# GP-MoLFormer-Sim: Test Time Molecular Optimization Through Contextual Similarity Guidance

**Jiří Navrátil*, Jarret Ross*, Payel Das, Youssef Mroueh, Samuel C Hoffman, Vijil Chenthamarakshan, Brian Belgodere**

IBM Research
1101 Kitchawan Rd, Yorktown Heights, NY 10598 USA
{jiri,daspa}@us.ibm.com

## Abstract

The ability to design molecules while preserving similarity to a target molecule and/or property is crucial for various applications in drug discovery, chemical design, and biology. We introduce in this paper an efficient training-free method for navigating and sampling from the molecular space with a generative Chemical Language Model (CLM), while using the molecular similarity to the target as a guide. Our method leverages the contextual representations learned from the CLM itself to estimate the molecular similarity, which is then used to adjust the autoregressive sampling strategy of the CLM. At each step of the decoding process, the method tracks the distance of the current generations from the target and updates the logits to encourage the preservation of similarity in generations. We implement the method using a recently proposed ∼47M parameter SMILES-based CLM, GP-MoLFormer, and therefore refer to the method as GP-MoLFormer-Sim, which enables a test-time update of the deep generative policy to reflect the contextual similarity to a set of guide molecules. The method is further integrated into a genetic algorithm (GA) and tested on a set of standard molecular optimization benchmarks involving property optimization, molecular rediscovery, and structure-based drug design. Results show that, GP-MoLFormer-Sim, combined with GA (GP-MoLFormer-Sim+GA) outperforms existing training-free baseline methods, when the oracle remains black-box. The findings in this work are a step forward in understanding and guiding the generative mechanisms of CLMs.

**Extended version** — https://arxiv.org/abs/2506.05628

## 1 Introduction

Finding new functional molecules with desired structure and properties involves solving a constrained multi-objective optimization problem, which is crucial in many applications such as drug discovery and new material design. Given the large size of the molecular space, brute-force search around known substructures is often inefficient and costly for such tasks. Existing molecular optimization algorithms therefore mainly involve reinforcement learning, deep generative models, genetic algorithms, or a combination thereof. Recent works show that traditional genetic algorithm (GA)-based methods with domain-specific operators are competitive when compared to costlier alternatives that involve deep learning models (Gao et al. 2022; Tripp and Hernández-Lobato 2023). Earlier efforts that have successfully combined GAs with deep learning for better search typically require further training of the deep learning model, more specifically of the deep generative model, to adapt the generative policy for generating high-reward samples corresponding to the specific optimization problem (Ahn et al. 2020; Kim et al. 2024).

Different from earlier approaches, here we propose a *training-free* method for equipping a pre-trained deep generative model for targeted search. The proposed method exploits the contextual similarity between a target molecule and a set of generated molecules with a generative chemical language model (CLM). We update the autoregressive decoding policy of the generative model on the fly as a means to guide the generation toward high-reward samples. We use the recently proposed SMILES-based GP-MoLFormer model (Ross et al. 2025) as the base generative CLM to generate molecules, and therefore refer to this test-time contextual similarity-based guided generation method as GP-MoLFormer-Sim. We first show the performance of GP-MoLFormer-Sim on a similarity-based lead optimization task where the goal is to generate molecules of high similarity with respect to a given target molecule in a sample-efficient manner. Experiments on this task show that the proposed method outperforms random search as well as a reinforcement learning-based baseline.

We further integrate GP-MoLFormer-Sim with a genetic algorithm-based search process, where GP-MoLFormer-Sim enables generating offspring of the high-reward samples (see Figure 1). We refer to this approach as GP-MoLFormer-Sim+GA. Results on the popular Practical Molecular Optimization (PMO) benchmark (Gao et al. 2022) show that GP-MoLFormer-Sim+GA yields better performance on 23 molecular optimization tasks when the oracle is a black-box, compared to the current state-of-the-art GA-based training-free baselines including the ones that call large language models like GPT-4 for proposing high-reward samples. To our knowledge, this is the first demonstration of using test-time update of a CLM-based deep generative policy for for molecular optimization.

*These authors contributed equally.

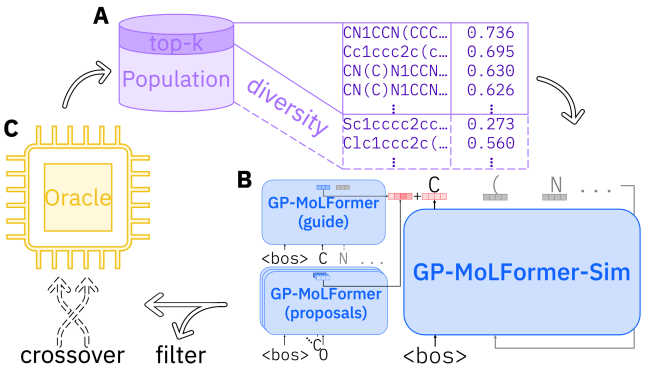


Figure 1: Overview of the GP-MOLFORMER-SIM+GA process. (A) The top-k highest scoring molecules so far are chosen as guides with additional diverse candidates, if desired. (B) Using GP-MOLFORMER-SIM, generate new candidates conditioned on closeness to top guides. GP-MOLFORMER-SIM adjusts the logits of the base model at every iteration using embedding similarity of the guide sequence so far to each proposed next token. (C) Prune (filter) or augment (with graph-based crossover operation) generations and score them with the oracle. These new samples are added back to the population and the process is repeated until the oracle budget is met.

## 2 Guided Generation

### 2.1 Background information — GP-MOLFORMER

GP-MOLFORMER is a chemical language foundation model, which is a GPT-style autoregressive decoder trained with linear attention and rotary embeddings (Ross et al. 2025)[1]. The model used in here is trained on ∼650M canonicalized SMILES obtained from ZINC and PubChem databases. Unconditional sampling from this chemical language model would allow exploring the chemical space. For details of the GP-MOLFORMER model and its performance on unconditional SMILES generation task, see Ross et al. (2025).

