MOLLINGO: MOLECULE-NATIVE REPRESENTATIONS FOR LLM-POWERED SCIENTIFIC AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have shown strong general reasoning capabilities, yet molecular design remains challenging because it requires precise structural reasoning, specialized computational tools, and iterative search over a large chemical space. We present **MolLingo**, a multi-agent framework that addresses these challenges through chemically informed representations, tool-grounded reasoning, and coordinated optimization. We introduce BRICS-based Fragment Enumeration (BFE), which combines chemically valid BRICS fragmentation with corpus-level fragment frequencies to represent molecules as recurring chemical blocks. Encoding these blocks with both SMILES and common chemical names exposes meaningful structural units while preserving molecular connectivity. Building on this molecular representation, MolLingo integrates a Chemist Agent, a Literature Agent, an Orchestrator, and shared memory, together with tools for ADMET prediction, molecular docking, and binding retrieval. For p*roperty optimization*, across GPT-5.4, Claude-4.6-Sonnet, and Gemini-3-Pro, block-based input improves the joint objective of property improvement and structural similarity over raw SMILES under matched LLM backbones. For *binding optimization*, incorporating docking-derived structural context increases mean docking-score improvement from 2.4% with direct prompting to 10.4% with GPT-5.4. Iterative experiments further show that memory improves structural preservation and reduces redundant edits, while adaptive search allocation improves optimization as the candidate budget increases. Together, these results show that molecular representation, structural context, and agentic coordination provide complementary signals for LLM-guided molecular design. Code is available at `https://anonymous.4open.science/status/MolLingo-7450`.

1 INTRODUCTION

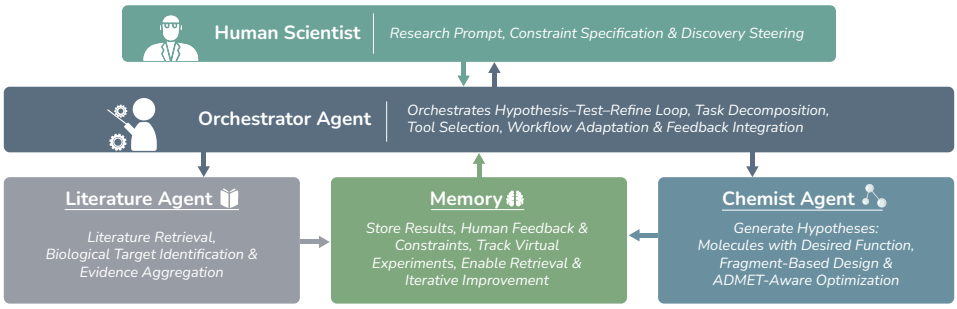


Figure 1: MolLingo agent-based architecture.

Large language models (LLMs) have demonstrated strong capabilities across natural language processing, vision, and mathematical reasoning (Achiam et al., 2023). Yet their performance remains limited on chemistry-intensive tasks such as generating and optimizing drug candidates (Guo et al., 2023; Li et al., 2024), where successful reasoning requires not only chemical knowledge, but also precise understanding of molecular structure, access to specialized computational tools, and iterative

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

decision-making over a large chemical search space. We hypothesize that this gap arises from three key challenges.

First, the representation presented to an LLM is poorly aligned with how chemical structure is naturally organized. SMILES linearizes a molecular graph into a one-dimensional sequence (Weininger, 1988). To reason about structural modifications directly from SMILES, an LLM must implicitly reconstruct the underlying molecular graph and identify chemically meaningful substructures from atom- and bond-level symbols, which becomes particularly difficult for long or structurally complex molecules (Guo et al., 2023). Fragment-based molecular language models therefore motivate representations based on recognizable functional groups, ring systems, and recurring molecular building blocks rather than arbitrary substrings of a full-molecule SMILES (Edwards et al., 2025; Yue et al., 2024). Second, molecular design often depends on information that cannot be reliably inferred from language alone. Predictive models, molecular docking, structural databases, and experimental literature provide complementary evidence about properties, binding interactions, and biological mechanisms. Prior chemistry agents demonstrate the value of enabling LLMs to invoke specialized tools and incorporate their outputs into subsequent decisions (Bran et al., 2024; Boiko et al., 2023). Third, drug optimization is inherently iterative. A proposed modification must be evaluated, compared with previous candidates, and used to inform the next round of design. This trial-and-error process requires maintaining search history, avoiding redundant modifications, and balancing exploration of new chemical space with exploitation of promising directions, paralleling agent frameworks that refine decisions using actions, observations, and memory (Yao et al., 2023; Shinn et al., 2023).

We address these challenges with **MolLingo**, a multi-agent framework for tool-guided molecular design. We first introduce BRICS-based Fragment Enumeration (BFE), which combines chemically valid BRICS fragmentation with corpus-level fragment frequencies to represent molecules as recurring, chemically meaningful blocks. Each block can be expressed using both its SMILES representation and a common chemical name, providing an interface that preserves molecular structure while exposing concepts that are more likely to be familiar to LLMs through pretraining. Rather than requiring the model to reason entirely over a long atom-level SMILES string, this block-based representation enables molecular modifications to be formulated in terms of recognizable functional groups and building blocks. Building on this representation, MolLingo organizes molecular design as an agentic, iterative workflow. A *Chemist Agent* performs chemical reasoning and proposes molecular modifications while interacting with computational tools including ADMET predictors, molecular docking, and binding databases. A *Literature Agent* retrieves and synthesizes disease, target, and mechanistic evidence from the scientific literature. An *Orchestrator* coordinates these agents and manages the balance between exploration and exploitation during chemical-space search, while shared memory records previous candidates, evaluations, and design decisions to guide subsequent iterations. Together, MolLingo combines chemically aligned representations, external scientific tools, and iterative search to better connect the general reasoning capabilities of LLMs with the structural and computational requirements of molecular design.

Our evaluation separates three questions: whether block-based molecular representations improve ADMET optimization under a fixed LLM backbone; whether incorporating tool-derived information improves molecular design; and how memory and search allocation influence iterative molecular optimization. Experimental results show that block-based representations improve the joint property–similarity objective compared with raw SMILES, suggesting that block-based molecular descriptions provide a more effective way to probe the chemical knowledge encoded in LLMs. Docking-guided experiments further evaluate the value of structural context for binding optimization, and iterative experiments examine how memory and adaptive search affect optimization trajectories under different candidate budgets. Finally, we present an end-to-end case study that connects literature-based target reasoning with downstream molecular optimization, together with component-level ablations that isolate the contribution of individual design choices.

Our main contributions are:

- We propose a molecule-native representation that probes the parametric chemical knowledge of general-purpose LLMs through recurring, chemically meaningful BFE blocks rather than full-molecule SMILES alone.
- We introduce BFE, which selects reusable molecular blocks through BRICS-based enumeration and corpus frequency, reducing per-molecule vocabulary construction from $O(n^4)$ in graph BPE to $O(n^3)$, where n is the number of primitive BRICS blocks.

- We develop an agentic drug-discovery system that coordinates literature retrieval, molecular design, scientific tools, shared memory, and adaptive orchestration across the early-stage discovery workflow.

2 THE MOLLINGO FRAMEWORK

2.1 THE MOLLINGO ARCHITECTURE

MolLingo supports multiple stages of early drug discovery, spanning **target retrieval**, **hit generation**, and **hit-to-lead optimization**. An overview of the early-stage computational drug discovery pipeline is provided in Fig. 4 and Appendix A. To coordinate these tasks, MolLingo organizes the workflow across three specialized agents: an **Orchestrator Agent**, a **Literature Agent**, and a **Chemist Agent** (Fig. 1). These agents interact through a centralized **shared memory** that records retrieved biological evidence, molecular candidates, computational evaluations, and previous design decisions. Rather than operating as a fixed sequence of independent modules, each agent conditions its actions on the accumulated state of the workflow, allowing information gathered at earlier stages to guide downstream molecular search and optimization.

Shared Memory. The shared memory provides a persistent state accessible to all agents throughout the workflow. It stores structured information, including candidate molecules, molecular properties, docking scores, and target annotations, together with unstructured information such as literature summaries and design rationales. Agents read relevant context from memory before taking an action and write newly generated evidence or results back after completion. The memory therefore provides continuity across otherwise separate retrieval, screening, and optimization steps, while also allowing previous candidates and evaluations to inform subsequent decisions. During iterative molecular optimization, this history can help preserve similarity to the starting molecule and discourage repeated modifications that have already been explored. Details of the memory schema and retrieval mechanism are provided in Appendix H.

Orchestrator Agent. The Orchestrator controls the progression of the discovery workflow. Given an initial objective, such as identifying compounds for a disease-associated target or optimizing an existing molecular candidate, it determines the next task to perform and delegates it to the appropriate specialized agent. Depending on the current state of the search, this may involve gathering additional biological evidence, identifying candidate binders, evaluating a proposed molecule, or initiating another round of molecular optimization. The Orchestrator uses the information accumulated in shared memory to adapt subsequent actions and to balance exploration of alternative candidates with further optimization of promising ones. The orchestration strategy and prompt design are described in Appendix I.

Literature Agent. The Literature Agent retrieves and synthesizes biological evidence relevant to the current discovery objective. Given a disease or biological context, it searches sources including PubMed, UniProt, and ChEMBL to identify reported disease–target relationships and summarize the supporting evidence. Its role is to organize existing evidence rather than to establish novel target validity. The resulting target annotations, supporting literature, and evidence summaries are stored in shared memory and can subsequently guide compound identification and optimization. This disease-to-target step differs from the target-to-molecule search performed by the Chemist Agent.

Chemist Agent. The Chemist Agent performs molecular retrieval, evaluation, and modification. During **hit identification**, it searches protein–ligand binding data from ChEMBL (Gaulton et al., 2012) for known candidate binders and can invoke DrugCLIP (Gao et al., 2023) as a screening alternative when suitable experimental binders are unavailable. During **hit-to-lead optimization**, it uses protein–ligand docking to obtain pocket geometry and residue-level interaction information and uses this structural context to guide fragment-level molecular modifications. During **lead optimization**, the agent iteratively proposes and evaluates molecular edits using external tools for docking, ADMET prediction, synthesizability, and drug-likeness. Each evaluated candidate and its associated computational results are recorded in shared memory, allowing later iterations to build on previous successes and avoid repeatedly exploring unproductive modifications. Together, these components separate high-level workflow planning, biological evidence retrieval, and molecular reasoning while maintaining a shared state across the discovery process. MolLingo therefore provides a common agentic framework in which molecular representations, literature evidence, computational tools, and previous search experience can jointly inform subsequent molecular design decisions.

2.2 BLOCK-BASED SMILES FOR MOLECULAR EDITING

SMILES represents molecular connectivity as a one-dimensional character sequence, but it does not explicitly expose the chemically meaningful units over which molecular edits are often made. Small character-level changes can invalidate the syntax or alter connectivity in unintended ways, while a single chemical substructure may correspond to multiple, non-contiguous regions of the SMILES string (Fig. 2a,b). We instead represent each molecule as an ordered sequence of chemically meaningful blocks, such as ring systems, linkers, and recurring pharmacophoric or structural fragments. Each block is represented by a block-level SMILES string and, when available, a common chemical name (Fig. 2c). The representation is constructed in the following steps.

(1) Vocabulary construction. BRICS-based Fragment Enumeration (BFE) builds a fragment vocabulary from a corpus of 20M drug-like molecules. Our intuition follows byte-pair encoding (BPE) in language: primitive units that frequently occur together can be grouped into larger units that capture recurring structural patterns. In molecules, we begin with BRICS fragments (Degen et al., 2008), which are obtained by breaking chemically valid, synthesis-aware bonds, and seek frequently occurring combinations of these fragments as higher-level molecular blocks. A direct graph analogue of BPE would repeatedly identify and merge the most frequent adjacent fragment pairs across the corpus. However, graph merging is computationally expensive because each merge requires repeated molecular partitioning, graph reconstruction, and sanitization across many rounds. BFE avoids this iterative process by exploiting the structure of BRICS decompositions. For each molecule, it enumerates all valid contiguous blocks by cutting pairs of BRICS bonds, treating the molecular termini as virtual boundaries and discarding branched decompositions, while accumulating block frequencies in a single pass through the corpus. This produces a vocabulary of recurring, synthetically grounded molecular blocks without repeated corpus-wide merging. Appendix B provides the full algorithm and complexity analysis.

(2) Molecular fragmentation. Given a new molecule, we use the learned BFE vocabulary to select a decomposition into recurring molecular blocks. We first identify all BRICS bonds and enumerate candidate non-branching decompositions, producing a set $\mathcal{T}$ of possible block sequences. Each candidate $t \in \mathcal{T}$ is represented as $[t_1, \ldots, t_k]$, where each t_i is a molecular block. Because different combinations of adjacent primitive BRICS fragments can produce valid decompositions, the same molecule may admit alternatives such as $[t_1 t_2, t_3, t_4]$ or $[t_1, t_2 t_3, t_4]$.

We select among these candidates using a coarse-to-fine criterion that favors larger, well-supported blocks. We first retain decompositions with the minimum number of blocks, thereby preferring longer composite units that capture more local structural context. We then require every block to occur in the learned vocabulary with frequency at least $f_{\min}$. If multiple candidates remain, we choose the decomposition whose block frequencies are most balanced:

$$t^* = \arg\min_{t \in \mathcal{T}^*} \operatorname{std}\left(\{\mathrm{freq}(t_i)\}_{i=1}^{k}\right),$$

where $\mathcal{T}^*$ denotes the candidates satisfying the minimum-block-count and frequency-threshold criteria, and $\mathrm{freq}(\cdot)$ is the corpus frequency of a block. This final criterion avoids decompositions dominated by a mixture of extremely common and poorly represented blocks. The complete fragmentation procedure is given in Algorithm 2.

(3) Block-based SMILES with common names. After selecting the optimal decomposition, each block is converted to a SMILES string with wildcard atoms marking its attachment points. Because the retained decompositions are non-branching, each block has at most two neighboring blocks and therefore at most two attachment points, denoted by isotope-labeled wildcards such as `[1*]` and `[2*]`. The block-level SMILES preserves the exact atoms, substituents, and attachment positions required to reconstruct the original molecular connectivity.

Fig. 2 illustrates the representation for imatinib. BFE decomposes the molecule into the following blocks: `[2*]c1cccnc1`, `[2*]Nc1nccc([1*])n1`, `[2*]c1ccc(C)c([1*])c1`, `[1*]NC(=O)c1ccc([2*])cc1`, `[1*]CN1CCN(C)CC1`. These blocks are paired with scaffold-level names including `pyridine`, `2-aminopyrimidine`, `toluene`, `benzamide`, and `piperazine`. The names provide familiar chemical descriptors, while the corresponding SMILES strings retain the exact local structure and connectivity. At a higher level, the molecule can therefore be summarized as

`pyridine` → `2-aminopyrimidine` → `toluene` → `benzamide` → `piperazine`.

This representation explicitly exposes molecular block boundaries and attachment points while preserving the information required to reconstruct the original molecule. It also provides the LLM with both precise structural encodings and familiar chemical names, allowing fragment-level edits to be expressed at a more chemically meaningful granularity than raw SMILES alone. The contribution of common-name annotations is evaluated separately in Appendix M.1.

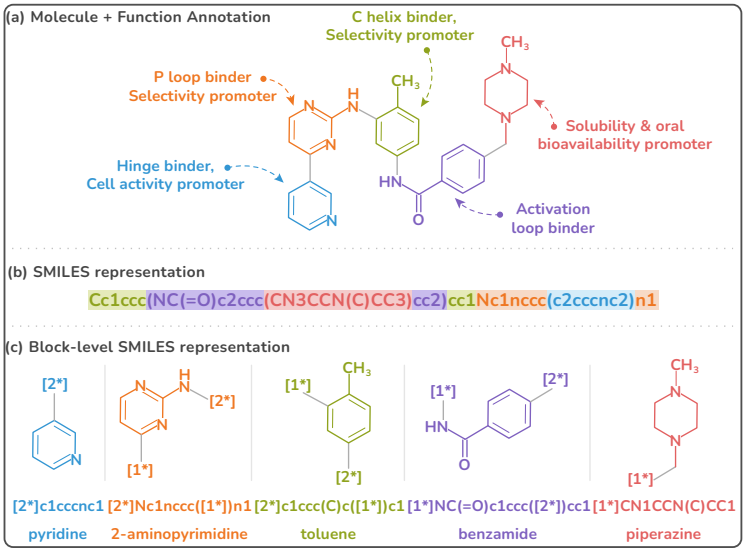


Figure 2: Molecular representations. (a) Functional blocks in the molecular graph. (b) Raw SMILES does not explicitly expose these block boundaries. (c) The block-based representation pairs each block SMILES with a common chemical name while retaining attachment information.

