# LEARNING TO EVOLVE: SCALING OPEN-ENDED DISCOVERY WITH RELATIVE-PROGRESS RL

**Xuan Li**[1] **Zhanke Zhou**[1] **Zongze Li**[1] **Jiangchao Yao**[2] **Bo Han**[1*]

[1]TMLR Group, Hong Kong Baptist University [2]CMIC, Shanghai Jiao Tong University

## ABSTRACT

Evolution is a promising way for Large Language Models (LLMs) to tackle open-ended problems, such as molecular optimization. Existing training-free methods of evolution rely on context engineering that cannot reliably yield desired solutions. On the other hand, Reinforcement Learning with Verifiable Rewards (RLVR) is a learning-centric alternative, but it prioritizes final solutions over the multi-turn process of evolution, which cannot bring stable improvement. To address this, we propose Learning to Evolve (LtE), which learns a policy for iterative refinement by turning per-turn evaluator scores into turn-wise and trajectory-wise credit assignments. LtE uses (i) a turn-level advantage based on the score improvement over the initial solution and (ii) a trajectory-level advantage that accumulates these improvements over the entire trajectory. These two rewards are combined for credit assignment across turns and across trajectories, aligning the learning with progress improvement across evolution turns. We conduct experiments on molecular optimization tasks. LtE produces higher-quality solutions with the same budgets as training-free and RLVR methods and enables test time scale-up.

## 1 INTRODUCTION

Large language models (LLMs) are adopting evolution to solve open-ended problems (Novikov et al., 2025; Gottweis et al., 2025), where the model reflects and revises solutions over multiple turns to improve a verifiable objective measuring solution quality. This approach leverages LLMs' internal knowledge, rather than the random mutations of conventional approaches, to effectively discover promising solutions in problems like algorithm synthesis and drug discovery (Gao et al., 2025).

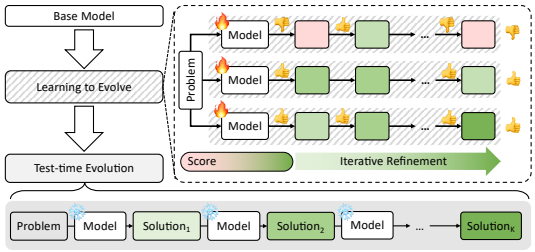


Figure 1: Evolution with an evaluator for scoring. The model generates refinements; improvements are reinforced, while regressions are discouraged.

Despite this promise, most current frameworks are training-free, relying on test-time search with fixed parameters. This is often sample-inefficient: refinements may not improve the candidate, so gains depend on extensive sampling (Lange et al., 2025). The issue is sharper in specialized domains where valid solutions are rare, making training-free evolution costly in samples (Chen et al., 2025b).

A *learning-centric* alternative is RLVR, using evaluator scores to update the model. However, extending single-turn RLVR to multi-turn evolution is nontrivial (Kumar et al., 2025): later refinements depend on earlier steps, and different turns can contribute unequally. In practice, RLVR often reduces the whole rollout to a single final reward and applies it to every turn, which blurs credit assignment and makes it hard to tell which refinements drove progress (Guo et al., 2025b; Zhang et al., 2025d).

To address this, we propose *Learning to Evolve (LtE)*, as shown in Fig. 1. LtE uses evaluator scores to update the model, helping it adapt to the domain and improve in fewer turns. LtE runs a multi-turn

*Correspondence to Bo Han (bhanml@comp.hkbu.edu.hk).

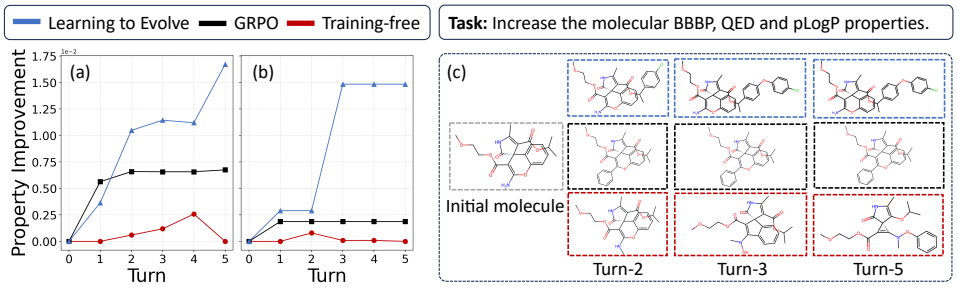


Figure 2: Comparison of different approaches on multi-property molecular optimization. (a) Mean property improvement over evolution turns, averaged across multiple problems. (b) Property improvement on the representative problem visualized in (c). (c) Example optimization trajectories showing the initial molecule and intermediate molecules at turns 2, 3, and 5: Learning to Evolve (blue; scaffold-preserving peripheral growth), GRPO (black; single-site hard replacement), and Training-free (red; single-site local editing); dashed boxes indicate the molecules produced at each turn. As the evolution horizon increases, LtE yields a larger improvement.

*refine-evaluate-update* loop: at each turn, the model proposes a refined solution, the evaluator scores it, and the score is used to guide the next update. The main challenge is turn-level credit assignment: early turns may get little signal before constraints are met, and change quickly once they are met.

LtE builds a stable learning signal by shaping feedback into *progress* and using relative improvement at two levels. Specifically, we define a turn-level reward as the validity-gated improvement in score over the initial solution and a trajectory-level return as the sum of turn-level rewards across turns. LtE then computes (i) an intra-trajectory advantage that ranks turns by their improvement relative to other turns in the same trajectory, and (ii) an inter-trajectory advantage that ranks full trajectories by their total progress. These advantages are combined to train the policy with token-wise updates that favor refinement steps and evolution runs that deliver sustained improvement.

We conduct empirical analysis of LtE in molecular optimization, chosen specifically for its complex, multi-turn dependencies. Because successful molecular design requires a sequence of strictly validated refinement, we adopt this task for assessing LtE's ability to master multi-turn evolution. We utilize benchmarks with single and multiple properties optimization, including TOMG-Bench (Li et al., 2024) and MuMoInstruct (Dey et al., 2025). Notably, LtE outperforms both training-free baselines and RLVR counterparts, attaining high relative improvement and strong success rates under comparable validity. Notably, LtE continues to improve as the evolution horizon increases, suggesting that learning turn-level credit assignment yields a policy that is capable of multi-turn refinement.

## 2 Preliminaries

### 2.1 Problem Formulation

We study evolution as iterative optimization guided by verifiable feedback. Let $\mathcal{X}$ denote the space of problems and $\mathcal{Y}$ the space of candidate solutions. We consider a dataset $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$, where each task is a pair $(x, y_0)$ consisting of a problem $x \in \mathcal{X}$ and an initial valid solution $y_0 \in \mathcal{Y}$.

**Evaluator.** We define an evaluator that returns (i) a validity indicator and (ii) a scalar quality score:

$$\mathcal{E} : \mathcal{X} \times \mathcal{Y} \to \{0,1\} \times \mathbb{R},\ \mathcal{E}(x,y) = \big(v(x,y),\ q(x,y)\big), \tag{1}$$

where $v(x,y) \in \{0,1\}$ indicates whether $y$ is valid for $x$. Here $q(x,y) \in \mathbb{R}$ is a task-specific quality function that is defined for all $y$ when $v(x,y) = 1$. We denote $v(x,y)$ and $q(x,y)$ as $v$ and $q$ when the context is clear.

**Objective.** Given an input $x$ and a valid initial solution $y_0$, the model generates a sequence of refined solutions $\{y_1, \ldots, y_K\}$. Let $\mathcal{S} = \{y_0, \ldots, y_K\}$ be the full solution pool. We define the valid subspace as $\mathcal{V}(x) = \{y \in \mathcal{Y} \mid v(x,y) = 1\}$. We select the optimal solution $\hat{y}$ by maximizing the quality score $q(x,y)$ over all valid candidates: $\hat{y} = \arg\max_{y \in \mathcal{S} \cap \mathcal{V}(x)} q(x,y)$. Since $y_0 \in \mathcal{S}$ and $y_0 \in \mathcal{V}(x)$, the feasible set is non-empty.

