ToolMol: Evolutionary Agentic Framework for Multi-objective Drug Discovery

Anonymous Authors[1]

Abstract

Advances in large language models (LLMs) have recently opened new and promising avenues for small-molecule drug discovery. Yet existing LLM-based approaches for molecular generation often suffer from high rates of invalid and low-quality ligand candidates, a result of the syntactic limitations of current models with regard to molecular strings. In this paper, we introduce `ToolMol`, an evolutionary agentic framework for de novo drug design. `ToolMol` combines a multi-objective genetic algorithm with an agentic LLM operator that iteratively updates the ligand population. We build a comprehensive toolbox of RDKit-backed functions that allows our agentic operator to consisently make precise ligand modifications. `ToolMol` achieves state-of-the-art performance on multi-objective property optimization tasks, discovering drug-like and synthesizable ligands that have $> 10\%$ stronger predicted binding affinity compared to existing methods, evaluated on three protein targets. `ToolMol` ligands additionally achieve state-of-the-art results in gold-standard Absolute Binding Free Energy scores, gaining over existing methods by over 35%. By studying chain-of-thought reasoning traces, we observe that tool-calling enables the model to more faithfully execute its planned modifications, efficiently exploiting the strong chemical prior knowledge in LLMs.

1. Introduction

Small molecule drug discovery is a resource-intensive process that requires generated compounds to satisfy many crucial properties, historically requiring many rounds of wet lab trial-and-error. Advances in machine learning have yielded many generative methods that aim to solve this problem. Most previous work has focused on specialized generative models such as VAEs (Eckmann et al., 2022; 2025; Noh et al., 2022), diffusion models (Lee et al., 2023; Zhou et al., 2024; Joshi et al., 2025; Guan et al., 2024; Dorna et al., 2024), and group equivariant diffusion models (Hoogeboom et al., 2022; Vadgama et al., 2026; Liu et al., 2025). Optimization frameworks such as evolutionary algorithms (Jensen, 2019) and Bayesian optimization algorithms (Zhu et al., 2023) have also been applied to this problem. However, existing generative methods often do not target important molecular properties, limiting their ability to generate ligands that simultaneously achieve desirable binding affinity, drug-likeness and synthesizability (Crucitti et al., 2024).

Large Language Models (LLMs) have recently begun to garner interest as a method to generate small molecules, showing promise in generating strong, drug-like ligands (Wang et al., 2025). Unlike specialized generative models, LLMs benefit from large-scale pretraining on domain-relevant, scientific text, which gives them the distinct advantage of inherent familiarity with the optimization task, as well as with the practices and heuristics of chemical research (e.g. common reactions and lead optimization techniques)(White, 2023). Extensive LLM-related works have demonstrated the ability of current LLMs to predict molecular properties (Guo et al., 2023) and generate novel structures (Flam-Shepherd & Aspuru-Guzik, 2023). Most recently, MOLLEO (Wang et al., 2025) proposed a genetic algorithm that directly incorporates LLMs as a mutation and crossover operator to generate molecular offspring, outperforming many specialized generative models in generating ligands with desirable properties.

However, a significant drawback with current LLMs is that they often fail at generating syntactically valid molecular strings, even when prompted to inspect their outputs carefully. We observe that this failure occurs consistently, appearing in more than 30% of attempted molecule generations on average, even on strong reasoning models like GPT-OSS-120B (OpenAI et al., 2025). This significantly hinders the progress of current LLM-based methods. For instance, MOLLEO falls back on weaker non-LLM based, deterministic crossover/mutation operators when the LLM operator fails to generate a valid SMILES result. We claim that this method of allowing an LLM to directly output

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Submitted to the 2026 Workshop on Generative and Agentic AI for Biology (ICML 2026). Do not distribute.

molecular strings is an imperfect method of utilizing LLMs for efficient, property-based drug discovery.

In this work, we introduce `ToolMol`, a novel drug discovery algorithm focused on de novo small molecule drug design. ToolMol combines a multi-objective genetic algorithm with an agentic LLM operator that iterates upon the ligand population using a set of deterministic, RDKit-backed tools. ToolMol solves the problem of invalid molecular generations by exploiting the highly-optimized LLM tool-calling functionality prevalent in current models. Instead of allowing the LLM to modify the molecular string encoding directly, ToolMol abstracts this process by providing the LLM with tools that allow it to simply provide structural parameters for its desired modifications. This not only greatly decreases the number of invalid SMILES strings generated by the LLM, but also yields a significant improvement in the molecular properties of generated ligands, including predicted binding affinity, drug-likeness, and synthesizability.

We summarize the contributions of this work as below:

- We present `ToolMol`, an evolutionary agentic drug discovery framework that combines a multi-objective genetic algorithm with an agentic LLM operator to consistently generate syntactically valid and property-optimizing molecules.
- We achieve state-of-the-art results in multi-objective property optimization across three protein targets, with predicted binding affinity gains exceeding 10% over prior methods, as well as state-of-the-art Absolute Binding Free Energy scores for two studied targets, demonstrating the practical utility of LLMs for de novo drug design.
- We study the effectiveness of our framework through case studies, and observe that the agentic tool-calling process significantly improves concordance between the LLM's reasoning trace and the actual ligand modifications.

2. Related Work

Generative models for molecular design A variety of generative architectures have been developed for molecular design, each learning an implicit distribution over chemical space and leveraging an external oracle to guide generation toward molecules with desirable binding properties. VAE-based approaches such as (Eckmann et al., 2022; Jin et al., 2018; Eckmann et al., 2025; Gómez-Bombarelli et al., 2018) have shown promise, but are generally unaware of the 3D protein structure. To address this, DecompOpt (Zhou et al., 2024) and DecompDiff (Guan et al., 2024) are diffusion models that condition on the protein structure, and are further guided toward optimal ligand molecules by an oracle and some external optimization algorithm. Pocket2Mol (Peng et al., 2025) employs a graph neural network, composed of several encoder and predictor modules, that auto-regressively predicts the location and type of each subsequent ligand atom based on existing ligand atoms and the protein pocket. PAFlow (Zhou et al., 2025) employs a conditional flow-matching algorithm guided by a learnable number-of-atoms predictor model to generate molecules that better match the size of the binding pocket.

Multi-objective frameworks A critical challenge in drug design is multi-objective optimization, as viable drug candidates must satisfy several property criteria simultaneously. TAGMol (Dorna et al., 2024) and DrugDiff (Oestreich et al., 2025) use supplementary guide models to influence the Langevin dynamics during sampling, resulting in generated molecules that satisfy multiple property criteria. Graph-GA (Jensen, 2019) employs an evolutionary algorithm that keeps track of an active population of molecules, applying deterministic crossover and mutation rules to progressively optimize multiple desired properties. OMTRA (Dunn et al., 2025) presents a multi-modal, flexible flow matching model for structure-based drug design. HN-GFN (Zhu et al., 2023) utilizes a multi-objective Bayesian optimization algorithm combined with a GFlowNet to optimize for several properties, including molecular diversity.

LLMs for molecular generation The use of LLMs in drug discovery is currently limited. Current approaches address general-purpose chemistry tasks (Bran et al., 2023; Boiko et al., 2023; Ma et al., 2024; Choi et al., 2026), or fine-tune LLMs to design strong binders in one shot (Sheikholeslami et al., 2025). Genetic algorithms that utilize LLMs for crossover/mutation operations are better suited to the task of property optimization because they are able to incorporate feedback from oracles. MOLLEO (Wang et al., 2025), which represents the state-of-the-art in LLM-guided drug generation, augments Graph-GA (Jensen, 2019) by replacing algorithmic crossovers and mutations with LLM-driven structural modifications, and achieves strong results across three protein targets. The authors demonstrate the potential of these inherently chemistry-aware LLMs to be a competitive generative method in drug discovery. A relevant prior work on tool-calling in biochemistry is El Agente Estructural (Choi et al., 2026), a multimodal framework that equips an LLM with tools for visual and geometric inspection of molecules in 3D space. However, it is designed for human-in-the-loop interaction rather than integration into an optimization loop.

In this work, we design a minimal yet effective toolbox that improves the quality of LLM-suggested crossover and mutation operations within a genetic algorithm. To our knowledge, this is the first work to utilize LLM tool-calling within an evolutionary framework to optimize molecular properties for drug discovery.

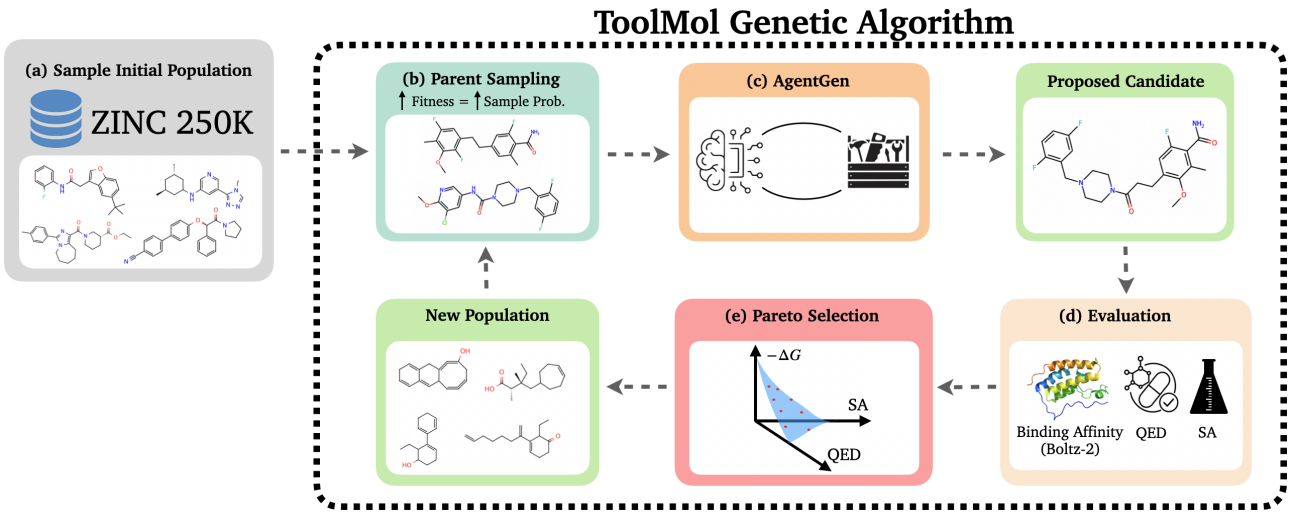


Figure 1. **Overview of ToolMol.** (a) We sample an initial ligand population from ZINC 250K. (b) Parent ligands are sampled for crossovers & mutations with probability proportional to their fitness. (c) An agent with access to a set of modification tools generates new ligands using structures from the selected parents. (d) New offspring are evaluated by an oracle for all relevant objectives. (e) A new population is formed from the non-dominated Pareto frontier of the current population. Steps b → e are repeated until an oracle budget is reached.

