EXLLM: EXPERIENCE-ENHANCED LLM OPTIMIZATION FOR MOLECULAR DESIGN AND BEYOND

Anonymous authors
Paper under double-blind review

ABSTRACT

Molecular design involves an enormous and irregular search space, where traditional optimizers such as Bayesian optimization, genetic algorithms, and generative models struggle to leverage expert knowledge or handle complex feedback. Recently, LLMs have been used as optimizers, achieving promising results on benchmarks such as PMO. However, existing approaches rely only on prompting or extra training, without mechanisms to handle complex feedback or maintain scalable memory. In particular, the common practice of appending or summarizing experiences at every query leads to redundancy, degraded exploration, and ultimately poor final outcomes under large-scale iterative search. We introduce ExLLM, an LLM-as-optimizer framework with three components: (1) a compact, evolving experience snippet tailored to large discrete spaces that distills non-redundant cues and improves convergence at low cost; (2) a simple yet effective k-offspring scheme that widens exploration per call and reduces orchestration cost; and (3) a lightweight feedback adapter that normalizes objectives for selection while formatting constraints and expert hints for iteration. ExLLM sets new state-of-the-art results on PMO and generalizes strongly—in our setup, it sets records on circle packing and stellarator design, and yields consistent gains across additional domains—requiring only a task-description template and evaluation functions to transfer.

1 INTRODUCTION

Molecular design underpins drug discovery and materials science, yet the search space is vast and highly discrete, making efficient optimization difficult. Classical machine learning approaches: Bayesian optimization (BO), genetic algorithms (GA), reinforcement learning (RL), multi-objective optimization (MOO), and MCMC, treat the problem largely as black-box search (Tripp et al., 2021; Jensen, 2019; Nigam et al., 2019; Liu et al.; Verhellen, 2022; Xie et al., 2021; Sun et al., 2022; Olivecrona et al., 2017; Jin et al., 2020). Deep generative models improve proposal quality by learning molecular distributions (e.g., JTVAE, VJTNN, DST, diffusion and transformer-based models such as MOOD, MolGPT, MOLGEN) and enable latent-space search (Jin et al., 2018a;b; Fu et al., 2021; Lee et al., 2023; Bagal et al., 2021; Fang et al., 2024; Abeer et al., 2024). However, these lines typically rely on scalarized rewards and fixed pipelines, making it hard to incorporate rich priors (chemist heuristics, textual rules) and to handle heterogeneous feedback (multiple objectives, hard/soft constraints) without task-specific re-engineering; practical protocols such as PMO further highlight the need to optimize under a fixed evaluation budget due to costly oracle calls (Gao et al., 2022).

Large language models (LLMs) offer a complementary opportunity: they encode broad domain knowledge, support reasoning, and can be steered with prompts (Vaswani, 2017; AI4Science & Quantum, 2023; Brown, 2020). Recent work explores LLMs as optimizers or operators within evolutionary loops (e.g., OPRO, LMEA, AlphaEvolve; reasoning–acting with ReAct) and reports encouraging results on numerical, coding, and planning tasks (Yang et al., 2024; Liu et al., 2024b; Novikov et al., 2025a; Yao et al., 2022; Wu et al., 2024a). In molecular design, systems such as ChemCrow, LICO, MolReGPT, Prompt-MolOpt, and MOLLEO demonstrate that pre-trained LLMs or LLM–GA hybrids can guide candidate generation and multi-parameter search (M. Bran et al., 2024; Nguyen & Grover, 2024; Li et al., 2024; Wu et al., 2024b; Wang et al., 2024a). Yet these efforts remain early-stage: most are heavily prompt-dependent or require additional parameter training,

and lack a memory mechanism tailored to molecular optimization which has large, discrete search loops. Existing memory systems were developed primarily for QA, coding, or short-horizon decision making; they append per-step summaries and retrieve them at inference (RAG, RETRO, MemoryBank, MemLLM, A-Mem, Memory-R1), which—when naively reused over long optimization runs—inflate prompts, accumulate redundancy, and bias the search (Lewis et al., 2020; Borgeaud et al., 2022; Zhong et al., 2024; Modarressi et al., 2024; Xu et al., 2025; Yan et al., 2025). Moreover, unified handling of heterogeneous feedback (multi-objective signals, constraints, and expert textual hints) remains limited in practice.

We propose **ExLLM**, an LLM-as-optimizer framework designed for molecular optimization. ExLLM treats the LLM as the optimizer itself and introduces three complementary mechanisms: a *single, evolving* experience distilled from good and bad cases to avoid memory bloat; a k*-offspring* sampling scheme that leverages the autoregressive factorization to widen exploration per query; and a unified *feedback adapter* that integrates objectives, constraints and textual feedback for iterative prompting. The framework is simple to transfer with our task template custom evaluation functions and does not require any training.

Contributions. (1) We introduce an **evolving experience** mechanism tailored to large discrete spaces: a compact, low-redundancy snippet updated each generation, which improves convergence and results while controlling cost and avoiding exploration collapse, contrasting with retrieval-style memories (Zhao et al., 2023; Lewis et al., 2020; Borgeaud et al., 2022). (2) We propose a k**-offspring** strategy that increases exploratory breadth per LLM call and yields consistent gains under a fixed budget, with an empirical trade-off curve for k. (3) We develop a **feedback adapter** that unifies multi-objective signals for selection and incorporates constraints/expert text as concise prompts; promoting critical, variable constraints to explicit objectives improves stability without additional training. (4) We demonstrate strong results on molecular optimization: ExLLM achieves a PMO aggregate score of **19.165** (max 23), ranking **first on 17/23** tasks and improving over the previous SOTA (17.862) by **+7.3%** (Gao et al., 2022; Wang et al., 2024a). We package these contributions into a general optimizer for large discrete spaces: with only a task-description template and evaluation functions, it sets new records in circle packing and stellarator design, and delivers consistent gains across additional domains (e.g., MOTSP/MOCVRP, SACS, NK2R peptide, GCU operator design).

2 Related Work

2.1 Molecular Design with Machine Learning

Molecular optimization has long been approached as a black-box search problem using Bayesian optimization (BO), genetic algorithms (GA), reinforcement learning (RL), multi-objective optimization (MOO), and MCMC variants. Representative **GA/BO** lines include GB-GA and its extensions—e.g., diversity-aware discriminators and Tanimoto-kernel Gaussian processes—which remain competitive on classic property maximization tasks (Jensen, 2019; Nigam et al., 2019; Tripp et al., 2021; Gómez-Bombarelli et al., 2018). BO-style **MOO** with learned encoders (MLPS) (Liu et al.) and graph-based MOO (Verhellen, 2022) likewise seek Pareto-optimal solutions in latent or combinatorial spaces. Sampling-based methods such as MARS (**MCMC**) and Monte-Carlo tree search further explore chemical spaces probabilistically to identify molecules with desired properties (Xie et al., 2021; Sun et al., 2022). **RL** pipelines train generative policies from reward signals (e.g., REINVENT and rationale-guided GNNs) (Olivecrona et al., 2017; Jin et al., 2020), with recent variants such as DyMol, Augmented Memory and Genetic-GFN improving multi-objective handling, utilize data augmentation and experience replaying, or blending evolutionary operators with flow-based generators (Shin et al., 2024; Guo & Schwaller, 2024; Kim et al., 2024). While these families achieve strong results on several PMO tasks (Gao et al., 2022), they typically optimize only via function evaluations, making it difficult to incorporate domain priors (chemist heuristics, expert constraints) beyond hand-crafted rewards, and requiring nontrivial re-engineering or additional training when objectives or constraints change.

Deep generative models advanced the field by learning molecular distributions and enabling higher-quality proposals. Autoencoding and structured-latent approaches (e.g., JTVAE,VJTNN, DST) capture scaffold and substructure regularities (Jin et al., 2018a;b; Fu et al., 2021), and diffusion and transformer-based models (MOOD, MolGPT, MOLGEN) further improve sample fidelity and cross-domain applicability (Lee et al., 2023; Bagal et al., 2021; Fang et al., 2024). Latent-space

optimization (LSO) shows that multi-objective search in the learned space can be effective for deep generators (Abeer et al., 2024). However, limitations exist: (1) Models largely learn from data and scalar rewards; codifying rich textual heuristics, design rules, or exception-heavy lab wisdom into training signals is cumbersome and brittle. (2) Combining multiple objectives, hard/soft constraints, and heuristic rules in a single training loop often requires bespoke reward shaping, delicate weighting, or separate pipelines. Many systems require task-specific fine-tuning or reward engineering when the optimization goal changes, limiting plug-and-play transfer to new objectives or domains (Gao et al., 2022; Liu et al.; Verhellen, 2022).

2.2 Molecular Design with LLM

LLMs with domain knowledge are increasingly explored for drug and materials discovery (AI4Science & Quantum, 2023). Agentic tool-use systems such as ChemCrow show that LLMs can plan, call external chemistry tools, and iteratively refine candidates (M. Bran et al., 2024). In LICO (Nguyen & Grover, 2024), in-context learning is strengthened by pretraining with separated embedding and prediction layers to improve molecule generation; later Moayedpour et al. (2024) extends this idea to multi-objective settings. MolReGPT targets few-shot molecular optimization with additional parameterization for rapid adaptation (Li et al., 2024). MOLLEO integrates LLMs with a genetic algorithm to guide mutations/edits during search, illustrating a viable LLM-as-optimizer pattern (Wang et al., 2024a), but it lacks an explicit experience-reuse mechanism, explores this framework at a relatively early stage, and is mainly scoped to molecular-design benchmarks. Prompt-MolOpt explicitly leverages the domain knowledge of LLMs via property-specific prompt embeddings and a sequence-to-sequence Transformer trained on substructure-annotated pairs, demonstrating strong zero-/few-shot behavior on property-driven tasks (Wu et al., 2024b). However, existing LLM-based molecular design approaches are still highly prompt-dependent or need additional training, and lack an experience mechanism tailored to vast exploration spaces that can distill and reuse knowledge during optimization. The ability to handle complex feedback is still limited.

2.3 LLM-as-Optimizer and Memory Mechanism

We have put this part to appendix 7.2.

3 Method

We first provide an overview of our framework, also shown in figure 8, and then give detailed elaborations of the key components.We cast the LLM as an evolutionary optimizer under a fixed evaluation budget B following PMO benchmark settings Gao et al. (2022). Let P_t denote the population at generation t:

Initialization. Construct the initial population P_0 (e.g., random seeds, scaffold templates, or domain priors).

LLM-as-optimizer with k-offspring. For every randomly selected pair of parents $(x_i, x_j) \subseteq P_t$, a proprietary commercial LLM (e.g., GPT, Gemini) proposes k candidate offspring:

$$C_t(x_i, x_j) = \{y^{(1)}, \ldots, y^{(k)}\}, \qquad y^{(i)} \sim p_\theta\big(\cdot \;\big|\; (x_i, x_j),\ \text{task template},\ E_t\big),$$

where p_θ is the autoregressive distribution and E_t is the distilled experience from the previous generation (§3.1). The complete candidate set at iteration t is then

$$C_t = \bigcup_{(x_i, x_j) \in \mathcal{M}(P_t)} C_t(x_i, x_j),$$

where $\mathcal{M}(P_t)$ denotes the set of parent pairs selected from the current population P_t. The prompt is constructed using a designed template that includes the task description, requirements, objective specifications, parent information, mutation/crossover instructions, output guidelines, and other constraints.