### 2.2 RLVR FOR MULTI-TURN EVOLUTION

For each task $(x, y_0) \sim \mathcal{D}$, we sample $G$ independent $K$-turn trajectories $\{\mathcal{T}^{(i)}\}_{i=1}^{G}$ using $\pi_{\theta_{\text{old}}}$, where trajectory $i$ produces $\{y_t^{(i)}\}_{t=1}^{K}$. RLVR commonly builds a single outcome reward per trajectory and converts it to an advantage $\tilde{A}^{(i)}$ that is shared across all turns:

$$\mathcal{J}(\theta) = \mathbb{E}_{\substack{(x,y_0)\sim\mathcal{D},\\ \{\mathcal{T}^{(i)}\}\sim\pi_{\theta_{\text{old}}}}} \left[ \frac{1}{G}\sum_{i=1}^{G} \frac{1}{K}\sum_{t=1}^{K} \frac{1}{|y_t^{(i)}|} \sum_{k=1}^{|y_t^{(i)}|} \min\Big(\rho_{t,k}^{(i)}(\theta)\tilde{A}^{(i)}, \text{clip}\big(\rho_{t,k}^{(i)}(\theta), 1\pm\epsilon\big)\tilde{A}^{(i)}\Big)\right] - \beta\mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}), \tag{2}$$

where $y_{t,k}^{(i)}$ denotes the $k$-th token of $y_t^{(i)}$. The token-level importance ratio is $\rho_{t,k}^{(i)}(\theta) = \pi_\theta\Big(y_{t,k}^{(i)} \mid y_{t,<k}^{(i)}\Big) / \pi_{\theta_{\text{old}}}\Big(y_{t,k}^{(i)} \mid y_{t,<k}^{(i)}\Big)$.

**Reward aggregation and advantage.** At each turn, the evaluator provides $\mathcal{E}(x, y_t^{(i)})$, yielding a per-turn reward $r_t^{(i)}$. We aggregate per-turn rewards into a trajectory-level outcome reward as follows:

$$R^{(i)} = \text{Agg}\big(\{r_t^{(i)}\}_{t=1}^{K}\big), \tag{3}$$

where $\text{Agg}(\cdot)$ can be, e.g., $\max_t$ (best over turns) or $\sum_t$ (dense shaping). To retain most of the turn-wise information, we use $\sum_t$ by default. Similar to the single-turn setting, the trajectory-level advantage is applied to all turns and tokens in $\mathcal{T}^{(i)}$ (i.e., $A_{t,i} = A_i$ for $t \in \{1, \ldots, K\}$).

## 3 LEARNING TO EVOLVE

In this section, we present Learning to Evolve (LtE), which trains a model to iteratively refine an initial solution using multi-turn RL with evaluator feedback. The core challenge is extending RLVR to a refinement horizon with sparse, validity-constrained rewards and nontrivial credit assignment across both turns and competing evolution attempts. LtE addresses this by shaping feedback into progress, combining turn-level and trajectory-level advantages, and optimizing a token-wise objective. We describe the multi-turn evolution procedure in Sec. 3.1, the optimization method in Sec. 3.2, and task-specific implementations in Sec. 3.3.

### 3.1 SCALABLE MULTI-TURN EVOLUTION

We formulate evolution as an iterative *refine-evaluate-update* process that starts from an initial solution $y_0$ for task input $x$. We compute initial feedback $f_0$ and initialize the evolution history $h_0 = [(y_0, f_0)]$. At turn $t$, the policy observes the state $s_{t-1} = [x; \text{Mem}(h_{t-1})]$, *i.e.*, the task input and a bounded length of recent history.

**History and memory representation.** We maintain an ordered history $h_t = [(y_0, f_0), \ldots, (y_t, f_t)]$ that stores the solutions and their feedback. To reduce computation, we do not store intermediate thinking tokens and instead retain only the final solution text $y_t$ and score $f_t$. Because $h_t$ grows linearly with $t$, we expose to the policy a fixed-size memory buffer of the most recent $M$ pairs:

$$\text{Mem}(h_t) = [(y_j, f_j), \ldots, (y_t, f_t)], \quad j = \max(0, t - M + 1). \tag{4}$$

**Multi-turn evolution.** For each turn $t \in \{1, \ldots, K\}$, the policy proposes a refinement conditioned on the current state, the evaluator scores it, and we append it to history:

- **Refine:** generate a refined solution $y_t \sim \pi_\theta(\cdot \mid s_{t-1})$.
- **Evaluate:** obtain evaluator feedback $f_t = \mathcal{E}(x, y_t)$.
- **Update:** append the new pair and refresh the state $h_t = h_{t-1} \,\|\, [(y_t, f_t)]$, $\quad s_t = [x; \text{Mem}(h_t)]$.

This design yields $O(M)$ memory per trajectory, and $K$ evaluator calls for refinements per trajectory.

**Evolution trajectories.** During training, we sample a batch of $G$ trajectories $\{\mathcal{T}^{(i)}\}_{i=1}^{G}$ from the same starting point $(x, y_0, f_0)$ to enable controlled comparisons across stochastic rollouts:

$$\mathcal{T}^{(i)} = (x, y_0, f_0, y_1^{(i)}, f_1^{(i)}, \ldots, y_K^{(i)}, f_K^{(i)}). \tag{5}$$

Given these trajectories, the learning algorithm must (i) attribute credit across turns within each trajectory (which refinements materially improved the solution), and (ii) compare trajectories within the batch (which evolutionary runs achieved better outcomes). Next, we introduce our policy update rule based on a turn-level advantage $A_t^{(i)}$ for each generated solution $y_t^{(i)}$.

### 3.2 Turn-level Optimization for Evolution

We cast multi-turn evolution as a stable RL problem by shaping evaluator feedback into a progress reward that only credits valid outputs and measures improvement over the initial solution. We then assign credit both within a trajectory (to identify the most helpful refinement steps) and across trajectories sampled for the same instance (to rank evolution strategies by overall progress). These signals are combined into a single advantage to train the policy with a token-wise clipped objective and KL regularization for stability.

**Turn-level reward for $y_t^{(i)}$.** At turn $t$ of trajectory $\mathcal{T}^{(i)}$, the evaluator returns feedback $f_t^{(i)} = (v_t^{(i)}, q_t^{(i)})$, where $v_t^{(i)} \in \{0, 1\}$ indicates whether the solution passes a task-specific validity check, and $q_t^{(i)} \in \mathbb{R}$ is a quality score (defined only when the output is valid and treated as arbitrary when invalid). We first define a *validity-gated* score:

$$S\Big(y_t^{(i)}\Big) = \mathbb{I}\Big[v_t^{(i)} = 1\Big] \times q_t^{(i)}. \tag{6}$$

This gating ensures that invalid solutions receive zero score and prevents the optimizer from exploiting the quality signal without satisfying validity constraints.

However, raw scores $S(\cdot)$ can be difficult to compare across turns because trajectories may start from different baselines and early turns may be systematically lower-scoring. To obtain a turn-wise learning signal that reflects *evolutionary progress* within a trajectory, we anchor all rewards to the initial solution $y_0$ and define the turn-level reward as the improvement over the starting point:

$$R\Big(y_t^{(i)}\Big) = S\Big(y_t^{(i)}\Big) - S(y_0). \tag{7}$$

By construction, $R(y_t^{(i)}) > 0$ indicates that the refinement at turn $t$ improves upon the initial solution, while $R(y_t^{(i)}) \leq 0$ indicates no progress (or regression) relative to $y_0$.

**Trajectory-level reward for $\mathcal{T}^{(i)}$.** To evaluate an entire $K$-turn evolution run, we aggregate turn-wise progress into a trajectory-level return. Using the anchored turn reward in Eq. 7, we define the trajectory-level reward for $\mathcal{T}^{(i)}$ as the cumulative progress over the horizon:

$$R\Big(\mathcal{T}^{(i)}\Big) = \sum_{t=1}^{K} R\Big(y_t^{(i)}\Big) = \sum_{t=1}^{K} \Big(S\Big(y_t^{(i)}\Big) - S(y_0)\Big). \tag{8}$$

Anchoring each turn to the same reference solution $y_0$ makes rewards comparable across evolution turns and across trajectories sampled from the same starting point, even when the evaluator's raw scores are instance-dependent and lack a consistent global scale. The resulting return $R(\mathcal{T}^{(i)})$ therefore promotes refinement strategies that yield *sustained* improvement over the full horizon, rather than optimizing for a single isolated intermediate gain.

