M^{4}olGen: Multi-Agent, Multi-Stage Molecular Generation under Precise Multi-Property Constraints

Anonymous ACL submission

Abstract

Generating molecules that satisfy precise numeric constraints over multiple physicochemical properties is critical and challenging. Although large language models (LLMs) are expressive, they struggle with precise multi-objective control and numeric reasoning without external structure and feedback. We introduce **M^{4}olGen**, a fragment-level, retrieval-augmented, two-stage framework for molecule generation under multi-property constraints. **Stage I: Prototype generation**: a multi-agent reasoner performs retrieval-anchored, fragment-level edits to produce a candidate near the feasible region. **Stage II: RL-based fine-grained optimization**: a fragment-level optimizer trained with Group Relative Policy Optimization (GRPO) applies one- or multi-hop refinements to explicitly minimize the property errors toward our target while regulating edit complexity and deviation from the prototype. A large, automatically curated dataset with reasoning chains of fragment edits and measured property deltas underpins both stages, enabling deterministic, reproducible supervision and controllable multi-hop reasoning. Unlike prior work, our framework better reasons about molecules by leveraging fragments and supports controllable refinement toward numeric targets. Experiments on generation under two sets of property constraints (QED, LogP, Molecular Weight and HOMO, LUMO) show consistent gains in validity and precise satisfaction of multi-property targets, outperforming strong LLMs and graph-based algorithms.

1 Introduction

Generating molecules that satisfy precise numeric constraints is a fundamental and critical task in scientific discovery, with applications in drug development, materials design, de novo design and molecular property optimization (Sanchez-Lengeling and Aspuru-Guzik, 2018; Fromer and Coley, 2023). Optimizing compounds to meet numeric multi-property targets improves real development outcomes with desired attributes (Wager et al., 2016; M. Bran et al., 2024). Much of the molecular generation literature treats molecular discovery as maximizing one or a few surrogate properties, rather than matching user-specified numerical targets; approaches that offer precise, simultaneous control over multiple properties remain scarce.

While numerous application-specific target properties can be used in practice (Zeni et al., 2025; Ding et al., 2024), for our study we pick fundamental properties often used in prior work (Loeffler et al., 2024; Brown et al., 2019; Jain et al., 2023; Cai et al., 2024). Specifically, we begin our study with simple properties, namely drug-likeness (QED), lipophilicity (logP), and molecular weight (MW) that shape permeability, exposure, and overall developability (Bickerton et al., 2012; Giaginis et al., 2018). While these are simplified surrogates, they are (i) fast and reproducible to evaluate (enabling large-scale training and ablations), (ii) continuous and numeric, which is essential for testing precise multi-objective control, and (iii) standardized across open benchmarks, supporting fair comparison. We then conduct more challenging experiments on energy properties, HOMO and LUMO (Brédas, 2017; Fukui et al., 1954), critical in diverse applications (Kim et al., 2013). Our goal is to validate a multi-agent, numerically conditioned generation framework under multiple verifiable, compute-efficient proxies. As we show with HOMO-LUMO experiments, our approach can be used in richer oracles as we scale to more realistic discovery settings.

To fine-tune the optimizer, we construct a large dataset of more than 2 million molecules decomposed into BRICS (Degen et al., 2008) fragments along with their corresponding properties. From this dataset, we derive a neighbor relational dataset of 1.17 million pairs for controllable reasoning automatically. Each molecule in this dataset is paired

with an explicit one-hop neighbor list: molecules that differ by exactly one fragment (add, remove, or replace) and that pass the RDKit validity and edit sanity checks. By chaining these one-hop moves, we gradually grow neighbor forests from any starting molecule. These structures enable long and controllable reasoning chains. Unlike prior LLM-agent work that maximizes a property toward extremes, we target precise numeric matching of user-specified multi-property values. This is essential for inverse design but underexplored in the current LLM-for-molecule literature. We demonstrate that this architecture markedly improves adherence to numeric multi-property constraints and surpasses prior LLM-based methods by large margins.

In summary, we contribute (i) M^{4}olGen, a molecular generation framework that couples retrieval-augmented prototyping with GRPO-based fragment-level optimization to achieve *exact numeric control* over multiple properties; (ii) a scalable *multi-hop* refinement mechanism that boosts output quality while explicitly regulating edit complexity and deviation from the starting structure; (iii) a public dataset of $\sim$2.95M molecules with BRICS fragment annotations and a neighbor set of $\sim$1.17M single-edit pairs that enable fragment-level learning and controllable reasoning; and (iv) comprehensive experiments and ablations demonstrating strong results compared to baselines and clear additive gains from each component.

2 Related Work

Molecular Generation with Property Control. Deep generative models have been widely applied to molecular design using graph or sequence-based representations such as SMILES, including VAEs (Gómez-Bombarelli et al., 2016), GANs like MolGAN (Cao and Kipf, 2018), an graph-based models such as GCPN (You et al., 2018), GraphAF (Shi et al., 2020), and MoFlow (Zang and Wang, 2020). Spanning Tree Graph Generation (STGG) (Ahn et al., 2021; Jolicoeur-Martineau et al., 2025) achieves strong multi-objective performance by combining sequential modeling with tree-based representations. Reinforcement learning approaches, such as MolDQN (Zhou et al., 2018) and GFlowNets (Bengio et al., 2023; Jain et al., 2023), enable property-driven optimization, often extended to QED, LogP, and SA. However, these single-agent methods struggle to exactly satisfy multiple numeric constraints, reflecting exploration–exploitation trade-offs.

LLMs for Molecular Design and Reasoning. Large language models such as ChemGPT (Frey et al., 2023), ChemBERTa (Chithrananda et al., 2020), MolT5 (Edwards et al., 2022), ChemFM (Cai et al., 2024), and Chemformer (Irwin et al., 2021) capture chemical syntax and semantics for general-purpose molecular generation, but remain limited in precise numerical reasoning and property control. Chain-of-Thought prompting (Wei et al., 2022) improves multi-step reasoning in LLMs, and analogous strategies have been proposed for molecules (Jin et al., 2024; Jang et al., 2024; Zheng et al., 2024), yet exact satisfaction of multiple physicochemical constraints remains challenging. Instruction Multi-Constraint Molecular Generation (Zhou et al., 2025) shows that LLMs can satisfy multiple property constraints through teacher–student supervision and interval-based conditioning, but operates within bounded ranges and lacks reinforcement learning for multi-objective optimization, limiting exploration.

Multi-Agent Planning and Reasoning in Molecule Design. Agent-based systems have long been studied in robotics, distributed AI, and resource allocation (Wooldridge, 2009; Weiss, 1999), but most AI-driven molecule design approaches remain single-agent, with one generative model guided by property predictors. Recent multi-agent systems decompose the design process into specialized roles such as generation, property evaluation, and refinement, improving exploration efficiency and controllability through cooperation and hierarchical coordination. Examples include Prompt-to-Pill (Vichentijevikj et al., 2025), ROBIN (Ghareeb et al., 2025), DrugAgent (Liu et al., 2024), Honeycomb (Zhang et al., 2024) and ChemCrow (M. Bran et al., 2024). MoLLEO (Wang et al., 2024) integrates LLMs into evolutionary search over chemical space, and MT-Mol (Kim et al., 2025) introduces a tool-augmented multi-agent system for molecular optimization. The methods target property maximization (e.g., maximizing QED) rather than precise user-specified numerical targets that vary across queries (e.g., QED = 0.72, LogP = -1.8, MW = 310), making direct comparison difficult since success criteria differ: improvement against a baseline versus accuracy against exact targets. We view these directions as complementary, as optimization toward favorable regions is recoverable as a special

case of conditional generation within a desired range. Building on this line of work, we introduce a retrieval-augmented multi-agent reasoner that iteratively constructs locally optimal prototypes before refinement, combining in-distribution retrieval with domain knowledge to improve controllability under numeric constraints.