Feedback aggregation. Each $y \in C_t$ is evaluated to obtain: (i) a vector of objective values $\mathbf{f}(y) = [f_1(y), \ldots, f_M(y)]$; the raw objective values are preserved so that they can be explicitly

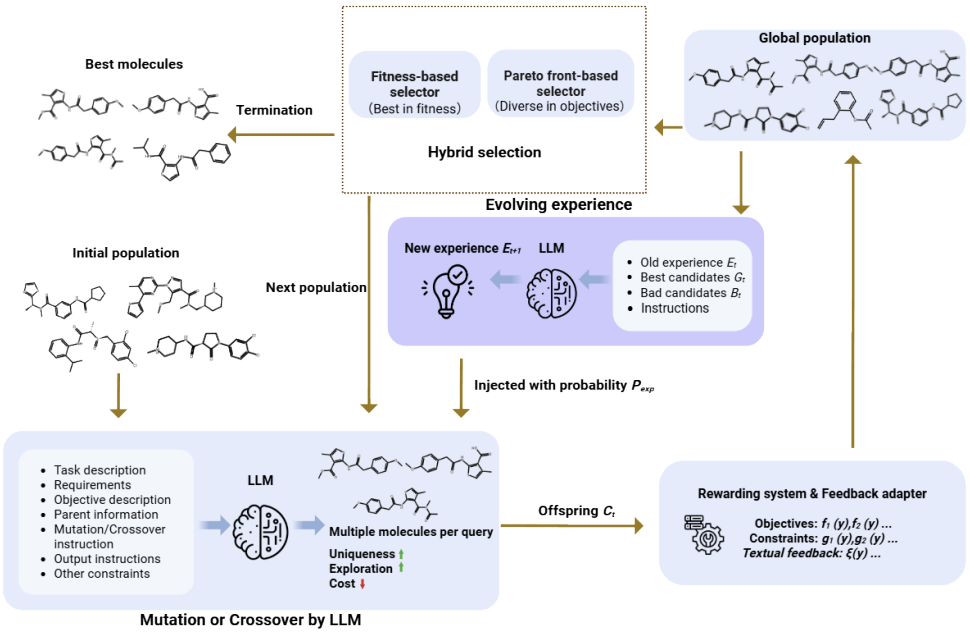

Figure 1: Overall framework of ExLLM. The process begins with an initialized population, followed by LLM-based k-offspring generation, evaluation and feedback aggregation, hybrid selection (fitness + Pareto), and experience update. These steps repeat until the evaluation budget is exhausted.

included in the prompt, which facilitates the LLM's understanding of task semantics. All objectives form a unified vector representation that is subsequently employed for Pareto-based selection. (ii) constraint values $\mathbf{g}(y) = [g_1(y), \ldots, g_J(y)]$; (iii) optional expert/textual feedback $\xi(y)$. The feedback adapter (§3.3) normalizes objectives into a comparable vector used for selection, and formats $\mathbf{g}(y)$ and $\xi(y)$ into compact, structured text that can be injected back to the LLM in the next generation.

Selection (Fitness + Pareto). To construct the next generation, we employ a hybrid strategy: half of the candidates are selected by ranking according to the scalar fitness value

$$F(y) \;=\; \sum_{i=1}^{M} w_i \, \widehat{f}_i(y), \qquad \sum_i w_i = 1, \tag{1}$$

where weights are equal in our experiments, the remaining half are drawn from the Pareto front based on dominance relations over the normalized objective vectors. This design balances exploitation of high-performing individuals with preservation of diversity across the multi-objective space. Because some molecules may achieve relatively high fitness values and exhibit structural diversity with good potential, but share similar objective distributions which make them dominated by many other points, fitness-based selection ensures these candidates are retained. Meanwhile, molecules with lower fitness may still excel on specific objectives and remain non-dominated; such molecules are preserved by Pareto-front selection for their potential.

Experience update. From all historically evaluated candidates up to generation t, we identify a set of *good* examples G_t (top-r by F) and sample a set of *bad* examples B_t uniformly from the lower half of the fitness ranking. We then update the experience E_{t+1} by combining E_t with distilled insights from $G_t \cup B_t$ and discarding stale content; see §3.1. Then it jumps to LLM-as-optimizer with k-offspring and repeats until the total evaluation calls reach B (PMO protocol) (Gao et al., 2022).

3.1 Evolving Experience

Why not a retrieval-style memory. As discussed in §7.3, traditional memories maintain *per-query* summaries and then *re-inject* many of them at every LLM call. We implemented this variant as a control: after each generation, we summarize the new LLM outputs, rank the pool, and insert the top-K ($K \leq 100$) entries into every subsequent prompt (Table 1). In large discrete search, this leads

Memory Method	Hypervolume	Top10 AUC	Uniqueness	LLM queries	Cost (USD)	Running time (h)	Memory (MB)
Retrieval-style	0.427 ± 0.155	3.904 ± 0.427	0.139 ± 0.043	18055 ± 567	> 100	> 24	≈ 350
No memory	0.545 ± 0.189	3.974 ± 0.061	0.511 ± 0.062	3625 ± 1036	2.940 ± 0.734	0.262 ± 0.029	0.000 ± 0.000
Our design	0.750 ± 0.007	4.070 ± 0.026	0.615 ± 0.035	3312 ± 725	4.291 ± 1.743	0.390 ± 0.200	0.002 ± 0.000

Table 1: Experiments with different memory mechanisms on a 5-objective molecular optimization task, identical to the task in Table 2, using Gemini-2.5-flash without thinking mode.

to (i) **memory bloat** and higher prompt cost, (ii) **exploration collapse** with repeated proposals. Some runs of retrieval style memory were terminated early after repeatedly failing to produce novel valid candidates. Accordingly, the reported cost and runtime are lower bounds measured at termination, and memory is the approximate peak usage. We also observed a marked drop in uniqueness (sometimes even below 10%) and hypervolume, large LLM query costs, significantly longer running time and much larger storage space used compared to no memory and our design.

Inspired by summary-style experience from ReEvo and ExpeL, we keep one continually updated experience E_t into few hundreds of words. It captures actionable, transferable cues (e.g., frequently binding constraints, robust edit patterns, common invalid cases) without retrieving multiple entries. This avoids prompt bloat and reduces the risk of over-constraining exploration. From all historically evaluated candidates up to generation t, form an evidence set $D_t = G_t \cup B_t$, where G_t are the top-r candidates by fitness F and B_t are uniformly sampled from the lower-half of the fitness ranking. Let $S_\theta(\cdot)$ denote the *same LLM* used by the optimizer, prompted to produce a single concise memo (a few hundred words) that merges new evidence with prior experience. We update: $E_{t+1} = S_\theta\big(E_t, D_t\big)$ Operationally, S_θ folds in "good" and "bad" cases to reduce bias, promotes non-redundant micro-insights, and overwrites outdated content so that E_t remains compact, general and up-to-date. At generation $t{+}1$, E_{t+1} is inserted directly into the LLM context without retrievals. To prevent over-exploitation, the experience is included in the prompt with probability $p_{\text{exp}} \in [0, 1]$, sampled independently per call:

$$\mathbb{I}\{\text{inject } E_t\} \sim \text{Bernoulli}(p_{\text{exp}}).$$

This mechanism keeps prompts short on average, amortizes summarization cost (one update per generation), and maintains exploration headroom. We ablate p_{exp} in §5 and observe that intermediate values yield the best trade-off between sample efficiency and diversity.

3.2 Utilize Autoregressive Exploration by k-Offspring

How can we increase exploration within a closed-source LLM without finetuning? We exploit their autoregressive factorization to sample k offspring per parent in a single call. Because later proposals can condition on earlier samples within the same context, this yields *diverse-but-plausible* edits with low orchestration overhead. This strategy provides a simple way to strengthen exploratory breadth: under a fixed evaluation budget, generating the same number of candidates requires fewer LLM queries, fewer prompt tokens overall, and less running time than issuing one-off calls. However, overly large k may over-explore a local region and reduce marginal gains. We therefore study the trade-off by varying k and report the resulting gains and generalization under both single- and multi-objective settings in the ablations.

3.3 Handling Complex Feeback and Generalization

The adapter normalizes all objectives to $[0, 1]$ to prevent any large-magnitude objective from dominating the fitness; because our fitness follows a "larger-is-better" convention, we convert minimization goals to maximization by taking 1 minus their normalized values. (ii) Constraints and textual feedback are converted into a concise, formatted message that highlights violations, near-feasible margins, and expert hints.When a constraint is critical and variable, we optionally promote it to an explicit objective (e.g., minimize scaffold similarity or stellarator error), improving stability without parameter updates. We empirically validate these design choices and their impact on optimization in §7.4.3.

Based on the efficient and effective evolving experience designed for large discrete search spaces, as well as the easy-to-scale k-offspring mechanism and the unified feedback adapter, our framework

can be readily transferred to other problems and domains while maintaining strong performance. To facilitate adoption, we provide a simple template in which users only need to prepare two files: one specifying the task description and another defining the evaluation functions that return complex feedback. We demonstrate the strong performance of our framework across many problems and domains in Appendix 7.4.

4 EXPERIMENT

Task Settings **Five-objective optimization.** Following Wang et al. (2024a), we consider a five-objective molecular optimization task with a fixed evaluation budget of B=5,000 oracle calls. The objectives are: minimize SA (Synthetic Accessibility), DRD2 (Dopamine Receptor D2 affinity), and GSK3β (Glycogen Synthase Kinase 3 Beta); and maximize QED (Quantitative Estimate of Drug-likeness) and JNK3 (c-Jun N-terminal kinase 3). Although PMO (Gao et al., 2022) tasks are closely related to GuacaMol (Brown et al., 2019), PMO explicitly emphasizes budgeted evaluation due to the practical cost of property assessment (e.g., simulations or wet-lab proxies). A fixed budget thus encourages efficient exploration and enables fair comparison. While under a fixed budget, the initial population can substantially affect outcomes, yet this factor is often under-specified in evolution-based methods. To control for it, we compute fitness over ZINC250K (Irwin et al., 2012) and construct three fixed initial populations (size 100 each):**Best-init**: top-100 by fitness; **Worst-init**: bottom-100;**Random-init**: 100 uniformly sampled molecules. All algorithms are run from the same three initial populations, yielding a fair and comprehensive comparison. Best- and random-initialization reflect common practical use, whereas worst-initialization stresses robustness on a harder search. Beyond this five-objective setting, we follow the official PMO protocols and evaluate the full PMO benchmark to assess overall generality. For all experiments unless otherwise noted, ExLLM uses a fixed set of parameters with fixed proprietary LLM, experience injection probability p_{exp}=0.5, k=2, crossover probability 0.8, mutation probability 0.2, and population size 50. For the five-objective experiments, we reproduce MOLLEO using their code and use GPT-4o-2024-05-13 for both MOLLEO and our model.

Metrics Objectives are normalized and direction-unified as in Sec. 3.3; fitness F follows Eq. equation 1 with equal weights, consistent with Wang et al. (2024a). **Hypervolume.** Multi-objective coverage on the normalized (maximization) objective vectors using the reference point $\mathbf{r} = (1.1, \ldots, 1.1)$; larger values indicate better Pareto coverage. **AUC** Area under the curve of F versus oracle evaluations up to budget B; this rewards methods that reach high values with fewer oracle calls (Gao et al., 2022). **Validity**: fraction passing RDKit parsing. **Uniqueness**: fraction of unique molecules proposed. Novelty: fraction absent from ZINC250K. Diversity: average pairwise Tanimoto distance between the Morgan fingerprints within the top-100. **Efficiency** LLM queries, API cost (USD), wall-clock time (hours) under the same budget.

Baselines We compare against strong baselines spanning GA, BO, MCMC, RL, DL, and LLM-based methods: GB-GA, GB-BO, JT-VAE, MARS, REINVENT, MOLLEO, DyMol, and Genetic-GFN. For PMO-provided implementations (GB-GA, GB-BO, JT-VAE, MARS, REINVENT), we use the official PMO code and its default hyperparameters. For MOLLEO, DyMol, and Genetic-GFN, we use the authors' public code with their default settings as documented in code/papers. For RL-based baselines (REINVENT, DyMol, Genetic-GFN), we use the scalar fitness F as the reward signal during training.

