What Are LLM Agents Actually Learning on Low-Budget Molecular Optimization Benchmarks?

Anonymous ACL submission

Abstract

Recent work reports large gains for LLM agents on low-budget molecular optimization, with 2–3× AUC improvements over reinforcement-learning and genetic-algorithm baselines on the PMO benchmark and similar gains on protein-targeted docking benchmarks. We investigate whether these gains reflect genuine optimization or benchmark artifacts, by re-evaluating a representative LLM agent under controlled blinding protocols and against a trivial random-walk baseline. We find that on PMO at the 50-call budget, the agent's apparent advantage collapses below classical baselines once task names and curated reference compounds are removed from its prompt; a 2×2 ablation shows that retrieval from a small reference dictionary, not iterative optimization, is the dominant driver of the headline number. On a new protein-targeted lead-optimization benchmark with an AutoDock Vina composite oracle, a single-edit halogen/methyl random-walk mutator outperforms the LLM agent, a graph-based GA, and a non-iterative GPT baseline on every aggregate metric—a property of the oracle landscape under tight budgets, not of LLM agents per se. We release both protocols (name-blinded PMO scoring; random-walk lower bound for Vina-based protein benchmarks) as standard sanity checks for future LLM-agent papers in molecular optimization.

1 Introduction

LLM agents have recently been reported to substantially outperform classical baselines on molecular optimization at small oracle budgets. On the Practical Molecular Optimization (PMO) benchmark (Gao et al., 2022) at 50 oracle calls, agents such as Krüger et al. (2026) report a sum-of-AUC of 19.85 versus 7.06 for REINVENT (Olivecrona et al., 2017)—a 2.8× improvement. Concurrent work on Kim et al. (2025) reports 15.42 summed AUC at the 1K-call PMO setting. Le et al. (2025) reports comparable gains on the ZINC molecular-editing benchmarks (Liu et al., 2024). The framing of these results is consistent: LLMs encode medicinal-chemistry priors that classical RL and GA methods do not access, and so an LLM-driven agent can navigate chemistry space more sample-efficiently.

This paper investigates whether the reported gains reflect genuine optimization ability, or whether they reflect benchmark artifacts that an LLM exploits in ways the headline numbers do not capture. We approach this from two complementary angles.

Angle 1: Task-name leakage in PMO. Of the 23 PMO tasks, 14 contain a drug name in the task identifier itself: "celecoxib_rediscovery", "osimertinib_mpo", "drd2", and similar. A pretrained LLM has very plausibly seen the SMILES of celecoxib and the chemistry of D2 receptor antagonists in its pre-training corpus. To the extent this is true, the LLM agent's PMO score reflects *retrieval* of memorised molecules rather than *optimisation*. Existing LLM-agent papers (Kim et al., 2025; Le et al., 2025; Nguyen and Grover, 2025; Krüger et al., 2026) typically include the task name in the prompt at every iteration. We re-run the same agent under a strict name-blinded protocol that replaces the task identifier with an anonymous label and disables curated reference compounds, and compare the resulting AUC to the standard setting.

Angle 2: Protein-targeted Vina benchmarks vs. trivial random walk. The natural step from surrogate PMO oracles to "real" lead optimisation is a docking-based benchmark. Several recent agents target this regime, e.g. via Passaro et al. (2025) or AutoDock Vina via DOCKSTRING (García-Ortegón et al., 2022). We construct a 50-pair, 10-target protein-binding lead-optimisation benchmark with a multi-objective composite oracle (Vina docking improvement + QED (Bickerton et al.,

2012) + Tanimoto similarity to starter (Bajusz et al., 2015)). On this benchmark we then ask whether LLM agents add value over a baseline that captures the most trivial form of local search: random single-atom mutations of the starter (halogen addition, methyl/methoxy substitution at aromatic positions). The result is sobering: within a 30-call budget, the random walk matches or exceeds the LLM agent, the GraphGA baseline, and the non-iterative GPT-direct baseline on every aggregate metric. The Vina + multi-objective composite landscape contains too many easy single-edit improvements for the budget to discriminate between agents.

Contributions. We make three contributions.

1. **Name-blinded PMO protocol and full re-evaluation (Section 4).** We run a public LLM-agent implementation on all 23 PMO tasks under a strict name-blinded protocol, with single-seed budgets of 50 oracle calls. The agent's sum top-1 AUC drops from **15.10** in the leakage-permissive setting to **6.46** in the blinded setting, falling *below* REINVENT's published 7.06. A 2×2 ablation (task-name on/off $\times$ reference-compound dictionary on/off) shows that the dominant component is the reference-compound dictionary ($\Delta = +2.18$ on a 5-task subset), with task-name leakage contributing a smaller marginal effect ($\Delta = +0.71$).

2. **Random-walk floor for protein-targeted Vina benchmarks (Section 5).** We introduce a halogen/methyl single-edit random-walk baseline. On our PROTEINLEADOPT-50 benchmark with a 30-call AutoDock Vina + composite oracle, this baseline outperforms all tested methods including the LLM agent, GraphGA, and GPT-direct. We argue this is a property of the oracle landscape, not of LLM agents.