3. ToolMol

We introduce ToolMol, an agentic, multi-objective genetic algorithm framework that utilizes a tool-calling LLM to make precise and guided modifications on the ligand population. In this section, we first introduce our optimization problem, then describe the genetic algorithm followed by the tool-calling process that comprise ToolMol.

Problem Statement We can broadly represent our molecular optimization problem as

$$\mathbf{M}^* = \underset{m \in M}{\operatorname{argmax}}\, \Phi(m)$$

where m is any valid molecule (ligand) and M is the entire valid chemical space. Φ is an evaluation function that yields the fitness of m for n objectives. This function can be defined in several ways; one naive approach is to treat the "fitness" of a candidate (the viability or utility of the member) as a weighted sum of all objectives, i.e. $\Phi(m) = \sum_i \mathrm{w}_i * \phi_i(m)$, where ϕ_i is the ith objective evaluator and w_i is the weight given to that objective.[1] These weights are arbitrary and can be difficult to choose in practice. Here, we avoid the need to make such arbitrary choices via partial ordering of molecules and the Pareto frontier. Formally, we can compare two molecules by notating $m' \succ m$, meaning that m' strictly dominates m if and only if $\forall i : \phi_i(m') > \phi_i(m)$. We can then define the optimal set M^* to be the non-dominated Pareto frontier, given by $M^* = \{m \in S : \nexists m', m' \succ m\}$, or the set of all molecules that are not dominated by any other molecule. In other words, M^* is constructed by all ligands for which no other ligand strictly exceeds it in every objective. Because the full chemical space M is far too large to ever be sufficiently explored, we search in a limited subspace of M (roughly determined in practice by random initial seeding), and aim to find the non-dominated Pareto frontier M^* within this subspace.

In this work, we consider 3 objectives: the binding affinity (ΔG, in kcal/mol) of m to a particular protein binding target, as estimated with Boltz-2 (Passaro et al., 2025), the quantitative estimate of drug-likeness (Bickerton et al., 2012, QED), and the estimated synthetic accessibility (Ertl & Schuffenhauer, 2009, SA). All 3 of these objectives are standard targets in generative modeling, with QED and SA ensuring some level of ligand-structure soundness and estimated binding affinity measuring the practical utility of the molecule as a binder of the targeted protein pocket.

3.1. Multi-objective Genetic Algorithm

The underlying framework for the ToolMol algorithm is a multi-objective genetic algorithm (MOGA). The pseudocode for this process is detailed in Algorithm 1 (see Figure 1 for a visual guide). We begin with an initial population $\mathcal{M}_0$ randomly sampled from the ZINC 250K (Sterling & Irwin, 2015) dataset, which provides a good starting base of drug-like & synthesizable structures. We define an "oracle budget" B, which determines how many molecules we evaluate before we terminate the algorithm. We determine the stopping point in this way because predicting binding affinity is generally very computationally expensive; we set

[1] Note that if any objective i is minimizing instead of maximizing, we take the negation of the objective evaluator to be ϕ_i

a hard limit on the number of total evaluations regardless of the population or offspring size.

Algorithm 1 ToolMol Genetic Algorithm

Input: Initial population $\mathcal{M}_0$, offspring size n, oracle budget $B = 1000$
Output: All molecule generations $\mathcal{M}_{\text{out}}$
$\mathcal{M}_c \leftarrow \mathcal{M}_0$ `// Current population`
$\mathcal{M}_{\text{out}} \leftarrow \mathcal{M}_0$ `// All molecules`
while oracle budget $< B$ **do**
  offspring $\leftarrow [\,]$
  for $i \leftarrow 1$ **to** n **do**
    Sample $m_0, m_1 \sim \mathcal{M}_c$ with probability $\propto k^{\Phi(m)}$ for const k
    offspring.append(AGENTGEN(m_0, m_1))
  end
  $\mathcal{M}_{\text{out}} \leftarrow \mathcal{M}_{\text{out}} \cup$ offspring
  $\mathcal{M}_c \leftarrow \mathcal{M}_c \cup$ offspring
  for $m \in \mathcal{M}_c$ **do**
    Compute $\phi_i(m)$ for each objective i
  end
  $\mathcal{M}_c \leftarrow$ PARETOFRONTIER$(\mathcal{M}_c)$
end
return $\mathcal{M}_{\text{out}}$

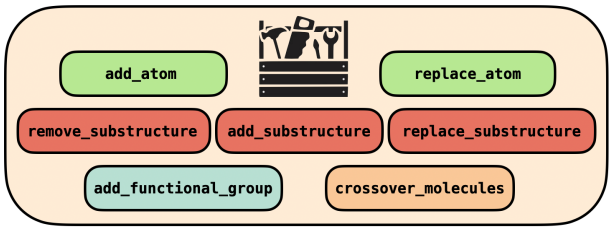
We sample parent molecules m_0, m_1 from the current population with probabilities proportional to $k^{\Phi(m)}$ for some constant k. $\Phi(m)$ is the scalar fitness of m, given by $\Phi(m) = \sum_i f_i(m)$, where $f_i(m)$ is the ith objective scaled to $[0, 1]$.[2] We also explore an alternative sampling method based on Pareto ordering, the results of which may be found in Appendix C. We pass the sampled parent molecules into the agentic LLM operator, AGENTGEN, which is described in the next section. Specifically, for every pair of sampled parents, AGENTGEN creates one new candidate, which is added to the current set of offspring. After n new candidates, we merge the current population with the new offspring. We then evaluate all resulting molecules for each objective and form the next generation by taking the non-dominated Pareto frontier of the current population. We continue the evolution until we exhaust our oracle budget.

3.2. AgentGen: Tool-calling LLM

Next, we describe the AGENTGEN function, which represents the tool-calling process that the agentic operator takes to generate new molecules. This process is inspired by the crossover & mutation operations carried out by classical genetic algorithms, and leverages tool-calling to allow an LLM to act as the sole GA operator.

Given two input molecules m_1, m_2, we first format an initial prompt based on our desired objectives. In order to give the LLM sufficient context to execute accurate tool-calls, we append detailed structure information on both input molecules, given by RDKit. This consists of an identification of every atom in each molecule, its RDKit index, number of substitutable hydrogens, neighboring atoms, centrality within the ligand, and more. This information is necessary for the LLM to provide correct parameters for its tool calls, and aids it in making more informed decisions by providing structural context about the input molecules. Full details about the input and intermediate prompt (given by PROMPTFORMAT) can be found in Appendix B.2.

We then begin the tool-calling iteration process, which proceeds for at most *max_steps* iterations (we use *max_steps* $= 10$ in our experiments, although we note that we rarely ever meet or exceed this threshold). At any given step, the LLM has access to a toolbox of 7 RDKit-backed functions that aid with structural modifications. For all functions, the LLM is responsible for providing all parameters specified in the function definition. If any specification results in an invalid operation (e.g. no available valence), the tool returns a failed state and specifies the particular error.



The LLM-callable functions are listed in the graphic above. For complete details on each tool and its function parameters, see Appendix B.1.

The LLM is encouraged to call `crossover_molecules` on the first step, when initially passed two input molecules from the parent population. On subsequent steps, the LLM is encouraged to use other tools on the molecule resulting from the crossover operation. This simulates how a standard genetic algorithm typically performs a crossover on two parent candidates, then an optional mutation on the resulting offspring (Jensen, 2019).

A single tool call either succeeds and returns the new modified molecule, or fails and returns a message detailing the reason for the error. In both cases, information about the executed tool call and structural details about the new molecule are added to the conversation history for the next tool-calling iteration. This process repeats until we either hit the *max_steps* iteration budget or until the LLM decides that it has made sufficient modifications. Because all modifications are made in the deterministic graph space defined by the RDKit `Mol` object, the final molecule returned by this process is guaranteed to be a valid molecule with a valid

[2] For the unbounded binding affinity metric, we scale by setting a lower bound of 0 kcal/mol and a safe upper bound of -13 kcal/mol, a value that we have never observed our affinity predictor exceed.

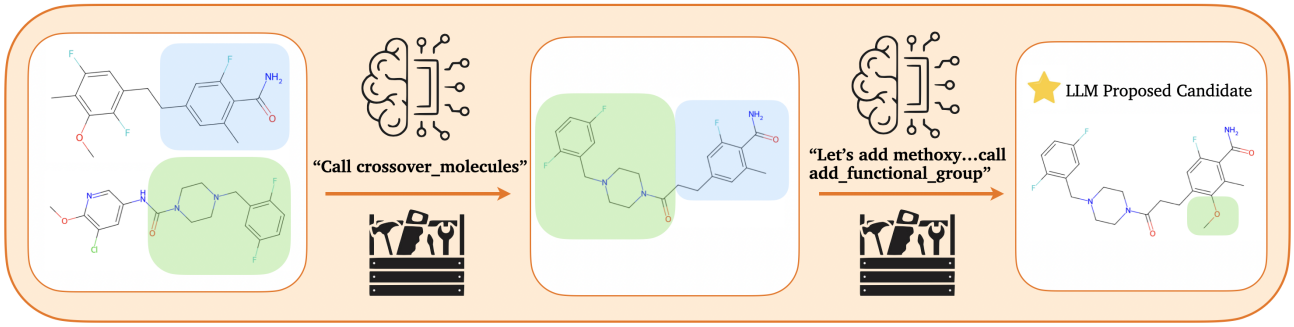


Figure 2. **An example tool-calling process.** The agent first decides to perform a crossover on the input molecules, utilizing `crossover_molecules`. Then it decides to attach a methoxy group to the benzene structure, utilizing `add_functional_group`. At this point, it decides that the modifications are sufficient, and the new molecule is added to the offspring population.

SMILES encoding. The only exception is if the LLM fails to call any function correctly for *max_steps* iterations, which we observe to be extremely uncommon. A short example of this process is demonstrated in Figure 2.

4. Experiments

4.1. Experimental Setup

We evaluate the effectiveness of ToolMol on the multi-objective task of optimizing for protein-ligand binding affinity while preserving drug-likeness and synthetic accessibility.