### 2.2 Target-guided generation with GP-MOLFORMER — GP-MOLFORMER-SIM

Guiding the autoregressive sampling from CLMs like GP-MOLFORMER towards specific molecules is of a paramount interest, as it enables generating new variations given target molecules of importance. Specifically, given a single molecule we are interested in exploring the molecular neighborhood where the similarity is defined through the cosine in the embedding space of the same generative model (GP-MOLFORMER).

More formally, we wish to generate a new sequence $s$ with guidance from molecules $m_j, j = 1 \dots N$ that are canonical SMILES sequences. We build the sequence $s$ incrementally by gradually sampling tokens from a new policy that mixes

[1]https://huggingface.co/ibm-research/GP-MoLFormer-Uniq

**Algorithm 1:** Guided Generation with kernel approximation

**Require:** GETEMBEDDINGGPT() , GETLOGITS()
1: **Inputs:** $\alpha$ (mixing parameter), $\tau$ (softmax temperature), $T$ (RFF temperature)
2: $s \leftarrow$ BOS, $t \leftarrow 1$
3: **while** EOS is not met **do**
4: **Append and Embed Generated**
5: **for** $i \in$ Vocab **do**
6: $x_i = \text{GETEMBEDDINGGPT}(s \oplus i)$
7: $x_i \leftarrow RandomFea(x_i, T)$ // Optional
8: $x_i \leftarrow \frac{x_i}{||x_i||}$
9: **end for**
10: **Embed targeted molecule up to time** $t$
11: **for** $j \in$ targetMolecules **do**
12: $y_j = \text{GETEMBEDDINGGPT}(m_j[1:t])$
13: $y_j \leftarrow RandomFea(y_j, T)$ // Optional
14: $y_j \leftarrow \frac{y_j}{||y_j||}$
15: **end for**
16: **Compute pairwise cosine** (**Vocabsize** $\times N$ **where** $N =$ **number of targetMolecules**)
17: $S_{ij} = \langle x_i, y_j \rangle, i = 1 \dots$ Vocabsize, $j = 1 \dots N$
18: $\bar{S}_i = \frac{1}{N} \sum_{j=1}^{N} S_{ij}$, for $i = 1 \dots$ Vocabsize
19: **Tilting the logits**
20: $u \leftarrow \text{GETLOGITS}(s)$
21: Standardize $u$ and $\bar{S}$
22: $u \leftarrow \frac{1}{\tau}((1-\alpha)u + \alpha\bar{S})$
23: Sample token $d$ with $P \propto \text{SOFTMAX}(u)$
24: $s \leftarrow s \oplus d, t \leftarrow t + 1$
25: **end while**
26: **return** $s$

the likelihood under GP-MOLFORMER (logit $u$) and the *contextual* similarity of the new sequence to target molecules in the embedding space of GP-MOLFORMER, $\bar{S}$. Algorithm 1 summarizes this procedure. The logits of the new guiding policy are $\frac{1}{\tau}((1-\alpha)u + \alpha\bar{S})$, where $\alpha \in [0, 1]$ controls the mixing strength, interpolating between unconditional sampling from GP-MOLFORMER and pure similarity based sampling, and $\tau$ is a sampling temperature that allows control over the entropy of the sampling. The sequence is generated iteratively until the <eos> token is selected.

Note that the cosine similarity is used to "tilt" the logits of the GP-MOLFORMER using similarity to a neighborhood formed by target molecules in the embedding space. We can push this idea further using a kernel density estimator (KDE) with a gaussian kernel. The temperature $T$ of the kernel induces further locality control. We can approximate the KDE using Random Fourier Features (Rahimi and Recht 2007) (lines 7 and 13 in Algorithm 1).

Because our proposed algorithm is a test-time guidance algorithm, it does not need any training procedure and enjoys multiple advantages of scalability, parallelism, efficiency, and versatility in its applicability to multiple domains. Training-free methods play a role in applications that require instantaneous response. Furthermore they avoid issues associated with supervised finetuning methods, including catastrophic forgetting and the general parametric complexity of

weights updates. The complexity of our algorithm is linear in the vocabulary size (2362), the dimensionality of GP-MOLFORMER embedding (768) and the number of target molecules. For experiments with random features, we also used 768 random features. As our method is training-free, we do not require any computational resources or timing observations for training. A single A100 GPU is used for each inference task (we also note that GPU memory is not a concern as our memory footprint only occupied no more than 8 GB of VRAM). Even though our guided method has many more moving parts and occupies much more memory than the base model unconditional generation, the extra computation cost of the guided method is only roughly four times slower than the unconditional generation. As an example, generating a single token from the unconditional model takes, on average, $0.013$ seconds while generating a guided token takes $0.049$ seconds. When generating a batch of 20 molecules, the runtime is $1.02$ seconds (producing avg. molecule length of 43 tokens) for unconditional and $3.97$ seconds for guided generation (of molecules with 40 tokens on average).

In Figure 2, we showcase a depiction of the Algorithm 1 in action on a guided generation task (for details of the task, see Section 4), which aims to generate molecules within the individual neighborhood of five trypsin inhibitor targets (large circles) with varying guiding strength $\alpha$ and sampling temperature $\tau$ settings. In blue, we see the unconditional generation from GP-MOLFORMER. When comparing the first two panels for the same guidance strength $\alpha$, we see the effect of sampling temperature $\tau$: the higher temperature ($\tau = 0.40$) leads to a higher entropy resulting in a larger spread around the target molecules. On the other hand, comparing the outer two panels, for fixed sampling temperature $\tau = 0.20$, we observe that larger mixing $\alpha = 0.5$ leads to tighter clustering around the target molecules, away from the unconditional baseline in blue. It should be mentioned that the algorithm is not specific to using a single guide, and can be extended to guiding the generation to multiple target molecules simultaneously. Additional visualizations involving a guidance by multiple targets simultaneously can be found in Figure 5 in Navratil et al. (2025).

**Theoretical Justification** It can be shown that the computations on lines 22–23 in Algorithm 1 represent a solution of a closed-form optimization problem involving token similarity to the target molecule distribution and a KL divergence from the smoothed GP-MOLFORMER distribution. Full detail is laid out in Section A in Navratil et al. (2025).