3. **Recommendations for benchmark design (Section 6).** We propose two concrete additions to standard LLM-agent evaluation in molecular optimization: (i) report performance under a name-blinded re-run alongside the headline number, and (ii) report performance against a random-walk floor on any docking-based composite benchmark. Both protocols are cheap to run and prevent the kinds of inflated claims our experiments expose.

We do not claim a new state-of-the-art LLM agent. Our contribution is a calibrated reading of where existing low-budget LLM-agent results sit relative to leakage-free and trivial-baseline reference points, and the protocols by which future agents can be evaluated more honestly.

2 Related Work

Molecular optimization benchmarks. The Practical Molecular Optimization (PMO) benchmark (Gao et al., 2022) unifies 23 molecular-optimization tasks behind a common oracle interface and was designed to expose how few oracle calls are sufficient to converge. Classical baselines—REINVENT (Olivecrona et al., 2017), GraphGA (Jensen, 2019), GP-BO (Tripp et al., 2021)—typically require thousands of evaluations to reach high scores; PMO at the 1K and 10K-call budgets is now standard (Gao et al., 2022). PMO oracles are accessed via Therapeutics Data Commons (Huang et al., 2021). DOCKSTRING (García-Ortegón et al., 2022) adds protein-targeted AutoDock Vina objectives over 58 targets. Our PROTEINLEADOPT-50 (§5.1) builds on the DOCKSTRING tooling but moves from de novo generation to lead optimization with explicit starters and a multi-objective composite.

Classical molecular optimization. REINVENT (Olivecrona et al., 2017) fine-tunes a SMILES-based generator with policy-gradient reinforcement learning. GraphGA (Jensen, 2019) is a graph-based genetic algorithm. Augmented Memory (Guo and Schwaller, 2024) extends REINVENT with experience replay. These methods perform well at the standard 10K-call budget but converge slowly under the stricter low-budget regime that motivates LLM-agent papers.

LLM agents in chemistry. ChemCrow (M. Bran et al., 2024) integrates GPT-4 with 18 chemistry tools. Coscientist (Boiko et al., 2023) couples an LLM controller with laboratory automation. These systems frame the LLM as an orchestrator of external tools rather than as the optimizer itself; we do not interact with them directly.

LLM agents for molecular optimization. ChatDrug (Liu et al., 2024) introduced a conversational refinement loop with retrieval and domain feedback over 39 editing tasks. AgentDrug (Le et al., 2025) extends this with a nested loop: an inner RDKit `ParseError` validation loop and an outer scalar-gradient feedback loop, evaluated on ZINC-based property-editing tasks. MolLEO (Wang et al., 2025) embeds an LLM as the mutation/crossover

operator inside a genetic algorithm. LICO (Nguyen and Grover, 2025) fine-tunes a base LLM with embedding adapters and reports state-of-the-art on PMO-1K. MT-Mol (Kim et al., 2025) uses five role-specialized analyst agents wrapping curated RDKit functions and a scientist/verifier/reviewer pipeline; it reports SOTA on 17 of 23 PMO-1K tasks (sum AUC 15.42). MT-Mol's prompt explicitly includes the top-100 ZINC reference molecules per task, which is a stronger form of reference leakage than ours. MECo (Zhu et al., 2025) converts editing intentions into executable RDKit code. MolAct (Yang et al., 2025) trains a smaller open LLM with agentic reinforcement learning. SEISMO (Krüger et al., 2026) is the most directly comparable to our LLM-agent setup: a strictly online, trajectory-conditioned LLM workflow with a 50-call budget on PMO using Claude Opus 4.5, reporting a sum top-1 AUC of 19.85. SEISMO does ablate on whether the task description is shown, but does not blind *names*: even in its no-description condition the LLM may receive a structurally similar task identifier. Our blinded protocol is strictly stricter.

Positioning. This paper is not a method paper. We use LEADOPT-AGENT as a vehicle to study how the published LLM-agent gains in this area decompose into (i) genuine optimization, (ii) retrieval of memorized molecules from task names and reference dictionaries, and (iii) easy-to-find local edits that any baseline would also find. Our contribution is the protocols (§4 and §5) by which the existing literature can be re-evaluated, plus the empirical re-evaluation we perform on one representative agent.

3 The LLM Agent Used in This Study

We use a single, public, ReAct-style LLM agent (LEADOPT-AGENT) for all experiments. It is not the contribution of this paper; we use it as a representative agent against which to study benchmark behaviour. Below we describe its key components for reproducibility, and note where each component corresponds to a leakage channel that we ablate in Section 4.

3.1 Architecture

The agent runs four sequential phases.

Phase 0 (task understanding). The agent reads the task identifier (e.g., `celecoxib_rediscovery`) and looks it up in a curated dictionary `REFERENCE_COMPOUNDS` that maps each task to a list of known active SMILES. **This is the first leakage channel we ablate.**

Phase 1 (oracle probing). The agent scores the references and a fixed set of generic probes (benzene, biphenyl, aspirin) to seed the trajectory.

