INPUT-OUTPUT SAMPLING FOR EFFICIENT MULTI-OBJECTIVE MOLECULAR OPTIMIZATION WITH LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-objective molecular optimization with LLMs is bottlenecked by limited oracle budgets. Existing methods often struggle to maintain stable trade-offs among competing properties, and to select samples that provide informative learning signals, causing expensive evaluations to be spent on redundant molecules. To this end, we propose MOLDAS, a multi-objective reinforcement learning framework with Dual-sided Adaptive Sampling that selects original molecules at the input side and optimized candidates at the output side. At the input level, MOLDAS balances structural diversity with proximity to the policy's current capability boundary, selecting molecules that are underexplored but still learnable for oracle evaluation. At the output level, it sends the lowest-probability answer segments in each rollout group to the oracle. Moreover, it identifies high-value completions within rollout groups via reward dispersion and applies an auxiliary supervised fine-tuning loss for direct imitation. To further stabilize training, a lightweight dynamic weighting scheme calibrates property improvement, structural similarity, and reasoning quality as optimization progresses. MOLDAS is trained in a two-stage simulator-to-oracle paradigm, first with an inexpensive simulator and then adapted under a capped budget of high-fidelity oracle calls that mirrors real experimental constraints. Experiments show that MOLDAS raises the learning ceiling under abundant low-fidelity feedback and learns faster under limited high-fidelity feedback, achieving a 3.7% relative improvement over vanilla GRPO in Stage 1 and over 16.1% in Stage 2 on the optimization score. Sampling audits show reduced structural redundancy and higher molecular diversity, and lower performance fluctuation than fixed-weight training. These results suggest that our method offers a practical route to efficient and stable multi-objective molecular optimization with LLMs.

1 INTRODUCTION

Drug optimization is a central task in medicinal chemistry, aiming to discover molecular analogues with improved properties while remaining close to the parent scaffold (Quancard et al., 2025). A range of molecular properties must be considered, including absorption, distribution, metabolism, excretion, and toxicity (ADMET) (Fu et al., 2024). Considering these properties yields a combinatorially large space of possible analogues, so candidates cannot be exhaustively enumerated and must instead be found through efficient search.

Approaches to drug optimization fall into two families. Classical optimizers use generative models, genetic search, and policy-gradient updates over a property predictor (Jin et al., 2018; You et al., 2018; Jensen, 2019; Olivecrona et al., 2017; Zhou et al., 2019). These optimizers provide little rationale for each proposed edit. LLMs instead produce an explicit reasoning trace followed by an optimized SMILES string, while reinforcement learning (RL) steers generation toward molecules with improved predicted properties (Guo et al., 2025; Liu et al., 2026). Recent systems extend this interface with explicit chemical reasoning (Li et al., 2025; Zhao et al., 2025; Kim et al., 2025; Zhu et al., 2025; Wen et al., 2026). Because the reasoning trace gives an explicit rationale for each proposed edit, LLM-based optimization has attracted growing research interest and become a recent research trend.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Nevertheless, they still face three coupled challenges. (1) **Feedback is budget-limited:** experimental evaluation is impractical, so an external oracle provides model-based supervision, while each query consumes scarce budget (Fu et al., 2024). More rollouts improve structural coverage, whereas fewer rollouts weaken the within-group reward contrast required by Group Relative Policy Optimization (GRPO); the policy must therefore allocate calls across inputs and outputs. (2) **The objectives are conflicting and their difficulty changes during training.** Static reward weights can over-optimize a saturated objective, such as similarity, increasing the scalar reward while degrading the overall multi-objective trade-off. (3) **Informative inputs lie near a moving capacity boundary:** examples that once separated successful from failed rollouts can become trivial, while previously unreachable neighborhoods become learnable. A fixed curriculum therefore becomes stale, so input selection must track the evolving optimizer.

To address these challenges, we propose MOLDAS (*Molecular Optimization via Dual-sided Adaptive Sampling*), which treats feedback allocation as part of the RL problem. The method first chooses inputs by combining diversity ordering of the global pool with an uncertainty gate around the policy's current capacity boundary. Specifically, MOLDAS also uses the within-group reward dispersion already computed by GRPO (Shao et al., 2024) to gate the input sampler. Over the rollouts of each group, it sends the least likely answer segments to the oracle under the call budget. This keeps coverage broad while directing more training to underexplored regions that remain learnable. An online replay pool returns inputs whose earlier feedback was reliable and provides replayed inputs with a direct teacher-forcing signal for the reference reasoning and answer. To balance the competing objectives, a controller updates each reward weight according to its recent training trend and applies a bounded penalty to explicit rule violations. Training begins with an inexpensive simulator and then adapts the policy with a capped number of high-fidelity oracle calls.

The main contributions of this paper are as follows:

- We propose MOLDAS, a dual-sided algorithm that allocates feedback to training inputs and oracle-scored rollouts while adapting reward weights to changing objective difficulty.
- We develop a simulator-to-oracle training protocol with replay and teacher-forced supervision under a capped oracle budget.
- On the CHEMBL benchmark, MOLDAS improves the optimization score and target-property F1 score over vanilla GRPO; audits show reduced structural redundancy, greater molecular diversity, and more stable training.

2 RELATED WORK

Molecular optimization. Generative molecular optimization spans latent-variable and graph models (Jin et al., 2018; You et al., 2018) and genetic or tree-search procedures (Jensen, 2019). These methods typically optimize a scalarized objective over a fixed endpoint panel. LLM-based methods use chemical tools or chemical-text fine-tuning as proposal operators: Beam Enumeration extracts probable fragments for self-conditioned generation, while MOLLEO proposes evolutionary edits in chemical space (Edwards et al., 2022; M. Bran et al., 2024; Boiko et al., 2023; Guo & Schwaller, 2024b; Wang et al., 2025). Recent work trains the reasoning trace directly: Mol-R1, MolReasoner, and Logos use explicit chemically grounded reasoning, DrugR adds step-by-step pharmacological rationales, and MT-Mol and MECo separate reasoning from execution (Li et al., 2025; Zhao et al., 2025; Wen et al., 2026; Liu et al., 2026; Kim et al., 2025; Zhu et al., 2025). These methods improve proposal quality but do not address how to allocate limited oracle calls. MOLDAS keeps the LLM, GRPO objective, and DrugR's ADMET reward interface fixed (Liu et al., 2026), and instead selects the inputs and rollouts worth evaluating.

