Learning to Evolve: Open-ended Molecular Optimization with Progress-shaped RL

Xuan Li [1] **Zhanke Zhou** [1] **Zongze Li** [1] **Jiangchao Yao** [2] **Tongliang Liu** [3] **Bo Han** [1]

Abstract

Open-ended molecular optimization requires models that learn reusable refinement strategies from evaluator feedback, rather than merely expanding test-time search. Yet training-free optimization leaves the model unchanged, while outcome-only reinforcement learning with verifiable rewards (RLVR) collapses a *multi-turn* refinement trajectory into a *single* reward, obscuring which edits truly improve molecular quality under validity constraints. To turn molecular evolution trajectories into policy-improving supervision, we introduce *Learning to Evolve* (L2E), a long-horizon RL framework that extends RLVR from verifying final outputs to learning refinement dynamics. Instead of assigning one outcome reward to an entire generation, L2E constructs *turn-specific evolution advantages* from validity-gated evaluator feedback, coupling local edit progress with cumulative trajectory utility across sibling refinement chains. This turns intermediate molecular states into learnable supervision, enabling policy optimization over *how candidates evolve*, without critic models, reference trajectories, or preference labels. Empirically, L2E delivers strong gains across molecular optimization benchmarks. On TOMG-Bench, it achieves 19.36 RI on LogP and 4.14 RI on MR, outperforming GRPO by 7.8× and 7.6×, respectively. On multi-property BDP optimization, L2E reaches 4.11 RI on seen instructions and 3.63 RI on unseen ones, improving over the strongest RLVR baselines by 2.8× and 3.6×. These results show that learning credit over evolution trajectories is a key step toward scalable, policy-improving molecular discovery.

[1]TMLR Group, Department of Computer Science, Hong Kong Baptist University [2]Cooperative Medianet Innovation Center, Shanghai Jiao Tong University [3]Sydney AI Centre, The University of Sydney. Correspondence to: Bo Han <bhanml@comp.hkbu.edu.hk>.

The 3^{rd} *AI for Math Workshop* at the 43^{rd} International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.

1. Introduction

Molecular optimization is a central problem in drug discovery and materials design, where small structural edits can substantially alter biological activity and physicochemical properties. Open-ended molecular optimization has shifted from one-shot generation to evaluator-guided evolution: a model proposes a molecule, receives evaluator feedback, and refines the candidate over multiple turns. Large language models (LLMs) are natural generators for this loop, but most refinement systems remain limited to test-time search: they produce rich evolution trajectories yet do not convert them into policy updates (Novikov et al., 2025; Gottweis et al., 2025; Gao et al., 2025). The key objective is to transform molecular evolution trajectories into supervision that updates the generator and improves future refinement.

This objective gives rise to a long-horizon credit assignment problem on the evolution trajectories. Here, a single trajectory contains multiple edits that improve a target property, preserve the scaffold without making progress, or violate validity and similarity constraints. Training-free refinement exposes these mixed-quality updates but cannot learn from them (Gao et al., 2025; Lange et al., 2025). Reinforcement learning with verifiable rewards (RLVR) can update the policy from evaluator scores (Guo et al., 2025a; Kumar et al., 2025), but standard RLVR is designed for single-step verification, where a completed output receives one reward. In multi-turn refinement, this collapses the entire evolution trajectory into a single signal, assigning the same credit to productive, redundant, and invalid edits alike (Guo et al., 2025b; Zhang et al., 2025d). Single-step RLVR asks whether the final molecule is good; however, open-ended evolution must learn which decisions made it better.

We introduce *Learning to Evolve* (L2E), a long-horizon RL framework that extends RLVR from verifying final outputs to learning evolution dynamics. For each task and initial molecule, L2E samples multiple evolution trajectories from the same anchor. At every turn, the policy proposes a molecular edit, the evaluator returns validity-gated quality feedback, and the resulting molecule becomes both an evaluated output and the state for future evolution. Rather than treating only the final molecule as supervision, L2E treats the entire evolution trajectory as a structured learning object.

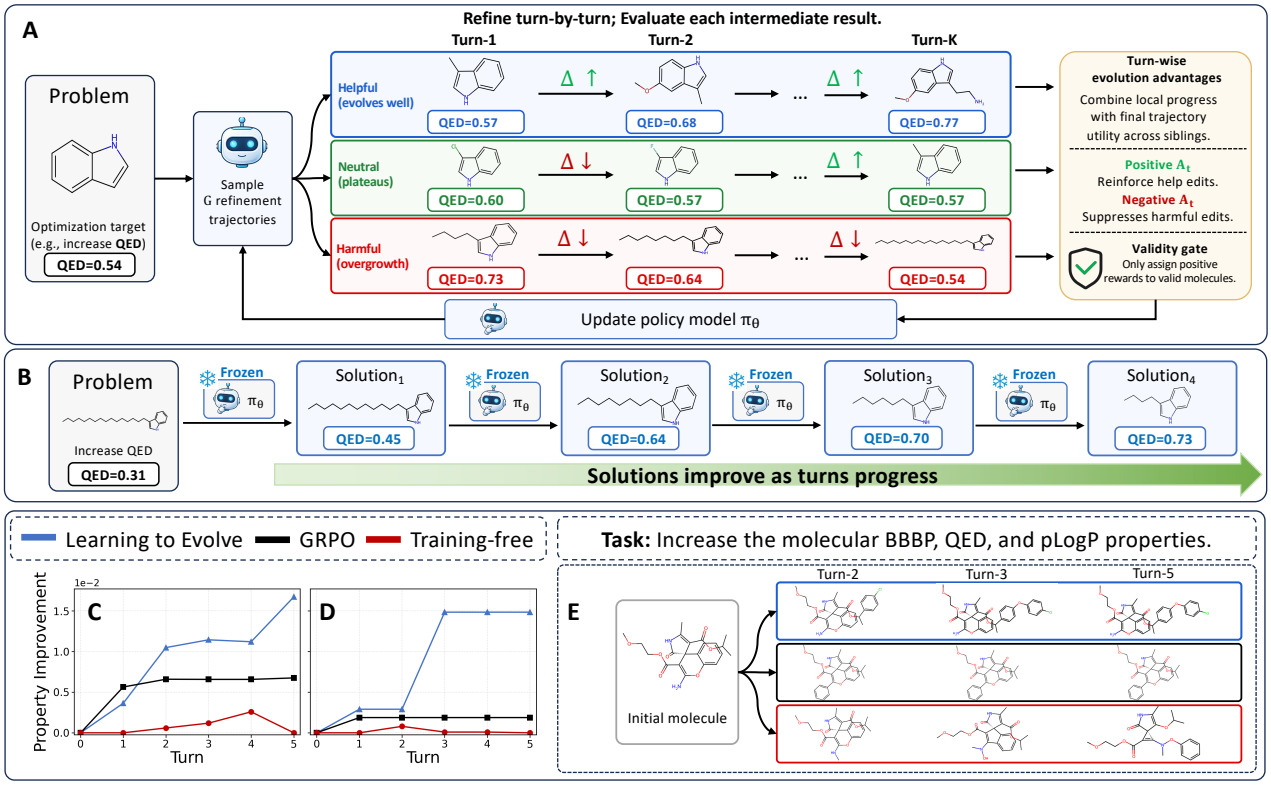

Figure 1. **Overview of Learning to Evolve (L2E). (A)** L2E trains on sibling refinement trajectories, using validity-gated evaluator scores to assign turn-wise evolution advantages. **(B)** At test time, the frozen refiner iteratively improves molecules across turns. **(C–D)** L2E achieves larger property gains over evolution turns on average and on a representative multi-property task. **(E)** Example trajectories show scaffold-preserving refinements by L2E and less consistent edits by baselines.

