

MADDIE: Multi-Agent Drug Discovery Engine

Ye Liu^{1†}, Reza Averly^{2†}, Avery Meyer¹, Frazier N. Baker²,
Ruoxi Gao², Daniel Adu-Ampratwum³, Xia Ning^{1,2,3,4*}

¹Biomedical Informatics, The Ohio State University.

²Computer Science and Engineering, The Ohio State University.

³Medicinal Chemistry and Pharmacognosy, The Ohio State University.

⁴Translational Data Analytics Institute, The Ohio State University.

*Corresponding author(s). E-mail(s): ning.104@osu.edu;
Contributing authors: liu.12989@osu.edu; averly.1@osu.edu;
meyer.1938@osu.edu; baker.3239@osu.edu; gao.2396@osu.edu;
adu-ampratwum.1@osu.edu;

[†]These authors contributed equally to this work.

Abstract

Drug discovery remains a resource-intensive and time-consuming process, involving billions of dollars and a decade of development to advance a drug to market. While many *in silico* tools have accelerated this process, operating them still demands substantial time, effort, and expertise from medicinal chemists. Autonomous systems built on Large Language Models (LLMs) have emerged as a promising solution, yet current designs limit their performance. Existing single-agent systems scale poorly to long-horizon, multi-step, and realistic tasks, whereas multi-agent systems typically execute predefined workflows that cannot capture the iterative, non-linear nature of a real drug discovery campaign. Here, we present MADDIE, a multi-agent drug discovery engine. It comprises a central agent, ORCHESTRATOR, and four specialized worker agents: HITIDENTIFIER, LEADOPTIMIZER, SYNTHESIZER, and EVALUATOR, each representing a critical step in a drug discovery campaign. Under ORCHESTRATOR’s orchestration and coordination, these worker agents are composed into flexible workflows, enabling MADDIE to adapt its strategy and navigate the drug discovery pipeline end-to-end. Across 50 protein targets, MADDIE produces molecules satisfying key property requirements and their synthesis routes, outperforming existing agentic systems, generative models, and general-purpose LLMs by a wide margin. Our analysis reveals how these agents collaborate and that MADDIE can employ different strategies across targets to achieve the campaign’s goal, while still being

efficient in time and tokens. Together, they demonstrate MADDIE as a low-cost, effective, and efficient autonomous engine for accelerating drug discovery. Code and dataset is available via <https://github.com/ninglab/MADDIE>.

1 Introduction

Drug discovery is a complex, capital-intensive, and time-consuming process: bringing one approved drug to the market typically costs more than one billion of dollars and takes more than a decade [1–4]. To expedite this process, many *in silico* tools have been developed [5–10], each designed for a specific task in the pipeline (e.g., predicting binding affinity). Using them, however, has traditionally fallen to medicinal chemists. Autonomous systems capable of intelligently using these tools have gained considerable attention recently [11–19], substantially reducing the time and effort demanded of medicinal chemists while showing promising results. These systems are typically built upon Large Language Models (LLMs), reasoning engines capable of intelligent decision-making and planning over complex tasks [20]. LIDDIA [11] is one of the first such systems: a *single*-agent capable of autonomously navigating drug discovery *in silico*. Following LIDDIA, several systems have been introduced, with more attention towards *multi*-agent systems [12, 14, 15, 18, 19]. Some of them are designed for a specific task under the drug discovery pipeline, while others for end-to-end drug discovery [14, 18].

While promising, these agentic systems leave room for improvements. Single-agent systems such as LIDDIA [11] and AgentD [19] demonstrate how LLMs can intelligently coordinate different *in silico* tools for drug discovery, yet their performance does not scale well to more realistic and complex tasks (Section 4.1). Multi-agent systems such as PharmAgents [14], Prompt-to-Pill [18] and F.A.D.E. [17] are designed for complex tasks, but they typically assume a predefined workflow. This assumption does not reflect real-world drug discovery, which is a highly non-linear process governed by many requirements. For instance, successful drugs must bind well to the target, while maintaining favorable pharmacokinetic and physicochemical property profiles [21]. However, identifying such molecules is highly difficult, requiring numerous iterations of searching the expansive chemical space [22]. Other multi-agent systems have also been proposed that enable more flexible workflows but are still limited in adaptability [13, 16].

Building upon these insights, we propose MADDIE, a **Multi-Agent Drug Discovery Engine**. As illustrated in Figure 1, the system is composed of five agents, a central agent called **(1) ORCHESTRATOR**, and four specialized worker agents: **(2) HITIDENTIFIER**; **(3) LEADOPTIMIZER**; **(4) SYNTHESIZER**; and **(5) EVALUATOR**. ORCHESTRATOR serves as an intelligent orchestration component that governs the whole campaign. Unlike static workflows, ORCHESTRATOR dynamically determines the workflow through its actions, sustains a long-horizon campaign using its memory module and monitors the progress by maintaining its evolving repository of candidate molecules. Under its orchestration, MADDIE’s worker agents are composed into

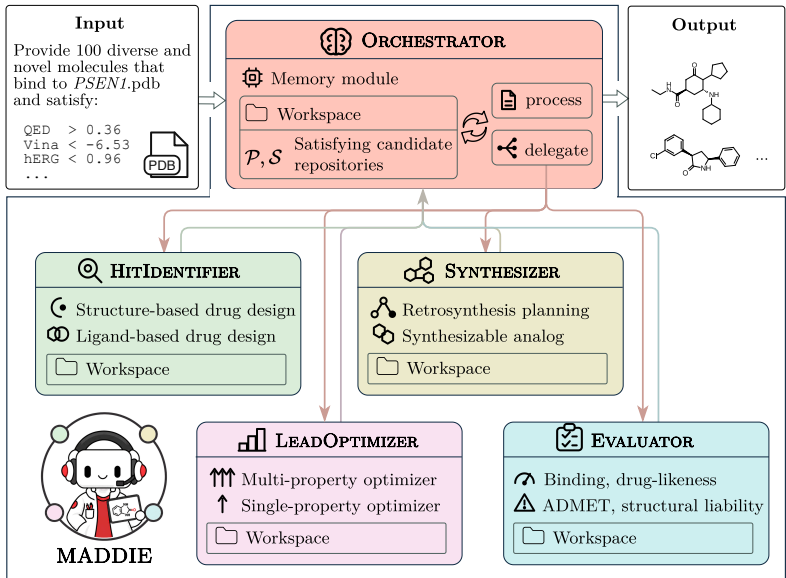


Fig. 1: MADDIE architecture, consisting of ORCHESTRATOR, workspace and four worker agents: HITIDENTIFIER, LEADOPTIMIZER, SYNTHESIZER and EVALUATOR.

flexible workflows, enabling MADDIE to adapt its strategy as a campaign unfolds, mirroring the iterative, decision-driven nature of real-world drug discovery. Each worker agent carries out a step critical in advancing a drug discovery campaign—from identifying hits and optimizing them, to proposing synthesis routes and evaluating the quality of molecules. MADDIE’s worker agents are equipped with state-of-the-art *in silico* tools, including generative models for *de novo* molecular generations, thus allowing MADDIE to explore beyond known chemical space and propose novel molecules designed for each campaign’s objectives.

