EFFICIENT EVOLUTIONARY SEARCH OVER CHEMICAL SPACE WITH LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Molecular discovery, when formulated as an optimization problem, presents significant computational challenges as the 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. We consider both commercial and open-source LLMs trained on large corpora of chemical information as crossover and mutation operations in EAs. We perform an extensive empirical study on multiple tasks involving property optimization and molecular similarity, 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.

1 INTRODUCTION

Molecular discovery involves a complex and iterative process where practitioners design molecule candidates, synthesize them, evaluate their properties, and refine initial hypotheses. This process can be slow and laborious, making it difficult to meet the increasing demand for new molecules in domains such as pharmaceuticals, opto-electronics, and energy storage (Tom et al., 2024). This has resulted in significant efforts in developing better search (Kristiadi et al., 2024), prediction (Atz et al., 2021), and generation (Du et al., 2022a) algorithms to generate promising molecular candidates. However, many challenges remain, especially in evaluating molecular properties due to expensive evaluations (oracles), such as wet-lab experiments, bioassays, and computational simulations (Gensch et al., 2022; Stokes et al., 2020).

Natural language processing (NLP) has been increasingly used to represent molecular structures (Chithrananda et al.; Schwaller et al., 2019; Öztürk et al., 2020) and extract chemical knowledge from literature Tshitoyan et al. (2019). The link 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) (Weininger, 1988; Daylight Chemical Information Systems, 2007; Krenn et al., 2020). These methods represent 2D molecular graphs as text, allowing molecules and their text descriptions to be represented using the same modality. Recently, Large Language Models (LLMs) have been utilized in several chemistry-related tasks, such as predicting molecular properties (Guo et al., 2023b; Jablonka et al., 2024), retrieving optimal molecules (Kristiadi et al., 2024; Ramos et al., 2023; Ye et al., 2023), automating chemistry experiments Bran et al. (2023); Boiko et al. (2023); Yoshikawa et al. (2023); Darvish et al. (2024), and generating molecules with target properties (Flam-Shepherd & Aspuru-Guzik, 2023; Liu et al., 2024; Ye et al., 2023). LLMs likely possess some knowledge of these domains because they have been trained on massive amounts of textual data from the internet (including scientific knowledge) to achieve general-purpose language comprehension (White, 2023). While these works have shown that LLMs possess at least a preliminary understanding of chemistry, which is helpful for some chemical discovery tasks, many are based on in-context learning and prompt engineering (Guo et al., 2023b). This can pose issues when designing molecules with strict numerical objectives (AI4Science & Quantum, 2023). Furthermore, methods based on LLM-prompting alone can generate molecules

that are less fit since there is nothing physically grounding an LLM, or they can generate invalid SMILES, meaning that outputs cannot be decoded to chemical structures (Skinnider, 2024).

In this work, we look at evolutionary algorithms (EAs), which are heuristic-based, derivative-free optimization algorithms that only provide the objective value as feedback for a given query point (Song et al., 2024). Because objective functions for molecular properties can be complex (for example, spectral data or bioassays), obtaining their parameters and gradients can be nontrivial. Hence, EAs are often used in molecular discovery (Kadan et al., 2023; Nigam et al., 2024), and have even outperformed many gradient-based methods (Tripp & Hernández-Lobato, 2023). However, one issue with EAs is that they randomly navigate chemical space based on pre-defined genetic operators which are target objective agnostic. At the same time, chemistry-aware LLMs can provide knowledge of the target objective and modify a molecule accordingly, but their outputs are noisy and typically do not generate optimal molecules in a single step. We propose an evolutionary process called **Mol**ecular **L**anguage-Enhanced **E**volutionary **O**ptimization (MOLLEO) that combines EAs with LLMs as crossover and mutation operators to push the distribution of proposed molecules from LLMs to candidates with optimized chemical properties. We first validate the performance of our approach using three flavors of chemistry-aware LLMs on 12 property optimization and molecule similarity tasks in the practical molecular optimization (PMO) benchmark Gao et al. (2022). We find that MOLLEO consistently outperforms existing baselines with all language models tested. We further show strong positive gains of MOLLEO in many multiobjective and protein-ligand docking settings, demonstrating the utility of LLMs as genetic operators. Finally, we conduct an extensive ablation study to illustrate the capabilities and vulnerabilities of LLMs for molecular optimization.

2 Related Work

2.1 Molecular Optimization

Molecular design is a fundamental problem in the chemical sciences and is essential to a wide range of real-world challenges, including medicine, mechanical engineering, and sustainability Sanchez-Lengeling & Aspuru-Guzik (2018). The main obstacle for efficiently searching molecules of interest is the gigantic and rugged chemical space with slow and expensive experimental validations Bohacek et al. (1996); Stumpfe & Bajorath (2012). A classical approach is to make the chemical space combinatorial with expert-defined rules and leverages efficient search and discrete optimization methods to find molecular structures with optimal properties of interest directly. These methods include Monte Carlo Tree Search (MCTS) Yang et al. (2017), reinforcement learning (RL) Olivecrona et al. (2017a); Guo & Schwaller (2023), genetic algorithms (GA) Jensen (2019); Fu et al. (2021); Nigam et al. (2022); Fu et al. (2022) and others Du et al. (2024). In recent years, machine learning methods, especially generative methods, have been applied to accelerate molecular optimization. These deep generative models learn a continuous probabilistic model from empirical datasets and sample new molecular structures from the learned distribution. This class of models include autoregressive models (ARs) Popova et al. (2019); Gao et al. (2021), variational autoencoders (VAEs) Gómez-Bombarelli et al. (2018); Jin et al. (2018), flow models Madhawa et al. (2019); Shi et al. (2020), diffusion models Hoogeboom et al. (2022); Schneuing et al. (2022) and many others Du et al. (2024). Beyond generating arbitrary molecular structures, these models often model a conditional probability distribution on certain molecular properties or combine an optimization loop to search for molecules with optimal properties of interest iteratively. These methods include gradient-based optimization, Bayesian optimization, or latent space traversal methods Gómez-Bombarelli et al. (2018); Griffiths & Hernández-Lobato (2020); Zang & Wang (2020); Du et al. (2022b); Wei et al. (2024).

