LEARNING RELIABLE SKILLS FROM SPARSE ONLINE FEEDBACK FOR COLD-START LLM-BASED MOLECULAR OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Molecular optimization seeks drug-like candidates with strong target-specific properties under small budgets of costly evaluations. LLM-based optimizers are attractive because they propose molecules from natural-language objectives and adapt in context without training a task-specific generator. Yet methods that endow an optimizer with reusable skills typically require large offline trajectories, either to train a policy or to build a retrieved skill library — resources a cold-start target does not have. While simpler cold-start methods such as OPRO present candidates as a score-ranked list of independent molecules, preserving scores but discarding the skills latent in the online history. We propose OS-DEV (Online Skill Distillation with Evidence-Aware Verification), a training-free feedback layer that turns each evaluation into a best-referenced trace and periodically distills improving and degrading traces into target-specific proposed skills. Two independent analysts extract complementary hypotheses about what to pursue and avoid, and a role-independent verifier re-examines each proposed skill against the supporting and contradicting observations accumulated in the current run, attaching an interpretable high/medium/low confidence judgment before the resulting verified skill state guides subsequent generation. On all 58 DOCKSTRING targets with 200 oracle evaluations per target and a shared Qwen3.5-4B generator, OS-DEV reaches an overall score of 0.8487, improving over ACE (0.8254) by 0.0233 and over an agentic reference combining Claude Code with DeepSeek-V4-Pro (0.8270) by 0.0217, and leads on 30 of 58 targets. Ablations isolate the contributions of the trace display, the dual analysts, and the evidence-aware verifier, and confirm that organizing sparse feedback into reusable, evidence-linked skills improves LLM-based molecular optimization without retraining the proposal model.

1 INTRODUCTION

Molecular optimization seeks drug-like candidates with strong target-specific properties under small budgets of costly experimental or computational evaluation. Recent LLM systems have broadened the role of language models in chemistry from tool-using assistants and autonomous agents (Bran et al., 2024; Boiko et al., 2023) to molecular generation and search (Edwards et al., 2026; Li et al., 2026). LLM-based optimizers offer a particularly simple interface for the latter setting: they translate a natural-language objective and a history of scored molecules directly into new SMILES proposals, adapting in context without training a task-specific generator. OPRO (Yang et al., 2024) and BOPRO (Agarwal et al., 2025), for example, reuse previous evaluations through an in-context, score-ranked history presented at each generation round.

A large and growing body of work improves such optimizers by giving them reusable *skills*: structured guidance about which molecular changes help or hurt on a target.Most of these methods acquire skills from *offline* resources: they are distilled from large pre-collected optimization trajectories, used to train a reusable policy, or assembled into a cross-target skill library that is retrieved at call time (Wang et al., 2026; 2025; Ran et al., 2025; Ni et al., 2026; Yuan et al., 2026). This premise often fails in realistic molecular optimization. High-quality offline trajectories for a newly chosen target are rarely available; the expensive docking oracle leaves only a handful of online samples; and retrieval from the small, locally accumulated pool of one run is unreliable. Simpler

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.

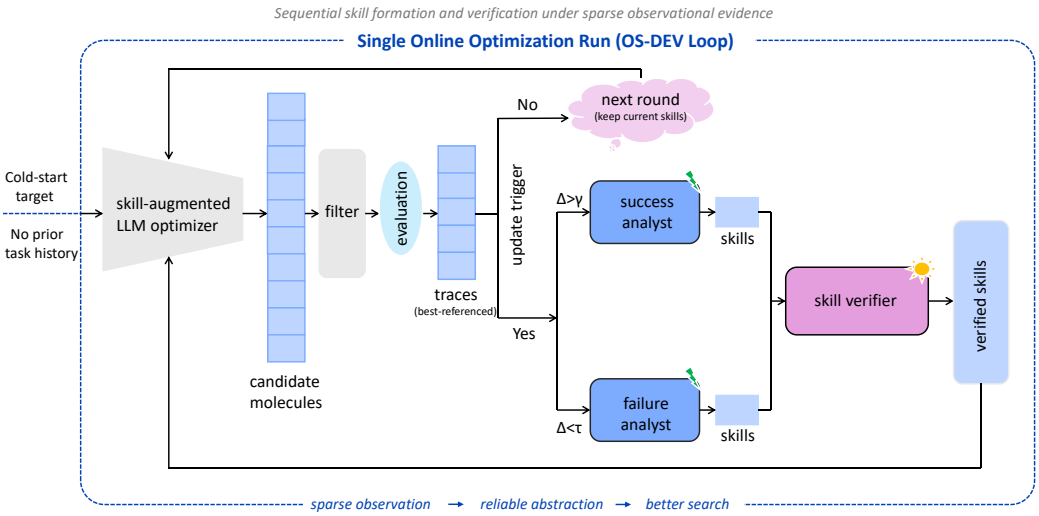


Figure 1: OS-DEV within one online run: best-referenced traces → dual analysts → skill verifier → verified skill state → optimizer, refreshed periodically or on plateau.

cold-start methods such as OPRO instead keep the raw history and present evaluated candidates as a score-ranked list of independent molecules. This outcome-only representation preserves scores but discards the skills latent in the online trajectory, forcing the LLM to re-derive the useful relations from raw SMILES on every generation round.

We identify this as an overlooked bottleneck in LLM-based optimization: expensive oracle evaluations are retained as examples, but not organized into reusable, trustworthy skills. The central question in a cold-start, scarce-budget setting is therefore not whether an optimizer can memorize more examples, but *how to turn a few early online observations into skills that are extracted, kept accurate, and correctly re-applied under the latest optimization state*. Making this work is hard for three reasons, which we set out in turn.

First, because the LLM proposes molecules freely, a candidate is not necessarily a single chemical edit of a certain molecule; observed score gains are entangled across simultaneous structural changes, and scalar improvements alone do not say which contrast to pursue or avoid. Second, and more seriously, skills inferred from very few samples are inherently unstable. Over-generalizing a structural pattern from a handful of feedback or ignoring the applicability conditions under which that pattern holds will yield unreliable rules. Untrusted skills injected into the prompt can actively misguide the optimizer; we call this failure *skill hallucination*. Third, optimization proceeds in stages: a rule that helps during early exploration can hurt during late refinement, so old skills cannot be assumed to stay valid, and a mechanism is needed to refresh guidance as evidence arrives.

To address these challenges we propose **OS-DEV** (Online Skill Distillation with Evidence-Aware Verification), a training-free feedback layer that leaves the oracle and the proposal model unchanged. At each refresh it re-encodes the history as *best-referenced traces*, distills them into candidate skills through two analysts, and has a role-independent verifier re-check every skill against all observations accumulated so far, labeling it high/medium/low before it enters the prompt; the whole state is then replaced, never accumulated. The method updates no parameters, needs no external corpus, and lets every distilled statement be traced to observations from the current run. Figure 1 summarizes this pipeline, which Section 3 details in full.

We evaluate OS-DEV on all 58 DOCKSTRING targets (García-Ortegón et al., 2022) under 200 oracle evaluations per target. With a shared Qwen3.5-4B generator, OS-DEV reaches an overall score of **0.8487**, improving over the OPRO baseline (0.8197) by 0.0290 and over a substantially higher-capacity agentic reference that combines the Claude Code harness with the DeepSeek-V4-Pro model (0.8270) by 0.0217. Controlled ablations on the full suite isolate the contribution of each component: the best-referenced trace representation, the dual analysts, the evidence-aware verifier,

and domain-informed reasoning. Each removal degrades performance, and the verifier accounts for the largest drop attributable to any single component.

Our contributions are:

- We formulate cold-start molecular optimization as the problem of extracting, verifying, and refreshing reusable skills from sparse online feedback, and identify *skill hallucination*, the skills that over-generalize sparse evidence, as the failure that scarce-budget settings make most acute.
- We introduce OS-DEV, a training-free feedback layer that distills best-referenced traces into target-specific proposed skills with a dual-analyst procedure, and adds a role-independent Skill Verifier that re-examines each skill against supporting and contradicting observations before it enters the prompt.
- We show on 58 DOCKSTRING targets that OS-DEV improves both final performance and generation-round progress under a fixed oracle budget, and that ablations of the trace representation, the analysts, and the verifier isolate the contribution of each component.

2 PRELIMINARIES

2.1 PROBLEM SETUP

Objective. Molecular optimization searches a space $\mathcal{X}$ of chemically valid molecules for candidates that score well against a target protein p; each candidate $x \in \mathcal{X}$ is represented by a SMILES string. We write the objective as a target-indexed function $f_p : \mathcal{X} \to [0, 1]$, the composite used by the BOPRO protocol,

$$f_p(x) \;=\; 0.2 \cdot \mathrm{QED}(x) \;+\; 0.8 \cdot \mathrm{normalized}\big(-\mathrm{Vina}_p(x)\big), \tag{1}$$

where $\mathrm{QED}(x) \in [0, 1]$ estimates drug-likeness, $\mathrm{Vina}_p(x)$ is the docking score of x against p, and $\mathrm{normalized}(-v) = (-v - 3)/9$ maps $\mathrm{Vina}_p \in [-12, -3]$ linearly onto $[0, 1]$.