Reinforcement learning for molecular optimization. Reinforcement learning fine-tunes pretrained molecular generators against property predictors (Olivecrona et al., 2017; Blaschke et al., 2020; Zhou et al., 2019); recent LLM-based optimizers follow the same recipe, including DFRL-Mol's curriculum-guided structure and property analysis and RePO's reference supervision (Wu et al., 2026; Li et al., 2026). Most methods use PPO-style updates (Schulman et al., 2017; Ouyang et al., 2022) or group-relative variants (Shao et al., 2024; Ahmadian et al., 2024). MOLDAS retains GRPO's group-normalized advantage and composes with corrections for entropy collapse and ad-

vantage bias (Yu et al., 2025). Because scalarized rewards control multi-objective structure only implicitly, constrained RL uses Lagrangian, trust-region, or PID formulations (Altman, 1999; Achiam et al., 2017; Stooke et al., 2020), whereas multi-objective RL seeks Pareto coverage through preference conditioning or weight adaptation (Roijers et al., 2013; Hayes et al., 2022). We follow the latter line but track each reward channel's training trend rather than a preference vector or explicit Pareto front; this requires neither preference specifications nor objective scales. Our bounded rule penalty is related to reward shaping and reward hacking (Ng et al., 1999; Amodei et al., 2016; Skalse et al., 2022), and prevents invalid outputs from being traded against the smooth objectives.

Sample-efficient molecular optimization. Sample-efficient methods reduce expensive evaluations through search, replay, or input selection. Genetic- guided and Local Search GFlowNets focus proposals on promising regions (Kim et al., 2024b;c); Symmetric Replay Training, Augmented Memory, and Saturn reuse feedback through equivalent trajectories or alternative SMILES representations (Kim et al., 2024a; Guo & Schwaller, 2024a; Guo et al., 2026). DEPO, prioritized replay, and learnability-aware selection instead prioritize the training inputs (Tang et al., 2026; Schaul et al., 2016; Mindermann et al., 2022). None jointly models molecular structural diversity and an external oracle budget. MOLDAS addresses both: it selects inputs using capacity-boundary and inter-group diversity signals, then sends the lowest-probability answer segments of each group to the oracle. This distinguishes it from methods that only replay high-reward outputs or prioritize generic training inputs.

3 PRELIMINARIES

3.1 RL AND SUPERVISED OBJECTIVES

We use GRPO (Shao et al., 2024) to update the policy from group-relative rewards. For an input x_i, the policy samples G outputs $\{y_{i,1}, \ldots, y_{i,G}\}$, and output $y_{i,j}$ receives the reward $R_{i,j}$ defined in Section 3.2. GRPO compares rewards within each sampled group to form relative advantages, then applies a clipped policy-gradient update with a KL regularizer. We additionally use dual-clip stabilization for negative advantages (Appendix A.3). RL updates the policy according to rewards from generated outputs, whereas SFT minimizes the token-level negative log-likelihood of a reference output. During replay, we combine the two objectives as $\mathcal{L}_{\mathrm{GRPO}} + \lambda_{\mathrm{SFT}}\mathcal{L}_{\mathrm{SFT}}$, as defined in Eq. (3). The SFT term is computed only for replayed inputs; fresh inputs contribute only to the RL objective.

3.2 PROBLEM DEFINITION

We study multi-objective molecular optimization with LLMs, where a language model proposes chemical edits and RL shifts its distribution toward candidates with better predicted properties (Guo & Schwaller, 2024b; Wang et al., 2025). Let $\mathcal{M}$ be the space of molecules representable by a SMILES string, and let $\mathbf{p}\colon \mathcal{M} \to \mathbb{R}^d$ be the ADMET property profile that the oracle returns for a molecule. An input x is a hit molecule $m_{\mathrm{orig}} \in \mathcal{M}$ together with its profile $\mathbf{p}(m_{\mathrm{orig}})$. We constrain each analogue to a fixed band of Tanimoto similarity around the hit, $s_{\mathrm{lo}} \le \mathrm{sim}(a, m_{\mathrm{orig}}) \le s_{\mathrm{hi}}$. The lower bound preserves the parent scaffold; the upper bound excludes near-copies with unchanged structure.

Given an input x, the policy produces an output $y = (c, a)$, where c is a chain of thought that justifies the edit and a is the analogue in the answer field. We use output, completion, and rollout interchangeably for this sampled object. For input x_i, the policy samples G outputs, written $y_{i,1}, \ldots, y_{i,G}$, which are scored separately. The task is to find an analogue satisfying the similarity constraint that improves the target ADMET properties.

What makes this problem hard is evaluation. The target properties cannot be read off the structure, so each candidate is scored by the external ADMETlab oracle (Fu et al., 2024). Because the oracle is learned and its feedback is budgeted, direct optimization can exploit model errors and exhaust the available calls. A policy may therefore score well under the oracle without improving the properties it approximates (Ng et al., 1999; Skalse et al., 2022). The reward alone does not determine which inputs or rollouts should receive an oracle call, so we treat feedback allocation as part of the optimization problem. We now define the reward:

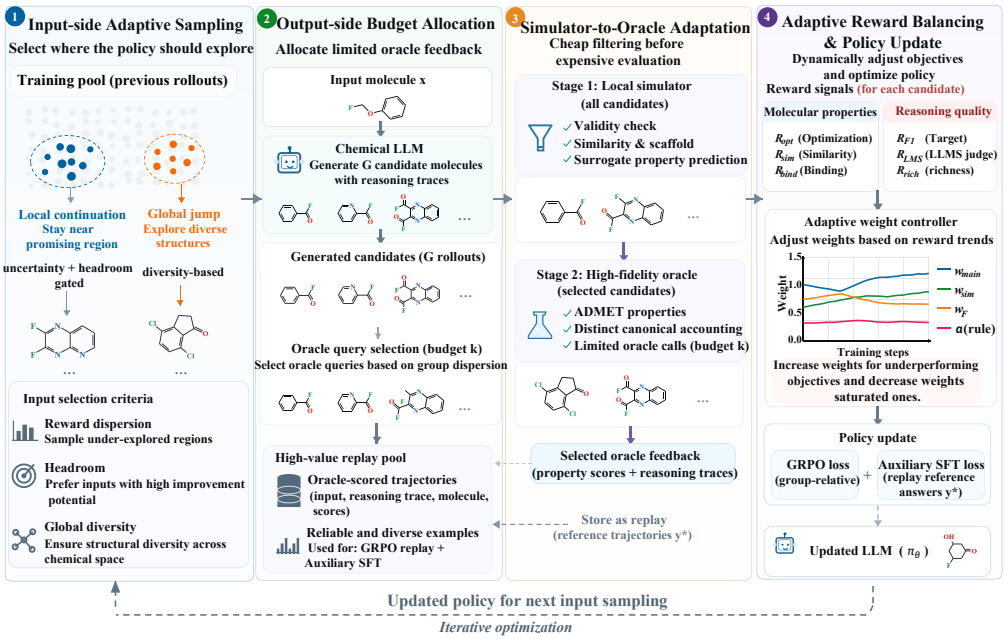


Figure 1: Overview of the MOLDAS training loop.