**Intra-trajectory advantage for $y_t^{(i)}$.** While $R(y_t^{(i)})$ measuring absolute progress relative to the shared anchor $y_0$, we want to attribute credit *within* a single evolution run: which turn in trajectory $\mathcal{T}^{(i)}$ contributed more than others given the same history and horizon. To this end, we construct a within-trajectory baseline using the mean and standard deviation of turn rewards along $\mathcal{T}^{(i)}$:

$$\mu_{\text{intra}}^{(i)} = \frac{1}{K}\sum_{t=1}^{K} R\Big(y_t^{(i)}\Big) = \frac{1}{K} R\Big(\mathcal{T}^{(i)}\Big), \quad \sigma_{\text{intra}}^{(i)} = \sqrt{\frac{1}{K}\sum_{t=1}^{K}\Big(R\Big(y_t^{(i)}\Big) - \mu_{\text{intra}}^{(i)}\Big)^2}. \tag{9}$$

We then define the *intra-trajectory* (turn-wise) advantage as a normalized deviation from this baseline:

$$A^{(i)}_{t,\text{intra}} = \frac{R\Big(y^{(i)}_t\Big) - \mu^{(i)}_{\text{intra}}}{\sigma^{(i)}_{\text{intra}} + \epsilon}, \tag{10}$$

where $\epsilon > 0$ is a small constant for numerical stability. This normalization yields a scale-robust learning signal that ranks refinement steps *relative to other turns in the same trajectory*. Turns with $A^{(i)}_{t,\text{intra}} > 0$ correspond to refinements that achieve above-average progress for that particular evolution history, and are therefore reinforced more strongly than below-average turns.

**Inter-trajectory advantage for $\mathcal{T}^{(i)}$.** The intra-trajectory advantage in Eq. 10 attributes credit within a single evolution history, but it does not provide a *global* reference across different evolution attempts. A refinement step can be above average within its own trajectory while the overall strategy remains inferior to other rollouts on the same instance. To promote genuinely better evolution strategies, we therefore compare multiple trajectories sampled from the same starting point $(x, y_0)$.

For each trajectory $\mathcal{T}^{(i)} \in \{\mathcal{T}^{(i)}\}_{i=1}^{G}$, we compute a scalar return $R(\mathcal{T}^{(i)})$ (Eq. 8) summarizing its total progress over $K$ turns. We then normalize these returns within the group to obtain an *inter-trajectory* advantage:

$$\mu_{\text{inter}} = \frac{1}{G}\sum_{i=1}^{G} R\Big(\mathcal{T}^{(i)}\Big), \quad \sigma_{\text{inter}} = \sqrt{\frac{1}{G}\sum_{i=1}^{G}\Big(R\big(\mathcal{T}^{(i)}\big) - \mu_{\text{inter}}\Big)^2}, \quad A^{(i)}_{\text{inter}} = \frac{R\big(\mathcal{T}^{(i)}\big) - \mu_{\text{inter}}}{\sigma_{\text{inter}} + \epsilon}, \tag{11}$$

where $\epsilon > 0$ is a small constant for numerical stability.

This signal depends only on *relative* performance among concurrent rollouts on the same problem instance and is therefore robust to instance difficulty and the absolute scale of evaluator scores. In effect, $A^{(i)}_{\text{inter}}$ serves as a trajectory-level ranking signal: it encourages the policy to adopt refinement strategies that yield stronger aggregate progress than alternative evolution runs.

**Hierarchical advantage for $y^{(i)}_t$.** The two advantages capture complementary aspects of credit assignment: $A^{(i)}_{\text{inter}}$ ranks *trajectories* by overall strategy quality, whereas $A^{(i)}_{t,\text{intra}}$ identifies which *turns* within a trajectory were particularly effective. To leverage both signals in a single policy-gradient update, we define a hierarchical (unified) advantage for each turn-level action $y^{(i)}_t$ as

$$A^{(i)}_t = A^{(i)}_{\text{inter}} + \omega\, A^{(i)}_{t,\text{intra}}, \tag{12}$$

where $\omega \geq 0$ controls the trade-off between emphasizing globally superior evolution strategies and reinforcing locally strong refinement steps.

**Policy update (token-wise objective).** We optimize $\pi_\theta$ with a token-level PPO/GRPO-style objective so that every generated token in every refinement turn is weighted by the hierarchical advantage in Eq. 12. For each task $(x, y_0) \sim \mathcal{D}$, we sample a group of $G$ independent $K$-turn evolution trajectories $\{\mathcal{T}^{(i)}\}_{i=1}^{G}$ using the behavior policy $\pi_{\theta_{\text{old}}}$, where trajectory $i$ yields refinements $\{y^{(i)}_t\}_{t=1}^{K}$. Let $y^{(i)}_{t,k}$ denote the $k$-th token of $y^{(i)}_t$, and $y^{(i)}_{t,<k}$ the corresponding prefix. We maximize the following token-wise clipped objective:

$$\begin{aligned}\mathcal{J}(\theta) = \mathbb{E}_{\substack{(x,y_0)\sim\mathcal{D},\\ \{\mathcal{T}^{(i)}\}\sim\pi_{\theta_{\text{old}}}}}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{K}\sum_{t=1}^{K}\frac{1}{|y^{(i)}_t|}\sum_{k=1}^{|y^{(i)}_t|}\Big(\min\big(\rho^{(i)}_{t,k}(\theta)A^{(i)}_t,\, \text{clip}(\rho^{(i)}_{t,k}(\theta), 1-\epsilon, 1+\epsilon)\, A^{(i)}_t\big)\Big)\Bigg] \\ - \beta\mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\,,\end{aligned} \tag{13}$$

where $\epsilon$ is the PPO clipping threshold, $\beta$ controls KL regularization, and $\pi_{\text{ref}}$ is a fixed (or slowly updated) reference policy used to stabilize training. The token-level importance ratio conditions on both the turn state and the already-generated prefix: $\rho^{(i)}_{t,k}(\theta) = \pi_\theta\Big(y^{(i)}_{t,k} | s^{(i)}_{t-1}, y^{(i)}_{t,<k}\Big) / \pi_{\theta_{\text{old}}}\Big(y^{(i)}_{t,k} | s^{(i)}_{t-1}, y^{(i)}_{t,<k}\Big)$. This formulation performs credit assignment at the trajectory level via $A^{(i)}_{\text{inter}}$, at the turn level via $A^{(i)}_{t,\text{intra}}$, and applies the resulting hierarchical advantage to all tokens composing the refinement $y^{(i)}_t$.

### 3.3 IMPLEMENTATION: MOLECULE OPTIMIZATION TASKS

In molecule optimization, each refinement $y_t^{(i)}$ is a molecule in text form (e.g., SMILES). At each turn, the evaluator outputs feedback $f_t^{(i)} = (v_t^{(i)}, q_t^{(i)})$, where $v_t^{(i)} \in \{0, 1\}$ indicates whether the solution $y_t^{(i)}$ is a valid SMILES string and $q_t^{(i)} \in \mathbb{R}$ is a quality score. Following Eq. 6, we assign zero score to invalid SMILES and ensure that only valid solutions contribute to the rewards in Sec. 3.2.