Phase 2 (hypothesis formation). The LLM is shown the task name, an oracle description from `ORACLE_DESCRIPTIONS` (e.g., "ECFP4 similarity to celecoxib"), and the probe scores, and is asked to articulate an SAR hypothesis. **The task name and description here are the second leakage channel.**

Phase 3 (iterative optimization). The main loop. At each iteration the LLM proposes 5 new SMILES grounded in the hypothesis and the current top-5 trajectory; valid proposals are scored; trajectory and hypothesis are updated. A plateau detector switches between exploitation and diversification.

Phase 4 (refinement). Local edits around the top trial.

3.2 Implementation details

LLM backbone. `gpt-4o-mini` via the OpenAI API; `response_format=JSON`; temperature 0.7. A typical task uses 8–15 LLM calls and costs ∼$0.05.

Oracle access. For PMO we call the TDC oracle (Huang et al., 2021) directly. For PROTEINLEADOPT-50 we call AutoDock Vina via the public DOCKSTRING wrapper (García-Ortegón et al., 2022).

Parallelism. For PROTEINLEADOPT-50 we run multiple (target, starter) pairs in parallel via `ProcessPoolExecutor`; within each pair, candidates dock concurrently via `ThreadPoolExecutor`. A persistent on-disk docking cache eliminates redundant Vina calls across runs.

What is identical across conditions. The agent's loop, prompts, validation, parallelism, and oracle access are identical in all C1–C4 conditions in Section 4. The only differences are which dictionary entries are visible to the LLM in its prompt and what the task identifier string is. The TDC oracle itself is unmodified. Single-seed runs are used throughout.

4 The PMO Story: Task-Name and Reference Leakage Dominate the Apparent Gain

LLM agents reported on PMO are typically given (i) the task name in the prompt at every iteration (e.g., "celecoxib_rediscovery"), (ii) a brief natural-language description of the oracle (e.g., "ECFP4 similarity to celecoxib"), and (iii) one or more curated reference compounds for that task (e.g., the SMILES of celecoxib itself, or known kinase inhibitors for the kinase-activity oracles). Each of these three signals can leak the answer to a pre-trained LLM. We measure how much.

4.1 Protocol

We re-run a public LLM-agent implementation, LEADOPT-AGENT, on all 23 PMO tasks under four information conditions corresponding to a 2×2 ablation on (task name visible, reference compound dictionary populated):

- **C1 (++) name+ref+**: the standard setting. Real task name in prompt, full `REFERENCE_COMPOUNDS` dictionary, real `ORACLE_DESCRIPTIONS` dictionary.
- **C2 (+–) name+ ref–**: real task name and oracle description, but the `REFERENCE_COMPOUNDS` and `SEED_COMPOUNDS` dictionaries are emptied.
- **C3 (-+) name– ref+**: task identifier replaced by an anonymous label `task_42` in every prompt; oracle description replaced by "Unknown oracle. Probe to learn its shape."; reference dictionary kept.
- **C4 (–) name– ref–**: name, description, and reference dictionary all hidden. The agent receives only the anonymous task label and the oracle's scalar feedback.

The TDC oracle itself is unmodified across conditions; we only change what the agent sees in its prompt. Each condition is run for a single seed, 50 oracle calls per task, on `gpt-4o-mini`.

4.2 Headline result

Table 1 reports per-task and aggregate top-1 AUC for the standard (**C1**) and fully blinded (**C4**) conditions across all 23 PMO tasks, alongside the published REINVENT-50 reference numbers from Gao et al. (2022).

The aggregate is striking: the LLM agent's sum AUC collapses from 15.10 in C1 to 6.46 in C4—a drop of 8.65 AUC points or 57% of the original score. The blinded agent in fact underperforms REINVENT's 7.06. Of the 23 tasks, 20 see a positive drop on blinding; three (*osimertinib_mpo*, *ranolazine_mpo*, *sitagliptin_mpo*) actually score *higher* in the blinded condition, suggesting that the curated references for those tasks can mislead the LLM in some cases. The largest drops are on rediscovery tasks and ML-based activity oracles whose names match a known drug or target: *drd2* (−0.93), *jnk3* (−0.79), *troglitazone_rediscovery* (−0.79), *celecoxib_rediscovery* (−0.73), *gsk3b* (−0.70), *thiothixene_rediscovery* (−0.67). On purely combinatorial tasks like *valsartan_smarts* (−0.00) and *median2* (−0.01), there is essentially no difference.

4.3 Decomposition: name leakage vs. reference dictionary

Table 2 reports the full 2×2 on a 5-task subset (*drd2*, *jnk3*, *perindopril_mpo*, *fexofenadine_mpo*, *osimertinib_mpo*). Aggregating across the five tasks (sum of AUC):

- Removing the reference-compound dictionary while keeping the task name visible: 3.65 → 1.47 (drop of 2.18).
- Removing the task name and description while keeping the reference dictionary: 3.65 → 2.94 (drop of 0.71).
- Removing both: 3.65 → 1.32 (drop of 2.33).

The dominant component is the curated reference dictionary. This is unsurprising: for tasks like *drd2*, the dictionary directly hands haloperidol's SMILES (a known D2 antagonist with score 0.99) to the agent in Phase 0; the agent's role then reduces to noting that the seed already scores high and minimally perturbing it. The marginal effect of task-name leakage on top of an already-loaded reference dictionary is smaller because the references are already doing most of the work. When references are removed, the task name still leaks information—the LLM can recall what celecoxib looks like from its name alone—but this name-only retrieval is less effective than handing the SMILES directly.