Policy Optimization for Multi-Property Objectives. Reinforcement learning underpins molecular optimization, with policy-gradient methods like REINFORCE (Williams, 2004) and PPO (Schulman et al., 2017) adapted to molecule design and MolDQN (Zhou et al., 2018) applying Q-learning to multi-objective settings. These approaches struggle to balance multiple numeric objectives precisely. Group Relative Policy Optimization (GRPO) (Shao et al., 2024; Zhang et al., 2025), originally introduced for preference-based learning and RLHF, optimizes policies via group-relative advantages that reward candidates outperforming their peers. While GRPO and its variants are well known for strengthening LLM reasoning (DeepSeek-AI et al., 2025), we are the first to adapt it to numerically conditioned generation, integrating fragment-level refinement and controllable multi-hop optimization into the generation loop to yield a principled RL framework for satisfying numeric multi-property targets.

3 Methodology

We propose M^{4}olGen (Figure 1) a multi-stage, goal-conditioned framework for constrained molecular generation that casts numeric targets (QED, LogP, MW, HOMO, LUMO) as a verifiable distance-to-target objective over an actionable fragment-edit space. In the following methodology part, we will describe our system using the first set of properties (QED, LogP, MW) as an example for clearer presentation. Stage I performs retrieval-augmented prototyping: a local reasoner edits fragments using in-distribution exemplars and RDKit feedback to place a candidate near the feasible region. Stage II applies a GRPO-trained fragment-level optimizer in a multi-hop manner to minimize the distance-to-target while regulating edit complexity and deviation from the starting structure. Trained on a large, property-annotated neighbor dataset, M^{4}olGen generalizes across target tuples and delivers precise, simultaneous control of set 1 (QED, LogP, MW) and set 2 (HOMO, LUMO).

3.1 Stage I: Prototype Generation with Retrieve-augmented Multi-agent Reasoning

The objective of Stage I is to generate a chemically valid prototype m_{local} that serves as a high-quality starting point for numeric optimization. This is accomplished via a collaborative multi-agent framework that decomposes the input query, retrieves similar molecules from a large database, and incrementally proposes fragment-level edits based on domain knowledge.

Query Interpretation. Given a natural-language request q (e.g., *"Generate a molecule with QED*=0.75*, LogP*=2.7*, MW*=310*"*), this module extracts the exact numeric targets for each property and returns a target property vector

$$\mathbf{p}_{\text{tgt}} = \big(p_{\text{QED}},\, p_{\text{LogP}},\, p_{\text{MW}}\big), \quad p_{\text{QED}} \in [0,1],\ p_{\text{LogP}} \in \mathbb{R},\ p_{\text{MW}} > 0. \tag{1}$$

We use $\mathbf{p}$ *for "properties"* and the subscript "tgt" to denote targets. A rule-based parser identifies numeric constraints and synonyms.

Reference Retrieval. Given the target property vector $\mathbf{p}_{\text{tgt}}$, we query a large annotated molecule corpus Ω to obtain a set of *reference molecules* that lie close to the targets under per–property tolerances:

$$\mathcal{M} = \big\{\, m \in \Omega \ :\ |\, p_i(m) - p_{i,\text{tgt}} \,| \le \epsilon_i \quad \forall\, i \in \{\text{QED}, \text{LogP}, \text{MW}\} \big\}. \tag{2}$$

Here $p_i(m)$ denotes the i-th property of molecule m (computed via RDKit), and ϵ_i are small, property-specific tolerant ranges, e.g., ± 0.05 for QED (0–1 scale), ± 0.5 for LogP (small medicinally meaningful shift), and ± 25 Da for MW. They are chosen to be tight enough to keep the references in-distribution yet broad enough to ensure sufficient references. The retrieved references are then used to *anchor* Stage I: they provide in-distribution exemplars that guide fragment-level edits, constrain the search toward the feasible region, and seed candidate/neighbor structures consumed by the multi-hop optimizer in Stage II.

Prototype Reasoner. This LLM-driven module proposes stepwise, fragment-level edits to turn an initial seed (either "start from scratch" or molecules sampled from the reference set $\mathcal{M}$) into a high-quality *prototype* close to the target. At iteration t, the reasoner selects an action $a_t \in$

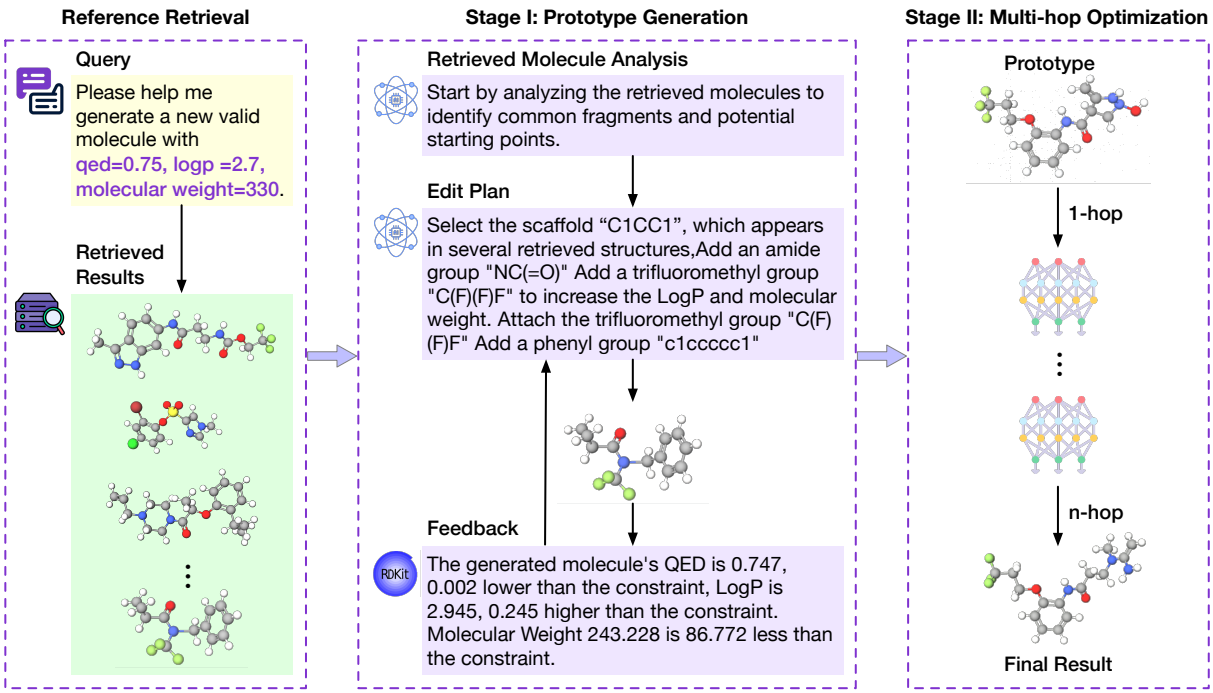

Figure 1: The flow chart of M^{4}olGen. The first two blocks involve **Retrieval and Prototyping**, where molecular candidates are first retrieved based on the given constraints (QED, LogP, MW) and then analyzed by a local reasoner to extract constraints, analyze retrieved molecules, and propose an editing plan based on evaluator's feedback to generate prototypes iteratively. The third block describes **Multi-Hop Optimization**, where the prototypes are optimized through one-hop and n-hop controllable editing steps by the molecule optimizer trained by GRPO.