Targets. In this work, we focus on three functionally & structurally unique protein-binding targets:

1. **c-MET** (MET_HUMAN): Hepatocyte growth factor receptor
2. **BRD4** (BRD4_HUMAN): Bromodomain-containing protein 4
3. **ACAA1** (THIK_HUMAN): 3-ketoacyl-CoA thiolase, peroxisomal

The targets c-MET and BRD4 have significant medicinal chemistry literature (Hong et al., 2016; Organ & Tsao, 2011), while ACAA1 has not been significantly explored as a drug target. In particular, ACAA1 has no associated experimental binding-affinity measurements in BindingDB (Liu et al., 2007), a database of experimentally measured interactions between drug-target proteins and ligands. Thus, the results for ACAA1 report on the performance of the LLM on a target that has not appeared frequently within its pretraining dataset.

Pipeline. For ToolMol, we seed 60 small molecules from ZINC 250K (Sterling & Irwin, 2015) to comprise our initial population, with an offspring size of 35. We utilize an exponential constant $k = 10$ for the parent sampling step.

For all LLM-based components, including baselines, we use GPT-OSS-120B (OpenAI et al., 2025). We estimate ligand-protein binding affinities with the recent biomolecular foundation model Boltz-2 (Passaro et al., 2025). This decision is primarily motivated by the high accuracy Boltz-2 demonstrates in favoring molecules that score highly on gold-standard Absolute Binding Free Energy (Feng et al., 2022, ABFE) metrics. We provide a brief correlation analysis between predicted Boltz-2, ABFE, and AutoDock (Trott & Olson, 2009) scores in Appendix E.2 to further justify this decision.

Baselines. We evaluate ToolMol against the following methods.

1. **Pocket2Mol** (Peng et al., 2025) E(3)-equivariant autoregressive model that generates 3D molecules conditioned on a protein pocket via diffusion.
2. **TAGMol** (Dorna et al., 2024) 3D structure-based framework that decouples diffusion sampling from gradient-based property guidance, using predicted binding affinity, QED, and SA to steer generation.
3. **PAFlow** (Zhou et al., 2025) Conditional flow matching method that leverages a protein-ligand interaction predictor to guide generation toward high-affinity, drug-like molecules.
4. **Graph-GA** (Jensen, 2019) Genetic algorithm operating via predefined crossover and mutation rules on molecular graphs.
5. **ShinkaEvolve** (Lange et al., 2025): We adapt the hybrid MAP-Elites (Mouret & Clune, 2015) and islands algorithm from ShinkaEvolve, an LLM-based evolutionary approach which achieved fantastic results in algorithm design. We adapt this method for drug design, and test variants with both MOLLEO-style LLM mutations and the ToolMol toolbox (details of our implementation are in Appendix D).

6. **MOLLEO** (Wang et al., 2025) MOLLEO extends Graph-GA by replacing predefined genetic operators with an LLM that directly performs crossovers and mutations on molecular candidates, optimizing jointly over binding affinity, QED, and SA.

We note that Pocket2Mol, TAGMol, and PAFlow are designed for oracle-free inference sampling, while all other baseline methods explicitly use affinity feedback as guidance during search. We also note that we use Boltz-2 for affinity prediction for all relevant baselines.

Evaluation Metrics. We consider the following evaluation criteria:

- **Binding Affinity (BA)**: Mean binding affinity (kcal/mol) of the top 10 strongest binding molecules to the particular protein target, predicted by Boltz-2.
- **Filtered Affinity (FA)**: We further filter our ligand sample by only considering ligands that satisfy sufficient Quantitative Estimate of Drug-likeness (Bickerton et al., 2012, QED) and Synthetic Accessibility (Ertl & Schuffenhauer, 2009, SA) scores. We filter by QED > 0.5 and SA < 3.0, then take the mean binding affinity of the top 10 strongest binding molecules that survive this filter. This is a crucial metric that measures the strength of generated ligands that may actually pass the first stage of a real-world wet lab synthesis.
- **Hypervolume (HV)**: Measures the Euclidean volume of the 3D-space (affinity, QED, SA) covered by the non-dominated Pareto frontier formed by the set of all generated molecules. We scale all objectives to $[0, 1]$ and use a reference point of $(1, 1, 1)$ for our calculations.

Results. Table 1 shows the results of running all baselines and ToolMol on all 3 protein targets, along with a Quantitative Estimate of Drug-likeness (Bickerton et al., 2012, QED) maximization objective and a Synthetic Accessibility (Ertl & Schuffenhauer, 2009, SA) minimization objective. We aim to generate molecules that yield the strongest possible binding affinity, that also simultaneously maintain strong-enough QED and SA properties. This is motivated by processes in real-world drug discovery pipelines, where molecules are strongly optimized for binding affinity, but must also be sufficiently drug-like and synthesizable to be realistic candidates. We run all GA-based methods (Graph-GA, MOLLEO, ToolMol) on 5 different seeded sets of initial molecules, and ShinkaEvolve on 3 different seeded sets. Both methods terminate after 1000 Boltz-2 oracle evaluations. We generate 3 seeds of 1000 sampled molecules from Pocket2Mol, TAGMol, and PAFlow for each target, matching the oracle budget for the GA & ShinkaEvolve.

All metrics are reported on a sample of the entire generated ligand pool. For each method, we first Butina cluster the full pool of all generated molecules (with similarity threshold = 0.6), then from each resulting cluster, we take the molecule with the strongest binding affinity in that cluster. This way, we most effectively assess the quality of all structurally unique generations, encouraging diversity in results and favoring consistent strong metrics across a wide region of chemical space.

ToolMol achieves the best average rank across seven methods and nine metrics. Notably, it outperforms every baseline in both multi-objective metrics: filtered mean and hypervolume. We consider these metrics to be the most important, as they are most pertinent to our multi-objective problem statement. ToolMol also consistently outperforms MOLLEO in single-objective binding affinity. Additionally, integrating the ToolMol toolbox into ShinkaEvolve yields the strongest binding affinity scores on two targets. This demonstrates the generalizability of our tool-calling framework, as our toolbox yields consistent improvements when integrated

Table 1. Results of ToolMol compared to generative modeling and LLM-based baselines across three protein targets: c-MET, BRD4, and ACAA1. N/A indicates zero observations across all seeded runs. Best results are **bolded**, second best are underlined

Metric	Pocket2Mol	TAGMol	PAFlow	Graph-GA	ShinkaEvolve (No Tools)	ShinkaEvolve (Tools)	MOLLEO	ToolMol (ours)
c-MET (MET_HUMAN)								
BA (↓)	-11.27 ± 0.29	$\mathbf{-11.45 \pm 0.11}$	-10.63 ± 0.02	-10.21 ± 0.26	-10.17 ± 0.28	-11.08 ± 0.07	-10.15 ± 0.19	-11.00 ± 0.09
FA (↓)	-9.39 ± 0.27	-7.39 ± 0.43	-7.58 ± 0.44	-9.19 ± 0.29	-9.72 ± 0.14	-9.72 ± 0.35	-9.62 ± 0.11	$\mathbf{-10.35 \pm 0.17}$
HV (↑)	0.58 ± 0.007	0.56 ± 0.005	0.50 ± 0.0006	0.57 ± 0.01	0.58 ± 0.01	0.59 ± 0.008	0.60 ± 0.01	$\mathbf{0.62 \pm 0.01}$
BRD4 (BRD4_HUMAN)								
BA (↓)	-10.02 ± 0.24	-9.54 ± 0.10	-8.33 ± 0.09	-9.79 ± 0.26	-9.59 ± 0.09	$\mathbf{-10.80 \pm 0.19}$	-9.87 ± 0.23	-10.64 ± 0.28
FA (↓)	-8.61 ± 0.17	-8.06 ± 0.08	-7.54 ± 0.12	-9.07 ± 0.31	-9.38 ± 0.03	-9.20 ± 0.07	-9.48 ± 0.19	$\mathbf{-9.91 \pm 0.18}$
HV (↑)	0.52 ± 0.02	0.53 ± 0.008	0.43 ± 0.01	0.56 ± 0.02	0.56 ± 0.008	0.57 ± 0.001	0.59 ± 0.01	$\mathbf{0.60 \pm 0.01}$
ACAA1 (THIK_HUMAN)								
BA (↓)	-8.45 ± 0.53	-8.58 ± 0.06	-7.90 ± 0.02	-8.81 ± 0.13	-8.78 ± 0.31	$\mathbf{-10.20 \pm 0.11}$	-8.41 ± 0.41	-9.70 ± 0.23
FA (↓)	-7.39 ± 0.37	-6.67 ± 0.09	N/A	-8.05 ± 0.16	-8.51 ± 0.19	-8.11 ± 0.08	-8.12 ± 0.45	$\mathbf{-8.78 \pm 0.15}$
HV (↑)	0.48 ± 0.09	0.46 ± 0.004	0.33 ± 0.02	0.50 ± 0.01	0.51 ± 0.005	0.53 ± 0.01	0.51 ± 0.02	$\mathbf{0.54 \pm 0.008}$
Avg. Rank (↓)	5.00	6.11	7.56	5.00	3.89	2.56	3.89	**1.56**

into two distinct optimization algorithms: classical genetic algorithms and MAP-Elites.

We note that while certain generative modeling baselines such as Pocket2Mol and TAGMol exceed our method in pure single-objective affinity, the drastic drop in filtered binding affinity for those methods reveals that the crucial QED and SA properties are not sufficiently fulfilled. This implies that the majority of the high scoring affinity compounds are not drug-like or synthesizable enough to be practical. This is further supported by the low hypervolume scores for these generative baselines. Out of all tested methods, ToolMol is the most successful at creating molecular candidates that balance high binding affinity with desirable secondary objectives, reflecting high real-world utility as a generative framework.

4.2. Absolute Binding Free Energy (ABFE)

To further demonstrate the usefulness of ToolMol for real-world drug design, we compute Absolute Binding Free Energy (ABFE) scores for its generated molecules. ABFE uses expensive molecular dynamics simulations to accurately calculate binding free energy (Heinzelmann & Gilson, 2021). It is the current gold-standard for computational binding affinity prediction, and thus reflects a higher degree of accuracy in predicting real-world experimental activity. We benchmark against MF-LAL (Eckmann et al., 2025), a state-of-the-art multi-fidelity approach to drug design that specifically targets ABFE scores through high-fidelity guided VAE decoding. Following the exact ABFE setup from MF-LAL, we evaluate ToolMol on the two targets reported in the MF-LAL paper, c-MET and BRD4. We use two sets of top 15 molecules from ToolMol, one ranked solely on binding energy and the other after applying the QED > 0.5, SA < 3.0 filter. These results are shown in Table 2. Additional details about parameters used in these ABFE calculations can be found in Appendix E.

