelligence*, June 2024.

[15] Yuanqi Du, Xian Liu, Nilay Mahesh Shah, Shengchao Liu, Jieyu Zhang, and Bolei Zhou. Chemspace: Interpretable and interactive chemical space exploration. *Transactions on Machine Learning Research*, 2022.

[16] Jerome Eberhardt, Diogo Santos-Martins, Andreas F Tillack, and Stefano Forli. Autodock vina 1.2. 0: New docking methods, expanded force field, and python bindings. *Journal of chemical information and modeling*, 61(8):3891–3898, 2021.

[17] Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. Translation between molecules and natural language. *arXiv preprint arXiv:2204.11817*, 2022.

[18] Sean Ekins, J Dana Honeycutt, and James T Metz. Evolving molecules using multi-objective optimization: applying to adme/tox. *Drug discovery today*, 15(11-12):451–460, 2010.

[19] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

[20] Daniel Flam-Shepherd and Alán Aspuru-Guzik. Language models can generate molecules, materials, and protein binding sites directly in three dimensions as xyz, cif, and pdb files. *arXiv preprint arXiv:2305.05708*, 2023.

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

[22] Tianfan Fu, Wenhao Gao, Cao Xiao, Jacob Yasonik, Connor W Coley, and Jimeng Sun. Differentiable scaffolding tree for molecular optimization. *arXiv preprint arXiv:2109.10469*, 2021.

[23] 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.

[24] Wenhao Gao, Rocío Mercado, and Connor W Coley. Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design. *arXiv preprint arXiv:2110.06389*, 2021.

[25] Tobias Gensch, Gabriel dos Passos Gomes, Pascal Friederich, Ellyn Peters, Théophile Gaudin, Robert Pollice, Kjell Jorner, AkshatKumar Nigam, Michael Lindner-D'Addario, Matthew S Sigman, et al. A comprehensive discovery platform for organophosphorus ligands for catalysis. *Journal of the American Chemical Society*, 144(3):1205–1217, 2022.

[26] AM Geoffrion. Proper efficiencyand the theory of vector optimization. *J. Math. Anal. Appl*, 22, 1968.

[27] 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.

[28] DE Graff, EI Shakhnovich, and CW Coley. Accelerating high-throughput virtual screening through molecular pool-based active learning, chem. *Sci*, 12:7866–7881, 2021.

[29] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chemical science*, 11(2):577–586, 2020.

[30] Jeff Guo and Philippe Schwaller. Augmented memory: Capitalizing on experience replay to accelerate de novo molecular design. *arXiv preprint arXiv:2305.16160*, 2023.

[31] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.

[32] Taicheng Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh Chawla, Olaf Wiest, Xiangliang Zhang, et al. What can large language models do in chemistry? a comprehensive benchmark on eight tasks. *Advances in Neural Information Processing Systems*, 36:59662–59688, 2023.

[33] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[34] Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pages 8867–8887. PMLR, 2022.

[35] 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*, 2021.

[36] John J Irwin, Khanh G Tang, Jennifer Young, Chinzorig Dandarchuluun, Benjamin R Wong, Munkhzul Khurelbaatar, Yurii S Moroz, John Mayfield, and Roger A Sayle. Zinc20—a free ultralarge-scale chemical database for ligand discovery. *Journal of chemical information and modeling*, 60(12):6065–6073, 2020.

[37] Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3(1):015022, 2022.

[38] Kevin Maik Jablonka, Philippe Schwaller, Andres Ortega-Guerrero, and Berend Smit. Leveraging large language models for predictive chemistry. *Nature Machine Intelligence*, pages 1–9, 2024.

[39] 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.

[40] 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.

[41] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. Pubchem 2023 update. *Nucleic acids research*, 51(D1):D1373–D1380, 2023.

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

[43] Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and Geoff Pleiss. A sober look at llms for material discovery: Are they actually good for bayesian optimization over molecules? *arXiv preprint arXiv:2402.05015*, 2024.

[44] Irwin D Kuntz. Structure-based strategies for drug design and discovery. *Science*, 257(5073):1078–1082, 1992.

[45] Nathanael Kusanda, Gary Tom, Riley Hickman, AkshatKumar Nigam, Kjell Jorner, and Alan Aspuru-Guzik. Assessing multi-objective optimization of molecules with genetic algorithms against relevant baselines. In *AI for Accelerated Materials Design NeurIPS 2022 Workshop*, 2022.

[46] Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley. Evolution through large models. In *Handbook of Evolutionary Machine Learning*, pages 331–366. Springer, 2023.

[47] Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations*, 2022.

[48] Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. Large language models as evolutionary optimizers. *arXiv preprint arXiv:2310.19046*, 2023.

[49] Shengchao Liu, Weili Nie, Chengpeng Wang, Jiarui Lu, Zhuoran Qiao, Ling Liu, Jian Tang, Chaowei Xiao, and Animashree Anandkumar. Multi-modal molecule structure–text model for text-based retrieval and editing. *Nature Machine Intelligence*, 5(12):1447–1457, 2023.

[50] Shengchao Liu, Jiongxiao Wang, Yijin Yang, Chengpeng Wang, Ling Liu, Hongyu Guo, and Chaowei Xiao. Conversational drug editing using retrieval and domain feedback. In *The Twelfth International Conference on Learning Representations*, 2024.

[51] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[52] Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

[53] Adrian Mirza, Nawaf Alampara, Sreekanth Kunchapu, Benedict Emoekabu, Aswanth Krishnan, Mara Wilhelmi, Macjonathan Okereke, Juliane Eberhardt, Amir Mohammad Elahi, Maximilian Greiner, et al. Are large language models superhuman chemists? *arXiv preprint arXiv:2404.01475*, 2024.

[54] 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.

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

[56] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de novo design through deep reinforcement learning. *CoRR*, abs/1704.07555, 2017.

[57] Hakime Öztürk, Arzucan Özgür, Philippe Schwaller, Teodoro Laino, and Elif Ozkirimli. Exploring chemical space using natural language processing methodologies for drug discovery. *Drug Discovery Today*, 25(4):689–705, 2020.

[58] Qizhi Pei, Wei Zhang, Jinhua Zhu, Kehan Wu, Kaiyuan Gao, Lijun Wu, Yingce Xia, and Rui Yan. BioT5: Enriching cross-modal integration in biology with chemical knowledge and natural language associations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1102–1123, 2023.

[59] Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. Molecularrnn: Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv:1905.13372*, 2019.

[60] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[61] Mayk Caldas Ramos, Shane S Michtavy, Marc D Porosoff, and Andrew D White. Bayesian optimization of catalysts with in-context learning. *arXiv preprint arXiv:2304.05341*, 2023.

[62] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[63] Arne Schneuing, Yuanqi Du, Charles Harris, Arian Jamasb, Ilia Igashov, Weitao Du, Tom Blundell, Pietro Lió, Carla Gomes, Max Welling, et al. Structure-based drug design with equivariant diffusion models. *arXiv preprint arXiv:2210.13695*, 2022.

[64] Philippe Schwaller, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee. Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction. *ACS central science*, 5(9):1572–1583, 2019.

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

[66] Michael A Skinnider. Invalid smiles are beneficial rather than detrimental to chemical language models. *Nature Machine Intelligence*, pages 1–12, 2024.

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

[68] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

[69] Dagmar Stumpfe and Jürgen Bajorath. Exploring activity cliffs in medicinal chemistry: miniperspective. *Journal of medicinal chemistry*, 55(7):2932–2942, 2012.