### 2.3 GP-MOLFORMER-SIM augmented with genetic algorithm — GP-MOLFORMER-SIM+GA

Typical GA combines *mutation* and *crossover* steps to augment a current candidate pool to aid exploration, followed by sampling the fittest (highest-scoring) compounds (as per the black-box oracle function) to form the next generation in a cyclical process. In our work, we adopt the cyclical nature of a GA and combine it with the ability of the GP-MOLFORMER-SIM method to produce novel molecules with high efficiency that are close to targets already known to have a desirable property. The process is illustrated in Figure 1 and captured in Algorithm 2. In every generation, we maintain a set of compounds with known property of interest — the oracle value. The best $G$ candidates are selected to serve as guides for GP-MOLFORMER-SIM (going from A→B in Figure 1). The selection process takes into account high oracle scores as well as diversity. For each of the $K$ guides, the GP-MOLFORMER-SIM module (Fig. 1 B) generates a set of novel candidates forming new *mutated* offspring. In order to reduce oracle budget expenditure, a pruning step ("filter" in Figure 1 B→C) is applied to reduce the offspring set size by removing candidates that are below a certain threshold of Tanimoto similarity ($T_{sim}$) measured from the current guide set (details are given in Section C in Navratil et al. (2025)). Optionally, a graph-action based crossover operation (Jensen 2019) is also applied to create offspring from best guides. The offspring set is then sent to oracle for scoring and is merged to the compound pool (Fig. 1 C→A), thus closing the GA cycle. During the process, the average of top-10 scoring compounds are recorded along with oracle budget expenditure. A sample optimization trajectory visualized in a 2D t-SNE chart can be found in Figure 3 in Navratil et al. (2025).

Algorithm 2: GP-MoLFormer-Sim+GA

**Require:** Oracle $F : \mathbb{M} \to \mathbb{R}$, GP-MOLFORMER-SIM()

1: Inputs: $G, K, B, D$
2: generation $\leftarrow 0$, budget $\leftarrow 0$
3: Initialize $P \sim$ ZINC
4: Store scores $R[P] \leftarrow F(P)$
5: Record (budget spent, generation, avg. $K$ best scores)
6: **while** Oracle budget $\leq$ B **do**
7: Sample $S \subset P$ // Select $G$ best candidates as guides
8: Optional: $S \leftarrow S \cup (S' \subset P)$ // Augment by $D$ diverse candidates
9: Generate $N \leftarrow$ GP-MOLFORMER-SIM$(s) \quad \forall s \in S$ // Use GP-MoLFormer-Sim to create neighbors (mutations) of guides
10: Select $P' \subset P \cup N$ // e.g, best by $T_{sim}$ to current top in $P$
11: Optional: augment $P'$ // e.g., via "graph-based" crossover of current best guides
12: generation $\leftarrow$ generation + 1
13: Store $R[P' \setminus P] \leftarrow F(P' \setminus P)$
14: Set $P \leftarrow P'$
15: Record (budget spent, generation, avg. $K$ best scores)
16: **end while**
17: **return** Array of tuples (generation, budget spent, avg. top-$K$ score)

## 3 Related Work

**Molecule optimization** The goal of molecule optimization is to iteratively modify molecule structures to improve desired properties like binding affinity, solubility, drug likeliness, etc. The ability to represent molecules using text-based encodings like SMILES (Weininger 1988) and SELFIES (Krenn et al. 2020), enables the application of natural language pro-

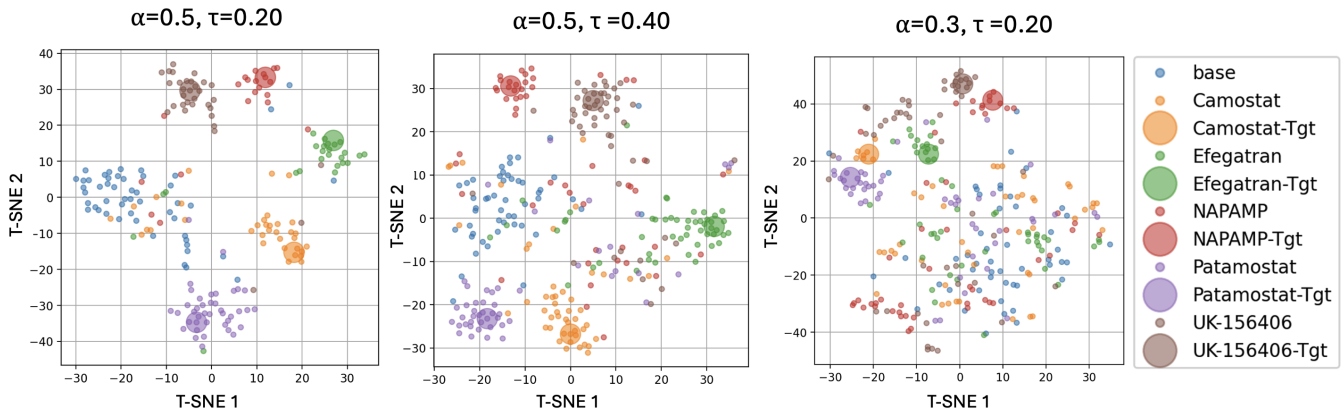


Figure 2: Guided generations (dots) around five trypsin inhibitor targets (large circles) with various guiding strength $\alpha$ and sampling temperature $\tau$ settings. "Base" molecules from un-guided generation. Green, orange, red, purple and brown correspond to five different targets. Molecules are visualized in GP-MOLFORMER embeddings space projected onto two t-SNE dimensions.

cessing techniques to tackle this problem. Existing methods have used techniques such as reinforcement learning (RL) (Olivecrona et al. 2017; Blaschke et al. 2018; Loeffler et al. 2024; Segler et al. 2017; Neil et al. 2019; Zhou et al. 2019), variational autoencoders (Gómez-Bombarelli et al. 2018; Jin, Barzilay, and Jaakkola 2018), Bayesian optimization (Moss et al. 2020; Tripp, Simm, and Hernández-Lobato 2021), GFlowNets (Shen, Pandey, and Ester 2023; Bengio et al. 2021, 2023), genetic algorithms (Lee et al. 2023), query-based optimization (Hoffman et al. 2022), and diffusion models (Lee, Jo, and Hwang 2023). Recently, large language model-based methods (Ye et al. 2025; Wang et al. 2024) have appeared as a promising method for molecule design, when used in combination with other methods like genetic algorithms (Wang et al. 2024).