4.1 FIVE-OBJECTIVE EXPERIMENT RESULTS

Table 10 reports mean $\pm$ std over five seeds; best scores are **bold** and second best are underlined. RL-based baselines (REINVENT, DyMol, Genetic-GFN) do not expose a mechanism to fix the initial population, so they are included only in the random-init condition. ExLLM attains the best F and AUC across all settings. For hypervolume, ExLLM is best under **worst-init** and **random-init**, and second best under **best-init**. In both random- and worst-initialization, the **mean Top-10** F of ExLLM exceeds the **Top-1** F of the second-best method, indicating strong improvement under the same budget. Novelty w.r.t. ZINC 250K is near 100% across models and settings; since this offers little differentiation for our objectives, we omit it from the table for space. Under best-init, MOLLEO matches ExLLM on Top-1 F, but ExLLM's mean Top-10 F and AUC remain higher

Metric	Initial	GB-GA	JT-VAE	GB-BO	MARS	REINVENT	MOLLEO	DyMol	Genetic-GFN	ExLLM(ours)
(Worst initial)										
Top1 F	2.689	4.048 ± 0.114	3.838 ± 0.042	3.665 ± 0.129	3.891 ± 0.018	-	4.096 ± 0.155	-	-	**4.229 ± 0.050**
Top10 F	2.683	4.019 ± 0.101	3.784 ± 0.027	3.647 ± 0.135	3.852 ± 0.019	-	4.044 ± 0.157	-	-	**4.186 ± 0.029**
AUC-Top10	-	3.789 ± 0.079	3.712 ± 0.027	3.489 ± 0.104	3.740 ± 0.010	-	3.825 ± 0.102	-	-	**3.915 ± 0.010**
Hypervolume	0.163	0.474 ± 0.190	0.364 ± 0.075	0.167 ± 0.050	0.488 ± 0.110	-	0.720 ± 0.172	-	-	**0.737 ± 0.038**
Diversity	0.876	0.583 ± 0.032	0.847 ± 0.007	0.657 ± 0.033	0.826 ± 0.011	-	0.656 ± 0.111	-	-	0.603 ± 0.055
Uniqueness	-	0.786 ± 0.032	1.000 ± 0.000	1.000 ± 0.000	0.488 ± 0.128	-	0.672 ± 0.032	-	-	0.829 ± 0.021
Validity	-	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	0.930 ± 0.075	-	-	0.937 ± 0.010
(Random initial)										
Top1 F	3.804	4.017 ± 0.095	3.874 ± 0.067	4.003 ± 0.121	3.931 ± 0.055	4.230 ± 0.196	4.190 ± 0.076	4.232 ± 0.170	4.243 ± 0.253	**4.336 ± 0.246**
Top10 F	3.741	3.975 ± 0.095	3.831 ± 0.019	3.968 ± 0.104	3.868 ± 0.025	4.136 ± 0.219	4.076 ± 0.020	4.164 ± 0.132	4.202 ± 0.213	**4.300 ± 0.164**
AUC-Top10	-	3.861 ± 0.052	3.771 ± 0.011	3.802 ± 0.057	3.789 ± 0.011	3.930 ± 0.133	3.949 ± 0.021	4.001 ± 0.054	4.078 ± 0.150	**4.116 ± 0.040**
Hypervolume	0.236	0.643 ± 0.268	0.428 ± 0.127	0.507 ± 0.287	0.409 ± 0.111	0.742 ± 0.259	0.860 ± 0.088	0.868 ± 0.146	0.871 ± 0.288	**0.905 ± 0.200**
Diversity	0.884	0.623 ± 0.047	0.778 ± 0.012	0.717 ± 0.017	0.819 ± 0.015	0.640 ± 0.111	0.670 ± 0.015	0.581 ± 0.069	0.633 ± 0.066	0.494 ± 0.032
Uniqueness	-	0.821 ± 0.032	0.956 ± 0.005	1.000 ± 0.000	0.477 ± 0.120	0.690 ± 0.132	0.575 ± 0.075	0.986 ± 0.005	0.349 ± 0.004	0.872 ± 0.015
Validity	-	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.999 ± 0.000	0.979 ± 0.002	0.938 ± 0.007	1.000 ± 0.000	0.998 ± 0.000	0.908 ± 0.019
(Best initial)										
Top1 F	4.329	4.583 ± 0.154	4.329 ± 0.000	4.605 ± 0.047	4.419 ± 0.074	-	**4.699 ± 0.000**	-	-	**4.699 ± 0.000**
Top10 F	4.132	4.582 ± 0.167	4.132 ± 0.000	4.467 ± 0.066	4.181 ± 0.029	-	4.564 ± 0.064	-	-	**4.628 ± 0.043**
AUC-Top10	-	4.130 ± 0.088	4.091 ± 0.000	4.237 ± 0.112	4.137 ± 0.028	-	4.362 ± 0.075	-	-	**4.481 ± 0.055**
Hypervolume	0.917	0.968 ± 0.183	0.917 ± 0.000	**1.275 ± 0.027**	0.975 ± 0.041	-	1.168 ± 0.106	-	-	1.175 ± 0.067
Diversity	0.793	0.424 ± 0.070	0.792 ± 0.001	0.630 ± 0.024	0.788 ± 0.002	-	0.600 ± 0.052	-	-	0.491 ± 0.057
Uniqueness	-	0.729 ± 0.041	1.000 ± 0.000	1.000 ± 0.000	0.432 ± 0.053	-	0.678 ± 0.000	-	-	0.942 ± 0.006
Validity	-	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.999 ± 0.000	-	0.913 ± 0.022	-	-	0.790 ± 0.024

Table 2: Unconstrained molecular design results, objectives: QED↑ + SA↓ + DRD2↓ + GSK3β ↓ + JNK3↑

than all other methods, suggesting more reliable batch-level gains. ExLLM maintains **80–95% uniqueness** across three settings, while MOLLEO is typically **55–70%**, because they use the same LLM, this indicates our framework has higher exploration ability. Validity is slightly lower than some baselines because we generate SMILES and discard invalid strings rather than post-hoc repairing them; this has negligible impact on final outcomes, and PMO notes SMILES is not necessarily inferior to 100%-valid representations (Gao et al., 2022). The diversity of the final top-100 set is somewhat lower, which reflects a common fitness–diversity trade-off: tighter, high-value exploration can reduce spread. Notably, ExLLM delivers substantial gains over the initial populations in all three init schemes, while trading some diversity for finer exploitation; coverage remains competitive as evidenced by strong hypervolume.

4.2 Results on PMO

We evaluate ExLLM on the full PMO suite covering property, name-based, and structure-based optimization (Table 3). Best values are **bold**; the second and third best are underlined. Following the official scoring, we report an aggregate leaderboard score with a maximum of 23 points. ExLLM attains a total score of 19.165, ranking first in 17/23 tasks and achieving the highest overall ranking. Compared to the previous SOTA (17.862), this corresponds to a +7.3% improvement in the aggregate score. To assess the contribution of the experience module, we also report an ablation without the evolving experience (ExLLM w/o experience), which still reaches **18.165**. This result suggests that the k-offspring exploration is a strong contributor on its own, while the evolving experience provides consistent additional gains overall.

4.3 Extended Experiments

We put the full tables and plots of the following experiments in the appendix. **Cross-domain transfer.** ExLLM achieves new records on circle packing and stellarator design, and shows strong performance on MOTSP/MOCVRP, SACS, NK2R peptide, and GCU operator design, with minimal task-specific changes. **Constraint handling.** We compare prompt-only penalties with promoting constraints to explicit objectives. The latter yields more stable optimization and better budgeted performance on tasks where constraint values are critical and variable.

Task type	Objective(↑)	REINVENT	Augmented Memory	Graph GA	GP BO	MOLLEO	Genetic GFN	ExLLM (Ours)
Property optimization	QED	0.941 ± 0.000	0.941 ± 0.000	0.940 ± 0.000	0.937 ± 0.000	**0.948 ± 0.000**	0.942 ± 0.000	0.943 ± 0.000
	JNK3	0.783 ± 0.023	0.773 ± 0.073	0.553 ± 0.136	0.564 ± 0.155	**0.790 ± 0.027**	0.764 ± 0.069	0.732 ± 0.078
	DRD2	0.945 ± 0.007	0.962 ± 0.005	0.964 ± 0.012	0.923 ± 0.017	0.968 ± 0.012	0.974 ± 0.006	**0.980 ± 0.003**
	GSK3β	0.865 ± 0.043	**0.889 ± 0.027**	0.788 ± 0.070	0.851 ± 0.041	0.863 ± 0.047	0.881 ± 0.042	0.818 ± 0.050
Name-based optimization	mestranol_similarity	0.618 ± 0.048	0.764 ± 0.035	0.579 ± 0.022	0.627 ± 0.089	0.972 ± 0.009	0.708 ± 0.057	**0.980 ± 0.005**
	albuterol_similarity	0.896 ± 0.008	0.918 ± 0.026	0.874 ± 0.020	0.902 ± 0.019	0.985 ± 0.024	0.949 ± 0.010	**0.989 ± 0.000**
	thiothixene_rediscovery	0.534 ± 0.013	0.562 ± 0.028	0.479 ± 0.025	0.559 ± 0.027	0.727 ± 0.052	0.583 ± 0.034	**0.910 ± 0.004**
	celecoxib_rediscovery	0.716 ± 0.084	0.784 ± 0.011	0.582 ± 0.057	0.728 ± 0.048	0.864 ± 0.034	**0.891 ± 0.033**	**0.891 ± 0.033**
	troglitazone_rediscovery	0.452 ± 0.048	0.556 ± 0.052	0.377 ± 0.010	0.405 ± 0.007	0.562 ± 0.019	0.511 ± 0.054	**0.726 ± 0.111**
	perindopril_mpo	0.537 ± 0.016	0.598 ± 0.008	0.538 ± 0.009	0.493 ± 0.011	0.600 ± 0.031	0.595 ± 0.014	**0.797 ± 0.016**
	ranolazine_mpo	0.760 ± 0.009	0.802 ± 0.003	0.728 ± 0.012	0.735 ± 0.013	0.769 ± 0.022	0.819 ± 0.018	**0.855 ± 0.021**
	sitagliptin_mpo	0.021 ± 0.003	0.479 ± 0.039	0.433 ± 0.075	0.186 ± 0.055	0.584 ± 0.067	**0.634 ± 0.039**	0.555 ± 0.048
	amlodipine_mpo	0.642 ± 0.044	0.686 ± 0.046	0.625 ± 0.040	0.552 ± 0.025	0.773 ± 0.037	0.761 ± 0.019	**0.874 ± 0.010**
	fexofenadine_mpo	0.769 ± 0.009	0.686 ± 0.010	0.779 ± 0.025	0.745 ± 0.009	0.847 ± 0.018	0.856 ± 0.039	**0.984 ± 0.006**
	osimertinib_mpo	0.834 ± 0.046	0.856 ± 0.013	0.808 ± 0.012	0.762 ± 0.029	0.835 ± 0.024	0.860 ± 0.008	**0.902 ± 0.018**
	zaleplon_mpo	0.347 ± 0.049	0.438 ± 0.082	0.456 ± 0.007	0.272 ± 0.026	0.510 ± 0.031	0.552 ± 0.033	**0.723 ± 0.007**
	median1	0.372 ± 0.015	0.335 ± 0.012	0.287 ± 0.008	0.325 ± 0.012	0.352 ± 0.024	0.379 ± 0.010	**0.384 ± 0.007**
	median2	0.294 ± 0.006	0.290 ± 0.006	0.229 ± 0.017	0.308 ± 0.034	0.275 ± 0.045	0.294 ± 0.007	**0.475 ± 0.002**
Structure-based optimization	isomers_c7h8n2o2	0.842 ± 0.029	0.954 ± 0.033	0.949 ± 0.036	0.662 ± 0.071	**0.984 ± 0.008**	0.969 ± 0.003	**0.984 ± 0.001**
	isomers_c9h10n2o2pf2cl	0.642 ± 0.054	0.830 ± 0.016	0.719 ± 0.047	0.469 ± 0.180	0.874 ± 0.053	0.897 ± 0.007	**0.961 ± 0.028**
	deco_hop	0.666 ± 0.044	0.688 ± 0.060	0.619 ± 0.004	0.629 ± 0.018	0.942 ± 0.013	0.733 ± 0.109	**0.956 ± 0.014**
	scaffold_hop	0.560 ± 0.019	0.565 ± 0.008	0.517 ± 0.007	0.548 ± 0.019	**0.971 ± 0.004**	0.615 ± 0.100	0.916 ± 0.127
	valsartan_smarts	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	**0.867 ± 0.092**	0.135 ± 0.271	0.831 ± 0.043
	Total (↑)	14.036	15.356	13.823	13.182	17.862	16.213	**19.165**
	Rank (↓)	5	4	6	7	2	3	**1**

Table 3: Top-10 AUC of tasks in PMO (Gao et al., 2022) benchmark, including single-objective optimization and multi-objective optimization for 3 task types. ExLLM attains a total score of **19.165** (**+7.3%** improvement compared to the previous SOTA by MOLLEO), ranking **first in 17/23 tasks** and achieving the highest overall ranking.