[70] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

[71] Gary Tom, Stefan P. Schmid, Sterling G. Baird, Yang Cao, Kourosh Darvish, Han Hao, Stanley Lo, Sergio Pablo-García, Ella M. Rajaonson, Marta Skreta, and et al. Self-driving laboratories for chemistry and materials science. *ChemRxiv*, 2024.

[72] 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.

[73] Vahe Tshitoyan, John Dagdelen, Leigh Weston, Alexander Dunn, Ziqin Rong, Olga Kononova, Kristin A Persson, Gerbrand Ceder, and Anubhav Jain. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, 571(7763):95–98, 2019.

[74] Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*, 2023.

[75] Guanghao Wei, Yining Huang, Chenru Duan, Yue Song, and Yuanqi Du. Navigating chemical space with latent flows. *arXiv preprint arXiv:2405.03987*, 2024.

[76] 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.

[77] Andrew D White. The future of chemistry is language. *Nature Reviews Chemistry*, 7(7):457–458, 2023.

[78] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.

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

[80] Geyan Ye, Xibao Cai, Houtim Lai, Xing Wang, Junhong Huang, Longyue Wang, Wei Liu, and Xiangxiang Zeng. Drugassist: A large language model for molecule optimization. *arXiv preprint arXiv:2401.10334*, 2023.

[81] Naruki Yoshikawa, Marta Skreta, Kourosh Darvish, Sebastian Arellano-Rubach, Zhi Ji, Lasse Bjørn Kristensen, Andrew Zou Li, Yuchi Zhao, Haoping Xu, Artur Kuramshin, et al. Large language models for chemistry robotics. *Autonomous Robots*, 47(8):1057–1086, 2023.

[82] 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.

[83] Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 617–626, 2020.

Appendix

A Extended descriptions

A.1 Extended related work

Benchmarking LLMs on Chemistry Tasks ChemLLMBench benchmarked several widely-used LLMs on a set of eight chemistry tasks, such as property prediction, reaction prediction, and molecule captioning [32]. The results showed that while LLMs can perform well in selection tasks, they struggle with tasks requiring more in-depth chemical reasoning, such as property-conditioned generation. This motivates the need for improving how LLMs are used in generative tasks. Similarly, SciBench evaluated LLMs on free-response college-level exam questions across various science disciplines, including chemistry, which required complex, multi-step solutions [74]. Their results indicated that LLMs were unable to generate correct solutions for the majority of questions [74]. However, progress of LLMs has been noted in general question-answering capabilities: a recent work introduced ChemBench, a dataset of over 7,000 question-answer pairs aimed at providing a systematic understanding of LLM capabilities across different subdomains in chemistry [53]. It was concluded that state-of-the-art LLMs such as GPT-4 and Claude 3 were able to beat human chemists on these questions on average, although they still struggle with physical and commonsense chemical reasoning.

LLMs and Evolutionary Algorithms Previous research has demonstrated that language models can be incorporated as operators in evolutionary algorithms in applications such as code and prompt generation [46]. For example, OPRO and LMEA use LLMs to optimize solutions for different mathematical optimization problems [78, 48]. Other works have shown that LLMs can be used as crossover and mutation operators to directly optimize prompts using a training set, outperforming human-engineered prompts [19, 31]. Other applications of LLMs in evolutionary frameworks have been code synthesis (FunSearch [62]), generation of reward functions in RL for robot control (Eureka [51], and resource allocation in public health settings [4].

A.2 Computational Resources

Our experiments were computed on NVIDIA A100-SXM4-80GB and T4V2 GPUs. Some of our experiments utilized the GPT-4 model; this refers to the `gpt-4-turbo` checkpoint from 2023-07-01 [2]. All GPT-4 checkpoints were hosted on Microsoft Azure[3].

A.3 Limitations

All benchmarks and tasks evaluated in this study are proxies for real chemical properties and may not correctly capture the true chemical performance of molecules in the real world. Thus, the effectiveness of our model in real-world applications remains to be thoroughly validated.

A.4 Broader Impact

The methods proposed in this paper aim to find compounds with desired properties more efficiently, which can benefit many areas, including drug discovery and materials design. While we do not foresee negative societal impacts from our methods, we acknowledge the potential of their dual use for nefarious purposes. We encourage discussions around these issues and strongly support the development and deployment of safeguards to prevent them.

B Hyperparameters

In this section, we report the hyperparameters that were used in Graph-GA, the baseline genetic algorithm that we build our method upon. We kept the best hyperparameters that were determined

[2].https://platform.openai.com/docs/models

[3] *.openai.azure.com

Metric	MoleculeSTM	BioT5	GPT-4
Percent valid molecules	peridopril_mpo: 0.938 JNK3: 0.928	peridopril_mpo: 1.000 JNK3: 1.000	peridopril_mpo: 0.862 JNK3: 0.835
Percent molecules with higher fitness after editting	peridopril_mpo: 0.456 JNK3: 0.206	peridopril_mpo: 0.568 JNK3: 0.513	peridopril_mpo: 0.240 JNK3: 0.263
Mean fitness increase	peridopril_mpo: +0.033 JNK3: +0.022	peridopril_mpo: +0.208 JNK3: +0.0320	peridopril_mpo: +0.032 JNK3: +0.0262

Table 4: Viability of LLM edits. We prompt different LLMs with descriptions of the JNK3 and perindopril_mpo target objectives on an initial random pool of molecules drawn from 5 random seeds. We report the percentage of valid molecules (number of valid molecules / number of total molecules), the percentage of molecules with higher fitness after editting, and the mean fitness increase of those molecules.

in [23]. In each iteration, Graph-GA samples two molecules with a probability proportional to their fitnesses for crossover and mutation and then randomly mutates the offspring with probability $p_m = 0.067$. This process is repeated to generate 70 offspring. The fitnesses of the offspring are measured, and the top 120 most fit molecules in the entire pool are kept for the next generation. For docking experiments, we reduce the number of generated offspring to 7 and the population size to 12 due to long experiment runtimes. We set the maximum number of oracle calls to 10,000 for all experiments except docking, where we set it to 1,000. We kept the default early-stopping criterion the same as in PMO [23], which is that we terminate the algorithm if mean score of the top 100 molecules does not increase by at least 1e-3 within five epochs.

C Ablation studies

C.1 Performance of single-step molecule editing

To motivate the incorporation of LLMs into a GA framework, we directly query the LLMs we consider to edit a molecule towards a certain property and calculate: (1) the percentage of valid molecules that are output (given that not all SMILES are valid molecules) and (2) which of the output molecules have higher fitness. We show these results on the JNK3 inhibition task in Table 4 and find that MolSTM and GPT-4 are not always able to produce valid molecules, whereas BioT5 always is due to its use of SELFIES. We also find that BioT5 produced more molecules with higher fitness values compared to the other LLMs.

In Table 6, we show the performance of directly querying LLMs with an initial pool of molecules on additional tasks. We find that while LLMs are able to edit the molecule pool to improve the fitness marginally, using them in an optimization framework results in much better fitness values.