**Genetic algorithms for molecular optimization** Genetic algorithms have emerged as a state-of-the-art method for molecule optimization tasks (Gao et al. 2022; Tripp and Hernández-Lobato 2023). They work by mimicking an iterative evolutionary process using operations like mutation and crossover on the molecule representations and allowing favorable candidates to survive to the next generation. Some examples include STONED (Nigam et al. 2021) which operates on SELFIES representations, GEAM (Lee, Jo, and Hwang 2023) which operates on molecule fragments, Graph-GA (Jensen 2019) which applies graph-based mutation/crossover operators for GA, Mol-GA (Tripp and Hernández-Lobato 2023) which incorporates quantile uniform sampling to maintain diversity while rewarding the best candidates, MOLLEO (Wang et al. 2024) which uses Graph-GA in combination with a large language model, genetic guided GFlowNets (Kim et al. 2024), and SynNet which incorporates synthesis constraints (Gao, Mercado, and Coley 2021). The present work differs from those earlier ones, as it combines a test-time guided generation using a small chemical language model with GA for optimization.

**Test-time steering of autoregressive language models** Recently, several approaches have been proposed to steer the output of language models to desired outputs without retraining the entire model. Deng and Raffel (2023) proposed Reward-guided Decoding, which uses a reward model to score generations as they are produced and rescales sampling probabilities to favor high-reward tokens. Another approach is Self-disciplined Autoregressive Sampling (SASA) (Ko et al. 2024), which uses the contextual representations learned from the LLM itself to guide it to generate non-toxic text. Lee et al. (2024) uses conditional activation steering to selectively apply or withhold activation steering using LLM activation patterns. While GP-MOLFORMER-SIM also relies on test-time steering of a (chemical) language model, different from the prior works it does not involve training of an external or an internal reward model (including the need for explicit labels) to be used as guidance during decoding, nor does it require analyzing activation patterns of the decoder. Rather, the proposed method exploits contextual similarity with the target at each step of decoding and updates the logits accordingly.

# 4 Experiments

## 4.1 Similarity-guided molecule generation

This task involves generating chemical SMILES similar to a query molecule. We consider five trypsin inhibitors from Hartenfeller et al. (2012) as the targets. The baselines considered are random sampling from a 50k pool of unconditionally generated molecules using GP-MOLFORMER, a random search in the reaction template and reactant space until a termination condition is met (Gottipati et al. 2020), and a RL-tuned graph isomorphism network (GIN) model that rewards molecules of high similarity to the target (Gupta et al. 2024). We report mean Tanimoto similarity ($T_{sim}$), estimated using Morgan fingerprints with a radius of 2, as well as min, max, and mean drug-likeness (QED) over 5 generated sets (one per target) containing the top-k most similar molecules ($k = 1, 10, 10^2, 10^3$ and $10^4$).

Table 1 reports the mean similarity ($T_{sim}$) of top-k most similar generations obtained using GP-MOLFORMER-SIM (GPMFS) and baseline methods. Similarity values of the

| config | top-k | $T_{sim}$ mean | QED min | QED mean | QED max | config | top-k | $T_{sim}$ mean | QED min | QED mean | QED max |
|---|---|---|---|---|---|---|---|---|---|---|---|
| GPMFS (Ours) | $10^0$ | 1.000 | 0.225 | 0.225 | 0.225 | S Model (Gupta et al. 2024) | $10^0$ | 0.694 | 0.289 | 0.289 | 0.289 |
| | $10^1$ | 0.972 | 0.158 | 0.238 | 0.357 | | $10^1$ | 0.618 | 0.12 | 0.201 | 0.289 |
| | $10^2$ | 0.877 | 0.075 | 0.241 | 0.555 | | $10^2$ | 0.554 | 0.049 | 0.206 | 0.712 |
| | $10^3$ | 0.763 | 0.037 | 0.240 | 0.738 | | $10^3$ | 0.499 | 0.024 | 0.262 | 0.923 |
| | $10^4$ | 0.573 | 0.016 | 0.213 | 0.901 | | $10^4$ | 0.439 | 0.013 | 0.302 | 0.946 |
| Random Generations | $10^0$ | 0.438 | 0.555 | 0.555 | 0.555 | Random Search (Gupta et al. 2024) | $10^0$ | 0.477 | 0.483 | 0.483 | 0.483 |
| | $10^1$ | 0.391 | 0.141 | 0.491 | 0.722 | | $10^1$ | 0.450 | 0.353 | 0.448 | 0.559 |
| | $10^2$ | 0.348 | 0.048 | 0.520 | 0.866 | | $10^2$ | 0.417 | 0.109 | 0.418 | 0.841 |
| | $10^3$ | 0.290 | 0.019 | 0.550 | 0.943 | | $10^3$ | 0.377 | 0.041 | 0.385 | 0.841 |
| | $10^4$ | 0.225 | 0.011 | 0.571 | 0.947 | | $10^4$ | 0.333 | 0.022 | 0.316 | 0.929 |

Table 1: Average similarity and QED values of the five trypsin inhibitor targeted generations, generated by various methods.

top-ranked generations show that target similarity-guided decoding using GPMFS performs better than test-time baselines like random sampling and random search. The proposed method also outperforms a graph generative model that is RL-tuned to optimize the target similarity (S model) across all values of k up to $10^4$. The reported min, mean, and max QED values show the inverse relation between QED and similarity for these targets, given the mean QED value of these five targets is only 0.234. Nevertheless, 132 molecules are found to have a QED value $> 0.7$ in the top-10000 most similar molecules generated by GPMFS. Those show a mean similarity of 0.47 and a max similarity of 0.67, showcasing the potential of the proposed method to guide generations on-the-fly toward a target molecule, while yielding useful molecules. A small sample of molecules generated by the GPMFS and their respective targets are visualized in Figure 2 for varying parameter settings (final parameter values can be found in Section C in Navratil et al. (2025)).