5 Ablation Study

To fully characterize the framework, we ask: (1) how to determine k and what's the trade-off?; (2) how frequently should evolving experience be injected; (3) how does performance change with number of objectives; (4) how should constraints be handled for stable, budgeted gains? We hence quantify budgeted gains vs. local over-exploration by varying $k \in \{1, \ldots, 6\}$ in single-/multi-objective settings with two LLM backbones. And then identify when experience helps vs. hinders exploration by sweeping $p_{\text{exp}} \in [0, 1]$. We finally Tests scalability from 1 to 6 objectives including comparison with MOLLEO. Detailed tables and the 4^{th} ablation study are in the appendix.

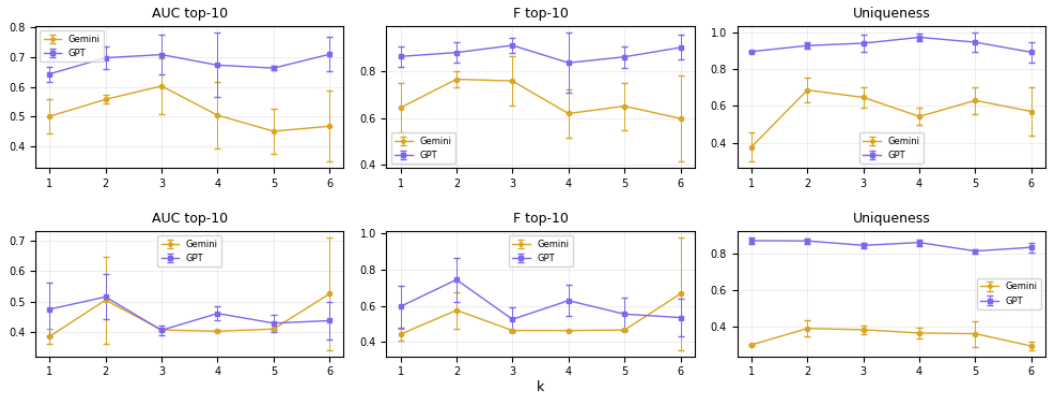

Figure 2: k-offspring ablation studies. Top: single-objective optimization (JNK3). Bottom: five-objective optimization. The most consistent improvement on results and exploration occurs at k=2 and 3, while higher k may lead to local over-exploration.

k-offspring trade-offs Figure 2 shows the results of varying k without using experience. For the five-objective plot, we subtract 3.5 from both AUC and F before plotting to improve readability. Increasing k improves exploratory breadth, especially in the single-objective JNK3 task, yet overly

large k tends to over-explore locally and limits global search under a fixed budget. Across tasks, k=2 yields the largest and most stable gains in our sweep; k=3 is less stable, and larger k becomes increasingly unstable or degrades improvement.

P_{exp}	Top10 F	AUC-Top10	Hypervolume	Uniqueness	Validity	Diversity
0.0	4.245 ± 0.121	4.015 ± 0.074	0.850 ± 0.224	0.870 ± 0.014	0.900 ± 0.003	0.556 ± 0.106
0.3	4.260 ± 0.145	4.004 ± 0.077	0.881 ± 0.201	0.879 ± 0.014	0.905 ± 0.010	0.509 ± 0.057
0.5	**4.301 ± 0.164**	**4.116 ± 0.040**	**0.905 ± 0.200**	0.872 ± 0.015	0.908 ± 0.019	0.494 ± 0.032
0.7	3.986 ± 0.032	3.892 ± 0.007	0.555 ± 0.188	0.843 ± 0.016	0.930 ± 0.015	0.586 ± 0.070
0.9	4.030 ± 0.098	3.923 ± 0.051	0.571 ± 0.228	0.856 ± 0.020	0.926 ± 0.002	0.496 ± 0.021

Table 4: The ablation study of P_{exp} shows that incorporating experience can notably improve performance and convergence, but it must be properly controlled to avoid restricting the exploration direction.

Experience Injection Our evolving experience is lightweight and general, yielding steady performance gains with only a modest impact on diversity. That said, in molecular optimization which has large discrete spaces, P_{exp}should be properly controlled to prevent over-conditioning the search and restricting exploration.

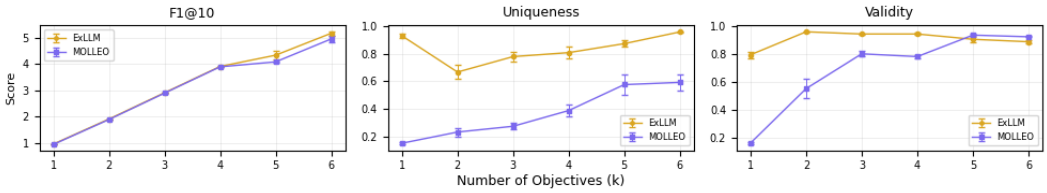

Figure 3: Experiment of different number of objectives.

More number of Objectives We conduct experiments using random initialization across scenarios with one to six objectives, starting with only QED↑ to QED↑ + SA↓ + DRD2↓ + GSK3β ↓ + JNK3↑ + BBBP↑ (Blood-Brain Barrier Permeability) as shown in figure 3 (full table in appendix 7.6). BBBP (Blood-Brain Barrier Permeability) as a sixth objective, as it is a more complex and less predictable property with limited domain knowledge. As the number of objectives increases, the performance gap between ExLLM and MOLLEO widens, particularly when optimizing more than four objectives, showing the gains of increasing complexity of the task. In MOLLEO uniqueness and validity tend to degrade significantly when optimizing fewer objectives, this indicates the exploration ability is seriously affected. ExLLM consistently achieves high uniqueness and validity, showing its stable exploration ability across different task complexity. We put reproducibility in appendix 7.1.

6 Conclusion

We presented **ExLLM**, an LLM-as-optimizer framework for large discrete optimization, instantiated for multi-objective molecular design. The method couples three simple components: a *single, evolving experience* that distills non-redundant cues at low cost; a *k-offspring* sampling scheme that widens exploration per query; and a *feedback adapter* that normalizes objectives for selection while formatting constraints and expert hints for iteration. Under a fixed evaluation budget, ExLLM attains state-of-the-art results on PMO (total score **19.165**, ranking first on **17/23** tasks, **+7.3%** over prior SOTA) and generalizes beyond chemistry, setting new records on circle packing and stellarator design while delivering strong performance across additional domains. The probabilistic experience injection improves efficiency without over-constraining exploration. Looking forward, we plan to (i) adapt k and the experience-injection rate online, (ii) incorporate oracle uncertainty and 3D/physics-informed feedback, and (iii) broaden evaluation to additional scientific design tasks. We release templates for rapid transfer, lowering the barrier to applying ExLLM as a general optimizer for large discrete spaces.

REFERENCES

ANM Nafiz Abeer, Nathan M Urban, M Ryan Weil, Francis J Alexander, and Byung-Jun Yoon. Multi-objective latent space optimization of generative molecular design models. *Patterns*, 5(10), 2024.

Microsoft Research AI4Science and Microsoft Azure Quantum. The impact of large language models on scientific discovery: a preliminary study using gpt-4. *arXiv preprint arXiv:2311.07361*, 2023.

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

Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *Ieee access*, 8: 89497–89509, 2020.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Shuvayan Brahmachary, Subodh M Joshi, Aniruddha Panda, Kaushik Koneripalli, Arun Kumar Sagotra, Harshil Patel, Ankush Sharma, Ameya D Jagtap, and Kaushic Kalyanaraman. Large language model-based evolutionary optimizer: Reasoning with elitism. *arXiv preprint arXiv:2403.02054*, 2024.

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3): 1096–1108, 2019.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Santiago A Cadena, Andrea Merlo, Emanuel Laude, Alexander Bauer, Atul Agrawal, Maria Pascu, Marija Savtchouk, Enrico Guiraud, Lukas Bonauer, Stuart Hudson, et al. Constellaration: A dataset of qi-like stellarator plasma boundaries and optimization benchmarks. *arXiv preprint arXiv:2506.19583*, 2025.

Yin Fang, Ningyu Zhang, Zhuo Chen, Lingbing Guo, Xiaohui Fan, and Huajun Chen. Domain-agnostic molecular generation with chemical feedback. In *The Twelfth International Conference on Learning Representations*, 2024.

Erich Friedman. Erich's packing center. `https://erich-friedman.github.io/packing/`, 2025. Accessed: 2025-09-22.

Tianfan Fu, Wenhao Gao, Cao Xiao, Jacob Yasonik, Connor W Coley, and Jimeng Sun. Differentiable scaffolding tree for molecular optimization. *arXiv preprint arXiv:2109.10469*, 2021.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems*, 35: 21342–21357, 2022.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

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

Beichen Huang, Xingyu Wu, Yu Zhou, Jibin Wu, Liang Feng, Ran Cheng, and Kay Chen Tan. Exploring the true potential: Evaluating the black-box optimization capability of large language models. *arXiv preprint arXiv:2404.06290*, 2024.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7): 1757–1768, 2012.

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

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. Aide: Ai-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018a.

Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecular optimization. *arXiv preprint arXiv:1812.01070*, 2018b.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *International conference on machine learning*, pp. 4849–4859. PMLR, 2020.

Hyeonah Kim, Minsu Kim, Sanghyeok Choi, and Jinkyoo Park. Genetic-guided GFlownets for sample efficient molecular optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=B4q98aAZwt.

Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. In *International Conference on Machine Learning*, pp. 18872–18892. PMLR, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Jiatong Li, Yunqing Liu, Wenqi Fan, Xiao-Yong Wei, Hui Liu, Jiliang Tang, and Qing Li. Empowering molecule discovery for molecule-caption translation with large language models: A chatgpt perspective. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. *arXiv preprint arXiv:2203.15386*, 2022.

Fei Liu, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Algorithm evolution using large language model. *arXiv preprint arXiv:2311.15249*, 2023.

Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Forty-first International Conference on Machine Learning*, 2024a.

Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. Large language models as evolutionary optimizers. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. IEEE, 2024b.

Wanyi Liu, Long Chen, and Zhenzhou Tang. Large language model aided multi-objective evolutionary algorithm: a low-cost adaptive approach. *arXiv preprint arXiv:2410.02301*, 2024c.

Yiping Liu, Jiahao Yang, Zhang Xinyi, Yuansheng Liu, Bosheng Song, Hisao Ishibuchi, et al. Multi-objective molecular design through learning latent pareto set.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, pp. 1–11, 2024.

Saeed Moayedpour, Alejandro Corrochano-Navarro, Faryad Sahneh, Shahriar Noroozizadeh, Alexander Koetter, Jiri Vymetal, Lorenzo Kogler-Anele, Pablo Mas, Yasser Jangjou, Sizhen Li, et al. Many-shot in-context learning for molecular inverse design. *arXiv preprint arXiv:2407.19089*, 2024.

Ali Modarressi, Abdullatif Köksal, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. Memllm: Finetuning llms to use an explicit read-write memory. *arXiv preprint arXiv:2404.11672*, 2024.

Tung Nguyen and Aditya Grover. Lico: Large language models for in-context molecular optimization. *arXiv preprint arXiv:2406.18851*, 2024.

AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alán Aspuru-Guzik. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. *arXiv preprint arXiv:1909.11655*, 2019.

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

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

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9:1–14, 2017.

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

Frederike Sass, Tao Ma, Jeppe H Ekberg, Melissa Kirigiti, Mario G Ureña, Lucile Dollet, Jenny M Brown, Astrid L Basse, Warren T Yacawych, Hayley B Burm, et al. Nk2r control of energy expenditure and feeding to treat metabolic diseases. *Nature*, 635(8040):987–1000, 2024.

Dong-Hee Shin, Young-Han Son, Deok-Joong Lee, Ji-Wung Han, and Tae-Eui Kam. Dynamic many-objective molecular optimization: Unfolding complexity with objective decomposition and progressive optimization. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 6026–6034. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/666. URL `https://doi.org/10.24963/ijcai.2024/666`. Main Track.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Mengying Sun, Jing Xing, Han Meng, Huijun Wang, Bin Chen, and Jiayu Zhou. Molsearch: search-based multi-objective molecular generation and property optimization. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 4724–4732, 2022.