C.2 Incorporating LLM-based genetic operators into Graph-GA

There are many ways to incorporate LLMs as genetic operators in a GA framework. We investigate several options. First, we investigate using LLMs as a crossover operator. For GPT-4 and BioT5, we gave each model two parent molecules as input and a description of the objective, and asked the model to produce a molecule as an output. Because MolSTM aligns molecule embeddings with text embeddings, our crossover operation was to either take a linear or spherical interpolation of the parent molecule embeddings and maximize the similarity of the resulting embedding to the text objective. For the mutation operator, we prompted each LLM with a molecule and a description of the objective. Finally, we investigated the impact of applying a selection pressure in the form of a filter, where we only mutated the top Y molecules and pruned the resulting offspring by distance to the best molecule overall. We show the results for all operator settings we tried in Table 5 and show which operators we ended up using for each LLM in the final framework.

Operators	Graph-GA (Baseline)	MOLLEO (MOLSTM)	MOLLEO (BIOT5)	MOLLEO (GPT-4)
(Default Graph-GA settings) CROSSOVER: Random MUTATION: Random, $p_m = 0.067$	peridopril_mpo: 0.538±0.009 JNK3: 0.553±0.136 ✓	N/A	N/A	N/A
CROSSOVER: LLM MUTATION: Random, $p_m = 0.067$	N/A	peridopril_mpo: 0.499±0.012[linear] 0.505±0.018[spherical] JNK3: 0.722±0.046 [linear] 0.744±0.055 [spherical]	peridopril_mpo: 0.727±0.013 JNK3: 0.436±0.052	peridopril_mpo: 0.600±0.031 JNK3: 0.790±0.027 ✓
CROSSOVER: Random MUTATION: LLM, $p_m = 0.067$	N/A	peridopril_mpo: 0.532±0.034 JNK3: 0.631±0.327	peridopril_mpo: 0.676±0.034 JNK3: 0.650±0.096	peridopril_mpo: 0.552±0.024 JNK3: 0.673±0.047
CROSSOVER: Random MUTATION: LLM, $p_m = 1$	N/A	peridopril_mpo: 0.513±0.040 JNK3: 0.553±0.193	peridopril_mpo: 0.686±0.343 JNK3: 0.708±0.030	peridopril_mpo: 0.615±0.058 JNK3: 0.762±0.044
CROSSOVER: Random MUTATION: Selected top Y molecules, randomly mutated, pruned offspring by distance to top-1 molecule	peridopril_mpo: 0.579±0.044 JNK3: 0.571±0.109	N/A	N/A	N/A
CROSSOVER: Random MUTATION: Selected top Y molecules, mutated with LLM, pruned offspring by distance to top-1 molecule	N/A	peridopril_mpo: 0.554±0.034 JNK3: 0.730±0.188 ✓	peridopril_mpo: 0.740±0.032 JNK3: 0.728±0.079 ✓	peridopril_mpo: 0.575±0.074 JNK3: 0.758±0.031

Table 5: **Top-10 AUC on 5 random seeds for the JNK3 and perindopril_mpo tasks using different combinations of genetic operators.** The operators used for each model to compute the final results in the main paper are indicated with a ✓ symbol.

C.3 Optimization trends over single-objective tasks.

In Figure 5, we show the optimization curves for three tasks: JNK3, perindopril_mpo, and isomers_c9h10n2o2pf2cl.

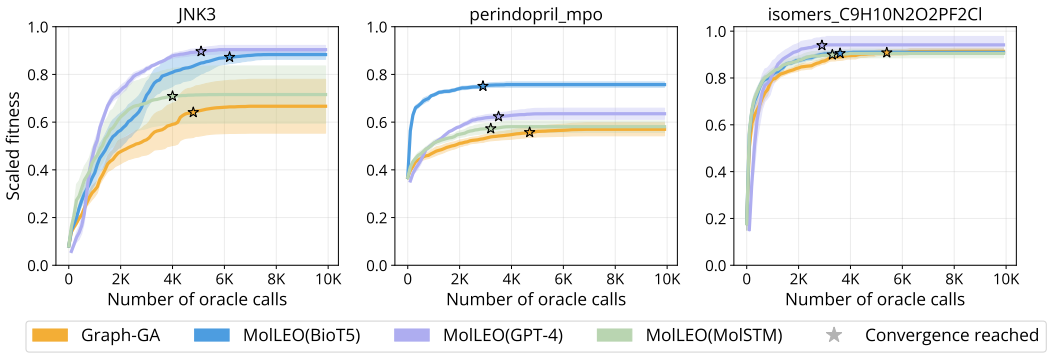


Figure 5: Average of top-10 molecules generated by MOLLEO and Graph-GA models for three tasks over an increasing number of oracle calls. For each model, we show the convergence point with a star. The model is considered to have converged if the mean score of the top 100 molecules does not increase by at least 1e-3 within five epochs.

	JNK3	isomers_c9h10n2o2pf2cl	perindopril_mpo
Initial population	0.085 ± 0.010	0.101 ± 0.025	0.281 ± 0.026
MolSTM - direct query	0.084 ± 0.008	0.201 ± 0.040	0.390 ± 0.008
MOLLEO (MOLSTM)	**0.716 ± 0.240**	**0.905 ± 0.0372**	**0.572 ± 0.041**
BioT5 - direct query	0.109 ± 0.012	0.260 ± 0.076	0.648 ± 0.019
MOLLEO (BIOT5)	**0.883 ± 0.040**	**0.909 ± 0.015**	**0.759 ± 0.019**
GPT-4 - direct query	0.164 ± 0.076	0.686 ± 0.127	0.388 ± 0.075
MOLLEO (GPT-4)	**0.926 ± 0.052**	**0.935 ± 0.048**	**0.643 ± 0.094**

Table 6: Ablation studies of LLM editing based on direct user queries. Top-10 average objective scores are reported.

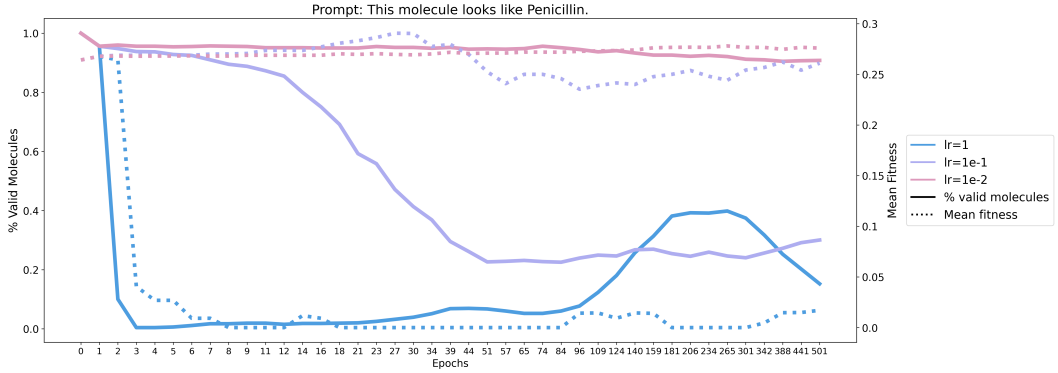
C.4 MoleculeSTM hyperparameter selection

MolSTM has several hyperparameters; in this section, we motivate our choices for the final model. The first is the number of population members that are selected to undergo LLM-based mutations (Algorithm 1). In Table 7, we show the Top-10 AUC after choosing different numbers of top-scoring candidates for editing by MoleculeSTM. We find that 30 candidates resulted in the best performance. Note that we used a different prompt for this experiment than the one used to obtain results in Table 1 (see Appendix C.6). We use 30 candidates anytime the filter is employed for all models, although this hyperparameter can be ablated independently for each model.