We rigorously benchmark MADDIE with a wide range of methods designed to accelerate drug discovery. We highlight several insights from MADDIE here: **(1)** MADDIE outperforms existing baselines by a large margin, generating molecules satisfying key pharmaceutical properties required for successful drugs across 50 targets, along with providing their synthesis routes (Section 4.1). This is consistent with our motivation for the design of MADDIE: an autonomous system that can scale to realistic tasks and adapt throughout a campaign to generate satisfying molecules across targets; **(2)** MADDIE’s performance arises from the collaborative efforts of each agent, emphasizing their importance in contributing to the satisfying molecules (Section 4.2). Each worker agent makes a distinct contribution, while ORCHESTRATOR contributes the most. Notably, ORCHESTRATOR tracks the campaign through its evolving repository of molecules and directs MADDIE’s effort towards the most promising ones, substantially reducing the number of molecules MADDIE must explore to reach its objective; **(3)** MADDIE demonstrates its capability to adapt its workflow during campaigns for varying targets, employing distinct strategies required to produce

the satisfying molecules (Section 4.3). Even on a target with no existing marketed FDA-approved drugs [23], MADDIE can identify more than 100 promising molecules through its intelligent adaptation, emphasizing its capability; **(4)** MADDIE generates one promising molecule for PSEN1, a critical target associated with Alzheimer’s Disease, with the molecule’s *in silico* properties exceeding those of the reference ligand and the associated FDA-approved drug (Section 4.4); and **(5)** MADDIE serves as a low-cost and efficient autonomous drug discovery system, highlighting its potential to accelerate drug discovery (Section 4.5). Together, these findings indicate that dynamic orchestration of specialized worker agents through a central agent offers an effective design for autonomous drug discovery, and MADDIE can serve as a promising system in this domain.

2 Related Works

2.1 Agents for Drug Discovery

Agents powered by LLMs are capable of coordinating *in silico* tools across broader drug discovery workflows. Existing systems operate at different levels of scope and autonomy, from task-specific assistants to end-to-end drug discovery. These systems are typically classified into two categories: **(1)** single-agent system and **(2)** multi-agent system.

CACTUS [24] focuses on assistive cheminformatics tasks such as estimating molecular properties, retrieving structurally similar molecules, and evaluating drug-likeness. TxGemma [15] also proposes a conversational assistant for therapeutics predictions and question answering, allowing better interaction with medicinal chemists. AgentD [19] spans a wider range of the early-stage computational pipeline in drug discovery, from target and ligand information retrieval, to molecular generation, property prediction, and refinement. LIDDIA [11] provides an autonomous workflow where LLM-based reasoning guides the iterative process of finding satisfying molecules that fit a specific profile. These examples demonstrate that LLMs can coordinate *in silico* tools toward a larger drug discovery goal. However, their performance is typically limited and does not scale well for more complex and realistic tasks (Section 4.1).

Multi-agent systems distribute their tasks across specialized agents, enabling them to tackle complex problems, such as drug discovery, that can exceed the reach of any single-agent system. Their system designs and assigned roles for the agents vary among systems. DrugAgent [12] coordinates planner and instructor agents to develop, implement, and refine machine-learning solutions using drug discovery domain knowledge. Robin [25] assigns literature-search and data-analysis to separate agents, using their output to loop through hypothesis formulation, experiment planning, and result interpretation. More directly focused on target binding site identification and candidate generation, F.A.D.E. [17] uses a hierarchical multi-agent setup to produce ranked candidates and demonstrates its effectiveness through case studies on targets EGFR and CRBP1. Prompt-to-Pill [18] and PharmAgents [14] work at a broader scope, coordinating agents across drug discovery phases, from preclinical stage to clinical-trial simulation. Mozi [16] proposes a dual-layer multi-agent architecture for drug discovery, a layer for control and governance and another one for workflow. FROGENT [13]

shows an example of a decentralized multi-agent system, using an orchestrator agent that decomposes the task into a high-level plan, and the other agents to execute each subtask in the plan. Together, these systems demonstrate the overall capability of collaboration between different specialized agents on drug discovery tasks. However, their coordination generally centers on predefined workflows, and their system evaluations often focus on individual case studies. This is in contrast to MADDIE, which enables flexible workflow adaptation across different campaigns (Section 4.3).

2.2 Generative Models for Drug Design

Generative models have recently been employed for *de novo* drug design to accelerate drug discovery. These models generally fall under two categories: (1) structure-based drug design (SBDD) and (2) ligand-based drug design (LBDD).

SBDD methods generate ligand molecules conditioned on the three-dimensional structures of target binding pockets. Early approaches such as Pocket2Mol [26] use autoregressive sampling to generate molecular atoms sequentially, while more recent approaches are diffusion-based [27, 28]. Notably, conDitar-dev [10] decouples protein encoding from the diffusion process through a separately pretrained pocket encoder and performs inference-time noise optimization to improve molecular properties beyond binding affinity. Several works augment SBDD models with LLM-based refinement [29–31]. ELILLM [31], one of the state-of-the-art methods, encodes the initial molecules into the LLM latent space and performs Bayesian optimization to identify promising molecular representations.

LBDD generates molecules conditioned on the shape of the ligand that binds to the target structure’s pocket. Previous approaches include SQUID [32], using a fragment-based generative model to decode fragments and a scoring model to adjust the angles of rotatable bonds between fragments, and DESERT [33], an image-to-sequence model with an encoder-decoder architecture that generates molecular structures conditioned on voxelized shapes. DiffSMol [8], one of state-of-the-art methods for LBDD, encodes the ligand shape into pretrained embeddings and uses these embeddings to condition a diffusion-based molecular generation model.

Together, these generative models provide effective means of finding promising molecules. Generation alone, however, is only the first step of drug discovery, and the molecules these models produce often demand substantial manual downstream filtering and selection [10, 34]. MADDIE addresses this gap by integrating the filtering and selection into the system itself: HITIDENTIFIER and LEADOPTIMIZER propose the molecules, while EVALUATOR and SYNTHESIZER assess them—all under ORCHESTRATOR’s coordination.

3 Experimental Settings

Dataset

We curate a dataset to evaluate MADDIE. Concretely, we collect (1) a set of 50 protein targets, along with their 3D binding pocket structures, and (2) a set of drugs approved by U.S. Food and Drug Administration (FDA), each is associated with a target in our dataset. These drugs are used to define what molecular property

requirements MADDIE’s generated molecules should meet (Table 1). We provide the full protein targets, their annotations, the curated FDA-approved drugs, and the protocol to collect them in this paper (Appendix A).

The targets span major therapeutic areas including autoimmune disease, cancer, cardiovascular disease, diabetes, infectious disease, and neurological conditions. The dataset includes 42 targets from existing datasets: **(1)** LIDDIA [11] (12 targets) and **(2)** CrossDocked2020 [35] (30 targets), and additional 8 targets from OpenTargets [36]. We ensure that these targets do not exist in CrossDocked2020 training set, thus allowing MADDIE to be compared against methods trained on CrossDocked2020. We use the experimentally validated 3D structures available [37] and preprocess them to extract the binding pockets, similar to LIDDIA’s dataset preprocessing [11].

For each target, we identify whether it has drugs that are **(1)** small molecules, **(2)** FDA-approved, and **(3)** currently marketed as of July 2026. Out of 50 targets, 22 targets are associated with at least one drug. For the remaining targets (28), no drug associations are identified, indicating opportunities for drug development for these targets. We use these drugs to calculate the property requirements for our experiment (Appendix B).

Evaluation Metrics

We use several metrics to evaluate the performance of MADDIE. These metrics are based on **(1)** the molecular properties often considered in real-world drug discovery, **(2)** the synthesizability of the molecule, and **(3)** whether the molecule can satisfy multiple property and synthesizability requirements simultaneously.