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

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Jonas Verhellen. Graph-based molecular pareto optimisation. *Chemical Science*, 13(25):7526–7535, 2022.

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Streith-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, et al. Efficient evolutionary search over chemical space with large language models. *arXiv preprint arXiv:2406.16976*, 2024a.

Zeyi Wang, Songbai Liu, Jianyong Chen, and Kay Chen Tan. Large language model-aided evolutionary search for constrained multiobjective optimization. In *International Conference on Intelligent Computing*, pp. 218–230. Springer, 2024b.

Xingyu Wu, Sheng-hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. Evolutionary computation in the era of large language model: Survey and roadmap. *arXiv preprint arXiv:2401.10034*, 2024a.

Zhenxing Wu, Odin Zhang, Xiaorui Wang, Li Fu, Huifeng Zhao, Jike Wang, Hongyan Du, Dejun Jiang, Yafeng Deng, Dongsheng Cao, et al. Leveraging language model for advanced multiproperty molecular optimization via prompt engineering. *Nature Machine Intelligence*, pp. 1–11, 2024b.

Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. Mars: Markov molecular sampling for multi-objective drug discovery. *arXiv preprint arXiv:2103.10432*, 2021.

Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2024. URL https://arxiv.org/abs/2309.03409.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. *Advances in neural information processing systems*, 37:43571–43608, 2024.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners, 2023. URL https://arxiv.org/abs/2308.10144.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024.

7 APPENDIX

7.1 REPRODUCIBILITY

We commit to releasing the full implementation upon acceptance of the paper. The public repository will include all code to reproduce the results reported in this work, covering not only the experiments in the main text but also the extended templates and evaluation functions for other domains. All prompt templates used in our optimizer will also be released to ensure transparency and reproducibility.

7.2 RELATED WORK

7.3 LLM-AS-OPTIMIZER AND MEMORY MECHANISM

Recent work tends to use LLM as optimizer to utilize domain knowledge and reasoning ability to guide efficient high-dimensional exploration without training (AI4Science & Quantum, 2023; Wu et al., 2024a; Brown, 2020). OPRO and LMEA cast LLMs as crossover/mutation operators within GA-style loops, balancing exploitation–exploration via prompt design and temperature control (Yang et al., 2024; Liu et al., 2024b); AlphaEvolve further systematizes this LLM-in-the-loop evolutionary pattern, and ReAct exemplifies reasoning–acting procedures that can guide decision making (Novikov et al., 2025a; Yao et al., 2022). In molecular settings, constrained prompt engineering shows encouraging alignment effects (Wang et al., 2024b), while several studies report that GA+LLM pipelines can be efficient and competitive against standalone LLMs and traditional MOO when coupled with well-structured workflows (Liu et al., 2023; 2024a;c; Huang et al., 2024; Brahmachary et al., 2024), indicating LLM as a promising optimizer for numerical optimizations, coding and planning problems.

External memory is widely used to inject up-to-date knowledge and reduce parametric forgetting—ranging from RAG (retrieve-then-concatenate) (Lewis et al., 2020) and RETRO (cross-attending to retrieved neighbors) (Borgeaud et al., 2022) to persistent, editable stores such as MemoryBank (human-like forgetting/refresh) and MemLLM (explicit read/write) (Zhong et al., 2024; Modarressi et al., 2024). Hybrid controllers like A-Mem and RL-trained Memory-R1 further learn when to add/update/delete/consume memories (Xu et al., 2025; Yan et al., 2025). While effective for QA, code and planning tasks, these designs are not tailored to large discrete optimization: massive iteration counts and huge candidate spaces make append-only or dense-retrieval memories prone to memory bloat and redundancy, drive up per-query prompt cost, and bias exploration when similar histories are repeatedly injected, as shown in table 1. We address this by a compact, low-redundancy experience pool that is both efficient and effective with much lower costs.

7.4 MORE APPLICATIONS

7.4.1 CIRCLE PACKING IN A UNIT SQUARE

Task. Place n circles inside a unit square to maximize the common radius without overlap or boundary violations; this classic geometric packing problem has long-standing community records and curated best-known configurations (Friedman, 2025; Novikov et al., 2025a).

Challenges for our optimizer. (1) *End-to-end instability*: tiny coordinate nudges can flip feasibility, making naive prompt-only updates easily to voliate the constraints. (2) *Variable abstraction*: the raw state of the problem is described only by a list of circle centers and their individual radii (x_i, y_i, r_i). This representation is highly abstract and purely numerical, making it difficult for an LLM to derive intuitive geometric insights directly from the prompt. And the feedback is only the total radius.

Method	n=26	n=27	n=28	n=29	n=30	n=31	n=32
Current record Friedman (2025)	2.634+	2.685+	2.737+	2.790+	2.842+	2.889+	2.936+
AlphaEvolve	2.635977	-	-	-	-	-	2.937+
ExLLM	2.635983	2.65898	2.73740	2.79034	2.84267	2.88997	2.939+

Table 5: Circle packing in a unit square results for n=26 to 32.

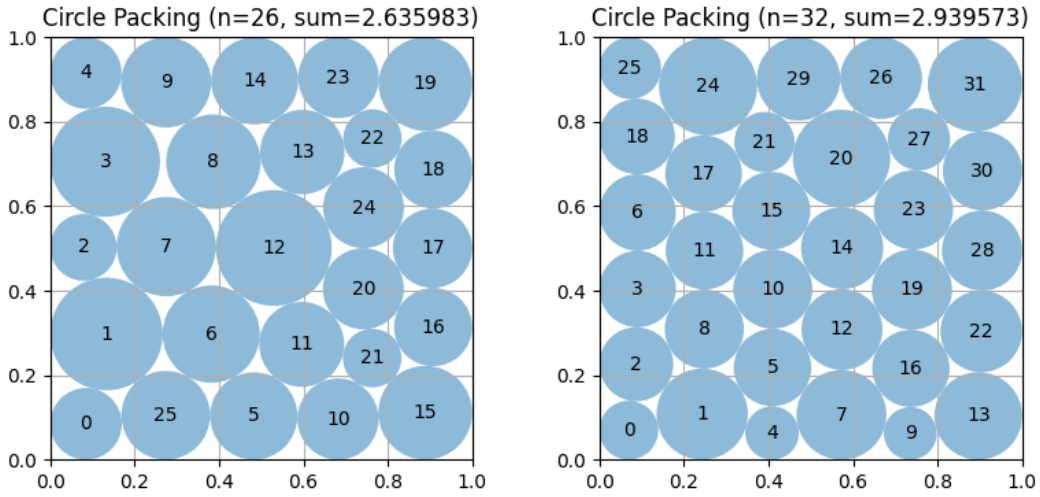

Figure 4: Best layouts found by our optimizer: (left) $n = 26$, (right) $n = 32$.

Our approach. We keep the ExLLM optimizer fixed (same hyperparameters as in the main text) and modify only the task template and the evaluation function. The template succinctly describes the target (pack n circles in a unit square) and specifies a simple output format (center coordinates and radii). To stabilize end-to-end outcomes, we post-process ExLLM's proposals with a *fixed* off-the-shelf solver—SLSQP from *SciPy*—which enforces non-overlap and boundary constraints and locally increases the radii given the LLM's initial centers/radii. In contrast to methods that retrain or tune a bespoke solver (e.g., evolving the solver itself), we hold the optimizer constant and use a fixed solver purely for feasibility/finetuning, while ExLLM focuses on generating strong initializations.

Results and visualization. All results are shown in figure 4 and table 5. In our setup, we obtain new best-known results for $n = 26$ and $n = 32$. For $n = 27$–31, our solutions match the publicly reported records up to the available three-decimal precision; because reference values beyond three decimals are not published, we cannot determine whether we strictly surpass them. We include figures of our best layouts and a table of achieved radii n. Notably, our algorithm often discovers multiple distinct arrangements that attain the same maximal score. The experiments were conducted with 2500 evaluation budget, and it takes about only 3 hours to complete and find achieved new records.

7.4.2 Stellarator Design

Task. Optimize quasi-isodynamic (QI) stellarator plasma boundaries under strict physics and engineering constraints using the ConStellaration benchmark (Cadena et al., 2025): (P2) *Simple-to-build QI* (favor coil simplicity subject to QI/geometry constraints) and (P3) *MHD-stable QI* (trade off compactness vs. coil simplicity under MHD stability and turbulence-proxy constraints). We adopt the official scoring and evaluation protocol of the benchmark release.

Challenges for our optimizer. (1) Constraint hardness & multiplicity: Problem 2 is single objective optimization with 5 constraints, and problem 3 is even harder with 2 objective optimization and 5 constraints including edge rotational transform, quasi isodynamicity residual, vacuum well etc. (2) Sparse nonzero solution: setting the number of field periods to 3 according to official code, sweeping the about 180k released designs yields no nonzero scores under the strict benchmark tolerances. (3) The official gradient-based (`scipy-trust-constr`) and derivative-free (`COBYQA`) baselines fail to produce feasible points on P2/P3; only the ALM–NGOpt pipeline attains nonzero scores, and sparsely so.

Our approach. We keep the ExLLM optimizer fixed (same hyperparameters as in the main text) and change only the task template and evaluation to match P2/P3. Because feasibility is *hard*—any violation yields a zero score—and the key feasibility indicator is a measurable scalar (the feasibility score must be <0.01), we treat this constraint as an **explicit objective**. Concretely, besides using the official targets, we add a feasibility term that rewards minimizing feasibility score; this follows our

adapter's rule for promoting critical, variable constraints into objectives. In practice, this substantially accelerates convergence to nonzero scores and stabilizes progress, the proportion of nonzero scores largely increases.

Results and visualization. On both benchmarks we surpass the official SOTA (ALM–NGOpt), obtaining many feasible points in a single run while tightly respecting constraints, 17% improvement on problem 2 and 3% improvement on problem 3. Figure 5 shows visualization of our best solutions; Table 6 compares scores.

Method	Simple-to-build QI (P2) ↑	MHD-stable QI (P3) ↑
Official `scipy-trust-constr`	Failed	Failed
Official `COBYQA`	Failed	Failed
Official best (ALM–NGOpt)	0.431	129.796 ; (Best single point: 97)
MOLLM (Ours)	**0.505**	**133.634** ; (Best single point: 103)

Table 6: ConStellaration benchmarks: ExLLM-based optimizer (MOLLM) exceeds the official SOTA on both P2 and P3, while generating hundreds of nonzero feasible points per run and adhering closely to the constraint set.

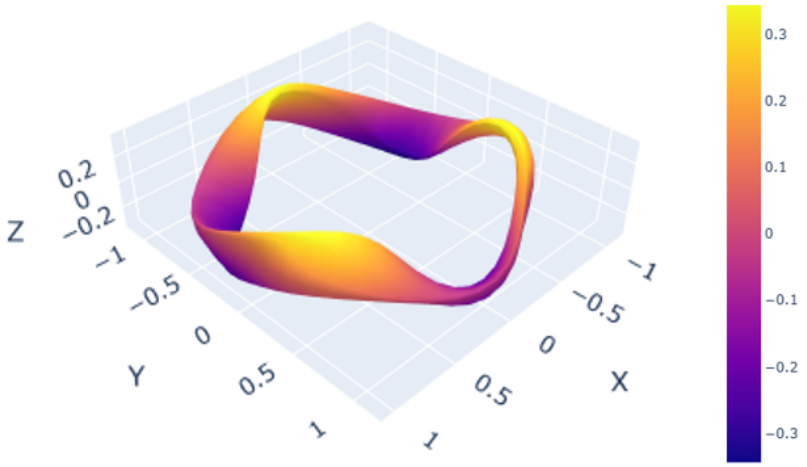

Figure 5: A visualization of our best solution for problem 2.