2.2 Language Models in Chemistry

LLMs have been widely investigated for their knowledge in scientific domains (Achiam et al., 2023; AI4Science & Quantum, 2023), as well as their ability to leverage chemistry tools for experimental tasks in chemical discovery and characterization (Bran et al., 2023; Boiko et al., 2023). Several works have benchmarked LLMs such as GPT-4 on chemistry tasks and found that LLMs can do better than human chemists in some zero-shot question-answering settings, but still struggle with chemical reasoning (Mirza et al., 2024; Guo et al., 2023b). Several smaller, open-source models have been specifically trained or fine-tuned on chemistry text (Taylor et al., 2022; Christofidellis et al., 2023; Pei et al., 2023).

Recently, language models have also been used to guide a given input molecular structure towards specific objective properties (*molecular editing*) (Liu et al., 2023b; Ye et al., 2023). This is important for optimizing compounds that need to satisfy multiple criteria, such as pharmaceutical development, where efficacy needs to be balanced with toxicity, and battery design, where power needs to be balanced with cell lifespan. In this paper, we focus on a different and more goal-oriented problem—molecular optimization to find molecules with desired properties instead of interactive editing. For readers who are interested, we provide more related works about how LLMs have been combined with EAs for code and text generation and benchmarking LLMs in chemical tasks in Appendix A.1.

3 PRELIMINARIES

Single-objective optimization. Molecule optimization for a single property can be formulated as:

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

where m is a molecular structure and M denotes the set of molecules constituting the entire chemical space. $O(m) : M \to \mathbb{R}$ is a (often black-box) scalar-value objective function that evaluates a certain property y of molecule m.

Multi-objective optimization. Oftentimes, molecules need to meet multiple, potentially competing objectives simultaneously. The goal of multi-objective optimization is to find the Pareto-optimal solution, where none of the objectives can be improved without deteriorating any of them Lin et al. (2022). The multi-objective optimization problem extends the single-objective problem as:

$$m^* = \arg\max_{m \in M} F[(O_1(m), O_2(m), ..., O_n(m))] \tag{2}$$

where F represents a composition of each individual objective. The easiest compositions to implement are weighted sums or products, but determining the weight of each objective function is nontrivial Kusanda et al. (2022). Instead, Pareto optimization focuses on another perspective that aims to find a set of nondominated solutions instead of a single optimal solution.

$$S = \{(o \in \mathbb{R}^n, m) : o = O(m), m \in M\} \tag{3}$$

$$P(S) = \{(o, m) \in S : \{o' \in S : o' \succ o, o' \neq o\} = \varnothing\} \tag{4}$$

where S is the set of objective values and $P(S)$ refers to the Pareto frontier of these objective values, $\succ$ denotes Pareto dominance which means o' is strictly better than o. In the end, the set $P(S)$ contains the set of molecules m on the Pareto frontier Geoffrion (1968); Ekins et al. (2010).

Black-box optimization. A single step t of the generic black-box optimization is:

$$x_t \leftarrow A(h_{0:t-1}), \quad y_t \leftarrow f(x_t), \tag{5}$$

where A is the algorithm generates a proposal x_t from the search space X and history h, and y_t is the objective value evaluated on x_t (Song et al., 2024). This process is repeated until some termination criterion T is reached. In our setup, we also extend A with an additional text input information `text_prompt` about the optimized objective O, i.e. $x_t \leftarrow A(\texttt{text_prompt}, h_{0:t-1})$.

For the baseline algorithm A, we consider genetic algorithms (GAs), which are a type of EA (Holland, 1992). GAs start from an initial population and then use biologically-inspired genetic operators, such as crossover, mutation, and selection, to evolve a pool of candidates. Crossover involves selecting a pair of "parents" from the population and combining their elements to generate a single offspring, while mutations operate on single members. Selection pressures can be applied to the population at various points to filter candidates based on objective values or other selection criteria. Once a new pool of candidates is created, the objective function $O : M \to \mathbb{R}$ is evaluated for all members.

4 METHODOLOGY

The MOLLEO framework, shown in Figure 1, builds upon Graph-GA (Jensen, 2019), and operates as follows.