The top 15 molecules ranked by Boltz-2 predicted affinity achieve strong ABFE scores for both targets, surpassing MF-LAL by a large margin. This comes at a modest cost to secondary objectives, though these molecules still exceed MF-LAL in synthesizability. The top 15 filtered ligands yield slightly weaker ABFE scores, but still beat MF-LAL in every metric, scoring higher on ABFE while maintaining more desirable QED and SA values. Notably, although ABFE feedback is not explicitly included within our optimization pipeline, we outperform the current state-of-the-art method for high ABFE-scoring molecules simply by optimizing Boltz-2 predicted affinity with a tool-assisted LLM. The fact that LLM-generated ligands can achieve state-of-the-art results in this area demonstrates great potential for LLMs to have a real, significant impact in computational drug discovery.

4.3. Ablations

We present ablations that specifically highlight the impact of the tool-calling process, demonstrating the isolated impact of the toolbox provided to the LLM in ToolMol. We compare with MOLLEO, which is the closest methodological neighbor to ToolMol.

First, we ablate the effect of the underlying genetic algorithm on the results. We run ToolMol on the exact MOLLEO genetic algorithm (MOLLEO GA); this isolates the particular impact of introducing function-calling to the crossover/mutation process by removing algorithmic differences of the underlying framework. Second, we ablate the consequences of the high invalid molecule generation rate faced by MOLLEO. We observe that across 1000 molecule generations, MOLLEO will consistently yield ∼350 invalid generations, due to formatting issues or syntactically invalid SMILES. For each invalid generation, MOLLEO immediately falls back on the default Graph-GA crossover / mutation crossovers. We can naively reduce the MOLLEO failed generation rate simply by forcing the LLM to retry its generation until it yields a valid result. This gives a stronger comparison between the MOLLEO generation process and the ToolMol function-calling process per 1000 generations by eliminating a large portion of the extraneous Graph-GA impact within MOLLEO. We give the LLM a maximum of 10 retry steps.

Table 3 compares the original MOLLEO & ToolMol with the two aforementioned ablations. We discuss two interesting results. First, simply integrating ToolMol's tool-

Table 2. ABFE results of top 15 molecules for ToolMol and MF-LAL (Eckmann et al., 2025). ToolMol achieves significantly higher ABFE scores for both sets of evaluated molecules.

Method	c-MET			BRD4		
	ABFE (↓)	QED (↑)	SA (↓)	ABFE (↓)	QED (↑)	SA (↓)
MF-LAL	-6.7 ± 3.1	$\underline{0.63 \pm 0.15}$	3.50 ± 0.58	-6.2 ± 3.9	$\underline{0.59 \pm 0.07}$	3.60 ± 0.55
ToolMol	$\mathbf{-7.96 \pm 2.77}$	0.45 ± 0.21	$\underline{3.26 \pm 0.36}$	$\mathbf{-8.4 \pm 3.9}$	0.27 ± 0.18	$\underline{3.37 \pm 0.45}$
ToolMol (filtered)	$\underline{-7.3 \pm 3.8}$	$\mathbf{0.66 \pm 0.08}$	$\mathbf{2.75 \pm 0.16}$	$\underline{-6.4 \pm 3.5}$	$\mathbf{0.63 \pm 0.10}$	$\mathbf{2.80 \pm 0.14}$

Table 3. Ablations: MOLLEO's invalid generation rate and ToolMol's genetic algorithm. Isolating the impact of the ToolMol toolbox reveals that the tool-calling process significantly improves results.

Target	Metric	MOLLEO	MOLLEO (Retry Failures)	ToolMol (MOLLEO GA)	ToolMol
c-MET	Binding Affinity (↓)	-10.15 ± 0.19	-9.98 ± 0.31	$\mathbf{-11.14 \pm 0.20}$	$\underline{-11.00 \pm 0.09}$
	Filtered Affinity (↓)	-9.62 ± 0.11	-9.72 ± 0.24	$\underline{-10.22 \pm 0.16}$	$\mathbf{-10.35 \pm 0.17}$
	Hypervolume (↑)	$\underline{0.60 \pm 0.01}$	0.57 ± 0.01	$\underline{0.60 \pm 0.02}$	$\mathbf{0.62 \pm 0.01}$
BRD4	Binding Affinity (↓)	-9.87 ± 0.23	-9.67 ± 0.31	$\underline{-10.61 \pm 0.33}$	$\mathbf{-10.64 \pm 0.28}$
	Filtered Affinity (↓)	-9.48 ± 0.19	-9.43 ± 0.26	$\underline{-9.87 \pm 0.32}$	$\mathbf{-9.91 \pm 0.18}$
	Hypervolume (↑)	$\underline{0.59 \pm 0.01}$	0.56 ± 0.02	$\underline{0.59 \pm 0.01}$	$\mathbf{0.60 \pm 0.01}$
ACAA1	Binding Affinity (↓)	-8.41 ± 0.41	-8.04 ± 0.12	$\mathbf{-9.87 \pm 0.18}$	$\underline{-9.70 \pm 0.23}$
	Filtered Affinity (↓)	-8.12 ± 0.45	-7.93 ± 0.11	$\underline{-8.77 \pm 0.24}$	$\mathbf{-8.78 \pm 0.15}$
	Hypervolume (↑)	0.51 ± 0.02	0.49 ± 0.01	$\underline{0.53 \pm 0.02}$	$\mathbf{0.54 \pm 0.008}$
	Avg. Rank (↓)	2.89	3.87	$\underline{1.78}$	**1.22**

box into MOLLEO's genetic algorithm alone yields significant improvements in binding affinity and QED/SA over MOLLEO's LLM-based modifications. Second, we report that the retry method for MOLLEO drops invalid LLM generations down to nearly 0%, yielding a single digit number of invalid strings every 1000 generations. This means that the majority of the final ligand pool is generated by the MOLLEO LLM operator. However, we observe that this does not improve the performance on our evaluation targets, and in fact degrades performance across nearly every metric. Thus, we further isolate the effect of the ToolMol function-calling process by focusing solely on the LLM operator in MOLLEO, and demonstrate that ToolMol's agent-generated ligands are still superior to MOLLEO's LLM-generated ligands in all metrics.

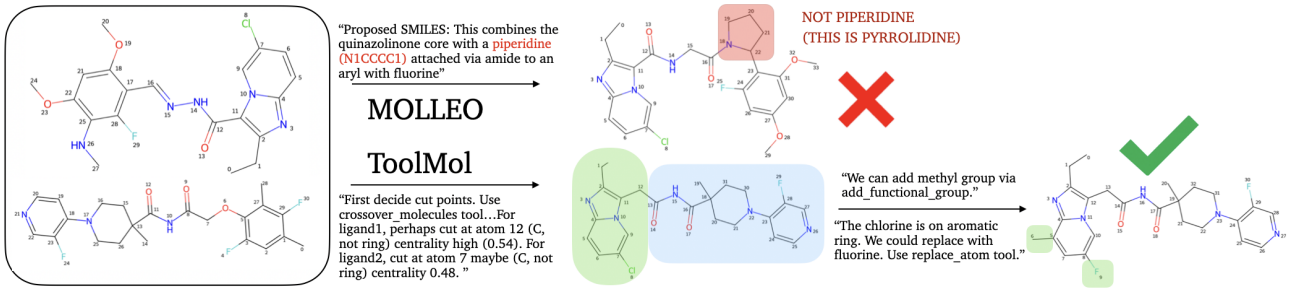
4.4. Case Study: Why does tool-calling improve performance?

To further understand why tool-calling benefits this black-box optimization problem, we simulate a single LLM modification step by providing 2 fixed input molecules (sampled from ZINC 250K), and compare the reasoning traces of GPT-OSS-120B using the ToolMol toolbox to GPT-OSS-120B using the MOLLEO modification scheme. Figure 3 shows how the input molecules are modified by these two methods and examines the modifications against the corresponding reasoning traces.

We observe that in the MOLLEO process, there are critical discrepancies between the planned modifications described in the LLM's reasoning trace and the actual resulting molecule. In contrast, every modification made to the molecule by ToolMol is exactly consistent with what the LLM describes in its reasoning trace. Full reasoning traces for this case study can be found in Appendix A.

For further quantitative confirmation, we repeat this experiment on 10 more pairs of distinct input ligands. Out of 10 generations, ToolMol yields **2** processes where there is a some discrepancy between the reasoning trace and the resulting modification, while MOLLEO yields **7** such er-

Figure 3. ToolMol & MOLLEO modification steps and reasoning traces. MOLLEO fails to execute its planned modifications, while ToolMol successfully executes its ideas.



roneous processes, a significant difference ($p = 0.02$, by 2-sided independent t-test). Both ToolMol errors arise from imprecise function parameters leading to a non-matching change, while MOLLEO frequently misidentifies structures and inserts incorrect groups into the output. Thus we observe that, in general, the generation process of ToolMol allows resulting modifications to be much more aligned with the desired changes outlined by the LLM. By abstracting away the potential for error between LLM reasoning and the generated result, we claim that ToolMol allows us to take better advantage of the immense chemical background that LLMs possess for generating optimal molecules for a particular task. We believe that this is largely why ToolMol achieves significantly stronger binding affinity results on every protein target that we tested on.

5. Discussion & Conclusion

We present ToolMol, an agentic multi-objective drug discovery framework that iteratively optimizes small molecule ligands for protein binding. We build a Pareto-optimizing genetic algorithm that utilizes an exponential-sampling procedure, and combine it with an LLM that has access to a structured toolbox of 7 deterministic, RDKit-backed operations. Rather than requiring an LLM to directly generate or modify molecular string encodings (a task prone to syntactic failure), this agentic framework reduces the potential failure surface by abstracting away the necessity for the LLM to be syntactically perfect in its outputs. We achieve state-of-the-art results in 3 protein-ligand binding tasks, consistently generating molecules that outscore baselines in predicted binding affinity, QED, and SA. Despite not being directly optimized for ABFE score anywhere in its pipeline, ToolMol also achieves exceptional results in this area, demonstrating great potential for LLMs to be a realistic tool in computational drug discovery. We hypothesize that the inherent chemical knowledge that LLMs hold benefits them in designing more realistic molecules, perhaps more similar to what an actual medicinal chemist might synthesize. This gives them a distinct advantage over modern generative models that still seem to struggle at synthesizing compounds that satisfy crucial molecular properties.

Limitations We acknowledge that there have been concerns about the accuracy of Boltz-2 as an affinity predictor. Recent studies have shown that the performance of Boltz-2 degrades significantly when evaluating on novel, out-of-distribution ligand scaffolds and protein targets (Li et al., 2026; Shepard et al., 2026). However, there is still significant evidence that Boltz-2 outperforms the primary industry alternative, AutoDock Vina, in pose and affinity prediction (Liu et al., 2026). We also observe that Boltz-2 shows better agreement with gold-standard Absolute Binding Free Energy than AutoDock, evaluated on one of our own relevant protein targets (see Appendix E.2). We believe that Boltz-2 is still the better option over alternatives like AutoDock, largely also due to the well-studied high inaccuracy of said alternatives (Ramírez & Caballero, 2016; Sasmal et al., 2019).