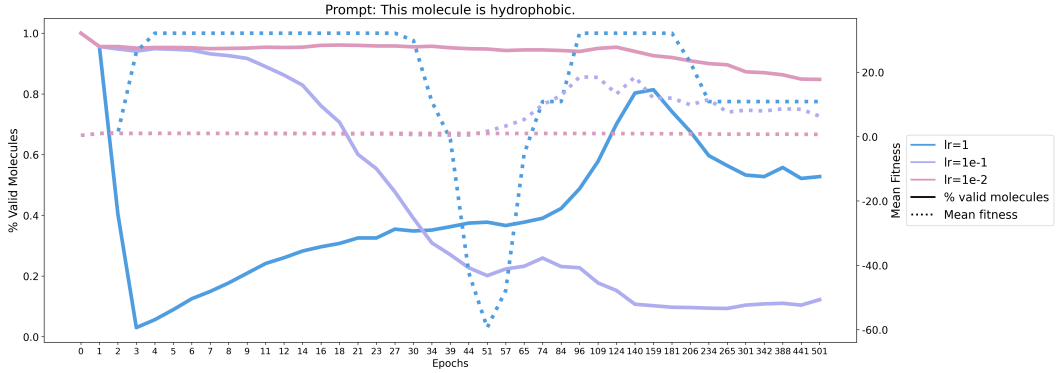
MoleculeSTM has several hyperparameters related to molecule generation since it involves gradient descent to optimize an input molecule embedding based on a text prompt. We look at two hyperparameters, the number of gradient descent steps (epochs) and learning rate, and plot the results in

Number of top-scoring candidates selected for mutation	Top-10 AUC
20	0.680±0.213
30	**0.730±0.188**
50	0.627±0.250

Table 7: Top-10 AUC on JNK3 binding task with varying number of top-scoring candidates selected to undergo LLM-based mutations.



(a)



(b)

Figure 6: Mean fitness and percent valid molecules with varying number of gradient descent epochs (plotted on log-scale) and learning rates in MoleculeSTM on two tasks: (a) molecular similarity to Penicillin (based on Tanimoto distance) and (b) molecule hydrophobicity (logP).

Figure 6. We find that if the learning rate is too large (lr=1), the mean fitness changes unpredictably, but if it is too small (lr=1e-2), there are minimal changes to the mean fitness. Setting the learning rate to 1e-1 results in more consistent improvements in mean fitness. We also set the number of epochs to 30 since more epochs are too time-consuming and fewer do not result in noticeable fitness changes.

C.5 GPT-4 ablations

We conduct experiments to understand the performance of MOLLEO (GPT-4) in the following settings: different numbers of offspring in each generation, different underlying GPT models, incorporating retrieval augmentation methods, and different rules from Graph-GA and SMILES-GA in Table 8 and Table 9, and describe the results in following sections.

	Number of offspring			RAG Search	
	20	70	200	w. RAG	w/o. RAG
jnk3	0.731±0.012	0.790±0.027	0.785±0.022	0.830±0.047	0.790±0.027
isomer_c9h10n2o2pf2cl	0.967±0.010	0.874±0.053	0.960±0.049	0.982±0.018	0.874±0.053
perindopril mpo	0.573±0.042	0.600±0.031	0.580±0.028	0.717±0.024	0.600±0.031

Table 8: Ablation study on MOLLEO (GPT-4). Impact of the number of offspring in each round and retrieval-augmented search (RAG).

	Different Versions of LLMs		Rules		
	GPT-3.5	GPT-4	No rules	Graph-GA rules	SMILES-GA rules
jnk3	0.669±0.104	0.790±0.027	0.765±0.047	0.790±0.027	0.774±0.084
isomer_c9h10n2o2pf2cl	0.902±0.021	0.874±0.053	0.871±0.085	0.874±0.053	0.872±0.029
perindopril mpo	0.564±0.022	0.600±0.031	0.562±0.042	0.600±0.031	0.583±0.031

Table 9: Ablation study on MOLLEO (GPT-4). Impact of different versions of LLMs and rules from different sources.

Number of offspring We vary the number of offspring generated in each iteration of MOLLEO (GPT-4) on three tasks and find that 70 offspring produces, on average, the best results, which is also the same number determined in [23]

Retrieval-augmented search To explore how retrieval can enhance LLMs in the optimization process, we incorporate a retrieval-augmented search module into MOLLEO (GPT-4). Specifically, after offspring are proposed, 1,000 molecules are randomly sampled from ZINC 250K. From these, 20 molecules are selected based on their Tanimoto similarity to the top 20 molecules in the current population. These retrieved molecules then replace the 20 worst molecules in the population. In Table 8, the results show that this approach is effective in improving the optimization results of MOLLEO (GPT-4) for each task.

GPT-3.5 vs. GPT-4 We tested MOLLEO using both GPT-4 and GPT-3.5, an older version of the model. In Table 9, we show that GPT-4 outperforms GPT-3.5 on two tasks, although GPT-3.5 still beats the baseline Graph-GA algorithm (Table 1). Interestingly, GPT-3.5 beats GPT-4 on a task based on structure-based optimization.

Different rules In Graph-GA, the default crossover and mutation operators are pre-defined by domain experts based on chemical knowledge. These pre-defined operators can be considered as rules guiding the generation process. Here we also consider rules from another source, SMILES-GA [82], which defines rules that operate on SMILES strings instead of graphs. To evaluate the impact of rules from different sources, we perform an ablation study on MOLLEO and also conduct experiments without any rules, where LLMs are repeatedly queried to propose molecules until the offspring size reaches the target number in each round. The results shown in Table 9 indicate that both Graph-GA and SMILES-GA rules are better than not using results at all, and Graph-based rules are better than SMILES-based rules.

C.5.1 GPT-4 in an active learning framework

We investigate the performance of GPT-4 when the EA framework is replaced with an active learning setting. This can be thought of as testing the impact of the genetic operators in the underlying genetic framework. In this setting, we initialize a population pool and randomly sample k molecules from the pool. We then pass the molecules to GPT-4 and query it for a new molecule with better objective values. After generating a batch of molecules, we integrate the batch back into the population without selection, allowing the population to grow until it reaches the budget of oracle calls. In our experiment, we set the budget to 10,000 oracle calls, the batch size to 100, and k to 2.

The results, shown in Table 10, indicate that the active learning setting achieves subpar performance compared to MOLLEO (GPT-4). This demonstrates that while LLMs like GPT-4 can modify existing molecules, they struggle to independently propose high-quality molecules, underscoring the necessity of the evolutionary process. Interestingly, we observe that the active learning setting performs relatively well on the `isomer` task compared to the other two; this can maybe be attributed to the `isomer` task being simple.

C.6 Impact of prompt selection

The choice of prompt for a given task is an important consideration, as some prompts can be better aligned with information the model knows. For example, the prompt we used in MOLLEO

	GPT4-AL	MOLLEO (GPT-4)
JNK3	0.583±0.042	0.790±0.027
isomer_c9h10n2o2pf2cl	0.873±0.048	0.874±0.053
perindopril mpo	0.539±0.046	0.600±0.031

Table 10: Ablation studies of active learning (AL) on GPT-4. We report the Top-10 AUC of single objective results.