The key idea is to convert this trajectory into a *turn-specific* evolution advantage. L2E separates two credit questions that outcome-level RLVR conflates: Which *edits* drive progress along a trajectory, and which *trajectories* yield the greatest improvement from the same starting molecule. The first question identifies *locally* productive, redundant, invalid, or destabilizing turns; the second distinguishes genuinely *successful* evolution strategies from locally good edits inside weak trajectories. By coupling these two forms of evidence into a single evolution credit, L2E optimizes the policy over how candidates improve across time, without a critic model, reference trajectories, or preference labels. The core contribution lies not in designing a new policy gradient optimizer, but in converting evaluator feedback into reusable credit signals for learning long-horizon molecular evolution.

We evaluate L2E on molecular optimization benchmarks with automated scoring, validity constraints, and single- and multi-property objectives. On TOMG-Bench, L2E achieves 19.36 relative improvement (RI) on lipophilicity (LogP) and 4.14 RI on molar refractivity (MR), outperforming GRPO by 7.8× and 7.6×, respectively. On multi-property optimization, L2E reaches up to 4.11 RI on seen instruction combinations and 3.63 RI on unseen combinations, improving over the strongest RLVR baselines by 2.8× and 3.6×. Ablations show that both local turn credit and trajectory-level utility are necessary, while horizon-scaling results show that L2E continues to benefit from longer evolution trajectories instead of plateauing. Our contributions are summarized as the following:

- We formulate open-ended molecular optimization as long-horizon evaluator-guided evolution, highlighting the mismatch between multi-turn evolution and outcome-only, single-step RLVR.
- We propose Learning to Evolve (L2E), an RL framework that transforms validity-gated evaluator feedback into turn-specific evolution advantages over evolution trajectories.
- We justify that learning credit over evolution trajectories improves single- and multi-property molecular optimization under matched evaluator budgets, with gains that generalize to unseen instruction combinations and longer evolution horizons.

2. Preliminaries

2.1. Problem Formulation of Multi-turn Evolution

We formulate multi-turn evolution as iterative refinement guided by an evaluator. Let $\mathcal{X}$ denote the space of tasks

and $\mathcal{Y}$ the space of candidate solutions. Each instance is a task-anchor pair $(x, y_0) \in \mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$, where x specifies the optimization objective and y_0 is a valid initial solution.

Evaluator. The evaluator maps a task-candidate pair to a validity label and a scalar quality score:

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

Here $v(x,y) \in \{0,1\}$ indicates whether y is valid for task x, and $q(x,y) \in \mathbb{R}$ measures its task-specific quality. In molecular optimization, v checks chemical validity and q measures the target property.

Trajectory generation. Starting from the valid anchor y_0, the policy π_θ generates a K-turn refinement trajectory through a *refine-evaluate-update* loop. Let $f_0 = \mathcal{E}(x, y_0)$ and initialize the evaluated history as $h_0 = [(y_0, f_0)]$. Since the history grows with the number of turns, we condition the policy on a bounded memory containing the most recent M evaluated candidates:

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

At each turn $t \in \{1, \ldots, K\}$, the policy uses the task and memory to propose a new candidate, the evaluator scores it, and the evaluated candidate is appended to the history:

$$y_t \sim \pi_\theta(\cdot \mid x; \mathrm{Mem}(h_{t-1})), \tag{2}$$

$$f_t = \mathcal{E}(x, y_t),\ \ h_t = h_{t-1} \| [(y_t, f_t)], \tag{3}$$

where $\|$ denotes concatenation. Each refinement is scored before being added to the history, allowing later turns to condition on previous molecules and evaluator feedback.

Selection objective. After K refinement turns, the candidate pool is $\mathcal{S} = \{y_0, y_1, \ldots, y_K\}$. Let $\mathcal{V}(x) = \{y \in \mathcal{Y} : v(x,y) = 1\}$ denote the set of valid candidates for task x. At inference time, we select the highest-quality valid candidate from the trajectory:

$$\hat{y} = \operatorname*{arg\,max}_{y \in \mathcal{S} \cap \mathcal{V}(x)} q(x,y). \tag{4}$$

The feasible set is non-empty because the anchor y_0 is valid by assumption. The goal is to generate trajectories that contain valid candidates with higher quality than the anchor.

3. Learning to Evolve

L2E trains a refinement policy by converting evaluator feedback from molecular evolution trajectories into turn-specific credit. The single-trajectory generation process is defined in Sec. 2.1. Here we describe how L2E samples anchor-conditioned rollouts, constructs evolution credit from their feedback, and optimizes the policy.

3.1. Anchor-Conditioned Evolution Rollouts

Independent rollouts from the same anchor. For each task-anchor pair (x, y_0), L2E samples G independent K-turn refinement trajectories from the behavior policy $\pi_{\theta_{\mathrm{old}}}$:

$$\mathcal{T}^{(i)} = \big(y_1^{(i)}, f_1^{(i)}, \ldots, y_K^{(i)}, f_K^{(i)}\big), \qquad i = 1, \ldots, G. \tag{5}$$

All trajectories start from the same task and anchor molecule, but diverge through stochastic refinement decisions. This creates an *anchor-conditioned comparison set*, where differences in evaluated quality reflect different refinement choices rather than different starting conditions.

Evaluator feedback and molecular quality. At turn t of trajectory i, the evaluator is applied to molecule $y_t^{(i)}$:

$$f_t^{(i)} = \mathcal{E}(x, y_t^{(i)}) = \big(v_t^{(i)}, q_t^{(i)}\big), \tag{6}$$

$$v_t^{(i)} \equiv v(x, y_t^{(i)}), q_t^{(i)} \equiv q(x, y_t^{(i)}). \tag{7}$$

Here $v_t^{(i)} \in \{0,1\}$ indicates chemical validity, and $q_t^{(i)} \in \mathbb{R}$ is the quality score used for task x, which combines target-property satisfaction with a soft similarity penalty to the anchor molecule:

$$q(x,y) = q_{\mathrm{prop}}(y; P^\star) - \lambda r_{\mathrm{sim}}(y; y_0, \delta), \tag{8}$$

$$\text{where}\quad q_{\mathrm{prop}}(y; P^\star) = \sum_{n=1}^{N} w_n\, g_n\big(P_n(y), P_n^\star\big), \tag{9}$$

$$r_{\mathrm{sim}}(y; y_0, \delta) = \max\big(0, \delta - \mathrm{Sim}(y, y_0)\big). \tag{10}$$

Here $P^\star$ is specified by task x and denotes the desired property directions, g_n is the signed transform for property P_n, w_n is the property weight, Sim is Tanimoto similarity over molecular fingerprints, and λ controls the similarity penalty. We set $\delta = 0.4$ as the similarity threshold.