**Quality score.** For valid solutions, the quality score $q_t^{(i)}$ measures property improvement while encouraging similarity to the initial molecule $y_0$. We first define a scalar property score:

$$q_{\text{prop}}(y; P^\star) = \sum_{i=1}^{N} w_i \, g_i\big(P_i(y), P_i^\star\big), \tag{14}$$

where $P(y) = \{P_1(y), \ldots, P_N(y)\}$ are the target properties, $P^\star = \{P_1^\star, \ldots, P_N^\star\}$ are the desired direction for each property (e.g., increase), $w_i \geq 0$ are weights, and $g_i(\cdot)$ increases as $y$ better matches the direction. Typically, molecular optimization requires preserving structural similarity to the $y_0$ to retain its biological activity (López-Pérez et al., 2024; Lipinski & Hopkins, 2004). To incorporate similarity as a soft constraint, let $\text{Sim}(\cdot, \cdot)$ be a similarity measure and $\delta$ a target threshold. We define the similarity penalty as the objective shown as follows (Wang et al., 2025c): $r_{\text{sim}}(y; y_0, \delta) = \max\big(0,\ \delta - \text{Sim}(y, y_0)\big)$, and define the evaluator quality score as

$$q_t^{(i)} = q_{\text{prop}}\Big(y_t^{(i)}; P^\star\Big) - \lambda \, r_{\text{sim}}\Big(y_t^{(i)}; y_0, \delta\Big), \tag{15}$$

where $\lambda \geq 0$ controls the trade-off between property improvement and similarity to $y_0$. By default, we keep the $\lambda = 1$ to balance the optimization. We employ $\delta = 0.4$ as the similarity threshold, aligning with expert judgments of molecular structural dissimilarity (Rogers & Hahn, 2010).

## 4 EXPERIMENTS

In this section, we evaluate the performance of our proposed method, LtE. We first outline the experimental setup in Sec. 4.1, followed by a detailed discussion of the quantitative results in Sec. 4.2. Finally, we provide further experiments on scaling the computing budgets in Sec. 4.3.

### 4.1 EXPERIMENT SETTINGS

In what follows, we describe the setting of the experiments, including the dataset, baselines, and evaluation metrics. Detailed settings are provided in Appendix E.1.

**Datasets.** We employ two instruction-based molecular optimization benchmarks: TOMG-Bench (Li et al., 2024) for single-property optimization and MuMOInstruct (Dey et al., 2025) for multiple-property ones. These datasets evaluate the capability of LLMs to modify molecular structures according to specific property constraints.

**Baselines.** We compare LtE against three groups of methods: (1) Molecule optimization baselines: GraphGA (Jensen, 2019), REINVENT (Olivecrona et al., 2017), and MOLLEO (Wang et al., 2025a); (2) General-purpose LLMs: instruction-tuned models, including Qwen-2.5-3B/7B (Yang et al., 2024a), Qwen-3-4B-2507 (Yang et al., 2025), and Llama-3.1-8B (Grattafiori et al., 2024); (3) RL fine-tuning baselines: GRPO and RLOO (Ahmadian et al., 2024). For GraphGA and REINVENT, we use the default optimization configurations from PMO (Gao et al., 2022). For MOLLEO, we use Qwen-2.5-3B as the base model for a fair comparison. All LLM baselines and LtE use $K{=}5$ turns. We report the best-scoring molecule among the generated solutions.

**Evaluation Metrics.** We report four metrics following prior work (Dey et al., 2025). Specifically, we employ (1) Validity (`Valid`): The fraction of the best-scoring generated solution is chemically valid, which is evaluated by RDKit (Landrum et al., 2025); (2) Success Rate (`SR`): The fraction of best-scoring molecules improves all target properties; (3) Similarity (`Sim`): The average Tanimoto similarity between the proposed solutions and the initial ones, computed using Morgan fingerprints. Note that if an optimization attempt fails (e.g., produces an invalid molecule), we return the initial molecule $y_0$; therefore, such unsuccessful cases contribute a similarity of 1.0; (4) Relative Improvement (`RI`): The average improvement of each target property relative to its initial value of $y_0$.

Table 1: Overall performance of single-property molecular optimization, where each reported result is obtained by averaging three independent runs.

| Model | QED | | | | LogP | | | | MR | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Valid↑ | SR↑ | Sim | RI↑ | Valid↑ | SR↑ | Sim | RI↑ | Valid↑ | SR↑ | Sim | RI↑ |
| **Evolutionary Algorithm** | | | | | | | | | | | | |
| Graph-GA | 1.000 | 0.440 | 0.842 | 0.156 | 1.000 | 0.440 | 0.838 | 0.476 | 1.000 | 0.440 | 0.822 | 0.127 |
| Reinvent 4 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 |
| MOLLEO | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.286 | 0.887 | 0.100 | 1.000 | 0.320 | 0.883 | 0.049 |
| **General-purpose LLMs** | | | | | | | | | | | | |
| Qwen2.5-3B Instruct | 0.477 | 0.267 | 0.900 | 0.035 | 0.573 | 0.343 | 0.865 | 0.358 | 0.493 | 0.313 | 0.885 | 0.036 |
| Qwen3-4B Instruct | 0.363 | 0.270 | 0.914 | 0.029 | 0.417 | 0.360 | 0.895 | 0.114 | 0.353 | 0.307 | 0.895 | 0.040 |
| Qwen2.5-7B Instruct | 0.750 | 0.570 | 0.796 | 0.094 | 0.810 | 0.682 | 0.763 | 0.534 | 0.777 | 0.587 | 0.772 | 0.116 |
| Llama3.1-8B Instruct | 0.677 | 0.427 | 0.853 | 0.063 | 0.776 | 0.625 | 0.777 | 0.471 | 0.753 | 0.593 | 0.782 | 0.076 |
| **Training Algorithm** | | | | | | | | | | | | |
| GRPO | 0.933 | 0.350 | 0.863 | 0.184 | 0.963 | 0.737 | 0.739 | 2.481 | 0.963 | 0.453 | 0.826 | 0.548 |
| RLOO | 1.000 | 0.407 | 0.911 | 0.378 | 1.000 | 0.993 | 0.812 | 9.984 | 1.000 | 0.410 | 0.937 | 2.496 |
| **LtE (Ours)** | 1.000 | 0.410 | 0.892 | 0.385 | 0.987 | 0.777 | 0.788 | 13.555 | 0.980 | 0.490 | 0.908 | 2.552 |

Table 2: Overall performance of multi-property molecular optimization, where each reported result is from averaging three independent runs. BDP, BDQ, and BPQ represent combinations of targets.

| Model | BDP | | | | BDQ | | | | BPQ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Valid↑ | SR↑ | Sim | RI↑ | Valid↑ | SR↑ | Sim | RI↑ | Valid↑ | SR↑ | Sim | RI↑ |
| **Evolutionary Algorithm** | | | | | | | | | | | | |
| Graph-GA | 1.000 | 0.134 | 0.944 | 0.343 | 1.000 | 0.118 | 0.951 | 0.653 | 1.000 | 0.059 | 0.976 | 0.183 |
| Reinvent 4 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 |
| MOLLEO | 1.000 | 0.082 | 0.973 | 0.085 | 1.000 | 0.020 | 0.992 | 0.008 | 1.000 | 0.030 | 0.987 | 0.181 |
| **General-purpose LLMs** | | | | | | | | | | | | |
| Qwen2.5-3B Instruct | 0.213 | 0.100 | 0.965 | 0.171 | 0.160 | 0.127 | 0.957 | 0.068 | 0.208 | 0.033 | 0.988 | 0.010 |
| Qwen3-4B Instruct | 0.014 | 0.010 | 0.998 | 0.010 | 0.023 | 0.023 | 0.995 | 0.004 | 0.013 | 0.000 | 1.000 | 0.000 |
| Qwen2.5-7B Instruct | 0.491 | 0.351 | 0.899 | 0.503 | 0.346 | 0.284 | 0.914 | 0.276 | 0.452 | 0.139 | 0.961 | 0.080 |
| Llama3.1-8B Instruct | 0.512 | 0.381 | 0.869 | 0.834 | 0.337 | 0.229 | 0.924 | 0.189 | 0.597 | 0.155 | 0.957 | 0.050 |
| **Training Algorithm** | | | | | | | | | | | | |
| GRPO | 0.409 | 0.196 | 0.937 | 0.499 | 0.114 | 0.059 | 0.977 | 0.203 | 0.409 | 0.076 | 0.975 | 0.174 |
| RLOO | 0.543 | 0.282 | 0.923 | 1.461 | 0.255 | 0.160 | 0.957 | 0.294 | 0.545 | 0.188 | 0.947 | 0.141 |
| **LtE (Ours)** | 0.560 | 0.347 | 0.849 | 4.111 | 0.203 | 0.157 | 0.940 | 0.254 | 0.657 | 0.139 | 0.935 | 0.154 |