Reward. The base reward of an output is a weighted sum over six property and reasoning channels. Writing $w = (w_k)_{k\in\mathcal{K}}$ for the weights and $r_k(y)$ for the value of channel k on output $y = (c, a)$,

$$R_{\text{base}}(y; w) = \sum_{k\in\mathcal{K}} w_k\, r_k(y), \qquad \mathcal{K} = \{\text{opt}, \text{sim}, \text{bind}, F_1, \text{lms}, \text{rich}\}, \tag{1}$$

so $R_{i,j} = R_{\text{base}}(y_{i,j}; w)$ for output $y_{i,j}$ of input x_i. Each weighting w selects one trade-off among the conflicting channels, and we hold it fixed in this section and treat it as a controlled quantity in Section 4.3. The six channels fall into the two groups that DrugR balances against each other. The SMILES group scores the molecule a: the optimization score r_{opt} on the target ADMET properties, the fingerprint similarity r_{sim}, and the binding energy r_{bind}. The reasoning group scores the chain of thought c: the target-property F_1 score r_{F_1}, the language modeling score r_{lms}, and reasoning richness r_{rich}. The reward and the rules that decide whether a generated molecule counts as a valid answer both follow DrugR (Liu et al., 2026). Within this construction, only the optimization channel consumes oracle budget; a similarity indicator gates that channel, so a candidate that has drifted too far from the hit receives no credit for optimization.

4 METHOD

To address the challenges of Section 1, MOLDAS builds on the GRPO backbone and the composite reward of Section 3. It adds three components around them, as shown in Figure 1. The original molecules that provide useful learning signals and the optimized candidates worth evaluating change during training. MOLDAS therefore uses a dual-level sampler. At each training step, the sampler selects the original molecules that enter the batch, prioritizing those that provide the most informative learning signals. It then selects the optimized molecular candidates generated from these originals for oracle feedback. The sampler also retains original molecules whose feedback proved reliable in a high-value pool, so the calls already spent on them continue to provide training value. Direct optimization of a learned scorer can exploit its errors. MOLDAS therefore introduces the simulator-to-oracle protocol. It shapes the policy with locally computed signals first and adapts it under a hard cap on high-fidelity evaluations. To address the problem that a fixed weighting reward cannot track how the difficulty of each channel changes, MOLDAS designs the rule-adaptive controller to move each weight against the trend of the channel it multiplies.

4.1 DUAL-LEVEL SAMPLER

Oracle calls are the binding constraint of training, and a call devoted to an input the policy has already mastered or to a rollout that duplicates one already scored yields no new information. The inputs that carry a training signal and the rollouts that merit an oracle call both shift as the policy improves, so selections must therefore follow the policy's current behavior rather than a schedule fixed in advance. GRPO already computes, at no extra cost, the dispersion of rewards within each rollout group. This within-group reward dispersion tracks the shift in the policy's competence. A large value indicates that the policy can succeed on an original molecule but not reliably, and that the group contains enough contrast to support a group-relative advantage. MOLDAS uses this dispersion to guide the next batch of original molecules toward groups in which the policy can succeed but not reliably.

4.1.1 UNCERTAINTY-GATED NEIGHBOURHOOD SAMPLING

The sampler builds each batch in two steps. The *global step* draws from the full training pool in an offline diversity order based on Morgan fingerprints (Rogers & Hahn, 2010), Murcko scaffolds (Bemis & Murcko, 1996), and functional-group coverage. The *local step* samples neighbours of a parent from the previous batch and ranks them with

$$a(c \mid u) = \underbrace{w_u \widetilde{\sigma}_u \, \mathrm{sim}(c,u)}_{\text{uncertainty}} + \underbrace{w_h \widetilde{h}_u \, \mathrm{sim}(c,u)}_{\text{headroom}} + w_p \, \mathrm{sim}(c,u) + w_n \, \mathrm{nov}(c) - \sum_r w_r \, \rho_r(c), \quad (2)$$

where $\mathrm{sim}(c,u)$ is the Tanimoto similarity to parent u. The uncertainty term uses the normalized group dispersion $\widetilde{\sigma}_u$; the headroom term uses $h_u = \max_c n_c - \mathrm{mean}_c \, n_c$ and its normalized value $\widetilde{h}_u$, where n_c is the candidate's normalized main score. Both terms focus on neighbours of the parent. The proximity term also favours such neighbours, while novelty $\mathrm{nov}(c)$ preserves global coverage. The repulsion terms $\rho_r(c)$ penalize local crowding, candidates already in the batch, and recently abandoned regions. We require $\mathrm{sim}(c,u) \in [0.60, 0.90]$ and deduplicate canonical structures before ranking.

The previous batch determines whether the local step runs. A group is *trusted* only if it contains at least two valid molecules, two successful oracle responses, and two distinct canonical structures. Wide dispersion in a trusted group triggers a local step around the dispersed input; narrow dispersion or untrusted feedback triggers a global step. To prevent indefinite local search, the sampler tracks a root that covers the most active inputs and retains the region only while at least half of those inputs remain within Tanimoto similarity 0.75 of the root. After a target of $\{3, 4, 5\}$ batches, it restarts from canonical inputs not yet learned by the policy and below similarity 0.60 to the old region. A restarted input becomes a parent for one batch only if its feedback passes the reliability gate.

4.1.2 OUTPUT SELECTION UNDER A CALL BUDGET

The output side must reconcile what the model generates with what the oracle charges for. A low-probability answer segment marks a region it has not yet mastered, where the call to oracle carries the most information. The sampler therefore ranks the G rollouts by the mean log-probability of their answer segment a, and sends the q lowest to the oracle without de-duplicating first.

However, a policy that has settled on a good substructure emits it repeatedly, so a group of G rollouts may contain fewer than G distinct canonical molecules. therefore, the sampler merges the calling of the oracle for the same canonical molecule, averaging the probabilities of the answer segments of multiple rollouts that correspond to the same canonical molecule (Appendix A.2).

4.1.3 THE HIGH-VALUE INPUT POOL

Input selection decides where to explore, while the pool retains inputs whose reliable feedback remains useful after the sampler moves elsewhere. The reliability of inputs are ranked by the mean main score, headroom, validity rate, and within-group structural distance, then de-duplicated by canonical structure and capped per scaffold. Replayed inputs are regenerated by the current policy and pass through the same reward path as fresh inputs. Replay therefore gives an on-policy GRPO loss over a different input distribution, rather than supervision from stale log-probabilities or cached answers (Appendix A.3).