4.4 Interpretation

Three concrete claims follow.

(1) Headline LLM-agent numbers on PMO at low budget reflect retrieval, not optimization.

Task	REINVENT (Gao et al., 2022)	v1 (C1) full	v1 (C4) blinded	Δ C1−C4
albuterol_similarity	0.358	0.980	0.521	+0.46
amlodipine_mpo	0.432	0.704	0.459	+0.24
celecoxib_rediscovery	0.262	0.980	0.251	+0.73
deco_hop	0.558	0.706	0.513	+0.19
drd2	0.195	0.980	0.045	+0.93
fexofenadine_mpo	0.573	0.832	0.525	+0.31
gsk3β	0.141	0.795	0.098	+0.70
isomers_c7h8n2o2	0.078	0.980	0.261	+0.72
isomers_c9h10n2o2pf2cl	0.257	0.686	0.349	+0.34
jnk3	0.075	0.813	0.021	+0.79
median1	0.139	0.335	0.092	+0.24
median2	0.154	0.128	0.121	+0.01
mestranol_similarity	0.275	0.980	0.318	+0.66
osimertinib_mpo	0.670	0.325	0.487	−0.16
perindopril_mpo	0.354	0.699	0.240	+0.46
qed	0.905	0.906	0.848	+0.06
ranolazine_mpo	0.297	0.163	0.264	−0.10
scaffold_hop	0.422	0.653	0.362	+0.29
sitagliptin_mpo	0.095	0.000	0.092	−0.09
thiothixene_rediscovery	0.261	0.892	0.220	+0.67
troglitazone_rediscovery	0.197	0.936	0.144	+0.79
valsartan_smarts	0.000	0.000	0.000	0.00
zaleplon_mpo	0.364	0.633	0.223	+0.41
Sum AUC (23 tasks)	**7.06**	**15.10**	**6.46**	**+8.65**
Ratio over REINVENT	1.00×	2.14×	0.91×	—

Table 1: Per-task top-1 AUC at 50 oracle calls. **C1**: standard LEADOPT-AGENT setting (task name in prompt; reference compound dictionary; oracle description). **C4**: fully blinded (task name replaced by `task_42`; oracle description replaced by "Unknown oracle"; reference dictionary emptied). Δ = C1−C4 is the leakage-attributable gain. The agent's sum AUC drops below REINVENT's published number under blinding. Single seed; `gpt-4o-mini`.

Task	C1	C2	C3	C4
drd2	0.980	0.018	0.980	0.045
jnk3	0.813	0.030	0.813	0.021
perindopril_mpo	0.699	0.310	0.283	0.240
fexofenadine_mpo	0.832	0.467	0.585	0.525
osimertinib_mpo	0.325	0.640	0.281	0.487
Sum (5 tasks)	**3.65**	**1.47**	**2.94**	**1.32**

Table 2: 2 × 2 ablation on 5 PMO tasks. Conditions: **C1**: name on, references on; **C2**: name on, references off; **C3**: name off, references on; **C4**: name off, references off. Each cell is top-1 AUC at 50 oracle calls, single seed. Removing the reference dictionary (C1→C2) drops sum AUC by 2.18; removing the task name (C1→C3) drops it by 0.71. Both signals leak.

Our blinded sum AUC of 6.46 is below REINVENT's 7.06, despite the LLM having both the in-context optimization trajectory and substantial pre-training chemistry knowledge. Reports of 2–3× improvements over RL/GA baselines (Kim et al., 2025; Krüger et al., 2026; Le et al., 2025) should be read with this in mind. SEISMO's recent result of 19.85 (Krüger et al., 2026) uses a much stronger LLM backbone (Claude Opus 4.5) but constructs its prompt on the same drug-name-visible task descriptions that we ablate here. We attempted to apply our blinding protocol to SEISMO directly using `gpt-4o-mini` (see Section 6); that substitution induced a mode-collapse failure unrelated to leakage, so the gap between SEISMO's published number and a blinded version remains an open empirical question.

(2) The reference-compound dictionary is the largest single confound. Curated dictionaries that map an oracle name to a known active compound are common across the recent literature: ChatDrug retrieves from a 10K-molecule database (Liu et al., 2024); AgentDrug uses retrieval as in-context examples (Le et al., 2025); MT-Mol retrieves the top-100 ZINC molecules per task (Kim et al., 2025). None of these papers reports a no-retrieval ablation that strictly cuts retrieval AND blinds the task name. We recommend such an ablation become standard.

(3) The blinded protocol is straightforward to adopt. We replace task names with anonymous identifiers and empty the reference dictionary; the rest of the agent is unchanged. The cost is a single re-run per task. We will release the blinded prompts so that future agents can be evaluated against this floor without engineering effort.