$\{\texttt{replace}, \texttt{add}, \texttt{remove}\}$ and applies it to obtain a new intermediate molecule along with the previous trajectory:

$$m_t \;=\; \mathrm{Edit}(m_{t-1};\, a_t)\,, \qquad m_t \in \mathcal{M}_{\text{valid}}, \quad (3)$$

where $\mathcal{M}_{\text{valid}}$ denotes RDKit-parseable structures that pass basic valence and sanity checks. Decisions are guided by three information sources: (i) *reference molecules* $\mathcal{M}$ retrieved near the target, (ii) an *experience pool* of prior edits (neighbor pairs/trees) summarizing successful local transformations, and (iii) *property feedback* (QED/LogP/MW) computed by RDKit on every candidate. The reasoner stops early when the distance-to-target falls below a threshold τ or when a maximum number of steps $T_{\max}$ is reached. The final result of this process is denoted as m_{local}.

Validity and Error Estimation. Given the current prototype m_{local}, we compute per–property deviations from the targets:

$$\Delta_{\text{i}}(m) = \big|\, p_i(m_{\text{local}}) - p_{i,\text{tgt}} \,\big|, \qquad i \in \{\text{QED, LogP, MW}\}, \quad (4)$$

and aggregate them into a distance-to-target objective $E(m) = \sum_i w_i\, \Delta_i(m)$ with property-specific weights. These errors are fed into the Stage II optimizer prompt to enable targeted refinement.

Stage I Objective. Formally, Stage I seeks a valid prototype along the reasoning trajectory $\mathcal{G} = \{m_0, \ldots, m_T\}$ that minimizes the distance-to-target:

$$m_{\text{local}} = \arg \min_{m \in \mathcal{G} \cap \mathcal{M}_{\text{valid}}} \sum_{i \in \{\text{QED, LogP, MW}\}} w_i \,\big|\, p_i(m) - p_{i,\text{tgt}} \,\big|. \quad (5)$$

The algorithm is stated in Appendix A.2. This stage reliably moves the candidate into the feasible region by leveraging relevant molecules, past experience, and tool feedback. However, a multi-agent reasoner that is not further trained has a performance limitation on fine-grained, precise multi-property control. Stage II addresses this by applying a GRPO-trained, fragment-level optimizer in a controlled multi-hop fashion to further reduce the total error $E(m)$ while regulating edit complexity and deviation from the starting structure.

3.2 Stage II: Fragment-Level Optimization via GRPO (Multi-Hop Extension)

While Stage I reliably moves a candidate toward the feasible region, precise control of multiple numeric properties (e.g., QED, LogP, MW) remains difficult for text-only planning because LLMs have

difficulty dealing with numeric-related design and lack a mechanism to explicitly minimize the distance to target values. Our insight is to treat refinement as an optimization problem over an actionable fragment-edit space with fast, verifiable feedback from chemistry oracles. We therefore train an optimization policy with **GRPO** (Group Relative Policy Optimization) (DeepSeek-AI et al., 2025) because its group-wise, rank-based updates are stable and sample-efficient without ground-truth demonstrations, and because it can directly optimize a reward that faithfully encodes the numeric targets. RDKit oracles and policy models provide the property feedback at each step.

Fragmentization and Action Space. Let $m_0 := m_{\text{local}}$ be the prototype from Stage I. We decompose molecules into chemically meaningful building blocks using **BRICS** (Break Retrosynthetically Interesting Chemical Substructures) (Degen et al., 2008), a rule-based scheme that cuts retrosynthetically plausible bonds formed or broken during synthetic processes, leading to fragments that are synthetically accessible and chemically meaningful. This yields fragments that support localized edits, preserve validity, and keep the search space tractable where $\Phi(m)$ is the fragment set for molecule m and f are the fragments:

$$\Phi(m) \;=\; \{\, f_1, \ldots, f_k \,\}. \tag{6}$$

At hop $h \in \{1, \ldots, H\}$, the optimizer selects one fragment-level action $a_h \in \{\texttt{add}, \texttt{remove}, \texttt{replace}\}$ and applies it to obtain a new candidate

$$m_h \;=\; \text{Edit}(m_{h-1}; a_h), \qquad m_h \in \mathcal{M}_{\text{valid}}, \tag{7}$$

where $\mathcal{M}_{\text{valid}}$ denotes RDKit-parseable structures that pass basic valence and sanity checks. A hop budget H controls structural complexity and deviation from the starting structure.

Optimizer and Input Representation. Our optimizer $\mathcal{O}_\phi$ is a sequence model (an LLM policy) fine-tuned with GRPO on our neighbor-pair corpus of single-fragment edits. Following (Guevorguian et al., 2024), we extend the tokenizer with `<SMILES>`, `</SMILES>`, `<QED>`, `</QED>`, `<LogP>`, `</LogP>`, `<MW>`, `</MW>` so that molecules and targets are explicit in the prompt. At each hop, the policy conditions on $(m_{h-1}, \Phi(m_{h-1}), \mathbf{p}_{\text{tgt}})$ and proposes one edited molecule; after H hops we return $m^* := m_H$.

Reward and GRPO Objective. We define a distance-to-target objective and convert it to a scalar reward using fast RDKit oracles:

$$E(m) \;=\; \sum_{i \in \{\text{QED,LogP,MW}\}} w_i \,\big|\, p_i(m) - p_{i,\text{tgt}} \,\big|, \qquad r_{\text{prop}}(m) \;=\; 1 - E(m). \tag{8}$$

The full reward combines format, property, diversity, and validity terms:

$$R(m) \;=\; \underbrace{r_{\text{format}}(m)}_{\text{valid SMILES / instruction}} \;+\; \underbrace{r_{\text{prop}}(m)}_{\text{scaled property match}} \;-\; \underbrace{r_{\text{repeat}}(m)}_{\text{repetition penalty}} \;-\; \underbrace{r_{\text{invalid}}(m)}_{\text{RDKit parse / valence penalty}}. \tag{9}$$

Here w_i are *weights* that balance units and priorities. We also optionally regularizes complexity (e.g., hop count or similarity). GRPO samples a group of candidates, ranks them by $R(m)$, get the normalized rewards from reward functions, and updates the policy to increase the likelihood of higher-ranked edits while discouraging weaker ones.