For replayed inputs, the oracle scores only the molecule at the end of a rollout, so its reasoning receives feedback only through the molecule's reward. Each input also carries a reference answer $y^\star$ that provides a direct signal in the reference-guided formulation of molecular optimization (Li et al., 2026). This teacher-forced term feeds the reference prefix $y^\star_{i,<t}$ at each position and scores the next reference token, so every token is supervised under the correct history. Writing A_i for the answer segment, the loss is

$$\mathcal{L}^{(i)}_{\mathrm{SFT}} = -\frac{1}{|A_i|}\sum_{t\in A_i}\log \pi_\theta\big(y^\star_{i,t} \mid x_i,\, y^\star_{i,<t}\big)\,, \qquad \mathcal{L} = \mathcal{L}_{\mathrm{GRPO}} + \lambda_{\mathrm{SFT}}\,\mathcal{L}_{\mathrm{SFT}}, \tag{3}$$

where x_i is the input prompt for replayed molecule i and λ_{SFT} weights the imitation term. At each step, $\mathcal{L}_{\mathrm{SFT}}$ is averaged over replayed inputs only; fresh inputs contribute to $\mathcal{L}_{\mathrm{GRPO}}$ but not to the SFT loss. Normalization by $|A_i|$ weights reference answers equally regardless of reasoning length.

4.2 Simulator shaping and oracle adaptation

Direct optimization of a single high-fidelity scorer is fragile. The ADMET oracle is a learned proxy (Fu et al., 2024), so a policy that optimizes it directly can exploit discrepancies from the underlying properties rather than learning the chemistry the scorer is meant to approximate. This is the failure mode associated with shaped rewards (Ng et al., 1999; Amodei et al., 2016; Skalse et al., 2022). The schedule therefore separates the two roles of the scorer. A *simulator* stage shapes the policy with locally computed signals, and an *oracle* stage adapts it under a capped budget of high-fidelity calls.

In the simulator stage, validity, format, similarity, and scaffold signals are computed locally with RDKit (Landrum, 2013), and a locally deployed surrogate predictor scores properties without a remote call (Appendix A.2). The non-monotonic similarity feasibility margin in Eq. (7) supplies the inexpensive shaping signal. This stage is cheap per step, and it carries most of the policy's initial adaptation, so the high-fidelity budget is spent on a policy that already produces valid analogues rather than on one that is still learning what a molecule looks like.

In the oracle stage, only a fraction of each batch enters the high-fidelity path and the remaining inputs use local signals without consuming oracle budget. Since the budget is the quantity the whole method is trying to protect, it is accounted for strictly. Each call is reserved before it is issued, so concurrent requests cannot overshoot the cap, and the count is over distinct canonical structures across steps and processes, so the same molecule is never charged twice and a cache hit costs nothing. If the budget runs out or the oracle fails, training stops with an error rather than substituting a proxy or a zero reward, so no update is ever driven by fabricated feedback.

4.3 Rule-adaptive reward balancing

Fixed weights cannot track changing channel difficulty: a saturated objective remains overemphasized, while an improving channel needs less pressure. The controller therefore updates each weight from the trend of its channel. Let w_k be the current weight and $\bar{r}_k^{(t)}$ the channel mean in the t-th window of T_{win} steps. It computes

$$\tau_k = \bar{r}_k^{(t)} - \bar{r}_k^{(t-1)}, \qquad \hat{\tau}_k = \tau_k / \|\tau\|_\infty, \tag{4}$$

and updates

$$\tilde{w}_k = \max\{w_k^{\min},\, w_k\,(1-\rho\,\hat{\tau}_k)\}, \qquad w_k \leftarrow \tilde{w}_k \cdot \frac{\sum_{j\in\mathcal{K}} w_j^0}{\sum_{j\in\mathcal{K}} \tilde{w}_j}, \qquad \rho = 0.2. \tag{5}$$

Here $w_k^{\min}$ prevents a channel from vanishing and ρ bounds its change per window. A declining channel has $\hat{\tau}_k < 0$ and gains weight, whereas an improving channel has $\hat{\tau}_k > 0$ and loses weight. The normalization places trends in $[-1, 1]$, and the final factor preserves the initial total weight $\sum_j w_j^0$. We use $T_{\mathrm{win}} = 10$, with w^0 selected by grid search.

A smooth reward cannot enforce hard requirements, since an unparseable string, a copied input, or an analogue outside the similarity band may still receive partial credit. We therefore define a bounded penalty $P_{i,j}$ for rollout j of input i and use $R^{(t)}_{i,j} = R_{\mathrm{base}}(y_{i,j}; w^{(t)}) - \alpha_t P_{i,j}$, where $R_{i,j}$ is defined in Section 3.2 and R_{base} is given by Eq. (1). The penalty is introduced gradually:

$$\alpha_t = \min\big\{1,\, \max\{0,\, (t - T_{\mathrm{warm}})/T_{\mathrm{ramp}}\}\big\}, \tag{6}$$

where T_{warm} delays its onset and T_{ramp} controls the ramp to full strength. The reward therefore changes continuously during warm-up.

5 RESULTS

5.1 BASELINES AND EVALUATION

All four arms start Stage 1 from the same DrugR-initialized LLaMA-3 8B checkpoint (Dubey et al., 2024; Liu et al., 2026), isolating training mechanisms from initialization:

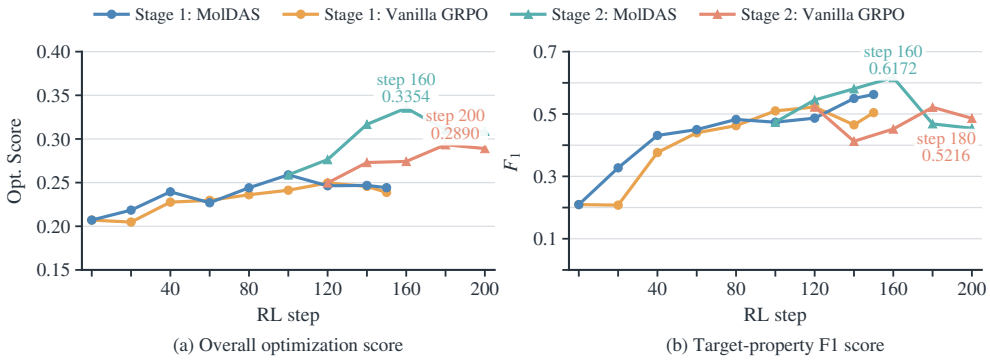
- **GRPO.** The plain baseline uses standard group-relative policy optimization (Shao et al., 2024) with the fixed composite reward and uniform input sampling.
- **MOLDAS (ours).** Starting from the same GRPO backbone, it adds dual-sided adaptive sampling for original molecules and optimized candidates, a high-value replay pool with teacher-forced SFT, and the rule-adaptive reward controller described in Section 4.
- **DFRL-Mol.** This baseline uses DFRL-Mol's curriculum experience pool and easy-to-hard schedule (Wu et al., 2026) in the shared GRPO pipeline.
- **Genetic GFN.** This baseline uses the population and fitness-biased parent-selection idea from Genetic GFN (Kim et al., 2024b) with the shared parent-conditioned LLM interface.