We evaluate the generated molecules on several key properties critical for a successful drug. We first consider properties commonly used in the literature [10]: **(1)** binding affinity, measured by Vina docking score [5] (**Vina**); **(2)** drug-likeness, or desirability of a molecule using quantitative estimate of drug-likeness [38] (**QED**); and **(3)** synthesizability estimate, using synthetic accessibility score [39] (**SAS**). In addition to common properties, we further assess the toxicities of these molecules. Concretely, we measure the probability that a molecule may **(1)** induce mutations in DNA [40] (**Mut**), **(2)** fail clinical trials owing to toxicity [41] (**CTx**), and **(3)** inhibit the hERG potassium channel [42] (**hERG**). Finally, we apply Lilly medicinal filters [43] (**Lly**) to flag whether the molecules have undesirable traits, such as interfering with assay measurements (e.g., fluorescence) or lacking druggability (e.g., molecules lacking both oxygen and nitrogen). We assess the synthesizability of a molecule by determining whether a synthesis route can be generated (**Syn**). We employ the same approach as SYNTHESIZER in MADDIE to measure the synthesizability of the molecule (Section 7.2.3). This is a more rigorous assessment than synthesizability estimate (**SAS**; Section 7.2.4).

For every molecular property and synthesizability defined previously, we identify a threshold as a requirement for a molecule to meet. These thresholds are determined using existing FDA-approved drugs curated in our dataset as references (Appendix B), ensuring that they are realistic rather than arbitrary cutoffs. We define a molecule as *satisfying* only if it *jointly* meets all requirements.

Table 1: The thresholds used in the experiments and whether each method can measure the corresponding properties and synthesizability during generation (✓ indicates it can). The thresholds are calculated using marketed FDA-approved drugs in our dataset (Appendix B).

Method	Vina	QED	SAS	Mut	CTx	hERG	Lly	Syn
	<-6.53	>0.36	<5.35	<0.51	<0.71	<0.96	True	True
<i>Agents for Drug Discovery</i>								
MADDIE (ours)	✓	✓	✓	✓	✓	✓	✓	✓
LIDDIA [11]	✓	✓	✓	—	—	—	—	—
Prompt-to-Pill [18]	✓	✓	—	✓	—	✓	—	—
AgentD [19]	—	✓	—	—	—	—	—	—
<i>Generative Models for Drug Design</i>								
Pocket2Mol [26]	—	—	—	—	—	—	—	—
TargetDiff [27]	—	—	—	—	—	—	—	—
conDitar [10]	—	—	—	—	—	—	—	—
DiffSMol [8]	—	—	—	—	—	—	—	—
ELILLM [31]	—	—	—	—	—	—	—	—
<i>Generalist LLMs</i>								
Nemotron-3-Ultra [44]	—	—	—	—	—	—	—	—
Qwen3-30B [45]	—	—	—	—	—	—	—	—

We report the performance of MADDIE using several metrics: **(1)** N_{tout} , the number of targets for which at least one output molecule is produced, as a measure of MADDIE’s coverage in delivering molecules for the targets; **(2)** N_{out} and N_{sat} , the average number of output and satisfying molecules per target, respectively; **(3)** %Pass, the average percentage of satisfying molecules among the output molecules per target; and **(4)** the per-requirement pass rates (%Vina, %QED, %SAS, %Mut, %CTx, %hERG, %Lly, %Syn), by measuring the average percentage of output molecules satisfying each respective requirement per target.

Baselines

To evaluate the effectiveness of MADDIE, we compare it with a wide range of baselines, each cast as a *generate-then-filter* pipeline: a molecular generation method followed by a filtering stage. The molecule generation methods span three categories: **(1)** *Agents for Drug Discovery*, **(2)** *Generative Models for Drug Design* and **(3)** *Generalist LLMs*. The molecules generated from each method are then filtered against each requirement, if the method cannot assess the corresponding property or synthesizability during generation. Table 1 presents which property or synthesizability each method can assess during its generation. Every molecule from generative models for drug design and generalist LLMs is filtered against *every* requirement after generation. In contrast, molecules from agents for drug discovery baselines are filtered only against the requirements that cannot be assessed by the method during generation. Only molecules that pass the filtering are considered as the output molecules of that method. Coupling a generation method with a filtering stage enables a fair comparison with MADDIE, while also being a widely adopted paradigm in drug discovery [10, 34].

We use a variety of methods for each category of the baselines. **(1)** For *Agents for Drug Discovery*, we compare against two state-of-the-art single-agent systems, LIDDIA [11] and AgentD [19], and one multi-agent system, Prompt-to-Pill [18]. These agents are designed for end-to-end drug discovery workflow, similar to ours, and serve as direct points of comparison for MADDIE. They are also selected since their implementations are publicly accessible (Appendix C). **(2)** For *Generative Models for Drug Design*, we compare against a wide variety of models: Pocket2Mol [26], TargetDiff [27], conDitar [10], DiffSMol [8] and ELILLM [31]. These models include both SBDD (Pocket2Mol, TargetDiff, conDitar, ELILLM) and LBDD (DiffSMol) approaches. **(3)** For *Generalist LLMs*, we compare against two open-source LLMs, Qwen3-30B [45] and Nemotron-3-Ultra [44]. We focus our comparison of MADDIE to open-source LLMs for reproducibility and accessibility: their weights are publicly available and are less likely to be discontinued than proprietary LLMs. Each LLM is competitive on general reasoning and scientific benchmarks [44, 45], making them strong baselines. Qwen3-30B is also used as the reasoning engine for MADDIE, while Nemotron-3-Ultra (550B) is among the largest and state-of-the-art open-source LLMs available. We only select two due to our experimentation budget.

Each method is instructed to produce the same fixed number of output molecules per target as MADDIE for a fair comparison. Methods that cannot be instructed in this way, including the five generative models for drug design and AgentD, are instead instructed to produce as many *intermediate* molecules as MADDIE produces, per target. Specifically, we define an *intermediate* molecule as any molecule that HITIDENTIFIER, LEADOPTIMIZER, and SYNTHESIZER produce during a campaign, whether or not they serve as an output molecule. The number of *intermediate* molecules is used as a basis for how many molecules the generative models and AgentD should generate, and also for analysis (Section 4.2). This ensures that these baselines receive the same opportunities as MADDIE to explore the chemical space (by generating the same number of molecules), thus supporting a fair comparison.

Implementation Details

In MADDIE, we adopt Qwen3-30B [45] as the backbone LLM for ORCHESTRATOR and all four worker agents, an open-source LLM with strong performance across varying tasks while also small enough to be deployed locally. The model is deployed with vLLM [46] and decoded with a temperature of 0.7, a top- p of 0.8, and a context window of 64K tokens. These are the recommended parameters for Qwen3-30B [45]. We also provide implementation details for the baselines (Appendix D), along with more details on MADDIE (Appendix E).

4 Experimental Results

4.1 Overall Performance

Table 2 reports the performance of MADDIE and the baselines. For each target, every method is instructed to produce 100 output molecules, a number large enough for further exploration (e.g., downstream selection for wet-lab validation) and small enough to keep evaluation tractable. In this setting, we compute the metrics only on

Table 2: Main experimental results of MADDIE against baselines.