Protocol. Because f_p can be queried only through an expensive docking oracle, an optimization protocol is allotted a budget of B oracle evaluations for a single target. It begins from a small shared warmstart of B_0 evaluated molecules and then proceeds in rounds, proposing candidates $x_1, x_2, \cdots \in \mathcal{X}$ on the basis of the pairs $\{(x_i, y_i)\}_{i<t}$ observed so far and receiving $y_t = f_p(x_t)$ from the oracle.

Problem. A protocol is scored by the best objective value it reaches,

$$\mathrm{Best}_B \;=\; \max_{1 \le t \le B} f_p(x_t), \tag{2}$$

and the aim is to maximize it subject to the oracle budget. We call the setting *cold-start* when, before the run, no target-specific history, offline trajectory, or cross-target library for p exists, so that the protocol's only inputs are the B_0 warmstart scores and the evaluations it has itself bought. Under this budget the oracle returns a scalar per candidate and nothing else: it exposes neither a gradient, nor a structural edit path from one candidate to the next, nor any decomposition of the score into the two terms of Equation 1. Recovering reusable structure from a handful of such scalar observations for a new target is the problem this paper addresses.

2.2 RELATED WORK

We organize prior work by the answer it gives to *where reusable knowledge comes from*; OS-DEV's answer is to distill it online from the current run's sparse feedback and verify it before use, in contrast with four alternatives.

Inference-time optimization over scored feedback. OPRO (Yang et al., 2024) conditions the next batch on a score-ranked history without updating parameters, establishing the interface OS-DEV builds on; surrogate-assisted variants pre-screen candidates before the budget is spent: BO-PRO (Agarwal et al., 2025) selects informative observations via a model in a continuous embedding space, and short-term graph memory (Yang et al., 2026) re-ranks each round's proposal pool with

an online graph surrogate trained only on recorded oracle scores. These methods present candidates as independent molecule–score records, exposing neither the candidate–incumbent relation nor reusable skills. OS-DEV instead exposes both; we compare with OPRO and BOPRO under the same base model and oracle budget, asking whether verified, relation-exposing skills are more useful than independent records or surrogate-ranked pools.

Skills from offline experience. MolMem (Wang et al., 2026) and POLO (Wang et al., 2025) acquire reusable experience before the run via trained policies or evolving memory; RePO (Li et al., 2026) confronts the same absence of step-by-step trajectories, but answers it by anchoring RL updates to dataset reference molecules—compiling the guidance into policy parameters; Trace2Skill (Ni et al., 2026) and ClawTrace (Yuan et al., 2026) distill agent trajectories into transferable skills. F2R (Gao et al., 2026) goes further, distilling accumulated history into retrievable knowledge through guided self-reflection, though its pool is built across tasks and retrieved items are trusted as-is, without per-item verification. These show that trajectory-to-skill distillation works, but rely on offline corpora, reference datasets, or cross-task memory that a cold-start target lacks. OS-DEV keeps the skill idea without the offline prerequisite and without any parameter update: skills come only from current-run observations. Because these methods presuppose offline corpora or parameter training that our setting forbids, they are not eligible baselines; the comparison is therefore architectural rather than empirical.

Structured, evolving context. ACE (Zhang et al., 2026) maintains a playbook through incremental helpful/harmful updates; RE-TRAC (Zhu et al., 2026) carries evidence and uncertainty across trajectories; Reflexion (Shinn et al., 2023) and Self-Refine (Madaan et al., 2023) revise outputs through verbal feedback. These show explicit, revisable context helps. OS-DEV differs in *what* it revises and *how*: ACE grows its playbook by incrementally adding and pruning entries, whereas OS-DEV re-derives its state from the current run's observations and re-verifies each skill before use. Under the shared protocol, OS-DEV outperforms this baseline (Table 1).

Reliable reasoning and hallucination mitigation. Self-reported confidence is often unreliable (Chen & Yi, 2026). MT-Mol (Kim et al., 2025) uses reasoning verification in a multi-agent setting. OS-DEV applies this idea at the skill level: a role-independent verifier re-checks each skill against cited traces and labels it high/medium/low according to evidence sufficiency, contradictions, and target specificity. Unlike methods that silently discard uncertain claims, we keep confidence visible so the optimizer can weigh them, and repeat verification as evidence accumulates. Our ablations isolate the verifier's contribution.

3 METHOD

OS-DEV is a training-free feedback layer around an LLM-based optimizer for a single cold-start target: no target-specific history, offline trajectories, or cross-target memory. It leaves the molecular oracle and proposal model unchanged. The optimizer still conditions each round of generation on the accumulated score-ranked history (the same interface as the OPRO baseline); OS-DEV additionally maintains a compact, verified textual guidance, which we call the *verified skill state*. This state is inserted into the generation context and replaced as the run progresses. The layer therefore preserves the optimizer's cold-start interface while making the reusable structure latent in the online trajectory explicit and trustworthy. The subsections below specify each stage in turn, and Algorithm 1 summarizes the resulting online loop.

3.1 SCORE-RANKED HISTORY AND BEST-REFERENCED TRACES

Write $y_i = f_p(x_i)$ for the score of the i-th evaluated molecule under the target-specific objective of Equation 1, and let B be the oracle budget. As in the OPRO-style baseline, the optimizer keeps the full score-ranked history of evaluated molecules (x_i, y_i) and shows the top-scoring examples in each generation prompt. This outcome-only representation preserves scores but hides *why* one candidate beat another. OS-DEV therefore retains the relation between consecutive decisions: whenever the skill state is refreshed, it re-encodes the history as **best-referenced traces**. Before evaluating candidate x_i the optimizer holds an incumbent x^*_{i-1} (the best molecule available before that evaluation) with score y^*_{i-1}; the trace is the pair (x^*_{i-1}, x_i) together with the signed change $\Delta_i = y_i - y^*_{i-1}$

and its QED/Vina components. Anchoring each observation to the pre-evaluation incumbent gives it a common decision-relative meaning — the candidate was better or worse than what the optimizer could already achieve — and prevents look-ahead leakage. Because an LLM proposes molecules freely, adjacent candidates need not be related by a single chemical edit, so OS-DEV deliberately does *not* pre-compute a structural diff between x^*_{i-1} and x_i and instead lets the analyst infer which contrast matters from the two raw SMILES; the incumbent–candidate pairing is thus only an observed correlation in the trajectory and does not imply a single-edit relation between the two, and any mechanism read off it is a hypothesis to be checked, which is precisely the verifier's role.

3.2 From Traces to Proposed Skills

At each refresh the two analysts each receive a subset of traces formatted as a numbered list, one per evaluation, each showing the incumbent, the candidate, the composite scores (and QED/Vina components) of both, and the signed change. OS-DEV partitions the traces by outcome direction. The **success analyst** receives traces with $\Delta_i > \gamma$ (γ=0.02; at most 15: 10 most recent plus 5 sampled to retain both local and older evidence) and returns 3–5 proposed skills of the form

$$[\text{observable pattern}] + [\text{mechanistic hypothesis}] \mid \text{traces: } n_1, \ldots, n_k, \tag{3}$$

linking a structural pattern (e.g., para-fluorination, phenyl-to-pyridine replacement) to a plausible reason it helped the composite score (e.g., a hinge H-bond, filling a hydrophobic sub-pocket), and marking the trace indices that support it.

The **failure analyst** receives two kinds of negative evidence, namely traces with $\Delta_i < \tau$ (τ=-0.02) and syntactically invalid SMILES (unparseable by RDKit, or rejected by the validity filter), and returns avoidance skills: patterns associated with degradation (lost interactions, unfavorable strain, poor drug-likeness) and recurring invalid-generation patterns to suppress. Every skill is explicitly a *proposed* hypothesis grounded in observed score directions, with its supporting-trace indices retained; because it is distilled from few traces it may over-generalize, a failure mode that the verifier stage, described in the next subsection, is designed to catch.

3.3 Skill Verification and Confidence

Because a skill is inferred from only a handful of traces, any single proposed skill may be a hallucination: the cited traces may be too few, internally contradictory, or consistent only with a generic pattern that holds for any protein. The **Skill Verifier** is therefore a second, role-separated LLM call that re-examines each proposed skill before it enters the prompt.Its input is the proposed skills (each with its cited trace indices) together with the full set of observations accumulated so far, not just the cited subset, so that it can look for both supporting and contradicting observations. Its prompt asks it to audit each skill on three criteria — *evidence sufficiency*, *mechanism plausibility*, and *target specificity* — and to return, for each skill,

$$\text{Skill } n\text{: Confidence: high} \mid \text{medium} \mid \text{low} \quad — \quad \text{one-sentence justification.} \tag{4}$$

Confidence therefore assesses how much *empirical support* a skill has accumulated under sparse evidence, and this screening signal lets the verifier improve the overall quality of the distilled skills. The analyst and verifier share the same base model to ensure experimental fairness; the verifier's rigor comes from an independent inference call driven by a dedicated reviewer prompt ("rigorous, skeptical reviewer"). The verification stage thus acts as a soft gate that keeps the optimization loop running smoothly: if a verifier call fails or its output cannot be parsed, the skill is kept at `medium` rather than discarded.