Multi-hop Refinement and Control. Applying the optimizer in a controlled multi-hop manner enables gradual, interpretable refinement: small, local edits accumulate to tighten requirement satisfaction, while the hop budget and regularizers bound complexity and deviation from the prototype. In practice, a modest H suffices to reliably reduce $E(m)$ thanks to fragment locality and fast RDKit evaluation, and the same mechanism supports adaptive planning and curriculum-style difficulty scaling during training and evaluation. Training uses our automatically curated neighbor-pair corpus coupling each molecule with properties, fragment decomposition, and single-edit neighbor relations (Appendix A.1).

4 Experiment

We conduct the experiments for the following three claims. **(C1) Precise multi-property control:** we benchmark M^{4}olGen against strong LLMs and graph methods under identical compute budgets, reporting per-property MAE and a normalized total error to demonstrate simultaneous control of QED/LogP/MW and HOMO/LUMO. **(C2) Necessity and effectiveness of the two-stage design:** we perform ablations that toggle retrieval in Stage I and vary the GRPO optimizer hops (1/2/3), to show that retrieval-augmented prototyping plus multi-hop refinement is required for tight numeric align-

ment. **(C3) Generalization without per-target retraining & controllable edit complexity:** we uniformly sample 100 target tuples across admissible ranges, run 10 trials per tuple/baseline (best-of-10 under a fixed budget), and analyze performance as a function of hop budget, establishing broad generalization and explicit control of deviation from the prototype.

4.1 Experimental Setup

Training Details. In Stage I, we employ GPT-4o (OpenAI, 2024a) or Qwen3-14B-chem-dyn-tokenizer (Summykai, 2024) as the prototype-reasoning LLM. For the Stage II training, we select ChemDFM-v1.5-8B (Zhao et al., 2025) as the base model, which achieves overall great performance among chemical generation tasks among models within 8B. We first train ChemDFM-v1.5-8B for 5000 steps with supervised fine-tuning to strength the output format as cold start. This can accelerate the convergence speed for the following GRPO training since the reward function can get effective feedback sooner than randomly exploring the format first. Then the model is trained for 37,500 steps with GRPO. The scalars we choose to normalize errors for the reward function are α_q=1, α_l=6, α_w=100, as we consider error values 1 in QED, 6 in LogP and 100 in MW as the maximum thresholds. There is no scalar for HOMO and LUMO. We directly use the MAE in the reward function. These scalars are flexible to tune depending on personal usage. The invalidity penalty and wrong format penalty are both -10 while the repetition penalty is accumulated by 0.1 for each time. At each step, we sample 8 candidates using stochastic decoding (temperature $T = 1.0$, top-$p = 0.9$, top-$k = 50$). The model was trained to convergence on a single NVIDIA A100 (40 GB).

Baselines. We aim to investigate the power of LLMs for generating new molecules under precise constraints. Thus, most of the baselines we choose are LLMs. In the LLM-based solutions, we have gpt-4.1 (OpenAI, 2025), Gemini-Flash (Google, 2025), claude-haiku (Anthropic, 2024), gpt-4o-2024-05-13(latest version) (OpenAI, 2024b), SmileyLlama-8B (Cavanagh et al., 2025) and DrugAssist-7B (Ye et al., 2023). They cover most commonly used commercial models and generation-oriented fine-tuned chemical LLMs including the SFT (Supervised Fine-Tuning) and DPO (Direct Preference Optimization) techniques. In addition to LLM baselines, we also compare to commonly used graph-based and hybrid algorithms. STGG+ (Jolicoeur-Martineau et al., 2025), which is a strong autoregressive generative model that uses spanning tree-based graph generation to perform multi-property conditional generation. We also include a graph genetic algorithm (Graph GA) (Jensen, 2019), which is based on target-specific optimization; for each target tuple we run it from scratch with oracle calls of 500 and 1000 (denoted GA-500 and GA-1000). There is no LLM-based solution achieving decent performance (below 3eV as total MAE) on HOMO-LUMO constrained generation. Thus, we only include Graph GA in the HOMO-LUMO baselines as it is the strongest solution for reward-aware scenario.

Metrics. We compute all the properties of generated samples and compare them with the target to get the MAE (mean absolute error). MAE is commonly used among molecular generation and design benchmarks (Wu et al., 2018). However, different properties have different ranges, and individual properties need to be normalized to the same range for multi-objective molecule design (Luukkonen et al., 2023). Therefore, we normalize the error by dividing QED error by 1, LogP error by 10 and MW error by 700 since QED range is from 0 to 1, LogP range is from -10 to 10 and most in-distribution MW range is from 100 to 800. Note that the normalizer for each error can be tuned when dealing with custom distribution or specific-property-preferred generation. Besides the whole range normalization, we also add the scalars we used for the optimizer's GRPO training (1 for QED, 6 for LogP and 100 for MW). No scalars are needed in HOMO-LUMO experiments as their value ranges are the same. Beyond MAE, we assess **set quality**. *Uniqueness* is the fraction of distinct molecules among the outputs, indicating the absence of duplicates. *Diversity* measures how dissimilar the set is on average, computed from ECFP4 fingerprints (Rogers and Hahn, 2010) with the Tanimoto similarity.

4.2 Results and Analysis

Protocol. GRPO is ground-truth–free and reward-based, so performance is not tied to a particular training distribution. We deliberately include HOMO/LUMO (computed via a DimeNet++ surrogate of DFT) to test M^{4}olGen under high-fidelity, simulation-grade oracles, not just fast RDKit estimators. To test generalization, we uniformly sample 100 target tuples

Table 1: Overall metrics across methods. Lower is better for error metrics; higher is better for diversity and uniqueness. Best per column in **bold**; second best underlined.

Method	QED err	logP err	MW err	Scaled total err	Norm. total err	Diversity	Uniqueness
LLMs							
gpt-4.1	0.115	0.697	49.182	0.723	0.255	0.823	1.0
gpt-4o-2024-05-13	0.115	0.847	60.203	0.858	0.285	0.868	1.0
Gemini-2.5-Flash	**0.078**	0.974	86.174	1.102	0.299	0.842	0.97
Claude-3.7-Sonnet	0.104	1.025	39.583	0.671	0.263	0.868	1.0
Claude-3.5-haiku	0.117	1.174	46.904	0.782	0.301	0.791	1.0
SmileyLlama-8B	0.374	2.385	196.235	2.734	0.893	0.853	1.0
DrugAssist-7B	0.176	2.44	165.047	2.233	0.656	0.845	0.38
Graph algorithms							
STGG+	<u>0.079</u>	0.418	23.56	0.385	<u>0.155</u>	0.879	1.0
Graph GA-500	0.131	0.806	15.016	0.415	0.233	<u>0.884</u>	1.0
Graph GA-1000	0.123	0.529	**7.95**	0.291	0.187	**0.886**	1.0
Our methods							
1-hop-GPT4o	0.130	0.423	10.404	0.305	0.187	0.879	1.0
2-hop-GPT4o	0.111	0.339	10.489	0.272	0.160	0.883	1.0
3-hop-GPT4o	0.103	<u>0.284</u>	<u>9.799</u>	**0.249**	**0.146**	<u>0.884</u>	1.0
1-hop-Qwen	0.152	0.319	16.176	0.367	0.204	0.869	1.0
2-hop-Qwen	0.132	**0.209**	11.864	0.285	0.168	0.876	1.0
3-hop-Qwen	0.120	0.237	10.365	<u>0.263</u>	0.159	0.878	1.0

Table 2: Performance comparison across methods on HOMO/LUMO–constrained molecular generation.