(MOLSTM) for the JNK3 inhibition task was "`This molecule inhibits JNK3.`" However, there are multiple ways of describing inhibition and multiple ways of identifying the enzyme (JNK3, c-Jun N-terminal kinase 3). To that end, we investigate the impact of prompt selection on downstream performance.

To generate a set of prompts, we prompted GPT-4 to generate ten synonymous phrases for an input prompt. We then computed the Spearman rank-order correlation coefficient (Spearman's ρ) of each phrase on an initial molecule pool between the cosine similarity generated by MoleculeSTM and the ground truth fitness values. Finally, we ran the genetic optimization using MOLLEO (MOLSTM) with the input prompt and the prompt with the highest Spearman rank-order correlation coefficient.

On the JNK3 task, the default prompt we wrote was "`This molecule inhibits JNK3.`", which had a Spearman's ρ of -0.0161. The prompt with the largest Spearman's ρ (0.1202) was "`This molecule acts as an antagonist to JNK3.`" When we ran MOLLEO (MOLSTM) with the default input prompt, the top-10 AUC was 0.643 ± 0.226. When we ran MOLLEO (MOLSTM) using the prompt with the largest Spearman's ρ, the top-10 AUC was 0.730 ± 0.188. This demonstrates that prompt selection can influence downstream results, especially for smaller models, and opens the door for future work in this area.

D Extended experiment results

D.1 Diversity analysis in multi-objective optimization

We show the structural diversity and objective diversity for multi-objective optimization in Table 11. Structural diversity reflects the chemical diversity of the Pareto set and is computed by taking the average pairwise Tanimoto distance between Morgan fingerprints of molecules in the set. Objective diversity illustrates the objective value coverage of the Pareto frontier and is computed by taking the pairwise Euclidean distance between objective values of molecules in the Pareto set.

D.2 Case study: Sample molecules from final pool

Below, we show the top ten molecules across all runs from the MOLLEO and Graph-GA for two tasks: `deco_hop` and `EGFR docking`.

D.2.1 Task 1: `deco_hop`

The goal of the `deco_hop` task is to generate molecules that contain specific substructures while not containing others; these substructures are shown in Figure 8. The final deco_hop score is calculated as the mean of substructure presence/absence (binary score) and Tanimoto distance to the target molecule. We showcase our best-generated molecules from the deco_hop task in Figure 9.

D.2.2 Task 2: `EGFR docking`

The goal of the `EGFR docking` task is to generate molecules that have a low binding affinity to epidermal growth factor receptors in humans (EGFR, PBD ID: 2RGP. Molecules are docked against EGFR using AutoDock Vina [16], and the output is the docking score of the binding process. We showcase our best-generated molecules from this task in Figure 10.

Task 1: maximize QED (↑), minimize SA (↓), maximize JNK3 (↑)		Summation (Top-10 AUC) (↑)	Hypervolume (↑)	Structural diversity (↑)	Objective diversity (↑)
Summation	Graph-GA	1.967 ± 0.088	0.713 ± 0.083	0.741 ± 0.115	0.351 ± 0.079
	MolLEO (MolSTM)	2.177 ± 0.178	0.625 ± 0.162	0.803 ± 0.011	**0.362 ± 0.074**
	MolLEO (BioT5)	1.946 ± 0.222	0.592 ± 0.199	**0.805 ± 0.196**	0.341 ± 0.091
	MolLEO (GPT-4)	2.367 ± 0.044	**0.752 ± 0.085**	0.726 ± 0.063	0.292 ± 0.076
Pareto optimality	Graph-GA	2.120 ± 0.159	0.603 ± 0.082	0.761 ± 0.034	0.219 ± 0.117
	MolLEO (MolSTM)	2.234 ± 0.246	0.472 ± 0.248	0.739 ± 0.015	0.306 ± 0.085
	MolLEO (BioT5)	2.325 ± 0.164	0.630 ± 0.120	0.724 ± 0.020	0.339 ± 0.062
	MolLEO (GPT-4)	**2.482 ± 0.057**	0.727 ± 0.038	0.745 ± 0.057	0.322 ± 0.104
Task 2: maximize QED (↑), minimize SA (↓), maximize GSKB3 (↑)					
Summation	Graph-GA	2.186 ± 0.069	0.719 ± 0.055	0.778 ± 0.122	0.379 ± 0.101
	MolLEO (MolSTM)	2.349 ± 0.132	0.303 ± 0.024	**0.820 ± 0.010**	**0.440 ± 0.037**
	MolLEO (BioT5)	2.306 ± 0.120	0.693 ± 0.093	0.803 ± 0.013	0.384 ± 0.045
	MolLEO (GPT-4)	2.543 ± 0.014	**0.832 ± 0.024**	0.715 ± 0.052	0.391 ± 0.021
Pareto optimality	Graph-GA	2.339 ± 0.139	0.640 ± 0.034	0.816 ± 0.028	0.381 ± 0.071
	MolLEO (MolSTM)	2.340 ± 0.254	0.202 ± 0.054	0.770 ± 0.017	0.188 ± 0.010
	MolLEO (BioT5)	2.299 ± 0.203	0.645 ± 0.127	0.759 ± 0.022	0.371 ± 0.047
	MolLEO (GPT-4)	**2.631 ± 0.023**	0.820 ± 0.024	0.646 ± 0.017	0.191 ± 0.026
Task 3: maximize QED (↑), JNK3 (↑), minimize SA (↓), GSKB3 (↓), DRD2 (↓)					
Summation	Graph GA	3.856 ± 0.075	0.162 ± 0.048	0.821 ± 0.024	0.226 ± 0.057
	MolLEO (MolSTM)	4.040 ± 0.097	0.474 ± 0.193	0.783 ± 0.027	**0.413 ± 0.064**
	MolLEO (BioT5)	3.904 ± 0.092	0.266 ± 0.201	**0.828 ± 0.005**	0.243 ± 0.081
	MolLEO (GPT-4)	4.017 ± 0.048	0.606 ± 0.086	0.726 ± 0.064	0.289 ± 0.050
Pareto optimality	Graph GA	4.051 ± 0.155	0.606 ± 0.052	0.688 ± 0.047	0.294 ± 0.074
	MolLEO (MolSTM)	3.989 ± 0.145	0.381 ± 0.204	0.792 ± 0.030	0.258 ± 0.019
	MolLEO (BioT5)	3.946 ± 0.115	0.367 ± 0.177	0.784 ± 0.020	0.367 ± 0.177
	MolLEO (GPT-4)	**4.212 ± 0.034**	**0.696 ± 0.029**	0.641 ± 0.037	0.266 ± 0.062

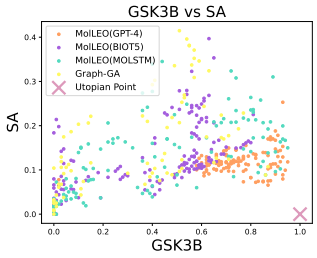
Table 11: Multi objective results. The best model for each task is bolded.

E Prompts

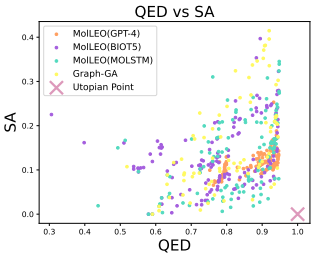
For each model, we show the prompts used for each task. When creating the prompts, we followed the format of examples in the original source code as closely as possible.