Method	N_{tout}	N_{out}	N_{sat}	%Pass	%Vina	%QED	%SAS	%Mut	%CTx	%hERG	%Lly	%Syn
<i>Agents for Drug Discovery</i>												
MADDIE	50	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
LIDDIA	49	36.0	28.0	80.0	83.1	96.1	100.0	100.0	100.0	100.0	100.0	100.0
Prompt-to-Pill	31	14.2	2.9	20.9	55.0	88.5	100.0	41.0	100.0	98.1	100.0	100.0
AgentD	35	44.2	44.2	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
<i>Generative Models for Drug Design</i>												
Pocket2Mol	50	33.1	33.1	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
TargetDiff	48	15.8	15.8	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
conDitar	49	56.7	56.7	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
DiffSMol	27	24.7	24.7	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
ELILLM	27	4.5	4.5	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
<i>Generalist LLMs</i>												
Nemotron-3-Ultra	38	23.2	23.2	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Qwen3-30B	3	8.0	8.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

targets for which a method produces at least one output molecule (e.g., generated molecules after filtering for baselines), ensuring fairness across methods. We also compare the property values of molecules generated by MADDIE against baselines, the reference ligands, and our curated FDA-approved drugs (Appendix F.1).

Only MADDIE delivers the requested 100 satisfying molecules on all 50 targets. The closest baselines fall well short: conDitar delivers 56.7 satisfying molecules over the 49 targets it covers, Pocket2Mol 33.1 over 50, and AgentD 44.2 over 35. This demonstrates MADDIE’s advantage in consistently finding satisfying molecules across targets. Such consistency arises from MADDIE’s multi-agent design: MADDIE can adaptively generate, optimize, and evaluate according to what the current molecules lack, using its ORCHESTRATOR to decide which worker agent to delegate next. This forms a dynamic decision-making loop that improves molecules over time, thereby succeeding on a variety of targets.

Agentic baselines employ evaluation tools during generation but still leave requirements unsatisfied. As illustrated in Table 1, each agent is capable of measuring some properties for the requirements: LIDDIA measures Vina, QED and SAS during its campaign, Prompt-to-Pill measures Vina, QED, Mut and hERG, and AgentD measures QED. For LIDDIA and Prompt-to-Pill, however, the per-requirement pass rates still fall short on exactly these properties, 83.1% on Vina for LIDDIA and 41.0% on Mut for Prompt-to-Pill. These failures likely arise from the reasoning engine powering these agents: the LLMs themselves. LLMs may misinterpret the tool outputs or hallucinated, leading it to accept a molecule that does not satisfy the requirements—a well-known weakness of LLM-based agent systems [47]. In contrast, MADDIE addresses this weakness by accepting a molecule only after it has been evaluated by EVALUATOR and SYNTHESIZER (Section 7).

Generative models for drug design achieve perfect passing rates by filtering, but retain far fewer molecules than requested. For these baselines, filtering is applied against every requirement after generation, and molecules failing any requirement are discarded, leading to 100% pass rates for every target (with at

least one output molecule). Filtering, however, cannot improve on what the methods produce: conDitar and Pocket2Mol retain only 56.7 and 33.1 satisfying molecules on average, respectively, both well below the requested 100. When the method rarely produces molecules meeting the requirements, filtering may retain none of them, as with DiffSMol for 23 of the targets. This exposes a fundamental limitation for methods without feedback from tools during generation: the pass rate can always be enforced to 100% through filtering, but the effective yield, or the number of satisfying molecules cannot be improved.

Generalist LLMs produce very few satisfying molecules. For instance, Nemotron-3-Ultra, one of the state-of-the-art open-source LLMs [44], produces on average only 23.2 satisfying molecules on 38 targets. Qwen3-30B, the LLM that drives every agent in MADDIE, covers only 3 targets on its own, emphasizing that the gap to the 50 reached by MADDIE is attributable to its design rather than solely to the capability of the underlying language model.

4.2 Ablation Study

We conduct ablation experiments to assess the effectiveness of the different agents in the MADDIE framework. Specifically, we remove SYNTHESIZER (Section 7.2.3), LEADOPTIMIZER (Section 7.2.2) and ORCHESTRATOR (Section 7.1) individually, resulting in three variants of MADDIE: **(1)** *w/o* SYNTHESIZER, without the capability to assess synthesizability; **(2)** *w/o* LEADOPTIMIZER, which cannot optimize any property; and **(3)** *w/o* ORCHESTRATOR, which only allows a fixed workflow during a campaign, respectively. In MADDIE *w/o* ORCHESTRATOR, we set the workflow such that each agent is invoked sequentially in this order: HITIDENTIFIER, LEADOPTIMIZER, EVALUATOR, then SYNTHESIZER. The workflow for MADDIE *w/o* ORCHESTRATOR is repeated until the requested number of satisfying molecules is obtained. We do not remove HITIDENTIFIER as it serves as the starting point of a campaign and EVALUATOR as every candidate molecule in MADDIE is required to pass that worker agent. We measure N_{itm} , the average number of intermediate molecules generated per target, and we provide the results in Table 3.

SYNTHESIZER is responsible for more than half of MADDIE’s satisfying molecules. MADDIE *w/o* SYNTHESIZER generates only 40.2 instead of 100 satisfying molecules, mostly because they fail the synthesis requirement (%Syn). SYNTHESIZER also generates synthesizable analogs for candidate molecules without a route, and removing it causes HITIDENTIFIER and LEADOPTIMIZER to make up the shortfall by generating and optimizing more candidates. This is shown by the number of intermediate molecules generated: 1,554.8 molecules as opposed to 1,043.8 with SYNTHESIZER on average.

LEADOPTIMIZER reduces the number of intermediate molecules MADDIE generates by nearly 30%. Without it, the number increases from 1,043.8 to 1,327.7 as presented in Table 3. In MADDIE *w/o* LEADOPTIMIZER, a candidate molecule that fails a requirement cannot be improved further, forcing this variant to rely on HITIDENTIFIER for generation. In addition, for challenging targets that HITIDENTIFIER struggles with (e.g., ABCD1_ALD), MADDIE *w/o*

Table 3: Ablation study.

Method	N_{tout}	N_{out}	N_{sat}	%Pass	%Syn	N_{itm}
MADDIE	50	100.0	100.0	100.0	100.0	1,043.8
<i>w/o</i> SYNTHESIZER	50	100.0	40.2	40.2	40.2	1,554.8
<i>w/o</i> LEADOPTIMIZER	49	100.0	100.0	100.0	100.0	1,327.7
<i>w/o</i> ORCHESTRATOR	50	100.0	100.0	100.0	100.0	1,743.8

LEADOPTIMIZER cannot deliver the requested molecules, resulting in only covering 49 instead of 50 targets.

ORCHESTRATOR *greatly decreases the number of molecules explored to find the satisfying molecules*. A campaign without ORCHESTRATOR requires generating 1,743.8 molecules on average, around 70% more than with ORCHESTRATOR, to reach the same goal of 100 molecules satisfying all requirements. This gap reflects the role of ORCHESTRATOR: rather than generating molecules blindly, ORCHESTRATOR intelligently uses feedback from its prior decisions and accumulated molecules to decide what to generate next, concentrating its effort on molecules likely to satisfy the requirements. As a result, fewer molecules are required to explore, leading to a more efficient and effective method.

Taken together, these results show that every agent contributes by reducing the amount of exploration a campaign requires. They differ, however, in their role of reducing the intermediate molecules: **(1)** SYNTHESIZER generates synthesizable analogs, allowing other agents to focus on the other requirements; **(2)** LEADOPTIMIZER improves existing candidates, providing a more efficient exploration for MADDIE; and **(3)** ORCHESTRATOR intelligently adapts to what existing molecules require, enabling MADDIE to focus on the most promising candidate molecules.