3.4 Verified Skill State and Deployment

OS-DEV injects a **verified skill state** into each generation prompt after the score-ranked history and before the generation instruction.The state is regenerated text with two sections, "What Works and Why" and "What Fails and Why", under a header naming the target and the number of molecules analyzed.Each surviving skill is a bullet of the form *[pattern] + [mechanism hypothesis]* with its supporting-trace indices and its verifier label appended, e.g. `(traces: 1,3; verifier: medium)`. Low-confidence skills are *kept visible* rather than silently dropped: the label lets the

optimizer weigh a skill by its evidence instead of treating every distilled hypothesis as equally reliable, while still leaving uncertain guidance where the optimizer can ignore it if contradicted by the raw history. Together with the skill state, OS-DEV appends a one-line **stage hint** that shifts from exploration (diverse scaffolds and functional groups) to refinement of promising scaffolds to fine-tuning the best candidates, as the fraction ρ of the oracle budget spent crosses 0.2 and 0.8. Finally, a separate **chemical validity filter** (RDKit parse; MW$\leq$650, logP$\leq$10, HBD$\leq$20, HBA$\leq$30) runs on every generated candidate *before* the oracle and drops those it rejects without spending budget; rejected candidates are returned to the failure analyst as invalid examples. Threshold values are fixed; Section 4.3 ablates the filter as a general budget-preservation mechanism that is independent of the skill pipeline and improves each baseline in turn (Figure 6a).

3.5 Refresh: Replacement-Based Guidance Update

Because optimization is non-stationary, a rule that helped during early exploration can hurt during late refinement, so the deployed guidance must be refreshed. OS-DEV checks two conditions after each generation round. The **periodic trigger** fires every T_{update}=2 rounds. The **plateau trigger** fires when the incumbent has failed to improve for K=5 consecutive rounds. If either trigger fires (including both simultaneously), OS-DEV re-runs the full pipeline (trace re-encoding, dual analysts, verifier) over the enlarged history and *replaces* the previous skill text with the newly generated one; a plateau-triggered refresh also resets the plateau counter. This replacement policy, rather than incrementally editing the existing text, keeps the deployed state aligned with the latest evidence and is the point of contrast with incremental playbook methods such as ACE (Zhang et al., 2026), compared empirically in Section 4.2 (Table 1).

4 Experiments

Our experiments address two questions: whether OS-DEV improves molecular optimization under a fixed oracle budget, and whether its gains can be traced to the proposed representation, distillation, and verification components.

4.1 Experimental Setup

Benchmark. DOCKSTRING (García-Ortegón et al., 2022) provides 58 protein targets with precomputed Vina (Trott & Olson, 2010) docking scores and a Python API for evaluating novel molecules. Targets span seven families: Kinase (22), Enzyme (12), Nuclear Receptor (9), Protease (7), GPCR (5), Cytochrome P450 (2), and Chaperone (1). We follow the BOPRO protocol and use the objective of Equation 1. Each target uses 200 oracle evaluations. All 58 targets use 3 random seeds each (174 runs per method). We first average the seeds within each target and then average the 58 target means for the overall score; family scores average the targets in each family. Per-target means and standard deviations for every target are reported in Appendix C.

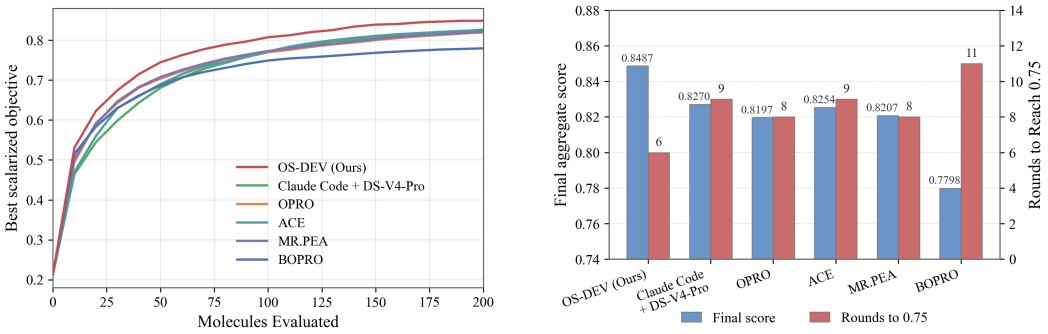
Models. The primary experiments use Qwen3.5-4B (Qwen Team, 2026) for candidate generation, skill distillation, and skill verification. Appendix D reports additional results for the same pipeline instantiated at the 2B and 9B scales.

Baselines. We compare OS-DEV with OPRO (Yang et al., 2024), BOPRO (Agarwal et al., 2025), MR.PEA (Sun & Wu, 2026), and ACE (Zhang et al., 2026) under the same Qwen3.5-4B base model, and additionally report a high-capacity agentic reference that combines the Claude Code (Anthropic, 2026) harness with the DeepSeek-V4-Pro (DeepSeek-AI, 2026) model.

Comparison protocol. All methods share the same warmstart of 3 evaluations, which does not count toward the 200-evaluation budget, and the same decoding hyperparameters, producing up to ten candidates per round. OS-DEV discards candidates rejected by its chemical validity filter without replacement or oracle evaluation, and repeated SMILES are skipped without consuming budget; its analyst and verifier calls add LLM inference but no oracle evaluations; since oracle calls are clearly more expensive than LLM inference in the molecular-optimization setting, the primary comparison controls the expensive docking budget rather than total inference cost. We use $T_{\text{update}} = 2$, $K = 5$, $\gamma = 0.02$, and $\tau = -0.02$ throughout the main experiments.

Table 1: Final scores on DOCKSTRING (58 targets), by target family. Within each row the best score is set in bold and the second-best is underlined.

Family	OS-DEV (Ours)	Claude Code	OPRO	ACE	MR.PEA	BOPRO
Kinase	**0.8433**	0.8425	0.8138	0.8229	0.8107	0.7641
Enzyme	**0.8566**	0.8268	0.8381	0.8406	0.8327	0.7898
Nucl. Rec.	0.8191	**0.8219**	0.8129	0.8190	0.7942	0.7736
Protease	**0.7970**	0.7632	0.7597	0.7624	0.7835	0.7262
GPCR	**0.9383**	0.8639	0.8617	0.8653	0.8819	0.8570
CYP450	**0.9001**	0.7912	0.8750	0.8773	0.8962	0.8456
Chap.	**0.9497**	0.8679	0.8907	0.8930	0.9389	0.9165
Overall	**0.8487**	0.8270	0.8197	0.8254	0.8207	0.7798



(a) (b)

Figure 2: Main results. (a) Aggregate best-so-far score by molecules evaluated, over all 58 targets. (b) Final score (blue) and the number of generation rounds required to reach 0.75 (red).

4.2 Main Results

Final scores. Table 1 presents per-family results. OS-DEV reaches an overall score of 0.8487 and improves on every comparator; the largest margin is 0.0689 over BOPRO under the same base model (BOPRO's original study, with a 123B model, reports 0.835 (Agarwal et al., 2025)), and the smallest 0.0217 over the Claude Code + DeepSeek-V4-Pro agentic reference. It ranks first in 6 of 7 families; the exception is Nuclear Receptor, where OS-DEV is second, marginally behind the agentic reference (0.8219 vs. 0.8191). Its largest gains over the strongest competing method in each family occur on GPCR (+0.0564), Enzyme (+0.0160), and Protease (+0.0135); the per-target breakdown appears in Appendix C.

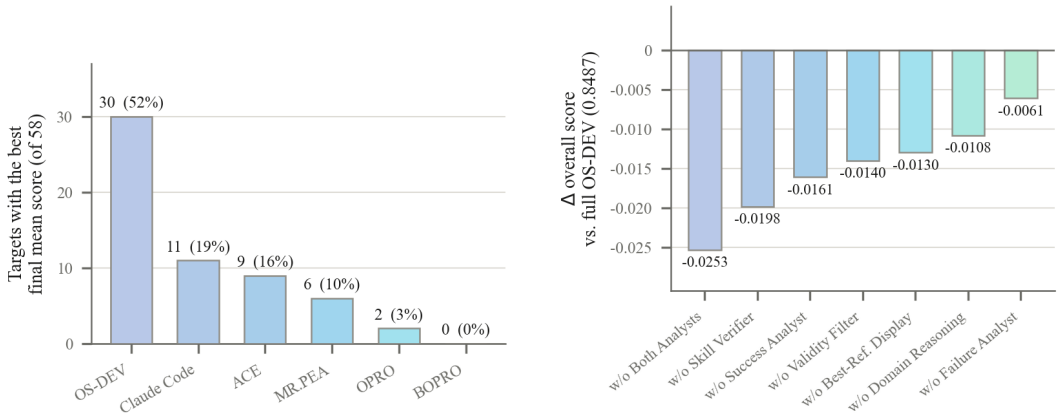
Optimization progress. Figure 2a shows the aggregate best-so-far score by molecules evaluated. OS-DEV reaches 0.75 after six generation rounds, versus eight for OPRO and MR.PEA, nine for Claude Code + DeepSeek-V4-Pro, and eleven for BOPRO. The first verified skill becomes available in the third generation round, after which OS-DEV maintains a higher aggregate curve.