Impact Statement

Our work has the potential to accelerate the early stages of computational drug discovery by enabling more efficient identification of high-affinity, drug-like ligand candidates. Positive impacts include reducing the time and cost of lead optimization, improving the quality of computationally generated drug candidates entering wet-lab validation, and providing a modular, interpretable framework where an LLM's reasoning for each molecular modification is transparent and traceable through tool calls. By open-sourcing our toolbox, we also lower the barrier to entry for researchers exploring LLM-assisted molecular design. Potential negative impacts include over-reliance on computationally predicted binding scores without sufficient experimental validation, which could lead to wasted resources in downstream wet-lab efforts. We additionally recognize that improved molecular optimization frameworks may be utilized to generate chemically dangerous compounds. However, since our work does not consider complicated properties and requirements for generation and synthesis of harmful compounds, our contribution is not imminently problematic in this direction.

References

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4(2):90–98, January 2012. ISSN 1755-4349. doi: 10.1038/nchem.1243. URL http://dx.doi.org/10.1038/nchem.1243.

Boiko, D. A., MacKnight, R., Kline, B., and Gomes, G. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A. D., and Schwaller, P. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

Choi, C., Zou, Y., Müller, M., Hao, H., Kang, Y., Pérez-Sánchez, J. B., Gustin, I., Xu, H., Wang, A., Vakili, M. G., Crebolder, C., Aspuru-Guzik, A., and Bernales, V. El agente estructural: An artificially intelligent molecular editor, 2026. URL https://arxiv.org/abs/2602.04849.

Crucitti, D., Pérez Míguez, C., Díaz Arias, J. , Fernandez Prada, D. B., and Mosquera Orgueira,

A. De novo drug design through artificial intelligence: an introduction. *Frontiers in Hematology*, Volume 3 - 2024, 2024. ISSN 2813-3935. doi: 10.3389/frhem.2024.1305741. URL https://www.frontiersin.org/journals/hematology/articles/10.3389/frhem.2024.1305741.

Dorna, V., Subhalingam, D., Kolluru, K., Tuli, S., Singh, M., Singal, S., Krishnan, N. M. A., and Ranu, S. Tagmol: Target-aware gradient-guided molecule generation, 2024. URL https://arxiv.org/abs/2406.01650.

Dunn, I., Toft, L., Katz, T., Gupta, J., Shah, R., Hettiarachchi, R., and Koes, D. R. Omtra: A multi-task generative model for structure-based drug design, 2025. URL https://arxiv.org/abs/2512.05080.

Eckmann, P., Sun, K., Zhao, B., Feng, M., Gilson, M. K., and Yu, R. Limo: Latent inceptionism for targeted molecule generation, 2022. URL https://arxiv.org/abs/2206.09010.

Eckmann, P., Wu, D., Heinzelmann, G., Gilson, M. K., and Yu, R. Mf-lal: Drug compound generation using multi-fidelity latent space active learning, 2025. URL https://arxiv.org/abs/2410.11226.

Ertl, P. and Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1(1), June 2009. ISSN 1758-2946. doi: 10.1186/1758-2946-1-8. URL http://dx.doi.org/10.1186/1758-2946-1-8.

Feng, M., Heinzelmann, G., and Gilson, M. K. Absolute binding free energy calculations improve enrichment of actives in virtual compound screening. *Scientific Reports*, 12(1), August 2022. ISSN 2045-2322. doi: 10.1038/s41598-022-17480-w. URL http://dx.doi.org/10.1038/s41598-022-17480-w.

Flam-Shepherd, D. and Aspuru-Guzik, A. Language models can generate molecules, materials, and protein binding sites directly in three dimensions as xyz, cif, and pdb files, 2023. URL https://arxiv.org/abs/2305.05708.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Guan, J., Zhou, X., Yang, Y., Bao, Y., Peng, J., Ma, J., Liu, Q., Wang, L., and Gu, Q. Decompdiff: Diffusion models with decomposed priors for structure-based drug design, 2024. URL https://arxiv.org/abs/2403.07902.

Guo, T., Guo, K., Nan, B., Liang, Z., Guo, Z., Chawla, N. V., Wiest, O., and Zhang, X. What can large language models do in chemistry? a comprehensive benchmark on eight tasks, 2023. URL https://arxiv.org/abs/2305.18365.

Heinzelmann, G. and Gilson, M. K. Automation of absolute protein-ligand binding free energy calculations for docking refinement and compound evaluation. *Scientific Reports*, 11(1), January 2021. ISSN 2045-2322. doi: 10.1038/s41598-020-80769-1. URL http://dx.doi.org/10.1038/s41598-020-80769-1.

Hong, S. H., Eun, J. W., Choi, S. K., Shen, Q., Choi, W. S., Han, J.-W., Nam, S. W., and You, J. S. Epigenetic reader brd4 inhibition as a therapeutic strategy to suppress e2f2-cell cycle regulation circuit in liver cancer. *Oncotarget*, 7 (22):32628–32640, April 2016. ISSN 1949-2553. doi: 10.18632/oncotarget.8701. URL http://dx.doi.org/10.18632/oncotarget.8701.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d, 2022. URL https://arxiv.org/abs/2203.17003.

Jensen, J. H. 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. ISSN 2041-6539. doi: 10.1039/c8sc05372c. URL http://dx.doi.org/10.1039/C8SC05372C.

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

Joshi, C. K., Fu, X., Liao, Y.-L., Gharakhanyan, V., Miller, B. K., Sriram, A., and Ulissi, Z. W. All-atom diffusion transformers: Unified generative modelling of molecules and materials, 2025. URL https://arxiv.org/abs/2503.03965.

Lange, R. T., Imajuku, Y., and Cetin, E. Shinkaevolve: Towards open-ended and sample-efficient program evolution, 2025. URL https://arxiv.org/abs/2509.19349.

Lee, S., Jo, J., and Hwang, S. J. Exploring chemical space with score-based out-of-distribution generation, 2023. URL https://arxiv.org/abs/2206.07632.

Li, Y., Zhan, R.-H., Rao, J., Liu, M., Sang, P., Zeng, X., Zheng, M., Li, X., and Yang, L. Structure-informed machine learning for drug discovery: a task-centric perspective. *Brief. Bioinform.*, 27(1), January 2026.

Liu, C., Vadgama, S., Ruhe, D., Bekkers, E., and Forré, P. Clifford group equivariant diffusion models for 3d molecular generation, 2025. URL https://arxiv.org/abs/2504.15773.

Liu, T., Lin, Y., Wen, X., Jorissen, R. N., and Gilson, M. K. Bindingdb: a web-accessible database of experimentally determined protein-ligand binding affinities. *Nucleic Acids Research*, 35(Database):D198–D201, January 2007. ISSN 1362-4962. doi: 10.1093/nar/gkl999. URL http://dx.doi.org/10.1093/nar/gkl999.

Liu, Y., Tang, H., Niu, T., and Wang, J. A comparative study of deep learning and classical modeling approaches for protein–ligand binding pose and affinity prediction in coronavirus main proteases. *Journal of Chemical Information and Modeling*, 66(1):731–743, 2026. doi: 10.1021/acs.jcim.5c02481. URL https://doi.org/10.1021/acs.jcim.5c02481. PMID: 41429653.

Ma, T., Lin, X., Li, T., Li, C., Chen, L., Zhou, P., Cai, X., Yang, X., Zeng, D., Cao, D., and Zeng, X. Y-mol: A multiscale biomedical knowledge-guided large language model for drug development, 2024. URL https://arxiv.org/abs/2410.11550.

Mouret, J.-B. and Clune, J. Illuminating search spaces by mapping elites, 2015. URL https://arxiv.org/abs/1504.04909.

Noh, J., Jeong, D.-W., Kim, K., Han, S., Lee, M., Lee, H., and Jung, Y. Path-aware and structure-preserving generation of synthetically accessible molecules. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16952–16968. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/noh22a.html.

Oestreich, M., Merdivan, E., Lee, M., Schultze, J. L., Piraud, M., and Becker, M. DrugDiff: small molecule diffusion model with flexible guidance towards molecular properties. *J. Cheminform.*, 17(1):23, February 2025.

OpenAI, :, Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R. K., Bai, Y., Baker, B., Bao, H., Barak, B., Bennett, A., Bertao, T., Brett, N., Brevdo, E., Brockman, G., Bubeck, S., Chang, C., Chen, K., Chen, M., Cheung, E., Clark, A., Cook, D., Dukhan, M., Dvorak, C., Fives, K., Fomenko, V., Garipov, T., Georgiev, K., Glaese, M., Gogineni, T., Goucher, A., Gross, L., Guzman, K. G., Hallman, J., Hehir, J., Heidecke, J., Helyar, A., Hu, H., Huet, R., Huh, J., Jain, S., Johnson, Z., Koch, C., Kofman, I., Kundel, D., Kwon, J., Kyrylov, V., Le, E. Y., Leclerc, G., Lennon, J. P., Lessans, S., Lezcano-Casado, M., Li, Y., Li, Z., Lin, J., Liss, J., Lily, Liu, Liu, J., Lu, K., Lu, C., Martinovic, Z., McCallum, L., McGrath, J., McKinney, S., McLaughlin, A., Mei, S., Mostovoy, S., Mu, T., Myles, G., Neitz, A., Nichol, A., Pachocki, J., Paino, A., Palmie, D., Pantuliano, A., Parascandolo, G., Park, J., Pathak, L., Paz, C., Peran, L., Pimenov, D., Pokrass, M., Proehl, E., Qiu, H., Raila, G., Raso, F., Ren, H., Richardson, K., Robinson, D., Rotsted, B., Salman, H., Sanjeev, S., Schwarzer, M., Sculley, D., Sikchi, H., Simon, K., Singhal, K., Song, Y., Stuckey, D., Sun, Z., Tillet, P., Toizer, S., Tsimpourlas, F., Vyas, N., Wallace, E., Wang, X., Wang, M., Watkins, O., Weil, K., Wendling, A., Whinnery, K., Whitney, C., Wong, H., Yang, L., Yang, Y., Yasunaga, M., Ying, K., Zaremba, W., Zhan, W., Zhang, C., Zhang, B., Zhang, E., and Zhao, S. gpt-oss-120b gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Organ, S. L. and Tsao, M.-S. An overview of the c-met signaling pathway. *Therapeutic Advances in Medical Oncology*, 3(1 suppl):S7–S19, November 2011. ISSN 1758-8359. doi: 10.1177/1758834011422556. URL http://dx.doi.org/10.1177/1758834011422556.