MolLEO (MolSTM) prompts

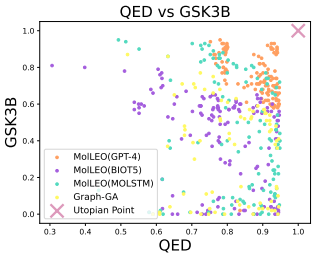
```
QED
This molecule is like a drug.

JNK3
This molecule inhibits JNK3.

GSK3β
This molecule inhibits GSK3B.

DRD2
This molecule inhibits DRD2.

mestranol_similarity
This molecule looks like Mestranol.

thiothixene_rediscovery
This molecule looks like Thiothixene.

perindopril_mpo
This molecule looks like Perindopril and has 2 aromatic rings.

ranolazine_mpo
This molecule looks like Ranolazine, is highly permeable, is hydrophobic, and has 1 F
atom.

sitagliptin_mpo
This molecule has the formula C16H15F6N5O, looks like Sitagliptin, is highly permeable,
and is hydrophobic.

Isomers_C9H10N2O2PF2Cl
```



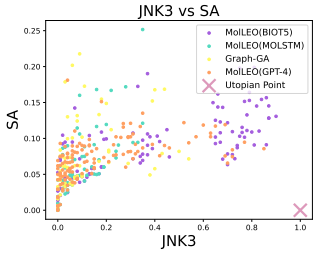
(a) GSK3β vs. SAscore in task 2



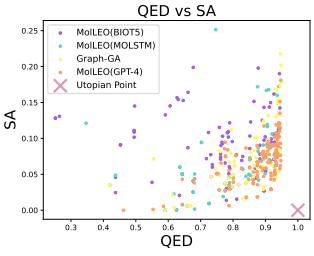
(b) QED vs. SAscore in task 2



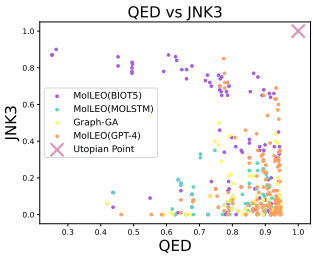
(c) QED vs. GSK3β in task 2



(d) JNK3 vs. SAscore in task 1



(e) QED vs. SAscore in task 1



(f) QED vs. JNK3 in task 1

Figure 7: 2D plots for multi-objective optimization in task 1 and task 2

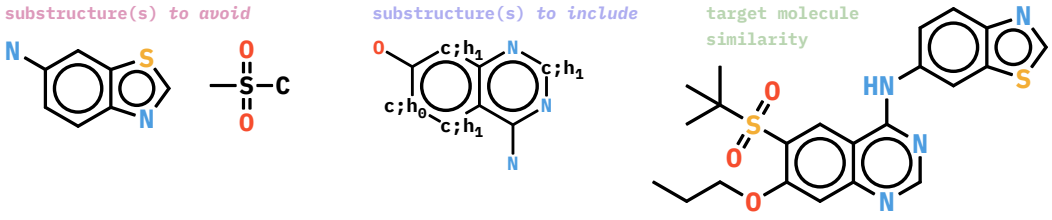
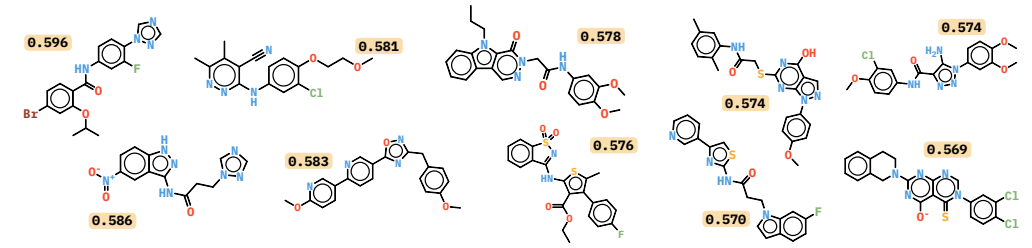
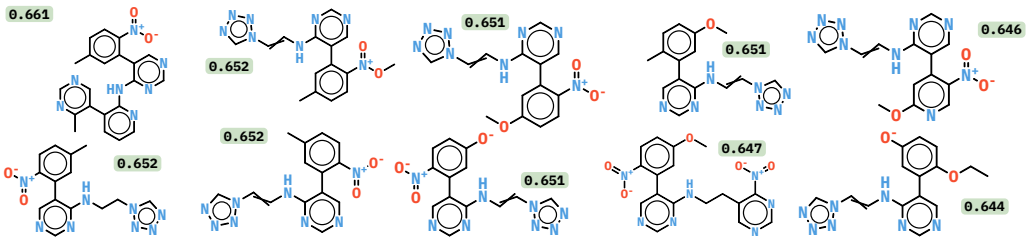


Figure 8: **Substructures to be included or avoided in the deco_hop task.**

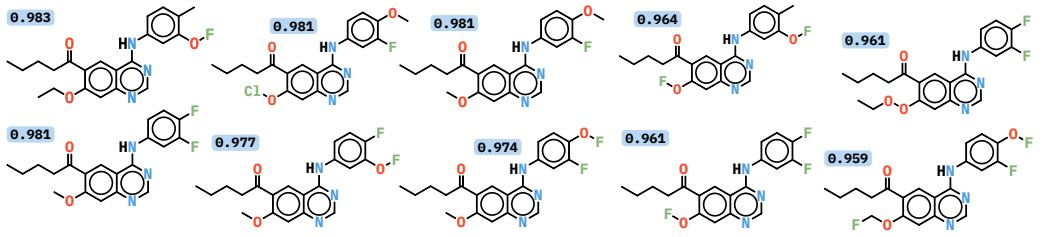
```
This molecule has the atoms C9H10N2O2PF2Cl.

deco_hop
This molecule does not contain the substructure [#7]-c1ccc2ncsc2c1, which is a
6-aminobenzothiazole, does not contain the substructure CS([#6])(=O)=O, which is a
dimethyl sulfone, contains the scaffold, which is a 4-amino-7-hydroxyquinazoline, and
is similar to CCCOc1cc2ncnc(Nc3ccc4ncsc4c3)c2cc1S(=O)(=O)C(C)(C)C.

scaffold_hop
This molecule does not contain the scaffold [#7]-c1n[c;h1]nc2[c;h1]c(-[#8])[c;h0]
[c;h1]c12, contains the substructure [#6]-[#6]-[#6]-[#8]-[#6]~[#6]~[#6]~[#6]~[#6]-
[#7]-c1ccc2ncsc2c1, and is similar to CCCOc1cc2ncnc(Nc3ccc4ncsc4c3)c2cc1S(=O)(=O)C(C)(C)C.

maxjnk3_maxqed_minsa
This molecule is synthesizeable, looks like a drug, and inhibits JNK3.

maxgsk3b_maxqed_minsa
This molecule is synthesizeable, looks like a drug, and inhibits GSK3B.

maxjnk3_maxqed_minsa_mindrd2_mingsk3b
This molecule is synthesizable, does not inhibit GSKB3, does not inhibit DRD2, looks
like a drug, and inhibits JNK3.

3pbl_docking
This molecule inhibits DRD3.

2rgp_docking
This molecule inhibits EGFR.
```



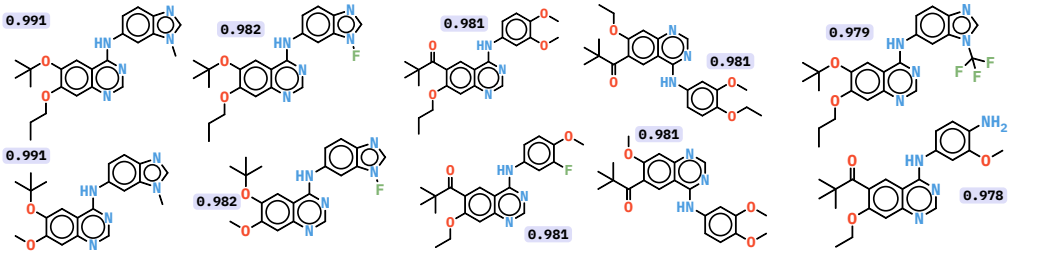
(a) Graph-GA



(b) MOLLEO (MOLSTM)



(c) MOLLEO (BIOT5)



(d) MOLLEO (GPT-4)

Figure 9: Molecules with best deco_hop scores generated by Graph-GA and each MOLLEO model. The deco_hop score of each molecule is written beside it. Higher deco_hop scores are better.