5 The Protein Story: Random-Walk Mutation as a Lower Bound

PMO uses ML-based or formula-matching oracles whose names leak the answer. The natural step toward "real" lead optimization is a docking-based oracle: AutoDock Vina, GNINA, or a co-folding model such as Boltz-2 (Passaro et al., 2025). Multiple recent agent papers report large gains in this regime (Le et al., 2025; Krüger et al., 2026). We construct a 50-pair protein-targeted lead-optimization benchmark with a multi-objective composite oracle and ask the same question: do LLM agents add value over a trivial baseline?

5.1 The PROTEINLEADOPT-50 benchmark

We select 10 medically-relevant protein targets from the DOCKSTRING dataset (García-Ortegón et al., 2022) spanning kinase (ABL1, JAK2, MAPK14), GPCR (ADRB2, DRD2), nuclear-receptor (ESR2, PGR), DNA-repair (PARP1), serine-protease (F2 thrombin), and CYP (CYP3A4) families. For each target we sample 5 starter molecules with Vina docking scores in the 30–50% percentile of that target's DOCKSTRING distribution, yielding $10 \times 5 = 50$ (target, starter) pairs. Starters are reasonable but not optimal hits, representing realistic lead-optimization headroom.

The composite oracle is

$$\mathcal{O}(S) = \mathcal{I}(S) + 0.5 \cdot \mathrm{QED}(S) - 2 \cdot \mathcal{P}(S, S_0) \quad (1)$$

with docking-improvement term $\mathcal{I}(S) = \min(1, \max(0, d_{\text{base}} - d(S))/5)$ where $d(S)$ is the Vina docking score and d_{base} the starter's, drug-likeness $\mathrm{QED}(S)$ (Bickerton et al., 2012), and similarity penalty $\mathcal{P}(S, S_0) = \max(0, 0.3 - \tau(S, S_0))$ where τ is the Morgan-fingerprint Tanimoto similarity to the starter (Bajusz et al., 2015). The composite rewards docking improvement up to 5 kcal/mol, rewards drug-likeness, and heavily penalises divergence from the starter ($\tau < 0.3$). Vina is evaluated via the public DOCKSTRING wrapper at ~6 s per molecule on 8 CPU cores. The oracle budget is 30 calls per pair.

We evaluate four methods: **LLM agent (LEADOPT-AGENT)** as in Section 4 adapted to protein-targeted lead opt, with the top-5 DOCKSTRING molecules per target as in-context references; **GPT-direct**, a single `gpt-4o-mini` prompt for 30 candidates with no iteration; **GraphGA**, a graph-based GA with atom-replacement, substituent-addition, and atom-removal operators and population size 20; and **Random Walk**, random RDKit single-edit mutations drawing small substituents (F, Cl, Br, OMe, methyl, hydroxyl, primary amine) at random aromatic-ring positions, with no LLM and no oracle-aware selection.

Method	n	ΔDock (kcal/mol)	Succ. rate	Comp. (mean)	QED (mean)
GPT-direct	50	−0.36	0.24	0.51	0.80
LLM agent	50	−0.81	0.36	0.55	0.78
GraphGA	50	−0.84	0.40	0.57	0.78
Random walk	**50**	**−0.98**	**0.60**	**0.58**	0.77

Table 3: Aggregate results on PROTEINLEADOPT-50 (30-call budget per pair, single seed). All four methods completed all 50 pairs. Random Walk uses neither LLM nor oracle feedback; it samples random small substituent additions at aromatic positions. Random Walk wins on ΔDock against the LLM agent on 33 of 50 pairs.

5.2 Results

Table 3 reports aggregate metrics. **Random Walk outperforms every other method on every aggregate metric**: average ΔDock −0.98 kcal/mol vs. −0.84 (GraphGA), −0.81 (LLM agent), −0.36 (GPT-direct); success rate 0.60 vs. 0.40/0.36/0.24; mean composite 0.58 vs. 0.57/0.55/0.51. On the 50 matched pairs Random Walk wins on ΔDock against the LLM agent in 33 of 50 pairs (66%).

Why this happens. Inspecting Random Walk trajectories: adding a single Cl, Br, or CF_3 to an aromatic position typically lowers the Vina docking score by 0.5–1.5 kcal/mol via Vina's volume and lipophilicity terms. Single-atom additions preserve the Murcko scaffold and stay above the $\tau \geq 0.3$ similarity floor, so the similarity penalty does not bite; QED drops by less than 0.1, so the QED reward remains high. The oracle's effective gradient at 30 calls is therefore dominated by single-edit lipophilic improvements, which a random mutator hits by chance. Concretely, on JAK2 starter `CN1CCC(=Cc2c[nH]c3ccc(F)cc23)CC1` (baseline −8.3 kcal/mol), Random Walk added a chlorine at one aromatic position to obtain ΔDock = −1.2 kcal/mol within 9 oracle calls.