Comparison beyond outcome-level RLVR. Standard outcome-level RLVR reduces each trajectory to a single scalar reward and assigns the resulting advantage to every turn in that trajectory. This compares complete rollouts, but it cannot determine which intermediate refinements caused improvement. L2E instead preserves all intermediate evaluations. The resulting $G \times K$ feedback structure supports two comparisons: within a trajectory, turns can be compared to identify locally productive refinements; across independent trajectories from the same anchor, rollouts can be compared to identify evolution chains that sustain higher quality under the same initial condition.

3.2. Evolution Credit from Trajectory Feedback

Validity-gated feedback. L2E first converts evaluator feedback into a score that respects feasibility. For turn t in trajectory i, we define

$$S_t^{(i)} = \mathbb{I}\Big[v_t^{(i)} = 1\Big] \cdot q_t^{(i)} = \mathbb{I}\Big[v(x, y_t^{(i)}) = 1\Big] \cdot q(x, y_t^{(i)}). \tag{11}$$

Thus, property quality contributes to learning only when the generated molecule is valid. This is essential in molecular evolution, where an edit can improve a target property numerically while producing an infeasible molecule.

Trajectory-feedback matrix. For a fixed anchor (x, y_0), the G independent rollouts of length K produce an anchor-conditioned feedback matrix:

$$\mathbf{S}(x, y_0) = \begin{bmatrix} S_1^{(1)} & \cdots & S_K^{(1)} \\ \vdots & \ddots & \vdots \\ S_1^{(G)} & \cdots & S_K^{(G)} \end{bmatrix} \in \mathbb{R}^{G \times K}. \tag{12}$$

This matrix is the structured supervision object used by L2E. Outcome-level RLVR compresses each row into a single scalar reward, removing information about where improvement occurred. L2E preserves the full matrix: the *temporal* dimension records how quality changes within a trajectory, while the *trajectory* dimension compares evolution strategies from the same starting molecule.

Two credit questions. The feedback matrix reveals two credit questions that cannot be answered by a single outcome reward. The first question is *within-trajectory*: which *edits* produce locally useful molecular states, and which are redundant, invalid, or destabilizing? The second question is *across-trajectory*: which *evolution trajectories* sustain higher quality from the same anchor? These questions are complementary but non-substitutable. Temporal comparison locates useful turns but cannot determine whether the full trajectory is strong, while trajectory comparison ranks evolution strategies but cannot identify the edits responsible for their success. L2E addresses these two questions through temporal credit and trajectory credit, respectively.

Temporal credit. To answer the first question, L2E contrasts each turn against other turns:

$$C_{t,\text{turn}}^{(i)} = \frac{S_t^{(i)} - \frac{1}{K}\sum_{s=1}^{K} S_s^{(i)}}{\text{std}_s\left(S_s^{(i)}\right) + \epsilon_{\text{std}}}. \tag{13}$$

This term assigns positive credit to turns whose resulting molecules are above the trajectory average and negative credit to turns that underperform relative to the same evolution chain. It gives the learning signal temporal resolution: edits in the same rollout no longer share identical credit.

Trajectory credit. To answer the second question, L2E measures the utility of each trajectory by cumulative quality:

$$U(\mathcal{T}^{(i)}) = \sum_{t=1}^{K} S_t^{(i)}. \tag{14}$$

The trajectory utilities are then normalized across the G independent rollouts from the same anchor:

$$C_{\text{traj}}^{(i)} = \frac{U(\mathcal{T}^{(i)}) - \frac{1}{G}\sum_{j=1}^{G} U(\mathcal{T}^{(j)})}{\text{std}_j\left(U(\mathcal{T}^{(j)})\right) + \epsilon_{\text{std}}}. \tag{15}$$

Because this comparison is performed within an anchor-conditioned group, it controls for task difficulty, initial molecular quality, and evaluator scale. This term rewards evolution strategies that sustain high-quality candidates over the refinement horizon.

Turn-specific evolution advantage. L2E couples the two credit signals into a turn-specific advantage:

$$A_t^{(i)} = C_{\text{traj}}^{(i)} + \omega C_{t,\text{turn}}^{(i)}, \qquad \omega \geq 0. \tag{16}$$

The trajectory term asks whether the rollout is a successful evolution strategy; the temporal term asks whether the current edit is useful within that strategy. Their combination assigns credit to each refinement according to both its local effect and its role in a successful long-horizon trajectory. This is the key distinction from outcome-level RLVR: instead of broadcasting one reward to all turns, L2E turns intermediate evaluator feedback into reusable supervision for learning how molecules evolve.

3.3. Policy Optimization with Evolution Advantages

Objective. Given $A_t^{(i)}$, L2E applies a standard token-wise clipped policy-gradient objective:

$$\mathcal{J}(\theta) = \mathbb{E}\Bigg[\frac{1}{GK}\sum_{i=1}^{G}\sum_{t=1}^{K}\frac{1}{|y_t^{(i)}|}\sum_{k=1}^{|y_t^{(i)}|}\min\Big(\rho_{t,k}^{(i)}A_t^{(i)}, \text{clip}\Big(\rho_{t,k}^{(i)}, 1-\epsilon, 1+\epsilon\Big)A_t^{(i)}\Big)\Bigg] - \beta\,\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\,. \tag{17}$$

$$\text{where} \quad \rho_{t,k}^{(i)} = \frac{\pi_\theta\Big(y_{t,k}^{(i)} \mid x; \text{Mem}(h_{t-1}^{(i)}), y_{t,<k}^{(i)}\Big)}{\pi_{\theta_{\text{old}}}\Big(y_{t,k}^{(i)} \mid x; \text{Mem}(h_{t-1}^{(i)}), y_{t,<k}^{(i)}\Big)}.$$

Here $y_{t,k}^{(i)}$ is the k-th token of the molecule generated at turn t, $y_{t,<k}^{(i)}$ is its prefix, ϵ is the clipping threshold, β controls KL regularization, and π_{ref} is the reference policy.

Optimizer-agnostic learning signal. The optimizer is standard; the learning signal is the contribution. Outcome-level RLVR assigns one normalized advantage to every token in a rollout, whereas L2E assigns a distinct advantage to each refinement turn using the contrasts in $\mathbf{S}(x, y_0)$. This converts evaluator feedback into reusable credit for learning long-horizon molecular evolution.