Table 3: Overall performance of multi-property molecular optimization on *unseen instructions*, where each reported result is from averaging three independent runs. BDP, BDQ, and BPQ represent combinations of targets.

| Model | BDP | | | | BDQ | | | | BPQ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Valid↑ | SR↑ | Sim | RI↑ | Valid↑ | SR↑ | Sim | RI↑ | Valid↑ | SR↑ | Sim | RI↑ |
| **Evolutionary Algorithm** | | | | | | | | | | | | |
| Graph-GA | 1.000 | 0.134 | 0.944 | 0.343 | 1.000 | 0.118 | 0.951 | 0.653 | 1.000 | 0.059 | 0.976 | 0.183 |
| Reinvent 4 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 |
| MOLLEO | 1.000 | 0.052 | 0.981 | 0.162 | 1.000 | 0.039 | 0.992 | 0.023 | 1.000 | 0.020 | 0.993 | 0.019 |
| **General-purpose LLMs** | | | | | | | | | | | | |
| Qwen2.5-3B Instruct | 0.210 | 0.103 | 0.962 | 0.187 | 0.154 | 0.111 | 0.962 | 0.085 | 0.221 | 0.056 | 0.982 | 0.026 |
| Qwen3-4B Instruct | 0.072 | 0.062 | 0.981 | 0.113 | 0.085 | 0.078 | 0.972 | 0.077 | 0.026 | 0.003 | 0.999 | 0.001 |
| Qwen2.5-7B Instruct | 0.399 | 0.237 | 0.926 | 0.387 | 0.363 | 0.248 | 0.927 | 0.176 | 0.370 | 0.162 | 0.955 | 0.058 |
| Llama3.1-8B Instruct | 0.502 | 0.320 | 0.896 | 2.511 | 0.327 | 0.219 | 0.928 | 0.185 | 0.604 | 0.168 | 0.943 | 0.066 |
| **Training Algorithm** | | | | | | | | | | | | |
| GRPO | 0.464 | 0.247 | 0.911 | 1.014 | 0.098 | 0.052 | 0.978 | 0.056 | 0.399 | 0.099 | 0.968 | 0.213 |
| RLOO | 0.502 | 0.237 | 0.932 | 0.842 | 0.284 | 0.147 | 0.957 | 0.323 | 0.512 | 0.132 | 0.961 | 0.108 |
| **LtE (Ours)** | 0.560 | 0.323 | 0.862 | 3.627 | 0.203 | 0.137 | 0.950 | 0.158 | 0.584 | 0.125 | 0.941 | 0.140 |

### 4.2 Quantitative Results

We summarize the empirical observations regarding single-objective and multi-objective optimization.

**LtE enables effective yet controlled single-property refinement.** Tab. 1 reveals a clear separation between methods. Training-free approaches, including evolutionary algorithms and general-purpose LLMs, maintain relatively high similarity (Sim≈0.76-1.00 for LogP), and their improvements remain small ($\leq$0.127 on MR). In contrast, LtE achieves high property gains on LogP and MR (RI=13.555 and 2.552) while preserving high structural similarity (Sim=0.788 on LogP). Relative to RLVR baselines, LtE retains comparable refinement quality while delivering larger gains: for LogP, it surpasses RLOO in improvement with similar similarity. Compared with GRPO, LtE yields stronger progress without sacrificing validity. Overall, these results support our motivation that turn-level credit assignment enables more reliable conversion of evaluator feedback into effective refinements.

**LtE helps the model to achieve better performance on multi-property molecular optimization.** Tab. 2 reports optimization over multi-property combinations, which is harder than single-objective refinement due to coupled constraints and target trade-offs. LtE achieves better results than other methods. Importantly, these improvements remain feasible: LtE maintains high similarity with the

Table 4: Overall performance of single-property molecular optimization with Qwen-2.5-7B Instruct, where each reported result is obtained by averaging three independent runs.

| Model | QED | | | | LogP | | | | MR | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Valid↑ | SR↑ | Sim | RI↑ | Valid↑ | SR↑ | Sim | RI↑ | Valid↑ | SR↑ | Sim | RI↑ |
| **General-purpose LLMs** | | | | | | | | | | | | |
| Qwen2.5-7B Instruct | 0.750 | 0.570 | 0.796 | 0.094 | 0.810 | 0.682 | 0.763 | 0.534 | 0.777 | 0.587 | 0.772 | 0.116 |
| **Training Algorithm** | | | | | | | | | | | | |
| GRPO | 0.970 | 0.577 | 0.864 | 0.408 | 0.980 | 0.893 | 0.795 | 8.924 | 0.993 | 0.603 | 0.850 | 2.431 |
| RLOO | 0.907 | 0.573 | 0.827 | 0.406 | 0.990 | 0.930 | 0.710 | 10.058 | 0.970 | 0.560 | 0.809 | 2.603 |
| **LtE (Ours)** | 0.987 | 0.613 | 0.761 | 0.422 | 0.980 | 0.843 | 0.725 | 19.899 | 0.990 | 0.457 | 0.825 | 6.055 |

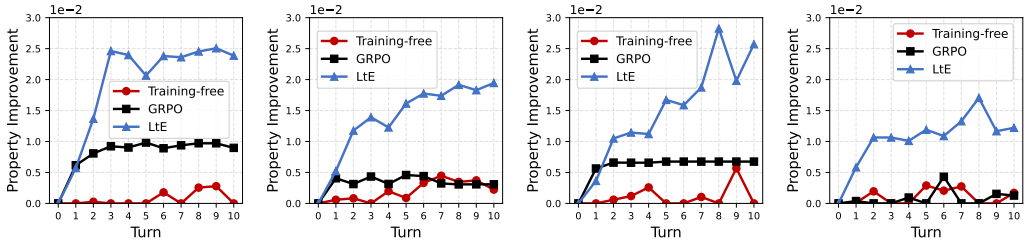


Figure 3: Scaling the number of evolution rounds on multi-property optimization with seen instructions (BDQ, BDP, BPQ) and unseen instructions (BDQ), respectively. Training-free approaches yield only small, inconsistent gains. GRPO delivers early improvements but quickly plateaus, whereas LtE continues to improve steadily as additional evolution turns are introduced.

initial molecules. While BDQ and BPQ offer smaller headroom overall, LtE achieves comparable improvements to RL training baselines (RI=0.254 on BDQ and 0.154 on BPQ).

**Generalization to Unseen Tasks.** Tab. 3 evaluates multi-property optimization under *unseen instruction styles* that never appear in training, directly evaluating whether the model has learned an *evolve-by-refinement* behavior rather than relying on training samples. Notably, LtE continues to produce noticeable gains on the multiple target optimization: it achieves the best BDP improvement (RI=3.627), surpassing both training baselines and general-purpose LLMs. On BDQ and BPQ, LtE remains competitive (RI=0.158 on BDQ and 0.140 on BPQ). Overall, these results support that LtE learns a robust refinement strategy that can refine solutions beyond the training instruction style.

### 4.3 Further Studies

**Increasing the Model Parameter-sizes.** In Tab. 4, we report results with Qwen2.5-7B Instruct. Crucially, increasing the parameter sizes allows LtE to convert capacity into substantially greater property gains, achieving the best RI on the more challenging objectives, surpassing its counterparts by a large margin, indicating the effectiveness of the turn-wise credit assignment.

**Extending the Evolution Turns.** We study how performance scales with a longer refinement horizon by increasing the number of evolution turns. Fig. 3 shows that simply allocating more turns to the training-free baseline does not reliably improve outcomes: gains remain close to zero and fluctuate across turns. GRPO exhibits a different failure mode: it achieves most of its improvement within the first several turns and then largely plateaus, with only marginal changes thereafter. In contrast, LtE continues to accumulate progress as additional turns are introduced, enlarging the gap over baselines.