4.3 Campaign Dynamics Analysis

We analyze how a campaign unfolds in MADDIE: first through the transition probabilities across the worker agents aggregated over all the targets (Section 4.3.1), then through the flow of molecules from the worker agent that produces them to the output (Section 4.3.2), and finally through the trajectories of the worker agents on representative targets (Section 4.3.3).

Each transition below is a delegation made by ORCHESTRATOR, and the agent itself is excluded in these figures as it serves as the central agent to which all worker agents must report. Only the four specialized worker agents are presented.

4.3.1 Overall Transition Probabilities

Figure 2 shows the empirical transition probabilities among the worker agents, calculated from the delegation decisions ORCHESTRATOR makes over the 50 targets. Each edge represents a transition probability.

EVALUATOR *is the most frequently involved worker agent during a campaign*. After being invoked, HITIDENTIFIER goes to EVALUATOR with probability 0.97, LEADOPTIMIZER with probability 1.00 and SYNTHESIZER with probability 0.82. Every molecule that is generated, either by HITIDENTIFIER or SYNTHESIZER (e.g.,

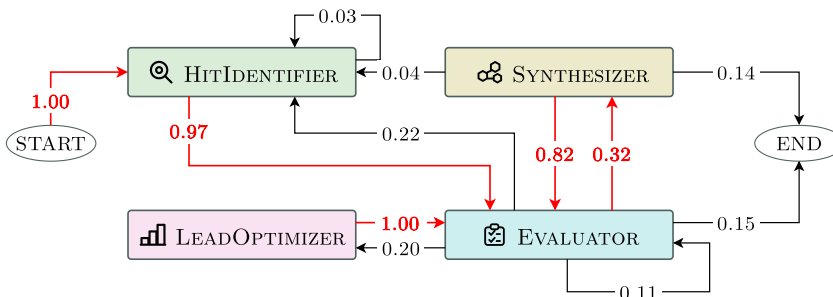


Fig. 2: Transition probabilities among worker agents during a campaign. Red color represents the transition with the highest probability out of each worker agent.

synthesizable analogs), or optimized is therefore assessed before it is processed any further. Any decision made in MADDIE on which molecules to progress is attributed to the evaluated properties from EVALUATOR, emphasizing its role as a quality checkpoint for MADDIE.

MADDIE adapts its campaign based on feedback from EVALUATOR. In Figure 2, transitions out of EVALUATOR (except to itself) are almost equal in probability across worker agents. Following EVALUATOR, the campaign returns to HITIDENTIFIER to seek new molecules with probability 0.22, moves to LEADOPTIMIZER to optimize one or more deficient properties with probability 0.20, or moves to SYNTHESIZER to search for a synthesis route with probability 0.32. Since no transition probability dominates (out of EVALUATOR), a typical campaign in MADDIE does not follow a fixed trajectory after evaluation. Instead, MADDIE adapts, invoking the next worker agent according to EVALUATOR’s feedback throughout a campaign.

Every campaign is guardrailed by either EVALUATOR or SYNTHESIZER before it finishes. A campaign never finishes until either of them is called at the end. This behavior follows similarly to real-world drug discovery, since the last step of a campaign is typically an assessment of properties and synthesizability, ensuring that the molecules are grounded in evidence.

4.3.2 Molecule Flow Across Agents

Figure 3 traces the molecules produced by HITIDENTIFIER, LEADOPTIMIZER, and SYNTHESIZER during a campaign. The figure illustrates how each worker agent contributes to the satisfying molecules, and how these worker agents collaborate to achieve those molecules.

HITIDENTIFIER generates the most molecules but contributes the fewest satisfying ones. HITIDENTIFIER sends 541.4 molecules to EVALUATOR on average (Figure 3a), more than twice as many as LEADOPTIMIZER and SYNTHESIZER, which send 241.5 (Figure 3b) and 260.9 (Figure 3c), respectively. Yet only 22.5 of them end up satisfying all the requirements, compared with 33.4 and 72.8 for the other two agents, respectively. This contrast reflects a division of effort among these agents: HITIDENTIFIER explores, providing a wide range of candidates other agents can use,

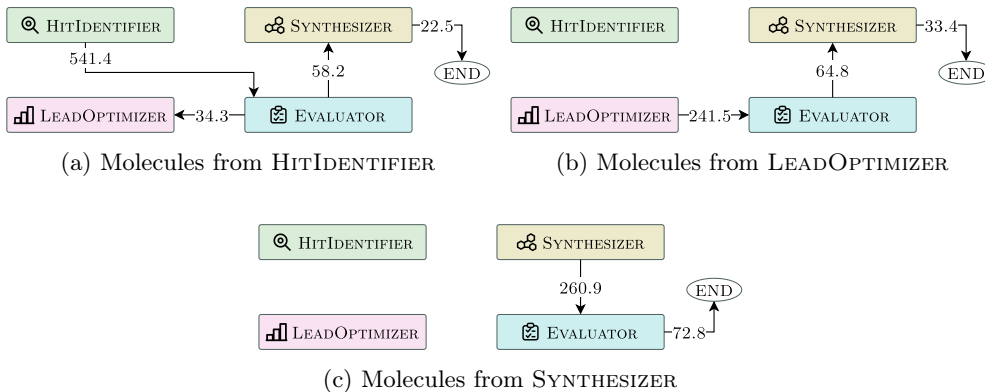


Fig. 3: Flow of molecules from the worker agent at which they originate to the campaign output. Each subfigure traces the molecules originating from one worker agent; each edge gives the number of those molecules subsequently passed to another worker agent or reaching the output, averaged over the 50 targets.

while LEADOPTIMIZER and SYNTHESIZER exploit, using these candidates to generate molecules more likely to meet the requirements. It is through this coupling of exploration and exploitation that MADDIE can produce satisfying molecules in its campaign.

The satisfying molecules are drawn from all three agents, with SYNTHESIZER contributing the largest share. The three contributions are not independent: LEADOPTIMIZER optimizes the molecules that HITIDENTIFIER generates, and SYNTHESIZER generates its analogs from molecules that HITIDENTIFIER and LEADOPTIMIZER produce, after being assessed by EVALUATOR. SYNTHESIZER produces the most satisfying molecules because it often acts last, generating analogs from candidate molecules that already satisfy most requirements. Each agent thus contributes to the final molecules, either directly or through another agent.

4.3.3 Agent Execution Across Campaigns

We further examine campaigns for different targets in MADDIE to examine how they unfold. We also provide the number of molecules stored in $\mathcal{P}$ and $\mathcal{S}$ under ORCHESTRATOR (Section 7), representing the number of molecules satisfying (1) all requirements except synthesizability, and (2) all requirements. During a campaign, MADDIE can generate more satisfying molecules than requested; in that case, it returns only the first satisfying molecules generated, up to the number requested.