4. Experiments

In this section, we evaluate the performance of our proposed method, L2E. We outline the experimental setup in Sec. 4.1 and report the main results in Sec. 4.2. We then ablate the design choices and compare against step-level and training-free baselines in Sec. 4.3, and test generalization across model scales and evaluator families in Sec. 4.4.

Table 1. Single-property molecular optimization. Mean performance over 3 independent seeds, best-of-$N=5$ inference chains. **Bold**: best per ranked column (Vld, SR, RI). Sim is reported for context and not ranked across methods.

Model	QED				LogP				MR			
	Vld↑	SR↑	Sim	RI↑	Vld↑	SR↑	Sim	RI↑	Vld↑	SR↑	Sim	RI↑
Evolutionary Algorithms												
GraphGA	**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	**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 (no training)												
Qwen2.5-3B	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	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	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	0.677	0.427	0.853	0.063	0.776	0.625	0.777	0.471	0.753	0.593	0.782	0.076
RL Fine-tuning (Qwen2.5-3B)												
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
L2E (Ours)	**1.000**	0.410	0.918	**0.386**	**1.000**	**1.000**	0.768	**19.357**	0.980	**0.697**	0.740	**4.138**

Table 2. Multi-property optimization (SR↑ / RI↑). **Seen**: instruction combinations present during training. **Unseen**: held-out combinations never seen during training. Full validity and similarity breakdowns are in Tab. 11.

Model	Seen Instructions						Unseen Instructions					
	BDP		BDQ		BPQ		BDP		BDQ		BPQ	
	SR	RI	SR	RI	SR	RI	SR	RI	SR	RI	SR	RI
Evolutionary Algorithms												
GraphGA	0.134	0.343	0.118	**0.653**	0.059	**0.183**	0.134	0.343	0.118	**0.653**	0.059	0.183
REINVENT	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
MOLLEO	0.082	0.085	0.020	0.008	0.030	0.181	0.052	0.162	0.039	0.023	0.020	0.019
General-Purpose LLMs												
Qwen2.5-3B	0.100	0.171	0.127	0.068	0.033	0.010	0.103	0.187	0.111	0.085	0.056	0.026
Qwen3-4B	0.010	0.010	0.023	0.004	0.000	0.000	0.062	0.113	0.078	0.077	0.003	0.001
Qwen2.5-7B	0.351	0.503	**0.284**	0.276	0.139	0.080	0.237	0.387	**0.248**	0.176	0.162	0.058
Llama3.1-8B	**0.381**	0.834	0.229	0.189	0.155	0.050	0.320	2.511	0.219	0.185	**0.168**	0.066
RL Fine-tuning (Qwen2.5-3B)												
GRPO	0.196	0.499	0.059	0.203	0.076	0.174	0.247	1.014	0.052	0.056	0.099	**0.213**
RLOO	0.282	1.461	0.160	0.294	**0.188**	0.141	0.237	0.842	0.147	0.323	0.132	0.108
L2E (Ours)	0.347	**4.111**	0.157	0.254	0.139	0.154	**0.323**	**3.627**	0.137	0.158	0.125	0.140

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 B.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 LLMs' ability to modify molecular structures to meet specific property constraints.

Baselines. We compare L2E with three method groups: (1) Molecule optimization baselines: GraphGA (Jensen, 2019), REINVENT (Olivecrona et al., 2017), and MOLLEO (Wang et al., 2025a); (2) General-purpose LLMs: 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 configurations from PMO (Gao et al., 2022). For the LLM-assisted evolution method, MOLLEO, we use Qwen-2.5-3B as the base model for a fair comparison. All LLM baselines and L2E use $K=5$ turns. We report the best-scoring molecule among the generated solutions. All RL methods share Qwen2.5-3B with the same training budgets.

Evaluation Metrics. Following prior work (Dey et al., 2025), we report four metrics: 1) *Validity* (Vld), the fraction of best-scoring generated molecules that are chemically valid (parsed by RDKit (Landrum et al., 2025)); 2) *Success Rate* (SR), the fraction of best-scoring molecules that improve all properties; 3) *Similarity* (Sim), the mean Tanimoto similarity between proposed and initial molecules computed on Morgan fingerprints (failed optimizations return y_0 and contribute 1.0); 4) *Relative Improvement* (RI), the mean gain of the target property relative to its initial value at y_0. Vld, SR, and RI are higher-is-better and ranked across methods; Sim is reported for context and is not ranked.

Table 3. **Left:** ablation of advantage levels. **Right:** ω sensitivity sweep. **Bold** marks per-column maximum; † flags RI values inflated by structural collapse, not genuine progress.

Variant	LogP			QED			MR		
	SR	Sim	RI	SR	Sim	RI	SR	Sim	RI
A_{inter} only (ω=0)	0.757	0.788	13.724	**0.570**	0.863	**0.409**	0.650	0.812	3.450
A_{intra} only	0.920	0.678	**157.305**†	0.480	0.676	0.387	**0.910**	0.502	**13.135**†
L2E (full, ω=1)	**1.000**	0.768	19.357	0.410	0.918	0.386	0.697	0.740	4.138

ω	LogP		QED		MR	
	SR	RI	SR	RI	SR	RI
0	0.757	13.724	**0.570**	**0.409**	0.650	3.450
1	**1.000**	**19.357**	0.410	0.386	**0.697**	**4.138**
5	0.860	14.673	0.410	0.380	0.563	2.815

Table 4. Single-property optimization with the Qwen2.5-7B-Instruct, averaged over 3 seeds.

Model	QED				LogP				MR			
	Vld↑	SR↑	Sim	RI↑	Vld↑	SR↑	Sim	RI↑	Vld↑	SR↑	Sim	RI↑
General-Purpose LLM (no training)												
Qwen2.5-7B	0.750	0.570	0.796	0.094	0.810	0.682	0.763	0.534	0.777	0.587	0.772	0.116
RL Fine-tuning (Qwen2.5-7B backbone)												
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
L2E (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**

4.2. Quantitative Results

Single-property optimization. Tab. 1: L2E achieves RI = 19.357 on LogP and 4.138 on MR, outperforming training-free methods (RI $\leq$ 0.534) and RLVR baselines by a wide RI margin (LogP RI $\approx 2\times$ RLOO; MR RI $\approx 1.7\times$). The gap is largest on LogP and MR, both unbounded targets where multi-turn refinement can compound across many turns; on QED, which is bounded in $[0, 1]$ and saturates within a few productive edits, L2E ties RLOO (RI 0.386 vs. 0.378) instead of outpacing it. We read this as the long-horizon credit signal paying off precisely when the evaluator admits cumulative progress, and degrading to a tie when it does not. L2E also trades a small amount of similarity (Sim 0.768 vs. RLOO 0.812 on LogP) for a $2\times$ RI gain while staying well above the 0.4 Tanimoto contract, indicating that the turn-level advantage explores beyond conservative edits without breaking validity. SR matches RLOO on LogP (1.000 vs. 0.993) and exceeds all baselines on MR (0.697), so the RI gains are not bought by inflated invalid candidates.