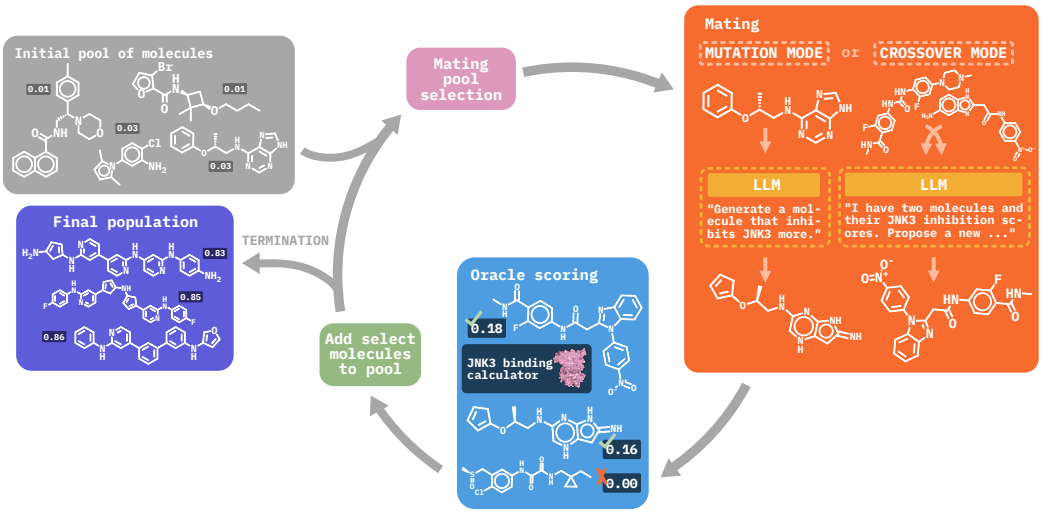

Figure 1: **Overview of MOLLEO.** Given an initial pool of molecules, mates are selected using default Graph-GA (Jensen, 2019) heuristics. Large language models (LLMs) are then engaged as mutation or crossover operators to edit molecules using a text prompt describing the target objective. The offspring molecules are scored using an oracle and the best-scoring ones are passed onto the next generation. This process is repeated until the maximum allowed Oracle calls have been made.

An initial pool of molecules is randomly selected, and their fitness is calculated using a black-box oracle of O. Two parents are then sampled with a probability proportional to their fitnesses and combined using a CROSSOVER operator to generate a child, followed by a random MUTATION. This is repeated `num_crossover` times, and children are added to the pool of offspring. Finally, we measure the fitness of the offspring using O, add them to the population, and keep the n_c of the most fit members to pass on to the next generation. This process is repeated until the maximum number of allowed oracle calls has been made (oracle budget). This process is outlined in Algorithm 1.

We incorporate chemistry-aware LLMs into the structure of Graph-GA. One way we investigate this is by instead of using the random CROSSOVER operation. We generate molecules that maximize the objective fitness function guided by the objective description. We also investigate adding a MUTATION operator to mutate the fittest members of the current population. This selective pressure was motivated by the fact that LLMs can generate noisy edits (in that an edited molecule has lower fitness compared with the initial input molecule, see Appendix A.3). So we construct a filter to select which edited molecules to keep based on structural similarity (Nigam et al., 2022). We sort the existing population by fitness and then apply a mutation to the top population members and add them to the pool of offspring. We prune the offspring 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 A.5.1.

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 A.5.1) and, in what follows, describe the operators that resulted in the best performance.

4.1 GRAPH-GA

• CROSSOVER: (default Graph-GA crossover) Crossover takes place at a ring position or non-ring position with equal likelihood. Parents are cut randomly into fragments and then fragments from both parents are combined. Invalid molecules are filtered out and a random spliced molecule is returned Jensen (2019).
• 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 Jensen (2019).

Algorithm 1: MOLLEO Algorithm

Data: $\mathbb{M}_0$, the initial molecule pool; O, the oracle; n_c, the population size; n_o. the number of offspring
Result: Optimized molecule population $\mathbb{M}^*$
begin
  for $m \in \mathbb{M}_0$ **do**
    Compute $O(m)$;
  $t \leftarrow 0$;
  while $t <$ `oracle_budget and not_converged` **do**
    `offspring = []`;
    while `len(offspring)` $<$ `num_crossovers` **do**
      m_0, m_1 = `sample_molecules`($\mathbb{M}_t$);
      $\tilde{z}$ = `CROSSOVER`(m_0, m_1);
      `offspring.append`($\tilde{z}$);
    $\mathbb{M}_t \leftarrow$ `sorted`($\mathbb{M}_t$);
    $i \leftarrow 0$;
    while $i <$ `num_mutations` **do**
      $\tilde{z}$ = `MUTATION`$(\mathbb{M}_t[i])$;
      `offspring.append`($\tilde{z}$);
      $i \leftarrow i + 1$;
    `offspring` $\leftarrow$ `search(offspring)[:` n_o`]`
    $\mathbb{M}_t \leftarrow$ `offspring`;
    for $m \in \mathbb{M}_t$ **do**
      Compute $O(m)$;
    $\mathbb{M}_t \leftarrow$ `sorted`($\mathbb{M}_t$)$[: n_c]$;
    $t \leftarrow t + 1$;
  $\mathbb{M}^* \leftarrow \mathbb{M}_t$;
  Return $\mathbb{M}^*$;

4.2 MOLLEO (GPT-4)

• `CROSSOVER`: Two parent molecules are sampled using the default Graph-GA algorithm (with a probability proportional to their fitness). 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 A.3.
• `MUTATION`: (default Graph-GA mutation)

4.3 MOLLEO (BIOT5)

• `CROSSOVER`: (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 them to the pool of offspring. Because SELFIES can always be decoded into a molecular structure, there are no issues with BioT5 generating invalid molecules. Given the X offspring from crossover and the Y offspring from this editing procedure, we then select the top n_c offspring overall to keep by selecting the most structurally similar offspring using Tanimoto distance to the fittest molecule in the entire pool Nigam et al. (2022).

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.

Method Objective (↑)	REINVENT	Graph GA	GP BO	MolLEO (MolSTM)	MolLEO (BioT5)	MolLEO (GPT-4)
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**
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**
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
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	**0.740±0.032**	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**
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

4.4 MolLEO (MolSTM)

• `CROSSOVER`: (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 (Liu et al., 2023b). 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} (Irwin et al., 2022), 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:

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

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

$$\mathcal{L}(x_t) = \texttt{-cosine_sim}(E_{Mc}(A_{gc}(x_t)),\ E_{Tc}(\texttt{text_prompt})) + \lambda ||x_t - x_0||_2$$

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}. If s_{out} cannot be decoded into a valid molecule (see Appendix A.3), 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.