7.4.3 MOCPOP (Multi-Objective Combinatorial Path Optimization Problems)

Task. The Multi-Objective Traveling Salesman Problem (MOTSP) and the Multi-Objective Capacitated Vehicle Routing Problem (MOCVRP) are both classical NP-hard combinatorial optimization tasks. MOTSP seeks a single Hamiltonian circuit that, starting and ending at a given depot, visits every city exactly once while simultaneously minimizing multiple conflicting objectives. MOCVRP designs a set of capacitated vehicle routes that originate from a common depot, serve all customer demands, and jointly minimize the total travel distance and the makespan. All benchmark instances used in this study were generated with the scalable generator proposed by Lin et al. (2022)

Challenges for our optimizer. (1) Implicit dimensional coupling: MOTSP exhibits a scale–sensitive embedding, a microscopic rescaling of one coordinate can flip the relative distances of a city pair from the global minimum to the global maximum. Consequently, the gradient direction that appears Pareto-improving in one scalarization step may become strongly misleading after an infinitesimal δ-shift, rendering classical directional updates ineffective and causing severe oscillations on the Pareto front. (2) Pareto-conflict trap: In MOCVRP, the total travel distance and the makespan form a deeply antagonistic pair, minimizing the former tends to squeeze all routes into a few "spokes," whereas minimizing the latter forces a balanced, yet longer, set of tours. A single scalarized surrogate—no matter how sophisticated the weight schedule—collapses the true Pareto front into a narrow ridge and inevitably misses the knee regions that dominate most practical trade-offs. (3) Destructive heuristic search bias: Offspring generated by canonical route-based crossover (e.g., OX, MPX)

inherit disconnected fragments or violate capacity constraints with > 90 % probability. The resulting infeasible individuals are rejected outright, so the search wastes the vast majority of evaluations and the effective population size collapses—an effect we term "lethal recombination drag." Advanced repair mechanisms partially alleviate the issue, yet they simultaneously erode the schema that parental high-fitness routes originally encoded, decelerating convergence toward the Pareto set

Our approach. For MOCPOP, our ExLLM employs deliberately fuzzy prompts that fully exploit the intrinsic experience and reasoning capacity of the large language model. The model proposes inference-based, feasible offspring instead of rigidly applying heuristics such as OX or PMX, which would otherwise produce a high proportion of invalid children. We also integrate an LLM + Solver scheme: after the large model generates high-quality feasible solutions, a solver—guided by a heuristic also proposed by the LLM—further refines these solutions to produce a superior population. This hybrid strategy guarantees both the quality and the speed of ExLLM's search.

Results. Across both benchmarks, ExLLM attains highly competitive performance (Table 8). MOTSP instances comprise $n = 100$ cities, and MOCVRP instances comprise $n = 100$ customers and $m = 20$ vehicles. We adopt hyper-volume as the universal performance indicator. Baselines include the human-designed solver Pymoo (Blank & Deb, 2020), as well as recent search-based algorithms ReEvo (Ye et al., 2024), AlphaEvolve (Novikov et al., 2025b), and other representative model. ExLLM achieves new SOTA hyper-volume on MOCVRP and ranks second only to AlphaEvolve on MOTSP, demonstrating the effectiveness of the LLM-guided hybrid paradigm.

Method	**MOTSP(n=100)**	**MOCVRP(n=50,m=20)**
Pymoo	0.983488	0.955802
ReEvo	1.028890	<u>1.034541</u>
GreedyRefine (Shinn et al., 2023)	1.031825	1.031764
AIDE (Jiang et al., 2025)	1.020798	1.005552
Funsearch (Romera-Paredes et al., 2024)	1.023301	1.032126
AlphaEvolve	**1.029279**	1.031803
ExLLM(ours)	<u>1.027333</u>	**1.070412**

Table 7: MOCPOP benchmark: ExLLM surpasses SOTA hyper-volume on MOCVRP and ranks second to AlphaEvolve on MOTSP.

7.4.4 SACS (STRUCTURAL ANALYSIS COMPUTER SYSTEM)

Task. To validate the cross-domain transferability of our framework, we apply ExLLM to a challenging real-world engineering problem: the structural optimization of offshore platform jackets. This system intelligently adjusts the dimensions of structural components to simultaneously achieve three conflicting goals: 1) Minimize the structural weight to reduce material and construction costs; 2) Ensure structural strength is sufficient to withstand extreme axial load conditions; and 3) Ensure structural strength is sufficient to resist extreme bending and torsional loads. The optimization is performed using the SACS finite element analysis software as the evaluation oracle.

Challenges for our optimizer. (1) *Complex mixed-variable parameter space*: The optimization involves a hybrid of discrete and continuous variables. It requires selecting standard cross-sections from a catalog using discrete string representations, while simultaneously optimizing other continuous design parameters. This mixed-variable nature presents a significant challenge for many optimization algorithms. (2) *Computationally expensive feedback loop*: Each proposed design must be evaluated using the SACS finite element software, which involves computationally intensive simulations. This places a strict limit on the number of evaluations possible, demanding high sample efficiency from the optimizer. (3) *Balancing competing objectives*: The goals of minimizing weight while maximizing strength against two different types of critical loads are inherently contradictory. Finding a balanced solution on the Pareto front requires sophisticated exploration and exploitation strategies. (4) *Black-box feedback*: The optimizer receives performance metrics (weight, stress, displacement) from SACS without direct access to the software's internal gradients, treating it as a black box.

Our approach. Demonstrating the framework's 'plug-and-play' capability, we kept the core ExLLM optimizer and its hyperparameters identical to those used in the molecular design experiments. The transfer to this new domain required only the creation of a task-specific template—which instructs the LLM to propose modifications to a vector of design parameters—and a corresponding evaluation function that interfaces with the SACS software. The feedback adapter then processes the results

from SACS—structural weight and stress safety factors—and formats them for the next optimization iteration. The evolving experience mechanism is particularly valuable for identifying promising design patterns (e.g., which members are most sensitive to change) and avoiding repeated evaluation of inferior designs, which is critical given the high cost of SACS simulations.

Results. By applying ExLLM, we successfully automated the iterative design process for the offshore jacket structure. As shown in Table 8, our LLM-based approach discovered a design with a final weight of only 13.6 tons, a reduction of over 93% compared to the 218.0-ton human-designed baseline, while maintaining the maximum stress ratio within safe limits (<1.0). This result significantly outperforms traditional optimization algorithms like Genetic Algorithm (GA), MOEAD, and Random Search (RS) in terms of final structural weight. The convergence plots in Figure 6 further illustrate the efficiency of our method. The LLM-based optimizer (labeled LLM and LLM Ablation) demonstrates substantially faster convergence and achieves superior final values for both the Top 10 Fitness (F) and hypervolume metrics compared to other baselines. This indicates that the ExLLM framework not only finds better solutions but also does so with significantly fewer evaluation calls, a critical advantage for computationally expensive, real-world engineering problems.

Method	Final Total Weight (tons)	Maximum Stress Ratio
Human Baseline Design	218.00	0.024
LLM (ours)	**13.60**	0.508
LLM Ablation (ours)	14.90	0.814
GA (Genetic Algorithm)	32.24	0.093
MOEAD	37.76	0.137
RS (Random Search)	49.33	0.435

Table 8: Comparison of optimization results for the offshore jacket structure.

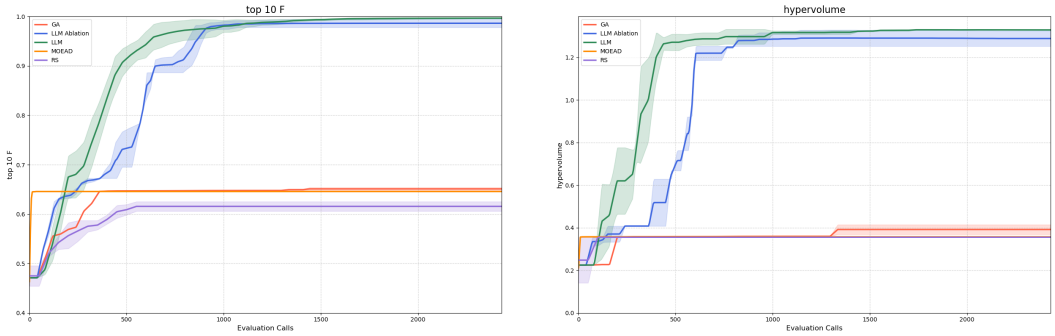

Figure 6: Performance comparison on the SACS benchmark. (Left) Top 10 F metric vs. evaluation calls. Our LLM-based methods show significantly faster convergence and achieve a higher final fitness score. (Right) Hypervolume evolution vs. evaluation calls. The LLM-based approaches achieve a substantially larger and more stable hypervolume, demonstrating superior performance in balancing the multi-objective trade-offs.

7.4.5 Peptide design for NK2R

Task and background. This task comes from the peptide design competition organized by Tsinghua University's FBS(Frontiers In Biological Structures) lab, aiming to discover anti-obesity therapeutics via the Neurokinin-2 receptor (NK2R)[1]. NK2R (UniProt P21452) is a promising target because activating it can reduce food intake via central mechanisms and increase energy expenditure peripherally; thus, efficacious NK2R agonists could provide dual-action weight-loss benefits Sass et al. (2024). It requires designing short peptide agonists that can outcompete the native ligand Neurokinin A (NKA) at NK2R under the competition's evaluation protocol (AlphaFold3 complex modeling with NK2R, Gα, NKA and designed peptide).

Challenge specification. Candidates must satisfy two hard success criteria: (i) sequence similarity to NKA (`HKTDSFVGLM`) must be $< 30\%$ by `mmseqs2`; (ii) under the same AlphaFold3 setup, the

[1] `https://www.fbs.frcbs.tsinghua.edu.cn/competition/2025Peptide`

peptide's complex ipTM with NK2R must exceed the ipTM of NKA (i.e., $\text{ipTM}_{\text{ours}} > \text{ipTM}_{\text{NKA}}$). Higher ipTM is better, and successful designs will proceed to wet-lab synthesis and activity testing in later rounds.

Challenges for our optimizer. (1) *Hard constraints dominate success*: without meeting the similarity/length filters and the "beat-NKA" ipTM criterion, candidates effectively score zero. (2) *Sparse, coupled feedback*: AlphaFold3 provides a small set of structural scores (e.g., ipTM), and the success condition is relative to NKA rather than an absolute threshold, coupling objectives and constraints tightly.

Our approach. We keep the ExLLM optimizer fixed (same hyperparameters as in the main text) and provide the task template and evaluation function using AlphaFold3. The task template compresses all rules (length, similarity, formatting) and objectives into a compact, structured description; the feedback adapter enforces filters and exposes per-iteration diagnostics. We optimize a two-term objective: maximize $\text{ipTM}_{\text{ours}}$ (peptide–NK2R complex) and minimize ipTM_{NKA} under the same NK2R target, subject to the length and `mmseqs2` similarity constraints. We start from 25 random UniProt peptides (≤ 40 aa); none beat NKA initially. Under a budget of 1,000 evaluations, ExLLM designs many candidates with positive margins.

Results. Table 9 lists representative top candidates (sorted by Δ); all reported sequences satisfy the competition's length and similarity filters in our screen. Notably, the best margin reaches $+0.67$ ipTM, and we routinely observe dozens of non-zero successes in a single run.

Peptide (ours)	ipTM(NKA)	ipTM(ours)	Δ
Peptide1	0.16	0.83	**0.67**
Peptide2	0.10	0.76	**0.66**
Peptide3	0.11	0.76	**0.65**
Peptide4	0.12	0.75	**0.63**
Peptide5	0.16	0.78	**0.62**
Peptide6	0.16	0.77	**0.61**
Peptide7	0.14	0.74	**0.60**
Peptide8	0.18	0.76	**0.58**
Peptide9	0.17	0.74	**0.57**
Peptide10	0.22	0.77	**0.55**

Table 9: NK2R results (AF3 ipTM). Candidates are ranked by margin $\Delta = \text{ipTM}_{\text{ours}} - \text{ipTM}_{\text{NKA}}$; larger is better. Budget: 1,000 evaluations.

7.4.6 GCU OPERATOR DESIGN

Task. This track targets high-performance kernel development on Tencent's GCU using the official operator SDK and `TopsCC` toolchain; submissions are auto-graded for both correctness and latency across 10 test cases per operator. The qualifier includes three operators (`Var`, `SiLU`, `GEMM v2`). Competition page: `https://aiarena.tencent.com/aiarena/zh/match/open-competition-2025`.