Stage 2 continues each arm from its own validation-selected Stage 1 checkpoint under the same oracle budget. Original molecules and bioactivity records come

Implementation details. Training uses the VERL/HybridFlow stack (Sheng et al., 2024) on four A100 GPUs, with FSDP2 parameter sharding, vLLM rollout generation, group-decoupled advantage normalization, and dual-clip policy-gradient stabilization. The high-fidelity scorer is an external ADMET service queried over HTTP (Fu et al., 2024).

5.2 TWO-STAGE TRAINING

All arms train to saturation in the simulator and then continue from their selected checkpoints in the ADMETLab stage under the capped budget. Equal Stage 1 and Stage 2 compute within each arm isolates differences due to feedback allocation. Figure 2 shows the same ordering in the simulator and oracle stages. We therefore compare baselines and ablations in the simulator stage, avoiding high-fidelity calls for every comparison while retaining the oracle-stage ranking. Whereas the allocation mechanisms are unchanged between the two stages, the analysis and ablation of oracle-call efficiency are confined to Stage 1, and we donot repeat them in Stage 2.



(a) Overall optimization score (b) Target-property F1 score

Figure 2: Comparison of MOLDAS and vanilla GRPO across the simulator and oracle stages. The curves report Opt. Score and the target-property F_1 against RL step.

Figure 2 shows the outcome. Our arm reaches its best Opt. Score of 0.3354 at an earlier step and at a higher value than the vanilla GRPO baseline's Opt. Socre of 0.2890, with an improvement of 16.1%. The gap is wider on the reasoning score. The ordering persists in the oracle stage: continuing each

arm from its own saturation point, our arm again peaks above the baseline on both metrics, and its advantage on the reasoning score widens.

5.3 MAIN COMPARISON

Table 1 compares methods under the common Stage 1 protocol. To interpret the baseline results, we instantiate DFRL-Mol and Genetic GFN within a common LLM optimization interface that isolates each method's defining mechanism. DFRL-Mol uses its curriculum experience pool and easy-to-hard schedule, whereas Genetic GFN uses population-based parent selection weighted by fitness. Both baselines share the DrugR-initialized backbone, prompt format, reward definition, data split, optimizer, and training budget. These implementations are listed under their standard method names in Table 1.

Table 1: Comparison on the fixed test set. Entries are the best values for the selected checkpoints under the final evaluation scorer. Stage 1 (simulator) covers all arms; Stage 2 (oracle) reports the arms of Figure 2 under the capped budget. Ablation and control rows are indented and italicized.

Method	Opt. Score	F_1
Stage 1 (simulator)		
Vanilla GRPO	0.2496	0.5225
DFRL-Mol (Wu et al., 2026)	0.2250	0.3691
Genetic GFN (Kim et al., 2024b)	0.2518	0.2015
MOLDAS (ours)	**0.2589**	**0.5627**
w/o active sampling	0.2508	0.5455
w/o output-side selection	0.2568	0.5541
w/o adaptive weighting	0.2544	0.2549
SFT + pretrain	0.2330	0.3155
Stage 2 (oracle)		
Vanilla GRPO	0.2890	0.5216
MOLDAS (ours)	**0.3354**	**0.6172**

The full MOLDAS achieves the highest values in both columns, reaching 0.2589 on Opt. Score and 0.5627 on F_1, compared with 0.2496 and 0.5225 for GRPO. The two baselines fall short for different reasons, and both are informative. DFRL-Mol receives a higher score from the simulator than from the ADMETLab oracle, where it reaches 0.2250 on Opt. Score. This discrepancy suggests reward hacking: the policy may exploit errors in the simulator without improving the underlying molecular properties, which is the failure mode the two-stage schedule exists to avoid. The Genetic GFN baseline has the opposite profile. Its Opt. Score is competitive and stable across both scorers, but its F_1 stays near 0.2 because of how the method works: it edits a molecule that has already been optimized and supplies that parent inside the prompt, so the model only has to rewrite a good answer rather than diagnose what is wrong with the input, and the target-property term has little to reward.

The indented rows report component ablations under Stage-1. Removing active sampling lowers both Opt. Score and F_1 to 0.2508 and 0.5455, respectively. Removing output-side selection yields 0.2568 on Opt. Score and 0.5541 on F_1. Removing adaptive weighting leaves Opt. Score close to the full model at 0.2544 but reduces F_1 to 0.2549. The SFT + pretrain control reaches 0.2330 and 0.3155, showing that supervised initialization alone does not reproduce the gains from feedback allocation. The results support the complementary roles of sampling and adaptive weighting.

5.4 ANALYSIS

We analyze the two feedback allocation decisions that determine how MOLDAS learns under a limited oracle budget: which inputs enter each batch and which reward channels receive emphasis. Sampling audits and checkpoint comparisons test whether the dual-level sampler improves the training distribution while following the policy's changing capacity boundary. Training logs then reveal whether the rule-adaptive controller reallocates emphasis as the relative difficulty of the objectives changes. Diversity ordering lowers mean and maximum nearest-neighbour similarity from 0.4697 to 0.3245 and from 0.9246 to 0.4085, respectively, while source coverage remains 99.98%. At the intermediate checkpoint, neighbourhood sampling raises Opt. Score from 0.2893 to 0.3056 and F_1

from 0.5716 to 0.6017; diversity ordering alone reaches 97.0% validity with 161 canonical structures and 71 scaffolds. The controller remains effective at longer horizons, where sampling loses its score advantage: its final window has invalid, copy, and low-similarity rates of 0.77%, 0.56%, and 0.21%, respectively.

To make the allocation behavior concrete, Figure 3 shows two audited records. The global decision at step 5, lane 45 responds to low parent similarity and replaces a phenyl ring with a pyridyl ring. The local decision at step 21, lane 44 keeps a high similarity parent when both normalized headroom and group dispersion are high. These rows illustrate the two sampler actions; they are not aggregate estimates. Each panel also reports one complete reasoning sentence from the matched checkpoint rollout. For each row, u denotes the selected parent, and $\widetilde{h}_u$ and $\widetilde{\sigma}_u$ are the normalized headroom and group dispersion defined in Eq. (2).