5 Experiments

5.1 Experimental Setup

Benchmarks. We evaluate MolLEO on 15 total tasks from two molecular generation benchmarks, PMO Gao et al. (2022) and TDC (Huang et al., 2021). The tasks are organized into the following categories:

1. *Similarity-based optimization*, which optimizes for molecules based on target structures. These include isomer generation based on a target molecular formula (`isomers_c9h10n2o2pf2cl`), similarity to known drugs (`mestranol_similarity`, `thiothixene_rediscovery`), three multi-property optimization tasks (MPO) that aim to rediscover drugs while optimizing for other properties such as LogP and TPSA, and two tasks based on matching scaffolds and substructure motifs (`deco_hop`, `scaffold_hop`). While tasks purely based on rediscovering existing drugs may be trivial for LLMs if they were trained on them, they can signal whether an LLM knows how to make perturbations towards desired molecules, demonstrating basic chemical knowledge.
2. *Property optimization.* We first consider a trivial property optimization task (`QED` Bickerton et al. (2012), 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 previously-trained classifiers that take in a SMILES string and output a value $p \in [0, 1]$, where $p \geq 0.5$ is taken to mean that the molecule inhibits protein activity. Finally, we include three protein-ligand docking tasks from TDC Graff et al. (2021), which are more difficult

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.

		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**

tasks closer to real-world drug design compared to simple physicochemical properties Cieplinski et al. (2020). 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 (Eberhardt et al., 2021), and the output is the docking score of the binding process.

Evaluation metrics. To consider both the optimization ability and sample efficiency of each method, we follow the evaluation metrics in (Gao et al., 2022), using the area under the curve of the top-k average property value (top-k AUC) versus the number of oracle calls as the primary metric. This metric rewards methods that achieve high values with fewer oracle calls. For this study, we set $K = 10$, as it is useful to identify a small, distinct set of molecular candidates suitable for later stages of development. AUC values are min-max scaled to the range [0,1] to standardize results. We restrict the budget of oracle calls to 10,000, although the algorithm terminates early if the average fitness of the top-100 molecules does not change by $1e-3$ within five epochs. We restrict the budget to 1000 calls for the docking experiments since the tasks are significantly more time-consuming. We report all metrics over five random seeds.

For multi-objective optimization, we chose four metrics to evaluate solutions on Pareto frontiers. Top-10 AUC summation, which sums the fitness values for each of the tasks for the top molecules. Hypervolume measures the dominant region under the Pareto optimal solutions in the objective space. Structural diversity reflects the chemical diversity of the Pareto set through the average pairwise Tanimoto similarity between Morgan fingerprints of molecules in the set. Similarly, objective diversity illustrates the coverage of the Pareto frontiers through pairwise Euclidean distance between objective values of the molecules in the Pareto set.

Data. We randomly sample an initial pool of 120 molecules from ZINC 250K (Sterling & Irwin, 2015) following PMO.

Base evolutionary algorithm. We build on Graph-GA (Jensen, 2019) as our baseline evolutionary algorithm owing to its simple architecture and competitive performance. 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. We reduce the number of generated offspring to 7 for the docking experiments and the population size to 12 due to long experiment runtimes.

Base LLMs. We analyze three LLMs in MolLEO as genetic operators in MolLEO. One of the considered models is GPT-4 (Achiam et al., 2023) — a transformer trained using next-token prediction and reinforcement learning from human feedback, which has achieved state-of-the-art performance on chemistry question-answering tasks (Mirza et al., 2024). The other two considered models are open-sourced models trained on domain-specific chemistry text. Compared to GPT-4, they have fewer parameters and have been trained on smaller datasets. BioT5, among other data, is trained on the string representations of molecules called SELFIES to predict missing tokens (including those

at the end of a sentence) (Pei et al., 2023). Because of its ability to generate SELFIES representations, it always produces valid molecules, unlike other models. Finally, MoleculeSTM is trained using a contrastive loss on the pairs of molecular structures and text descriptions and is aligned with an open-source generative model to decode molecule embeddings to SMILES strings (Liu et al., 2023b).

Baselines. We use the top-performing models from the PMO benchmark (Gao et al., 2022) as baselines. These are REINVENT (Olivecrona et al., 2017b), an RNN that uses a reinforcement learning-based policy to guide generation, Graph-GA, Gaussian process Bayesian optimization (GP BO) (Tripp et al., 2021).

Prompts. For each model, we show the prompts in Appendix A.7. 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 A.8.

5.2 Quantitative Evaluation

Incorporating LLMs into GA optimization. To motivate the utility of using chemistry-aware LLMs in GA pipelines, in Figure 2 we show the fitness distribution of an initial pool of random molecules on binding to JNK3. We then do 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, so single-step editing is not sufficient (see Appendix A.3 for quantitative experiments on this). We then show the fitness distributions of the populations as the genetic optimization progresses and find that fitnesses increase to higher values on average given the same number of oracle calls.

Single-objective optimization. We show the results of single-objective optimization across 12 tasks in PMO in Table 1. We report the top-10 AUC for each task, as well as the overall rank of each model. We find that employing any of the three LLMs we tested as genetic operators improves performance over the default Graph-GA, as well as all other baselines we test. Notably, MolLEO (GPT-4) ranks top-1 in 9 out of 12 tasks, demonstrating its utility in molecular generation. MolLEO (BioT5), which incorporates a much smaller language model trained on domain-specific data, obtained a total score close to that of MolLEO (GPT-4), and has the benefit of being free to use. We note that the performance of MolLEO (BioT5) is generally better than that of MolLEO (MolSTM). Empirically, we show in Appendix A.3 that BioT5 produces valid molecules more often and those molecules have higher fitness than those generated by MoleculeSTM on average. This could be due to several reasons, such as differences in training data or poor alignment between the MoleculeSTM encoder and the generative model they use.