Per-target consistency. The advantage is not confined to a few high-scoring targets: OS-DEV attains the highest final mean score on 30 of 58 targets — more than the other five methods combined — and wins all five GPCR targets and eight of twelve Enzyme targets (Figure 3a).

4.3 Ablation Studies

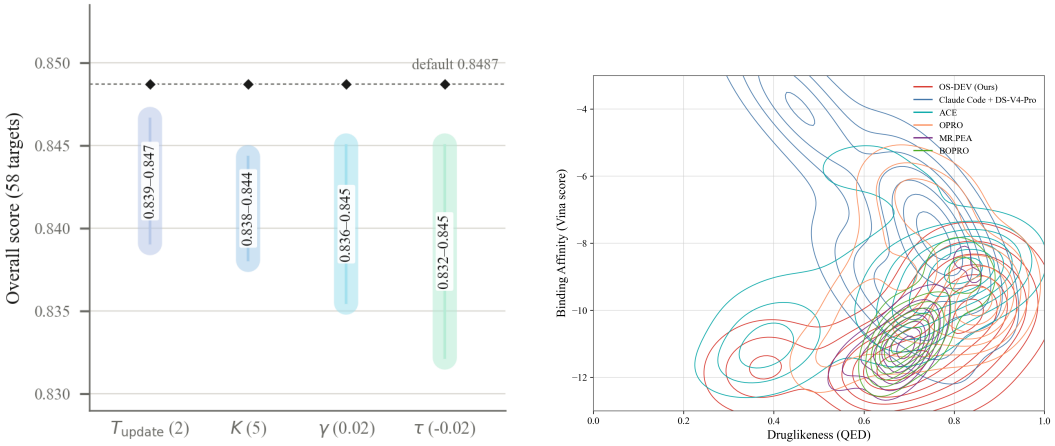
We evaluate one-at-a-time variants on the full 58-target suite under the same 200-evaluation oracle budget; Figure 3b summarizes all of them.

Best-referenced trace display. We test whether the incumbent–candidate contrast is useful independently of the full pipeline, by removing it from OS-DEV and, in the opposite direction, adding it



(a) (b)

Figure 3: Summary of the main comparisons. (a) Per-target win rate: the number of the 58 DOCK-STRING targets on which each method attains the best final mean score, with that method's share of the 58 targets in parentheses. (b) Component ablations on the full 58-target suite, as the change in overall score relative to full OS-DEV (0.8487); each row removes one component.



(a) (b)

Figure 4: Parameter sensitivity and exploration behaviour. (a) Sensitivity to the four pipeline parameters; for each control the bar spans the alternative settings tested and the diamond marks the default, which is the best setting in every case. (b) Joint QED–Vina density contours of molecules evaluated on ADORA2A.

to OPRO without any distillation. Removing the display reduces OS-DEV from 0.8487 to 0.8357, while adding the same display to OPRO raises OPRO from 0.8197 to 0.8244 without introducing analysts or a verifier. The two directions show that exposing the contrast helps on its own, but the much larger gap between OPRO and full OS-DEV requires the distillation and verification stages that the remainder of this section describes.

Analyst contribution. Against the no-analyst baseline (Figure 3b), the failure-only and success-only variants reach 0.8326 and 0.8426, respectively, and full OS-DEV reaches 0.8487. Both directions of evidence therefore carry useful information, and keeping both analyst views beats retaining either one alone, so both are used throughout.

Evidence-aware verification. Disabling the verification stage while keeping the dual analysts and trace display fixed drops the overall score from 0.8487 to 0.8289, a loss of 0.0198 — larger than removing either analyst, so the review contributes beyond extracting the skill itself.

Domain reasoning. A variant in which the analysts report the observed structural differences without the mechanistic hypothesis, and the verifier correspondingly drops its mechanism-plausibility criterion, reaches 0.8379 against 0.8487 for the full model, so the chemical rationale contributes more than a restatement of descriptor changes.

Validity filter. The chemical validity filter screens candidates before the oracle. Removing it lowers OS-DEV from 0.8487 to 0.8347, and adding it to OPRO, BOPRO, MR.PEA, and ACE raises all four as well (Appendix D).

4.4 Sensitivity and Model Scaling

The default settings are consistently strongest: $T_{\text{update}} = 2$ attains 0.8487 versus 0.8391–0.8466 for alternatives, and $K = 5$ attains 0.8487 versus 0.8381–0.8443 (Figure 4a). Nearby settings remain competitive for γ (0.8355–0.8450) and τ (0.8322–0.8450), indicating the method is not brittle to threshold choice; complete per-family tables for all four controls appear in Appendix D (Tables 7–10). A round-start incumbent control, which re-anchors each trace to the incumbent at the start of its round rather than the pre-evaluation best, changes the overall score by less than 0.002 (Table 11, appendix), confirming that the specific anchoring convention is not responsible for the reported gains. Scaling the shared generator, analysts, and verifier across the 2B, 4B, and 9B variants is reported in Figure 6b (appendix).

4.5 Case Study: Exploration Behavior on ADORA2A

Figure 4b examines the joint QED–Vina distribution of molecules evaluated for ADORA2A. OS-DEV places more density in the region with QED between 0.6 and 0.9 and Vina between -12 and -8, which receives a high composite objective because it combines drug-likeness and predicted binding affinity, whereas the model-matched OPRO baseline remains more dispersed. We select ADORA2A as a representative target to illustrate how the proposal distribution shifts when verified, trajectory-derived guidance is present in the prompt.

4.6 Discussion and Scope

Together, the ablations attribute OS-DEV's gain to a pipeline rather than any single prompt phrase: the trace display helps by itself, both analyst directions are needed, the verifier accounts for the largest drop of any single component, while removing both analysts costs more still because it leaves no distilled skills, and domain reasoning and validity screening add complementary value beyond descriptor restatements and budget saving.

OS-DEV does not train a transferable medicinal-chemistry model. Its skills are target- and run-specific, are regenerated from the evidence accumulated within the current run, and are used only as in-context guidance. This narrow scope is useful when training is unavailable and observations are expensive, but it also creates limitations. The analyst may assign an incorrect mechanistic rationale to a real score association; the verifier assesses empirical support, not pose-level or wet-lab validity, and a docking oracle may reward artifacts that would not survive pose inspection; and three seeds do not fully characterize stochastic variation. Whether a verified textual rule predicts held-out improvements within the same target, transfers to related targets, or is supported by structural or experimental evidence is left to future work.

5 Conclusion

We study molecular optimization in the cold-start regime, where prior knowledge is hardest to apply and oracle evaluations are scarce, and propose OS-DEV: a training-free feedback layer that distills each run's sparse feedback into skills and re-checks them with a role-independent verifier before they enter the prompt. On all 58 DOCKSTRING targets OS-DEV reaches 0.8487, ahead of ACE (0.8254) and of a substantially larger agentic reference (0.8270), with ablations attributing the gain to the trace representation, the dual analysts, the evidence-aware verifier, and domain reasoning. The distilled mechanisms remain optimization hypotheses rather than pose- or wet-lab-validated explanations, and whether verified skills transfer to related targets is a natural next step.

AI USE STATEMENT

In this work, we used generative AI tools only for drafting and revising parts of the manuscript text (abstract, introduction, related work, and method exposition), for preparing LaTeX source, and for summarising related literature during the research process. We have not used generative AI tools to generate experimental data or to fabricate results. All experimental results reported in this paper were obtained by running the described methods. We reviewed all AI-assisted text, checked the reported numerical claims against the recorded experimental outputs, and take responsibility for the final content of this work, including text and claims produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Our method is a training-free feedback layer built on public components (DOCKSTRING (García-Ortegón et al., 2022) for the oracle, AutoDock Vina (Trott & Olson, 2010) for docking, RDKit (Landrum, 2016) for molecular validity filtering, and an open-weight Qwen3.5-4B generator). The method does not update model parameters and requires no external training corpus, so a run is fully specified by the evaluation protocol in Section 4.1, the distillation and verification prompts in the appendix, and the random seed. All 58 DOCKSTRING targets use 3 seeds each; warmstart, oracle accounting, and budget use are described in Section 4.1. We will release code and prompts, and provide anonymous source code as supplementary material.

REFERENCES

Dhruv Agarwal, Manoj Ghuhan Arivazhagan, Rajarshi Das, Sandesh Swamy, Sopan Khosla, and Rashmi Gangadharaiah. Searching for optimal solutions with LLMs via Bayesian optimization. In *International Conference on Learning Representations (ICLR)*, Singapore, 2025.

Anthropic. Claude code. Agentic coding tool; documentation at `https://code.claude.com/docs`, 2026.