MADDIE adaptively employs different strategies throughout a campaign. Figure 4a follows a campaign against ABCD1_ALD, a challenging target [23] with no marketed FDA-approved drugs as of July 2026. Over successive rounds of delegation (by ORCHESTRATOR), generation (by HITIDENTIFIER), optimization (by LEADOPTIMIZER), and assessment (by EVALUATOR and SYNTHESIZER), MADDIE identifies a total of 136 satisfying molecules, demonstrating its capability. We highlight several patterns of MADDIE in this campaign: **(1)** the most common strategy is *generate-then-optimize*, which invokes HITIDENTIFIER followed by LEADOPTIMIZER,

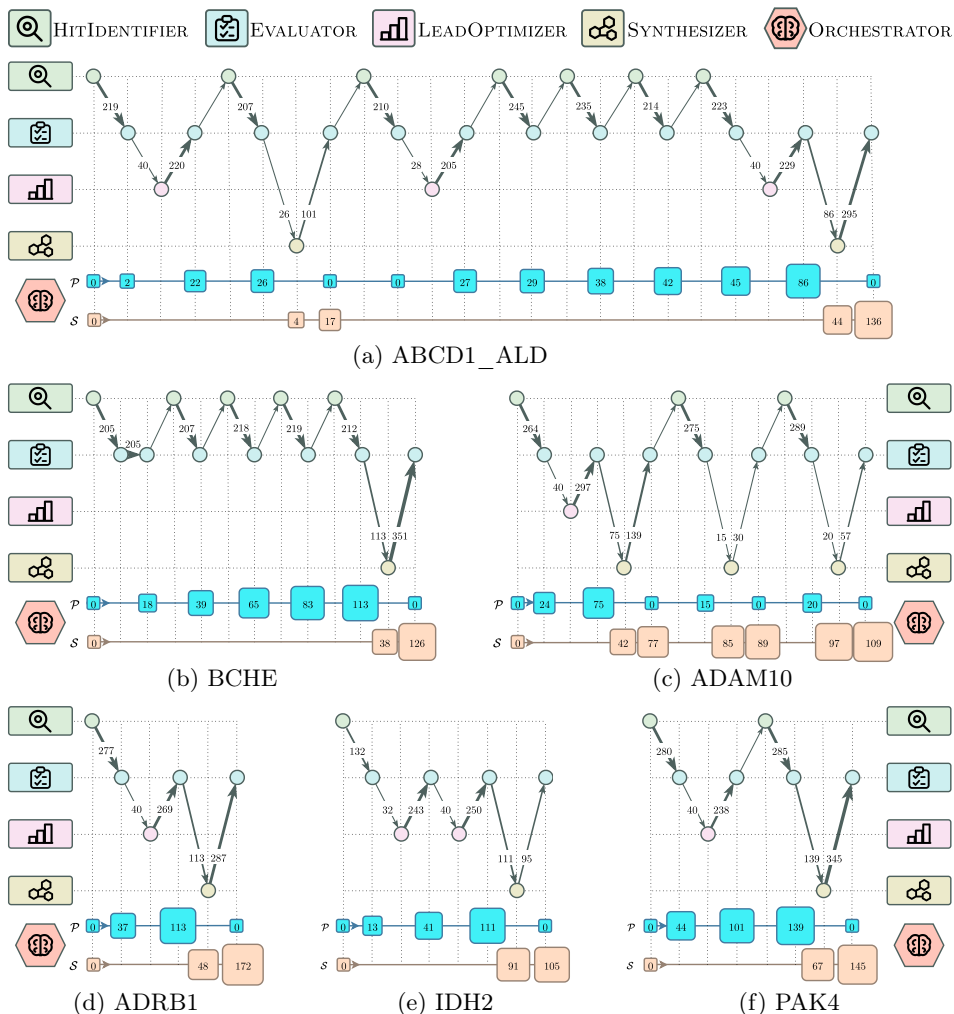


Fig. 4: Trajectories of MADDIE over six representative targets: (a) ABCD1_ALD, (b) BCHE, (c) ADAM10, (d) ADRB1, (e) IDH2, and (f) PAK4. Rows represent the agents in MADDIE and columns represent one delegation or step during the campaign. The number on the arrow denotes the number of molecules passed to the next agent. The number on the box denotes the number of molecules stored in each repository after the delegation. The arrow width (worker agents) and box size (ORCHESTRATOR) are proportional to the number of molecules.

with EVALUATOR assessing the molecules in between generations. MADDIE applies this strategy twice in the first half of the campaign and once near the end; (2) ORCHESTRATOR adapts its strategy throughout the campaign. Most notably, ORCHESTRATOR employs *generate-then-optimize* during the first half, observes the results (from the molecules stored in $\mathcal{P}$ and $\mathcal{S}$), then adopts a new strategy, *multi-generate-then-optimize*, which calls HITIDENTIFIER multiple times before optimizing

via LEADOPTIMIZER at the end (with EVALUATOR in between). This strategy dominates the second half of the campaign, accumulating 86 property satisfying molecules in $\mathcal{P}$; (3) EVALUATOR is called after every generation—whether by HITIDENTIFIER, LEADOPTIMIZER or SYNTHESIZER—emphasizing its importance; (4) SYNTHESIZER is the least frequently called agent (only twice). However, it is only after SYNTHESIZER does $\mathcal{S}$ increase: SYNTHESIZER’s synthesis analog generation drives the sharp increase in satisfying molecules, from 44 to 136, thus ending the campaign. This highlights SYNTHESIZER as a critical agent despite its infrequent use. Together, the campaign illustrates how worker agents, coordinated through ORCHESTRATOR’s feedback-driven decisions, can solve a difficult task—generating 100 satisfying molecules for a target with no existing marketed FDA-approved drugs.

MADDIE dynamically adapts its campaign to different targets. We present five more campaigns, each of which MADDIE approaches with distinct strategies. (1) The campaign on BCHE [48, 49], presented in Figure 4b, is notable in that LEADOPTIMIZER is never called: MADDIE relies solely on HITIDENTIFIER to generate molecules, with SYNTHESIZER invoked only at the end to produce synthesizable analogs. (2) For target ADAM10 [50], shown in Figure 4c, MADDIE pursues two kinds of strategies: HITIDENTIFIER followed by LEADOPTIMIZER and then SYNTHESIZER, and HITIDENTIFIER followed directly by SYNTHESIZER. Each strategy is interleaved with EVALUATOR between successive generations. The two strategies differ in whether LEADOPTIMIZER is invoked, suggesting that ORCHESTRATOR observes whether the molecules require optimization and adapts accordingly. (3) ADRB1, IDH2, and PAK4, each illustrated in Figure 4d, 4e, and 4f respectively, follow shorter trajectories with a similar strategy, all beginning with HITIDENTIFIER and then LEADOPTIMIZER, but diverging after: ADRB1 proceeds directly to SYNTHESIZER and finishes, while IDH2 performs an additional round of optimization, and PAK4 an additional round of generation, before each concludes with a final SYNTHESIZER call. Together, these campaigns show that MADDIE does not execute a fixed workflow, but instead, it adapts, mirroring the adaptive workflows a medicinal chemist typically employs in real-world drug discovery. We discuss additional representative trajectories on two targets, ABO_BGAT and HTR2A (Appendix F.2).

4.4 Case Study on PSEN1

To assess MADDIE’s generated molecules beyond computational predictions, we ask a domain expert to evaluate them. We focus on three targets associated with Alzheimer’s disease: PSEN1 [51], ACHE [52], and BCHE [53]. We screen the molecules according to two criteria before expert evaluation. The criteria are (1) whether their *in silico* properties exceed those of reference ligands and at least one of the associated FDA-approved drugs for the target, and (2) whether their structures are chemically unstable through manual validation. We discuss the results for PSEN1 here, and the remaining targets are illustrated in Appendix G.