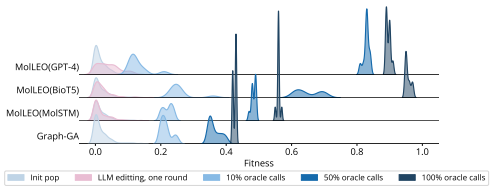

Figure 2: **Population fitness over increasing number of iterations for JNK3 binding.** In the lightest blue, we plot the fitness distribution of the initial molecule pool. We then pass the molecules through a single round of LLM edits (pink curve). Finally, we show the fitness distribution of the top-10 molecules over 10%, 50%, and 100% of the oracle calls made.

For the tasks `deco_hop` and `scaffold_hop`, there was only a small gain for open-source MolLEO models. This is likely because the task description involves negative matching and recognition of SMARTS patterns (e.g., `This molecule does not contain the scaffold [#7]-c1n[c;h1]nc2[c;h1]c(-[#8])[c;h0][c;h1]c12`), which the models were likely not trained on.

We also find that MolLEO has better sample efficiency compared to baseline algorithms, as they can find better optimal molecules with fewer oracle calls; we show this in Appendix Figure 5. 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.

In Figure 3, we plot the average docking scores of the top-10 best molecules of three protein-ligand docking tasks for MolLEO (BioT5) and Graph-GA. These tasks are more complex than simple property optimization and similarity-based optimization, and closer to real-world settings of molecular

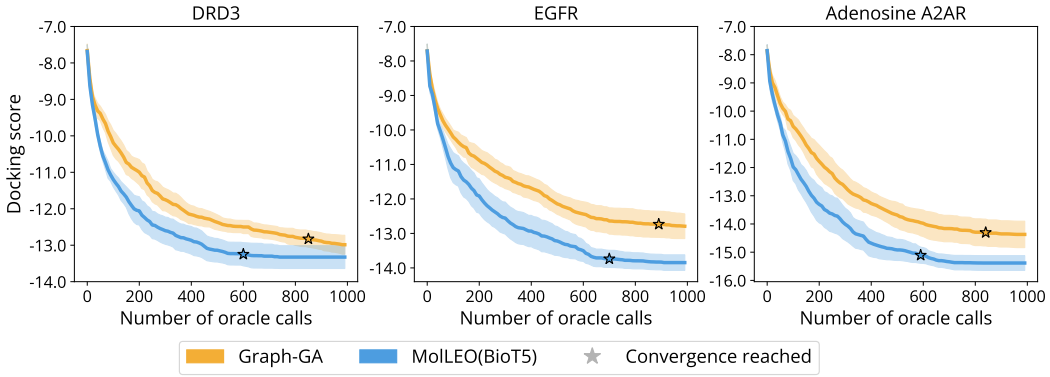

Figure 3: **Average docking score of top-10 molecules when docked against DRD3, EGFR, or Adenosine A2A receptor proteins.** Lower docking scores are better. For each model, we show the convergence point (the point at which the population scores no longer changed) with a star.

generation. We find that MOLLEO (BIOT5) can generate molecules with lower (better) docking scores than the baseline model for all three proteins and converge faster to the optimal set. In practice, this could translate to requiring fewer bioassays to screen molecules, which is both cost- and time-effective.

Multi-objective optimization. In Table 2, we show the results of multi-objective optimization for three tasks. Tasks 1 and 2 are motivated 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 GSK3B (Task 2). Task 3 is even more challenging as it targets five objectives at the same time: maximizing QED, maximizing binding to JNK3, minimizing binding to GSK3B and DRD2, and minimizing SA. We investigate two strategies for multi-objective optimization: (1) summation of individual objectives as a single objective and (2) Pareto set selection, which uses Pareto optimal solutions as the mating pool for the next generation. 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 our best model (MOLLEO (GPT-4)) and Graph-GA on 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 assume that this performance drop may come from their inability to capture large, information-dense contexts. We show the structural and the objective diversity in Appendix A.2.

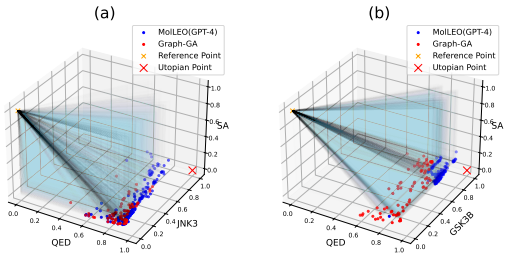

Figure 4: **Pareto frontier visualizations for for Graph-GA and MOLLEO (GPT-4) on multi-objective tasks.** (a) Shows Task 1 (min SA score, max JNK3 binding, max QED) and (b) Shows Task 2 (min SA score, max GSK3B binding, max QED).

6 CONCLUSION, TAKEAWAY AND FUTURE WORK

In this paper, we propose MOLLEO, a marriage between EAs and LLMs that leverages the advantages of both methods to achieve state-of-the-art performance in molecular optimization, encompassing a variety of single- and multi-objective property optimization, rediscovery and structure-based drug design tasks. We demonstrate the capability and versatility of language models in accelerating molecular discovery. **Towards general decision making with LLMs in scientific discovery.** As an initial study, we envision the following directions to be further studied: (1) pre-training/fine-tuning in specific contexts, (2) human-in-the-loop design, (3) deployment in chemical discovery workflow, (4) adapting to other optimization and design problem in science (proteins, RNAs, crystals, etc.) or general domain.

REFERENCES

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.

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. Optimus: Optimization modeling using mip solvers and large language models. *arXiv preprint arXiv:2310.06116*, 2023.

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.

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

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.

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.

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.

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

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.

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

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*, pp. 6140–6157. PMLR, 2023.

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.

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.

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

Yuanqi Du, Tianfan Fu, Jimeng Sun, and Shengchao Liu. Molgensurvey: A systematic survey in machine learning models for molecule design. *arXiv preprint arXiv:2203.14500*, 2022a.

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*, 2022b.

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. ISSN 2522-5839. doi: 10.1038/s42256-024-00843-5. URL https://doi.org/10.1038/s42256-024-00843-5.