Multi-property optimization. Tab. 2: L2E leads on BDP (RI,=,4.111 seen, 3.627 unseen, both top of column), the one combination whose property set (BBBP+DRD2+pLogP) contains an unbounded target. On BDQ and BPQ, which both include QED, L2E remains close to the strongest RL fine-tuning baselines (RLOO, GRPO) but does not surpass the evolutionary-algorithm peak; we attribute this to the same ceiling effect observed on single-property QED, which removes the long-horizon advantage L2E exploits on unbounded objectives. Crucially, the BDP advantage is preserved on the held-out (unseen) instruction split, where L2E remains first by a wide margin, indicating that the cumulative-credit behavior generalizes across instruction phrasings rather than memorizing seen combinations.

4.3. Further Studies

Advantage decomposition and ω sensitivity. Tab. 3 (left) ablates the two advantage levels: *inter-only* (ω=0) loses turn-level precision (LogP SR drops to 0.757); *intra-only* maximizes per-step deviation but causes structural collapse (Sim 0.502 to 0.678 across properties), inflating RI on LogP (157.3†) and MR (13.1†) with molecules that violate the Tanimoto similarity constraint. Full L2E (ω=1) is the only setting that preserves SR, Sim, and RI on all three properties. Tab. 3 (right) sweeps $\omega \in 0, 1, 5$: ω=5 over-weights the intra term and degrades SR; default ω=1 gives the best SR/RI balance. Read together, the three rows trace a credit-allocation frontier: inter-only preserves the scaffold but cannot drive progress, intra-only drives progress but loses the scaffold, and the dual-level coupling at ω=1 is the only setting non-dominated on the (SR, Sim, RI).

Reward shaping. Fig. 3 compares L2E (anchor-to-y_0) against delta-from-previous ($S(y_t) - S(y_{t-1})$) and best-so-far ($S(y_t) - \max_{i<t} S(y_i)$). Delta-from-previous collapses to zero on plateau turns; best-so-far decays once the policy reaches a peak, removing gradient at long-horizon moments. Anchor-to-y_0 stays dense across the trajectory, and the empirical ranking matches this argument.

Comparisons with Step-level and Training-free Baselines. Tab. 5 (left) compares L2E with POLO, a training-based step-level RL baseline, and GEPA, a training-free search baseline using 20 proposals per step. L2E achieves the highest RI on every property: 19.357 on LogP versus 11.268 for POLO and 0.652 for GEPA, and 4.138 on MR versus 3.245 for POLO and 0.187 for GEPA. These gaps separate two effects. The large GEPA-to-POLO jump on LogP, about $17\times$, shows that policy training is essential relative to training-free proposal search. The smaller but consistent

Table 5. **Left:** L2E vs. training-free GEPA and step-level POLO under matched compute (Qwen2.5-3B-Instruct backbone, same oracle budget). **Middle:** generalization to protein–ligand docking on dockstring EGFR (lower kcal/mol = stronger binding); L2E applied *without retuning*. **Right:** top-10% slice on TOMG-Bench (SR_{10}, RI_{10} over the highest-scoring decile per input). **Bold**: best per ranked column.

Method	LogP			QED			MR		
	SR	Sim	RI	SR	Sim	RI	SR	Sim	RI
GEPA	0.180	0.621	0.652	0.110	0.587	0.041	0.130	0.564	0.187
POLO	0.913	0.762	11.268	0.387	0.901	0.340	0.633	0.733	3.245
L2E	**1.000**	0.768	**19.357**	**0.410**	0.918	**0.386**	**0.697**	0.740	**4.138**

Method	Mean Score↓	Top-1 Score↓
GRPO	−10.0	−11.4
L2E	−10.3	−11.8

Method	LogP		QED		MR	
	SR_{10}	RI_{10}	SR_{10}	RI_{10}	SR_{10}	RI_{10}
GRPO	**1.000**	36.41	**1.000**	0.798	**1.000**	5.92
RLOO	**1.000**	39.83	**1.000**	0.812	**1.000**	7.18
L2E	**1.000**	**45.67**	**1.000**	**0.815**	**1.000**	**8.43**

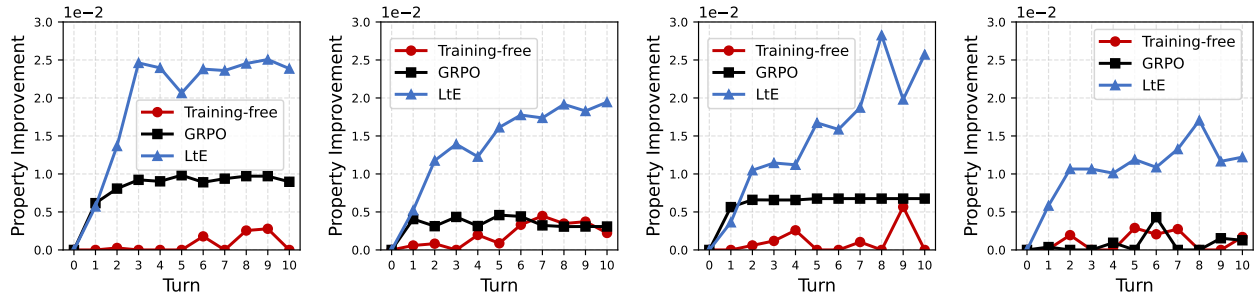

Figure 2. Scaling the number of evolution rounds on multi-property optimization with seen instructions (BDQ, BDP, BPQ) and unseen instructions (BDQ), respectively. Training-free approaches yield only small, inconsistent gains. GRPO delivers early improvements but quickly plateaus, whereas L2E continues to improve steadily as additional evolution turns are introduced.

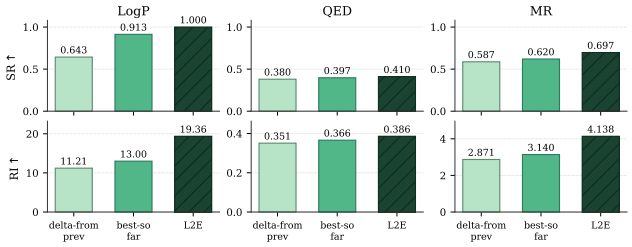

Figure 3. Reward-shaping ablation under identical dual-level advantage (ω=1). **Top row:** success rate (SR↑). **Bottom row:** relative improvement (RI↑).

POLO-to-L2E jump, about 1.7× on LogP, 1.3× on MR, and 1.14× on QED, isolates the added value of trajectory-level credit assignment beyond step-level RL. Thus, L2E does not merely recover the gains of training-based optimization; it further improves how refinement trajectories are learned.