Passaro, S., Corso, G., Wohlwend, J., Reveiz, M., Thaler, S., Somnath, V. R., Getz, N., Portnoi, T., Roy, J., Stark, H., Kwabi-Addo, D., Beaini, D., Jaakkola, T., and Barzilay, R. Boltz-2: Towards accurate and efficient binding affinity prediction. June 2025. doi: 10.1101/2025.06.14.659707. URL http://dx.doi.org/10.1101/2025.06.14.659707.

Peng, X., Luo, S., Guan, J., Xie, Q., Peng, J., and Ma, J. Pocket2mol: Efficient molecular sampling based on 3d protein pockets, 2025. URL https://arxiv.org/abs/2205.07249.

Pettersen, E. F., Goddard, T. D., Huang, C. C., Meng, E. C., Couch, G. S., Croll, T. I., Morris, J. H., and Ferrin, T. E. ¡scp¿ucsf chimerax¡/scp¿: Structure visualization for researchers, educators, and developers. *Protein Science*, 30(1):70–82, October 2020. ISSN 1469-896X. doi: 10.1002/pro.3943. URL http://dx.doi.org/10.1002/pro.3943.

Ramírez, D. and Caballero, J. Is it reliable to use common molecular docking methods for comparing the binding affinities of enantiomer pairs for their protein target? *International Journal of Molecular Sciences*, 17(4):525, April 2016. ISSN 1422-0067. doi: 10.3390/ijms17040525. URL http://dx.doi.org/10.3390/ijms17040525.

Sasmal, S., El Khoury, L., and Mobley, D. L. D3r grand challenge 4: ligand similarity and mm-gbsa-based pose prediction and affinity ranking for bace-1 inhibitors.

Journal of Computer-Aided Molecular Design, 34(2): 163–177, November 2019. ISSN 1573-4951. doi: 10.1007/s10822-019-00249-1. URL http://dx.doi.org/10.1007/s10822-019-00249-1.

Sheikholeslami, M., Mazrouei, N., Gheisari, Y., Fasihi, A., Irajpour, M., and Motahharynia, A. Druggen enhances drug discovery with large language models and reinforcement learning. *Scientific Reports*, 15(1), 2025. ISSN 2045-2322. doi: 10.1038/s41598-025-98629-1. URL http://dx.doi.org/10.1038/s41598-025-98629-1.

Shepard, V., Musin, A., Chebykina, K., Zeninskaya, N. A., Mistryukova, L., Avchaciov, K., and Fedichev, P. O. Harvest: Unlocking the dark bioactivity data of pharmaceutical patents via agentic ai. March 2026. doi: 10.64898/2026.03.15.711910. URL http://dx.doi.org/10.64898/2026.03.15.711910.

Sterling, T. and Irwin, J. J. Zinc 15 – ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, November 2015. ISSN 1549-960X. doi: 10.1021/acs.jcim.5b00559. URL http://dx.doi.org/10.1021/acs.jcim.5b00559.

Trott, O. and Olson, A. J. Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, 31(2):455–461, June 2009. ISSN 1096-987X. doi: 10.1002/jcc.21334. URL http://dx.doi.org/10.1002/jcc.21334.

Vadgama, S., Islam, M. M., Buracas, D., Shewmake, C. A., Moskalev, A., and Bekkers, E. J. Probing equivariance and symmetry breaking in convolutional networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=ghyYc7hgSU.

Wang, H., Skreta, M., Ser, C.-T., Gao, W., Kong, L., Strieth-Kalthoff, F., Duan, C., Zhuang, Y., Yu, Y., Zhu, Y., Du, Y., Aspuru-Guzik, A., Neklyudov, K., and Zhang, C. Efficient evolutionary search over chemical space with large language models, 2025. URL https://arxiv.org/abs/2406.16976.

White, A. D. The future of chemistry is language. *Nature Reviews Chemistry*, 7(7):457–458, May 2023. ISSN 2397-3358. doi: 10.1038/s41570-023-00502-0. URL http://dx.doi.org/10.1038/s41570-023-00502-0.

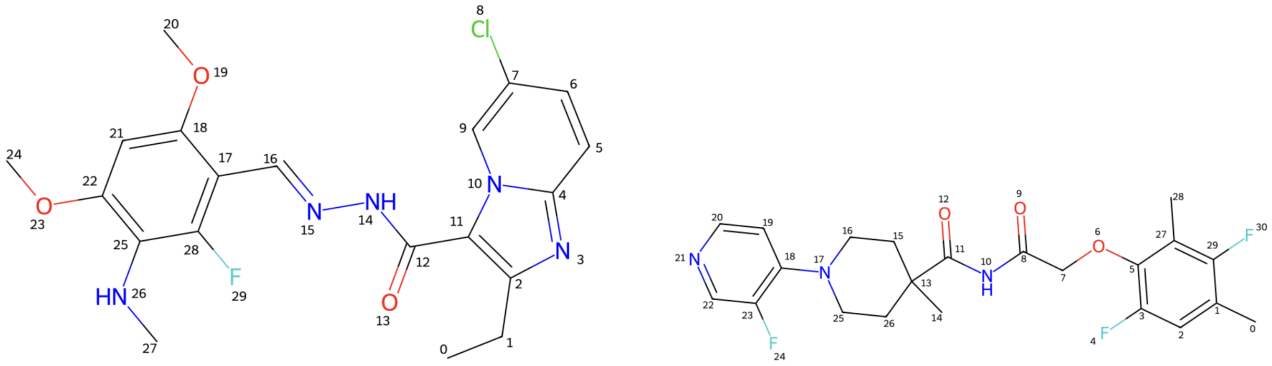
Zhou, J., Qian, H., Tu, S., and Xu, L. Prior-guided flow matching for target-aware molecule design with learnable atom number, 2025. URL https://arxiv.org/abs/2509.01486.

Zhou, X., Cheng, X., Yang, Y., Bao, Y., Wang, L., and Gu, Q. Decompopt: Controllable and decomposed diffusion models for structure-based molecular optimization, 2024. URL https://arxiv.org/abs/2403.13829.

Zhu, Y., Wu, J., Hu, C., Yan, J., Hsieh, C.-Y., Hou, T., and Wu, J. Sample-efficient multi-objective molecular optimization with gflownets, 2023. URL https://arxiv.org/abs/2302.04040.

A. Case Study

We provide further details on the case study presented in Figure 3 of the main paper. Below are the initial two molecules given to both MOLLEO and ToolMol.



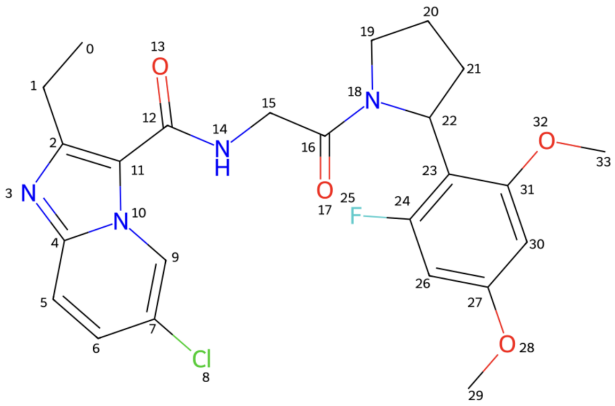
First, we see how the LLM makes modifications **without** the ToolMol toolbox. Here are excerpts from its reasoning trace that describe how it intends to modify the molecule:

MOLLEO Reasoning Trace

"...We can propose a hybrid: combine quinazoline core with a morpholine or piperazine side chain (like in many kinase inhibitors). Add halogen for potency, add -OCH3 groups for interactions. Use a hinge-binding motif: a N,N-dimethylcarboxamide? Actually c-MET hinge binder often uses 4-anilino quinazoline. We could take quinazoline from first (c1nc2ccc(Cl)cn2c1C=O), but modify the N-linked hydrazone to an anilino group with fluorine. Add a piperidine amide from second. Proposed SMILES: something like: "CCc1nc2ccc(Cl)cn2c1C(=O)NCC(=O)N1CCCC1c1ccc(F)cc1O". This combines the quinazolinone core with a piperidine (N1CCCC1) attached via amide to an aryl with fluorine...Letś design stepwise: Core: CCc1nc2ccc(Cl)cn2c1C(=O)N — attach a linker: N-CH2-CH2-C(=O)-N1CCCC1 (piperidine). Then attach a phenyl ring to piperidine nitrogen...Thus SMILES: "CCc1nc2ccc(Cl)cn2c1C(=O)NCC(=O)N1CCCC1c1c(F)cc(OC)cc1OC"."

Below is a visualization of the molecule generated from this process:

Figure 4. This figure shows the initial molecules, and resultant molecules after LLM modifications using MOLLEO and ToolMol. We see that MOLLEO fails to generate the required molecule



We observe that while many parts of the final molecule are consistent with what is described by the reasoning trace, there are certain parts that are entirely inconsistent with the LLM's planned modifications. For instance, it insists that it is using the piperidine ring from ligand 2, but the ring at indices 18-22 is actually a 5-membered pyrrolidine ring; it hallucinates

the SMILES for piperidine multiple times throughout its reasoning trace. The resulting molecule is actually a reasonable crossover between the inputs, but is simply not fully consistent with the reasoning trace that generated it. We observe this trend commonly within MOLLEO generations, where the resulting modification(s) do not exactly align with the desired theoretical modifications that the LLM envisioned.

We contrast this to the ToolMol function-calling process. Here are the reasoning traces for multiple tool steps.

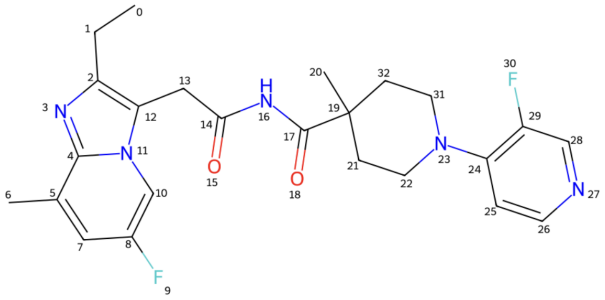
ToolMol Reasoning Trace

Need to use tools to make modifications. First decide cut points. Use crossover_molecules tool. Choose roughly half indices with high centrality not in ring? Must not split at ring index...For ligand1, perhaps cut at atom 12 (C, not ring) centrality high (0.54). For ligand2, cut at atom 7 maybe (C, not ring) centrality 0.48. Use those. Weĺl call crossover.