## 4.2 Sample-efficient molecular optimization — PMO benchmark

The open-source benchmark for practical molecular optimization, PMO (Gao et al. 2022), has served as an enabler for the transparent and robust evaluation of diverse sets of molecular optimization algorithms. It involves 23 single-objective optimization tasks, that includes property optimization, molecular rediscovery, and structure-based drug design, with a specific focus on the sample efficiency. PMO includes comparing optimization algorithms involving reinforcement learning, Bayesian optimization, generative models, GFlowNets, and genetic algorithms. We compare GP-MOLFORMER-SIM+GA (GPMFS+GA) with existing GA-based molecular optimization methods on this benchmark, while focusing on sample efficiency. Following Gao et al. (2022), we measure the performance by the area under the curve (AUC) of the average property scores of the top-10 molecules versus oracle calls, with the number of maximum oracle calls being 10k. We utilize the task-specific oracles implemented in the Therapeutics Data Commons (TDC) library (Huang et al. 2021). Average and standard deviation of scores obtained from five independent runs starting from different random seeds are reported, unless stated otherwise.

Table 2 reports the performance of GPMFS+GA on the 23 tasks from the PMO benchmark. Since the proposed method relies on a combination of the test-time steering of the deep generative model and a modified genetic algorithm we show, in the main article, a comparison of the proposed method with GA-based baselines specifically designed for molecular design that do not require any training of the generative model. The baselines shown in Table 2 are Graph GA (Jensen 2019), STONED (Nigam et al. 2021), SynNet (Gao, Mercado, and Coley 2021), Mol-GA (Tripp and Hernández-Lobato 2023), and MOLLEO (Wang et al. 2024). We report the rank per task based on the top-10 AUC score obtained with a maximum of 10k oracle calls for each method. Average rank and average score over all tasks are also reported in Table 2. Additional comparison to more than 30 baselines is given in Table 8–15 in Navratil et al. (2025). To gauge synthesizability, average Synthetic Accessibility scores of the top 100 generated molecules in each task are reported in Table 16 in Navratil et al. (2025). Results in Table 2 show that the proposed method scores second among GA baselines, while MOLLEO that uses Graph-GA with GPT-4 comes first. On three tasks, namely GSK3, JNK3, and ranolazine_mpo, GPMFS+GA outperforms all baselines, while on another 9 tasks it ranks second. Given the computational and actual dollar cost associated with calling GPT-4, GPMFS+GA appears as a more cost-effective alternative.

We also compare GPMFS+GA with other MOLLEO variants that use a smaller domain-aware language model — namely, a BioT5 model and a MoleculeSTM model (see Table 15 in Navratil et al. (2025)). GPMFS+GA performs better than MOLLEO (MoleculeSTM), while trailing behind the BioT5 variant.

## 4.3 Comparison with MOLLEO in the black-box oracle setting

MOLLEO includes natural language prompting of the LLM to generate proposals based on the GA operations — crossover and mutation. While doing so, the prompt includes information about the task but also reveals the actual target of the oracle function. The latter contradicts the original intent

| Task | Our Rank | GPMFS+GA (Ours) | Graph-GA | STONED SELFIES | SynNet Synthesis | MOL-GA | MOLLEO (GPT-4) |
|---|---|---|---|---|---|---|---|
| albuterol_similarity | 4 | 0.824 (.071) | 0.838 (.016) | 0.745 (.076) | 0.584 (.039) | 0.896 (.035) | 0.985 (.024) |
| amlodipine_mpo | 3 | 0.680 (.064) | 0.661 (.020) | 0.608 (.046) | 0.565 (.007) | 0.688 (.039) | 0.773 (.037) |
| celecoxib_rediscovery | 2 | 0.716 (.067) | 0.630 (.097) | 0.382 (.041) | 0.441 (.027) | 0.567 (.083) | 0.864 (.034) |
| deco_hop | 2 | 0.710 (.058) | 0.619 (.004) | 0.611 (.008) | 0.613 (.009) | 0.649 (.025) | 0.942 (.013) |
| DRD2 | 4 | 0.956 (.010) | 0.964 (.012) | 0.913 (.020) | 0.969 (.004) | 0.936 (.016) | 0.968 (.012) |
| fexofenadine_mpo | 3 | 0.798 (.028) | 0.760 (.011) | 0.797 (.016) | 0.761 (.015) | 0.825 (.019) | 0.847 (.018) |
| GSK3 | 1 | 0.896 (.035) | 0.788 (.070) | 0.668 (.049) | 0.789 (.032) | 0.843 (.039) | 0.863 (.047) |
| isomers_c7h8n2o2 | 2 | 0.932 (.011) | 0.862 (.065) | 0.899 (.011) | 0.455 (.031) | 0.878 (.026) | 0.984 (.008) |
| isomers_c9h10n2o2pf2cl | 3 | 0.864 (.016) | 0.719 (.047) | 0.805 (.031) | 0.241 (.064) | 0.865 (.012) | 0.874 (.053) |
| JNK3 | 1 | 0.806 (.087) | 0.553 (.136) | 0.523 (.092) | 0.630 (.034) | 0.702 (.123) | 0.790 (.027) |
| median1 | 2 | 0.340 (.034) | 0.294 (.021) | 0.266 (.016) | 0.218 (.008) | 0.257 (.009) | 0.352 (.024) |
| median2 | 4 | 0.255 (.031) | 0.273 (.009) | 0.245 (.032) | 0.235 (.006) | 0.301 (.021) | 0.275 (.045) |
| mestranol_similarity | 2 | 0.658 (.118) | 0.579 (.022) | 0.609 (.101) | 0.399 (.021) | 0.591 (.053) | 0.972 (.009) |
| osimertinib_mpo | 5 | 0.819 (.004) | 0.831 (.005) | 0.822 (.012) | 0.796 (.003) | 0.844 (.015) | 0.835 (.024) |
| perindopril_mpo | 2 | 0.584 (.042) | 0.538 (.009) | 0.488 (.011) | 0.557 (.011) | 0.547 (.022) | 0.600 (.031) |
| QED | 6 | 0.940 (.001) | 0.940 (.000) | 0.941 (.000) | 0.941 (.000) | 0.941 (.001) | 0.948 (.000) |
| ranolazine_mpo | 1 | 0.812 (.024) | 0.728 (.012) | 0.765 (.029) | 0.741 (.010) | 0.804 (.011) | 0.769 (.022) |
| scaffold_hop | 2 | 0.531 (.016) | 0.517 (.007) | 0.521 (.034) | 0.502 (.012) | 0.527 (.025) | 0.971 (.004) |
| sitagliptin_mpo | 3 | 0.501 (.081) | 0.433 (.075) | 0.393 (.083) | 0.025 (.014) | 0.582 (.040) | 0.584 (.067) |
| thiothixene_rediscovery | 3 | 0.504 (.033) | 0.479 (.025) | 0.367 (.027) | 0.401 (.019) | 0.519 (.041) | 0.727 (.052) |
| troglitazone_rediscovery | 2 | 0.437 (.067) | 0.390 (.016) | 0.320 (.018) | 0.283 (.008) | 0.427 (.031) | 0.562 (.019) |
| valsartan_smarts | 2 | 0.158 (.317) | 0.000 (.000) | 0.000 (.000) | 0.000 (.000) | 0.000 (.000) | 0.867 (.092) |
| zaleplon_mpo | 3 | 0.504 (.022) | 0.346 (.032) | 0.325 (.027) | 0.341 (.011) | 0.519 (.029) | 0.510 (.031) |
| Average | 2.7 | 0.662 (.221) | 0.597 (.233) | 0.566 (.245) | 0.499 (.259) | 0.639 (.236) | 0.777 (.200) |
| Rank by avg. score | – | 2 | 4 | 5 | 6 | 3 | 1 |