As a qualitative probe of how the representation changes model behavior, we compare Qwen2-7B-Instruct attention patterns for imatinib represented as raw SMILES and as named molecular blocks. Attention associated with the queried functional group is more localized under the block-based representation (Appendix F). We treat this analysis as an illustration of differences in model attention rather than as evidence of chemical understanding or a causal explanation for downstream optimization performance.

2.3 LLM-Guided Molecular Design

MolLingo approaches molecular design as a reasoning task grounded in chemistry and biology knowledge. Rather than treating molecular optimization as a purely statistical generation process, MolLingo leverages the Chemist Agent to reason over block-based representations, structural context from protein–ligand interactions, and accumulated knowledge from the shared memory module. This section describes these three stages of the drug design pipeline: hit identification, where candidate binders are retrieved or screened against a target protein (§2.3.1); hit-to-lead progression, where promising hits are expanded through context-aware fragment growing (§2.3.2); and lead optimization, where candidate molecules are iteratively refined to improve binding affinity and drug-like properties (§2.3.3).

2.3.1 Hit Identification

The goal of hit identification is to find an initial set of molecules that bind to the target protein with sufficient affinity to serve as starting points for further optimization. MolLingo pursues two complementary strategies depending on data availability. When experimental binding data are available, the Chemist Agent queries protein–ligand binding databases such as ChEMBL (Gaulton et al., 2012) to retrieve known binders for the target protein. Retrieved candidates are ranked by binding affinity and drug-likeness score (QED (Bickerton et al., 2012)), balancing potency with molecular quality before being passed to the hit-to-lead stage. When experimental data are unavailable (as is common for novel or understudied targets), MolLingo performs ligand-based virtual screening over a curated fragment library using DrugCLIP (Gao et al., 2023), a contrastive learning model that embeds proteins and ligands into a shared semantic space, enabling efficient similarity-based retrieval

without explicit docking. The fragment library used for screening is described in Appendix C. The top-ranked fragments are then forwarded to the hit-to-lead stage as initial hits for further elaboration.

The DrugCLIP fallback is an available screening mechanism, but has not been evaluated here on genuinely novel or understudied targets or against standard virtual-screening baselines. The reported hit-identification comparison therefore supports only the evaluated ChEMBL-based setting, not generalization of this fallback to new targets.

2.3.2 HIT-TO-LEAD: CONTEXT-AWARE FRAGMENT GROWING

The hit-to-lead stage transforms an initial hit fragment into a more potent and drug-like lead compound by progressively growing the fragment within the binding site of the target protein. Unlike conventional fragment growing approaches that rely on exhaustive enumeration or purely data-driven generation, MolLingo performs context-aware fragment growing by combining structural information from molecular docking with the chemical reasoning capabilities of the Chemist Agent. This allows the system to propose chemically meaningful extensions at atomic positions where new fragments are most likely to form favorable interactions with the binding site, mimicking the reasoning process of a medicinal chemist who considers both the geometry of the pocket and the chemical nature of nearby residues when deciding where and how to grow a fragment.

Given a hit fragment F and a protein target P, we first perform molecular docking to obtain the binding pose of F within the active site of P, saving the resulting receptor and ligand coordinates as structures R and L, respectively. We then analyze the docking pose to identify the most promising atomic positions for fragment growing, which we refer to as *growth hotspots*. For each heavy atom $a_i \in \mathcal{A}_L$ of the docked ligand, we characterize two complementary properties of its local environment: the neighboring residues and the available space. Neighboring residues $\mathcal{R}_i$ are defined as all receptor residues containing at least one heavy atom within a contact threshold $d_c = 7.0$ Å of a_i:

$$\mathcal{R}_i = \{r \in R \mid \exists\, a \in r : \|a_i - a\| \leq d_c\}.$$

The available volume V_i is estimated by constructing a cubic grid $\mathcal{G}_i$ centered at a_i with edge length 5.0 Å and resolution 0.5 Å, and counting grid points that satisfy van der Waals clearance constraints with respect to both the receptor and the ligand: $V_i = n_i \times (0.5)^3$, where $n_i = \left|\{g \in \mathcal{G}_i \mid d_R(g) > 2.2 \text{ Å} \ \wedge \ d_L(g) > 1.2 \text{ Å}\}\right|$, and $d_R(g)$, $d_L(g)$ are the minimum distances from grid point g to any heavy atom in the receptor and ligand respectively. The threshold of 2.2 Å for the receptor approximates the van der Waals radius of a carbon atom (∼1.7 Å) plus a small probe radius, reflecting the minimum clearance required to avoid steric clash with receptor atoms (Bondi, 1964). The threshold of 1.2 Å for the ligand is set to the approximate van der Waals radius of a hydrogen atom, ensuring that grid points immediately adjacent to the existing ligand are excluded while preserving attachment-accessible space at ligand periphery.

Together, the atom type (type(a_i)), available volume (V_i), and neighboring residues ($\mathcal{R}_i$) describe each potential growth position. Hotspots are ranked by decreasing V_i, and the top-k candidates are retained. For each hotspot, the Chemist Agent receives three types of input: the target protein and docking-derived context (V_i, $\mathcal{R}_i$), the attachment atom's element type type(a_i), and the SMILES of the BFE block containing that atom. The full-fragment variant replaces only this last input with the SMILES of the full growing fragment. This design conditions fragment proposals on pocket geometry, steric accessibility, and nearby residues rather than target identity alone. The full pipeline is illustrated in Fig. 3, with pseudocode in Appendix E and a within-workflow representation comparison in Table 2.

2.3.3 LEAD OPTIMIZATION

Lead optimization aims to improve predicted ADMET properties while limiting structural deviation from the starting lead. The Chemist Agent iteratively edits the block-based representation introduced in §2.2. At each step, the agent receives the current molecule and its predicted ADMET profile (Appendix D). It proposes block substitutions or changes to functional groups within a block, accompanied by a rationale for the intended property change. These rationales are design hypotheses, and each candidate is evaluated by the property oracles.

The prompt favors conservative edits, and Tanimoto similarity monitors deviation from the original lead; similarity alone does not ensure preservation of binding affinity or binding mode. When memory is enabled, prior edits, rationales, and optional oracle scores are included in subsequent prompts.

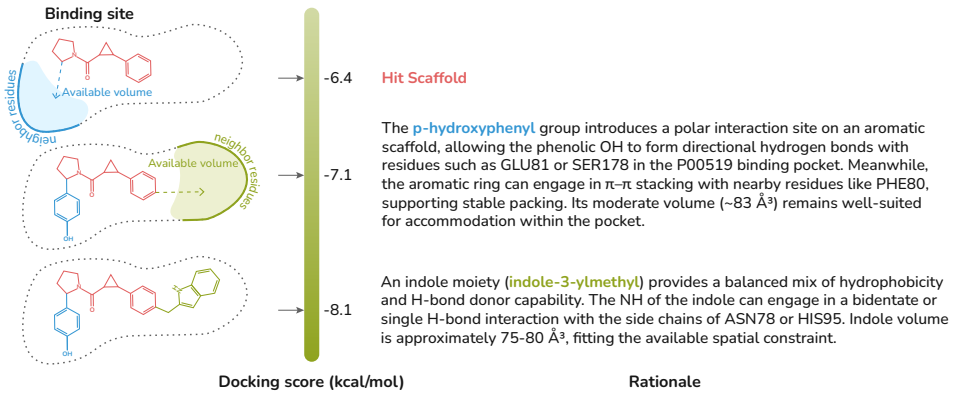


Figure 3: **Hit-to-Lead Optimization.** Starting from a hit scaffold and its docking pose with the target protein, the available volume within the binding site and neighboring amino acid residues are identified. Using this structural and biological context, the LLM reasons over fragment growth to iteratively expand the scaffold into a full molecule optimized for binding affinity.

Optimization uses either a fixed iteration budget or adaptive allocation by the Orchestrator, as specified for each experiment. Candidates are then selected using the relevant endpoint or aggregate ADMET objective.

3 Experimental Results

We evaluate four tasks: ADMET optimization on Withdrawn 2.0, hit-to-lead fragment growing on 30 protein targets, molecular property optimization on TOMG-Bench, and hit identification against DrugGEN on CDK2 and AKT1. The first two are presented below; TOMG-Bench and DrugGEN results appear in Appendix F. Target retrieval is evaluated separately on 30 diseases in Appendix G. Additional experiments examine memory, adaptive search allocation, and alternative optimizers, while Appendix N.1 provides a continuous workflow example.

3.1 Block-Based Representation for ADMET Optimization

We compare block-based and raw-SMILES inputs for lead optimization. Each model receives the same underlying molecule, its predicted ADMET profile, and the same objective: improve the target property while preserving structural similarity. Tables 1 and 8 use one iteration and 40 candidate variants per model, without cross-iteration memory or adaptive search allocation. Within-backbone comparisons therefore assess the representation under a common generation budget.

Dataset. We use the Withdrawn 2.0 database (Gallo et al., 2024), a curated collection of 602 drugs withdrawn from the market with documented reasons for withdrawal. Among these, 91 are labeled as withdrawn due to DILI (drug-induced liver injury) and 33 due to hERG cardiotoxicity, providing a realistic and clinically grounded benchmark for ADMET optimization. To maximize the reliability of our oracle models as evaluation metrics, we first measure their true positive rate on these samples and restrict evaluation to molecules that are correctly identified by the oracle. This yields 67 out of 91 DILI cases (true positive rate: 73.6%) and all 33 hERG cases (true positive rate: 100%), on which we report optimization results, with no data overlap between these samples and the training DILI and hERG datasets (TDC's).

Baselines. We compare MolLingo against a representative set of frontier LLMs prompted to perform molecular optimization directly on raw SMILES strings: GPT-5.4, Claude-4.6-Sonnet, Gemini-3-Pro, Qwen2-7B-Instruct, and Qwen2-14B-Instruct. These baselines are chosen to span a range of model scales and families, providing a comprehensive picture of how well general-purpose LLMs can perform molecular optimization without block-based representations. The prompt used for all baseline models is provided in Appendix K.

Metrics. We report six quantities: property improvement (%), Tanimoto similarity to the original molecule (%), their joint Imprv×Sim score, QED (drug-likeness), synthetic accessibility (Synth; lower is better), and validity (the percentage of chemically valid proposals). The DILI improvement

formula and the joint-score convention are given in Appendix L.1. The same ADMET predictors provide input profiles and evaluate generated molecules.

Results. Block-based input improves the joint score for all three backbones with paired evaluations (Table 1). For GPT-5.4, improvement increases from 8.096% to 13.028% and similarity from 32.902% to 36.608%, raising the joint score from 266.37 to 476.929. The joint score also increases for Claude-4.6-Sonnet (161.658 to 439.238) and Gemini-3-Pro (-687.093 to 839.797). Raw-SMILES baselines remain competitive: Qwen2-7B achieves the highest mean improvement, 25.170%, while MolLingo with Gemini-3-Pro achieves the highest similarity and joint score. All MolLingo variants, GPT-5.4, and Claude-4.6-Sonnet achieve 100% validity. The hERG results show the same direction of within-backbone joint-score changes (Appendix F).

Table 1: DILI Comparison of model performance across multiple chemical generation metrics. Bold values represent the top-performing model for each metric. Values in parentheses represent the standard deviation (std). (↓) indicates lower is better; (↑) indicates higher is better.

Model	Improvement (%) (↑)	Similarity (%) (↑)	Imprv × Sim (↑)	Drug-likeness (↑)	Synth (↓)	Validity (%) (↑)
GPT-5.4 (OpenAI, 2026)	8.096 (5.723)	32.902 (15.119)	266.37	0.611 (0.188)	3.098 (0.838)	**100**
MolLingo (GPT-5.4 base)	13.028 (7.724)	36.608 (15.576)	476.929	0.597 (0.189)	3.087 (0.854)	**100**
Claude-4.6-Sonnet (Anthropic, 2026)	9.543 (5.079)	16.940 (22.726)	161.658	**0.645** (**0.188**)	2.877 (0.871)	**100**
MolLingo (Claude-4.6-Sonnet base)	10.185 (7.365)	43.126 (15.705)	439.238	0.560 (0.086)	3.185 (0.336)	**100**
Gemini-3-Pro (Google DeepMind, 2025)	-46.611 (13.428)	14.741 (19.514)	-687.093	**0.645** (**0.201**)	2.884 (0.670)	97.711
MolLingo (Gemini-3-Pro base)	18.451 (10.326)	**45.515 (16.534)**	**839.797**	0.622 (0.073)	3.166 (0.349)	**100**
Qwen2-7B-Instruct (Qwen, 2024)	**25.170** (**14.098**)	15.998 (5.918)	402.67	0.616 (0.129)	2.560 (0.348)	85.276
Qwen2-14B-Instruct (Qwen, 2024)	22.537 (16.858)	28.249 (18.611)	636.648	0.561 (0.138)	**2.502** (**0.415**)	86.667

3.2 Hit-to-Lead: Context-Aware Fragment Growing

We compare docking-guided fragment growing with direct LLM prompting from the same hit fragment and target protein. Direct baselines receive fragment SMILES and target identity; MolLingo first docks the hit, extracts nearby residues and available volume, and conditions generation on these outputs and attachment-site information (§2.3.2). This evaluates the complete docking-guided workflow, since the generation prompts contain different information. A second comparison holds that workflow fixed and substitutes full growing-fragment SMILES for BFE-block SMILES to assess the representation used for generation.

Dataset & Metrics. We evaluate on a diverse set of 30 protein targets representing distinct disease areas, spanning a broad range of therapeutic indications and protein families. The targets span multiple protein families and binding-site environments. The full list of protein targets is provided in Table 11. We report five metrics: (i) Validity; (ii) Docking Score Improvement (%), the relative improvement in docking score compared to the original hit; (iii) Sim; (iv) Drug-likeness; (v) Synth.

Results. MolLingo achieves mean docking-score improvements of 10.441% with BFE-block SMILES and 9.872% with full fragment SMILES, compared with 2.437% for direct GPT-5.4 prompting (Table 2). Both docking-guided variants achieve 100% validity. Their mean improvements differ by only 0.569 percentage points. The small gap between the two molecular encodings is expected in this setting because the starting hits are typically fragment-sized molecules with relatively simple structures, so the full SMILES of a hit is often already close in granularity to a single BFE block. The primary contribution of this hit-to-lead task is therefore the docking-derived structural context supplied to the LLM: pocket geometry and residue-level interaction information identify where and how the molecule can be modified, enabling more targeted edits than direct prompting alone. Thus, the substantial improvement observed with both encodings primarily supports the value of structure-guided molecular editing, while the choice of representation has a smaller effect in this fragment-growing regime.

3.3 Additional Evaluations

The benchmarks in Sections 3.1 and 3.2 use minimal Chemist-Agent-only configurations, without cross-iteration memory or adaptive orchestration, to isolate molecular representation and docking-guided context from iterative agentic components. Appendices L.2 and L.3 present ablations of shared memory and the Orchestrator, respectively. In the two-iteration DILI experiment, adding memory increases similarity from 30.8% to 52.9%, while also increasing mean improvement from 33.5% to 35.2%. Search traces further show that memory eliminates exact repeated edits (from 16.7% without memory to 0%), helps the agent move away from modifications contradicted by

Table 2: Hit-to-lead workflow comparison from the same starting hit and target. MolLingo derives pocket context by docking before generation; direct baselines receive fragment SMILES and target identity without this context. The two MolLingo rows share the docking-guided workflow and differ in the SMILES supplied for growing. This separates the within-workflow representation comparison from the broader workflow comparison.