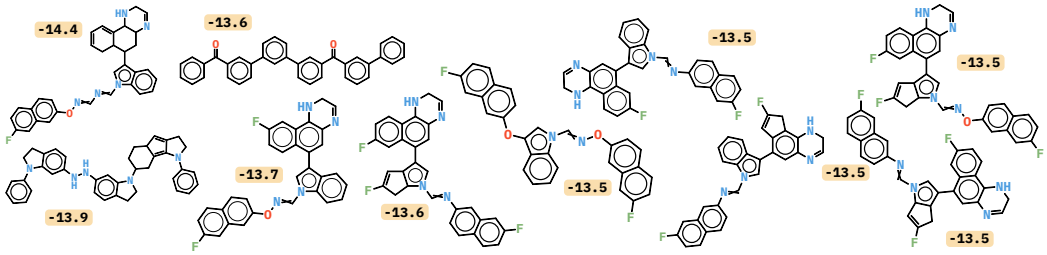
3eml_docking
This molecule binds to adenosine receptor A2a.

MOLLEO (BIOT5) prompts

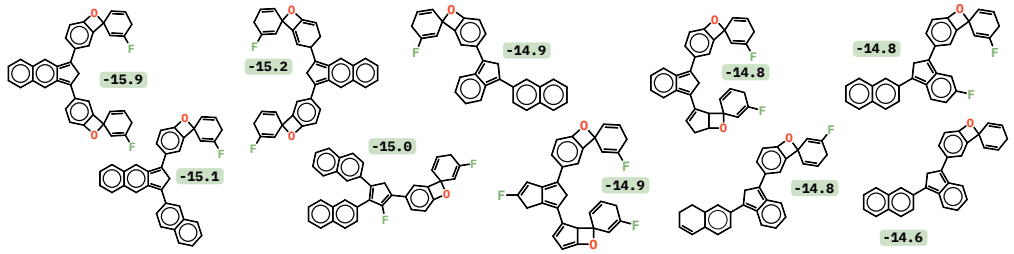
Template:

Definition: You are given a molecule SELFIES. Your job is to generate a SELFIES molecule that {OBJECTIVE}. Now complete the following example - Input: <bom>{selfies_input}<eom> Output:

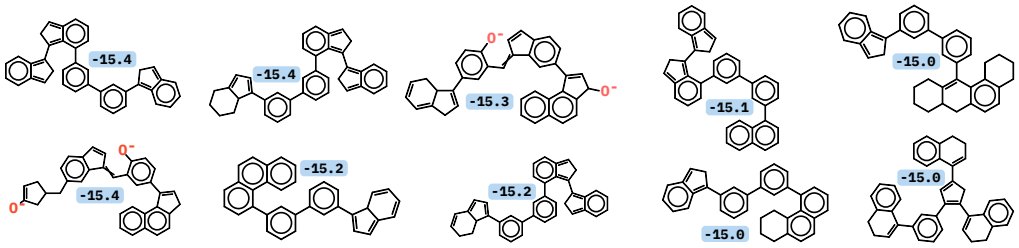
QED
OBJECTIVE: looks more like a drug



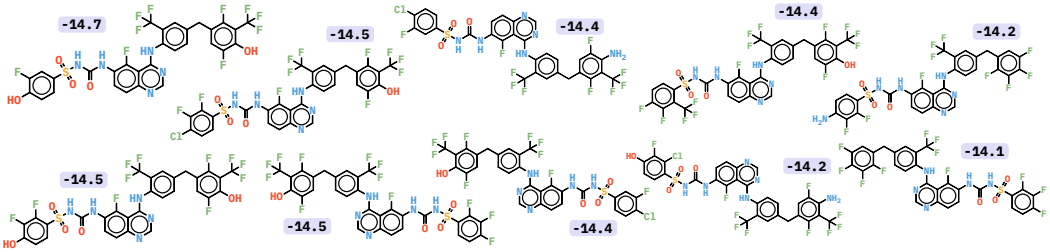
(a) Graph-GA



(b) MOLLEO (MOLSTM)



(c) MOLLEO (BIOT5)



(d) MOLLEO (GPT-4)

Figure 10: Molecules with best EGFR docking scores generated by Graph-GA and each MOLLEO model. The docking score of each molecule is written beside it. Lower docking scores are better.

JNK3
OBJECTIVE: inhibits JNK3 more

GSK3β
OBJECTIVE: inhibits GSK3B more

DRD2
OBJECTIVE: inhibits DRD2 more

mestranol_similarity
OBJECTIVE: looks more like Mestranol

thiothixene_rediscovery
OBJECTIVE: looks more like Thiothixene

perindopril_mpo

OBJECTIVE: looks more like Perindopril and has 2 aromatic rings

sitagliptin_mpo
OBJECTIVE: has the formula C16H15F6N5O, looks more like Sitagliptin, is highly permeable, and is hydrophobic

ranolazine_mpo
OBJECTIVE: looks more like Ranolazine, is highly permeable, is hydrophobic, and has 1 F atom

Isomers_C9H10N2O2PF2Cl
OBJECTIVE: has the formula C9H10N2O2PF2Cl

deco_hop
OBJECTIVE: does not contain the substructure [#7]-c1ccc2ncsc2c1, does not contain the substructure CS([#6])(=O)=O, contains the scaffold [#7]-c1n[c;h1]nc2[c;h1]c(-[#8])[c;h0][c;h1]c12, and is similar to [C][C][C][O][C][=C][C][=N][C][=N][C][Branch1][#C][N][C][=C][C][=C][N][=C][S][C][Ring1][Branch1][=C] [Ring1][=Branch2][=C][Ring1][S][C][=C][Ring2][Ring1][Ring2][S][=Branch1][C][=O][=Branch1][C][=O][C][Branch1][C][C][Branch1][C][C][C]

scaffold_hop
OBJECTIVE: does not contain the scaffold [#7]-c1n[c;h1]nc2[c;h1]c(-[#8])[c;h0][c;h1]c12, contains the substructure [#6]-[#6]-[#6]-[#8]-[#6]~[#6]~[#6]~[#6]~[#6]-[#7]-c1ccc2ncsc2c1, and is similar to the SELFIES [C][C][C][O][C][=C][C][=N][C][=N][C][Branch1][#C][N][C][=C][C][=C][N][=C][S] [C][Ring1][Branch1][=C][Ring1][=Branch2][=C][Ring1][S][C][=C][Ring2][Ring1][Ring2][S] [=Branch1][C][=O][=Branch1][C][=O][C][Branch1][C][C][Branch1] [C][C][C]

maxjnk3_maxqed_minsa
OBJECTIVE: is a greater inhibitor of JNK3, is more synthesizable and is more like a drug.

maxgsk3b_maxqed_minsa
OBJECTIVE: inhibits GSK3B more, is more synthesizable and is more like a drug.

maxjnk3_maxqed_minsa_mindrd2_mingsk3b
OBJECTIVE: is a greater inhibitor of JNK3, is more like a drug, inhibits GSK3B less, inhibits DRD2 less and is more synthesizable.