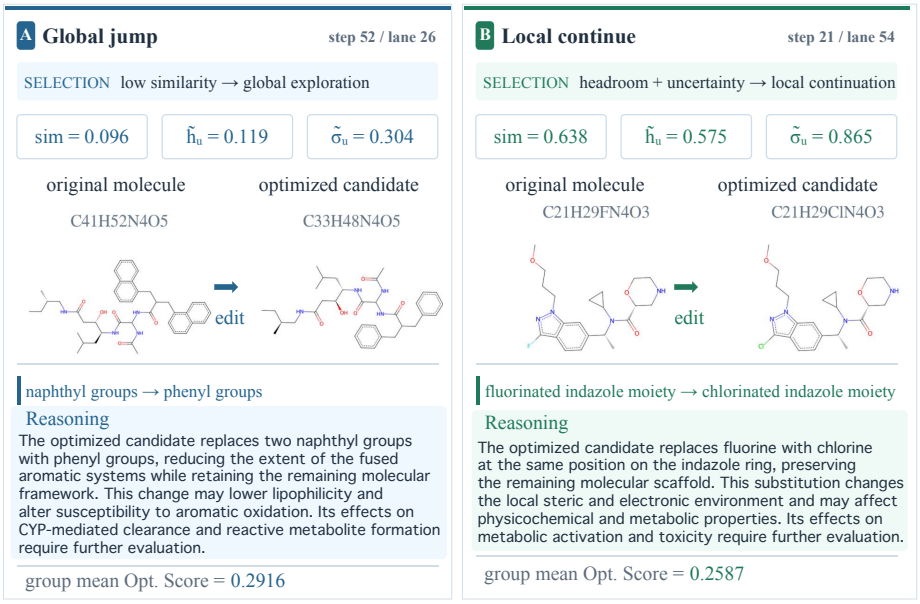


Figure 3: Audited input-sampler decisions with outputs, signals, parent, candidate, reasoning, and group-mean score.

The global case expands coverage when the current neighbourhood is not trusted, whereas the local case exploits a region with both headroom and reward dispersion. The two mechanisms therefore allocate feedback at complementary levels: the sampler chooses which original molecules enter training, and the controller decides which reward channels receive emphasis as their difficulty changes.

6 Conclusion

We studied multi-objective molecular optimization with LLMs under a bounded oracle budget, where each evaluation is expensive and the inputs that still carry a learning signal shift as the policy improves. MolDAS treats feedback allocation as part of the RL problem and implements it with three components. The first is a dual-level sampler that decides which inputs and which rollouts receive an oracle call. The second is the simulator-to-oracle protocol, which shapes the policy with locally computed signals before adapting it under a hard cap on high-fidelity calls. The third is a rule-adaptive controller that moves each reward weight against the trend of the channel it multiplies. The results quantify their roles. The sampler reduces mean nearest-neighbour similarity by 30.9% and adds five distinct Murcko scaffolds per batch at fixed size. The full pipeline achieves over 16.1% relative improvement in optimization score over vanilla GRPO.

Reproducibility statement

To facilitate reproducibility, the complete source code will be made publicly available soon in the project repository at https://anonymous.4open.science/r/MOLDAS-3715.

Ethics statement

This work concerns computational molecular optimization. Generated molecules are hypotheses for expert evaluation and are not claims of biological activity. Every reported property is a model prediction, not a measurement. We do not perform or enable synthesis, and the ADMET oracle is a predictive service rather than an experimental assay. Molecular optimization has dual-use potential.

AI assistance statement

AI-based tools were used during manuscript preparation for language polishing and information retrieval. They were not used to generate experimental data or to make scientific decisions. The authors verified the retrieved information, all reported results, and the final manuscript, and take responsibility for its content.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, 2017.

Arash Ahmadian, Chris Cremer, Matthias Gallé, et al. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *International Conference on Machine Learning*, 2024.

Eitan Altman. *Constrained Markov decision processes*. CRC Press, 1999.

Dario Amodei, Chris Olah, Jacob Steinhardt, et al. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of Medicinal Chemistry*, 39(15):2887–2893, 1996.

Thomas Blaschke, Josep Arús-Pous, Hongming Chen, et al. REINVENT 2.0: an integrated software package for expert-guided molecular generation and optimization. *Journal of Cheminformatics*, 12(1):59, 2020.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. Translation between molecules and natural language. In *Conference on Empirical Methods in Natural Language Processing*, 2022.

Lixia Fu, Shaohua Shi, Jiacai Yi, et al. ADMETlab 3.0: a comprehensive ADMET and physicochemical property prediction platform to cater to the drug discovery community. *Journal of Cheminformatics*, 16(1):104, 2024.

Anna Gaulton, Louisa J Bellis, A Patricia Bento, et al. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 40(D1):D1100–D1107, 2012.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jeff Guo and Philippe Schwaller. Augmented memory: Sample-efficient generative molecular design with reinforcement learning. *JACS Au*, 4(6):2160–2172, 2024a. doi: 10.1021/jacsau.4c00066.

Jeff Guo and Philippe Schwaller. Beam enumeration: Probabilistic explainability for sample efficient self-conditioned molecular design. In *International Conference on Learning Representations*, 2024b.

Jeff Guo, Junwu Chen, Anthony GX-Chen, and Philippe Schwaller. Sample-efficient generative molecular design using memory manipulation. *Nature Machine Intelligence*, 8(3):449–460, 2026. doi: 10.1038/s42256-026-01200-4.

Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.

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

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, 2018.

Hyeonah Kim, Minsu Kim, Sungsoo Ahn, and Jinkyoo Park. Symmetric replay training: Enhancing sample efficiency in deep reinforcement learning for combinatorial optimization. In *International Conference on Machine Learning*, 2024a.

Hyeonah Kim, Minsu Kim, Sanghyeok Choi, and Jinkyoo Park. Genetic-guided GFlowNets for sample efficient molecular optimization. In *Advances in Neural Information Processing Systems*, 2024b.

Hyomin Kim, Yunhui Jang, and Sungsoo Ahn. MT-Mol: Multi agent system with tool-based reasoning for molecular optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

Minsu Kim, Taeyoung Yun, Emmanuel Bengio, Dinghuai Zhang, Yoshua Bengio, Sungsoo Ahn, and Jinkyoo Park. Local search GFlowNets. In *International Conference on Learning Representations*, 2024c.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Greg Landrum. RDKit: Open-source cheminformatics software. *http://www.rdkit.org*, 2013.