4.4. Generalization

Robustness on a larger model scale. With a 7B backbone (Tab. 4), L2E achieves LogP RI 19.899 (≈ 2× RLOO's 10.058) and MR RI 6.055 (≈ 2.3× RLOO's 2.603), confirming that the trajectory-credit signal scales on unbounded properties. The QED gain is modest (0.386→0.422), consistent with the bounded ceiling already observed at 3B. LogP SR drops from 1.000 (3B) to 0.843 (7B) while RI doubles, reflecting an expanded action space: more aggressive edits trade strict-threshold SR for stronger top-end gains.

Robustness checks. Tab. 5 (middle) tests cross-evaluator generalization on dockstring (García-Ortegón et al., 2022) EGFR: L2E improves the mean docking score from −10.0 to −10.3 kcal/mol and top-1 from −11.4 to −11.8 kcal/mol, zero-shot at the evaluator level (the policy was trained on TOMG-Bench analytic scorers, never on AutoDock Vina). Tab. 5 (right) tests robustness across the metric distribution: on the top-10% slice per input, L2E retains its RI lead on every property (45.67 vs. 39.83 RLOO on LogP; 8.43 vs. 7.18 on MR), ruling out an outlier-driven mean.

Extending the Evolution Turns. We study how performance scales with a longer refinement horizon by increasing the number of turns. Fig. 2 shows that 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 plateaus. In contrast, L2E accumulate progress as additional turns are introduced, enlarging the gap over baselines.

5. Conclusion

In this paper, we introduce Learning to Evolve (L2E), a learning-centric framework that trains multi-turn refinement via turn- and trajectory-level credit assignment. L2E 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, L2E achieves larger relative improvements, outperforming training-free and RLVR baselines under the same budgets. Moreover, L2E continues to improve as the evolution horizon grows. More broadly, L2E provides a general recipe for training LLMs to *evolve* solutions with evaluator feedback. Future work includes extending L2E beyond molecular optimization to other open-ended problems to further enhance scientific discovery with verifiable objectives.

Impact Statement

This paper presents Learning to Evolve (L2E), 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. L2E 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.

References

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

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

Bai, L., Cai, Z., Cao, Y., Cao, M., Cao, W., Chen, C., Chen, H., Chen, K., Chen, P., Chen, Y., et al. Intern-s1: A scientific multimodal foundation model. *arXiv preprint arXiv:2508.15763*, 2025.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

Chai, J., Tang, S., Ye, R., Du, Y., Zhu, X., Zhou, M., Wang, Y., E, W., Zhang, Y., Zhang, L., and Chen, S. 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.

Chen, J., Zhang, B., Ma, R., Wang, P., Liang, X., Tu, Z., Li, X., and Wong, K.-Y. K. Spc: Evolving self-play critic via adversarial games for llm reasoning. *arXiv preprint arXiv:2504.19162*, 2025.

Chen, X., Lin, M., Sch"arli, N., and Zhou, D. Teaching large language models to self-debug. In *ICLR*, 2024.

Chithrananda, S., Grand, G., and Ramsundar, B. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

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

Fan, L., Tan, L., Chen, Z., Qi, J., Nie, F., Luo, Z., Cheng, J., and Wang, S. Haloperidol bound d2 dopamine receptor structure inspired the discovery of subtype selective ligands. *Nature communications*, 11(1):1074, 2020.

Feng, L., Xue, Z., Liu, T., and An, B. Group-in-group policy optimization for llm agent training. In *NeurIPS*, 2025.

Gao, H.-a., Geng, J., Hua, W., Hu, M., Juan, X., Liu, H., Liu, S., Qiu, J., Qi, X., Wu, Y., et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

Gao, S., Shi, Z., Zhu, M., Fang, B., Xin, X., Ren, P., Chen, Z., Ma, J., and Ren, Z. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *AAAI*, 2024.

Gao, W., Fu, T., Sun, J., and Coley, C. Sample efficiency matters: a benchmark for practical molecular optimization. In *NeurIPS*, 2022.

García-Ortegón, M., Simm, G. N., Tripp, A. J., Hernández-Lobato, J. M., Bender, A., and Bacallado, S. Dockstring: easy molecular docking yields better benchmarks for ligand design. *Journal of chemical information and modeling*, 62(15):3486–3502, 2022.

Gottweis, J., Weng, W.-H., Daryin, A., Tu, T., Palepu, A., Sirkovic, P., Myaskovsky, A., Weissenberger, F., Rong, K., Tanno, R., et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025a.

Guo, Y., Xu, L., Liu, J., Ye, D., and Qiu, S. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *arXiv preprint arXiv:2505.23564*, 2025b.

Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. In *ICLR*, 2024.

Hubert, T., Mehta, R., Sartran, L., Horváth, M. Z., Žužić, G., Wieser, E., Huang, A., Schrittwieser, J., Schroecker, Y., Masoom, H., et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, pp. 1–3, 2025.

Irwin, R., Dimitriadis, S., He, J., and Bjerrum, E. J. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3(1):015022, 2022.

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

Korovina, K., Xu, S., Kandasamy, K., Neiswanger, W., Poczos, B., Schneider, J., and Xing, E. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, pp. 3393–3403. PMLR, 2020.

Kumar, K., Ashraf, T., Thawakar, O., Anwer, R. M., Cholakkal, H., Shah, M., Yang, M.-H., Torr, P. H., Khan, F. S., and Khan, S. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Landrum, G., Tosco, P., Kelley, B., Rodriguez, R., Cosgrove, D., Vianello, R., Gedeck, P., Jones, G., Kawashima, E., Nealschneider, D., et al. rdkit/rdkit: 2025_03_1 (q1 2025) release. *Zenodo*, 2025.

Lange, R. T., Imajuku, Y., and Cetin, E. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025.

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

Li, J., Li, J., Liu, Y., Zhou, D., and Li, Q. Tomg-bench: Evaluating llms on text-based open molecule generation. *arXiv preprint arXiv:2412.14642*, 2024.

Li, J., Zhang, D., Wang, X., Hao, Z., Lei, J., Tan, Q., Zhou, C., Liu, W., Yang, Y., Xiong, X., et al. Chemvlm: Exploring the power of multimodal large language models in chemistry area. In *AAAI*, 2025.

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

Lipinski, C. A., Lombardo, F., Dominy, B. W., and Feeney, P. J. 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.

Liu, S., Wang, J., Yang, Y., Wang, C., Liu, L., Guo, H., and Xiao, C. Conversational drug editing using retrieval and domain feedback. In *ICLR*, 2024a.

Liu, Z., Li, S., Luo, Y., Fei, H., Cao, Y., Kawaguchi, K., Wang, X., and Chua, T.-S. Molca: Molecular graph-language modeling with cross-modal projector and unimodal adapter. In *EMNLP*, 2023.

Liu, Z., Hoang, T. Q., Zhang, J., Zhu, M., Lan, T., Kokane, S., Tan, J., Yao, W., Liu, Z., Feng, Y., N, R. R., Yang, L., Savarese, S., Niebles, J. C., Wang, H., Heinecke, S., and Xiong, C. Apigen: Automated PIpeline for generating verifiable and diverse function-calling datasets. In *NeurIPS Datasets and Benchmarks Track*, 2024b.