## 5 Conclusion

In this paper, we introduce Learning to Evolve (LtE), a learning-centric framework that trains multi-turn refinement via turn- and trajectory-level credit assignment. LtE converts per-turn evaluator scores into a progress signal by rewarding validity-gated improvements over the initial solution and crediting both turns and trajectories for improvement. Across molecular optimization benchmarks, LtE achieves larger relative improvements, outperforming training-free and RLVR baselines under the same budgets. Moreover, LtE continues to improve as the evolution horizon grows. More broadly, LtE provides a general recipe for training LLMs to *evolve* solutions with evaluator feedback. Future work includes extending LtE beyond molecular optimization to other open-ended problems (e.g., algorithm synthesis) to further enhance scientific discovery with verifiable objectives.

## REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Viraj Bagal, Rishal Aggarwal, PK Vinod, and U Deva Priyakumar. Molgpt: molecular generation using a transformer-decoder model. *Journal of chemical information and modeling*, 62(9):2064–2076, 2021.

Lei Bai, Zhongrui Cai, Yuhang Cao, Maosong Cao, Weihan Cao, Chiyu Chen, Haojiong Chen, Kai Chen, Pengcheng Chen, Ying Chen, et al. Intern-s1: A scientific multimodal foundation model. *arXiv preprint arXiv:2508.15763*, 2025.

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

Jingyi Chai, Shuo Tang, Rui Ye, Yuwen Du, Xinyu Zhu, Mengcheng Zhou, Yanfeng Wang, Weinan E, Yuzhi Zhang, Linfeng Zhang, and Siheng Chen. Scimaster: Towards general-purpose scientific ai agents, part i. x-master as foundation: Can we lead on humanity's last exam? *arXiv preprint arXiv:2507.05241*, 2025.

Jiaqi Chen, Bang Zhang, Ruotian Ma, Peisong Wang, Xiaodan Liang, Zhaopeng Tu, Xiaolong Li, and Kwan-Yee K Wong. Spc: Evolving self-play critic via adversarial games for llm reasoning. *arXiv preprint arXiv:2504.19162*, 2025a.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025b.

Xinyun Chen, Maxwell Lin, Nathanael Sch"arli, and Denny Zhou. Teaching large language models to self-debug. In *ICLR*, 2024.

Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

Vishal Dey, Xiao Hu, and Xia Ning. Generalizing large language models for multi-property molecule optimization. *arXiv preprint arXiv:2502.13398*, 2025.

Luyu Fan, Liang Tan, Zhangcheng Chen, Jianzhong Qi, Fen Nie, Zhipu Luo, Jianjun Cheng, and Sheng Wang. Haloperidol bound d2 dopamine receptor structure inspired the discovery of subtype selective ligands. *Nature communications*, 11(1):1074, 2020.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. In *NeurIPS*, 2025.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *AAAI*, 2024.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. In *NeurIPS*, 2022.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025a.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *arXiv preprint arXiv:2505.23564*, 2025b.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *ICLR*, 2024.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, pp. 1–3, 2025.

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

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

Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, pp. 3393–3403. PMLR, 2020.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Greg Landrum, Paolo Tosco, Brian Kelley, Ricardo Rodriguez, David Cosgrove, Riccardo Vianello, Peter Gedeck, Gareth Jones, Eisuke Kawashima, Dan Nealschneider, et al. rdkit/rdkit: 2025_03_1 (q1 2025) release. *Zenodo*, 2025.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025.

R Jo W Le Fevre. Molecular refractivity and polarizability. In *Advances in Physical Organic Chemistry*, volume 3, pp. 1–90. Elsevier, 1965.

Jiatong Li, Junxian Li, Yunqing Liu, Dongzhan Zhou, and Qing Li. Tomg-bench: Evaluating llms on text-based open molecule generation. *arXiv preprint arXiv:2412.14642*, 2024.

Junxian Li, Di Zhang, Xunzhi Wang, Zeying Hao, Jingdi Lei, Qian Tan, Cai Zhou, Wei Liu, Yaotian Yang, Xinrui Xiong, et al. Chemvlm: Exploring the power of multimodal large language models in chemistry area. In *AAAI*, 2025.

Christopher Lipinski and Andrew Hopkins. Navigating chemical space for biology and medicine. *Nature*, 432(7019):855–861, 2004.

Christopher A Lipinski, Franco Lombardo, Beryl W Dominy, and Paul J Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced drug delivery reviews*, 23(1-3):3–25, 1997.

Shengchao Liu, Jiongxiao Wang, Yijin Yang, Chengpeng Wang, Ling Liu, Hongyu Guo, and Chaowei Xiao. Conversational drug editing using retrieval and domain feedback. In *ICLR*, 2024a.

Zhiyuan Liu, Sihang Li, Yanchen Luo, Hao Fei, Yixin Cao, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. Molca: Molecular graph-language modeling with cross-modal projector and uni-modal adapter. In *EMNLP*, 2023.

Zuxin Liu, Thai Quoc Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh R N, Liangwei Yang, Silvio Savarese, Juan Carlos Niebles, Huan Wang, Shelby Heinecke, and Caiming Xiong. Apigen: Automated PIpeline for generating verifiable and diverse function-calling datasets. In *NeurIPS Datasets and Benchmarks Track*, 2024b.

Hannes H Loeffler, Jiazhen He, Alessandro Tibo, Jon Paul Janet, Alexey Voronov, Lewis H Mervin, and Ola Engkvist. Reinvent 4: modern ai–driven generative molecule design. *Journal of Cheminformatics*, 16(1):20, 2024.

Kenneth López-Pérez, Juan F Avellaneda-Tamayo, Lexin Chen, Edgar López-López, K Eurídice Juárez-Mercado, José L Medina-Franco, and Ramón Alain Miranda-Quintana. Molecular similarity: Theory, applications, and perspectives. *Artificial Intelligence Chemistry*, 2(2):100077, 2024.

Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. Octotools: An agentic framework with extensible tools for complex reasoning. *arXiv preprint arXiv:2502.11271*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*, 2023.

Siddharth M Narayanan, James D Braza, Ryan-Rhys Griffiths, Albert Bou, Geemi Wellawatte, Mayk Caldas Ramos, Ludovico Mitchener, Samuel G Rodriques, and Andrew D White. Training a scientific reasoning model for chemistry. In *NeurIPS*, 2025.

Tung Nguyen and Aditya Grover. Lico: Large language models for in-context molecular optimization. In *ICLR*, 2025.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

Qizhi Pei, Wei Zhang, Jinhua Zhu, Kehan Wu, Kaiyuan Gao, Lijun Wu, Yingce Xia, and Rui Yan. Biot5: Enriching cross-modal integration in biology with chemical knowledge and natural language associations. In *EMNLP*, 2023.

Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *ICLR*, 2024.

Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *NeurIPS*, 2024.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

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

Ning Shang, Yifei Liu, Yi Zhu, Li Lyna Zhang, Weijiang Xu, Xinyu Guan, Buze Zhang, Bingcheng Dong, Xudong Zhou, Bowen Zhang, et al. rstar2-agent: Agentic reasoning technical report. *arXiv preprint arXiv:2508.20722*, 2025.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, et al. Efficient evolutionary search over chemical space with large language models. In *ICLR*, 2025a.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

Ziqing Wang, Yibo Wen, William Pattie, Xiao Luo, Weimin Wu, Jerry Yao-Chieh Hu, Abhishek Pandey, Han Liu, and Kaize Ding. Polo: Preference-guided multi-turn reinforcement learning for lead optimization. *arXiv preprint arXiv:2509.21737*, 2025c.

Di Wu, Qi Chen, Xiaojie Chen, Feng Han, Zhong Chen, and Yi Wang. The blood–brain barrier: Structure, regulation and drug delivery. *Signal transduction and targeted therapy*, 8(1):217, 2023.

Yingce Xia, Peiran Jin, Shufang Xie, Liang He, Chuan Cao, Renqian Luo, Guoqing Liu, Yue Wang, Zequn Liu, Yuan-Jyue Chen, et al. Nature language model: Deciphering the language of nature for scientific discovery. *arXiv preprint arXiv:2502.07527*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *ICLR*, 2024b.

Geyan Ye, Xibao Cai, Houtim Lai, Xing Wang, Junhong Huang, Longyue Wang, Wei Liu, and Xiangxiang Zeng. Drugassist: A large language model for molecule optimization. *Briefings in Bioinformatics*, 26(1):bbae693, 2025.

Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, Yuliang Yan, Jiatong Li, Weiran Huang, Xiangyu Yue, Wanli Ouyang, et al. Chemllm: A chemical large language model. *arXiv preprint arXiv:2402.06852*, 2024.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025a.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025b.

Yiqun Zhang, Peng Ye, Xiaocui Yang, Shi Feng, Shufei Zhang, Lei Bai, Wanli Ouyang, and Shuyue Hu. Nature-inspired population-based evolution of large language models. *arXiv preprint arXiv:2503.01155*, 2025c.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. Rlvmr: Reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. *arXiv preprint arXiv:2507.22844*, 2025d.

Zihan Zhao, Bo Chen, Ziping Wan, Lu Chen, Xuanze Lin, Shiyang Yu, Situo Zhang, Da Ma, Zichen Zhu, Danyang Zhang, et al. Chemdfm-r: A chemical reasoning llm enhanced with atomized chemical knowledge. *arXiv preprint arXiv:2507.21990*, 2025a.

Zihan Zhao, Da Ma, Lu Chen, Liangtai Sun, Zihao Li, Yi Xia, Bo Chen, Hongshen Xu, Zichen Zhu, Su Zhu, et al. Developing chemdfm as a large language foundation model for chemistry. *Cell Reports Physical Science*, 6(4), 2025b.

# APPENDIX

## A ETHICS STATEMENT

The study does not involve human subjects, data set releases, potentially harmful insights, applications, conflicts of interest, sponsorship, discrimination, bias, fairness concerns, privacy or security issues, legal compliance issues, or research integrity issues.

## B IMPACT STATEMENT

This paper presents Learning to Evolve (LtE), a method that trains language models to improve solutions through multi-turn refinement using verifiable scores, with experiments in molecular optimization. The goal is to advance LLMs for evolution and make the process more compute-efficient. LtE can support scientific and engineering applications where solutions can be evaluated automatically by reducing trial-and-error searches and producing higher-quality results under the same computational budgets. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

## C LLM USAGE DISCLOSURE

This submission was prepared with the assistance of LLMs, which were utilized for refining content and checking grammar. The authors assume full responsibility for the entire content of the manuscript, including any potential issues related to plagiarism and factual accuracy. It is confirmed that no LLM is listed as an author.

## D RELATED WORK

In this section, we provide a comprehensive discussion on three axes:

### D.1 TRAINING-FREE EVOLUTION

Early approaches to training-free improvement focused on iterative refinement, where a model critiques and updates its initial response. Self-Refine (Madaan et al., 2023) and Self-Debug (Chen et al., 2024) demonstrated that LLMs could improve their performance by generating intrinsic feedback. This concept was formalized by Reflexion (Shinn et al., 2023), which utilizes verbal reinforcement learning to maintain a memory of past errors, effectively evolving the agent's context over multiple trials. However, the limit of this approach is that models cannot often self-correct without external oracles (Huang et al., 2024). To address this, recent methods utilize recursive introspection (Qu et al., 2024) or leverage LLMs as explicit optimizers (Yang et al., 2024b) to guide the refinement trajectory, treating the prompt or the answer as a variable to be optimized at test time.

Moving beyond single-instance refinement, population-based evolution mimics natural selection to improve reasoning at test time. The Darwin-Godel Machine (Zhang et al., 2025b) and nature-inspired evolution (Zhang et al., 2025c) propose open-ended frameworks where agents self-replicate and mutate to solve increasingly complex tasks. In the domain of scientific discovery, AlphaEvolve (Novikov et al., 2025) utilizes a coding agent to evolve algorithms dynamically. Similarly, SPC (Chen et al., 2025a) leverages self-play mechanisms where the model acts as both the generator and the selector, creating an adversarial evolutionary dynamic that refines capabilities without human annotation.

### D.2 MULTI-TURN RLVR

Real-world reasoning often requires external grounding. ToolLLM (Qin et al., 2024) established a massive baseline for instruction-tuning models on real-world APIs, while APIGen (Liu et al., 2024b) automated the generation of verifiable function-calling datasets to ensure reliability. More advanced frameworks like Confucius (Gao et al., 2024) introduced iterative tool learning via curriculum. Furthermore, SciMaster (Chai et al., 2025) and OctoTools (Lu et al., 2025) demonstrated how

extensible tool sets allow agents to solve complex scientific problems, while Alita (Qiu et al., 2025) proposed a generalist architecture for scalable reasoning with minimal predefinition.

The frontier of multi-turn reasoning lies in agentic RL, where models optimize entire reasoning trajectories, including tool use and self-correction, rather than just next-token prediction (Zhang et al., 2025a). DeepSeek-R1 (Guo et al., 2025a) illustrated the power of large-scale RL for incentivizing pure reasoning. To extend this to agentic workflows, GiGPO (Feng et al., 2025) introduced a robust framework for training agents in dynamic environments, utilizing hierarchical policy optimization to manage complex interaction spaces. Parallel efforts like RAGEN (Wang et al., 2025b) and rStar2-Agent (Shang et al., 2025) focus on self-evolution and rigorous reasoning chains.

### D.3 Optimizing for the Open-ended Problems

**Molecular optimization.** Molecular optimization involves modifying molecular structures to enhance desired properties, such as drug-likeness (Bickerton et al., 2012), while preserving structural similarity and biological activity (López-Pérez et al., 2024; Lipinski & Hopkins, 2004). Early approaches relied on reinforcement learning (e.g., REINVENT, ReLeaSE) (Olivecrona et al., 2017; Loeffler et al., 2024; Popova et al., 2018), genetic algorithms (Jensen, 2019), and Bayesian optimization (Korovina et al., 2020), with benchmarks like PMO established to standardize evaluation (Gao et al., 2022). Before the LLM era, research focused on pre-training Transformers like ChemBERTa and MolGPT on chemical syntax (Chithrananda et al., 2020; Irwin et al., 2022; Bagal et al., 2021) or exploring cross-modal integration (Pei et al., 2023; Liu et al., 2023). This evolution has culminated in the development of domain-specialized foundation models such as ChemLLM (Zhang et al., 2024), ChemDFM (Zhao et al., 2025b), and multimodal systems like ChemVLM and Intern-s1 (Li et al., 2025; Bai et al., 2025), which possess advanced reasoning capabilities and the ability to interpret chemical images (Zhao et al., 2025a; Xia et al., 2025; Narayanan et al., 2025).

Building on these foundation models, the most recent works leverage the reasoning power of LLMs specifically for molecular optimization tasks. Unlike traditional generative models, these approaches often utilize "text-to-molecule" paradigms, employing retrieval-augmented generation and conversational feedback to guide the optimization process interactively (Ye et al., 2025; Liu et al., 2024a). LLMs are increasingly applied as mutation operators to enable evolutionary search over chemical space (Wang et al., 2025a) or fine-tuned as property predictors and scoring oracles within knowledge-guided pipelines, such as LICO (Nguyen & Grover, 2025). Furthermore, current research focuses on generalizing these agents to handle multi-property optimization tasks simultaneously, addressing the complex trade-offs inherent in drug discovery (Dey et al., 2025). Notably, POLO introduces a preference-based optimization framework in a similar multi-turn manner for the molecular optimization task (Wang et al., 2025c).

**Mathematical discovery.** Earlier test-time improvement methods, Self-Refine (Madaan et al., 2023) uses model-generated grading to rewrite answers, and Reflexion (Shinn et al., 2023) adds memory so the agent learns across attempts, yet models still often fail to spot their own mistakes (Huang et al., 2024). Recent work moves beyond single-answer fixes via recursive deliberation (Qu et al., 2024), prompt optimization at inference time, and population-based, evolution-inspired search where many variants compete, mutate, and are selected (Zhang et al., 2025b;c), including self-play setups like SPC (Chen et al., 2025a). In math, this evolutionary lens appears in code evolving, where an LLM mutates programs and retains winners (Romera-Paredes et al., 2024), multi-LLM systems that co-evolve solutions, search strategies, and prompts under strict verification, recently improving complex matrix multiplication (Novikov et al., 2025), and hybrid pipelines, where formal proof assistants return error traces that guide iterative proof repair (Hubert et al., 2025).