Method	HOMO err	LUMO err	Total err	Diversity	Uniqueness
Graph GA-500	0.369	0.396	0.765	0.881	0.957
Graph GA-1000	0.251	0.353	0.604	0.894	1.0
1-hop	0.301	0.239	0.540	0.874	1.0
2-hop	0.112	0.115	0.227	0.867	1.0
3-hop	**0.060**	**0.095**	**0.155**	0.862	1.0

$(\mathrm{QED}, \mathrm{LogP}, \mathrm{MW})$ and $\mathrm{HOMO}, \mathrm{LUMO}$ across admissible ranges. For each tuple and each baseline we run 10 independent trials under the same compute budget and report the *best-of-10*. Across settings, our normalized total error (NTE) decreases monotonically with hop count shown in both Table 1 and Table 2.

Main Results. Table 1 compares LLMs, graph baselines, and our method. Our best configuration (3-hop-GPT-4o) attains the lowest **NTE** (normalized total error) of **0.146**, improving over the strongest commercial model (GPT-4.1, 0.255) by **42.7%** and outperform the best non-LLM baseline (STGG+). Meanwhile, Qwen-based configuration also achieves competitive performance with second best NTE(0.159). This points out that our method does not rely on commerical models. Notably, the gap between our open-weights Qwen3-14B configuration and the GPT-4o configuration is only 0.013 in normalized total error, indicating that M^{4}olGen's gains derive primarily from Stage II's fragment-level optimizer and the multi-agent architecture rather than the underlying foundation model, which makes the framework fully accessible without proprietary APIs. Per metric, we obtain the best **logP** error (**0.209**) and the second-best **MW** error (<u>9.799</u>; GA-1000 is **7.95**). Relative to STGG-50×, our 3-hop reduces logP from 0.566 to 0.284 (**49.8%**) and MW from 63.917 to 9.799 (**84.7%**); STGG-50× achieves the best QED (**0.050**), while ours remains competitive (0.103). Diversity and uniqueness are high (Div ≈ 0.884, Uniq $= 1.0$), on par with the best graph baseline (Graph GA-1000, Div $= 0.886$).

Table 2 shows the performance of the generation under HOMO and LUMO constraints (unit is eV). M^{4}olGen significantly outperforms the Graph GA baseline. The 1-hop setting already reduces the total error to 0.540, demonstrating that a single controlled fragment edit guided by property feedback can effectively move molecules toward the target electronic profile. The 2-hop strategy reduces the total error to 0.227, achieving more

Table 3: Ablation study on retrieval and fragment-level optimizer (lower is better).

Method	QED err	logP err	MW err	Norm. total err
Stage1 (no retrieval)	0.111	0.970	68.555	0.307
Stage1 + retrieval	**0.098**	0.769	63.240	0.265
Stage1 + retrieval + 1-hop	0.130	0.423	10.404	0.187
Stage1 + retrieval + 2-hop	0.111	0.339	10.489	0.160
Stage1 + retrieval + 3-hop	0.103	**0.284**	**9.799**	**0.146**

than a **2×** improvement over Graph GA-1000. The 3-hop configuration achieves the best overall performance, with a total error of **0.155** and particularly low HOMO and LUMO errors (0.060 and 0.095, respectively). Notably, both HOMO and LUMO errors are simultaneously reduced, suggesting balanced multi-property optimization rather than overfitting to a single objective. While Graph-GA becomes increasingly time-consuming as oracle call increases, M^{4}olGen achieves comparable or better performance with nearly 90% less inference time. Because the optimization cost is amortized after a single training phase, M^{4}olGen is substantially more efficient for repeated or large-scale molecule generation tasks.

Computational Cost. We analyze the overhead of both stages. Stage I (GPT-4o-based) consumes **21,243** tokens and **74.3**s on average per target, versus **10,230** tokens and **184.8**s for a GPT-4o best-of-10 single-agent baseline. Context caching across iterations makes the multi-agent variant 2.5× faster wall-clock despite higher token use, while reducing total error from **0.285** to **0.245**. For Stage II, each additional hop adds only **1,248** tokens and **33**s. This is strictly linear in hop count and yields **23.7%** / **14.4%** / **8.8%** successive gains at hops 1 / 2 / 3. This gives users a transparent accuracy–cost trade-off: 1-hop for high-throughput screening, 3-hop when precision is paramount.

4.3 Ablation Study

Interpretation. We ablate three design choices on a held-out set: (i) Stage I without retrieval (baseline), (ii) Stage 1 with retrieval, and (iii) Stage I with retrieval followed by a fragment-level optimizer using 1/2/3 hops. We report per-property errors (QED, logP, MW) and the normalized total error ($e_{\text{norm}} = |\Delta\text{QED}| + |\Delta \log P|/10 + |\Delta\text{MW}|/700$) in Table 3. For visualization, we plot the *drop percentage* relative to the no-retrieval baseline, for each metric and method (Figure 2).

Effect of Retrieval. Adding retrieval already yields consistent gains: the normalized total error drops by **13.7%** (0.307 → 0.265), driven primarily by improvements in logP (**20.7**% drop) and MW (7.8% drop). Retrieval also gives the best standalone QED error among non-optimized variants (**0.098**, **11.7**% drop).

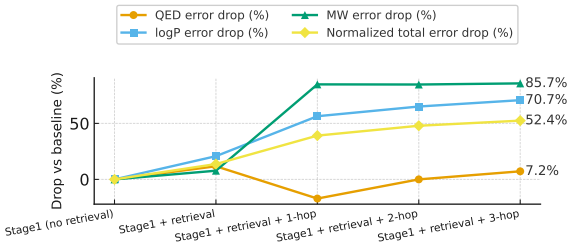

Figure 2: Ablation curves showing the *drop percentage* (higher is better) of each error metric relative to the no-retrieval baseline across methods. Curves are shown for QED, logP, MW, and the normalized total error.

Effect of the Fragment-level Optimizer. Introducing the optimizer produces the largest improvements, especially on MW. Moving from retrieval-only to 1/2/3 hops reduces MW error from ~ 63 to ~ 10 (**84.9**% drop vs. baseline), and steadily improves logP (drops of 56.4%, 65.1%, and **70.7**%). The overall normalized error decreases monotonically with more hops: 0.187 (1-hop, 39.1% drop), 0.160 (2-hop, 47.9% drop), and **0.146** (3-hop, **52.4**% drop). QED exhibits a small regression at 1-hop (as expected when trading off multi-objective targets), but recovers by 3-hop to a 7.0% drop versus baseline.

5 Conclusion

We introduced **M^{4}olGen**, a two-stage, fragment-level framework for *precise, property-constrained* molecular generation and a large, reasoning-ready dataset (BRICS fragments with neighbor pairs and measured property deltas) for training. Across QED, $LogP$, and MW and HOMO, LUMO targets, M^{4}olGen attains the lowest normalized total error among LLM and graph baselines, with monotonic gains as hop count increases. Taken together, these results validate our design choices and demonstrate the method's potential to scale to richer objectives.