Method	Validity (%) (↑)	Docking score improvement (%) (↑)	Similarity (%) (↑)	Drug-likeness (↑)	Synth (↓)
MolLingo (GPT-5.4 base) BFE-block SMILES	**100**	**10.441(4.997)**	65.190 (3.605)	0.630 (0.076)	3.343 (0.179)
MolLingo (GPT-5.4 base) Fragment SMILES	**100**	9.872 (6.381)	69.124 (7.023)	0.662 (0.063)	2.912 (0.188)
GPT-5.4 (OpenAI, 2026) Fragment SMILES	98.67	2.437 (2.618)	**75.871(8.982)**	0.769 (0.062)	**2.402(0.178)**
GPT-4.1 (OpenAI, 2025)	97.33	2.327 (2.805)	73.629 (8.375)	0.780 (0.060)	2.867 (0.164)
Claude-4.6-Sonnet (Anthropic, 2026)	**100**	4.308 (4.894)	16.527 (1.304)	0.864 (0.065)	2.846 (0.327)
Gemini-3-Pro (Google DeepMind, 2025)	94	-1.361 (1.135)	14.103 (3.707)	**0.878(0.072)**	3.104 (0.223)
Qwen2-14B-Instruct (Qwen, 2024)	**100**	4.874 (2.723)	20.471 (4.120)	0.774 (0.081)	2.960 (0.197)
Qwen2-7B-Instruct (Qwen, 2024)	**100**	4.353 (2.756)	20.010 (4.121)	0.758 (0.080)	2.935 (0.193)

subsequent oracle outcomes, and enables it to revisit promising sites with alternative substitutions. These findings suggest that memory supports history-aware refinement rather than merely forwarding intermediate results. With oracle-augmented memory in both conditions, adaptive orchestration increases the joint score from 1764.9 to 2812.2 while maintaining similar structural similarity.

We also compare alternative optimizers under a matched budget of 16 candidate oracle calls per molecule (Appendix M). Graph-GA achieves a mean DILI improvement of 20.2%, compared with 33.5% for named-block MolLingo, while preserving higher structural similarity. This result reflects a different improvement–similarity trade-off rather than uniform superiority. Moreover, Graph-GA generates candidates through random molecular edits and relies on oracle scores to select among them; without a scoring signal, it cannot direct the search toward the target property or provide a property-conditioned rationale for individual modifications. REINVENT and SELFIES achieve larger property improvements but substantially lower similarity, highlighting the distinction between unconstrained property optimization and local analog optimization.

4 Related Work

Agentic Systems for Chemistry. PharmAgents (Gao et al., 2025), M^{4}olGen (Li et al., 2026b), MADD (Solovev et al., 2025), and Tippy (Fehlis et al., 2025) organize agents around stages of molecular design, including target identification, generation, and property evaluation. Tool-augmented systems such as ChemCrow (Bran et al., 2024) and Coscientist (Boiko et al., 2023) likewise combine LLMs with chemistry tools. MolLingo studies block-based editing and docking-guided generation within a shared-memory architecture. Its optimization-stage ablations evaluate memory and search allocation, rather than establishing that multiple agents outperform a single model with the same tools.

Block-Based Molecular Representations. mCLM (Edwards et al., 2025) defines building blocks using three synthetic coupling reactions. FragGPT (Yue et al., 2024) and Trio (Ji et al., 2025) use BRICS-derived fragment sequences. GraphBPE (Shen & Póczos, 2024) learns recurring graph units through BPE-style merging, FragmentNet (Samanta et al., 2025) learns an adaptive graph fragmentation scheme, and FragBERTa (Kaushal & Penmatsa, 2026) uses SAFE sequences obtained through rule-based fragmentation. BFE instead uses corpus-level frequencies to choose among candidate BRICS decompositions. The selected blocks can be supplied to an LLM without backbone fine-tuning. The BRICS and naming ablations assess specific aspects of this design rather than fragmentation as a general concept.

5 Conclusion and Limitations

We introduced **MolLingo**, a multi-agent framework for molecular design that combines block-based molecular representations, tool-grounded reasoning, and iterative search. Across property optimization, docking-guided optimization, and memory ablations, MolLingo shows that chemically aligned inputs, structural context, and agentic coordination can jointly improve LLM-guided molecular design.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Anthropic. Claude Sonnet 4.6 System Card. Technical report, Anthropic, 2026. URL `https://www.anthropic.com/system-cards`.

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

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1): 235–242, 2000.

Yuemin Bian and Xiang-Qun Xie. Computational fragment-based drug design: current trends, strategies, and applications. *The AAPS journal*, 20(3):59, 2018.

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

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

A. Bondi. van der waals volumes and radii. *The Journal of physical chemistry*, 68(3):441–451, 1964. doi: 10.1021/j100785a001.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024. doi: 10.1038/s42256-024-00832-8.

Darko Butina. Unsupervised data base clustering based on daylight's fingerprint and tanimoto similarity: A fast and automated way to cluster small and large data sets. *Journal of Chemical Information and Computer Sciences*, 39(4):747–750, 1999.

Miles Congreve, Robin Carr, Chris Murray, and Harren Jhoti. A'rule of three'for fragment-based lead discovery? *Drug discovery today*, 8(19):876–877, 2003.

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

Carl Edwards, Chi Han, Gawon Lee, Thao Nguyen, Bowen Jin, Chetan Kumar Prasad, Sara Szymkuc, Bartosz A Grzybowski, Ying Diao, Jiawei Han, et al. mclm: A function-infused and synthesis-friendly modular chemical language model. *arXiv preprint arXiv:2505.12565*, 2025.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):8, 2009.

Yao Fehlis, Charles Crain, Aidan Jensen, Michael Watson, James Juhasz, Paul Mandel, Betty Liu, Shawn Mahon, Daren Wilson, Nick Lynch-Jonely, et al. Accelerating drug discovery through agentic ai: A multi-agent approach to laboratory automation in the dmta cycle. *arXiv preprint arXiv:2507.09023*, 2025.

Kathleen Gallo, Andrean Goede, Oliver-Andreas Eckert, Bjoern-Oliver Gohlke, and Robert Preissner. Withdrawn 2.0—update on withdrawn drugs with pharmacovigilance data. *Nucleic Acids Research*, 52(D1):D1503–D1507, 2024.

Bowen Gao, Bo Qiang, Haichuan Tan, Yinjun Jia, Minsi Ren, Minsi Lu, Jingjing Liu, Wei-Ying Ma, and Yanyan Lan. Drugclip: Contrastive protein-molecule representation learning for virtual screening. *Advances in Neural Information Processing Systems*, 36:44595–44614, 2023.

Bowen Gao, Yanwen Huang, Yiqiao Liu, Wenxuan Xie, Wei-Ying Ma, Ya-Qin Zhang, and Yanyan Lan. Pharmagents: Building a virtual pharma with large language model agents. *arXiv preprint arXiv:2503.22164*, 2025.

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

Google DeepMind. Gemini 3 Pro Model Card. Technical report, Google DeepMind, 2025. URL https://deepmind.google/models/model-cards/gemini-3-pro/.

Taicheng Guo, Kehan Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. What can large language models do in chemistry? a comprehensive benchmark on eight tasks. *Advances in neural information processing systems*, 36:59662–59688, 2023.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *arXiv preprint arXiv:2102.09548*, 2021.

Junkai Ji, Zhangfan Yang, Dong Xu, Ruibin Bai, Jianqiang Li, Tingjun Hou, and Zexuan Zhu. Toward closed-loop molecular discovery via language model, property alignment and strategic search. *arXiv preprint arXiv:2512.09566*, 2025.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Neerav Kaushal and Ajay MNV Penmatsa. Fragberta: a fragment-aware molecular representation model with sequential attachment-based fragment embeddings. *Machine Learning: Science and Technology*, 7:045046, 2026. doi: 10.1088/2632-2153/ae911d.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Jiatong Li, Junxian Li, Yunqing Liu, Dongzhan Zhou, and Qing Li. Tomg-bench: Evaluating llms on text-based open molecule generation. *arXiv preprint arXiv:2412.14642*, 2024.

Xuan Li, Zhanke Zhou, Zongze Li, Jiangchao Yao, Yu Rong, Lu Zhang, and Bo Han. RePO: Reference-guided policy optimization for molecular optimization via llm reasoning. In *International Conference on Learning Representations*, 2026a.

Yizhan Li, Florence Cloutier, Sifan Wu, Ali Parviz, Boris Knyazev, Yan Zhang, Glen Berseth, and Bang Liu. M4olGen: Multi-agent, multi-stage molecular generation under precise multi-property constraints. *arXiv preprint arXiv:2601.10131*, 2026b.

Thomas J Lynch, Daphne W Bell, Raffaella Sordella, Sarada Gurubhagavatula, Ross A Okimoto, Brian W Brannigan, Patricia L Harris, Sara M Haserlat, Jeffrey G Supko, Frank G Haluska, et al. Activating mutations in the epidermal growth factor receptor underlying responsiveness of non–small-cell lung cancer to gefitinib. *New England Journal of Medicine*, 350(21):2129–2139, 2004.

Thao Nguyen, Kuan-Hao Huang, Ge Liu, Martin D Burke, Ying Diao, and Heng Ji. Farm: Functional group-aware representations for small molecules. *arXiv preprint arXiv:2410.02082*, 2024.

OpenAI. Introducing GPT-4.1 in the api, 2025. URL https://openai.com/index/gpt-4-1/.

OpenAI. GPT-5.4 Model. OpenAI API documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.4.

Team Qwen. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. URL https://arxiv.org/abs/2407.10671.

Ankur Samanta, Rohan Gupta, Aditi Misra, Christian McIntosh Clarke, and Jayakumar Rajadas. Fragmentnet: Adaptive graph fragmentation for graph-to-sequence molecular representation learning. *arXiv preprint arXiv:2502.01184*, 2025.

Yuchen Shen and Barnabás Póczos. Graphbpe: Molecular graphs meet byte-pair encoding. *arXiv preprint arXiv:2407.19039*, 2024.

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

Gleb V Solovev, Alina B Zhidkovskaya, Anastasia Orlova, Nina Gubina, Anastasia Vepreva, Rodion Golovinskii, Ilya Tonkii, Ivan Dubrovsky, Ivan Gurev, Dmitry Gilemkhanov, et al. Madd: Multi-agent drug discovery orchestra. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 6956–6998, 2025.

Taffee T Tanimoto. Elementary mathematical theory of classification and prediction. 1958.

Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

Atabey Ünlü, Elif Çevrim, Melih Gökay Yiğit, Ahmet Sarıgün, Hayriye Çelikbilek, Osman Bayram, Deniz Cansen Kahraman, Abdurrahman Olğaç, Ahmet Sureyya Rifaioglu, Erden Banoğlu, et al. Target-specific de novo design of drug candidate molecules with graph-transformer-based generative adversarial networks. *Nature Machine Intelligence*, 7(9):1524–1540, 2025.

Ziqing Wang and Kaize Ding. Remol: Llm-guided molecular optimization with reinforcement learning. In *KDD 2025 PhD Consortium*, 2025. URL https://kdd2025.kdd.org/wp-content/uploads/2025/07/paper_20.pdf.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Jie Yue, Bingxin Peng, Yu Chen, Jieyu Jin, Xinda Zhao, Chao Shen, Xiangyang Ji, Chang-Yu Hsieh, Jianfei Song, Tingjun Hou, et al. Unlocking comprehensive molecular design across all scenarios with large language model and unordered chemical language. *Chemical Science*, 15(34):13727–13740, 2024.

APPENDIX CONTENTS

A	Background: Early-Stage Drug Discovery Pipeline	13
B	BRICS-Based Fragment Enumeration (BFE)	13
C	Fragment Library for Hit Identification	18
D	ADMET Oracle Models and Ranking Function	19
E	Hit-to-Lead: Docking-Guided Hotspot Identification and LLM-Based Fragment Growing	19
F	Benchmarking Results	20
G	Literature and Database Retrieval for Target Summarization	24
H	Shared Memory Module	24
I	Orchestrator Agent: Decision-Making and Coordination	26
J	MolLingo's Toolset	27
K	Implementation Details	28
L	Optimization Ablations	31
M	Alternative Representations and Optimizers	33
N	Use Cases	34
O	Evaluation Limitations	36

The appendices provide method details (Appendices B–E), supplementary benchmarks (Appendix F), agent and implementation specifications (Appendices H–K), a worked case study (Appendix N), and controlled optimization experiments (Appendices L and M). Evaluation limitations are collected in Appendix O.

A BACKGROUND: EARLY-STAGE DRUG DISCOVERY PIPELINE

Drug discovery is a lengthy and costly process spanning from initial biological hypothesis to clinical approval. The early computational stages, often referred to as the dry lab phase, are critical for narrowing down the vast chemical space to a small set of promising candidates before any wet lab experimentation begins. As illustrated in Fig. 4, this phase proceeds through five sequential stages. Target identification establishes the biological rationale by linking a disease phenotype to a specific protein or pathway whose modulation may yield therapeutic benefit. Hit identification screens large chemical libraries to find initial compounds (referred to as hits) that bind to the target with detectable affinity. Hit-to-lead progression elaborates the most promising hits through structural modifications to improve potency and selectivity while maintaining drug-like properties. Lead optimization further refines the lead compound to achieve a favorable balance of efficacy, selectivity, and ADMET properties. Finally, candidate selection identifies the molecule best suited for advancement into preclinical and clinical studies, based on its overall profile across all evaluated criteria. MolLingo is designed to automate and accelerate each of these computational stages through LLM-guided reasoning and multi-agent coordination, as described in the main text.

B BRICS-BASED FRAGMENT ENUMERATION (BFE)

VOCABULARY CONSTRUCTION

We construct the vocabulary from 20M drug-like molecules drawn from ChEMBL, ZINC, and the industrial fragment and screening libraries introduced in Nguyen et al. (2024). BRICS (Degen et al., 2008) identifies synthesis-aware candidate cut sites, and the resulting fragments serve as primitive units rather than individual atoms. These rules provide a chemical basis for fragmentation, but do not establish synthetic accessibility for every vocabulary entry or generated candidate.

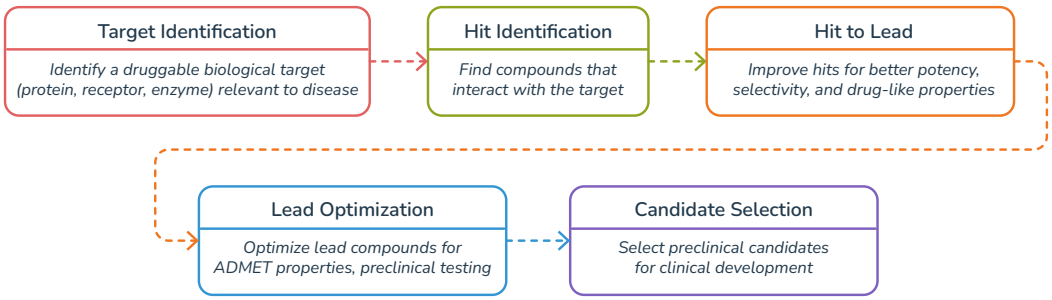


Figure 4: Overview of the early-stage computational drug discovery pipeline, covering the dry lab phases from target identification through candidate selection. Each stage progressively narrows the chemical space from thousands of initial hits to a single optimized drug candidate, guided by increasingly stringent criteria for potency, selectivity, and drug-likeness. MolLingo automates this end-to-end process through coordinated multi-agent reasoning, operating entirely in silico prior to experimental validation.

THE BFE ALGORITHM

Unlike text BPE, which operates on linear character sequences, molecular graphs present two key challenges for direct BPE application: (i) fragments can have more than two neighbors, making it impossible to represent branched molecules as a flat sequence of blocks without auxiliary notation; and (ii) merging two molecular graphs programmatically is expensive (see Proposition 1), making iterative graph-based vocabulary construction costly at corpus scale.

We invert the BPE construction logic: rather than iteratively merging fragments, we enumerate building blocks by considering every pair in the BRICS bond set augmented with virtual terminal boundaries. Lemma 1 establishes completeness for contiguous blocks in a path of primitive units; it is not a completeness claim for arbitrary branched subgraphs. For sequential blocks A-B-C-D, boundary-pair enumeration recovers $\{A, B, C, D, AB, BC, CD, ABC, BCD, ABCD\}$. The implementation performs each candidate break before discarding fragmentations that fail the branch check (Algorithm 1). This filter affects the accepted fragments, not the number of pairs processed. The path-based completeness result and the more general enumeration-cost bound are stated separately below.