Challenges for our optimizer. (1) *Limited public expertise.* GCU is a new accelerator; kernels are built on a new framework, so CUDA-centric patterns from LLM pretraining transfer poorly. (2) *Low naïve compile success.* Direct LLM-generated kernels compile successfully in $< 10\%$ of attempts. (3) *Long iteration latency.* Each compile–run test takes 5–10 minutes, making failed trials especially costly and throttling exploration.

Our approach. We keep **ExLLM** fixed (same hyperparameters as in the main text) and adapt only the task template and evaluation harness. First, we distill the official SDK documentation into a few-hundred-word *prior* (key APIs, vectorization rules, memory/resource limits) and inject it into the prompt. Second, the feedback adapter returns compact, structured diagnostics from `TopsCC`—compiler errors, resource overuse (register/SMEM), numeric checks, and latency. These messages are surfaced verbatim to ExLLM and summarized into the evolving experience so that subsequent generations see parents' error traces plus accumulated general rules summarized by evolving experience.

Results. With only **300 evaluations per operator**, ExLLM-produced kernels pass all platform accuracy checks and substantially reduce latency for `Var`, `SiLU`, and `GEMM v2`. Our compile

success rate rises from $< 10\%$ (naïve prompting) to **85%** with error-aware feedback and evolving experience, enabling rapid iteration despite 5–10 minute test cycles. The submission ranked in the **top-10** of the qualifier and was close to the top score (exact rank withheld for anonymity). All submitted kernels were authored by ExLLM under the above template and harness, with **no human-written lines of code**—every kernel was fully model-generated without manual edits.

7.5 EXPERIENCE EXAMPLES

This section demonstrates experience from two tasks, the 5-objective optimization and single-objective optimization. The experience of 5-objective optimization task with random initilization is extracted from the experiments in Table 1 in the main content. And the single-objective task is optimizing JNK3 on PMO benchmark.

Initial experience of 5 objectives

1. Excellent Molecules:

- **Balanced Substituents:** Molecules with balanced, non-bulky substituents tend to have lower SA values.
- **Heterocycles & Aromatic Rings:** Incorporation of heterocyclic systems and aromatic rings contributes to favorable DRD2 and QED values.
- **Hydrophobic and Polar Groups:** Presence of hydrophobic aromatic systems along with polar functional groups like amides, ethers, and amines enhances GSK3β and JNK3 selectivity.
- **Stereochemistry:** Utilization of chiral centers often aids in achieving higher QED and specificity for GSK3β and JNK3.

2. Poor-Performing Molecules:

- **High SA Scores:** Often due to bulky, complex substituents and extensive branching.
- **Low QED Values:** Simplicity or lack of functional diversity can result in lower QED scores.
- **High DRD2 Values:** Overly hydrophobic or basic molecules tend to have higher DRD2 values, possibly leading to off-target effects.

Strategies to Optimize New Molecules

1. Decrease SA Value:

- Favor linear or moderately branched structures with controlled stereochemistry.
- Avoid excessive bulky groups and complex fused rings.

2. Decrease DRD2 Value:

- Integrate balanced hydrophobic and hydrophilic groups to avoid nonspecific binding.
- Use heterocycles to enhance specificity.

3. Increase QED Value:

- Aim for a balance in molecular weight, lipophilicity, and aromatic character.
- Incorporate functional groups that enhance drug-likeness, such as amides, esters, and ethers.

4. Decrease GSK3β Value:

- Select functional groups known for specific enzyme interactions, like amides and imides.
- Leverage computational tools to tailor interactions for GSK3β.

5. Increase JNK3 Value:

- Incorporate chiral centers to improve selectivity.
- Include moieties known for JNK3 interactions, such as specific aromatic or heterocyclic systems.

Avoiding Suboptimal Properties

- **Reduce Molecular Complexity:** Avoid overly complex molecules with high SA values.
- **Enhance Functional Diversity:** Ensure a good mix of polar and non-polar groups to avoid low QED.
- **Modulate Hydrophobicity:** Avoid excessive hydrophobicity, which increases DRD2 values and off-target effects.

Final experience of 5 objectives

- **Aromatic Cores:** Utilize benzene and thiophene rings for stability and enhanced bioactivity.
- **Functional Groups:** Prefer amides, carbamates, esters, and ethers to enhance QED and JNK3 while reducing GSK3β.
- **Halogen Substitution:** Introduce halogens, especially monosubstitution in aromatic rings, to improve SA and bioactivity.
- **Structural Simplicity:** Favor simpler, smaller structures to achieve lower SA values.
- **Selective Substitution:** Utilize monosubstitution in aromatic rings to balance low SA and high bioactivity.
- **Bioactivity Optimization:** Enhance QED and JNK3 values while minimizing DRD2 and GSK3β values.
- **Avoid Bulky Groups:** Minimize bulky groups to maintain simplicity, lower SA, and sustained bioactivity.
- **Functional Integration:** Combine hydrophobic and polar groups strategically to optimize bioactivity and maintain low SA.
- **Linear and Compact Structures:** Avoid complex branching; favor linear and compact molecules to minimize DRD2 and GSK3β.

Initial experience of JNK3

Summary of Molecular Optimization Insights
1. Characteristics of High-Performing Molecules

- The presence of aromatic rings and heterocycles is prevalent.
- Functional groups such as amines, ethers, and sulfoxides are common.
- Chiral centers and stereochemistry play a significant role.
- Substitution on aromatic rings with electron-withdrawing groups like chlorine and fluorine enhances performance.
- Alkyl side chains and central amine linkages contribute to activity.

2. Strategies for Designing New Molecules

- Incorporate aromatic systems and heterocyclic structures to increase stability and specificity.
- Introduce functional groups like amines, ethers, and sulfoxides to enhance binding interactions.
- Leverage chiral centers and stereochemistry to improve efficacy and selectivity.
- Utilize electron-withdrawing groups for aromatic ring substitutions to enhance activity.
- Ensure balanced lipophilicity and solubility through careful side chain selection.

3. Reasons for Poor Performance in Low-Scoring Molecules

- Lack of aromatic or heterocyclic components reduces stability and binding efficacy.
- Absence of key functional groups like amines and ethers diminishes interaction potential.
- Insufficient stereochemistry and chirality lead to lower specificity and activity.
- Overly simple molecular structures lack necessary complexity and interaction sites.

4. Avoidance of Suboptimal Properties

- Prioritize the inclusion of aromatic systems and heterocyclic compounds.
- Add diverse functional groups to create better binding and interaction profiles.
- Design molecules with defined stereochemistry to enhance specificity.
- Maintain a balance of molecular complexity to ensure both efficacy and manageable synthesis.

Following these insights can guide the design of new molecules with enhanced JNK3 values and overall better performance. You can take advantage of these experiences to propose better molecules aligned with the optimization objectives.

Final experience of JNK3

Integrated Experience for Molecular Optimization
High-Performing Molecule Characteristics

- **Key Functional Groups:** Sulfonamide, sulfonyl, N-substituents (amines, alcohols).
- **Structural Features:** Aromatic and heterocyclic rings with flexible or cyclic linkers.
- **Hydrophobic/Hydrophilic Balance:** Achieved through diverse functional groups.
- **Electron-Withdrawing Groups:** Nitrogen-based groups on aromatic or heterocyclic rings.

Design Strategies

1. **Functional Groups:** Incorporate sulfonamide or sulfonyl moieties to enhance binding affinity and solubility.
2. **N-Substituents:** Employ functionalized side chains to improve molecular flexibility and target specificity.
3. **Electron-Withdrawing Groups:** Optimize electron interactions to strengthen binding affinity.
4. **Structural Flexibility:** Utilize cyclic and flexible linkers to promote favorable binding dynamics.

Reasons for Poor Performance in Low-Scoring Molecules

- **Lack of Key Groups:** Absence of critical functional groups reduces binding capacity and solubility.
- **Steric Hindrance:** Presence of bulky substituents hinders effective binding.
- **Suboptimal Electron Density:** Insufficient electron interactions weaken molecular efficacy.
- **Poor Functionalization:** Ineffective ring substitutions compromise performance.

Actionable Insights

- Incorporate diverse N-substituents and electron-withdrawing groups.
- Maintain an appropriate hydrophobic/hydrophilic balance.
- Avoid excessive steric hindrance in functional group placement.
- Design flexible or cyclic linkers to enhance dynamic binding interactions.

These insights can guide the development of improved molecules that better satisfy multi-objective optimization criteria.

7.6 Different number of objectives

	1 Objective		2 Objectives		3 Objectives		4 Objectives		5 Objectives		6 Objectives	
Metric	ExLLM	MOLLEO	ExLLM	MOLLEO	ExLLM	MOLLEO	ExLLM	MOLLEO	ExLLM	MOLLEO	ExLLM	MOLLEO
Top1 F	**0.948**	0.941	**1.901**	1.887	**2.901**	2.891	**3.901**	3.890	**4.413**	4.098	**5.183**	4.964
Top10 F	**0.948**	0.936	**1.901**	1.882	**2.901**	2.886	**3.901**	3.887	**4.300**	4.076	**5.164**	4.948
Uniqueness	**0.929**	0.150	**0.666**	0.231	**0.778**	0.273	**0.807**	0.387	**0.872**	0.575	**0.957**	0.591
Validity	**0.796**	0.159	**0.962**	0.552	**0.946**	0.803	**0.946**	0.783	0.908	**0.938**	0.890	**0.926**
Diversity	0.538	**0.865**	0.450	**0.646**	0.510	**0.627**	0.375	**0.614**	0.441	**0.573**	0.529	**0.611**

Table 10: Unconstrained molecular design results with 1 to 6 objectives. The sixth objective is BBBP.

7.7 Efficiency comparison

Apart from that, without early stopping, ExLLM only uses nearly $\frac{1}{3}$ LLM calls compared to MOLLEO, more than even 14x faster than MOLLEO in run time to achieve significantly better results, as shown in Table 11. The running time is also competitive to other methods that are not based on LLMs.

7.8 Hyperparameters

To validate the effectiveness of the key components in ExLLM, we conduct a series of ablation studies. In ExLLM, Pareto front selection and F-value selection are applied with equal probability in each

Method	ExLLM (GPT-4o)	MOLLEO (GPT-4o)	Graph GA	Gp-BO	Genetic-GFN	MARS	JT-VAE	DyMol	REINVENT
Running time(hours)	0.52	7.32	1.04	0.68	0.45	0.69	3.08	0.28	0.52
LLM calls	2908 ± 133	8517 ± 375	-	-	-	-	-	-	-
LLM costs (USD)	9.89 ± 1.13	44.78 ± 4.43	-	-	-	-	-	-	-

Table 11: Running time of MOLLEO and ExLLM

Method	Top1 F	Top10 F	Uniqueness	Validity	Diversity
Without MO Selection	3.830	3.791	0.999	0.816	0.842
With MO Selection	**4.187**	**4.152**	0.961	**0.915**	0.556

Table 12: Experiments of using MO.

iteration. The importance of this design is demonstrated in Table 12, where performance significantly deteriorates when multi-objective selection is removed. Furthermore, if an objective is included in the prompt but is not explicitly considered in MO selection, the performance of ExLLM declines substantially. This highlights the critical role of MO selection in ensuring effective optimization across multiple objectives.

7.9 Number of output molecules

Method (GPT-4o direct propose)	Top1 F	Top10 F	Uniqueness	Validity	Diversity
1 offspring	0.944	0.936	0.667	0.971	0.829
2 offsprings	0.947	0.935	**0.762**	0.975	0.826
3 offsprings	0.945	0.931	0.719	0.979	0.817
4 offsprings	0.947	0.937	0.740	0.978	0.834

Table 13: Experiments of different number of output molecules directly proposed by LLMs. The objective is maximizing QED.

We begin by using an LLM to directly generate 1,500 molecules for the task of maximizing QED, as shown in Table 14. As the number of offspring (i.e., output molecules per query) increases, the top F-values, validity, and diversity remain relatively stable. However, uniqueness improves substantially when more than one molecule is generated per query. This is attributed to the autoregressive nature of LLMs, where later outputs are conditioned on earlier ones within the same generation, implicitly encouraging exploration.