On PSEN1, MADDIE proposes a molecule that surpasses both the co-crystallized ligand and the approved drug across all six in silico properties. PSEN1 encodes the catalytic subunit of Gamma-secretase, an enzyme complex

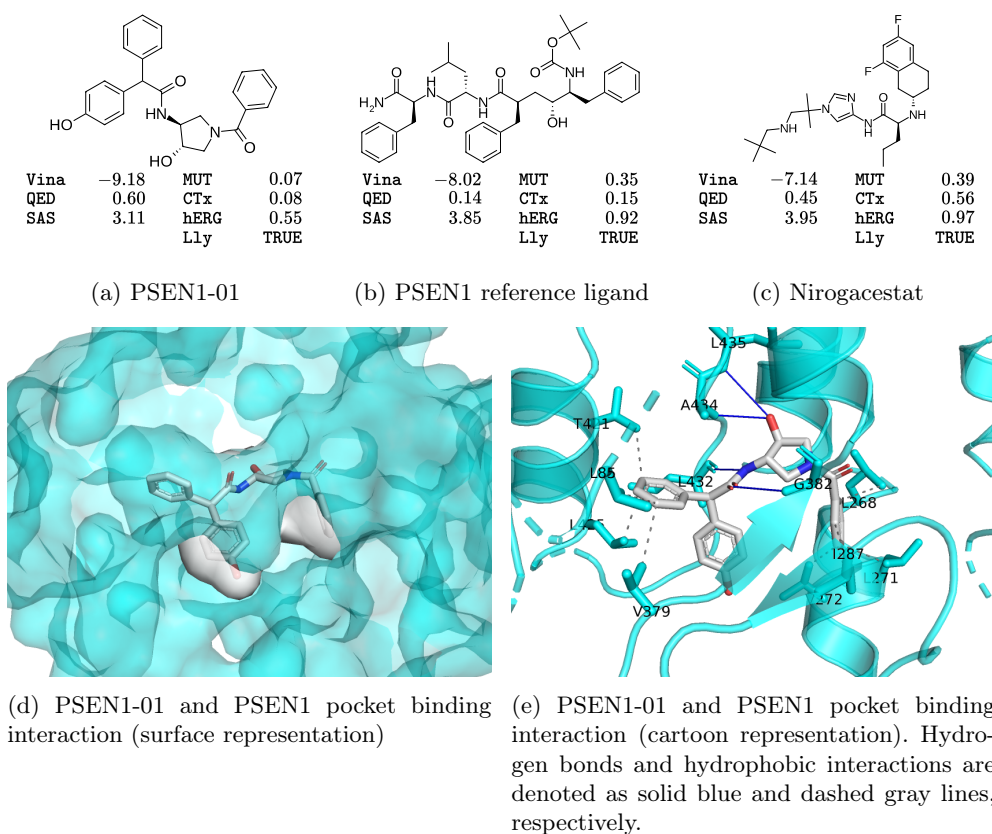


Fig. 5: PSEN1 case study.

linked to Alzheimer’s disease and certain tumors because it processes multiple critical membrane proteins, including the amyloid precursor protein (APP) and the cancer-related signaling protein [54]. Despite decades of medicinal chemistry [55], there is only one marketed FDA-approved drug (as of July 2026) associated with PSEN1, Nirogacestat—originally developed for Alzheimer’s disease, though currently it is used for the treatment of patients with desmoid tumors [56]. Against this target, MADDIE proposes 16 molecules with *in silico* properties better than either the co-crystallized reference ligand or Nirogacestat on binding affinity (Vina), drug-likeness (QED), synthesizability (SAS), and toxicity (Mut, CTx, hERG). We illustrate all 16 molecules in Figure A2. Notably, PSEN1-01 (Figure 5a) surpass both the reference ligand (Figure 5b) and Nirogacestat (Figure 5c) across all properties *in silico*, demonstrating its potential for this target.

PSEN1-01 interacts favorably with the binding site of PSEN1. Figure 5d and 5e illustrates the binding site and how PSEN1-01 interacts with it. PSEN1-01 buries the majority of its molecular surface into the site, with the 4-hydroxyphenyl ring left partially exposed. Notably, the molecule engages with the

two sequence motifs of PSEN1: the hydroxyl on its pyrrolidine ring accepts hydrogen bonds with A434 and L435 of the PAL motif [57], while the acetamide carbonyl accepts from G382 of the GxGD motif [58]. These hydrogen bonds are complemented by nine hydrophobic contacts that is distributed over two distinct subsites: the pendant phenyl against L85, V379, T421 and L425, and the benzoyl phenyl against L268, L271, V272 and I287 (2 contacts). Together, these interactions account for the binding improvement of PSEN1-01 over both the reference ligand and Nirogacestat, and show that it can satisfy the shape complementarity this binding site demands while still satisfying other key properties.

4.5 Computational Efficiency and Cost

We assess the efficiency of MADDIE along two dimensions: **(1)** the number of intermediate molecules generated to achieve the requested number of satisfying molecules (Section 4.5.1), and **(2)** the computational cost of a campaign, measured by running time and token consumption (Section 4.5.2).

4.5.1 Intermediate Molecules Explored by MADDIE

Table 4: Number of targets grouped by the number of intermediate molecules explored.

N_{itm}	Targets	Share (%)
< 750	10	20.0
750–1,000	18	36.0
1,000–1,250	9	18.0
1,250–1,500	8	16.0
$\geq 1,500$	5	10.0

Table 4 reports the number of intermediate molecules produced over the 50 targets. The targets are grouped into several categories based on the intermediate molecules, and we observe how many targets exist in these categories.

A majority (>50%) of the targets require less than 1,000 intermediate molecules to achieve the requested satisfying molecules. Averaged over all targets, MADDIE generates only 1,043.8 molecules to deliver 100 satisfying molecules, indicating that it explores the chemical space intelligently rather than through exhaustive enumeration. For the hardest 10% of targets, MADDIE generates more than 1,500 molecules (up to 2,601 molecules), yet still delivers the requested satisfying molecules, emphasizing its robustness on even challenging targets.

4.5.2 Computational Cost

We evaluate the computational cost of MADDIE by measuring **(1)** the running time and output token consumption, **(2)** the number of actions (e.g., *process* and *delegate*;

Table 5: Computational cost of MADDIE over 50 targets.

Metric	Cost		Action			Delegation per agent			
	Time (h)	Out. token	Total	Process	Delegate	HI	LO	EV	SY
Average	3.3	21,125	24.9	14.9	10.0	2.2	1.0	5.1	1.7
Shortest (ADAMTS5)	1.2	7,508	9	5	4	1	0	2	1
Longest (HDAC8)	11.6	30,448	44	38	6	1	1	3	1

Section 7.1) taken by ORCHESTRATOR, and **(3)** the number of delegations per worker agent. We also include the computational cost for the shortest and longest campaigns, and present the results in Table 5.

MADDIE serves as a fast and low-cost autonomous system for drug discovery. Across the 50 targets, MADDIE finishes a campaign in 3.3 hours and consumes 21,125 output tokens on average. MADDIE takes at least about one hour on ADAMTS5 [59] and almost half a day on HDAC8 [60], each represents the shortest and longest campaign of MADDIE, respectively. Relative to the satisfying molecules delivered, the cost of generating one satisfying molecules is roughly 2 minutes and 211 output tokens, a cost negligible against conventional drug discovery [3].