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

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

Jiangyu Chen and Ban Yi. Evidence-gated LLM priors for multi-objective Bayesian optimization. arXiv preprint arXiv:2606.01730, 2026.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. arXiv preprint arXiv:2606.19348, 2026.

Carl Edwards, Chi Han, Gawon Lee, Thao Nguyen, Sara Szymkuć, Chetan Kumar Prasad, Bowen Jin, Jiawei Han, Ying Diao, Ge Liu, Hao Peng, Bartosz A. Grzybowski, Martin D. Burke, and Heng Ji. mCLM: A modular chemical language model that generates functional and makeable molecules. In *International Conference on Learning Representations (ICLR)*, Rio de Janeiro, Brazil, 2026.

Wenhan Gao, Xiran Fan, Chin-Chia Michael Yeh, Jiarui Sun, Yuzhong Chen, Menghai Pan, Mahashweta Das, and Yi Liu. Feedback to reasoning: LLM-assisted molecular optimization with domain feedback and historical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12735–12754, San Diego, CA, 2026.

Miguel García-Ortegón, Gregor N. C. Simm, Austin J. Tripp, José Miguel Hernández-Lobato, Andreas Bender, and Sergio Bacallado. DOCKSTRING: Easy molecular docking yields better benchmarks for ligand design. *Journal of Chemical Information and Modeling*, 62(15):3486–3502, 2022.

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*, pp. 11544–11573, Suzhou, China, 2025.

Greg Landrum. RDKit: Open-source cheminformatics software, 2016. URL https://www.rdkit.org/.

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. In *International Conference on Learning Representations (ICLR)*, Rio de Janeiro, Brazil, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, New Orleans, LA, 2023.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2Skill: Distill trajectory-local lessons into transferable agent skills. arXiv preprint arXiv:2603.25158, 2026.

Qwen Team. Qwen3.5 small model series (Qwen3.5-0.8B/2B/4B/9B). Open-weight release; Hugging Face collection https://huggingface.co/collections/Qwen/qwen35, 2026.

Nian Ran, Yue Wang, Xiaoyuan Zhang, Zhongzheng Li, Qingsong Ran, Wenhao Li, and Richard Allmendinger. ExLLM: Experience-enhanced LLM optimization for molecular design and beyond. arXiv preprint arXiv:2502.12845, 2025.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, New Orleans, LA, 2023.

Yuran Sun and Chuan Wu. Prompt optimization with minimal unlabeled input via meta-reasoning. In *International Conference on Machine Learning (ICML)*, Seoul, South Korea, 2026.

Oleg Trott and Arthur J. Olson. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, 31(2):455–461, 2010.

Ziqing Wang, Yibo Wen, William Pattie, Xiao Luo, Weimin Wu, Jerry Yao-Chieh Hu, Abhishek Pandey, Han Liu, and Kaize Ding. POLO: Preference-guided multi-turn reinforcement learning for lead optimization. arXiv:2509.21737, 2025. URL https://arxiv.org/abs/2509.21737.

Ziqing Wang, Yibo Wen, Abhishek Pandey, Han Liu, and Kaize Ding. MolMem: Memory-augmented agentic reinforcement learning for sample-efficient molecular optimization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 43694–43712, San Diego, CA, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, Vienna, Austria, 2024.

Jiannan Yang, Veronika Thost, Xiang Ling, and Tengfei Ma. Oracle-budgeted molecular optimization with short-term graph memory. arXiv preprint arXiv:2607.28437, 2026.

Boqin Yuan, Yue Su, Renchu Song, Sen Yang, and Jing Qin. ClawTrace: Cost-aware tracing for LLM agent skill distillation. In *Agent Skills Workshop at the ACM Conference on AI and Agentic Systems (CAIS)*, San José, CA, 2026.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations (ICLR)*, Rio de Janeiro, Brazil, 2026.

Jialiang Zhu, Gongrui Zhang, Xiaolong Ma, Lin Xu, Miaosen Zhang, Ruiqi Yang, Song Wang, Kai Qiu, Zhirong Wu, Qi Dai, Ruichun Ma, Bei Liu, Yifan Yang, Chong Luo, Zhengyuan Yang, Linjie Li, Lijuan Wang, Weizhu Chen, Xin Geng, and Baining Guo. RE-TRAC: REcursive TRAjectory compression for deep search agents. arXiv:2602.02486, 2026. URL https://arxiv.org/abs/2602.02486.

A Prompt Templates and Optimization Algorithm

This appendix provides the full OS-DEV optimization procedure and the prompt templates used by the success analyst, the failure analyst, and the skill verifier.

A.1 Optimization Algorithm

Algorithm 1 OS-DEV online optimization loop (round-based plateau counter)

Evaluate the shared warmstart; store history H and pre-evaluation incumbent records
Set (x^*, y^*) to the best warmstart result; $b \leftarrow B$
$S \leftarrow \emptyset$; invalid examples $I \leftarrow \emptyset$; round $r \leftarrow 0$; plateau count $p \leftarrow 0$
repeat
    $r \leftarrow r + 1$; save round-start best $y_{\text{start}} \leftarrow y^*$
    Score-rank H; generate ≤ 10 candidates C conditioned on H and S
    for each candidate $x \in C$ **do**
        if $b = 0$ **then**
            break
        end if
        if x fails chemical validity screening **then**
            Add x to I; **continue**
        end if
        if x duplicates an evaluated molecule **then**
            continue
        end if
        Save pre-evaluation incumbent $(x_{\text{pre}}, y_{\text{pre}}) \leftarrow (x^*, y^*)$
        $y \leftarrow f_p(x)$; $b \leftarrow b - 1$; append (x, y) and $(x_{\text{pre}}, x, y - y_{\text{pre}})$ to history
        if $y > y^*$ **then**
            $(x^*, y^*) \leftarrow (x, y)$
        end if
    end for
    if $y^* > y_{\text{start}}$ **then**
        $p \leftarrow 0$
    else
        $p \leftarrow p + 1$
    end if
    if $b > 0$ and ($r \bmod T_{\text{update}} = 0$ or $p \geq K$) **then**
        Set plateau flag $q \leftarrow (p \geq K)$; form best-referenced traces from stored records
        Run the success and failure analysts on their selected traces and invalid examples I
        Review proposed skills using the evidence context specified in Section 3.3
        Replace S with the new labeled skill state, including the current stage hint
        if q **then**
            $p \leftarrow 0$
        end if
    end if
until $b = 0$

A.2 Prompt Templates

This section reproduces the prompt templates used at each stage. The generation prompt is the OPRO-style instruction unchanged from the baseline; OS-DEV only inserts the verified skill state between the scored history and the generation instruction. All distillers and the verifier run on the same base model as the generator.

Generation prompt (OPRO-style).

```
Your task is to find the optimal drug molecule that has both a high druglikeness
(QED) as well as a strong binding affinity (vina) to the target protein. While
both properties are important, binding affinity carries four times the scalar
```

```
weight of QED. Here is the target protein: {target}.

Here are your top previous guesses (from best to worst):
Molecule: {SMILES}
Score: {score}
... (up to 10 entries)

[verified skill state]        <- inserted here after the scored history

Now, guess exactly n=10 new molecule(s) that could give you a target score of
{target_score}. (Note: give only a list of SMILES strings in the provided JSON
format, e.g. {"response": ["SMILES1", ...]})
```

Note: the target score is set to the current best observed score at each round, so the model is asked to exceed the incumbent rather than to reach a fixed constant. Before the first refresh no skill block is present.

Success analyst. *System:* "You are a medicinal chemistry expert. Think step by step about protein–ligand interactions. Be specific and mechanistic."

```
You are a medicinal chemistry expert analyzing molecular optimization for
protein {target}.

Below are optimization traces where the proposed molecule IMPROVED the score
over the current best.

{traces}      # incumbent -> candidate, composite scores, QED/Vina, +delta

Reason about WHY each candidate achieves a better composite score: which
physicochemical property or interaction could explain the gain, and what does
this reveal about what the {target} binding pocket rewards.

Then synthesize 3-5 reusable optimization skills.

For each skill, mark which traces support it, e.g.:
```skills
1. [Add amide linkage] + [engages hinge H-bonds] | traces: 1,3
2. [Expand aromatic core] + [fills hydrophobic pocket] | traces: 2,4,5
```

Think step by step first, then put ONLY the final skills in a ```skills block.
```

Failure analyst. *System:* "You are a medicinal chemistry expert. Think step by step about why modifications fail. Be specific and mechanistic."

```
You are a medicinal chemistry expert analyzing molecular optimization for
protein {target}.

Below are optimization traces where the proposed molecule FAILED to improve the
score (decreased or invalid).

{traces}      # degraded contrasts and syntactically invalid SMILES

Reason about WHY each candidate lost score, and what kinds of candidates should
be avoided for {target}. Then synthesize 3-5 avoidance skills.

For each skill, mark which traces support it, e.g.:
```skills
1. [Avoid removing amide] + [loses hinge H-bond] | traces: 1,2
2. [Avoid invalid ring closure] + [breaks parsing] | traces: 3,4
```

Think step by step first, then put ONLY the final skills in a ```skills block.
```

Skill verifier. *System:* "You are a rigorous, skeptical reviewer. Only mark a skill high-confidence when its cited traces clearly and consistently support it."

```
You are an independent reviewer auditing distilled optimization skills for
protein {target}.

An optimization analyst distilled the skills below from real optimization
traces. Your job: judge whether each skill is genuinely supported by the traces
it cites, or is speculation. Be skeptical.

Here are all the traces (each trace shows the incumbent -> candidate with
composite score change):
```

```
{traces}

Now audit each skill:

{skills_list}    # each skill with its cited trace indices

For each skill, judge:
1. Evidence sufficiency: do the cited traces consistently support the claimed
   effect? Too few examples, or examples that contradict the claim, lower the
   confidence.
2. Mechanism plausibility: is the proposed mechanism chemically reasonable, or
   vague/fabricated?
3. Target specificity: does the skill say something specific to {target}, or is
   it generic advice that holds for any protein?

Respond with exactly one line per skill, in this format:
Skill <N>: Confidence: <high|medium|low> -- <one-sentence justification>
```

If the verifier call fails or its output cannot be parsed, skills are retained at `medium`; the verification stage never aborts the run.