Lemma 1 (Enumeration recovers all contiguous blocks in a path)**.** *Under the no-branch assumption, the adjacency graph of the primitive blocks in M is a path. Any contiguous sub-sequence $[b_i, \ldots, b_j]$ is a valid block and can be obtained by breaking exactly two bonds, where terminal boundaries of the molecule are treated as virtual bonds.*

Proof. Let $M = [b_1, b_2, \ldots, b_n]$. Augment the bond set by introducing two virtual terminal bonds: a left boundary bond (b_0, b_1) and a right boundary bond (b_n, b_{n+1}), where b_0 and b_{n+1} are sentinel nodes. Define the augmented bond set:

$$\tilde{B} = \{(b_0, b_1)\} \cup \{(b_k, b_{k+1})\}_{k=1}^{n-1} \cup \{(b_n, b_{n+1})\}$$

Then any contiguous sub-sequence $[b_i, \ldots, b_j]$ is recovered by breaking exactly two bonds from $\tilde{B}$:

$$S = \{(b_{i-1}, b_i),\ (b_j, b_{j+1})\}, \quad |S| = 2 \quad \forall\, 1 \le i \le j \le n.$$

Since every choice of $1 \le i \le j \le n$ yields a distinct contiguous sub-sequence via exactly this bond pair, the enumeration over all $S \subseteq \tilde{B}$ with $|S| = 2$ is complete. □

The virtual terminal bonds are bookkeeping boundaries, not a mechanism for handling branching. They allow terminal subsequences, including the full path, to be specified by two boundary choices even when fewer than two physical bonds must be cut. Branching is handled separately by the post-break discard step in Algorithm 1.

Proposition 1 (Per-molecule computational cost)**.** *Let M have n primitive blocks and let $m = |\tilde{B}|$ be its augmented candidate-bond count. Under the operation-cost model used here, assume $m =$*

$O(n)$ and that the graph processed per operation has size $O(n)$. Let $C_{break} = O(n)$ include graph copying, bond breaking, component extraction, and the subsequent branch check. Let $C_{merge} = O(n^2)$ be the assumed merge-and-sanitize cost. The exact count $m = n + 1$ holds for the path construction in Lemma 1; the asymptotic BFE upper bound below does not require path topology.

(i) ***BFE:***

$$T_{BFE} = \binom{m}{2} \cdot C_{break} = O(n^2) \cdot O(n) = O(n^3).$$

(ii) ***Graph BPE:***

$$T_{GraphBPE} = \frac{n(n-1)}{2} \cdot C_{merge} = \frac{n(n-1)}{2} \cdot O(n^2) = O(n^4).$$

Proof. **(i) BFE.** Algorithm 1 constructs every two-element subset of $\tilde{B}$ and breaks the molecule before checking for branching. It therefore processes $\binom{m}{2}$ candidates regardless of how many fragmentations are subsequently discarded. For the path case, $m = n+1$ and this is exactly $n(n+1)/2$; for general inputs satisfying $m = O(n)$, the count is $O(n^2)$. With the stated $O(n)$ per-candidate processing cost,

$$T_{\text{BFE}} = \binom{m}{2} \cdot O(n) = O(n^3).$$

This is an enumeration-cost upper bound, independent of the no-branch assumption used to establish path-block completeness.

(ii) Graph BPE. Starting from n blocks, at each merge step, the most frequent adjacent pair across the corpus is selected and collapsed into a single new block, reducing the block count by one. For a single molecule, this process repeats until all blocks are merged, yielding $\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$ merge operations in the worst case. Each operation involves `CombineMols`, which scales linearly in the number of atoms and bonds, followed by bond formation and mandatory sanitization. Assuming sanitization dominates (due to, e.g., ring perception) we take $C_{\text{merge}} = O(n^2)$. Therefore:

$$T_{\text{GraphBPE}} = \frac{n(n-1)}{2} \cdot O(n^2) = O(n^4).$$

□

Corollary 1 (Modeled per-molecule cost ratio)**.** *Instantiate the operation-cost model as $\widehat{C}_{break} = c_b n$ and $\widehat{C}_{merge} = c_m n^2$, with positive constants c_b and c_m. For the path-case action counts and $n > 1$,*

$$\frac{\widehat{T}_{GraphBPE}}{\widehat{T}_{BFE}} = \frac{\frac{n(n-1)}{2} c_m n^2}{\frac{n(n+1)}{2} c_b n} = \frac{c_m}{c_b} \frac{n-1}{n+1} n = \Theta(n).$$

This is a prediction of the specified cost model, not a guaranteed runtime ratio inferred by dividing asymptotic upper bounds.

Corollary 2 (Corpus-level cost)**.** *Let n_M be the primitive-block count of molecule M in corpus $\mathcal{C}$. Under the full-pass graph-merging model, $V - n_0$ corpus passes grow an initial vocabulary of size n_0 to size V, while BFE uses one pass. Summing the per-molecule upper bounds gives*

$$T_{BFE}^{corpus} = O\left(\sum_{M \in \mathcal{C}} n_M^3\right),$$

$$T_{GraphBPE}^{corpus} = O\left((V - n_0) \sum_{M \in \mathcal{C}} n_M^4\right).$$

For a size-homogeneous corpus with mean block count $\bar{n}$ and $n_M = \Theta(\bar{n})$, these reduce to $O(|\mathcal{C}|\bar{n}^3)$ and $O((V-n_0)|\mathcal{C}|\bar{n}^4)$. Under the instantiated cost model in Corollary 1, the corresponding modeled ratio scales as $\Theta((V - n_0)\bar{n})$. For heterogeneous molecule sizes, the sums must be retained rather than replacing averages of powers with powers of an average.

Empirical timing. Table 3 reports the merge-to-break time ratio for 300 randomly sampled molecules at each of three heavy-atom counts. No branching filter was applied. The approximately linear increase is consistent with the modeled operation-cost trend, but these measurements do not establish an asymptotic bound or measure the full corpus-level speedup. Heavy-atom count is the empirical size measure, distinct from the primitive-block count in the enumeration analysis.

Table 3: Empirical merge-to-break time ratios. Each size group contains 300 randomly sampled molecules, with no filtering by branching.

Heavy atoms	Molecules	Mean time ratio
10	300	11.84×
15	300	15.22×
20	300	20.35×

Algorithm 1 Phase 1 — Vocabulary Construction

Require: Corpus $\mathcal{C}$ of 20M molecules (ChEMBL, ZINC, screening and fragment libraries)
Ensure: Fragment frequency dictionary `vocab`

```
vocab ← ∅
for each molecule mol ∈ C do
    B̃ ← BRICS_BONDS(mol) + virtual terminal bonds
    bond_sets ← {S ⊆ B̃ | |S| = 2}
    for each S ∈ bond_sets do
        fragments ← BREAK_MOLECULE(mol, S)
        if HAS_BRANCH(fragments) then
            continue
        end if
        for block ∈ fragments do
            vocab[block] ← vocab[block] + 1
        end for
    end for
end for
return vocab
```

Molecular Fragmentation

Given a molecule, we decompose it into a sequence of building blocks from the learned vocabulary using `FragmentOnBonds`. For each molecule, we enumerate all valid non-branching decompositions by breaking all possible pairs of BRICS bonds, as in vocabulary construction. Each resulting decomposition is a candidate fragmentation: a sequence of blocks $[t_1, t_2, \ldots, t_k]$ where each t_i belongs to the vocabulary.

Among all candidate decompositions, we select the optimal one hierarchically. We first prioritize decompositions with the fewest fragments (i.e., longest blocks), as longer blocks carry richer structural and functional information. Among candidates of equal length (i.e., number of blocks), we retain only those whose blocks all appear in the vocabulary with frequency above a minimum threshold $f_{\min}$. If multiple candidates satisfy this condition, we select the one with the lowest standard deviation of block frequencies:

$$t^* = \arg\min_{t \in \mathcal{T}^*} \text{std}\left(\{freq(t_i)\}_{i=1}^{k}\right)$$

where $\mathcal{T}^*$ is the set of valid candidates with the fewest fragments and all block frequencies above $f_{\min}$. In our experiments, $f_{\min}$ is set to 20, yielding a final vocabulary of approximately 1.2M building blocks. This criterion favors decompositions whose blocks are uniformly well-represented in the vocabulary, avoiding decompositions that rely on rare or potentially noisy fragments. The pseudocode for molecular fragmentation is provided in Algorithm 2.

Table 4: Computational costs under the stated operation-cost model. n denotes primitive blocks per molecule, $m = |\tilde{B}| = O(n)$ the augmented bond count, n_M the block count of molecule M, V the target vocabulary size, and n_0 the initial vocabulary size. Sums range over the corpus. The final row uses the instantiated path-case cost model and a size-homogeneous corpus.

	# Actions	Action cost	Per-molecule	Corpus passes	Total
BFE	$\binom{m}{2}$	$O(n)$	$O(n^3)$	1	$O(\sum_M n_M^3)$
Graph BPE	$\frac{n(n-1)}{2}$	$O(n^2)$	$O(n^4)$	$V - n_0$	$O((V-n_0)\sum_M n_M^4)$
Modeled ratio	$\frac{n-1}{n+1}$	$(c_m/c_b)n$	$\Theta(n)$	$V - n_0$	$\Theta((V-n_0)\bar{n})$

Algorithm 2 Phase 2 — Molecular Fragmentation

Require: Molecule mol, vocabulary `vocab`, threshold `min_freq`
Ensure: Sequence of building blocks $[t_1, t_2, \ldots, t_k]$

$\tilde{B} \leftarrow$ BRICS_BONDS(mol) + virtual terminal bonds
`candidates` $\leftarrow [\,]$
▷ Enumerate all valid (non-branching) decompositions
for each $S \subseteq \tilde{B}$ such that $|S| = 2$ **do**
    `fragments` $\leftarrow$ BREAK_MOLECULE(mol, S)
    if not HAS_BRANCH(`fragments`) **then**
        `candidates`.APPEND(`fragments`)
    end if
end for
▷ Prefer fewer fragments (longer blocks)
SORT(`candidates`, key = LEN, order = ASCENDING)
for `fragmentation` $\in$ `candidates` **do**
    `freqs` $\leftarrow$ [`vocab`$[t]$ **for each** $t \in$ `fragmentation`]
    if $\forall f \in$ `freqs`, $f \geq$ `min_freq` **then**
        `valid` $\leftarrow \{c \in$ `candidates` $\mid |c| = |$`fragmentation`$|\}$
        return $\arg\min_{c \in \texttt{valid}}$ STD(`vocab`$[t], t \in c$)
    end if
end for
▷ Fallback: finest-grained decomposition
return $\arg\min_{c \in \texttt{candidates}}$ LEN(c)

BFE VS. BRICS FRAGMENTATION

To isolate the contribution of BFE's frequency-guided optimal decomposition selection from a naive BRICS decomposition baseline, we compare MolLingo using BFE fragmentation against a variant that simply breaks molecules at all possible BRICS bonds without vocabulary-guided selection (denoted BRICS). Both variants use Gemini-3-Pro as the base LLM and are evaluated on the DILI benchmark.

Table 5: Ablation comparing BFE fragmentation against naive BRICS fragmentation, using Gemini-3-Pro as the base LLM on the DILI benchmark. Values in parentheses denote standard deviation.

Model	Improvement (%) (↑)	Similarity (%) (↑)	Imprv×Sim (↑)	Drug-likeness (↑)	Synth (↓)	Validity (%) (↑)
MolLingo (Gemini-3-Pro)	**18.451 (10.326)**	**45.515 (16.534)**	**839.797**	0.622 (0.073)	3.166 (0.349)	**100**
BRICS (Gemini-3-Pro)	1.034 (8.274)	78.650 (15.761)	81.316	**0.642 (0.058)**	**2.934 (0.341)**	**100**

The naive BRICS baseline preserves higher similarity (78.7% versus 45.5%), while BFE achieves higher mean improvement (18.5% versus 1.0%) and a higher joint score (839.8 versus 81.3). This supports frequency-guided selection relative to the specified BRICS baseline. It does not isolate fragment size, fragment count, and frequency coverage as separate causes of the difference.

C Fragment Library for Hit Identification

Fragment-Based Drug Design

Fragment-based drug design (FBDD) has emerged as a powerful paradigm for early-stage drug discovery, complementing high-throughput screening by exploring chemical space more efficiently with smaller, simpler molecules (Bian & Xie, 2018). Rather than screening large, complex compounds, FBDD begins with low-molecular-weight fragments (typically < 300 Da) that bind weakly but efficiently to the target, and progressively elaborates them into drug-like leads through fragment growing, merging, or linking (Bian & Xie, 2018).

The quality of a fragment library is critical to the success of FBDD. Fragments should satisfy the "rule of three" (Congreve et al., 2003) (molecular weight ≤ 300 Da, cLogP ≤ 3, hydrogen bond donors ≤ 3, hydrogen bond acceptors ≤ 3), exhibit favorable ADMET properties to ensure downstream drug-likeness, and provide broad coverage of diverse chemical scaffolds to maximize the probability of identifying a hit against a novel target. Scaffold diversity is particularly important: a library enriched with structurally redundant fragments offers little advantage over a smaller one, as redundant fragments tend to bind the same pockets in the same way.

MolLingo Fragment Library

We construct a curated fragment library by aggregating commercially available fragment collections from ChemDiv[1], Enamine[2], ChemBridge[3], Life Chemicals[4], Vitas-M[5], Maybridge[6], Asinex[7], Eximed[8], InterBioScreen[9], Targetmol[10], Princeton BioMolecular[11], Otava[12], and Alinda Chemical[13]. The raw aggregated library contains **727,519** fragments spanning **109,778** unique Murcko scaffolds (Bemis & Murcko, 1996), providing broad coverage of drug-like chemical space.

We filter the library using the aggregate ADMET score and QED, retaining fragments that satisfy:

$$\text{ADMET score} > 2.5 \quad \text{and} \quad \text{QED score} > 0.7$$

This yields a high-quality subset of **12,337** fragments that balance broad scaffold diversity with favorable drug-like and pharmacokinetic properties. These filtered fragments constitute the screening library used by the Chemist Agent during hit identification (Section 2.3.1). Table 6 summarizes the library statistics.

Table 6: Summary statistics of the MolLingo fragment library.

Property	**Value**
Total fragments	727,519
Unique scaffolds	109,778
Fragments passing ADMET & QED filter	12,337
ADMET threshold	> 2.5
QED threshold	> 0.7

[1] https://www.chemdiv.com/
[2] https://enamine.net/
[3] https://chembridge.com/
[4] https://lifechemicals.com/
[5] https://vitasmlab.biz/
[6] https://chembridge.com/
[7] https://www.asinex.com/
[8] https://eximedlab.com/
[9] https://www.ibscreen.com/
[10] https://www.targetmol.com/
[11] https://princetonbio.com/
[12] https://www.otava.com/
[13] https://www.alinda.ru/synthes_en.htm

D ADMET ORACLE MODELS AND RANKING FUNCTION

ADMET ORACLE MODELS

Accurate prediction of ADMET properties is essential for guiding molecule optimization toward drug-like candidates and filtering out compounds with unfavorable pharmacokinetic or toxicity profiles early in the discovery process. We train a suite of ADMET oracle models on tasks from the Therapeutics Data Commons (TDC) (Huang et al., 2021) benchmark, selecting tasks where data quality and quantity are sufficient to build reliable predictive models, as indicated by their leaderboard performance. Our selection follows the protocol of mCLM (Edwards et al., 2025), covering six key ADMET endpoints that collectively assess toxicity, absorption, and permeability: DILI (Drug-Induced Liver Injury), hERG cardiotoxicity, AMES mutagenicity, HIA (Human Intestinal Absorption), Pgp (P-glycoprotein inhibition), and BBB (Blood-Brain Barrier Penetration). Model architectures and training procedures are adopted from mCLM (Edwards et al., 2025), yielding similar results.

ADMET SCORE AND RANKING FUNCTION

Given the predicted probabilities from each oracle model, we define the ADMET score of a molecule m as the sum of its favorable outcome probabilities across all tasks:

$$\text{ADMET}(m) = (1 - p_{\text{DILI}}) + (1 - p_{\text{AMES}}) + (1 - p_{\text{hERG}}) + (1 - p_{\text{Pgp}}) + p_{\text{HIA}}$$

where lower predicted probabilities are favored for DILI, AMES, hERG, and Pgp, and higher values are favored for HIA. BBB penetration is excluded because its desirability depends on the therapeutic indication. The aggregate score ranges from 0 to 5; the screening threshold is $\text{ADMET}(m) > 2.5$. This probability-based ranking score is distinct from the relative DILI-logit improvement metric in Appendix L.1. The threshold is a computational selection criterion, not a validated safety standard.