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

López-Pérez, K., Avellaneda-Tamayo, J. F., Chen, L., López-López, E., Juárez-Mercado, K. E., Medina-Franco, J. L., and Miranda-Quintana, R. A. Molecular similarity: Theory, applications, and perspectives. *Artificial Intelligence Chemistry*, 2(2):100077, 2024.

Lu, P., Chen, B., Liu, S., Thapa, R., Boen, J., and Zou, J. Octotools: An agentic framework with extensible tools for complex reasoning. *arXiv preprint arXiv:2502.11271*, 2025.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*, 2023.

Narayanan, S. M., Braza, J. D., Griffiths, R.-R., Bou, A., Wellawatte, G., Ramos, M. C., Mitchener, L., Rodriques, S. G., and White, A. D. Training a scientific reasoning model for chemistry. In *NeurIPS*, 2025.

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

Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J., Mehrabian, A., et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

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

Pei, Q., Zhang, W., Zhu, J., Wu, K., Gao, K., Wu, L., Xia, Y., and Yan, R. Biot5: Enriching cross-modal integration in biology with chemical knowledge and natural language associations. In *EMNLP*, 2023.

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

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., dahai li, Liu, Z., and Sun, M. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *ICLR*, 2024.

Qiu, J., Qi, X., Zhang, T., Juan, X., Guo, J., Lu, Y., Wang, Y., Yao, Z., Ren, Q., Jiang, X., et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.

Qu, Y., Zhang, T., Garg, N., and Kumar, A. Recursive introspection: Teaching language model agents how to self-improve. In *NeurIPS*, 2024.

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J., Ellenberg, J. S., Wang, P., Fawzi, O., et al. Mathematical discoveries from program search with large language models. *Nature*, 625 (7995):468–475, 2024.

Shang, N., Liu, Y., Zhu, Y., Zhang, L. L., Xu, W., Guan, X., Zhang, B., Dong, B., Zhou, X., Zhang, B., et al. rstar2-agent: Agentic reasoning technical report. *arXiv preprint arXiv:2508.20722*, 2025.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. R., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Wang, H., Skreta, M., Ser, C.-T., Gao, W., Kong, L., Strieth-Kalthoff, F., Duan, C., Zhuang, Y., Yu, Y., Zhu, Y., et al. Efficient evolutionary search over chemical space with large language models. In *ICLR*, 2025a.

Wang, Z., Wang, K., Wang, Q., Zhang, P., Li, L., Yang, Z., Yu, K., Nguyen, M. N., Liu, L., Gottlieb, E., et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

Wang, Z., Wen, Y., Pattie, W., Luo, X., Wu, W., Hu, J. Y.-C., Pandey, A., Liu, H., and Ding, K. Polo: Preference-guided multi-turn reinforcement learning for lead optimization. *arXiv preprint arXiv:2509.21737*, 2025c.

Wu, D., Chen, Q., Chen, X., Han, F., Chen, Z., and Wang, Y. The blood–brain barrier: Structure, regulation and drug delivery. *Signal transduction and targeted therapy*, 8(1): 217, 2023.

Xia, Y., Jin, P., Xie, S., He, L., Cao, C., Luo, R., Liu, G., Wang, Y., Liu, Z., Chen, Y.-J., et al. Nature language model: Deciphering the language of nature for scientific discovery. *arXiv preprint arXiv:2502.07527*, 2025.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., and Chen, X. Large language models as optimizers. In *ICLR*, 2024b.

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

Zhang, D., Liu, W., Tan, Q., Chen, J., Yan, H., Yan, Y., Li, J., Huang, W., Yue, X., Ouyang, W., et al. Chem-llm: A chemical large language model. *arXiv preprint arXiv:2402.06852*, 2024.

Zhang, G., Geng, H., Yu, X., Yin, Z., Zhang, Z., Tan, Z., Zhou, H., Li, Z., Xue, X., Li, Y., et al. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025a.

Zhang, J., Hu, S., Lu, C., Lange, R., and Clune, J. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025b.

Zhang, Y., Ye, P., Yang, X., Feng, S., Zhang, S., Bai, L., Ouyang, W., and Hu, S. Nature-inspired population-based evolution of large language models. *arXiv preprint arXiv:2503.01155*, 2025c.

Zhang, Z., Chen, Z., Li, M., Tu, Z., and Li, X. Rlvmr: Reinforcement learning with verifiable meta-reasoning

rewards for robust long-horizon agents. *arXiv preprint arXiv:2507.22844*, 2025d.

Zhao, Z., Chen, B., Wan, Z., Chen, L., Lin, X., Yu, S., Zhang, S., Ma, D., Zhu, Z., Zhang, D., et al. Chemdfm-r: A chemical reasoning llm enhanced with atomized chemical knowledge. *arXiv preprint arXiv:2507.21990*, 2025a.

Zhao, Z., Ma, D., Chen, L., Sun, L., Li, Z., Xia, Y., Chen, B., Xu, H., Zhu, Z., Zhu, S., et al. Developing chemdfm as a large language foundation model for chemistry. *Cell Reports Physical Science*, 6(4), 2025b.

A. Related Work

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

A.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., 2025) 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.

A.2. RLVR for Multi-turn Evolutio

Independent trajectories generation. For each task-anchor pair $(x, y_0) \sim \mathcal{D}$, the behavior policy $\pi_{\theta_{\text{old}}}$ samples G independent K-turn trajectories $\{\mathcal{T}^{(i)}\}_{i=1}^{G}$, where $\mathcal{T}^{(i)} = (y_1^{(i)}, \ldots, y_K^{(i)})$.

Reward and advantage. The evaluator scores every turn, and the resulting per-turn rewards $r_t^{(i)}$ are aggregated into a trajectory reward $R^{(i)}$, which is then normalized into a trajectory advantage $\tilde{A}^{(i)}$.

$$R^{(i)} = \sum_{t=1}^{K} r_t^{(i)}, \quad \tilde{A}^{(i)} = \frac{R^{(i)} - \frac{1}{G}\sum_{j=1}^{G} R^{(j)}}{\text{std}_j(R^{(j)}) + \epsilon}. \tag{18}$$

Policy update. This advantage is shared by all turns in $\mathcal{T}^{(i)}$ and used in the clipped policy objective:

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{GK}\sum_{i=1}^{G}\sum_{t=1}^{K}\frac{1}{|y_t^{(i)}|}\sum_{k=1}^{|y_t^{(i)}|}\min\left(\rho_{t,k}^{(i)}\tilde{A}^{(i)}, \text{clip}\left(\rho_{t,k}^{(i)}, 1-\epsilon, 1+\epsilon\right)\tilde{A}^{(i)}\right)\right] - \beta\,\mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}), \tag{19}$$

where the importance ratio $\rho_{t,k}^{(i)} = \pi_\theta\left(y_{t,k}^{(i)} | x, h_{t-1}^{(i)}, y_{t,<k}^{(i)}\right) / \pi_{\theta_{\text{old}}}\left(y_{t,k}^{(i)} | x, h_{t-1}^{(i)}, y_{t,<k}^{(i)}\right)$. Here $h_{t-1}^{(i)}$ is the evaluated history before turn t. Although this objective uses evaluator feedback to update the policy, it collapses the temporal structure of refinement: every edit in the same trajectory receives the same advantage, whether it improved the molecule, made no progress, or produced an invalid candidate. This motivates turn-specific credit for learning molecular evolution.