Table 4 gives the per-target breakdown. Random Walk wins on 7 of 10 targets; GraphGA wins on ESR2 and CYP3A4 (where halogenation alone is less productive and structural mutation helps); the

Target	LLM agent	GraphGA	GPT direct	RW
ABL1	−0.82	−0.16	+0.16	**−0.96**
ADRB2	−0.88	−0.68	−0.46	**−0.94**
DRD2	−0.86	−0.86	+0.66	**−1.08**
ESR2	−0.50	**−1.28**	−0.26	−0.96
F2	**−1.08**	−0.80	+0.20	−0.66
JAK2	−0.98	−0.72	−0.62	**−1.12**
MAPK14	−0.72	−0.84	−0.80	**−0.98**
PARP1	−0.82	−0.78	−0.68	**−1.06**
PGR	−0.84	−0.96	−1.04	**−1.10**
CYP3A4	−0.62	**−1.30**	−0.76	−0.98

Table 4: Per-target mean ΔDock (kcal/mol; more negative is better) on PROTEINLEADOPT-50. All methods evaluate 5 pairs per target. Random Walk wins on 7/10 targets; GraphGA on ESR2 and CYP3A4; the LLM agent on F2. Values in **bold** are best per row. RW = Random Walk.

LLM agent wins on F2 (thrombin), where kinase-style halogenation does not apply as readily.

Implication for benchmark design. This is not a result about LLM agents but about the AutoDock Vina + multi-objective composite + small-budget combination. To make the benchmark meaningful for agent comparison, future work should use a more accurate oracle (FEP, Boltz-2 (Passaro et al., 2025)), tighter structural constraints (e.g., $\tau \leq 0.5$ to require scaffold hopping), or selectivity objectives that penalise binding to off-targets. We release PROTEINLEADOPT-50 so future agent papers can be calibrated against the Random Walk floor; an agent that only narrowly beats Random Walk should not be claimed as effective.

6 Discussion

What we did NOT find. We did not find that LLM agents are useless for molecular optimization. The agent we tested clearly beats a non-iterative GPT-direct prompt, and on PMO with full information it produces SMILES that the oracle scores highly. What we found is more specific: *at the budgets and benchmarks at which the field currently reports gains, those gains can be largely or entirely explained by mechanisms other than online optimization*. On PMO at 50 oracle calls, the mechanism is in-context retrieval of memorized molecules. On a Vina-plus-composite protein benchmark at 30 oracle calls, the mechanism is that the achievable score is dominated by single-atom lipophilic edits that any local search method finds.

Failed attempt to control across agents. A natural question is whether the leakage effect generalises to other published LLM agents. We attempted to re-run SEISMO (Krüger et al., 2026) under our blinded protocol on a 5-task PMO subset, substituting `gpt-4o-mini` for SEISMO's published Claude Opus 4.5 backbone and patching only the task-description strings to remove drug names. The substitution failed: on `perindopril_mpo` in the standard (non-blinded) condition, all 51 iterations produced invalid SMILES—the model emitted variants of a single ring-unclosed scaffold and ignored the "Invalid SMILES" feedback, never recovering. We observed similar but less severe pathologies on three other tasks. SEISMO's trajectory-conditioned prompt is designed for stronger backbones, and `gpt-4o-mini` mode-collapses inside it. We therefore cannot read the resulting standard-vs.-blinded gap as evidence about SEISMO's leakage exposure. Disentangling backbone capability from prompt-leakage exposure would require running SEISMO with its intended backbone under our blinding patch, which is outside our compute budget. What we can say from indirect evidence is that SEISMO, MT-Mol (Kim et al., 2025), AgentDrug (Le et al., 2025), ChatDrug (Liu et al., 2024), MOLLEO (Wang et al., 2025), and LICO (Nguyen and Grover, 2025) all include the task name and at least one of the leakage channels we ablate; the blinded protocol can be applied to any of them with their original backbones.

What our LEADOPT-AGENT is useful for. The LLM agent is genuinely useful in two settings even after our negative findings. First, when the user actually wants to retrieve known scaffolds for a known target ("give me 20 celecoxib analogs"), the agent's name-aware behaviour is a feature, not a bug. The retrieval is fast and chemically sensible. Our criticism is of the benchmark's interpretation of that behaviour, not the behaviour. Second, on tasks where the LLM has no useful name-based prior (the median-similarity, isomer, and SMARTS tasks), the agent's blinded performance is competitive with or better than RL/GA baselines: *median2*, *valsartan_smarts*, and *qed* are examples where blinding makes essentially no difference. An honest LLM-agent claim would be that it helps on these tasks specifically, not on PMO as a whole.

Recommendations. For papers reporting LLM-agent results on molecular optimization at low bud-

get, we recommend two cheap protocols.

- **(R1) Report a blinded re-run alongside the headline number.** Replace task names with anonymous labels, empty any retrieval/reference dictionaries, replace oracle descriptions with "Unknown oracle. Probe to learn its shape.". Apply the protocol with each method's intended LLM backbone; cross-backbone substitution can fail catastrophically as in Section 6.
- **(R2) Report a Random Walk floor on any docking-based composite oracle.** The mutator we used (random small substituent at random aromatic position) is $\sim$50 lines of code; running it costs the same as running the agent.

These two protocols would catch the failure modes our paper documents and prevent the kinds of inflated claims our experiments expose.