For the construction of the screening fragment library, fragments are filtered by the composite criterion:

$$\text{ADMET}(m) > 2.5 \quad \text{and} \quad \text{QED}(m) > 0.7$$

to restrict the library to fragments meeting the chosen predicted-property and drug-likeness criteria before target-specific screening.

For each protein target, the Chemist Agent first queries ChEMBL for known binders (Gaulton et al., 2012). If no relevant binding data are available, or if the user requests de novo design, the agent screens the curated fragment library. DrugCLIP (Gao et al., 2023) ranks the fragments by protein–ligand embedding similarity, and the top k candidates are then docked against the target structure. The final hits are selected by docking score as starting points for hit-to-lead optimization.

E HIT-TO-LEAD: DOCKING-GUIDED HOTSPOT IDENTIFICATION AND LLM-BASED FRAGMENT GROWING

EVALUATION PROTOCOL

Algorithm 3 derives pocket context and attachment-site inputs from the starting hit and target. Table 2 compares this workflow with direct prompting and separately compares two molecular encodings within the workflow. The BFE-block and full-fragment variants share the docking and attachment-site inputs; their mean docking improvements differ by 0.569 percentage points. No significance or equivalence test is reported. The experiment does not separately ablate pocket geometry, residue identities, attachment-site information, and the remaining generation instructions.

ALGORITHM

The pipeline comprises three phases: docking the hit, identifying candidate growth sites from the pose, and generating fragment proposals conditioned on each site's estimated volume and neighboring residues. The docking pose and resulting context are computational estimates.

Algorithm 3 Docking-Guided Hotspot Identification and LLM-Based Fragment Growing

Require: Protein target P, hit fragment F, contact threshold $d_c = 7.0$ Å, maximum hotspots $k = 5$
Ensure: Ranked hotspots $\mathcal{H}$ and proposed fragment growths $\{\mathcal{B}_i\}_{i=1}^{k}$

Phase 1: Molecular Docking

Dock fragment F into the binding site of protein P
Save the docking pose as receptor structure R and docked ligand structure L

Phase 2: Hotspot Identification

Parse heavy atoms (non-hydrogen) from R and L, yielding $\mathcal{A}_R$ and $\mathcal{A}_L$
Build spatial index $\mathcal{N}$ over $\mathcal{A}_R$
for each heavy atom $a_i \in \mathcal{A}_L$ **do**
    Identify neighboring residues:

$$\mathcal{R}_i = \{r \in R \mid \exists\, a \in r : \|a_i - a\| \le d_c\}$$

    Estimate available volume around a_i
    Construct cubic grid $\mathcal{G}_i$ centered at a_i (edge 5.0 Å, resolution 0.5 Å)
    for each grid point $g \in \mathcal{G}_i$ **do**
        Compute:

$$d_R(g) = \min_{a \in \mathcal{A}_R} \|g - a\|, \qquad d_L(g) = \min_{a \in \mathcal{A}_L} \|g - a\|$$

    end for
    Count unoccupied grid points:

$$n_i = \left|\{g \in \mathcal{G}_i \mid d_R(g) > 2.2\ \text{Å} \wedge d_L(g) > 1.2\ \text{Å}\}\right|$$

    Compute available volume:

$$V_i = n_i \times (0.5)^3$$

    Record hotspot:

$$h_i = \big(a_i,\ \text{type}(a_i),\ V_i,\ \mathcal{R}_i\big)$$

end for
Sort $\{h_i\}$ by V_i in descending order
Select top-k hotspots:

$$\mathcal{H} = \{h_1, \ldots, h_k\}$$

Phase 3: LLM-Guided Fragment Growing

for each hotspot $h_i \in \mathcal{H}$ **do**
    Set S_i to the SMILES of the BFE block containing a_i (or full F in the fragment-SMILES variant)
    Construct biological context prompt:

$$\text{prompt}_i = \big(P,\ \text{type}(a_i),\ V_i,\ \mathcal{R}_i,\ S_i\big)$$

    Query LLM to propose building blocks $\mathcal{B}_i$ attachable at a_i
end for
return proposed fragment growths:

$$\{\mathcal{B}_i\}_{i=1}^{k}$$

F Benchmarking Results

Comparison with DrugGEN on Hit Identification

We benchmark MolLingo's hit identification pipeline against DrugGEN (Ünlü et al., 2025), a GAN-based molecular generation model, on two protein targets: CDK2 and AKT1. DrugGEN requires retraining a separate model for each target, making large-scale evaluation impractical; we therefore restrict this comparison to these two well-studied oncology targets for which DrugGEN provides top hit results directly, without the need for retraining on our end.

For a fair comparison, both methods leverage ChEMBL (Gaulton et al., 2012) as the source of known binders for each target, though in different ways: MolLingo uses ChEMBL binders as a hit screening library, ranking and filtering candidates following the procedure described in Appendix D, while DrugGEN uses ChEMBL binders as training data for its target-specific GAN model. For each target, we select the top 30 hits from each method and evaluate them across six metrics: docking score, QED, synthetic accessibility (SA), Lipinski's rule of five, Veber's oral bioavailability criteria, and PAINS filter pass rate.

Table 7 reports the results. MolLingo achieves more favorable docking scores than DrugGEN on both targets: -9.308 versus -8.527 kcal/mol on CDK2 and -6.368 versus -5.797 kcal/mol on AKT1, with lower standard deviations. It also yields higher QED and lower synthetic accessibility scores, while Lipinski, Veber, and PAINS scores are comparable. These computational results are obtained without target-specific retraining, but concern only two well-studied targets and ChEMBL-based hit retrieval. They do not evaluate the DrugCLIP fallback or establish performance on novel or understudied targets.

Table 7: Comparison of MolLingo and DrugGEN on hit identification for CDK2 and AKT1. Results are reported as mean $\pm$ standard deviation over the top 30 hits for each method. Bold values indicate the better performing method for each metric. (↓) indicates lower is better; (↑) indicates higher is better.

	Docking Score (↓)		QED (↑)		SA (↓)		Lipinski (↑)		Veber (↑)		PAINS (↑)	
	MolLingo	**DrugGEN**	**MolLingo**	**DrugGEN**	**MolLingo**	**DrugGEN**	**MolLingo**	**DrugGEN**	**MolLingo**	**DrugGEN**	**MolLingo**	**DrugGEN**
CDK2	$\mathbf{-9.308}$ $\pm\mathbf{0.424}$	-8.527 ±0.714	$\mathbf{0.63}$ $\pm\mathbf{0.095}$	0.518 ±0.111	$\mathbf{2.848}$ $\pm\mathbf{0.393}$	3.058 ±0.487	3.926 ±0.279	$\mathbf{3.955}$ $\pm\mathbf{0.208}$	$\mathbf{2.0}$ $\pm\mathbf{0.0}$	$\mathbf{2.0}$ $\pm\mathbf{0.0}$	$\mathbf{0.913}$ $\pm\mathbf{0.282}$	0.864 ±0.343
AKT1	$\mathbf{-6.368}$ $\pm\mathbf{0.251}$	-5.797 ±0.313	$\mathbf{0.658}$ $\pm\mathbf{0.038}$	0.567 ±0.109	$\mathbf{3.488}$ $\pm\mathbf{0.194}$	3.741 ±0.382	$\mathbf{3.991}$ $\pm\mathbf{0.012}$	3.862 ±0.345	$\mathbf{2.0}$ $\pm\mathbf{0.0}$	1.966 ±0.182	$\mathbf{1.0}$ $\pm\mathbf{0.0}$	$\mathbf{1.0}$ $\pm\mathbf{0.0}$

hERG Optimization Results

Table 8 reports results on 33 oracle-correct hERG-positive withdrawn drugs. GPT-5.4 operating on raw SMILES achieves 1.312% mean improvement and 37.155% similarity, compared with 4.050% and 42.878% for MolLingo with the same backbone. Their joint scores are 48.75 and 173.736, respectively. The paired Claude-4.6-Sonnet and Gemini-3-Pro comparisons likewise favor block-based representation on the joint metric: 10.617 versus 295.467 and -9.040 versus 34.526. MolLingo with Claude-4.6-Sonnet has the highest similarity (54.966%) and joint score, while raw-SMILES Qwen2-7B has the highest mean improvement (6.026%). Qwen2-14B achieves 4.309% improvement with 38.468% similarity, demonstrating that the Qwen2 rows exhibit different property–similarity trade-offs. Only Gemini-3-Pro has negative mean improvement among the raw-SMILES baselines. These results support within-backbone trade-off improvements, not a general inability to optimize using raw SMILES.

Excluded Baselines. Several additional models were considered for benchmarking but excluded for the following reasons. LLaMA-2-7B and Gemini-3.1-Flash did not follow the optimization instructions reliably. In particular, Gemini-3.1-Flash consistently failed to perform structural modifications: instead of proposing modified molecules, it analyzed the input molecule's constituent building blocks and predicted possible properties, without making any changes to the molecular structure. As reliable instruction following is a prerequisite for meaningful comparison, these models were excluded from the benchmark.

We also considered closely related molecular optimization methods, specifically RePO (Li et al., 2026a) and ReMol (Wang & Ding, 2025). ReMol was excluded because no public implementation was available. For RePO, Table 9 reports the published TOMG-Bench results rather than a rerun, because pretrained checkpoints were unavailable at the time of our experiments and training from scratch was beyond scope.

Molecule Optimization on TOMG-Bench

We further evaluate MolLingo's molecule optimization capability on TOMG-Bench (Li et al., 2024), a standardized benchmark for text-guided molecular optimization covering three property objec-

Table 8: hERG Comparison of model performance across multiple chemical generation metrics. Bold values represent the top-performing model for each metric. Values in parentheses represent the standard deviation (std). (↓) indicates lower is better; (↑) indicates higher is better.

Model	Improvement (%) (↑)	Similarity (%) (↑)	Imprv × Sim (↑)	Drug-likeness (↑)	Synth (↓)	Validity (%) (↑)
GPT-5.4 (OpenAI, 2026)	1.312 (2.684)	37.155 (10.921)	48.75	0.509 (0.172)	3.003 (0.658)	**100**
MolLingo (GPT-5.4 base)	4.050 (1.458)	42.878 (11.081)	173.736	0.492 (0.201)	3.276 (0.762)	**100**
Claude-4.6-Sonnet (Anthropic, 2026)	0.480 (0.690)	22.119 (10.982)	10.617	**0.706** (**0.153**)	2.645 (0.466)	93.939
MolLingo (Claude-4.6-Sonnet base)	5.370 (3.228)	**54.966** (**20.248**)	**295.467**	0.549 (0.096)	3.289 (0.323)	**100**
Gemini-3-Pro (Google DeepMind, 2025)	-0.562 (1.396)	16.085 (13.948)	-9.040	0.632 (0.200)	3.119 (0.670)	98.589
MolLingo (Gemini-3-Pro base)	0.675 (3.218)	51.186 (17.258)	34.526	0.553 (0.058)	3.196 (0.277)	**100**
Qwen2-7B-Instruct (Qwen, 2024)	**6.026** (**3.494**)	15.345 (6.879)	92.47	0.583 (0.114)	2.658 (0.489)	75.61
Qwen2-14B-Instruct (Qwen, 2024)	4.309 (3.562)	38.468 (30.794)	165.76	0.579 (0.176)	**2.616** (**0.937**)	89.024

tives: LogP (lipophilicity), QED (drug-likeness), and MR (molecular refractivity). We report three metrics for each objective: Success Rate (SR), molecular Similarity (Sim), and their product SR×Sim, which jointly captures optimization effectiveness and structural conservation.

Table 9 compares MolLingo with general-purpose LLMs, chemistry-specialized models (MolT5 and BioT5), instruction-tuned models (OpenMolIns), and RePO (Li et al., 2026a). Baseline provenance is specified in the caption. MolLingo achieves the highest SR and SR×Sim for all three objectives, and the highest similarity for LogP and MR. Its joint scores are 0.706 for LogP, 0.418 for QED, and 0.752 for MR, compared with RePO's 0.297, 0.236, and 0.294. RePO retains the highest QED similarity. Thus, the joint-score gains do not imply an improvement on every individual metric.

Within the GPT-5.4 comparison, MolLingo achieves higher SR, similarity, and joint scores on all three objectives. This paired result complements the cross-model comparison, whose rankings vary by objective and metric.

Table 9: Performance comparison on TOMG-Bench across LogP, QED, and MR optimization objectives. SR denotes Success Rate, Sim denotes Tanimoto Similarity, and SR×Sim represents their product, jointly capturing optimization effectiveness and structural conservation. Bold values indicate the best performing model for each metric. Except for MolLingo, GPT-5.4, and RePO results, all other results are adopted from Li et al. (2024).

Model Name	LogP			QED			MR		
	SR	Sim	SR×Sim	SR	Sim	SR×Sim	SR	Sim	SR×Sim
MolLingo (GPT-5.4 base)	**0.968**	**0.730**	**0.706**	**0.765**	0.546	**0.418**	**0.998**	**0.753**	**0.752**
GPT-5.4	0.871	0.612	0.533	0.432	0.284	0.123	0.903	0.691	0.624
RePO	0.415	0.715	0.297	0.312	**0.756**	0.236	0.399	0.736	0.294
GPT-4o	0.719	0.659	0.474	0.3952	0.618	0.244	0.6864	0.642	0.44
GPT-4-turbo	0.7662	0.6984	0.535	0.3946	0.6587	0.256	0.7388	0.6821	0.504
GPT-3.5-turbo	0.4048	0.6327	0.256	0.3316	0.5635	0.187	0.412	0.6263	0.258
Claude-3.5	0.797	0.7124	0.568	0.5361	0.7042	0.378	0.6962	0.7112	0.495
Claude-3	0.7984	0.6067	0.484	0.4678	0.5855	0.274	0.6094	0.6398	0.39
Gemini-1.5-pro	0.7712	0.7022	0.542	0.4704	0.6077	0.286	0.7876	0.6744	0.531
Llama3-70B-Instruct	0.5984	0.6028	0.361	0.2774	0.4828	0.134	0.5684	0.6032	0.343
Llama3-8B-Instruct	0.4642	0.3658	0.17	0.2568	0.4547	0.117	0.4332	0.4793	0.208
Llama3.1-8B-Instruct	0.399	0.4235	0.169	0.2655	0.4499	0.119	0.4164	0.483	0.201
Mistral-7B-Instruct-v0.2	0.222	0.4501	0.1	0.121	0.3244	0.039	0.1908	0.2578	0.049
Qwen2-7B-Instruct	0	0.2923	0	0	0	0	0.0002	0.4123	0
Yi-1.5-9B	0.2884	0.5461	0.157	0.1064	0.6596	0.07	0.205	0.3724	0.076
Chatglm-9B	0.3666	0.6902	0.253	0.1832	0.6506	0.119	0.3514	0.682	0.24
Llama-3.2-1B-Instruct	0.0644	0.5055	0.033	0.0714	0.4757	0.034	0.0822	0.441	0.036
MolT5-small	0.2158	0.1052	0.023	0.2214	0.1031	0.023	0.2316	0.1011	0.023
MolT5-base	0.2074	0.1051	0.022	0.2358	0.1054	0.025	0.1856	0.1073	0.02
MolT5-large	0.4244	0.1015	0.043	0.4654	0.119	0.055	0.4496	0.1072	0.048
BioT5-base	0.5158	0.1526	0.079	0.5068	0.158	0.08	0.506	0.1597	0.081
OpenMolIns-large (Llama-3.2-1B)	0.2898	0.5951	0.172	0.1996	0.5849	0.117	0.2644	0.5956	0.157
OpenMolIns-large (Llama-3.1-8B)	0.8054	0.6678	0.538	0.5224	0.6398	0.334	0.7122	0.6548	0.466
OpenMolIns-light (Galactica-125M)	0.3202	0.6547	0.21	0.269	0.6521	0.175	0.3508	0.6435	0.226
OpenMolIns-small (Galactica-125M)	0.4172	0.642	0.268	0.2956	0.6385	0.189	0.3958	0.6452	0.255
OpenMolIns-medium (Gal-125M)	0.5904	0.5812	0.343	0.4608	0.5859	0.27	0.5874	0.5873	0.345
OpenMolIns-large (Galactica-125M)	0.6454	0.5927	0.383	0.495	0.5962	0.295	0.6388	0.5973	0.382
OpenMolIns-xlarge (Gal-125M)	0.7362	0.5744	0.423	0.5786	0.5677	0.328	0.7124	0.5697	0.406

ATTENTION PROBING: BLOCK-BASED VS. RAW SMILES REPRESENTATION

We conduct a qualitative attention probe using Qwen2-7B-Instruct (Qwen, 2024). The two inputs describe imatinib using raw SMILES or named blocks, followed by the same functional-group query:

Sentence 1: Raw SMILES input

```
Cc1ccc(NC(=O)c2ccc(CN3CCN(C)CC3)cc2)cc1Nc1nccc(-c2cccnc2)n1.  The
building block responsible for hinge binder is:
```

Sentence 2: Block-based SMILES with common names

```
1,4-Dimethylpiperazine [1*]CN1CCN(C)CC1, benzene
[1*]c1ccc([2*])cc1, formamide [1*]NC([2*])=O, toluene
[2*]c1ccc(C)c([1*])c1, 2-AMINOPYRIMIDINE [2*]Nc1nccc([1*])n1,
PYRIDINE [2*]c1cccnc1.  The building block responsible for hinge
binder is:
```

We compute the average attention between all tokens belonging to each block and the tokens of the query phrase "hinge binder", and visualize the resulting attention distribution across blocks.