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.

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.

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.

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.

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.

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.

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.

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.

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

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.

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

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.

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

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*, 2023a.

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, 2023b.

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

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

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.

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.

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

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.

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

Amit Kadan, Kevin Ryczko, Andrew Wildman, Rodrigo Wang, Adrian Roitberg, and Takeshi Yamazaki. Accelerated organic crystal structure prediction with genetic algorithms and machine learning. *Journal of Chemical Theory and Computation*, 19(24):9388–9402, 2023.

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.

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.

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.

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

Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=QuObT9BTWo`.

Michael Lederman Littman. *Algorithms for sequential decision-making*. Brown University, 1996.

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

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, 2023b.

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. URL `https://openreview.net/forum?id=yRrPfKyJQ2`.

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. URL `https://openreview.net/forum?id=IEduRUO55F`.

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

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.

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.

AkshatKumar Nigam, Robert Pollice, Pascal Friederich, and Alán Aspuru-Guzik. Artificial design of organic emitters via a genetic algorithm enhanced by a deep neural network. *Chemical Science*, 2024.

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

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de novo design through deep reinforcement learning. *CoRR*, abs/1704.07555, 2017b. URL `http://arxiv.org/abs/1704.07555`.

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.

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*, pp. 1102–1123, 2023.

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

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.

Singiresu S Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2019.

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.

Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

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.

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.

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

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

Xingyou Song, Yingtao Tian, Robert Tjarko Lange, Chansoo Lee, Yujin Tang, and Yutian Chen. Position paper: Leveraging foundational models for black-box optimization: Benefits, challenges, and future directions. *arXiv preprint arXiv:2405.03547*, 2024.

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

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.

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

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.

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. doi: 10.26434/chemrxiv-2024-rj946.

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

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. URL https://openreview.net/forum?id=gS3XMun4cl_.

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.

Fabio Urbina, Filippa Lentzos, Cédric Invernizzi, and Sean Ekins. Dual use of artificial-intelligence-powered drug discovery. *Nature Machine Intelligence*, 4(3):189–191, 2022.

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.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

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.

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

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.

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*, 2023.

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. URL https://openreview.net/forum?id=Bb4VGOWELI.

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.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

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.

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.

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*, pp. 617–626, 2020.

A APPENDIX

A.1 EXTENDED RELATED WORK

Benchmarking LLMs on Chemistry Tasks ChemLLMBench proposed and benchmarked LLMs on a variety of eight chemistry tasks from property prediction and reaction prediction to molecule captioning Guo et al. (2023b). Unfortunately, the benchmark results suggest the capabilities of current LLMs are limited in solving those tasks compared to other machine learning models. Notably, most tasks in the benchmark are formulated in a question-answering format, which is different from the optimization problem proposed in this paper. SciBench evaluated the ability of LLMs in taking college-level exams in a variety of science disciplines and found that LLMs fell short of delivering satisfactory performances Wang et al. (2023). One recent work compiled a larger set of question-answer pairs for a more systematic understanding of the abilities of LLMs across the full spectrum of chemistry Mirza et al. (2024).

Language Models and Evolutionary Algorithms Several works have demonstrated the feasibility of using language models to imitate the operator in evolutionary algorithms (Lehman et al., 2023). OPRO Yang et al. (2024) and EvoPrompt (Guo et al., 2023a) progressively improved solutions in optimization tasks when provided with the problem description and past evaluation trajectories in natural language. Later, LMEA Liu et al. (2023a) connected LLMs with an EA by instructing LLMs to select parent solutions from the current population and perform crossover and mutation operations to generate offspring solutions. Rather than directly proposing solutions, FunSearch Romera-Paredes et al. (2024) proposed an evolutionary process with LLMs to solve combinatorial problems with program synthesis. Subsequently, Eureka Ma et al. (2024) leveraged LLMs and EA to design reward functions in reinforcement learning for robot control, demonstrating that reward functions optimized by LLMs can outperform those designed by human experts. This approach has been further extended to multi-agent RL for resource allocation problems in public health Behari et al. (2024).

Large Language Models for Decision Making. Decision-making represents a fundamental challenge in artificial intelligence and cognitive science, which involves the selection of actions to reach certain goals. One branch of decision making is arguably sequential decision making, which involves a sequence of actions including experiment planning, robot navigation, etc Littman (1996). A notable amount of studies have been conducted about in-context learning and prompt engineering to enhance the reasoning capabilities of LLMs Wei et al. (2022); Yao et al. (2024). LLMs are also considered as agents to accomplish tasks with access to tools Wu et al. (2023). Another branch of decision-making comes from optimization problems such that the ultimate goal is to find an optimal solution in which the common tools are mathematical programs in operation research and engineering Rao (2019). The opportunities to use LLMs to solve optimization problems have also been studied, including program search Romera-Paredes et al. (2024), prompt optimization Yang et al. (2023), and mathematical programming AhmadiTeshnizi et al. (2023).

A.2 DIVERSITY ANALYSIS IN MULTI-OBJECTIVE OPTIMIZATION

We show the structural diversity and objective diversity for multi-objective optimization in Table 4.

A.3 PERFORMANCE OF SINGLE-STEP MOLECULE EDITTING

Table 3: **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.

Table 4: Multi objective results. The best model for each task is bolded.