Table 2: Comparison of the guided generation GP-MOLFORMER-SIM+GA ("GPMFS+GA") to selected training-free GA-based baselines. Values are the mean (± standard deviation) AUC top-10 over 5 runs for each task. The baselines for Graph-GA (Jensen 2019), STONED SELFIES (Nigam et al. 2021), and SynNet (Gao, Mercado, and Coley 2021) are taken from Gao et al. (2022). MOL-GA values are taken from Tripp and Hernández-Lobato (2023) and MOLLEO values are taken from Wang et al. (2024). An additional comparison to a set of more than 30 baselines is given in Tables 8–15 in Navratil et al. (2025).

| Task | black-box | | |
|---|---|---|---|
| | GPMFS+GA (ours) | MOLLEO (redacted) | MOLLEO (original) |
| thiothixene | 0.504 (.033) | 0.462 (.031) | 0.692 (.013) |
| mestranol | 0.658 (.118) | 0.644 (.065) | 0.983 (.001) |

Table 3: Comparison of GP-MOLFORMER-SIM+GA with MOLLEO in the black-box oracle setting. Molecule name in prompt is redacted for MOLLEO in that setting. Mean (± standard deviation) AUC top-10 over 5 runs for each.

behind the Oracle-based PMO benchmarking where the Oracle serves as a proxy for an unknown target in a real-world application. For example, for the thiothixene rediscovery task, the prompt used in Wang et al. (2024) includes the following (*emphasis* ours): "OBJECTIVE: has a higher *thiothixene* rediscovery score. TASK: *thiothixene* rediscovery scores. OBJECTIVE_DEFINITION: The *thiothixene* rediscovery score measures a molecule's Tanimoto similarity with *thiothixene's* SMILES to check whether it could be rediscovered." Such a prompt breaks the black-box nature of the oracle as the GPT-4 model has memorized and can recall the thiothixene SMILES string perfectly, which is otherwise never disclosed to the other baselines. To illustrate this problem, we revise the prompt such that the name of the target molecule is redacted from the prompt. Results are reported in Table 3 for two exemplar tasks, namely thiothixene rediscovery and mestranol similarity. On both tasks, MOLLEO's performance (using a `gpt-4.1-mini` model) drops by ∼33% and becomes worse compared to GPMFS+GA when the molecule name is redacted. This result implies that MOLLEO's performance depends on the LLM's utilization of the task-relevant contextual information for proposing offspring. In contrast, our method operates in the black-box oracle mode, consistent with the philosophy of the PMO benchmark, to produce candidates during optimization and, as shown in Table 3 outperforms MOLLEO in that mode.

We also run an experiment where GPMFS+GA has access to the target SMILES information used in the oracle function (when applicable) and utilizes that to create the initial pool of candidates for optimization. Table 4 shows the performance gain achieved by the proposed method in that mode, again underscoring the inflationary effect of breaking the black-box nature of the oracle on the performance.

## 4.4 Ablation experiments

Algorithm 2 in Section 2.3 provides for several optional steps, namely: (1) using random Fourier features (RFF) to approximate a kernel distance in the GP-MOLFORMER embedding