Fig. 5 shows more localized attention to the functionally relevant block in the named-block input than in the raw-SMILES input. This single-molecule example illustrates an input-dependent attention pattern. Because both block boundaries and names change, it does not isolate their individual effects, establish chemical understanding, or demonstrate that the attention pattern causes improved optimization.

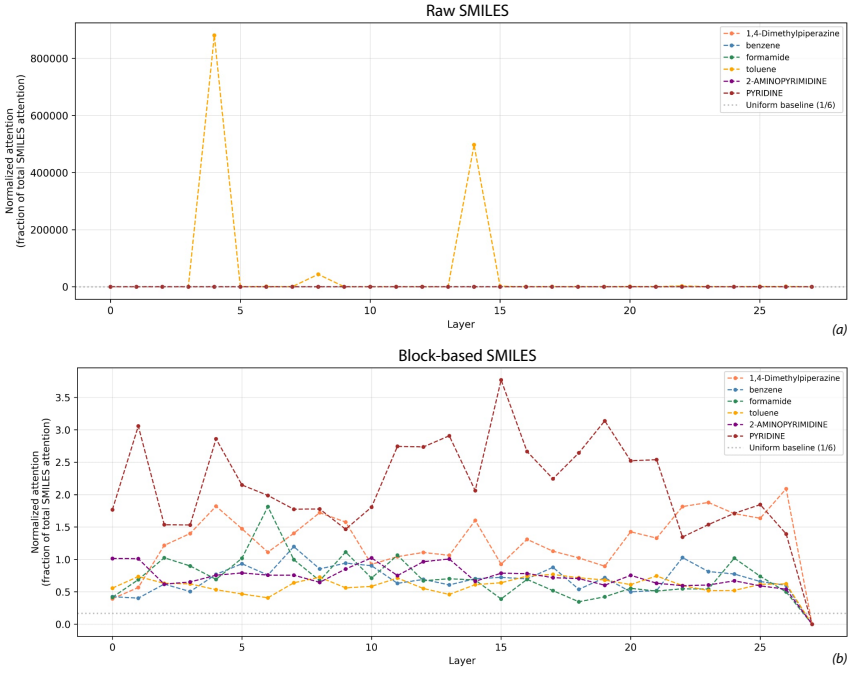


Figure 5: Qwen2-7B-Instruct attention for imatinib represented as raw SMILES (top) or named blocks (bottom). Each line shows average attention between block tokens and the functional-group query. The named-block example exhibits more localized attention; this is a qualitative comparison rather than a causal analysis.

G LITERATURE AND DATABASE RETRIEVAL FOR TARGET SUMMARIZATION

The Literature Agent summarizes reported protein targets and associated therapeutic agents through retrieval-augmented generation (RAG) over PubMed, UniProt, and ChEMBL. RAG grounds generated summaries in retrieved evidence (Lewis et al., 2020). The system constructs queries incorporating disease subtype, pathways, and known mutations, then standardizes retrieved associations to protein identifiers and aggregates them across sources. This is a literature and database retrieval task, not novel target discovery or experimental validation of therapeutic relevance. It differs from hit identification (§2.3.1), which takes a protein target as input and returns candidate molecules through ChEMBL retrieval or the DrugCLIP fallback.

Disease labels alone may omit molecular context relevant to target retrieval. For example, in non-small cell lung cancer, activating EGFR mutations can indicate sensitivity to EGFR-targeted therapy (Lynch et al., 2004). More generally, a system relying solely on the disease label may conflate molecular subgroups, leading to inclusion of targets that are not actionable for all patients. Incorporating subtype, pathway, and mutation information ensures that extracted targets are context-specific, providing a more precise and biologically meaningful representation.

The structured output supports downstream target prioritization and integration with predictive models. Its scope is to organize existing evidence in the supplied disease and molecular context; retrieval coverage and the strength of the underlying associations limit the conclusions that can be drawn.

We evaluate target retrieval and summarization on 30 diseases spanning oncology, neurology, cardiovascular disease, metabolic disorders, and autoimmune conditions. For 28 of the 30 diseases, all evaluated fields—the primary therapeutic target, its UniProt identifier, and its supporting reference—are correct. The remaining two cases do not satisfy the evaluation criterion because the returned proteins lack sufficiently established evidence as primary therapeutic targets. These results measure retrieval and summarization against expected evidence, not discovery of previously unknown targets.

The two cases that did not return the expected primary target are counted as unsuccessful under this evaluation criterion. Table 10 reports the full results of the Literature Agent on all 30 diseases, including the identified protein target, UniProt ID, and supporting reference.

H SHARED MEMORY MODULE

The shared memory module serves as the central coordination mechanism of MolLingo, enabling persistent context and iterative reasoning across agents and pipeline stages. We describe its design in terms of schema, storage, retrieval, and lifecycle management.

SCHEMA AND CONTENT TYPES

The shared memory maintains two categories of information. **Structured records** are stored as typed key-value entries with a defined schema, including: target annotations (disease name, protein name, UniProt ID, relevant pathways and mutations), candidate molecules (SMILES, block-based representation, docking scores, ADMET scores, QED, SA), and pipeline state (current stage, completed subgoals, pending tasks). **Unstructured context** is stored as free-text entries, including reasoning traces from each agent, literature summaries retrieved by the Literature Agent, design rationales for proposed modifications, and human-in-the-loop feedback. Each entry is tagged with a timestamp, the agent that wrote it, and the pipeline stage at which it was created, enabling provenance tracking and stage-specific retrieval.

RETRIEVAL MECHANISM

Agents access the shared memory through two retrieval modes. **Exact retrieval** is used for structured records, where agents query by key (e.g., UniProt ID, molecule SMILES) to retrieve specific entries. **Semantic retrieval** is used for unstructured context, where a query string is embedded and matched against stored entries using cosine similarity over dense vector representations, allowing agents to retrieve relevant reasoning traces or literature summaries without knowing their exact content. In practice, the Orchestrator primarily uses exact retrieval to track pipeline state, while the

Table 10: Literature and database target retrieval results across 30 diverse diseases. For diseases with multiple identified targets, all returned targets are listed. * indicates the two cases where the primary expected target was not returned as the top result.

Disease	Protein Target	UniProt ID	Reference
Breast cancer	Estrogen Receptor Alpha (ERα)	P03372	Targeting the Estrogen Receptor...
	HER2 (ERBB2)	P04626	Targeting HER2-positive breast cancer...
Non-small cell lung cancer	KRAS	P01116	Protein Kinase C as a Therapeutic Target...
Melanoma	BRAF	P15056	BRAF as therapeutic target in melanoma
Chronic myeloid leukemia	BCR-ABL1	P00519	Effects of a selective inhibitor of the Abl...
Colon cancer	VEGF	P15692	Targeted Therapy Drugs for Colorectal Cancer
Prostate cancer	Androgen Receptor	P10275	Protein Degradation Inducers of Androgen Receptor...
Acute myeloid leukemia	FLT3	P36888	Midostaurin prolongs survival in FLT3-mutated AML
Rheumatoid arthritis	TNF-α	P01375	Targeting rheumatoid arthritis...
Psoriasis	Interleukin-17A	Q16552	Psoriasis: rationale for targeting IL-17
Multiple sclerosis*	MOG	Q16653	Identification of a pathogenic antibody response...
	MBP	P02686	Multiple Sclerosis: An Important Role...
Alzheimer's disease	Amyloid-beta (APP)	P05067	Amyloid beta-protein assembly...
Parkinson's disease	Alpha-synuclein	P37840	Targeting the alpha-synuclein protein...
Type 2 diabetes	PTP1B	P18031	PTP1B as a Therapeutic Target...
Hypercholesterolemia	PCSK9	Q8NBP7	Targeting PCSK9...
Hypertension*	Angiotensin II receptor type 1	P30556	Angiotensin II receptor blockers...
	Endothelin-1	P05305	Endothelin-1 is a potent vasoconstrictor...
	ERAP1	Q9NZ08	Potential Therapeutic Drug Targets...
	ACVRL1	Q04771	Potential Therapeutic Drug Targets...
	Renin	P00797	Identification of novel proteomic biomarkers...
Asthma	IL-4 receptor alpha chain	P24394	Monoclonal antibodies targeting IL4R...
COPD	Alpha-1 antitrypsin	P01009	Role of elastases in the pathogenesis of COPD...
HIV	CD4	P01730	HIV Immune Cell Receptors...
Hepatitis C	NS5A	Q9YKI6	Phosphorylation of hepatitis C virus NS5A...
Influenza	Hemagglutinin	P03437	Hemagglutinin protein of Asian strains...
Malaria	PfATP6	E1CC99	Artemisinins target the SERCA...
Tuberculosis	InhA	P9WIR7	The progress of M. tuberculosis drug targets
Crohn's disease	NOD2	Q9HC29	NOD2 and its role in Crohn's disease
Ulcerative colitis	Interleukin-23	P78558	Effectiveness and Safety of Selective IL-23...
Osteoporosis	RANKL	O14788	RANKL as a target for osteoporosis
Sickle cell disease	Hemoglobin subunit beta	P68871	Silencing of BCL11A...
	BCL11A	Q9H165	Silencing of BCL11A...
Epilepsy	NaV1.1 (SCN1A)	Q99250	SCN1A/NaV1.1 channelopathies...
Migraine	CGRP Receptor (CALCRL + RAMP1)	Q16602	Targeting enkephalins...
Obesity	FGF-21	P54652	Adipokines as Clinically Relevant Therapeutic Targets...
COVID-19	Spike glycoprotein	P0DTC2	SARS-CoV-2 Proteins...

Literature Agent and Chemist Agent use semantic retrieval to access accumulated biological and chemical context.

Memory Size and Eviction

The shared memory is designed to hold the full context of a single drug discovery session, which in practice involves a target and a moderate number of hits, leads, and candidate drugs (tens to hundreds). No eviction policy is applied within a session, as the total memory footprint remains manageable. When the research subject changes (for example, when the user switches to a new protein target or disease) the memory module supports a selective reset operation that clears all target-specific entries (candidate molecules, docking results, ADMET profiles, design rationales, and literature summaries associated with the previous target) while retaining session-level context such as user preferences, high-level design goals, and human-in-the-loop feedback. This selective reset prevents context contamination across different research subjects while preserving information that remains relevant across targets. A full memory reset is also available when starting an entirely new discovery campaign.

Read and Write Protocol

All agents interact with the shared memory through a unified read/write interface. Write operations append new entries without overwriting existing ones, preserving the full history of the discovery process. Read operations return the most recent entry matching the query by default, with an option to retrieve the full history of entries for a given key. The Orchestrator additionally maintains a session log that records the sequence of agent calls, tool invocations, and memory operations, providing a complete audit trail of the discovery workflow that supports human-in-the-loop review and debugging.

Algorithm 4 Orchestrator Decision-Making Loop

Require: Task objective $\mathcal{O}$, shared memory $\mathcal{M}$, available actions $\mathcal{A}$
Ensure: Completed drug discovery workflow
Decompose $\mathcal{O}$ into initial subgoal queue $\mathcal{Q}$
while $\mathcal{Q}$ is not empty **do**
  $q \leftarrow$ NEXTSUBGOAL($\mathcal{Q}$)
  $c \leftarrow$ RETRIEVECONTEXT($\mathcal{M}$, q)
  Construct prompt $p \leftarrow (q, c, \mathcal{A})$
  $a \leftarrow$ LLM(p) ▷ Reason and select action
  if a = INVOKELITERATUREAGENT **then**
    $r \leftarrow$ LITERATUREAGENT(q)
  else if a = INVOKECHEMISTAGENT **then**
    $r \leftarrow$ CHEMISTAGENT(q)
  else if a = CALLTOOL(t) **then**
    $r \leftarrow$ TOOL(t, q)
  else if a = REQUESTHUMANFEEDBACK **then**
    $r \leftarrow$ HUMANFEEDBACK(q)
  else if a = TERMINATE **then**
    break
  end if
  WRITEMEMORY($\mathcal{M}$, q, r)
  $\mathcal{Q} \leftarrow$ UPDATESUBGOALS($\mathcal{Q}$, q, r)
end while
return RETRIEVEFINALCANDIDATES($\mathcal{M}$)

I ORCHESTRATOR AGENT: DECISION-MAKING AND COORDINATION

The Orchestrator Agent follows a ReAct-style (Yao et al., 2023) decision-making loop, interleaving reasoning steps with tool invocations and memory operations to decompose and execute the drug discovery workflow. At each step, the Orchestrator is provided with the current task objective, the history of completed subgoals stored in shared memory, and the available agents and tools. It then reasons about the next action to take, invokes the appropriate agent or tool, observes the result, updates the shared memory, and repeats until the objective is satisfied.

DECISION-MAKING ALGORITHM

Algorithm 4 describes the Orchestrator's decision-making loop. The Orchestrator is implemented as a prompted GPT-5.4 model with access to a fixed set of callable actions: invoking the Literature Agent, invoking the Chemist Agent, calling a tool directly, writing to or reading from shared memory, requesting human feedback, and terminating the session. At each step, the full task context, including the original objective, completed subgoals, and relevant memory entries, is included in the prompt, allowing the Orchestrator to make informed decisions without maintaining internal state.

PROMPT TEMPLATE

The following prompt template is used at each step of the Orchestrator's decision-making loop. Fields in curly braces are filled in at runtime from the task context and shared memory.

Orchestrator Prompt Template

```
You are the Orchestrator of MolLingo, an autonomous drug discovery
system.  Your role is to decompose the given objective into
subgoals and coordinate the Literature Agent, Chemist Agent, and
available tools to complete the drug discovery workflow.
Current objective:  {OBJECTIVE}
Completed subgoals:  {COMPLETED_SUBGOALS}
```

```
Relevant context from memory: {MEMORY_CONTEXT}
Available actions: - invoke_literature_agent(task): retrieve and
synthesize biological knowledge for a given task or target. -
invoke_chemist_agent(task): perform molecular reasoning, hit
identification, fragment growing, or lead optimization. -
call_tool(tool_name, inputs): invoke a specific tool (e.g.,
docking, ADMET prediction, ChEMBL retrieval). -
request_human_feedback(question): ask the user for clarification,
constraints, or high-level guidance. - terminate(): end the
session and return final candidates.
Based on the current state, reason step by step about what needs to
be done next, then select the most appropriate action. Respond in
the following format:
Reasoning: {step-by-step reasoning}Action: {selected action and
inputs}
```

The Orchestrator's reasoning trace and selected actions are written to the shared memory at each step, providing a complete audit trail of the workflow and enabling human-in-the-loop review at any stage of the discovery process.

J MolLingo's Toolset

MolLingo's agents are equipped with a curated set of tools that support each stage of the drug discovery pipeline. We describe the core tools below, grouped by function.

Molecular Docking Docking is performed using AutoDock Vina (Trott & Olson, 2010). Co-crystal structures are retrieved from the RCSB Protein Data Bank (Berman et al., 2000), selecting the available structure with the best reported crystallographic resolution. The bound ligand is removed and its pocket defines the docking search box. If no co-crystal structure is available, an unliganded RCSB structure is used for blind docking. If no experimental structure is available, the workflow uses an AlphaFold-predicted structure (Jumper et al., 2021). These fallbacks extend the workflow to targets lacking co-crystal structures, but their accuracy depends on the available structure and pocket definition.