Limitations

While promising, our study is limited by its reliance on computed properties (e.g., RDKit estimators). Our evaluation focuses on five representative properties (QED, LogP, MW, HOMO, LUMO). The framework itself is property-agnostic: the neighbor-pair dataset captures structural edit logic and is reused across tasks, so extending to new objectives (e.g., SA, TPSA) requires only re-annotating existing pairs with the new property function and re-running the GRPO reward. This work serves as an initial validation of applying GRPO to explore discrete chemical action spaces for precise property control, where rapid reward feedback and fundamental physicochemical objectives are essential for stable training. Also, although deeper hops continue to improve performance, they incur substantially higher computation cost with diminishing returns and this is a practical design trade-off.

References

Sungsoo Ahn, Binghong Chen, Tianzhe Wang, and Le Song. 2021. Spanning tree-based graph generation for molecules. In *International Conference on Learning Representations*.

Anthropic. 2024. Claude 3 model family: Opus, sonnet, haiku (model card). Anthropic Model Card. Accessed: 2025-09-24.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. 2023. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55.

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.

Jean-Luc Brédas. 2017. Organic electronics: does a plot of the homo–lumo wave functions provide useful information?

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

Feiyang Cai, Tianyu Zhu, Tzuen-Rong Tzeng, Yongping Duan, Ling Liu, Srikanth Pilla, Gang Li, and Feng Luo. 2024. A foundation model for chemical design and property prediction. *arXiv e-prints*, pages arXiv–2410.

Nicola De Cao and Thomas Kipf. 2018. Molgan: An implicit generative model for small molecular graphs. *ArXiv*, abs/1805.11973.

Joseph M. Cavanagh, Kunyang Sun, Andrew Gritsevskiy, Dorian Bagni, Yingze Wang, Thomas D. Bannister, and Teresa Head-Gordon. 2025. Smileyllama: Modifying large language models for directed chemical space exploration. *Preprint*, arXiv:2409.02231.

Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. 2020. Chemberta: Large-scale self-supervised pretraining for molecular property prediction. *ArXiv*, abs/2010.09885.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jorg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. 2008. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem*, 3(10):1503.

Qianggang Ding, Santiago Miret, and Bang Liu. 2024. Matexpert: Decomposing materials discovery by mimicking human experts. *Preprint*, arXiv:2410.21317.

Carl N. Edwards, T. Lai, Kevin Ros, Garrett Honke, and Heng Ji. 2022. Translation between molecules and natural language. *ArXiv*.

Nathan C Frey, Ryan Soklaski, Simon Axelrod, Siddharth Samsi, Rafael G´omez-Bombarelli, Connor W. Coley, and Vijay Gadepally. 2023. Neural scaling of deep chemical models. *Nature Machine Intelligence*, 5:1297 – 1305.

Jenna C. Fromer and Connor W. Coley. 2023. Computer-aided multi-objective optimization in small molecule discovery. *Patterns*, 4(2):100678.

Kenichi Fukui, Teijiro Yonezawa, Chikayoshi Nagata, and Haruo Shingu. 1954. Molecular orbital theory of orientation in aromatic, heteroaromatic, and other conjugated molecules. *The Journal of Chemical Physics*, 22(8):1433–1442.

Johannes Gasteiger, Shankari Giri, Johannes T. Margraf, and Stephan Günnemann. 2022. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *Preprint*, arXiv:2011.14115.

Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, and 1 others. 2012. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107.

Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J. Szostkiewicz, Jon M. Laurent, Muhammed T. Razzak, Andrew D. White, Michaela M. Hinks, and Samuel G. Rodriques. 2025. Robin: A multi-agent system for automating scientific discovery. *Preprint*, arXiv:2505.13400.

Constantinos Giaginis, Fotios Tsopelas, and Anna Tsantili-Kakoulidou. 2018. The impact of lipophilicity in drug discovery: Rapid measurements by means of reversed-phase hplc. In *Rational Drug Design: Methods and Protocols*, pages 217–228. Springer.

Rafael Gómez-Bombarelli, David Kristjanson Duvenaud, José Miguel Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. 2016. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4.

Google. 2025. Gemini 1.5 flash (model card and models overview). Google AI Gemini API Documentation. Accessed: 2025-09-24.

Philipp Guevorguian, Menua Bedrosian, Tigran Fahradyan, Gayane Chilingaryan, Hrant Khachatrian, and Armen Aghajanyan. 2024. Small molecule optimization with large language models. *Preprint*, arXiv:2407.18897.

John J Irwin and Brian K Shoichet. 2005. Zinc- a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1):177–182.

Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. 2021. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernández-Garcıa, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. 2023. Multi-objective gflownets. In *International conference on machine learning*, pages 14631–14653. PMLR.

Yunhui Jang, Jaehyung Kim, and Sungsoo Ahn. 2024. Structural reasoning improves molecular understanding of llm. In *Annual Meeting of the Association for Computational Linguistics*.

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.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Suhang Wang, Yu Meng, and Jiawei Han. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Annual Meeting of the Association for Computational Linguistics*.

Alexia Jolicoeur-Martineau, Aristide Baratin, Kisoo Kwon, Boris Knyazev, and Yan Zhang. 2025. Any-property-conditional molecule generation with self-criticism using spanning trees. *Transaction of Machine Learning Research (TMLR)*.

Bong-Gi Kim, Xiao Ma, Chelsea Chen, Yutaka Ie, Elizabeth W Coir, Hossein Hashemi, Yoshio Aso, Peter F Green, John Kieffer, and Jinsang Kim. 2013. Energy level modulation of homo, lumo, and band-gap in conjugated polymers for organic photovoltaic applications. *Advanced Functional Materials*, 23(4):439–445.

Hyomin Kim, Yunhui Jang, and Sungsoo Ahn. 2025. Mt-mol: Multi agent system with tool-based reasoning for molecular optimization. *arXiv preprint arXiv:2505.20820*.

Sizhe Liu, Yizhou Lu, Siyu Chen, Xiyang Hu, Jieyu Zhao, Yingzhou Lu, and Yue Zhao. 2024. Drugagent: Automating ai-aided drug discovery programming through llm multi-agent collaboration. *arXiv preprint arXiv:2411.15692*.

Hannes H Loeffler, Jiazhen He, Alessandro Tibo, Jon Paul Janet, Alexey Voronov, Lewis H Mervin, and Ola Engkvist. 2024. Reinvent 4: modern ai–driven generative molecule design. *Journal of Cheminformatics*, 16(1):20.

Sohvi Luukkonen, Helle W. van den Maagdenberg, Michael T.M. Emmerich, and Gerard J.P. van Westen. 2023. Artificial intelligence in multi-objective drug design. *Current Opinion in Structural Biology*, 79:102537.

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(5):525–535.

OpenAI. 2024a. Gpt-4o system card. `https://openai.com/index/gpt-4o-system-card/`. Accessed: 2025-09-24.

OpenAI. 2024b. Gpt-4o (version: gpt-4o-2024-05-13). OpenAI Platform Documentation. Accessed: 2025-09-24.