| Task | Our Rank | GPMFS+GA (Ours) | MOLLEO (MolSTM) | MOLLEO (BioT5) | MOLLEO(Wang et al. 2024) (GPT-4) |
|---|---|---|---|---|---|
| albuterol | 1 | 0.994 (.005) | 0.929 (.005) | 0.968 (.003) | 0.985 (.024) |
| amlodipine | 3 | 0.719 (.061) | 0.674 (.018) | 0.776 (.038) | 0.773 (.037) |
| celecoxib | 1 | 0.885 (.010) | 0.594 (.105) | 0.508 (.017) | 0.864 (.034) |
| fexofenadine | 2 | 0.812 (.046) | 0.789 (.016) | 0.773 (.017) | 0.847 (.018) |
| median1 | 1 | 0.405 (.001) | 0.298 (.019) | 0.338 (.033) | 0.352 (.024) |
| median2 | 1 | 0.393 (.004) | 0.251 (.031) | 0.259 (.019) | 0.275 (.045) |
| mestranol | 1 | 0.993 (.011) | 0.596 (.018) | 0.717 (.104) | 0.972 (.009) |
| osimertinib | 3 | 0.818 (.002) | 0.823 (.007) | 0.817 (.016) | 0.835 (.024) |
| perindopril | 3 | 0.582 (.018) | 0.554 (.037) | 0.738 (.016) | 0.600 (.031) |
| ranolazine | 1 | 0.825 (.016) | 0.725 (.040) | 0.749 (.012) | 0.769 (.022) |
| sitagliptin | 4 | 0.435 (.054) | 0.548 (.065) | 0.506 (.100) | 0.584 (.067) |
| thiothixene | 1 | 0.862 (.020) | 0.508 (.035) | 0.696 (.081) | 0.727 (.052) |
| troglitazone | 1 | 0.905 (.002) | 0.381 (.025) | 0.390 (.044) | 0.562 (.019) |
| zaleplon | 1 | 0.684 (.024) | 0.475 (.018) | 0.465 (.026) | 0.510 (.031) |
| Average | 1.7 | 0.737 (.201) | 0.582 (.189) | 0.621 (.201) | 0.690 (.209) |
| Rank by avg. score | – | 1 | 4 | 3 | 2 |

Table 4: Top-10 AUC metrics for GP-MOLFORMER-SIM+GA ("GPMFS+GA") under access to the oracle's target SMILES in comparison to MOLLEO models. Mean (± standard deviation) AUC top-10 over 5 runs for each.

| Config | Avg. rank ↓ | Avg. score ↑ |
|---|---|---|
| GG | 5.0 | 0.603 |
| +XO | 3.4 | 0.672 |
| +RFF768 | 5.3 | 0.597 |
| +XO+DIV | 3.0 | 0.678 |
| +RFF768+XO | 2.5 | 0.682 |
| +RFF768+XO+DIV | 1.8 | 0.690 |

Table 5: Overall improvement due to specific method combinations in terms of average 1-based rank and top-10 AUC metric in GP-MOLFORMER-SIM. Includes adding "XO" as crossover, "RFF768" as 768-dimensional Random Fourier Features, and "DIV" as diversity augmentation to the vanilla Guided Generation (GG).

space, (2) using a genetic graph-based crossover (XO) between parent molecules (drawn from the set of best guides), and (3) adding a set of diverse guides (DIV) to enhance exploration. Results reported in Tables 2 and 4 were obtained employing all of these variants active. To tease apart individual effects of these options, we also ran a series of ablation experiments over the 23 PMO tasks. Table 5 summarizes the relative performance in multiple configurations, starting with guided generation (GG) with none of the three options to GG with the full set active. For each combination we report the average rank over all tasks as well as the average AUC metric. We observe each option adding a benefit, with the exception of the RFF, however, only when being added alone. The best configuration is `GG+RFF768+XO+DIV` which is used throughout our PMO experiments, unless otherwise stated. Detailed, per-task ablation results are given in Table 7 in Navratil et al. (2025).

## 5 Limitations

Given the need for discovering new and useful artifacts for various discovery applications, the proposed method can have broader impact beyond chemistry and biology. There remain open questions and limitations, however. For example, although the proposed framework is model and domain-agnostic, we have only experimented here with a specific chemical language model decoder and on specific molecular optimization tasks. It also remains an open question to what extent the method can benefit from including "negative" targets while subjected to an optimization task. Extending the method to multi-objective optimization (optimizing linear combination of multiple objectives with different importances) can also be a potential future research direction.

## 6 Conclusions

In this work, we present GP-MOLFORMER-SIM, a test-time approach for sequentially revising the generated output of a small chemical language model to maintain the contextual closeness between its generation and a given set of targets. Furthermore, we integrate the proposed method into a genetic algorithm as an effective proposer of mutations to produce high-quality offspring (GP-MOLFORMER-SIM+GA). Our method is validated on a variety of molecular optimization tasks. Evaluating this framework with a black-box oracle reveals performance improvement compared to a baseline that leverages large language model like GPT-4, demonstrating the effective trade-off between performance and computational efficiency of the proposed method. We believe the proposed guided generation method represents a versatile and valuable addition to the modeling toolbox in molecular optimization and beyond.

## References

Ahn, S.; Kim, J.; Lee, H.; and Shin, J. 2020. Guiding deep molecular optimization with genetic exploration. *Advances*

*in neural information processing systems*, 33: 12008–12021.

Bengio, E.; Jain, M.; Korablyov, M.; Precup, D.; and Bengio, Y. 2021. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34: 27381–27394.

Bengio, Y.; Lahlou, S.; Deleu, T.; Hu, E. J.; Tiwari, M.; and Bengio, E. 2023. Gflownet foundations. *Journal of Machine Learning Research*, 24(210): 1–55.

Blaschke, T.; Olivecrona, M.; Engkvist, O.; Bajorath, J.; and Chen, H. 2018. Application of generative autoencoder in de novo molecular design. *Molecular Informatics*, 37(1-2): 1700123.

Deng, H.; and Raffel, C. 2023. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. *arXiv preprint arXiv:2310.09520*.

Gao, W.; Fu, T.; Sun, J.; and Coley, C. W. 2022. Sample Efficiency Matters: A Benchmark for Practical Molecular Optimization. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Gao, W.; Mercado, R.; and Coley, C. W. 2021. Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design. *arXiv preprint arXiv:2110.06389*.

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. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2): 268–276.

Gottipati, S. K.; Sattarov, B.; Niu, S.; Pathak, Y.; Wei, H.; Liu, S.; Thomas, K. M. J.; Blackburn, S.; Coley, C. W.; Tang, J.; Chandar, S.; and Bengio, Y. 2020. Learning To Navigate The Synthetically Accessible Chemical Space Using Reinforcement Learning. arXiv:2004.12485.