## E Experiments Details

### E.1 Experiment Settings

We employ `verl` (Sheng et al., 2025) as our training backend and `verl-agent` (Feng et al., 2025) as the training framework. For the evolutionary algorithm, we use the default setting in PMO (Gao et al., 2022) and MOLLEO (Kwon et al., 2023). For general-purpose LLMs, we use the latest checkpoints from the corresponding HuggingFace repositories. All system prompts remain defaults.

Table 5: Prompt template for initial refinement (single-property optimization).

| |
|---|
| You are an expert medicinal chemist specializing in molecular optimization. Your goal is to propose a new molecule that satisfies the given constraints. Ensure the proposed molecule in SMILES format is valid, plausible, and does not duplicate the previous molecules. Prefer small edits that preserve chemical similarity.<br>Your task:<br>{task_description}<br><br>Now it's your turn to respond to the current step.<br>You should first conduct a reasoning process, which MUST be enclosed within <think> </think> tags.<br>After finishing your reasoning, suggest a valid molecule in SMILES format that either extends a previously successful modification or explores a new promising direction expected to satisfy the given constraints.<br>If you are ready to provide the self-contained solution, provide the molecule only inside <answer> ... </answer>. |

Table 6: Prompt template for initial refinement (multiple-property optimization).

| |
|---|
| You are an expert medicinal chemist specializing in multi-property molecular optimization. Your goal is to propose a new molecule that satisfies the given constraints across all target properties. Ensure the proposed molecule in SMILES format is valid, plausible, and does not duplicate previous molecules. Prefer small edits that preserve chemical similarity.<br>Your task:<br>{task_description}<br><br>Now it's your turn to respond to the current step.<br>You should first conduct a reasoning process, which MUST be enclosed within <think> </think> tags.<br>After finishing your reasoning, suggest a valid molecule in SMILES format that either extends a previously successful modification or explores a new promising direction expected to satisfy the given constraints.<br>If you are ready to provide the self-contained solution, provide the molecule only inside <SMILES> ... </SMILES>. |

**Sampling parameters.** All LLMs are inferred with consistent sampling parameters; we set the temperature as $0.4$ for single-property optimization, and the context window for each evolution turn is set to $2048$. For multi-property optimization, we set the content window as $4096$.

**Training configurations.** We fix the random seed as 42 and use a batch size of 128. Policy optimization uses a low learning rate of $1e-6$ We keep the KL parameters $\beta$ as default to `verl`. The trade-off term $\omega$ for $A^{(i)}_{\text{inter}}$ and $A^{(i)}_{t,\text{intra}}$ is set to 1.0 as default. We train the model with 2 epochs with a group size of 16. The training iteration is set to 5. Notably, all training methods, including GRPO and RLOO, are trained in the same multi-turn manner. All the experiments are conducted using 8 Nvidia A100 GPUs with 80 GB memories.

**Prompts.** We provide the prompt templates we adopted for single-property and multiple-property optimization task. Specifically, for each initial refinement, we employ prompts in Tab. 5 for single-property optimization and Tab. 6 for multiple-property ones. For subsequent turns, we employ Tab. 7 and Tab. 8 for single and multiple property optimization tasks, respectively.

### E.2 Evaluation Metrics Details

**Success Rate (SR).** SR is the fraction of test cases in which the optimized molecule improves all target properties while satisfying the similarity constraint $\text{sim}(m, m') \geq 0.4$. A successful optimization must meet the similarity constraint and achieve improvement in the desired direction for every target property. Formally, let $\Delta P_{i,j} = P_j(m'_i) - P_j(m_i)$ and define $\text{sgn}(w_j) = +1$ for

Table 7: Prompt template for subsequent turns (single-property optimization).

| |
|---|
| You are an expert medicinal chemist specializing in molecular optimization. Your goal is to propose a new molecule that satisfies the given constraints. Ensure the proposed molecule in SMILES format is valid, plausible, and does not duplicate the previous molecules. Prefer small edits that preserve chemical similarity.<br>Your task:<br>{task_description}<br><br>Prior to this step, you have already taken {step_count} step(s).<br>Below is the interaction history, where <answer> </answer> wrapped your past molecules and <results> </results> wrapped the corresponding evaluator feedback:<br>{memory_context}<br><br>Now it's your turn to respond to the current step.<br>You should first conduct a reasoning process, which MUST be enclosed within <think> </think> tags.<br>After finishing your reasoning, suggest a valid molecule in SMILES format that either extends a previously successful modification or explores a new promising direction expected to satisfy the given constraints.<br>If you are ready to provide the self-contained solution, provide the molecule only inside <answer> ... </answer>. |

Table 8: Prompt template for subsequent turns (multiple-property optimization).

| |
|---|
| You are an expert medicinal chemist specializing in multi-property molecular optimization. Your goal is to propose a new molecule that satisfies the given constraints across all target properties. Ensure the proposed molecule in SMILES format is valid, plausible, and does not duplicate previous molecules. Prefer small edits that preserve chemical similarity.<br>Your task:<br>{task_description}<br><br>Prior to this step, you have already taken {step_count} step(s).<br>Below is the interaction history, where <SMILES> </SMILES> wrapped your past molecules and <results> </results> wrapped the corresponding evaluator feedback:<br>{memory_context}<br><br>Now it's your turn to respond to the current step.<br>You should first conduct a reasoning process, which MUST be enclosed within <think> </think> tags.<br>After finishing your reasoning, suggest a valid molecule in SMILES format that either extends a previously successful modification or explores a new promising direction expected to satisfy the given constraints.<br>If you are ready to provide the self-contained solution, provide the molecule only inside <SMILES> ... </SMILES>. |

properties to maximize and $-1$ for properties to minimize. SR is computed as:

$$\mathrm{SR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[\mathrm{sim}(m_i, m_i') \geq 0.4 \ \wedge \ \bigwedge_{j=1}^{n} \Big(\mathrm{sgn}(w_j)\cdot \Delta P_{i,j} > 0\Big)\right], \tag{16}$$

where $N$ is the test set size, $m_i$ is the $i$-th molecule, $m_i'$ is its optimized variant, and $n$ is the number of target properties.

**Similarity (Sim).** Sim is the average Tanimoto similarity across all test cases:

$$\mathrm{Sim} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{sim}(m_i, m_i'). \tag{17}$$

For failed optimizations where no valid molecule meeting the constraints is found, we set $m_i' = m_i$, so $\mathrm{sim}(m_i, m_i') = 1.0$.

**Relative Improvement (RI).** RI is the average relative improvement across target properties, computed per test case as:

$$\mathrm{RI} = \frac{1}{N} \sum_{j=1}^{N} \mathrm{sgn}(w_j) \cdot \frac{F_j(m') - F_j(m)}{|F_j(m)|}. \tag{18}$$

The denominator $|F_j(m)|$ accommodates properties that can take negative values. Failed optimizations yield $\mathrm{RI} = 0$.

### E.3 Molecular Metrics.

We employ the following pharmacological metrics for the molecular optimization tasks:

- `QED` (Quantitative Estimation of Drug-likeness) (Bickerton et al., 2012): A composite drug-likeness score that summarizes how well a molecule matches common medicinal-chemistry priors.
- `LogP` (lipophilicity) (Lipinski et al., 1997): The octanol/water partition coefficient. Larger LogP indicates higher lipophilicity.
- `plogP` (penalized logP): A modified LogP objective that subtracts penalties related to synthetic accessibility and undesirable ring structures (e.g., overly large rings), encouraging molecules that are both lipophilic and chemically plausible.
- `MR` (molar refractivity) (Le Fevre, 1965): A physicochemical descriptor associated with molecular volume and electronic polarizability, commonly used as a proxy for how strongly a molecule may interact with its environment.
- `BBBP` (blood-brain barrier permeability) (Wu et al., 2023): A measure of whether a molecule can cross the blood-brain barrier.
- `DRD2` (dopamine receptor D2 binding affinity) (Fan et al., 2020): An indicator of a molecule's binding strength to the D2 dopamine receptor. Higher affinity implies stronger receptor engagement.