Jiatong Li, Weida Wang, Qinggang Zhang, Junxian Li, Di Zhang, Changmeng Zheng, Shufei Zhang, Xiaoyong Wei, and Qing Li. Mol-R1: Towards explicit long-CoT reasoning in molecule discovery. *arXiv preprint arXiv:2508.08401*, 2025. doi: 10.48550/arXiv.2508.08401.

Xuan Li, Zhanke Zhou, Zongze Li, Jiangchao Yao, Yu Rong, Lu Zhang, and Bo Han. Reference-guided policy optimization for molecular optimization via LLM reasoning. *arXiv preprint arXiv:2603.05900*, 2026. doi: 10.48550/arXiv.2603.05900.

Haoran Liu, Zheni Zeng, Yukun Yan, Yuxuan Chen, and Yunduo Xiao. DrugR: Optimizing molecular drugs through LLM-based explicit reasoning. *arXiv preprint arXiv:2602.08213*, 2026. doi: 10.48550/arXiv.2602.08213.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535, 2024.

David Mendez, Anna Gaulton, A Patricia Bento, et al. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research*, 47(D1):D930–D940, 2019.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, 2022.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, 1999.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

Jean Quancard, Anders Bach, Chiara Borsari, Russell Craft, Christian Gnamm, Stéphanie M. Guéret, Ingo V. Hartung, Hannes F. Koolman, Stefan Laufer, Susan Lepri, Josef Messinger, Kurt Ritter, Gianluca Sbardella, Andrea Unzue Lopez, Marina K. Willwacher, Brian Cox, and Robert J. Young. The European federation for medicinal chemistry and chemical biology (EFMC) best practice initiative: Hit to lead. *ChemMedChem*, 20(8):e202400931, 2025. doi: 10.1002/cmdc.202400931.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, et al. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024.

Joar Skalse, Nikolaus HR Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *Advances in Neural Information Processing Systems*, 2022.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID Lagrangian methods. In *International Conference on Machine Learning*, 2020.

Xinyu Tang, Zhenduo Zhang, Yurou Liu, Xin Zhao, Zujie Wen, Zhiqiang Zhang, and Jun Zhou. Towards high data efficiency in reinforcement learning with verifiable reward. In *International Conference on Learning Representations*, 2026.

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, Yuanqi Du, Alan Aspuru-Guzik, Kirill Neklyudov, and Chao Zhang. Efficient evolutionary search over chemical space with large language models. In *International Conference on Learning Representations*, 2025.

Haibin Wen, Zhe Zhao, Fanfu Wang, Tianyi Xu, Hao Zhang, Chao Yang, and Ye Wei. Logos: An evolvable reasoning engine for rational molecular design. *arXiv preprint arXiv:2603.09268*, 2026. doi: 10.48550/arXiv.2603.09268.

Zhenyi Wu, Pengcheng Zhao, Jianing Li, Qiong Wang, Qin Zhang, Hui Yu, Zhe Yu, and Jian-Yu Shi. DFRL-Mol: A dual-stage framework of reinforcement learning for multi-scenario molecule optimization. *Briefings in Bioinformatics*, 27(5):bbag493, 2026. doi: 10.1093/bib/bbag493.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, 2018.

Qiying Yu, Zheng Zhang, Ruofei Zhuang, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Guojiang Zhao, Zixiang Lu, Yutang Ge, Sihang Li, Zheng Cheng, Haitao Lin, Lirong Wu, Hanchen Xia, Hengxing Cai, Wentao Guo, Hongshuai Wang, Mingjun Xu, Siyu Zhu, Guolin Ke, Linfeng Zhang, and Zhifeng Gao. MolReasoner: Toward effective and interpretable reasoning for molecular LLMs. *arXiv preprint arXiv:2508.02066*, 2025. doi: 10.48550/arXiv.2508.02066.

Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9(1):10752, 2019.

Wenyu Zhu, Chengzhu Li, Xiaohe Tian, Yifan Wang, Yinjun Jia, Jianhui Wang, Bowen Gao, Ya-Qin Zhang, Wei-Ying Ma, and Yanyan Lan. Coder as editor: Code-driven interpretable molecular optimization. *arXiv preprint arXiv:2510.14455*, 2025. doi: 10.48550/arXiv.2510.14455.

A Appendix

A.1 Sampling audit: coverage, decisions, and batch composition

We audit the trajectory of a complete run.

Coverage. Across $10{,}800$ input selections, 99.98% of the sources are used, with a Gini coefficient of 0.0573 and a maximum per-source frequency of 3. The top-100 most frequently selected sources account for 2.78% of all selections. Active sampling in our setting therefore does not collapse onto a few favoured molecules.

Batch composition. Against 200 random rebatchings produced by vanilla GRPO on the same inputs, the diversity curriculum reduces the mean nearest-neighbour similarity within a batch from 0.4697 to 0.3245 (-30.9%) and the maximum nearest-neighbour similarity from 0.9246 to 0.4085 (-55.8%), while raising distinct Murcko scaffolds per 72-input batch from 65.53 to 70.58. The mechanism is the removal of local near-duplicates and the enrichment of scaffold coverage, precisely the regime in which a group-relative advantage estimator has something to compare.

A.2 Simulator score, budget accounting, and the student proxy

The policy starts from the DrugR checkpoint described in Section 5.1, and is shaped using the composite reward of Eq. (1) plus the rule controller. The simulator score for a rollout is computed entirely locally, in milliseconds, by a cascade. An answer whose structure we have already scored is credited from a content-addressed cache. Otherwise the property channels are scored by a single target-aligned student proxy, which supplies the 37 molecular property and safety fields exposed by the ADMET oracle without a remote call, and the similarity term is a bounded *feasibility margin* on the Tanimoto similarity to the original molecule,

$$\phi(s) = \begin{cases} -\min\left\{1,\ (s_{\mathrm{lo}} - s)/s_{\mathrm{lo}}\right\}, & s < s_{\mathrm{lo}}, \\ 1 - 2\,|s - \bar{s}|/(s_{\mathrm{hi}} - s_{\mathrm{lo}}), & s_{\mathrm{lo}} \le s \le s_{\mathrm{hi}}, \\ -\min\left\{1,\ (s - s_{\mathrm{hi}})/(1 - s_{\mathrm{hi}})\right\}, & s > s_{\mathrm{hi}}, \end{cases} \tag{7}$$