Gupta, A.; Current, S.; Ravindran, B.; Batra, R.; Raman, K.; et al. 2024. A Similarity-Agnostic Reinforcement Learning Approach for Lead Optimization. *openreview*.

Hartenfeller, M.; Zettl, H.; Walter, M.; Rupp, M.; Reisen, F.; Proschak, E.; Weggen, S.; Stark, H.; and Schneider, G. 2012. DOGS: reaction-driven de novo design of bioactive compounds. *PLoS computational biology*, 8(2): e1002380.

Hoffman, S. C.; Chenthamarakshan, V.; Wadhawan, K.; Chen, P.-Y.; and Das, P. 2022. Optimizing molecules using efficient queries from property evaluations. *Nature Machine Intelligence*, 4(1): 21–31.

Huang, K.; Fu, T.; Gao, W.; Zhao, Y.; Roohani, Y.; Leskovec, J.; Coley, C. W.; Xiao, C.; Sun, J.; and Zitnik, M. 2021. Therapeutics Data Commons: Machine Learning Datasets and Tasks for Drug Discovery and Development. arXiv:2102.09548.

Jensen, J. H. 2019. A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space. *Chemical science*, 10(12): 3567–3572.

Jin, W.; Barzilay, R.; and Jaakkola, T. 2018. Junction tree variational autoencoder for molecular graph generation. *arXiv:1802.04364*.

Kim, H.; Kim, M.; Choi, S.; and Park, J. 2024. Genetic-guided GFlowNets for Sample Efficient Molecular Optimization. arXiv:2402.05961.

Ko, C.-Y.; Chen, P.-Y.; Das, P.; Mroueh, Y.; Dan, S.; Kollias, G.; Chaudhury, S.; Pedapati, T.; and Daniel, L. 2024. Large Language Models can be Strong Self-Detoxifiers. *arXiv preprint arXiv:2410.03818*.

Krenn, M.; Häse, F.; Nigam, A.; Friederich, P.; and Aspuru-Guzik, A. 2020. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4): 045024.

Lee, B. W.; Padhi, I.; Ramamurthy, K. N.; Miehling, E.; Dognin, P.; Nagireddy, M.; and Dhurandhar, A. 2024. Programming refusal with conditional activation steering. *arXiv preprint arXiv:2409.05907*.

Lee, S.; Jo, J.; and Hwang, S. J. 2023. Exploring chemical space with score-based out-of-distribution generation. In *International Conference on Machine Learning*, 18872–18892. PMLR.

Lee, S.; Lee, S.; Kawaguchi, K.; and Hwang, S. J. 2023. Drug discovery with dynamic goal-aware fragments. *arXiv preprint arXiv:2310.00841*.

Loeffler, H. H.; He, J.; Tibo, A.; Janet, J. P.; Voronov, A.; Mervin, L. H.; and Engkvist, O. 2024. Reinvent 4: Modern AI–driven generative molecule design. *Journal of Cheminformatics*, 16(1): 20.

Moss, H.; Leslie, D.; Beck, D.; Gonzalez, J.; and Rayson, P. 2020. Boss: Bayesian optimization over string spaces. *Advances in neural information processing systems*, 33: 15476–15486.

Navratil, J.; Ross, J.; Das, P.; Mroueh, Y.; Hoffman, S. C.; Chenthamarakshan, V.; and Belgodere, B. 2025. GP-MoLFormer-Sim: Test Time Molecular Optimization through Contextual Similarity Guidance. arXiv:2506.05628.

Neil, D.; Segler, M.; Guasch, L.; Ahmed, M.; Plumbley, D.; Sellwood, M.; and Brown, N. 2019. Exploring deep recurrent models with reinforcement learning for molecule design. In *ICLR*.

Nigam, A.; Pollice, R.; Krenn, M.; Gomes, G. d. P.; and Aspuru-Guzik, A. 2021. Beyond generative models: superfast traversal, optimization, novelty, exploration and discovery (STONED) algorithm for molecules using SELFIES. *Chem. Sci.*, 12(20): 7079–7090.

Olivecrona, M.; Blaschke, T.; Engkvist, O.; and Chen, H. 2017. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9(1): 48.

Rahimi, A.; and Recht, B. 2007. Random Features for Large-Scale Kernel Machines. In Platt, J.; Koller, D.; Singer, Y.; and Roweis, S., eds., *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

Ross, J.; Belgodere, B.; Hoffman, S. C.; Chenthamarakshan, V.; Navratil, J.; Mroueh, Y.; and Das, P. 2025. GP-MoLFormer: A Foundation Model For Molecular Generation. arXiv:2405.04912.

Segler, M. H.; Kogej, T.; Tyrchan, C.; and Waller, M. P. 2017. Generating focused molecule libraries for drug discovery

with recurrent neural networks. *ACS Central Science*, 4(1): 120–131.

Shen, T.; Pandey, M.; and Ester, M. 2023. Tacogfn: Target conditioned gflownet for drug design. In *NeurIPS 2023 Generative AI and Biology (GenBio) Workshop*.

Tripp, A.; and Hernández-Lobato, J. M. 2023. Genetic algorithms are strong baselines for molecule generation. arXiv:2310.09267.

Tripp, A.; Simm, G. N.; and Hernández-Lobato, J. M. 2021. A fresh look at de novo molecular design benchmarks. In *NeurIPS 2021 AI for Science Workshop*.

Wang, H.; Skreta, M.; Ser, C.-T.; Gao, W.; Kong, L.; Strieth-Kalthoff, F.; Duan, C.; Zhuang, Y.; Yu, Y.; Zhu, Y.; et al. 2024. Efficient evolutionary search over chemical space with large language models. *arXiv preprint arXiv:2406.16976*.

Weininger, D. 1988. 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.

Ye, G.; Cai, X.; Lai, H.; Wang, X.; Huang, J.; Wang, L.; Liu, W.; and Zeng, X. 2025. Drugassist: A large language model for molecule optimization. *Briefings in Bioinformatics*, 26(1): bbae693.

Zhou, Z.; Kearnes, S.; Li, L.; Zare, R. N.; and Riley, P. 2019. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9(1): 10752.