OpenAI. 2025. Gpt-4.1 model. OpenAI Platform Documentation. Accessed: 2025-09-24.

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Simon Johansson, Hongming Chen, Sergey Nikolenko, Alan Aspuru-Guzik, and Alex Zhavoronkov. 2020. Molecular sets (moses): A benchmarking platform for molecular generation models. *Preprint*, arXiv:1811.12823.

David Rogers and Mathew Hahn. 2010. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754.

Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. 2018. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. 2020. Graphaf: a flow-based autoregressive model for molecular graph generation. *ArXiv*.

Summykai. 2024. Qwen3-14B-Chem-Dyn-Tokenizer. `https://huggingface.co/summykai/Qwen3-14B-chem-dyn-tokenizer`. Hugging Face model.

Ivana Vichentijevikj, Kostadin Mishev, and Monika Simjanoska Misheva. 2025. Prompt-to-pill: Multi-agent drug discovery and clinical simulation pipeline. *bioRxiv*.

Travis T Wager, Xinjun Hou, Patrick R Verhoest, and Anabella Villalobos. 2016. Central nervous system multiparameter optimization desirability: application in drug discovery. *ACS chemical neuroscience*, 7(6):767–775.

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, and 1 others. 2024. Efficient evolutionary search over chemical space with large language models. *arXiv preprint arXiv:2406.16976*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*.

Gerhard Weiss. 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press.

Ronald J. Williams. 2004. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Michael Wooldridge. 2009. *An Introduction to MultiAgent Systems*. John Wiley & Sons.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530.

Geyan Ye, Xibao Cai, Houtim Lai, Xing Wang, Junhong Huang, Longyue Wang, Wei Liu, and Xiangxiang Zeng. 2023. Drugassist: A large language model for molecule optimization. *Preprint*, arXiv:2401.10334.

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

Chengxi Zang and Fei Wang. 2020. Moflow: An invertible flow model for generating molecular graphs. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Zilong Wang, Aliaksandra Shysheya, Jonathan Crabbé, Shoko Ueda, Roberto Sordillo, Lixin Sun, Jake Smith, Bichlien Nguyen, Hannes Schulz, Sarah Lewis, Chin-Wei Huang, Ziheng Lu, Yichi Zhou, and 7 others. 2025. A generative model for inorganic materials design. *Nature*.

Huan Zhang, Yu Song, Ziyu Hou, Santiago Miret, and Bang Liu. 2024. Honeycomb: A flexible llm-based agent system for materials science. *Preprint*, arXiv:2409.00135.

Xiaoying Zhang, Hao Sun, Yipeng Zhang, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. 2025. Critique-grpo: Advancing llm reasoning with natural language and numerical feedback. *Preprint*, arXiv:2506.03106.

Zihan Zhao, Da Ma, Lu Chen, Liangtai Sun, Zihao Li, Yi Xia, Bo Chen, Hongshen Xu, Zichen Zhu, Su Zhu, Shuai Fan, Guodong Shen, Kai Yu, and Xin Chen. 2025. Developing chemdfm as a large language foundation model for chemistry. *Cell Reports Physical Science*, 6(4):102523.

Xin Zheng, Jie Lou, Boxi Cao, Xueru Wen, Yuqiu Ji, Hongyu Lin, Yaojie Lu, Xianpei Han, Debing Zhang, and Le Sun. 2024. Critic-cot: Boosting the reasoning abilities of large language model via chain-of-thoughts critic. In *Annual Meeting of the Association for Computational Linguistics*.

P. Zhou, J. Wang, C. Li, and 1 others. 2025. Instruction multi-constraint molecular generation using a teacher–student large language model. *BMC Biology*, 23:105.

Zhenpeng Zhou, Steven M. Kearnes, Li Li, Richard N. Zare, and Patrick F. Riley. 2018. Optimization of molecules via deep reinforcement learning. *Scientific Reports*.

A Appendix

A.1 Dataset Details

We combine all the molecules from ZINC (Irwin and Shoichet, 2005), CHEMBL (Gaulton et al., 2012) and MOSES (Polykovskiy et al., 2020) together, filter and delete the duplicates. From each molecule we obtain its SMILES, molecular formula, QED, logP, and molecular weight computed with RDKit and HOMO, LUMO with our pretrained evaluator based on DimeNet++ (Gasteiger et al., 2022). We further derive a fragment decomposition and an inter-fragment connectivity map (identifying the bonds between fragments). The final dataset contains 2,945,596 molecules and, to the best of our knowledge, is the largest resource coupling molecular properties with fragment-based structural annotations.

Starting from our unified corpus, we build a reasoning-ready resource through an automated pipeline: (i) **standardize & deduplicate** molecules via RDKit canonical SMILES, neutralize, and enforce valence/aromaticity sanity checks; (ii) **annotate properties** (QED, LogP, MW) with RDKit and (HOMO, LUMO) with pretrained DimeNet++; (iii) **fragmentize** each molecule with BRICS to obtain a fragment multiset $\Phi(m)$ and an inter-fragment connectivity map (which fragments are joined and at which bonds), yielding an actionable edit space; (iv) **construct neighbor pairs** by scanning for molecules that differ by exactly one fragment-level edit (`add`/`remove`/`replace`), while enforcing edit sanity (e.g., element-count conservation for `replace`) and RDKit validity for the edited product; and (v) **label supervision** by recording the edit type, edited fragments, and signed property deltas $(\Delta\mathrm{QED}, \Delta\mathrm{LogP}, \Delta\mathrm{MW})$, plus the distance-to-target objective used by our optimizer. This process yields a **neighbor-pair corpus of ∼1,171,193 single-edit pairs**. For each molecule we also materialize its **1-hop neighbor list** based on the fragment multiset edit distance as shown in figure 3, from which we grow neighbor trees/forests. These structures serve two roles: they seed *retrieval-anchored prototyping* in Stage I and provide *experience-based, reward-compatible supervision* for GRPO in Stage II, enabling controllable multi-hop refinement under exact numeric targets. Each entry is formatted as a natural language prompt with a one-step edit answer, e.g.:

> Given is an intermediate molecule SMILES <SMILES>O=C(NCc1nccc2ccccc12)c1ccc[nH]c1=O</SMILES>, which is composed of fragments ['C()=O', 'N', 'C', 'c1nccc2ccccc12', 'c1ccc[nH]c1=O']. Propose a single replace, add or remove step on fragment level that makes the new molecule's QED <QED>0.146</QED> lower, LogP <LogP>0.366</LogP> higher, and Molecular Weight <MW>53.068</MW> lower.

> Replace c1ccc[nH]c1=O with c1nc2nc(C)cc(C)n2n1 to form <SMILES>Cc1cc(C)n2nc(C(=O)NCc3nccc4ccccc34)nc2n1</SMILES>.

GRPO itself does not need any ground truth for editing, but all property changes are still derived from real data to preserve distribution realism.