B Representative Skills and Verification Records

The following snapshots illustrate the textual form returned by the analysts and the verdict the verifier attaches to each skill. Phrases such as "adding" and "replacing" summarize a difference observed in a best-referenced pair; they do not imply that the candidate was produced by a verified single-edit path, and the interaction rationales are hypotheses used to guide optimization rather than conclusions from docking poses or experiments. Each skill is shown with its supporting-trace indices and the verifier's confidence label, exactly as it is inserted into the generation prompt.

Example 1 (success skills, SRC kinase).

1. **Para-fluorination of phenyl rings.** Multiple transitions adding fluorine at the para position showed consistent improvements. The para-fluoro substituent fills a small hydrophobic sub-pocket while the C–F bond may engage weak polar interactions with backbone carbonyls. (`traces: 1,4; verifier: high`)
2. **Pyridine-for-phenyl replacement.** Replacing a phenyl ring with pyridine at solvent-exposed positions improved both QED and Vina. The nitrogen may enable water-mediated hydrogen bonds with the hinge region. (`traces: 2,5; verifier: medium`)
3. **Truncation of flexible alkyl chains.** Shortening alkyl substituents ($\geq$4 carbons) consistently raised QED without sacrificing Vina, suggesting an entropic penalty without compensatory contacts. (`traces: 3; verifier: medium`)

Example 2 (avoidance skills, SRC kinase).

1. **Meta-substitution on core aromatic rings.** Adding substituents at the meta position of the central scaffold consistently degraded scores, likely through steric clashes with the gatekeeper residue. (`traces: 1,2; verifier: medium`)
2. **Excessive halogenation.** Molecules with $\geq$3 halogen atoms showed reduced QED (below 0.5) and elevated logP, crossing standard drug-likeness thresholds without compensatory affinity gains. (`traces: 4; verifier: low`)

The label is a joint judgment over the three criteria, not a function of evidence count alone: a skill supported by a single trace is labelled at most `medium` in these snapshots, and `low` marks cited evidence that is thin or contradicted (Example 2). A `low` skill is still shown, carrying its label so that the optimizer can discount it. The verifier's one-sentence justification is not reproduced here.

C Per-Target Results and Uncertainty

Tables 3–6 report the mean and standard deviation over 3 random seeds for each of the 58 DOCKSTRING targets. Across the 58 targets, OS-DEV obtains the highest mean score on 30 targets, followed by the Claude Code + DeepSeek-V4-Pro reference (11), ACE (9), MR.PEA (6), OPRO (2), and BOPRO (0). OS-DEV wins all 5 GPCR targets and 8 of 12 Enzyme targets.

Table 2: Paired per-target comparison over 58 targets. Differences are OS-DEV minus the comparator; Win/Loss counts refer to OS-DEV.

Comparator	Mean diff.	Median	Win/Loss	95% CI	p (paired t)
Claude Code + DeepSeek-V4-Pro	$+0.0217$	$+0.0230$	40/18	$[+0.0111, +0.0322]$	2.0×10^{-4}
ACE	$+0.0233$	$+0.0239$	38/20	$[+0.0141, +0.0325]$	7.7×10^{-6}
MR.PEA	$+0.0280$	$+0.0335$	45/13	$[+0.0192, +0.0367]$	4.8×10^{-8}
OPRO	$+0.0290$	$+0.0317$	44/14	$[+0.0199, +0.0379]$	5.4×10^{-8}
BOPRO	$+0.0689$	$+0.0729$	57/1	$[+0.0603, +0.0772]$	1.1×10^{-22}

Table 3: Per-target scores: Kinase (22 targets). Mean ± standard deviation over 3 random seeds.

Target	OS-DEV	Claude Code	OPRO	ACE	MR.PEA	BOPRO
ABL1	0.9550±0.0203	0.9083±0.0193	0.8762±0.0259	0.8805±0.0249	0.8647±0.0738	0.8334±0.0108
AKT1	0.9285±0.0357	0.8815±0.0473	0.8511±0.0568	0.8542±0.0253	0.8525±0.0201	0.8111±0.0851
AKT2	0.8321±0.0734	0.8334±0.0342	0.8045±0.0337	0.8066±0.0360	0.8050±0.1541	0.7617±0.0520
CDK2	0.9078±0.0277	0.8740±0.0405	0.8401±0.0226	0.8426±0.0452	0.8392±0.1381	0.8038±0.0278
CSF1R	0.7867±0.0365	0.8177±0.0323	0.7917±0.0461	0.7936±0.0447	0.7729±0.0489	0.7358±0.0331
EGFR	0.8657±0.0359	0.8600±0.0363	0.8341±0.0414	0.8368±0.0203	0.8677±0.0840	0.7731±0.0379
FGFR1	0.9372±0.0620	0.9036±0.0693	0.8725±0.0477	0.8754±0.0690	0.8508±0.0539	0.8065±0.0831
IGF1R	0.8648±0.0467	0.8477±0.0966	0.8222±0.0317	0.8259±0.0561	0.8187±0.0519	0.7786±0.1332
JAK2	0.8052±0.0522	0.8216±0.0661	0.7844±0.0648	0.7866±0.0388	0.7967±0.1411	0.7391±0.0360
KDR	0.8729±0.0716	0.8510±0.0282	0.8380±0.0160	0.8431±0.0232	0.8339±0.1870	0.7761±0.0385
KIT	0.8373±0.0225	0.8352±0.0208	0.8198±0.0584	0.8393±0.0304	0.7959±0.0240	0.7672±0.0205
LCK	0.7813±0.0184	0.8101±0.0189	0.7718±0.0662	0.7746±0.0361	0.7679±0.0346	0.7338±0.0832
MAP2K1	0.8799±0.0067	0.8548±0.0919	0.8340±0.0042	0.8365±0.0213	0.8231±0.0668	0.7796±0.0714
MAPK1	0.8351±0.0260	0.8336±0.0037	0.8011±0.0432	0.8371±0.0495	0.8119±0.0431	0.7552±0.0169
MAPK14	0.7508±0.0558	0.7962±0.0310	0.7685±0.0228	0.7738±0.0296	0.7644±0.0548	0.7108±0.0274
MAPKAPK2	0.7609±0.0534	0.7997±0.0710	0.7742±0.1007	0.7776±0.0303	0.7700±0.0713	0.7162±0.0376
MET	0.8958±0.0537	0.8674±0.0375	0.8356±0.0151	0.8374±0.0432	0.8340±0.0811	0.7883±0.0421
PLK1	0.8497±0.0233	0.8494±0.0185	0.8172±0.0223	0.8517±0.0356	0.8044±0.0070	0.7596±0.0602
PTK2	0.8319±0.0686	0.8440±0.0349	0.8089±0.0381	0.8208±0.0470	0.8081±0.0973	0.7556±0.0260
PTPN1	0.7207±0.0365	0.7668±0.0429	0.7421±0.0544	0.7492±0.0348	0.7443±0.0096	0.7049±0.0582
ROCK1	0.8667±0.0396	0.8631±0.0462	0.8345±0.0636	0.8687±0.0181	0.8228±0.0919	0.7799±0.0418
SRC	0.7866±0.0373	0.8169±0.0309	0.7815±0.0506	0.7919±0.0114	0.7863±0.1079	0.7404±0.0857

Table 2 reports the paired per-target comparison against each baseline and the agentic reference. Interval and p-value are computed on the same quantity, the *mean* per-target difference: a percentile bootstrap over the 58 targets (20,000 resamples) and a two-sided paired t-test. Every interval excludes zero and every comparison survives a Bonferroni correction for the five tests.

C.1 Representative Optimized Molecules

Figure 5 displays one representative molecule generated by OS-DEV for a target from each of the seven protein families. These structures exhibit common drug-like features—amide linkages, aromatic rings, heterocycles, and halogen substituents—and their composite scores range from 0.865 to 0.961, confirming that the optimization reaches chemically plausible, high-scoring candidates across

Table 4: Per-target scores: Enzyme (12). Mean ± standard deviation over 3 random seeds.