We further extend this experiment within the full ExLLM framework, as presented in Table **??**, and identify the optimal number of offspring per query to be two. Accordingly, our implementation configures the LLM to generate two offspring per call for both crossover and mutation operations. This design significantly reduces the number of required LLM queries while achieving superior performance compared to generating one offspring per call (as used by MOLLEO) or three per call. In addition, when compared with directly generating 5,000 molecules using GPT-4o, ExLLM delivers a notable improvement in optimization quality, demonstrating the effectiveness of our framework.

7.10 Visualization of molecules

The number under each molecule is the F value of it.

Method	Top10 F	Top10 AUC	Uniqueness	Validity
ExLLM (Gemini)				
1 offspring	0.422 ± 0.028	0.300 ± 0.004	0.569 ± 0.060	0.976 ± 0.006
2 offsprings	**0.742 ± 0.007**	**0.585 ± 0.004**	**0.731 ± 0.048**	0.961 ± 0.024
3 offsprings	0.587 ± 0.052	0.486 ± 0.027	0.649 ± 0.056	0.974 ± 0.006
4 offsprings	0.650 ± 0.099	0.508 ± 0.065	0.683 ± 0.026	0.982 ± 0.006
5 offsprings	0.604 ± 0.012	0.528 ± 0.015	0.586 ± 0.106	0.986 ± 0.009
6 offsprings	0.657 ± 0.094	0.492 ± 0.082	0.638 ± 0.031	0.972 ± 0.002
ExLLM (GPT-4o)				
1 offsprings	0.861 ± 0.044	0.644 ± 0.025	0.895 ± 0.004	0.822 ± 0.035
2 offsprings	0.877 ± 0.043	0.698 ± 0.038	0.927 ± 0.018	0.835 ± 0.030

Table 14: Experiments of k-offspring per prompt. The objective is maximizing JNK3.

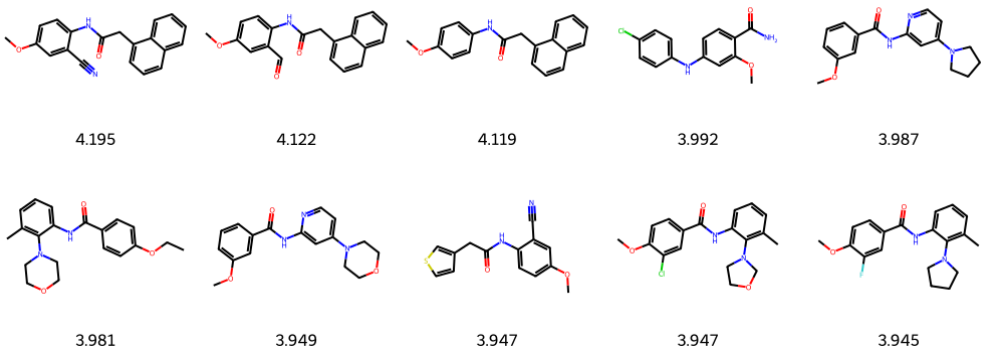

Figure 7: MOLLEO

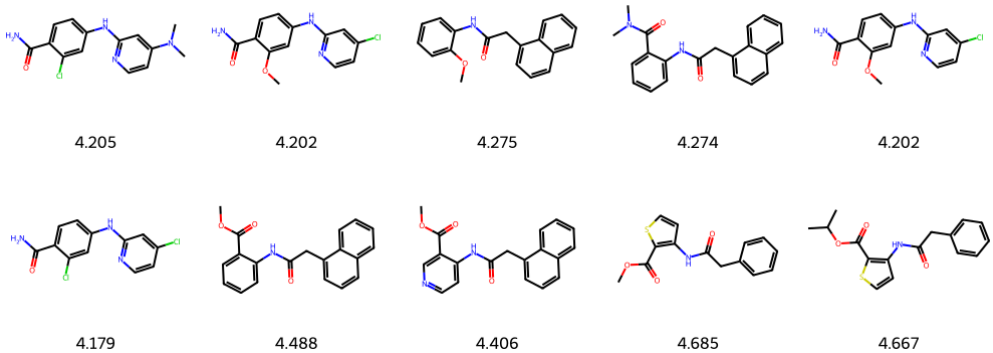

Figure 8: ExLLM (ours)

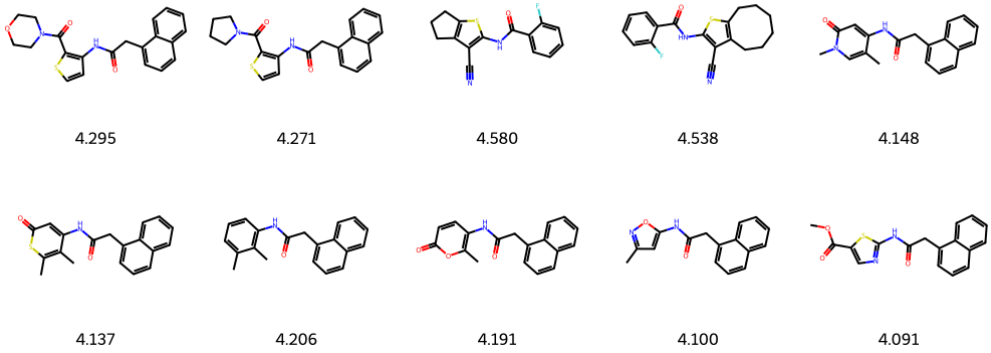

Figure 9: DyMol

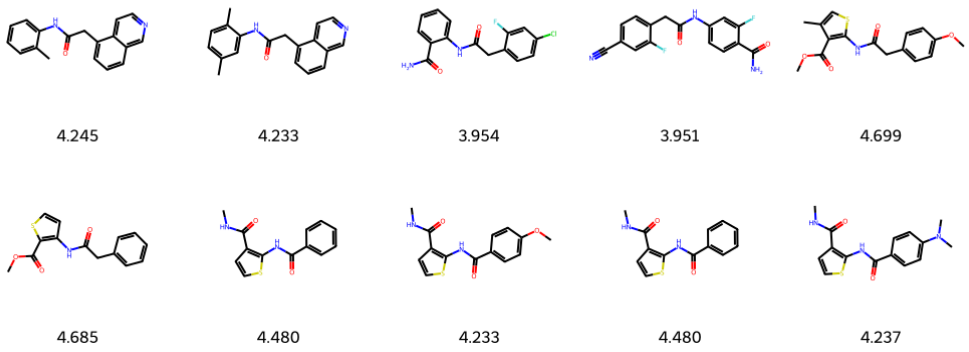

Figure 10: Genetic-GFN

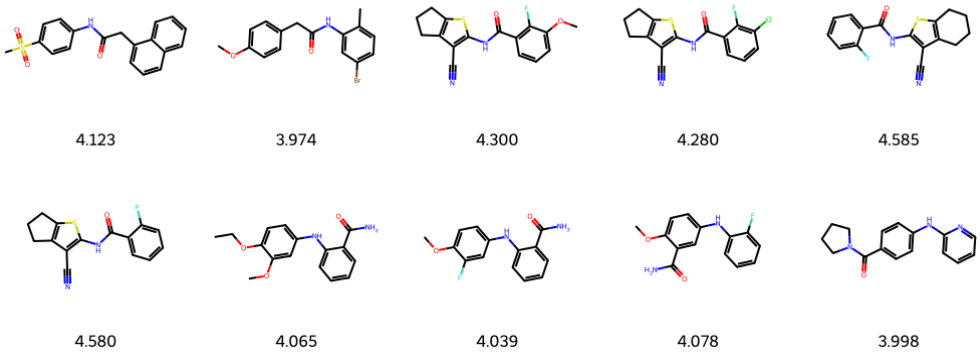

Figure 11: REINVENT

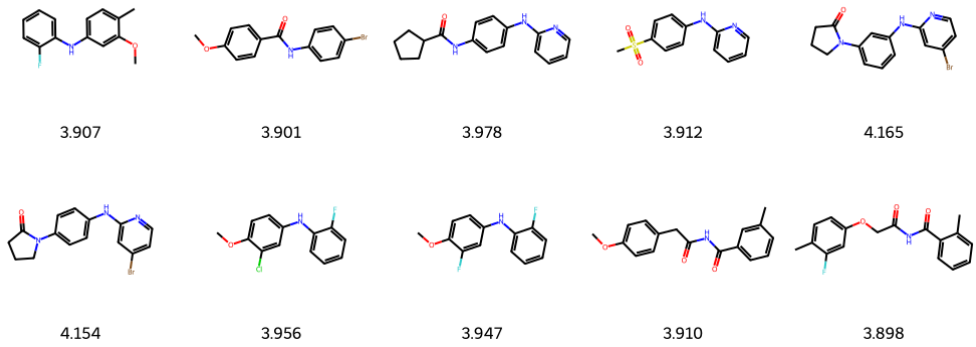

Figure 12: GB-GA

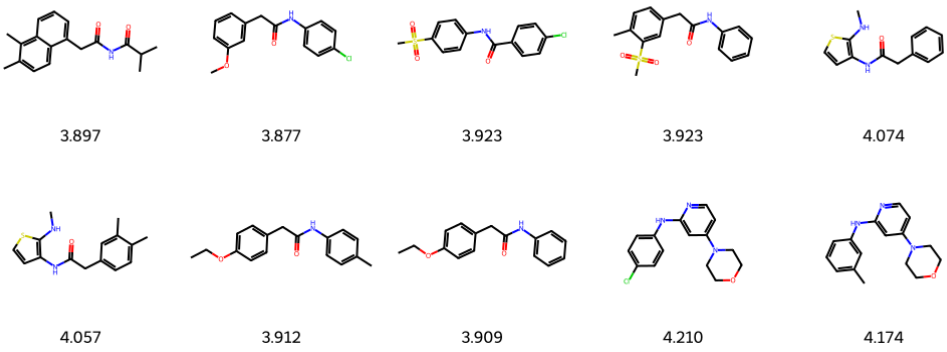

Figure 13: GB-BO

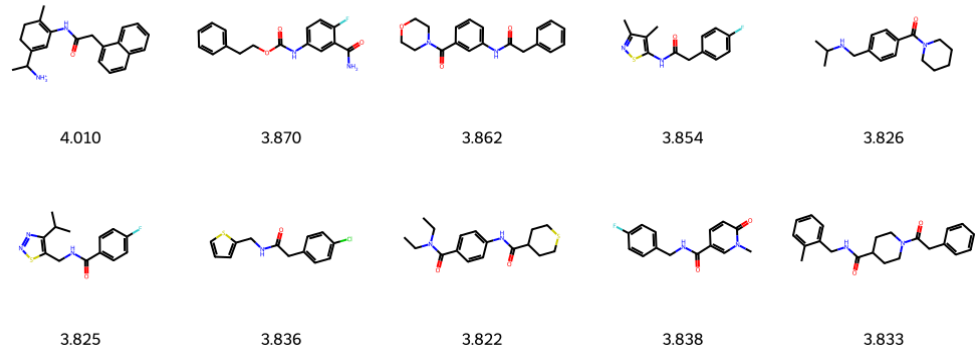

Figure 14: JT-VAE

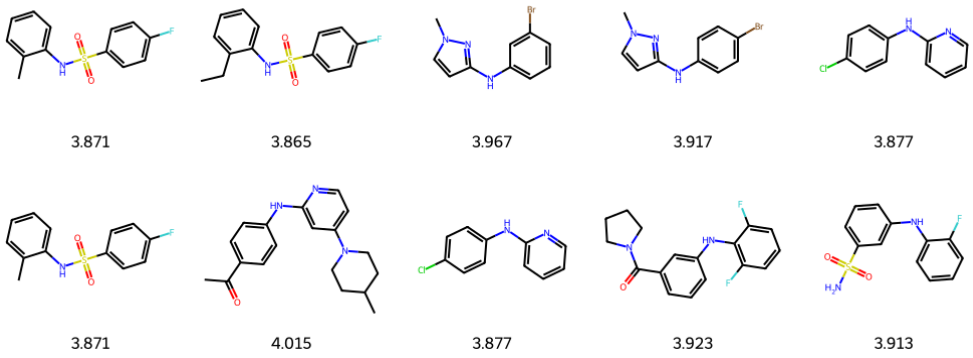

Figure 15: MARS