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

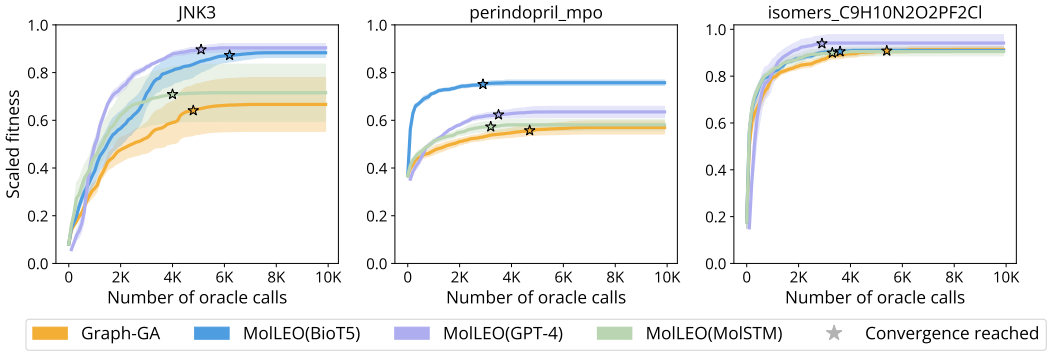

Figure 5: **Average of top-10 molecules generated by MolLEO and Graph-GA models for three tasks over the increasing number of oracle calls.** For each model, we show the convergence point (the point at which the population fitness no longer increased) with a star.

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

A.4 OPTIMIZATION TRENDS OVER SINGLE-OBJECTIVE TASKS.

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

A.5 ABLATION STUDIES

A.5.1 INCORPORATING LLM-BASED GENETIC OPERATORS INTO GRAPH-GA

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.

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

A.5.2 MOLECULESTM HYPERPARAMETER SELECTION

We investigate the selection of three hyperparameters used with open-source LLMs. The first is the number of population members that are selected to undergo LLM-based mutations (Algorithm 1). In Table 6, 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 A.8).

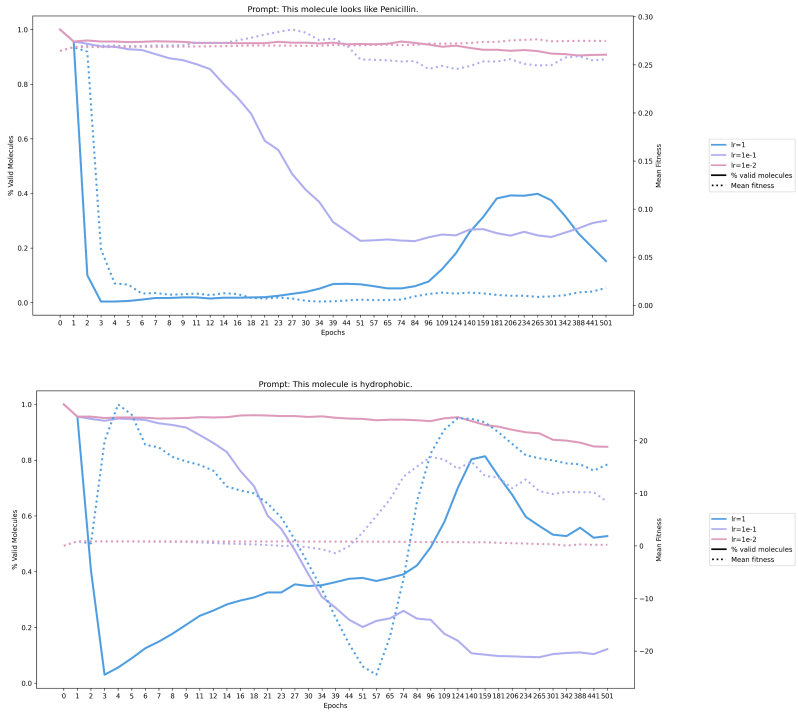

Figure 6: **Mean fitness and percent valid molecules with varying gradient descent epochs and learning rates in MoleculeSTM.**

Table 7: Ablation study on MOLLEO (GPT-4).

	number of offsprings			RAG search		different version of LLMs		different rules		
	20	70	200	w. RAG	w/o. RAG	GPT-3.5	GPT-4	w/o. rule	Gragph GA rule	SMILES GA rule
jnk3	0.731±0.012	0.790±0.027	0.785±0.022	0.830±0.047	0.790±0.027	0.669±0.104	0.790±0.027	0.765±0.047	0.790±0.027	0.774±0.084
isomer_c9h10n2o2pf2cl	0.967±0.010	0.874±0.053	0.960±0.049	0.982±0.018	0.874±0.053	0.902±0.021	0.874±0.053	0.871±0.085	0.874±0.053	0.872±0.029
perindopril mpo	0.573±0.042	0.600±0.031	0.580±0.028	0.717±0.024	0.600±0.031	0.564±0.022	0.600±0.031	0.562±0.042	0.600±0.031	0.583±0.031

Table 6: **Top-10 AUC on JNK3 binding task with varying number of top-scoring candidates selected to undergo LLM-based mutations.**

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

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

A.6 Ablations for GPT-4

We conduct further experiments to understand the sensitivity of MolLEO (GPT-4) with respect to the number of offspring in each generation, retrieval augmentation, language model capability, and different rules from Graph-GA and SMILES-GA in Table 7.

Number of offspring We vary the number of offspring generated in each generation of the GA algorithm and find that increasing the number of offspring often leads to some improvements in the optimization results, but there is no clear trend or instruction on how much is a good value.

Retrieval-augmented Search To understand how retrieval may help LLMs in the optimization process, we remove the retrieval part, which augments the model proposed molecule by the structurally similar molecules from a given dataset. We find that this is an essential step to improve the optimization results of MolLEO (GPT-4).

GPT-3.5 vs. GPT-4 To compare how the capability of LLMs may influence the optimization result, we find that GPT-3.5 performs much worse than GPT-4 on two single- and multi-property optimization tasks but surprisingly better on the solely similarity-based optimization task.

Different rules We validate the effectiveness of incorporating rule-based methods from Graph-GA and find that it brings decent improvement to the overall results, and the Graph-based rule performs slightly better than the SMILES-based rule.

A.7 Prompts

For each of the models, 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 for each model.