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.

A.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., 2025). 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).

B. Experiments Details

B.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.

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 β as default to `verl`. The trade-off term ω 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. 6 for single-property optimization and Tab. 7 for multiple-property ones. For subsequent turns, we employ Tab. 8 and Tab. 9 for single and multiple property optimization tasks, respectively.

Table 6. 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. Your task: {task_description} Now it's your turn to respond to the current step. You should first conduct a reasoning process, which MUST be enclosed within `<think> </think>` tags. 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. If you are ready to provide the self-contained solution, provide the molecule only inside `<answer>` ... `</answer>`.

Table 7. 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. Your task: {task_description} Now it's your turn to respond to the current step. You should first conduct a reasoning process, which MUST be enclosed within `<think> </think>` tags. 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. If you are ready to provide the self-contained solution, provide the molecule only inside `<SMILES>` ... `</SMILES>`.

Success Rate (SR). SR is the fraction of test cases in which the optimized molecule improves all target properties while satisfying the similarity constraint $\mathrm{sim}(m, m') \geq 0.4$. 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 $\mathrm{sgn}(w_j) = +1$ for properties to maximize and -1 for properties to minimize. SR is computed as:

$$\mathrm{SR} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left[\mathrm{sim}(m_i, m_i') \geq 0.4 \ \wedge \ \bigwedge_{j=1}^{n}\Big(\mathrm{sgn}(w_j)\cdot\Delta P_{i,j} > 0\Big)\right], \tag{20}$$

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:

$$\mathrm{Sim} = \frac{1}{N}\sum_{i=1}^{N}\mathrm{sim}(m_i, m_i'). \tag{21}$$

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:

$$\mathrm{RI} = \frac{1}{N}\sum_{j=1}^{N}\mathrm{sgn}(w_j)\cdot\frac{F_j(m') - F_j(m)}{|F_j(m)|}. \tag{22}$$

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

Table 8. 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. Your task: {task_description} Prior to this step, you have already taken {step_count} step(s). Below is the interaction history, where `<answer> </answer>` wrapped your past molecules and <results> </results> wrapped the corresponding evaluator feedback: {memory_context} Now it's your turn to respond to the current step. You should first conduct a reasoning process, which MUST be enclosed within `<think> </think>` tags. 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. If you are ready to provide the self-contained solution, provide the molecule only inside `<answer>` ... `</answer>`.

Table 9. 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. Your task: {task_description} Prior to this step, you have already taken {step_count} step(s). Below is the interaction history, where `<SMILES> </SMILES>` wrapped your past molecules and <result> </result> wrapped the corresponding evaluator feedback: {memory_context} Now it's your turn to respond to the current step. You should first conduct a reasoning process, which MUST be enclosed within `<think> </think>` tags. 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. If you are ready to provide the self-contained solution, provide the molecule only inside `<SMILES>` ... `</SMILES>`.

B.2. 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.

B.3. Evaluation Metrics Details

A condensed summary (SR and RI) of multi-property results for both seen and unseen instructions appears as Tab. 2 in the main text. Full results including validity and similarity are shown below.

Table 10. Full multi-property molecular optimization results (seen instructions). Each result is averaged over 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
L2E (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

Full multi-property results (seen + unseen instructions). Tab. 11 gives the complete breakdown of multi-property performance on unseen instructions, including validity and Tanimoto similarity.

C. Training Hyperparameters

Table 13 reports the full training configuration used for all RL methods (GRPO, RLOO, POLO, LtE) in the main results. All methods share the same backbone, optimizer, batch/group/turn settings, and evaluator, isolating the algorithmic contribution to the advantage computation.

D. Compute Budget Details

Table 14 reports the matched-compute comparison for the three RL training pipelines used in the main experiments. All methods share the same Qwen2.5-3B-Instruct backbone, batch size 128, group size 16, and 8×A100 setup; only the credit assignment and loss construction differ.

Table 11. Full multi-property results on *unseen* instructions (Vld, SR, Sim, RI), averaged over 3 seeds. L2E achieves the highest RI on BDP under both seen and unseen regimes, demonstrating zero-shot generalization to novel instruction combinations.

Model	BDP				BDQ				BPQ			
	Vld↑	SR↑	Sim	RI↑	Vld↑	SR↑	Sim	RI↑	Vld↑	SR↑	Sim	RI↑
Evolutionary Algorithms												
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	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	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	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	0.502	0.320	0.896	2.511	0.327	0.219	0.928	0.185	0.604	**0.168**	0.943	0.066
RL Fine-tuning (Qwen2.5-3B)												
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
L2E (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

Table 12. Standard deviations for RL training algorithms (3 seeds). Column order matches Tab. 1.

Method	QED				LogP				MR			
	Vld	SR	Sim	RI	Vld	SR	Sim	RI	Vld	SR	Sim	RI
GRPO	.006	.017	.008	.006	.006	.035	.018	.051	.015	.023	.013	.018
RLOO	.000	.006	.005	.005	.000	.006	.003	.124	.000	.000	.002	.045
L2E	.000	.000	.004	.002	.006	.029	.018	.788	.010	.017	.010	.012

Table 13. Training hyperparameters shared by all RL methods on TOMG-Bench. Method-specific deviations: POLO uses preference-loss weight 0.3, $\beta = 0.08$, temperature 0.9; LtE uses $\omega = 1$ (intra weight) and similarity-penalty threshold 0.4.

Setting	Value
Backbone model	Qwen2.5-3B-Instruct
Hardware	8×NVIDIA A100 (80GB)
Batch size (prompts per step)	128
Group size (rollouts per prompt)	16
Max turns per chain (train)	5
Max turns per chain (inference)	10
Total training steps	54 (1 epoch of 2 configured)
Total oracle calls per run	552,960
Learning rate	1e−6
KL coefficient	1e−3
Optimizer	AdamW (β_1=0.9, β_2=0.999)
Gradient clipping	1.0
Mixed precision	bf16
RL framework	verl

Table 14. Compute comparison under matched oracle budget on TOMG-Bench. LtE adds ∼16% wall-clock over GRPO from z-score normalization only.

Method	GPU-Hours	Wall-Clock (h)
GRPO	99.1	12.4
RLOO	120.9	15.1
L2E (Ours)	**115.5**	14.4