3pbl_docking
OBJECTIVE: inhibits DRD3 more

2rgp_docking
OBJECTIVE: inhibits EGFR more

3eml_docking
OBJECTIVE: binds better to adenosine receptor A2a

MOLLEO (GPT-4) prompts

Template:

I have two molecules and their {TASK}. {OBJECTIVE_DEFINITION}

(Smiles of Parent A, objective score of Parent A) (Smiles of Parent B, objective score of Parent B)

Please propose a new molecule that {OBJECTIVE}. You can either make crossover and mutations based on the given molecules or just propose a new molecule based on your knowledge.
Your output should follow the format: {«<Explanation»>: $EXPLANATION, «<Molecule»>: box{$Molecule}}. Here are the requirements:
1. $EXPLANATION should be your analysis.
2. The $Molecule should be the smiles of your propsosed molecule.
3. The molecule should be valid.

QED:
OBJECTIVE: has a higher QED score
TASK: QED scores
OBJECTIVE_DEFINITION: The QED score measures the drug-likeness of the molecule.

JNK3
OBJECTIVE: has a higher JNK3 score
TASK: JNK3 scores
OBJECTIVE_DEFINITION: The JNK3 score measures a molecular's biological activity against JNK3.

GSK3β
OBJECTIVE: has a higher GSK3β score
TASK: GSK3β scores
OBJECTIVE_DEFINITION: The GSK3β score measures a molecular's biological activity against GSK3β.

DRD2
OBJECTIVE: has a higher DRD2 score
TASK: DRD2 scores
OBJECTIVE_DEFINITION: The DRD2 score measures a molecule's biological activity against a biological target named the dopamine type 2 receptor (DRD2).

mestranol_similarity
OBJECTIVE: has a higher mestranol similarity score
TASK: mestranol similarity scores
OBJECTIVE_DEFINITION: The mestranol similarity score measures a molecule's Tanimoto similarity with Mestranol.

thiothixene_rediscovery
OBJECTIVE: has a higher thiothixene rediscovery score
TASK: thiothixene rediscovery scores
OBJECTIVE_DEFINITION: The thiothixene rediscovery score measures a molecule's Tanimoto similarity with thiothixene's SMILES to check whether it could be rediscovered.

perindopril_mpo
OBJECTIVE: has a higher perindopril multi-objective score
TASK: perindopril multi-objective scores
OBJECTIVE_DEFINITION: The perindopril multi-objective score measures the geometric means of several scores, including the molecule's Tanimoto similarity to perindopril and the number of aromatic rings.

sitagliptin_mpo
OBJECTIVE: has a higher sitagliptin multi-objective score
TASK: sitagliptin multi-objective scores
OBJECTIVE_DEFINITION: The sitagliptin multi-objective score measures the geometric means of several scores, including the molecule's Tanimoto similarity to sitagliptin, TPSA score, LogP score and isomer score with C16H15F6N5O.

ranolazine_mpo
OBJECTIVE: has a higher ranolazine multi-objective score
TASK: ranolazine multi-objective scores
OBJECTIVE_DEFINITION: The ranolazine multi-objective score measures the geometric means of several scores, including the molecule's Tanimoto similarity to ranolazine, TPSA score LogP score and number of fluorine atoms.

Isomers_C9H10N2O2PF2Cl:
OBJECTIVE: has a higher isomer score
TASK: isomer scores
OBJECTIVE_DEFINITION: The isomer score measures a molecule's similarity in terms of atom counter to C9H10N2O2PF2Cl.

deco_hop
OBJECTIVE: has a higher deco hop score
TASK: deco hop scores
OBJECTIVE_DEFINITION: The deco hop score is the arithmetic means of several scores, including binary score about whether contain certain SMARTS structures (maximize the similarity to the SMILE '[#7]-c1n[c;h1]nc2[c;h1]c(-[#8])[c;h0][c;h1]c12', while excluding specific SMARTS patterns '[#7]-c1ccc2ncsc2c1' and 'CS([#6])(=O)=O') and (2) the molecule's Tanimoto similarity to PHCO 'CCCOc1cc2ncnc(Nc3ccc4ncsc4c3)c2cc1S(=O)(=O)C(C)(C)C'.

scaffold_hop
OBJECTIVE: has a higher scaffold hop score
TASK: scaffold hop scores
OBJECTIVE_DEFINITION: The scaffold hop score is the arithmetic means of several scores, including (1) binary score about whether contains certain SMARTS structures (maximize the similarity to the SMILE '[#6]-[#6]-[#6]-[#8]-[#6]~[#6]~[#6]~[#6]~[#6]-[#7]-c1ccc2ncsc2c1', while excluding specific SMARTS patterns '[#7]-c1n[c;h1]nc2[c;h1]c(-[#8])[c;h0][c;h1]c12') and (2) the molecule's Tanimoto similarity to PHCO 'CCCOc1cc2ncnc(Nc3ccc4ncsc4c3)c2cc1S(=O)(=O)C(C)(C)C'.

maxjnk3_maxqed_minsa
OBJECTIVE: has a higher QED score, a higher JNK3 score, and a lower SA score
TASK: QED, SA (Synthetic Accessibility), and JNK3 scores.
OBJECTIVE_DEFINITION: *None*

maxgsk3b_maxqed_minsa

OBJECTIVE: has a higher QED score, a higher GSK3β score, and a lower SA score
TASK: QED, SA (Synthetic Accessibility), and GSK3β scores
OBJECTIVE_DEFINITION: *None*

maxjnk3_maxqed_minsa_mindrd2_mingsk3b
OBJECTIVE: has a higher QED score, a higher JNK3 score, a lower GSK3β score, a lower DRD2 score and a lower SA score
TASK: QED, SA (Synthetic Accessibility), JNK3, GSK3β and DRD2 scores
OBJECTIVE_DEFINITION: *None*

2rgp_docking
OBJECTIVE: binds better to EGFR
TASK: docking scores to EGFR
OBJECTIVE_DEFINITION: The docking score measures how well a molecule binds to EGFR. A lower docking score generally indicates a stronger or more favorable binding affinity.

3pbl_docking
OBJECTIVE: binds better to DRD3
TASK: docking scores to DRD3
OBJECTIVE_DEFINITION: The docking score measures how well a molecule binds to DRD3. A lower docking score generally indicates a stronger or more favorable binding affinity.

3eml_docking
OBJECTIVE: binds better to adenosine receptor A2a
TASK: docking scores to adenosine receptor A2a
OBJECTIVE_DEFINITION: The docking score measures how well a molecule binds to adenosine receptor A2a. A lower docking score generally indicates a stronger or more favorable binding affinity.