SMILES to Block-Based Representation Given a SMILES string, this tool tokenizes the molecule into a sequence of building blocks from the learned BFE vocabulary, following the procedure described in §2.2. Each block is converted to a block-based SMILES string with wildcard attachment points and optionally paired with its common chemical name, producing the block-based representation used by the Chemist Agent for molecular reasoning and optimization.

Fragment Screening with DrugCLIP Given a target and the curated fragment library (Appendix C), DrugCLIP (Gao et al., 2023) ranks fragments by protein–ligand embedding similarity. This score is used for retrieval rather than as an absolute affinity estimate. The top k fragments are subsequently docked for pose and score evaluation.

Context-Aware Fragment Growing Given a docked hit fragment and its binding pose, this tool identifies growth hotspots and queries the Chemist Agent to propose fragment extensions, following the procedure described in §2.3.2 and detailed in Algorithm 3.

ADMET Optimization Given a lead molecule and its ADMET profile, this tool queries the Chemist Agent to propose block-level modifications that improve ADMET properties while preserving structural similarity to the original molecule, following the procedure described in §2.3.3.

Supporting Tools The following utility tools support the Literature Agent and Chemist Agent throughout the pipeline:

- **ChEMBL binder retrieval.** Queries ChEMBL (Gaulton et al., 2012) for known binders of a given protein target, returning compounds with reported binding affinity data.

- **Molecular identifier conversion.**
  The utilities `name2smiles` and `smiles2name` convert between common names and SMILES; `uniprotid2name` and `name2uniprotid` map between protein names and UniProt identifiers; and `disease2drugs` retrieves approved drugs for a disease indication.
- **Drug-likeness filtering and ADMET ranking.** Filters candidate molecules by drug-likeness criteria and ranks them by the composite ADMET score described in §D.
- **Butina clustering.** Clusters a set of retrieved binders using the Butina algorithm (Butina, 1999) based on Tanimoto similarity, and selects one representative molecule per cluster. This ensures that the hits forwarded to the hit-to-lead stage are chemically diverse rather than redundant.
- **Property scoring.** Computes standard molecular property scores including synthetic accessibility (SA) (Ertl & Schuffenhauer, 2009), quantitative estimate of drug-likeness (QED) (Bickerton et al., 2012), and Tanimoto similarity (Tanimoto, 1958) between pairs of molecules, used throughout the pipeline for filtering, ranking, and similarity-constrained optimization.

K Implementation Details

Benchmark Controls and Backbone Selection

The ADMET representation benchmarks use identical objectives, tool access, output formats, and generation budgets within each backbone. Inputs differ only in molecular encoding. The hit-to-lead experiment instead compares complete workflows and includes an encoding ablation within the docking-guided workflow (Appendix E).

The Literature Agent is not invoked in either benchmark. Hit identification uses ChEMBL retrieval; DrugGEN uses ChEMBL as training data, whereas MolLingo uses it as a screening library. These data-access arrangements do not establish identical exposure to individual compounds or exclude overlap with LLM pretraining data.

GPT-5.4 was selected for the largest experiments primarily because of available API budget. The architecture supports substituting other backbones, and paired results for Claude-4.6-Sonnet and Gemini-3-Pro are reported in Tables 1 and 8.

Prompt Templates

Molecule Optimization Prompt

The following prompt template is used for all baseline LLM models in the ADMET optimization benchmark. The molecule SMILES and target ADMET property are filled in at inference time.

Molecule Optimization Prompt

```
Given the molecule {SMILES}, propose structural modifications aimed
at reducing its risk of {ADMET_PROPERTY}(e.g., liver injury for
DILI, cardiotoxicity for hERG).
For each proposed analog, briefly explain the rationale behind the
modification and how it may lower {ADMET_PROPERTY}risk.
In the conclusion, provide five candidate molecules with
potentially improved {ADMET_PROPERTY}profiles, formatted as a JSON
dictionary where each entry includes the SMILES string and a short
description of the modification.  Make sure all the proposed SMILES
are valid.
Example:
{
 "candidate_1": {
 "smiles": "CC(C1=CC=CC=C1)N(C(=O)N)O",
 "modification": "Removed sulfur-containing heterocycle to reduce potential
```

```
 bioactivation and reactive metabolite formation."
 },
 ...
 "candidate_5": {
 "smiles": "CC(C1=CC2=CC=CC=C2O1)N(C(=O)N)O",
 "modification": "Replaced sulfur with oxygen to decrease likelihood of forming
 toxic sulfur metabolites."
 }
}
```

Note that for MolLingo, the molecule is provided in block-based SMILES format with common block names rather than raw SMILES, while the output format remains identical. This is the only difference between the MolLingo prompt and the baseline prompt, isolating the effect of molecular representation on optimization performance.

HIT-TO-LEAD OPTIMIZATION PROMPT

The following prompt template is used for all baseline LLM models in the hit-to-lead benchmark. The molecule SMILES and protein target are filled in at inference time.

Hit-to-Lead Optimization Prompt

```
You are an expert medicinal chemist.  Given a hit molecule and a
protein target, your task is to propose structural modifications
that improve the binding affinity of the molecule to the target
protein, while maintaining sufficient structural similarity to the
original molecule to preserve its pharmacophore.
Molecule SMILES: {SMILES}
Protein target:  {PROTEIN_NAME}(UniProt:  {UNIPROT_ID})
Guidelines:
- Propose modifications that are likely to form new favorable
interactions (hydrogen bonds, hydrophobic contacts, or
electrostatic interactions) with the target protein.
- Each modification should be conservative | the proposed analog
should remain structurally similar to the original molecule
(Tanimoto similarity > 0.4).
- Avoid modifications that introduce known toxicophores or violate
Lipinski's rule of five.
- Each proposed molecule must be a valid, chemically synthesizable
structure.
For each proposed analog, briefly explain the rationale behind the
modification and how it may improve binding affinity to
{PROTEIN_NAME}.
In the conclusion, provide five candidate molecules, formatted as a
JSON dictionary where each entry includes the SMILES string and a
short description of the modification.  Make sure all proposed
SMILES are valid.
Example:
{
 "candidate_1": {
 "smiles": "CC(C1=CC=CC=C1)N(C(=O)N)O",
 "modification": "Added hydrogen bond donor at para position to form additional
 interaction with Asp residue in the binding site."
 },
 ...
 "candidate_5": {
 "smiles": "CC(C1=CC2=CC=CC=C2O1)N(C(=O)N)O",
 "modification": "Replaced phenyl with naphthalene to improve hydrophobic
 packing in the lipophilic pocket."
 }
}
```

In the docking-guided MolLingo workflow, the generation prompt includes the target protein, nearby residues, available volume, attachment-atom element type, and either the BFE-block SMILES or full growing-fragment SMILES, as described in §2.3.2. The direct baseline does not receive the docking-derived context. Table 2 therefore includes both a workflow comparison against direct prompting and a representation comparison between the two docking-guided variants; it does not independently isolate each structural-context input.

Hyperparameters and Pipeline Configuration

Several key parameters in MolLingo's pipeline are user-configurable and can be adjusted based on available computational resources and time constraints. These include the number of protein targets returned per disease by the Literature Agent, the number of top hits retrieved from ChEMBL per target, the number of fragment modifications proposed per hotspot in the hit-to-lead stage, the number of structural modifications proposed per iteration in lead optimization, and the number of lead optimization iterations. Increasing these values broadens the search space at the cost of additional computation and LLM calls, but does not necessarily improve optimization quality (Appendix L). The default pipeline settings are one top hit retrieved from ChEMBL per target, five fragment modifications proposed per hotspot in the hit-to-lead stage, 15 structural modifications proposed per lead optimization iteration, and two lead optimization iterations per molecule. These defaults do not apply to the controlled representation benchmarks or the ablations: Tables 1 and 8 use one iteration and 40 candidate variants for every model, while Appendix L specifies the settings for each ablation. The representation benchmarks compare BFE block-based SMILES with raw SMILES under an identical generation budget; the Orchestrator and cross-iteration memory play essentially no role in this single-iteration setting.

Butina clustering. Since ChEMBL binders for a given target often share common scaffolds, we apply the Butina algorithm (Butina, 1999) to cluster retrieved compounds by Tanimoto similarity using a distance threshold of 0.7, and select one representative molecule per cluster. This ensures that the hits forwarded to the hit-to-lead stage are chemically diverse rather than redundant.

List of Protein Targets for Hit-to-Lead Evaluation

Table 11 lists the 30 protein targets used in the hit-to-lead evaluation, spanning diverse therapeutic areas including oncology, epigenetics, metabolism, and apoptosis.

Table 11: Protein targets used for hit-to-lead evaluation, grouped by biological function.

UniProt ID	Gene Name	UniProt ID	Gene Name
Kinases			
P31749	AKT1	P28482	MAPK1
P27361	MAPK3	P06493	CDK1
P24941	CDK2	P11802	CDK4
P00519	ABL1	O60674	JAK2
P49841	GSK3B	O14757	CHEK1
Epigenetic regulators			
Q13547	HDAC1	Q92769	HDAC2
P26358	DNMT1	Q15910	EZH2
O60341	KDM1A		
Metabolic enzymes			
P00374	DHFR	P04818	TYMS
P04035	HMGCR	P49327	FASN
O75874	IDH1	P48735	IDH2
P00338	LDHA	P00558	PGK1
Apoptosis & proteolysis			
P42574	CASP3	Q14790	CASP8
P08253	MMP2	P14780	MMP9
Chaperones & DNA repair			
P07900	HSP90AA1	P08238	HSP90AB1
P09874	PARP1		

L Optimization Ablations

L.1 Experimental Setup and Metrics

We conduct DILI optimization experiments that vary memory, adaptive orchestration, candidate count, and iteration count. These experiments complement the single-iteration representation comparisons in Tables 1 and 8. The memory and Orchestrator designs are described in Appendices H and I. Each subsection specifies its generation budget and enabled components.

Evaluation metrics. For these DILI experiments, improvement is computed as

$$\text{Improvement}(\%) = 100\,\frac{z_{\text{original}} - z_{\text{optimized}}}{z_{\text{original}}}, \tag{1}$$

where z_{original} and $z_{\text{optimized}}$ are the DILI oracle's logit outputs for the original and optimized molecules. This metric measures a relative change in oracle logits, not a percentage reduction in a calibrated probability of liver injury. Similarity is the Tanimoto similarity to the original molecule, expressed as a percentage. Imprv×Sim is the product of the reported mean improvement and mean similarity, using their percentage values. We also report chemical validity and, for the budget sweeps, QED (drug-likeness; higher is better) and synthetic accessibility (Synth; lower is better).

L.2 Memory Ablation

Setup. We evaluate all molecules in the DILI evaluation set with the Orchestrator disabled, two optimization iterations, and eight candidate variants per iteration. We compare no memory, memory without stored oracle scores, and memory with stored oracle scores. Memory records the reference molecule, fragment substitutions, and the Chemist Agent's rationale for every variant, optionally including its DILI oracle score. On subsequent iterations, these records are included in the Chemist Agent's prompt. Omitting oracle scores from memory does not remove the oracle used to evaluate optimization outcomes.

Table 12: Memory ablation for DILI optimization with no Orchestrator, two iterations, and eight variants per iteration. Oracle scores are either omitted from or included in the stored history.

Condition	Improvement (%) ↑	Similarity (%) ↑	Imprv×Sim ↑	Validity (%) ↑
Without memory	33.509 (12.577)	30.796 (12.167)	1031.94	100
Memory, without oracle scores	35.174 (17.168)	52.898 (9.654)	1860.63	100
Memory, with oracle scores	36.926 (15.399)	47.795 (9.100)	1764.88	100

Results. Memory increases similarity to the original molecule and modestly increases mean DILI improvement (Table 12). Both memory conditions improve the joint score, while validity remains at 100%. Including oracle scores yields the highest mean improvement; omitting them yields the highest similarity and joint score.

Search behavior. Beyond aggregate performance, the observed optimization traces reveal three differences:

- **Avoiding redundant edits.** With memory, no modification in the next iteration is identical to a previous modification. Without memory, 16.7% of modifications repeat an earlier edit.
- **Abandoning unsuccessful sites.** When an oracle score contradicts the agent's stated rationale, the agent with oracle-augmented memory avoids re-editing the same block 95% of the time. The corresponding rate without memory is 80%; because the previous evidence is unavailable in that condition, choosing another block cannot be attributed to recall of the unsuccessful edit.
- **Revisiting successful sites.** With oracle-augmented memory, 8% of sites receive a second, different substitution after a favorable oracle outcome, compared with 0% without memory. Here, a favorable DILI outcome means a lower predicted toxicity logit, not a higher raw DILI logit.

These observations are consistent with memory supporting history-aware search: avoiding repeated edits, moving away from unsuccessful modifications, and further exploring promising sites.

L.3 Orchestrator Ablation

Setup. We retain memory with oracle scores and compare a fixed pipeline of two iterations and eight variants per iteration with adaptive orchestration. The Orchestrator reads the accumulated history and adjusts the number of iterations and candidate variants. It exploits promising sites by requesting additional substitutions at those sites, and broadens exploration when the target property remains unsatisfactory. Across molecules, it selects an average of 3.18 iterations and 27 variants, with maxima of five iterations and 35 variants.

Table 13: Fixed-budget versus adaptive DILI optimization with oracle-augmented memory in both conditions. The adaptive condition uses a larger generation budget, so the comparison does not isolate orchestration at matched compute.

Condition	**Improvement (%)** ↑	**Similarity (%)** ↑	**Imprv×Sim** ↑	**Validity (%)** ↑
Fixed pipeline (2 iterations, 8 variants)	36.926 (15.399)	47.795 (9.100)	1764.88	100
With Orchestrator	58.817 (19.965)	47.813 (10.121)	2812.22	100

Results. Adaptive orchestration increases mean improvement from 36.926% to 58.817% while maintaining nearly the same mean similarity (47.795% versus 47.813%) and 100% validity (Table 13). The joint score increases from 1764.88 to 2812.22. Because the adaptive configuration uses a larger generation budget, this comparison does not separate the effect of adaptive allocation from additional sampling.

L.4 Effect of the Number of Candidate Variants

Setup. Using GPT-5.4 as the backbone, we fix optimization to one iteration and vary the number of requested candidate variants $k \in \{8, 20, 30, 40\}$. This isolates candidate count from cross-iteration refinement. The $k = 40$ setting is the GPT-5.4-based MolLingo configuration reported in Table 1.

Table 14: Effect of candidate count on DILI optimization with GPT-5.4 and one iteration.

k	**Improvement (%)** ↑	**Similarity (%)** ↑	**Imprv×Sim** ↑	**Drug-likeness** ↑	**Synth** ↓	**Validity (%)** ↑
8	17.087 (8.943)	44.161 (20.631)	754.58	0.683 (0.115)	2.468 (0.735)	100
20	20.281 (10.202)	39.193 (14.905)	794.87	0.714 (0.111)	2.599 (0.626)	100
30	19.782 (9.688)	35.361 (14.402)	699.51	0.716 (0.118)	2.810 (1.199)	100
40	13.028 (7.724)	36.608 (15.576)	476.93	0.597 (0.189)	3.087 (0.854)	100

Results. Among the tested settings, $k = 20$ yields the highest mean improvement and joint score, while $k = 8$ best preserves similarity and has the lowest synthetic accessibility score (Table 14). More candidates do not monotonically improve performance. The $k = 40$ setting used in the representation benchmark has the lowest improvement and joint score in this sweep. These results characterize sensitivity to the generation budget but do not identify the mechanism underlying the decline at larger candidate counts.

L.5 Effect of the Number of Optimization Iterations

Setup. Using GPT-5.4 as the backbone, we vary the number of optimization iterations from one to five while requesting eight variants per iteration. The iteration count is fixed rather than selected by the Orchestrator. The two-iteration result matches the no-memory condition in Table 12.

Table 15: Effect of fixed iteration count on DILI optimization with GPT-5.4 and eight variants per iteration.