7 Conclusion

We re-evaluated a public LLM-agent implementation on the two evaluation regimes that dominate the recent literature on LLM agents for low-budget molecular optimization. On PMO at 50 oracle calls, the agent's published-style sum AUC of 15.10 collapses to 6.46—below REINVENT's 7.06—when task names and curated reference compounds are blinded. The dominant component is the reference-compound dictionary; task-name leakage adds a smaller marginal effect. On a 50-pair protein-targeted lead-optimization benchmark with an AutoDock Vina + composite oracle and a 30-call budget, a trivial random-walk halogen/methyl mutator outperforms the LLM agent and every other tested baseline. We argue this is not a result about LLM agents but about benchmark design: the named PMO tasks reward retrieval, and the Vina-plus-composite oracle at small budget rewards local lipophilic edits. We release both the name-blinded protocol and the random-walk floor as standard sanity checks and recommend that future LLM-agent papers in this area report against them.

What remains to be measured is the LLM agent's value when both confounds are removed: a low-budget benchmark with a more accurate oracle (FEP, Boltz-2 (Passaro et al., 2025)), task-name blinding, no curated reference dictionary, and a composite that does not reward halogenation. We expect this to be a much more sample-efficient regime where retrieval helps less and structural reasoning helps more, and where agent papers can report optimization-driven gains that are not a benchmark artifact.

Limitations

Single seed. Both the PMO blinded re-evaluation and the PROTEINLEADOPT-50 runs use single-seed evaluations per task or pair. The PMO blinding effect (-8.65 AUC across 23 tasks) is large enough that seed-to-seed variance does not change the qualitative finding, but our point estimates carry no confidence intervals. Multi-seed re-runs are straightforward and we plan to release them in a follow-up. The same caveat applies to the random-walk vs. LLM-agent comparison on PROTEINLEADOPT-50.

Single LLM backbone. All experiments use `gpt-4o-mini`. Stronger backbones (Claude Opus 4.5, GPT-5.1) would likely both raise the standard sum AUC and change the leakage–blinding gap, but the direction of the change is hard to predict from our data: blinding might collapse less under a stronger backbone (because the model can still optimise from generic chemistry knowledge), or it might collapse more (because the standard prompt's retrieval becomes even more effective).

Failed cross-agent re-evaluation. We attempted to re-run SEISMO (Krüger et al., 2026) under our blinding protocol with `gpt-4o-mini` as a controlled comparison (Section 6). The substitution induced a mode-collapse failure on `gpt-4o-mini`: on `perindopril_mpo` all 51 iterations produced invalid SMILES, with the model emitting variants of a single ring-unclosed scaffold and ignoring repeated "Invalid SMILES" feedback. This failure is unrelated to leakage and is an interaction between SEISMO's trajectory-conditioned long-history prompt and the smaller backbone. We therefore have no controlled measurement of leakage exposure for any agent other than LEADOPT-AGENT.

One representative agent. As a consequence of the previous limitation, our quantitative claim that "LLM-agent papers in this regime are leakage-driven" rests on a re-evaluation of one publicly-available agent. The other recent agents (AgentDrug, MT-Mol, MOLLEO, LICO, MECo, MolAct) include the same leakage channels in their prompts (task names, reference compounds, top-k retrieval); whether they also collapse to below-baseline performance under blinding is an open empirical question.

Coverage of the 2×2 ablation. The 2×2 ablation in Section 4 is run on 5 representative PMO tasks, not all 23. We extrapolate the magnitude of the reference-vs.-name decomposition from this subset. The full 23-task fully-blinded condition is run, however; only the two intermediate cells (name+ ref– and name– ref+) are restricted to 5 tasks.

Vina oracle approximation. PROTEINLEADOPT-50 uses AutoDock Vina via DOCKSTRING (García-Ortegón et al., 2022). Vina is fast but approximate; correlation with experimental binding affinity is moderate. The random-walk-wins finding is specific to this oracle. With a more accurate oracle (FEP, Boltz-2 (Passaro et al., 2025)) the finding may flip. We argue this is itself a recommendation: future protein-targeted lead-opt benchmarks should use stronger oracles.

Random-walk parameter choice. The random-walk baseline uses a fixed list of small substituents and random aromatic positions. Different substituent sets or position-selection strategies could change its absolute performance but the qualitative finding (that a non-LLM, non-oracle-aware mutator beats the LLM agent) is robust to these details.

Number of (target, starter) pairs evaluated by each method. All four methods (LLM agent, GraphGA, Random Walk, GPT-direct) completed the full 50 pairs within wall-clock. Per-pair matched comparisons (Section 5) reach the same conclusion as the per-method aggregates: Random Walk wins on 33 of the 50 shared pairs against the LLM agent.

Ethics Statement

This work develops LLM agents for molecular optimization in drug discovery. We acknowledge two ethical considerations.

Dual-use. Tools that improve binding to a protein target can in principle be applied to design molecules with harmful effects, including toxic compounds or controlled substances. We mitigate this by (i) targeting protein families that are public therapeutic targets in medicinal chemistry literature (kinases, GPCRs, nuclear receptors); (ii) evaluating only with public docking software; and (iii) not training or fine-tuning the LLM on toxicity- or controlled-substance-specific datasets. We do not provide an interface for arbitrary user-specified protein targeting, only the curated benchmark targets.

Reproducibility and access. The PMO benchmark and DOCKSTRING are open-source. Our PROTEINLEADOPT-50 benchmark, agent code, and run scripts are released under a permissive licence. The LLM backbone is a closed-source API; we report all prompt templates verbatim in Appendix A so the system can be replicated with comparable LLMs.