A.2 Stage I Algorithm

Algorithm 1 Stage I: Local Optimal Candidate Generation via Multi-Agent Planning

Require: User query q, molecule database $\mathcal{M}$, thresholds ϵ_i, max iterations T

```
P* ← Decomposer(q)
N ← Retriever(P*, M, ε_i)
m_0 ← InitialGeneration(q, N, P*)
Initialize reasoning history H ← []
for t = 1 to T do
    a_t ← Reasoner(q, N, H, P*)
    m_t ← Edit(m_{t-1}, a_t)
    H ← H ∪ {a_t}
    if is_valid(m_t, P*, ε_i) then
        return m_t
    end if
end for
return valid m_T if any
```

A.3 End-to-End Demo: From Local Reasoner to GRPO Refinement

Target. We aim for **QED** ≈ 0.70, **LogP** ≈ 1.50, and **MW** ≈ 300.

Stage 1 — Iterative construction (LLM planner). The planner begins from scratch and proposes *fragment-level* edits while reading back numeric feedback at each step.

Step 1. It proposes `CCN(CC)C(=O)C(C1CC1)S(=O)=O` based on relevant molecules, reasoning that a compact sulfonamide with small rings could balance QED and LogP. Feedback shows QED $= 0.674$ (below by 0.026), LogP $= 0.245$ (below by 1.255), MW $= 219.306$ (below by 80.694). The model decides to raise both LogP and MW.

Step 2. To add hydrophobic mass, it benzylates the amide nitrogen, yielding `CCN(Cc1ccc`

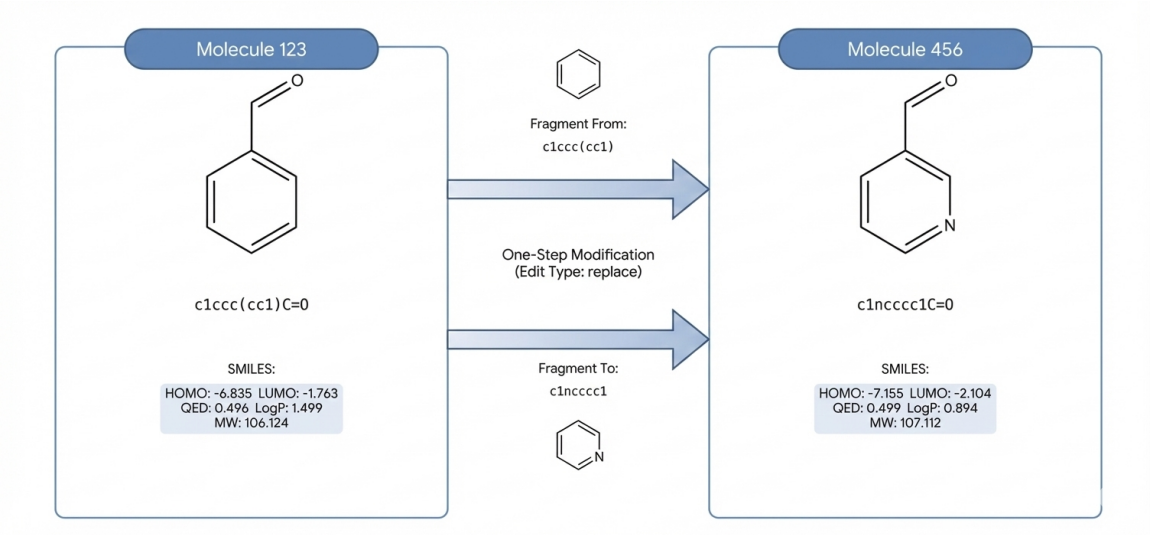

Figure 3: The demo of nodes and edges of molecule neighbor reasoning dataset.

cc1)C(=O)C(C1CC1)S(=O)=O. Feedback: QED $= 0.803$ (above by 0.103), LogP $= 1.425$ (just 0.075 low), MW $= 281.377$ (still 18.623 low). The ring helped; MW needs a modest push upward.

Step 3. It enlarges the small ring to a cyclohexyl to push MW/LogP: CCN(Cc1ccccc1)C(=O)C(C1CCCCC1)S(=O)=O. Feedback: QED $= 0.819$ (high by 0.119), LogP $= 2.595$ (high by 1.095), MW $= 323.458$ (high by 23.458). Overshot both LogP and MW.

Step 4. It trims to cyclopentyl: CCN(Cc1ccccc1)C(=O)C(C1CCCC1)S(=O)=O. Feedback: QED $= 0.820$ (high by 0.120), LogP $= 2.205$ (high by 0.705), MW $= 309.431$ (high by 9.431). Still too heavy and too lipophilic.

Step 5. To temper LogP/MW while retaining aromaticity, it swaps phenyl $\rightarrow$ pyridine: CCN(Cc1nccc1)C(=O)C(C1CCCC1)S(=O)=O. Feedback: QED $= 0.811$ (high by 0.111), LogP $= 1.600$ (high by 0.100), MW $= 310.419$ (high by 10.419). Closer on LogP, MW still a bit high.

Step 6 (seed for Stage 2). It reduces the ring to a butyl chain to lower MW/LogP: CCN(Cc1ncccc1)C(=O)C(CCC)S(=O)=O. Feedback: QED $= 0.764$ (high by 0.064), LogP $= 1.210$ (low by 0.290), MW $= 284.381$ (low by 15.619). This is the best Stage-1 candidate (normalized total error $= 0.116$) and becomes the seed for Stage 2.

Stage 2 — GRPO Refinement (Accepted Path with Reasoning). We now switch to the optimizer trained with GRPO. At each hop, we ask for a single fragment edit that moves QED/LogP/MW by specified deltas in the right directions, then accept only moves that improve the objective.

Hop 1. From the seed CCN(Cc1ncccc1)C(=O)C(CCC)S(=O)=O, we request: decrease QED by 0.064, increase LogP by 0.290, and increase MW by 15.619. **Reasoning.** The model replaces the sulfone side chain with a bicyclic, more drug-like fragment to add hydrophobic mass while modulating polarity. **Edit.** Replace C(=O)C(CCC)[SH](=O)=O $\rightarrow$ C1=CNC(N)C(O)C=C(C)CC1=C, producing CCN(Cc1ncccc1)C1=CNC(N)C(O)C=C(C)CC1=C. The move improves the objective and is *accepted*.

Hop 2. From that intermediate, we request: further decrease QED by 0.040, decrease LogP by 0.386, and decrease MW by 14.433. **Reasoning.** The optimizer softens hydrophobicity and trims mass while preserving the newly introduced scaffold connectivity. **Edit.** Replace N()C1=CNC(N)C(O)C=C(C)CC1=C $\rightarrow$ NC1=CNNC=CC(O)CC(C)C1, yielding CCNC1=CNNC=CC(O)CC(C)C1Cc1ncccc1. This further reduces the objective and is *accepted*.

Final Outcome. The best molecule along this path is CCNC1=CNNC=CC(O)CC(C)C1Cc1ncccc1 with QED $= 0.681$, LogP $= 1.700$, MW $= 302.422$, and a normalized total error of 0.042. In summary, Stage 1 quickly assembled a plausible prototype with sensible fragment choices, and Stage 2 applied two targeted, GRPO-guided edits that traded off hydrophobic mass and polarity to tighten alignment with all three numeric targets.

A.4 Use of LLMs

Large Language Models (LLMs) were used solely for writing refinement such as grammar and syntax improvements.