Iterations	Improvement (%) ↑	Similarity (%) ↑	Imprv×Sim ↑	Drug-likeness ↑	Synth ↓	Validity (%) ↑
1	17.087 (8.943)	44.161 (20.631)	754.58	0.683	2.468	100
2	33.509 (12.577)	30.796 (12.167)	1031.94	0.708	2.773	100
3	55.760 (12.274)	24.151 (15.005)	1346.66	0.703	2.77	100
4	50.728 (13.091)	20.983 (9.382)	1064.43	0.71	2.699	100
5	51.790 (14.002)	18.410 (8.033)	953.45	0.682	2.87	100

Results. Mean improvement rises from 17.087% to 33.509% to 55.760% over the first three iterations, while similarity declines from 44.161% to 30.796% to 24.151% (Table 15). Three iterations produce the highest mean improvement and joint score among the tested settings. Further iterations do not improve on this peak and continue to reduce similarity, which reaches 18.410% after five iterations. This sweep exposes the property–similarity trade-off that motivates molecule-specific stopping and search allocation, rather than a single global iteration count.

M Alternative Representations and Optimizers

We compare alternative molecular representations and non-LLM optimization methods under a common candidate-evaluation budget, then discuss additional representation formats that were not included in the quantitative comparison.

M.1 Matched-Oracle-Budget DILI Optimization

Protocol. Each method uses two optimization iterations with eight candidate variants per iteration and no shared memory. Every candidate receives one oracle evaluation, giving $8 \times 2 = 16$ candidate oracle calls per reference molecule for every method. This common iterative budget avoids comparing an iterative optimizer against a single-shot LLM. The comparison includes REINVENT and Graph-GA as non-LLM baselines, together with GPT-5.4 operating on SELFIES, raw SMILES, and BFE block-based SMILES with or without common block names. The representation comparisons hold the LLM backbone and generation budget fixed; comparisons with non-LLM methods additionally change the optimization algorithm. Matching oracle calls does not imply equal training cost, inference compute, or wall-clock time.

Metrics. We report DILI improvement, Tanimoto similarity to the reference molecule, their combined Imprv×Sim score, and QED (drug-likeness), using the definitions in Appendix L.1. The block-based SMILES condition with names is the same two-iteration, eight-variant, no-memory configuration reported in Table 12. It differs from the single-iteration, 40-variant representation benchmarks in Tables 1 and 8.

Table 16: Additional DILI optimization baselines under a matched budget of 16 candidate oracle calls per molecule: two iterations, eight variants per iteration, and no shared memory. All four representation conditions use GPT-5.4. Italicized similarity values highlight pronounced structural drift, rather than a formal threshold for analog validity.

Method	Improvement (%) ↑	Similarity (%) ↑	Imprv×Sim ↑	Drug-likeness ↑	Oracle calls
REINVENT	59.243 (23.055)	*14.220 (3.900)*	842.44	0.725 (0.125)	16
Graph-GA	20.170 (11.708)	51.765 (8.130)	1044.1	0.653 (0.146)	16
SELFIES + GPT-5.4	53.302 (15.043)	*19.543 (10.437)*	1041.68	0.538 (0.208)	16
SMILES + GPT-5.4	26.385 (17.560)	38.048 (8.465)	1003.9	0.730 (0.116)	16
Block SMILES with names	33.509 (12.577)	30.796 (12.167)	1031.94	0.708 (0.163)	16
Block SMILES without names	30.394 (14.632)	33.899 (19.794)	1030.34	0.681 (0.099)	16

Property improvement versus structural preservation. REINVENT and SELFIES achieve high mean DILI improvement (59.243% and 53.302%) at low similarity (14.220% and 19.543%; Table 16), indicating substantial departure from the reference molecule. Graph-GA achieves the highest similarity and joint score, while its mean DILI improvement is lower than that of either block-based condition. These results distinguish property optimization from local analog optimization; they do not identify the mechanism causing structural drift.

Effect of block names. Adding common chemical names increases mean improvement from 30.394% to 33.509% and QED from 0.681 to 0.708, but decreases similarity from 33.899% to 30.796%. The joint scores are nearly identical (1030.34 without names versus 1031.94 with names). Thus, this experiment does not show a uniform benefit from names across metrics. Relative to raw SMILES, named blocks yield higher mean improvement but lower similarity, with a modestly higher joint score (1031.94 versus 1003.9). These trade-offs are specific to the two-iteration setting and should not be conflated with the single-iteration results.

Graph-GA and reasoning-guided proposals. Graph-GA and named-block MolLingo obtain similar joint scores under the matched oracle budget (1044.1 and 1031.94). In the evaluated Graph-GA setup, candidates are generated through six types of random edits and selected using oracle scores. MolLingo generates proposals conditioned on the chemical objective and provides a rationale for each edit, with the oracle used for ranking and evaluation. This makes its proposals available for inspection and history-conditioned refinement (Appendix L.2), but does not establish superior oracle efficiency or mechanistic correctness of the rationales.

M.2 Additional Representation Considerations

InChI. An additional InChI trial yields only 6.67% valid outputs. This low validity limits its practical utility for molecular optimization in the tested configuration, so it is not included in the main matched-budget table. The result is specific to this trial rather than a claim that InChI cannot support molecular optimization under other models or decoding procedures.

IUPAC names. The model can generate new chemical names that do not resolve through a PubChem lookup. A failed lookup does not by itself establish chemical invalidity: a novel but valid name may lack a database entry. Without an independently validated IUPAC-to-structure conversion procedure, we cannot reliably determine validity or evaluate the resulting molecules. We therefore exclude this condition from the quantitative comparison rather than equating its lookup success rate with molecular validity.

SAFE. SAFE concerns molecular serialization, whereas BFE selects a decomposition into building blocks. It is therefore complementary to, rather than a direct replacement for, the decomposition algorithm: BFE-derived blocks could in principle be serialized using SAFE. An evaluation of SAFE serialization with the decomposition held fixed would address a different question from the choice of fragmentation method; we do not report such an experiment here.

N Use Cases

We first present a continuous worked example tracing MolLingo's decisions from disease context to a final optimized candidate (Appendix N.1). We then provide complementary stage-specific illustrations in Figures 6–8. Those figures demonstrate disease-to-target retrieval, protein-to-lead design, and lead optimization separately, rather than depicting the continuous trajectory described below.

N.1 End-to-End Case Study: Inhibitor-Resistant Metastatic Melanoma

The input is *BRAF/MEK inhibitor-resistant metastatic melanoma*. We trace a selected final candidate through target retrieval, hit identification, fragment growing, and ADMET optimization. This retrospectively selected example illustrates the workflow rather than average performance. Docking and ADMET values are computational predictions, and the accompanying rationales are the agent's design hypotheses.

Stage 1: Target identification. The Literature Agent considers BRAF V600E-mutant melanoma with acquired resistance to dabrafenib/trametinib attributed, in the input context, to MAPK pathway reactivation. Because BRAF and MEK are already targeted, its rationale is to investigate the downstream ERK1/2 kinases. It selects **MAPK1 (ERK2)** as the protein target for the subsequent stages.

Stage 2: Hit identification. The agent searches ChEMBL for active MAPK1-targeting scaffolds and selects the top 10 hits for molecular docking. The initial hit on the selected trajectory has a docking score of -6.5 kcal/mol. Table 17 lists its SMILES alongside the subsequent lead and final candidate. These case-specific selection counts differ from the default pipeline settings described in the implementation details.

Stage 3: Hit-to-lead generation through fragment growing. Docking-derived pocket analysis identifies an approximately 72.1 $Å^3$ growth pocket lined by ALA35, ILE56, TYR64, ARG67, and THR68. The Chemist Agent proposes adding a pyrimidine ring to occupy this volume. Its stated rationale is to improve hydrophobic packing, introduce hydrogen-bond acceptors, and add controlled polarity without excessive bulk. The top 20 fragment-grown candidates are retained according to their docking scores. The lead on the selected trajectory scores -7.3 kcal/mol, a 0.8 kcal/mol improvement over the initial hit.

Stage 4: ADMET-guided lead optimization. The top 20 leads undergo multi-property ADMET profiling, after which five candidates are retained for multi-objective optimization. For the selected lead, the primary liability is its high DILI oracle score of 0.934. The Chemist Agent proposes three modifications:

- **Replace the unsaturated sulfone-containing ring with cyclopentane.** The agent hypothesizes that removing the unsaturated sulfur-containing ring may reduce metabolic activation and reactive-metabolite formation.
- **Add a methoxy group.** The agent proposes this substitution to introduce polarity, with the hypothesis that it may reduce nonspecific liver accumulation.
- **Replace pyrimidine with pyridine.** The agent changes the heteroaromatic ring's electronics with the aim of reducing bioactivation risk.

The final candidate has a docking score of -8.0 kcal/mol, a further 0.7 kcal/mol improvement over the lead, and a Tanimoto similarity of 0.41 to the lead. Docking and structural similarity are evaluated separately; similarity is not a direct measure of binding-mode preservation.

Table 17: Compound progression in the selected melanoma case study. More negative docking scores are favorable; scores do not constitute experimental affinity measurements.

Stage	Molecule (SMILES)	Docking score (kcal/mol)
Hit	`O=S1(=O)C=CC(OS(=O)(=O)c2ccccc2)C1`	-6.5
Lead (fragment growing)	`O=S1(=O)C=CC(OS(=O)(=O)c2cccc(-c3ncccn3)c2)C1`	-7.3
Final candidate (DILI optimization)	`COC1CCC(OS(=O)(=O)c2cccc(-c3ccccn3)c2)C1`	-8.0

Outcome and trade-offs. The DILI score decreases from 0.934 to 0.580, a relative reduction of approximately 38% in the oracle score (Table 18). HIA increases from 0.633 to 0.879, AMES decreases from 0.304 to 0.081, and hERG decreases slightly. Pgp liability increases from 0.036 to 0.157, and the value of increased BBB penetration depends on the intended indication. Docking scores improve from -6.5 to -7.3 to -8.0 kcal/mol. The trajectory illustrates both the intended improvements and the remaining multi-property trade-offs.

Table 18: Multi-property ADMET oracle scores for the selected lead and final candidate. Changes are final minus lead, calculated from the displayed scores. Arrows indicate the preferred direction for the optimization profile; the desirability of BBB penetration depends on the therapeutic context.

Endpoint	Lead	Final candidate	Change
DILI (liver injury) ↓	0.934	0.580	−0.354
hERG (cardiotoxicity) ↓	0.685	0.673	−0.012
AMES (mutagenicity) ↓	0.304	0.081	−0.223
Pgp (efflux liability) ↓	0.036	0.157	+0.121
HIA (intestinal absorption) ↑	0.633	0.879	+0.246
BBBP (brain penetration)	0.431	0.654	+0.223

N.2 Complementary Stage-Specific Illustrations

Disease to Biological Target. Fig. 6 illustrates MolLingo's target identification capability using Parkinson's disease as input. The Literature Agent retrieves and synthesizes relevant biological knowledge to identify biologically relevant protein targets, returning UniProt identifiers and supporting evidence from the literature.

Protein to Lead. Fig. 7 illustrates ChEMBL binder retrieval for SNCA and a separate docking-guided fragment-growing example for ABL1. The panels demonstrate the two tool interactions rather than a continuous trajectory for a single target.

Lead Optimization. Fig. 8 shows lead optimization on imatinib. Given the molecule and its ADMET profile, the Chemist Agent proposes chemically motivated block-level modifications to improve blood–brain barrier penetration, while reporting predicted changes across all six ADMET endpoints. Each proposed molecule is evaluated by the ADMET oracle models, and the reasoning trace is stored in shared memory for iterative refinement.

O Evaluation Limitations

Our evaluation is limited to early-stage computational drug discovery: the in-silico, or "dry-lab," phase, and does not include molecule synthesis, biochemical assays, cell-based experiments, animal studies, or other wet-lab validation.



Figure 6: Disease to biological target: the Literature Agent identifies the primary therapeutic protein target and its UniProt identifier from a disease name input.

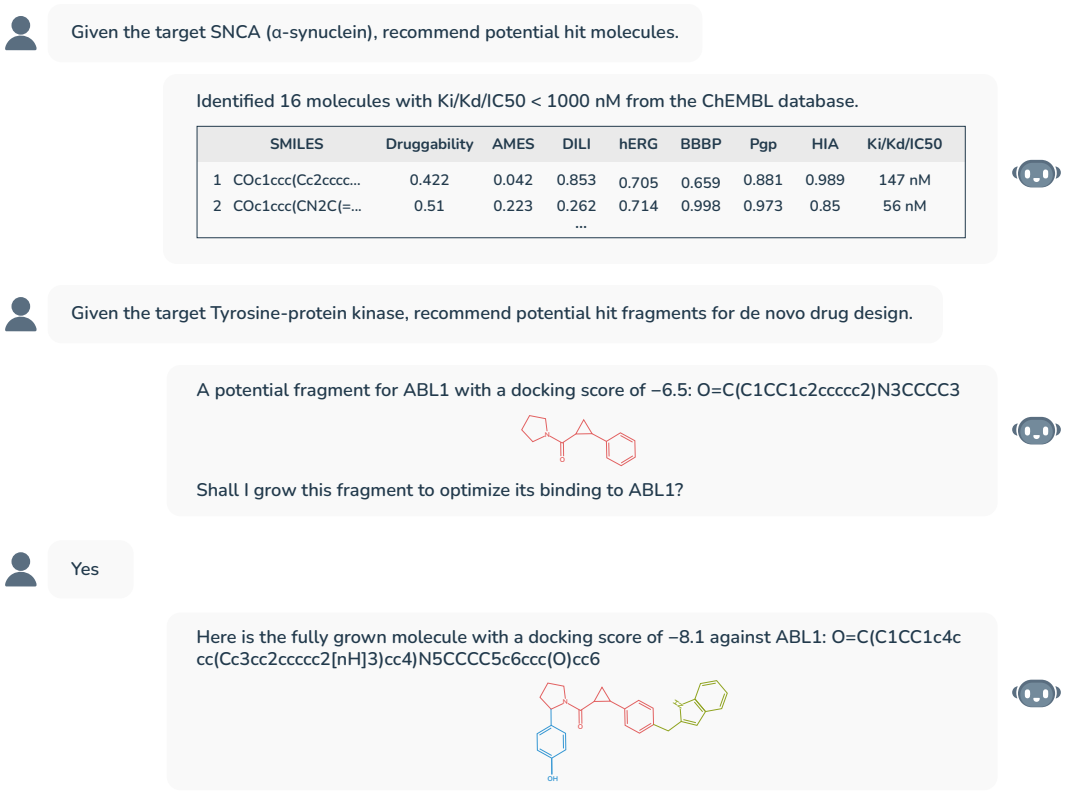


Figure 7: Protein-to-lead tool interactions: ChEMBL binder retrieval and docking-guided fragment growing are illustrated for different targets.

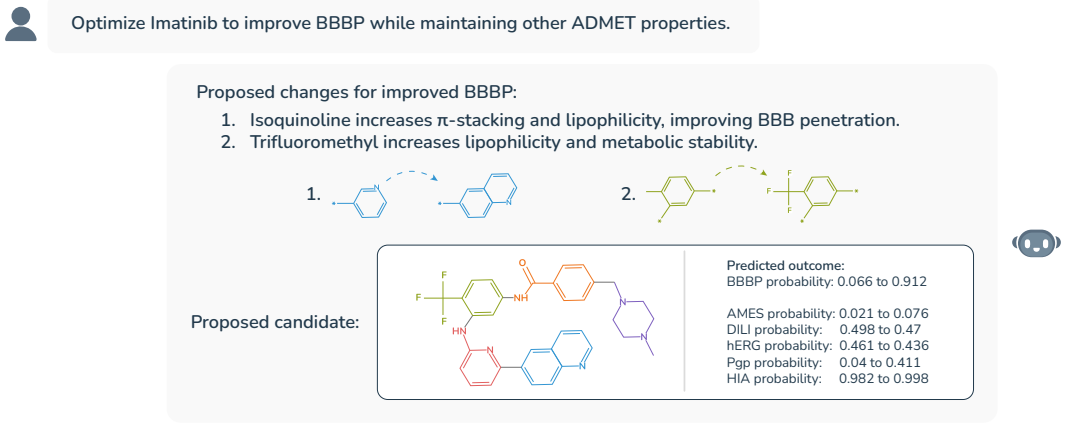


Figure 8: Lead optimization: the Chemist Agent iteratively refines a lead molecule through block-level modifications guided by ADMET profiling and accumulated reasoning traces.