...Typically, we might want to increase H-bond donors/acceptors, maybe add a fluorine, or replace a chlorine with fluorine to improve metabolic stability. Also maintain drug-like properties...Look at structure: Thereś a fused heterocycle (maybe quinazoline). The chlorine is on aromatic ring. We could replace with fluorine. Use replace_atom tool.

...Goal: better c-MET binding maybe increase H-bond donors/acceptors, aromaticity, add fluorine, etc. Looking at structure: It has fused heteroaryl (pyrimidine) etc. Maybe add a methyl group on the phenyl ring to increase hydrophobic interaction. Which atom index corresponds to aromatic carbon with substitution? Letś inspect aromatic ring indices...So we can add a methyl to a carbon with a hydrogen, e.g., atom 5 has substitutable_hydrogens 1 (per data). Itś a carbon in aromatic ring with one H. So we can add methyl group via add_functional_group.

This describes a 3-step function-calling process that the LLM carried out, first executing a crossover, then making two minor modifications to the resulting molecules. Below is the final generated molecule:



In contrast to the MOLLEO generation process, we observe that every modification made to the molecule is perfectly consistent with what the LLM describes in its reasoning trace. We can see that the ligands were cut at the correct desired indices (12 on ligand 1 and 7 on ligand 2) and merged together to form the desired crossover. Then the model describes replacing the chlorine with fluorine to "improve metabolic stability", which is correctly carried out at index 9. Finally, it wishes to add a methyl group to "increase hydrophobic interaction", which is also correctly done at index 5/6.

B. Full ToolMol Setup

B.1. Toolbox

In this section, we provide more detailed descriptions of each tool in the ToolMol toolbox, and specifications for usage of the parameters.

1. **`add_atom(mol, idx, element, bond)`**
   Adds a single atom of type `element` to the current molecule. The LLM provides the current `mol` as a SMILES string. The LLM must also provide the index of the existing atom that will receive a bond to the newly added atom, as well as the type of the bond (i.e. single, double, or triple).

2. **`replace_atom(mol, idx, element)`**

Replaces the atom at the specified index in the current molecule by a single atom of type `element`, attempting to preserve all existing bonds.

3. **`add_functional_group(mol, idx, group, bond)`**
   Adds a single predefined functional group to the current molecule at the specified index. We build a table mapping common functional groups and rings to their SMILES encodings (e.g. methyl, propyl, phenyl, etc), and expose the table to the LLM. It specifies the `group` parameter in English, and allows the SMILES specification to be handled by the table. Each table entry has a predefined attachment point, and the LLM additionally specifies the bond type for this attachment. The full table can be found in our released codebase.

4. **`add_substructure(mol, idx, substructure, bond)`**
   Adds a manually-specified SMILES string substructure to the current molecule at the specified index. Unlike the previous function, the LLM has full flexibility in adding a custom substructure. It must specify an attachment point in `substructure` in standard `[*1]` notation, as well as the bond type.

5. **`replace_substructure(mol, idx, old_substructure, new_substructure)`**
   Replaces an existing substructure within `mol` with a new substructure. The LLM specifies `old_substructure` using SMARTS (SMILES Arbitrary Target Specification) notation, and provides a new custom substructure as a SMILES. The index parameter here is used as an "anchor" to remove ambiguity if the SMARTS matches multiple substructures; only the substructure that contains the `idx` atom will be selected. Due to the complex nature of removing an entire substructure and reattaching a new one, this function is generally only used on terminal substructures, i.e. substructures that result in only one broken bond when completely removed.

6. **`remove_substructure(mol, idx, substructure)`**
   Removes an entire existing substructure within `mol`, specified by SMARTS notation. Similar to `add_substructure`, the LLM specifies an anchor `idx` to remove ambiguity in the case of multiple matches. Due to the possibility of creating fragments when removing central substructures, this function is also primarily used on terminal substructures.

7. **`crossover_molecules(mol1, idx1, mol2, idx2)`**
   Takes in two separate molecules as input and performs a crossover operation on them. The LLM specifies 2 indices, one for each molecule. The function then attempts to split both molecules at their respective indices. If successful, this results in 4 molecular fragments, from which one of the 4 possible crossover combinations is randomly chosen and returned. This function fails if either index does not result in 2 distinct fragments after the split operation (e.g. if the index is part of a ring structure).

B.2. ToolMol Prompts

Next, we share detailed information about the prompts and information given to the LLM in ToolMol. First, we introduce two functions that provide important structure and property-based information about an input molecule. These functions are non-callable by the LLM, and are instead deterministically provided before every LLM modification step.

1. **`get_ligand_structure(mol)`**
   This function returns the following information for every single atom in the input molecule: `atom_index`, `element`, `num_substitutable_hydrogens`, `num_available_valences`, `num_neighboring_atoms`, `neighbor_indices`, `is_in_ring`, and `centrality`. We observe that the LLM is able to parse this dense atom-wise information quite well, and find this is sufficient to provide the LLM with a solid structural understanding of the ligand. `centrality` is a measure of how central the atom is with respect to the molecular graph, and is calculated with the betweenness centrality formula. This is an important measure for the LLM to choose a crossover location, as we ideally want to split each molecule as evenly as possible to create the most reasonably-sized fragments. Choosing an atom with high centrality is likely to result in an evenly-split molecule.

2. **`calculate_properties(mol)`**
   This function returns basic molecular descriptors for the input directly provided by RDKit. It returns: `QED`, `SA`, `molecular_weight`, `LogP`, `TPSA`, `num_HBond_donors`, `num_HBond_acceptors`, `num_rotatable_bonds`, and `num_aromatic_rings`.This information primarily helps guide the model's decisions at intermediate steps.

First, below is the system prompt given to the LLM for every modification.

ToolMol System Prompt

You are a molecular design agent.
You may ONLY modify molecules using tools.
Only make one modification at a time.
Read the parameter descriptions for the tools very carefully.
Always ensure that your modifications don't break valence rules and do not result in a fragmented molecule.

Next is the full initial ToolMol prompt, given to the LLM at the beginning of a modification step.

Initial ToolMol Prompt

"Goal: I want to improve Binding Affinity against [PROTEIN_TARGET], minimize SA (Synthetic Accessibility), and maximize QED. Recall that a more negative binding affinity is better, and a more positive binding affinity is worse. Please propose a new molecule better than the current molecule. I have given you two candidate ligands. Please propose a new molecule that binds better to [PROTEIN_TARGET]. You are encouraged to make a crossover between the candidate molecules on the first step, then mutate the resulting molecule. Only make a few modifications (at most 3), then respond with FINAL_ANSWER. Do not let molecular weight exceed 700.

1. [LIGAND 1]
Binding Affinity against [PROTEIN_TARGET]: x
SA (Synthetic Accessibility): x
QED: x

2. [LIGAND 2]
Binding Affinity against [PROTEIN_TARGET]: x
SA (Synthetic Accessibility): x
QED: x

Ligand structure and possible attachment points for ligand 1: [`get_ligand_structure(mol1)`]
Ligand structure and possible attachment points for ligand 2: [`get_ligand_structure(mol2)`]
Molecule properties for ligand 1: [`calculate_properties(mol1)`]
Molecule properties for ligand 2: [`calculate_properties(mol2)`]"

We outline a multi-objective goal for the LLM, then provide both initial input ligands and the structure / property information using the functions described above. At this initial step, the model chooses to either use the crossover tool to create a new combination, or just uses another tool to modify one of the given input ligands. In either case, one intermediate ligand is produced. We append the tool called and the result to the conversation history.

Following this, we append the following intermediate prompt to the conversation history.

Intermediate ToolMol Prompts

Output FINAL_ANSWER if you have made sufficient modifications (make at most 3). Ensure that desired properties are maintained.
Current SMILES: [CURR_LIGAND]

Ligand structure and possible attachment points: [`get_ligand_structure(mol)`]
Molecule properties: [`calculate_properties(mol)`]

The model can choose to add additional mutations, and the intermediate prompt is appended to the conversation history after every modification step.

C. Additional Ablations

We provide 3 additional ablations regarding the setup of the genetic algorithm in ToolMol.

Population & Offspring Size We explore using a larger population size of 120 & and larger offspring size of 75, as well a smaller population size of 12 & offspring size of 7. This is in contrast to the population size of 60 & offspring size of 35 used in the ToolMol setup for the main paper.

Pareto Sampling We also explore an alternate sampling method of choosing parent molecules for crossover and mutation. Instead of sampling proportional to an exponentiated weighted scalar, we consider an approach based on Pareto ordering. Given all molecules $\mathcal{M}_c$ in the current population, we can define multiple Pareto frontiers; let P_1 be the set containing the non-dominated frontier on $\mathcal{M}_c$. Then P_2 is the set containing the non-dominated Pareto frontier on $\mathcal{M}_c \setminus P_1$, i.e. the next non-dominated frontier obtained after removing all molecules from the true non-dominated frontier from consideration. Then P_3 can be defined similarly as the set containing the non-dominated Pareto frontier on $\mathcal{M}_c \setminus (P_1 \cup P_2)$. For this alternate sampling method, we first select the top 3 Pareto frontiers (P_1, P_2, P_3) to proceed to the next generation population after a round of offspring. Then, sampling is determined by each molecule's "rank" in the Pareto ordering. Formally, for a given population of size n, the probability of a particular molecule m_j to be selected for crossover / mutation is $P(m_j) = \frac{g(x_j)}{\sum_i g(x_i)}, i \in \{1, ..., n\}$, where $x_i = \{1 \text{ if } m_i \in P_1, 2 \text{ if } m_i \in P_2, 3 \text{ if } m_i \in P_3\}$, and $g(x) = \frac{1}{1+x}$. We consider this sampling method because it aligns well with the Pareto approach we take to multi-objective optimization in the rest of the genetic algorithm.

Table 4 shows the results of the three aforementioned ablations, compared against the ToolMol setup shown in the main paper. We run all ablations on 3 different seeded initial populations.