Target	OS-DEV	Claude Code	OPRO	ACE	MR.PEA	BOPRO
ACHE	0.9598±0.0172	0.8802±0.0129	0.8899±0.0259	0.8924±0.0885	0.8910±0.0025	0.8429±0.0476
CA2	0.7429±0.0893	0.7583±0.0066	0.7690±0.0479	0.7688±0.0029	0.7688±0.0667	0.7202±0.0259
DHFR	0.9244±0.0367	0.8616±0.1119	0.8849±0.1095	0.8885±0.0417	0.8613±0.0453	0.8244±0.0908
GBA	0.8770±0.0436	0.8427±0.0809	0.8515±0.0365	0.8530±0.0356	0.8448±0.0466	0.8029±0.0764
HMGCR	0.6802±0.0743	0.7291±0.0365	0.7413±0.0703	0.7460±0.0294	0.7328±0.0204	0.7022±0.0889
HSD11B1	0.8620±0.1556	0.8292±0.0442	0.8476±0.0489	0.8497±0.0173	0.8640±0.0637	0.7866±0.0506
MAOB	0.8874±0.0625	0.8399±0.0469	0.8620±0.0609	0.8646±0.0098	0.8359±0.0337	0.8074±0.0449
NOS1	0.8453±0.0394	0.8179±0.0512	0.8265±0.0695	0.8303±0.0258	0.8141±0.0704	0.7886±0.0049
PARP1	0.8890±0.0564	0.8554±0.0285	0.8487±0.0330	0.8516±0.0251	0.8543±0.0507	0.8039±0.0264
PDE5A	0.9056±0.1122	0.8587±0.0027	0.8645±0.0010	0.8689±0.0226	0.8638±0.0350	0.8231±0.0353
PTGS2	0.9062±0.0385	0.8583±0.0261	0.8647±0.0252	0.8665±0.0240	0.8570±0.0741	0.8194±0.0612
REN	0.7994±0.0516	0.7908±0.1705	0.8068±0.1320	0.8066±0.0562	0.8040±0.1790	0.7556±0.0134

Table 5: Per-target scores: Nuclear Receptor (9). Mean ± standard deviation over 3 random seeds.

Target	OS-DEV	Claude Code	OPRO	ACE	MR.PEA	BOPRO
AR	0.7313±0.0936	0.7666±0.0679	0.7685±0.0348	0.7718±0.0072	0.7356±0.1151	0.7264±0.0408
ESR1	0.7740±0.0145	0.8048±0.0322	0.7824±0.0821	0.7850±0.1211	0.7634±0.0739	0.7472±0.0157
ESR2	0.7667±0.0366	0.7994±0.1021	0.7867±0.0630	0.7887±0.0352	0.7501±0.0508	0.7496±0.0400
NR3C1	0.8678±0.0219	0.8438±0.1244	0.8361±0.0408	0.8392±0.0322	0.8212±0.0546	0.7974±0.0853
PPARA	0.8384±0.0700	0.8344±0.0950	0.8099±0.0841	0.8404±0.0108	0.8037±0.0503	0.7883±0.0399
PPARD	0.8008±0.0226	0.8085±0.0169	0.8082±0.0148	0.8107±0.0127	0.7853±0.1000	0.7665±0.0129
PPARG	0.8712±0.1076	0.8576±0.0737	0.8437±0.0879	0.8474±0.0652	0.8732±0.1206	0.7996±0.0300
PGR	0.7959±0.0406	0.8071±0.0709	0.8049±0.0685	0.8078±0.0159	0.7807±0.0502	0.7603±0.0205
THRB	0.9258±0.0444	0.8747±0.0486	0.8755±0.0357	0.8797±0.0619	0.8350±0.0135	0.8275±0.0138

Table 6: Per-target scores: Protease (7), GPCR (5), Cytochrome P450 (2), Chaperone (1). Mean ± standard deviation over 3 random seeds.

Target	OS-DEV	Claude Code	OPRO	ACE	MR.PEA	BOPRO
Protease (7)						
ADAM17	0.8669±0.0467	0.7956±0.1003	0.7950±0.0225	0.7981±0.0285	0.8286±0.1918	0.7704±0.0297
BACE1	0.6463±0.0145	0.6755±0.1026	0.6731±0.0476	0.6753±0.0630	0.6946±0.0208	0.6420±0.0351
CASP3	0.7177±0.0230	0.7251±0.0667	0.7230±0.0426	0.7247±0.0529	0.7457±0.0138	0.6773±0.0766
DPP4	0.8714±0.0172	0.8130±0.0224	0.8083±0.0403	0.8107±0.0098	0.8231±0.0582	0.7660±0.0273
F10	0.8549±0.0411	0.7965±0.0071	0.7905±0.0356	0.7933±0.0103	0.8109±0.0835	0.7567±0.0210
F2	0.7321±0.0204	0.7218±0.0694	0.7262±0.0664	0.7291±0.0290	0.7532±0.1198	0.6901±0.0893
MMP13	0.8897±0.0333	0.8148±0.0613	0.8016±0.0317	0.8054±0.0476	0.8285±0.1026	0.7812±0.0200
GPCR (5)						
ADORA2A	0.9312±0.0343	0.8554±0.1050	0.8740±0.0563	0.8754±0.0186	0.8889±0.0226	0.8389±0.0310
ADRB1	0.9499±0.0283	0.8806±0.0118	0.8565±0.0545	0.8611±0.0525	0.8814±0.0280	0.8577±0.0664
ADRB2	0.9455±0.0161	0.8570±0.1078	0.8862±0.0470	0.8899±0.0924	0.8951±0.1261	0.8747±0.0112
DRD2	0.9470±0.0521	0.8708±0.0173	0.8518±0.0720	0.8569±0.0494	0.8630±0.0840	0.8626±0.0231
DRD3	0.9179±0.0517	0.8555±0.1114	0.8400±0.0548	0.8432±0.0358	0.8812±0.0877	0.8511±0.0499
Cytochrome P450 (2)						
CYP2C9	0.9193±0.0379	0.8132±0.1742	0.8850±0.0303	0.8878±0.0526	0.9134±0.0168	0.8542±0.0618
CYP3A4	0.8809±0.0672	0.7692±0.1040	0.8650±0.0263	0.8667±0.0245	0.8789±0.0147	0.8370±0.0394
Chaperone (1)						
HSP90AA1	0.9497±0.0126	0.8679±0.0897	0.8907±0.0646	0.8930±0.0594	0.9389±0.0173	0.9165±0.0503

diverse target classes. All molecules were generated by the Qwen3.5-4B model under the standard 200-evaluation budget.

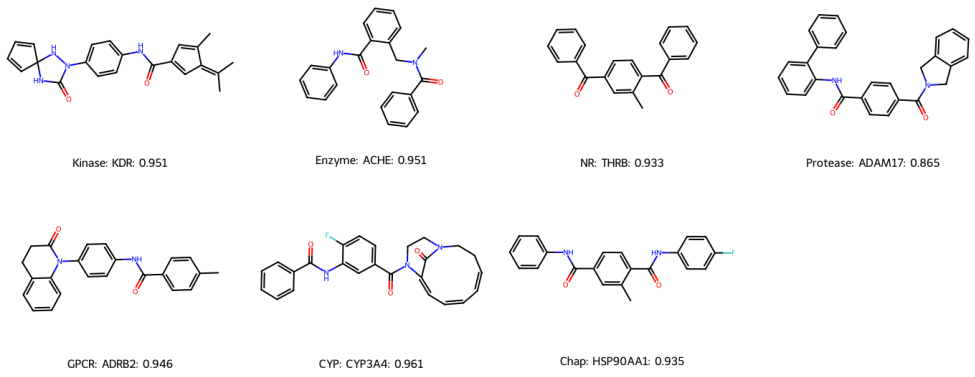


Figure 5: Representative molecules generated by OS-DEV, one per protein family. Individual molecule scores range from 0.865 (Protease, ADAM17) to 0.961 (CYP450, CYP3A4); these are not seed-averaged best scores.

Table 7: Sensitivity to update frequency T_{update} ($K = 5$ fixed).

T_{update}	Kin.	Enz.	NR	Prot.	GPCR	CYP	Chap.	Overall
1	0.8262	0.8507	0.8139	0.7920	0.9350	0.8960	0.9461	0.8391
2 (ours)	0.8433	0.8566	0.8191	0.7970	0.9383	0.9001	0.9497	0.8487
5	0.8381	0.8527	0.8220	0.7899	0.9363	0.9038	0.9463	0.8454
8	0.8408	0.8549	0.8175	0.7945	0.9368	0.8974	0.9490	0.8466

Table 8: Sensitivity to plateau patience K ($T_{\text{update}} = 2$ fixed).