MolLEO (MolSTM) prompts

```
QED
This molecule is like a drug.

Isomers_C9H10N2O2PF2Cl
This molecule has the atoms C9H10N2O2PF2Cl.

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

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

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

thiothixene_rediscovery
This molecule looks like Thiothixene.

mestranol_similarity
This molecule looks like Mestranol.

JNK3
This molecule inhibits JNK3.

GSK3B
This molecule inhibits GSK3B.

DRD2
This molecule inhibits DRD2.

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

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

maxgsk3b_maxqed_minsa
This molecule is synthesizable, does not inhibit GSKB3, does not inhibit DRD2, looks like a drug, and inhibits JNK3.

MOLLEO (BIOT5) prompts

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

Isomers_C9H10N2O2PF2Cl
Definition: You are given a molecule SELFIES. Your job is to generate a SELFIES molecule that has the formula C9H10N2O2PF2Cl. Now complete the following example - Input: <bom>{selfies_input}<eom> Output:

perindopril_mpo
Definition: You are given two molecule SELFIES. Your job is to combine them and generate a SELFIES molecule that looks more like Perindopril and has 2 or more aromatic rings Now complete the following example - Input: <bom>{selfies_input}<eom> Output:

sitagliptin_mpo
Definition: You are given a molecule SELFIES. Your job is to generate a SELFIES molecule that has the formula C16H15F6N5O, looks more like Sitagliptin, is highly permeable, and is hydrophobic. Now complete the following example - Input: <bom>{selfies_input}<eom> Output:

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

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

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

JNK3
Definition: You are given a molecule SELFIES. Your job is to generate a SELFIES molecule that inhibits JNK3 more. Now complete the following example - Input: <bom>{selfies_input}<eom> Output:

GSK3B
Definition: You are given a molecule SELFIES. Your job is to generate a SELFIES molecule that inhibits GSK3B more. Now complete the following example - Input: <bom>{selfies_input}<eom> Output:

DRD2
Definition: You are given a molecule SELFIES. Your job is to generate a SELFIES molecule that inhibits DRD2 more. Now complete the following example - Input: <bom>{selfies_input}<eom> Output:

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

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

3pbl_docking
Definition: You are given a molecule SELFIES. Your job is to generate a SELFIES molecule that inhibits DRD3 more. Now complete the following example - Input: <bom>{selfies_input}<eom> Output:

2rgp_docking
Definition: You are given a molecule SELFIES. Your job is to generate a SELFIES molecule that inhibits EGFR more. Now complete the following example - Input: <bom>{selfies_input}<eom> Output:

3eml_docking
Definition: You are given a molecule SELFIES. Your job is to generate a SELFIES molecule that binds better to adenosine receptor A2a. Now complete the following example - Input: <bom>{selfies_input}<eom> Output:

MOLLEO (GPT-4) prompts

I have two molecules and their $Objective$ scores. $The definition of the objective$

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

Please propose a new molecule that has a $Direction$ $Objective$ score. 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: {«<Explaination»>: $EXPLANATION, «<Molecule»>: box{$Molecule}}. Here are the requirements:
1. $EXPLANATION should be your analysis.
2. The $Molecule should be the smiles of your proposed molecule.
3. The molecule should be valid.

QED:
Direction: Higher
Objective: QED
Definition: The QED score measures the drug-likeness of the molecule.

Isomers_C9H10N2O2PF2Cl:
Direction: Higher
Objective: isomer
Definition: The isomer score measures a molecule's similarity in terms of atom counter to C9H10N2O2PF2Cl.

perindopril_mpo
Direction: Higher
Objective: perindopril multi-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
Direction: Higher
Objective: sitagliptin multi-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
Direction: Higher
Objective: ranolazine multi-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.

thiothixene_rediscovery
Direction: Higher
Objective: thiothixene rediscovery
Definition: The thiothixene rediscovery score measures a molecule's Tanimoto similarity with thiothixene's SMILES to check whether it could be rediscovered.

mestranol_similarity
Direction: Higher
Objective: mestranol similarity
Definition: The mestranol similarity score measures a molecule's Tanimoto similarity with Mestranol.

JNK3
Direction: Higher
Objective: JNK3
Definition: The JNK3 score measures a molecular's biological activity against JNK3.

GSK3β
Direction: Higher
Objective: GSK3β
Definition: The GSK3β score measures a molecular's biological activity against GSK3β.

DRD2
Direction: Higher

```
Objective:  DRD2
Definition:  The DRD2 score measures a molecule's biological activity against a
biological target named the dopamine type 2 receptor (DRD2).
deco_hop
Direction:  Higher
Objective:  deco hop
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
Direction:  Higher
Objective:  scaffold hop
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'.
```

A.8 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 (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 10 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.

A.9 Computational Resources

All our experiments are run on NVIDIA A100-SXM4-80GB and T4V2 GPUs. In some of our experiments, we utilize the GPT-4 model. The GPT-4 refers to the "`gpt-4-turbo`" model and in the OpenAI API model with checkpoint version 2023-07-01-preview webpage[1]. All GPT-4 checkpoints are hosted on Microsoft Azure[2].

A.10 Limitations

We note the following limitations of our work. First, more work should be done on proposed candidates from the final optimization result to interpret why the compounds are predicted as optimal, although setting up this analysis is extremely nontrivial. Secondly, while the docking experiments are a more difficult property optimization task, it is still unclear how the model would work on real-world settings.

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

[2] *.openai.azure.com

A.11 BROADER IMPACT

The methods proposed in this paper aim to improve the efficiency in exploring the chemical space to find compounds with desired properties, which can benefit many areas, including drug discovery and materials design. We do not foresee a special negative societal impact of them now, but the dual use of such approaches to find materials for nefarious purposes needs to be avoided (discussed in Urbina et al. (2022)).