Table 4. Additional Ablations on ToolMol GA: Population Size & Pareto-rank Sampling

Target	Metric	ToolMol (12 / 7)	ToolMol (120 / 70)	ToolMol (Pareto Sampling)	ToolMol
c-MET	Binding Affinity (↓)	$\mathbf{-11.14 \pm 0.20}$	-10.68 ± 0.12	$\underline{-11.07 \pm 0.19}$	-11.00 ± 0.09
	Filtered Affinity (↓)	-10.22 ± 0.16	-10.13 ± 0.09	$\underline{-10.27 \pm 0.06}$	$\mathbf{-10.35 \pm 0.17}$
	Hypervolume (↑)	$\underline{0.60 \pm 0.02}$	0.59 ± 0.01	$\underline{0.60 \pm 0.01}$	$\mathbf{0.62 \pm 0.01}$
BRD4	Binding Affinity (↓)	-10.61 ± 0.33	$\underline{-10.79 \pm 0.23}$	$\mathbf{-10.95 \pm 0.14}$	-10.64 ± 0.28
	Filtered Affinity (↓)	$\underline{-9.87 \pm 0.32}$	$\underline{-9.87 \pm 0.29}$	-9.67 ± 0.19	$\mathbf{-9.91 \pm 0.18}$
	Hypervolume (↑)	0.59 ± 0.01	$\mathbf{0.60 \pm 0.02}$	0.59 ± 0.004	$\mathbf{0.60 \pm 0.01}$
ACAA1	Binding Affinity (↓)	$\mathbf{-9.87 \pm 0.18}$	-9.54 ± 0.08	$\mathbf{-9.87 \pm 0.19}$	-9.70 ± 0.23
	Filtered Affinity (↓)	-8.77 ± 0.24	$\mathbf{-8.86 \pm 0.25}$	$\underline{-8.81 \pm 0.32}$	-8.78 ± 0.15
	Hypervolume (↑)	0.53 ± 0.02	$\mathbf{0.54 \pm 0.001}$	$\mathbf{0.54 \pm 0.009}$	$\mathbf{0.54 \pm 0.008}$
	Avg. Rank (↓)	2.67	2.56	$\underline{2.00}$	**1.89**

We observe that the configuration of ToolMol in the main paper beats all aforementioned ablations on average across all targets. Thus, we choose to report the values and setup of the rightmost column in comparison to other baselines in our main analysis, although it is likely that the Pareto sampling method is not significantly weaker, and even beats the exponential fitness setup on particular targets.

We also briefly test an alternate value for the exponential constant k used in parent sampling. We use $k = 10$ in the main paper, and test that against $k = e$ here for the c-MET target. Results are shown in Table 5.

Table 5. Ablation on exponential constant k: 10 vs e

Target	Metric	$k = e$	$k = 10$
c-MET	Binding Affinity (↓)	$\mathbf{-11.12 \pm 0.18}$	-11.00 ± 0.09
	Filtered Affinity (↓)	10.32 ± 0.15	$\mathbf{-10.35 \pm 0.17}$
	Hypervolume (↑)	0.61 ± 0.01	$\mathbf{0.62 \pm 0.01}$

We observe that while there is very little difference between the 2 constants, using $k = 10$ marginally improves the multi-objective metrics we care most about, and thus we choose to report that configuration in the main paper.

D. AlphaEvolve / ShinkaEvolve for Drug Discovery

In this section, we outline the ShinkaEvolve-inspired algorithm we built for small-molecule drug discovery. It is a non-sophisticated MAP-Elites approach with independent islands and random migration events. We maintain 4 separate MAP-Elites grids that are referred to as islands. Each grid is actually 1-dimensional, and stores molecule candidates within bins based on their molecular weight. There are 50 bins evenly discretizing molecular weight within the range [200, 900]. Any molecules outside of that range are placed into the outermost bins.

The core LLM modification step occurs when we sample two molecules from a particular grid for crossover / mutation operations, resulting in one new molecule. The sampling procedure closely follows ShinkaEvolve, where parent molecules are sampled based on a balance between fitness and how often that molecule has already been sampled for reproduction. Let $\Phi(m) = \sum_i f_i(m)$ be the fitness of a molecule, where $f_i(m)$ is the ith objective scaled to $[0, 1]$. Let $\alpha = \text{median}(\Phi(m_1), ..., \Phi(m_n))$ for all m_i currently in the MAP-Elites grid. Then let $s_i = \sigma(\lambda * (\Phi(m_i) - \alpha))$, where σ is the sigmoid function and λ is a constant that controls selection pressure. We use $\lambda = 1$ for our experiments. Further, let $h_i = \frac{1}{1+N(m_i)}$, where $N(m_i)$ counts the number of times m_i has already been chosen for reproduction. Thus for each molecule m_i, we have s_i which benefits molecules with high fitness, and h_i which benefits molecules that have not been chosen frequently. The final probability distribution is constructed by $P(m_i) = \frac{w_i}{\sum_j w_j}, j \in \{1, ..., n\}$, where $w_i = s_i * h_i$. After 2 molecules are selected according to this sampling formulation, they undergo LLM crossover / mutation steps either in a similar manner to MOLLEO, or with the ToolMol toolbox.

When a new generated molecule is trying to get placed into the grid, if the bin corresponding to the new molecule is not filled, the new molecule is immediately placed into that bin. If it is occupied, the new molecule replaces the current molecule in the bin only if it has a higher fitness, calculated by the $\Phi(m)$ formula described above.

On initialization of the algorithm, we sample 40 molecules from ZINC 250K, and place them uniformly at random across the 4 islands. Then, each island undergoes 10 independent molecule generations. After all islands complete their generations, a migration event occurs; we sample 2 molecules from each island uniformly at random, then send a copy of those molecules to another island, also chosen uniformly. Whether or not those migrated molecules are accepted into the island depends on the bin they land into and the fitness competition described above. Following ShinkaEvolve, we do not allow the absolute highest fitness molecule from each island to migrate, aiming to preserve some level of diversity between the islands. After 1000 total binding affinity oracle evaluations, the algorithm terminates, and all generated molecules (including ones discarded due to losing to fitness competition) are returned for downstream evaluation.

We do not implement the novelty rejection-sampling or the LLM ensemble described in ShinkaEvolve for our simplistic implementation. We note that this algorithm is still largely unexplored for drug discovery problems, and anticipate that there are likely significant gains to be made beyond our simplistic implementation that was designed primarily as a baseline. We plan to explore further variations of this algorithm for this multi-objective problem in future work.

E. Additional ABFE Information

E.1. ABFE Setup

For our ABFE calculations, we utilize the following Binding Affinity Tool `BAT.py` (Heinzelmann & Gilson, 2021) repository: `https://github.com/GHeinzelmann/BAT.py`. We simulate using OpenMM and the standard SDR method. We use the Boltz-2 predicted ligand pose as the starting pose for the simulation. Because Boltz-2 does not take a protein crystal structure as input and makes a prediction based on the given amino acid sequence, we first align the entire predicted Boltz-2 conformation to the protein crystal structure with ChimeraX (Pettersen et al., 2020), then extract only the ligand pose for ABFE. We observe this alignment to yield an RMSE of under 0.7 angstroms on average; thus we are comfortable using the aligned ligand pose with the crystal structure in ABFE calculations. We do not observe frequent steric clashes resulting from this process.

Our simulation steps parameters for the BAT.py framework are as follows:
eq_steps1 = 500000 (Number of steps for equilibration gradual release)
eq_steps2 = 15000000 (Number of steps for equilibration after release)

m_steps1 = 500000 (Number of steps per window for component m (equilibrium))
m_steps2 = 1000000 (Number of steps per window for component m (production))

n_steps1 = 500000 (Number of steps per window for component n (equilibrium))
n_steps2 = 1000000 (Number of steps per window for component n (production))

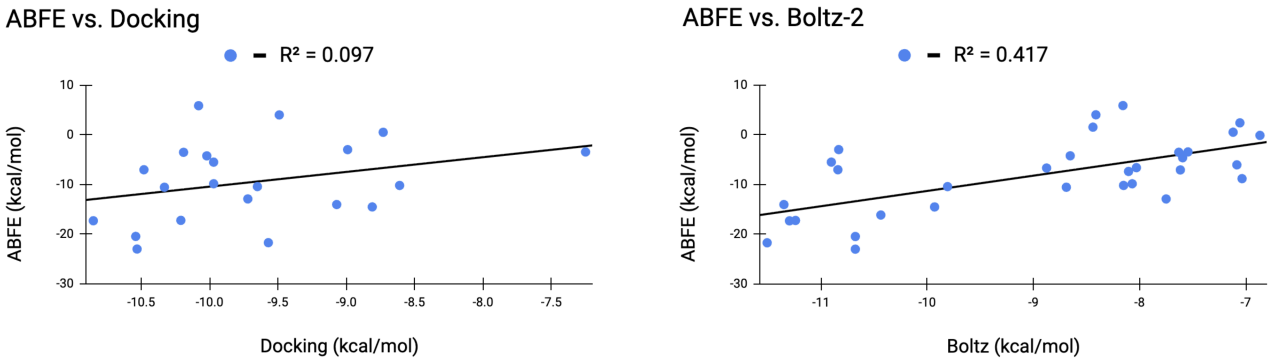
e_steps1 = 250000 (Number of steps per window for component e (equilibrium))
e_steps2 = 500000 (Number of steps per window for component e (production))

v_steps1 = 500000 (Number of steps per window for component v (equilibrium))
v_steps2 = 1000000 (Number of steps per window for component v (production))

On 8 NVIDIA RTX 4090 GPUs, one ABFE calculation typically takes around 12 hours to complete.

E.2. Correlation Analysis: ABFE vs Boltz-2 vs AutoDock

To justify our usage of Boltz-2 as a primary binding affinity oracle, we provide general analysis of the correlation between Boltz-2, AutoDock, and the gold-standard ABFE. In Figure 5, we take 32 compounds for c-MET, 16 of which are known binders, and 16 of which are presumed inactive binders. We calculate the ABFE, Boltz-2, and AutoDock binding affinities for all 32 compounds. We exclude results for any failed AutoDock or Boltz-2 runs.



(a) ABFE vs AutoDock scores

(b) ABFE vs Boltz-2 scores

Figure 5. Comparison of correlation between AutoDock & ABFE and Boltz-2 & ABFE for 32 known compounds for the c-MET protein target. We observe a significantly higher correlation between Boltz-2 and ABFE as compared to AutoDock.

We see that ABFE and AutoDock docking show $r^2 = 0.09$ among the 32 compounds, while ABFE and Boltz-2 show $r^2 = 0.42$. As an oracle nearly 1000x less computationally expensive than ABFE, Boltz-2 shows exceptional correlation with ABFE, especially in comparison to docking. Furthermore, we calculate the ROC-AUC score for Boltz-2 and docking, to see how well they can separate binders from non-binders. Boltz-2 scores 0.95 for this metric, while AutoDock scores 0.84. Due to computational and time constraints regarding expensive ABFE calculations, we are only able to provide results for the c-MET target at this time.

We demonstrate that Boltz-2 has stronger correlation with the most accurate gold-standard computational methods for one of our primary binding targets, which motivates us to employ Boltz-2 as a binding affinity oracle over the current industry-standard AutoDock, which itself has often been noted for its practical inaccuracy. We generally observe Boltz-2 to be approximately a factor of 10 more expensive to run than AutoDock; however, this difference is entirely negligible in comparison to the cost of molecular dynamics methods such as ABFE.