with $s_{\mathrm{lo}} = 0.60$ and $\bar{s} = (s_{\mathrm{lo}} + s_{\mathrm{hi}})/2$. Equation (7) is deliberately non-monotonic: it rewards a molecule for being *meaningfully different* from the original while penalizing both verbatim copying and arbitrary drift. This stage is cheap per step. The proxy's candidate structures were pooled from public ADMET datasets, then canonicalized, deduplicated, and filtered for overlap with prior optimization, oracle-cache, split, ranking, and audit assets. We used random seed 20260824 to construct scaffold-disjoint partitions with $10{,}000$ training, $2{,}000$ development, and $2{,}000$ sealed scaffold-OOD test molecules, and each molecule carries 119 raw fields, mapped through a frozen target contract to the 37 fields used by the student. For each learned endpoint, we compared Random Forest and Extra Trees predictors with 2048-bit ECFP4 fingerprints, with and without nine RDKit descriptors. All candidates used 300 trees, with `min_samples_leaf`=2 and `max_features=sqrt`. Low-cardinality endpoints with at most eight unique training values were treated as classification tasks, with the prediction given by the expectation over class probabilities; the remaining endpoints were fit by regression. We selected the model and representation with the lowest development-set MAE, fitted a train-only five-fold scaffold-grouped out-of-fold isotonic calibrator, and refit the selected

model on all $10{,}000$ training molecules. Molecular weight, TPSA, fraction Csp3, and QED were computed directly from the molecular structure with RDKit. The final composition contains 21 Extra Trees models with descriptors, 9 Random Forest models with descriptors, 3 Extra Trees models without descriptors, and these four direct structural calculations. The student was evaluated once on the $2{,}000$ scaffold-OOD molecules after model selection and calibration were fixed. We report Spearman correlation, Kendall's τ-b, MAE, and RMSE on each endpoint's native scale, so rank correlations are the appropriate cross-endpoint summary. Learned endpoints mostly achieve Spearman correlations between 0.70 and 0.89, with NPscore reaching 0.930. The four direct structural fields are effectively exact, with any residual error attributable to numerical precision in the oracle output. In the oracle stage, the policy is then adapted under a hard-capped high-fidelity budget. Only a fraction of each batch enters the oracle path. The remaining inputs receive locally computable constraint and proxy signals and consume no oracle budget. The cap is enforced strictly: each request reserves budget before it is issued, the count is over distinct canonical structures across steps and processes, and a cache hit is credited at zero cost. Crucially, if the budget is exhausted or the oracle fails, we raise and stop rather than substituting a proxy score or a zero reward, so that no training update is ever silently driven by fabricated feedback.

A.3 High-value feedback replay

The sampler decides where to explore *next*. A separate question is whether to keep exploiting inputs that already produced strong, reliable evidence. We maintain an online pool $\mathcal{D}_H$ of such inputs, eligible once their history contains a trusted group and scored by a median-aggregated quality

$$Q(x) = 0.50\ \mathrm{med}(\bar{n}) + 0.30\ \mathrm{med}(h) + 0.10\ \mathrm{med}(v) + 0.10\ \mathrm{med}(d), \tag{8}$$

where $\bar{n}$ is the group mean main score, h the headroom defined in Section 4.1.1, v the validity rate, and d the mean intra-group structural distance. The pool uses canonical de-duplication and a per-scaffold cap, so replay cannot concentrate on a single chemotype.

Training then proceeds on a mixture of the replay pool and randomly sampled control inputs, $p_{\mathrm{mix}} = p\,\mathrm{Unif}(\mathcal{D}_H) + (1-p)\,\mathrm{Unif}(\mathcal{D}_R)$ with $p = 0.30$. Replayed inputs are regenerated by the current policy through the identical reward path, so the replay term differs from the main loss only in the input distribution. The objective is the token-mean clipped policy-gradient loss with dual-clip stabilization plus a low-variance KL penalty. Because the replay weighting enters through the *input distribution* rather than as a per-example loss weight, token-level normalization stays consistent across the two sub-batches.

Output selection follows the same accounting rule across groups. It operates on distinct canonical structures rather than on individual rollouts. The G rollouts of input x_i are grouped by canonical structure. Each group is scored by the mean answer-segment log-probability of the rollouts it contains,

$$\ell_{i,R} = \frac{1}{|R|}\sum_{j\in R}\ell_{i,j}, \qquad \ell_{i,j} = \frac{1}{|A_{i,j}|}\sum_{t\in A_{i,j}}\log \pi_\theta(y_{i,j,t} \mid x_i,\, y_{i,j,<t})\,, \tag{9}$$

where R is that group and $A_{i,j}$ the answer segment of rollout j. The q groups with the lowest $\ell_{i,R}$ receive an oracle call. The oracle is queried once per distinct canonical structure, and the returned property feedback is written back to every rollout that resolves to that structure. Each rollout still enters the update with unit weight, so de-duplication changes the number of oracle calls rather than the training signal. A cache hit reuses existing feedback at zero cost, and the budget counts distinct canonical attempts rather than HTTP requests, so retries do not inflate reported usage.

A.4 Controller boundary behaviour

The controller is specified to be step-level: within a training step all rollouts are scored with one weight vector, and the vector is recomputed only after the step completes. We verify the following invariants by unit test. (i) At $t < T_{\mathrm{warm}}$ the effective strength is $\alpha_t = 0$ and the rule penalty is exactly zero. (ii) At the midpoint of the ramp $\alpha_t = 0.5$, and (iii) after the ramp $\alpha_t = 1$. (iv) The renormalization of Eq. (5) preserves the total weight mass exactly after every update, so the scale on which the group-relative advantage is computed never drifts. (v) A channel whose initial weight is zero stays at zero and is never revived by a trend, and every active weight is floored at a small fraction of its initial value. (vi) All rollouts of a step are scored with the same weight vector, and

validation rollouts and late-arriving results from a previous step leave the window and the weights unchanged.

A.5 ILLUSTRATIVE CONFIGURATION

The configuration used for the long-horizon runs reported in Section 5.4 is as follows. Data: $5{,}076$ training inputs and 565 validation inputs. Optimization: a batch of 72 inputs per step, with $G = 4$ rollouts per input at temperature 0.7 during training. Prompt and response lengths are capped at 512 and 640 tokens. Controller: rebalancing window $T_{\text{win}} = 10$ steps, adjustment factor $\rho = 0.2$, and initial weights w^0 set by grid search as described in Section 4.3. Sampler: global fraction 0.75 of the batch, local similarity band $[0.60, 0.90]$, and a confidence gate requiring two valid candidates, two successful oracle responses, and two distinct canonical structures. Temporal chain length is drawn from $\{3, 4, 5\}$ with chain similarity 0.75, match fraction 0.50, and restart similarity ceiling 0.60. Replay: proportion $p = 0.30$, with $\beta_{\text{KL}} = 2 \times 10^{-4}$ in the replay/stage-2 configuration. AdamW (Kingma & Ba, 2015) with actor learning rate scheduled $1 \times 10^{-6} \to 5 \times 10^{-7} \to 2 \times 10^{-7}$ at steps 30 and 45.