Computational cost. Each protein optimization run uses roughly $30 \times 6 \approx 180$ seconds of CPU-Vina docking and ∼$0.05 in API cost, making the full benchmark inexpensive to run. We do not pretrain or fine-tune any model; carbon footprint is dominated by the LLM API provider.

References

Dávid Bajusz, Anita Rácz, and Károly Héberger. 2015. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of Cheminformatics*, 7:1–13.

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

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

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor W. Coley. 2022. Sample efficiency matters: A benchmark for practical molecular optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 21342–21357.

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

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

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W. Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. 2021. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. In *Advances in Neural Information Processing Systems Track on Datasets and Benchmarks*.

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

Hyomin Kim, Yunhui Jang, and Sungsoo Ahn. 2025. MT-Mol: Multi agent system with tool-based reasoning for molecular optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11544–11573.

Fabian P. Krüger, Andrea Hunklinger, Adrian Wolny, Tim J. Adler, Igor Tetko, and Santiago David Villalba. 2026. SEISMO: Increasing sample efficiency in molecular optimization with a trajectory-aware LLM agent. *Preprint*, arXiv:2602.00663. Preprint, January 2026.

Khiem Le, Ting Hua, and Nitesh V. Chawla. 2025. AgentDrug: Utilizing large language models in an agentic workflow for zero-shot molecular editing. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 24448–24458.

Shengchao Liu, Jiongxiao Wang, Yijin Yang, Chengpeng Wang, Ling Liu, Hongyu Guo, and Chaowei Xiao. 2024. ChatGPT-powered conversational drug editing using retrieval and domain feedback. In *International Conference on Learning Representations (ICLR)*.

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

Tung Nguyen and Aditya Grover. 2025. LICO: Large language models for in-context molecular optimization. In *International Conference on Learning Representations (ICLR)*.

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

Saro Passaro, Gabriele Corso, Jeremy Wohlwend, Mateo Reveiz, Stephan Thaler, Vignesh Ram Somnath, Noah Getz, Tally Portnoi, Julien Roy, Hannes Stark, Daniel Kwabi-Addo, Dominique Beaini, Tommi Jaakkola, and Regina Barzilay. 2025. Boltz-2: Towards accurate and efficient binding affinity prediction. *bioRxiv*.

Austin Tripp, Gregor N. C. Simm, and José Miguel Hernández-Lobato. 2021. A fresh look at de novo molecular design benchmarks. In *NeurIPS 2021 AI for Science Workshop*.

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

Zhuo Yang, Yeyun Chen, Jiaqing Xie, Ben Gao, Shuaike Shen, Wanhao Liu, Liujia Yang, Beilun Wang, Tianfan Fu, and Yuqiang Li. 2025. MolAct: An agentic RL framework for molecular editing and property optimization. *Preprint*, arXiv:2512.20135.

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

A Prompt Templates

A.1 Standard (C1) prompt for Phase 3

```
You are an expert medicinal chemist
optimizing molecules for an oracle. Use
chemistry knowledge AND oracle feedback
to propose new SMILES that score higher.

Task: {TASK_NAME}
Reference compound: {REFERENCE}
Oracle hint: {ORACLE_DESCRIPTION}
Hypothesis: {HYPOTHESIS}
Top 5 trials so far: {TRAJECTORY}
Strategy: {EXPLOIT or DIVERSIFY}

Propose 5 new SMILES with rationales.
Return JSON: {"edits": [{"smiles":"...",
"rationale":"..."}, ...]}
```

A.2 Blinded (C4) prompt for Phase 3

The differences from C1, applied uniformly:

- `TASK_NAME` replaced by anonymous identifier `task_42`.
- `REFERENCE` replaced by `None` (the dictionary is empty).
- `ORACLE_DESCRIPTION` replaced by "Unknown oracle. Probe to learn its shape."
- Phase 1 probes use only the generic probes (benzene, biphenyl, aspirin) since the reference dictionary is empty; the agent has no task-specific seeds.

All other prompt machinery—system message, JSON schema, validation logic, plateau detection, parallelism—is identical between conditions.

B Hyperparameters

Phase 3 iterative loop.

- LLM: `gpt-4o-mini`, temperature 0.7, JSON response format
- Per-iteration proposals: $n = 5$
- Plateau improvement threshold: 0.005
- Max iterations: budget-limited, typically ≤ 10

Parallelism (PROTEINLEADOPT-50).

- Process workers per benchmark run: 15
- Docking threads per worker: 6
- Persistent dock cache: `(target, canonical SMILES)` $\rightarrow$ `Vina score`, file-locked JSON

C Reference Compound Dictionary

For reproducibility, the curated `REFERENCE_COMPOUNDS` entries used in our standard (C1) condition include known drugs for the named PMO tasks: e.g., the SMILES of perindopril for *perindopril_mpo*, of haloperidol for *drd2*, of celecoxib for *celecoxib_rediscovery*, of imatinib-analog for *jnk3*. The full dictionary is included in the released code. In the blinded (C4) condition the dictionary is set to empty: `REFERENCE_COMPOUNDS = {}`.