K	Kin.	Enz.	NR	Prot.	GPCR	CYP	Chap.	Overall
1	0.8252	0.8519	0.8150	0.7881	0.9300	0.8948	0.9418	0.8381
2	0.8337	0.8547	0.8206	0.7940	0.9366	0.8988	0.9484	0.8443
5 (ours)	0.8433	0.8566	0.8191	0.7970	0.9383	0.9001	0.9497	0.8487
8	0.8307	0.8522	0.8178	0.7909	0.9342	0.8981	0.9475	0.8416

D Sensitivity and Model-Scale Details

This section reports the parameter-sensitivity, control, and model-scale results referenced from Section 4.4. Every configuration uses all 58 targets and 3 seeds per target.

D.1 Parameter Sensitivity

We vary one parameter at a time from the default ($T_{\text{update}} = 2$, $K = 5$, $\gamma = 0.02$, $\tau = -0.02$) while holding the others fixed.

Update frequency T_{update}. Table 7 reports the effect of the periodic update interval. The default $T_{\text{update}} = 2$ obtains 0.8487, while the alternatives range from 0.8391 to 0.8466. Updating every round performs worst, consistent with summaries being formed from less mature trajectories; less frequent updates stay competitive but delay the incorporation of new feedback. Family-level optima differ, so a single refresh schedule is not uniformly best for every family.

Plateau patience K. Table 8 reports the effect of the plateau patience. The default $K = 5$ obtains 0.8487, while the alternatives range from 0.8381 to 0.8443. Triggering after a single non-improving round performs worst, and increasing K to 8 also reduces the score. The results favor an intermediate criterion that neither reacts to every short fluctuation nor waits through extended stagnation.

Success threshold γ. Table 9 reports the effect of the success threshold. The default $\gamma = 0.02$ obtains 0.8487, while 0.01, 0.05, and 0.08 yield 0.8450, 0.8417, and 0.8355. A stricter threshold discards too many improving contrasts to form useful summaries, whereas a looser one admits weakly improving pairs.

Failure threshold τ. Table 10 reports the effect of the failure threshold. The default $\tau = -0.02$ obtains 0.8487; $\tau = -0.01$ is close at 0.8450, whereas -0.05 and -0.08 fall to 0.8397 and 0.8322. As with γ, moderate changes preserve most of the gain while strict thresholds weaken the avoidance signal.

Table 9: Sensitivity to success threshold γ ($\tau = -0.02$ fixed).

γ	Kin.	Enz.	NR	Prot.	GPCR	CYP	Chap.	Overall
0.01	0.8380	0.8540	0.8172	0.7940	0.9347	0.8965	0.9473	0.8450
0.02 (ours)	0.8433	0.8566	0.8191	0.7970	0.9383	0.9001	0.9497	0.8487
0.05	0.8319	0.8508	0.8223	0.7887	0.9314	0.8919	0.9443	0.8417
0.08	0.8261	0.8458	0.8127	0.7835	0.9248	0.8866	0.9385	0.8355

Table 10: Sensitivity to failure threshold τ ($\gamma = 0.02$ fixed).

τ	Kin.	Enz.	NR	Prot.	GPCR	CYP	Chap.	Overall
-0.01	0.8361	0.8553	0.8205	0.7933	0.9359	0.8947	0.9450	0.8450
-0.02 (ours)	0.8433	0.8566	0.8191	0.7970	0.9383	0.9001	0.9497	0.8487
-0.05	0.8290	0.8509	0.8167	0.7871	0.9325	0.8946	0.9432	0.8397
-0.08	0.8209	0.8441	0.8114	0.7790	0.9228	0.8870	0.9340	0.8322

Table 11: Round-start incumbent control ablation.

Configuration	Overall Score	Δ from Full
OS-DEV (Full)	**0.8487**	—
Round-Start Incumbent	0.8468	-0.0019

D.2 Additional Controls

Table 11 reports the round-start incumbent control: fixing the incumbent to its round-start value yields 0.8468 overall, only 0.0019 below the default protocol, so the primary contrast signal comes from cross-round comparisons rather than within-round updates. Figure 6a reports the chemical validity filter and its application to the baselines as paired bars, with the absolute scores annotated directly. The filter contributes $+0.0140$ to OS-DEV and improves every baseline ($+0.0037$ for OPRO, $+0.0065$ for MR.PEA, $+0.0073$ for BOPRO, $+0.0049$ for ACE), making it a general budget-preservation mechanism; OS-DEV without the filter still reaches 0.8347, above every filtered baseline.

D.3 Model Scaling

Figure 6b reports overall scores for the 2B, 4B, and 9B variants of the shared generator, analysts, and verifier as grouped bars. Every method improves monotonically with scale, while OS-DEV remains the best compared method at each tested scale: 0.8371, 0.8487, and 0.8526 at 2B, 4B, and 9B, respectively, leading the strongest baseline by 0.0196, 0.0233, and 0.0196. The ranking of the remaining methods is also stable across scales.

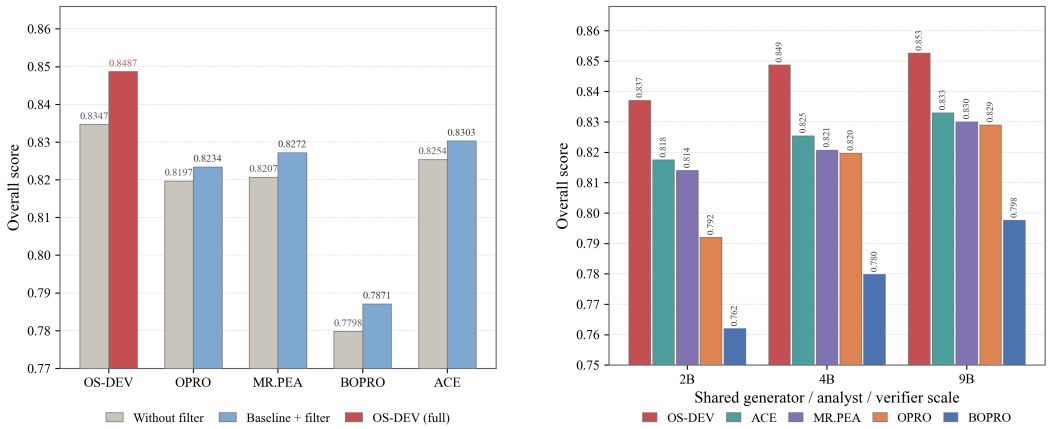
E Implementation and Computational Cost

Model roles. OS-DEV and the OPRO and MR.PEA baselines use an OPRO-style acquisition, which requires no surrogate model. MoLFormer-XL is loaded only by the BOPRO baseline, whose logEI acquisition needs candidate representations. No OS-DEV component depends on it.

Warmstart. Every method shares the same warmstart of 3 evaluations, which does not count toward the 200-evaluation budget. The optimizer begins from this warmstart, so the first refresh can already see a non-trivial history; before the first refresh no skill block is injected.

Skill distillation parameters. $\gamma = 0.02$, $\tau = -0.02$, $T_{\text{update}} = 2$, $K = 5$. Each analyst receives at most 15 entries (the 10 most recent plus 5 sampled from the remainder; all entries when fewer are available). Exploration phases are keyed to the fraction ρ of the budget spent: early ($\rho < 0.2$), mid ($0.2 \leq \rho < 0.8$), late ($\rho \geq 0.8$). All experiments use 200 oracle evaluations per target per seed.

Skill text handling. Each refresh *replaces* the previous skill text rather than merging summaries across rounds; the trajectory itself continues to grow, so later refreshes can revisit earlier records while adapting priorities to the current exploration stage. The success and failure analysts are separate calls with distinct prompts over disjoint trace subsets; the verifier is a further independent call. All three run on the generation model at temperature 1.0. The analyst and verifier calls add LLM inference but no oracle evaluations. Since oracle evaluations are the dominant cost in realistic molecular optimization while LLM inference is comparatively cheap, the reported comparison controls the docking budget, which is the practically meaningful cost axis.



(a) (b)

Figure 6: Additional controls and model scaling. (a) Chemical validity filter ablation and baseline filter controls: grey bars are the unfiltered configuration, blue bars add the filter to each baseline, and the red bar is full OS-DEV; bar labels give the absolute overall score. (b) Model-scale comparison (58 targets, 3 seeds per target): overall score of each method when the shared generator, analysts, and verifier are jointly scaled from 2B to 4B to 9B; bar labels give the exact scores.

Table 12: Hyperparameters and compute configuration.

Component	Setting
Generation model	Qwen3.5-4B (dense)
2B / 9B scales	Qwen3.5-2B / Qwen3.5-9B (dense)
Representation model	MoLFormer-XL, used only by the BOPRO baseline's surrogate
Inference framework	bfloat16, default chat template
LLM temperature / top-p / max tokens	1.0 / 0.9 / 512
Candidates per round	≤ 10
Warmstart evaluations	3 (does not count toward the budget)
Docking engine	AutoDock Vina, 300 s timeout
CPU threads	torch 2, MKL 2, OMP 2

Oracle accounting. Candidates rejected by the chemical validity filter (RDKit parse; MW ≤ 650, logP ≤ 10, HBD ≤ 20, HBA ≤ 30) and repeated SMILES are discarded without consuming the oracle budget. Rejected candidates are passed to the failure analyst as invalid examples.