Half of the delegations are made to EVALUATOR. Out of 10.0 delegations issued per target on average, 5.10 go to EVALUATOR, while HITIDENTIFIER, SYNTHESIZER and LEADOPTIMIZER receive 2.24, 1.66 and 1.02, respectively. EVALUATOR is thus delegated the same amount of time as the remaining three agents combined, consistent with the ablation study presented previously (Section 4.2). The number of delegations, however, does not determine the campaign duration: ADAMTS5 and HDAC8 both takes 1.2 and 11.6 hours, respectively, yet they differ by only two delegations (4 and 6). This indicates that campaign cost is determined by the work performed within each delegation rather than the number of delegations issued.

5 Discussions

We discuss the current limitations of MADDIE and highlight several potential directions for future work.

In vitro validation for MADDIE’s generated molecules

MADDIE is an autonomous multi-agent system for *in silico* drug discovery— with dedicated agents equipped with state-of-the-art *in silico* tools to generate, optimize, evaluate, and plan the synthesis of molecules. MADDIE has demonstrated its ability to generate molecules that satisfy requirements grounded in these tools. However, such computational tools remain imperfect proxies for laboratory synthesis and *in vitro* experiments, which are indispensable next steps to validate the real-world potential of molecules generated by MADDIE. Despite their importance, both synthesis and *in vitro* experimentation remain challenging to perform systematically and at scale [8]. This motivates future research into how to better scale and incorporate *in vitro* validation into MADDIE. In anticipation of such validation, MADDIE provides synthesis routes, predicted binding affinity, and ADMET profiling for molecules

it generates, providing extensive information needed to prioritize which candidates to advance. Furthermore, the results from *in vitro* experimentation can also be provided back to MADDIE, enabling the system to adapt its workflow and revise its strategy to produce molecules more likely to advance in a real-world drug discovery campaign.

Human-in-the-loop design for MADDIE

Domain experts are essential for successful drug discovery—a domain where reliable safety assessments is the highest priority and in-depth understanding is required. In this study, we couple MADDIE’s evaluation with assessments from a domain expert, serving primarily to verify the quality of the generated molecules, rather than to drive the generation process itself. A natural extension for MADDIE is integrating domain experts’ feedback directly into the system, enabling an interactive framework in which expert judgments can steer the generation process toward more desirable regions of chemical space. The current design for MADDIE is well-positioned to integrate this feedback: because ORCHESTRATOR already maintains an evolving repository of candidates and decides which to pursue, expert judgments can be supplied as additional information for ORCHESTRATOR to shape its decision. For instance, domain experts can flag undesirable substructures and scaffolds on grounds that current *in silico* tools do not capture, or indicate preferences among a set of candidates, thus steering the campaign towards more desirable molecules. More importantly, such feedback need not be time-consuming: a domain expert can provide occasional feedback while still meaningfully shaping the drug discovery campaign. Through a human-in-the-loop design, MADDIE can improve its reliability of producing molecules likely to advance in a real-world drug discovery campaign (e.g., *in vitro* validation), while serving as a collaborator for domain experts (rather than a system that can only be inspected).

MADDIE as a fully autonomous and self-driving system with laboratory integration

A more ambitious direction is to integrate automated laboratory platforms directly into MADDIE, enabling a fully autonomous, self-driving drug discovery system. Such a system allows MADDIE to synthesize molecules on automated synthesis platforms and submit the synthesized molecules for *in vitro* validation, closing the loop between *in silico* design and *in vitro* validation. SYNTHESIZER and EVALUATOR are the natural integration points, by equipping both agents with automated synthesis and *in vitro* validation platforms, respectively. ORCHESTRATOR can also serve as another point of integration. Since synthesis is costly and time-consuming, this makes candidate selection a resource allocation problem (e.g., selecting only the most promising molecules to synthesize under a fixed budget), a decision that requires a component with a global-level view of a campaign, which ORCHESTRATOR provides. Only molecules ORCHESTRATOR deems satisfactory should proceed to synthesis, with synthetic feasibility from SYNTHESIZER and predicted properties from EVALUATOR grounding that decision.

Several safeguards must be implemented before a fully autonomous self-driving system can be deployed responsibly. A system that acts in the physical world introduces safety concerns absent from a system that works purely *in silico*. For instance,

a molecule deemed unsatisfactory *in silico* can simply be discarded, whereas a synthesized molecule must be handled, stored, and disposed of responsibly. Safeguards are also needed to ensure molecules generated by MADDIE are not hazardous, whether due to error or misuse of the system. Such safeguards may include structural screening against molecules known to be toxic and hazardous and including medicinal chemists in the critical step of laboratory testing. Even so, the potential benefits are substantial. Grounding MADDIE’s decisions in laboratory feedback can let the system steer its workflow toward real-world outcomes, and the resulting laboratory measurements can accumulate into data well suited to improving MADDIE. More broadly, such a system can shift the role of agents in drug discovery from informing medicinal chemists to driving the drug discovery process itself, with the potential to substantially reduce the time and cost of bringing new medicine to patients.

6 Conclusions

We present MADDIE, an autonomous multi-agent system for *in silico* drug discovery capable of generating molecules with satisfying properties for drug development. Rather than executing a fixed workflow, MADDIE decides which worker agent to invoke through ORCHESTRATOR, drawing on its memory module and evolving repositories of candidates for its decision-making. Across 50 targets, MADDIE outperforms state-of-the-art baselines spanning generative models for drug design, agents for drug discovery, and generalist LLMs, successfully generating 100 satisfying molecules per target that meet key molecular property requirements while also providing their synthesis routes. Our ablation reveals that this performance is attributed to MADDIE’s collaboration between agents, where each worker agent contributes distinctly, while ORCHESTRATOR contributes the most by concentrating effort on the most promising candidates that greatly reduce the chemical space MADDIE must explore to find the satisfying molecules. Consistent with its design, MADDIE exhibits strong adaptability across campaigns, effectively employing different strategies as each target demands. Notably, MADDIE demonstrates intelligent strategy adaptations for a challenging target with no existing marketed FDA-approved drugs. On PSEN1, a critical target associated with Alzheimer’s disease, MADDIE generates PSEN1-01, a promising molecule with *in silico* properties outperforming those of the reference ligand and Nirogacestat, while also demonstrating favorable binding interaction with the challenging site of the target. MADDIE achieves strong performance at low cost and time, with one campaign roughly producing 21K output tokens and taking 3.3 hours on average. We envision MADDIE as a foundation for future autonomous drug discovery systems that can carry a molecule *in silico* to potential drug candidates, ultimately accelerating the timeline and cost of bringing new drugs to patients.

7 Methods

We introduce MADDIE, a multi-agent system for *in silico* drug discovery. It is composed of a central agent, ORCHESTRATOR, and four specialized worker agents. Given a target protein and a set of desired molecular properties, MADDIE aims to identify

molecules that bind to the target and satisfy all desired properties. MADDIE performs the task by distributing subtasks across worker agents, with ORCHESTRATOR deciding which subtask to pursue next. This design enables MADDIE to decompose a complex, multi-objective problem typical in drug discovery into manageable subtasks, enabling each agent to collectively advance the drug discovery campaign.

ORCHESTRATOR (Section 7.1) is responsible for the entire workflow of MADDIE, deciding how a drug discovery campaign proceeds by delegating to a worker agent and processing the molecules. Under its coordination, the worker agents perform distinct subtasks critical to a successful drug discovery campaign [61]: **(1)** HITIDENTIFIER (Section 7.2.1) serves as the entry point of a campaign, providing candidate molecules from which a campaign proceeds; **(2)** LEADOPTIMIZER (Section 7.2.2) serves as the refinement stage, advancing existing candidate molecules towards those with satisfying properties; **(3)** SYNTHESIZER (Section 7.2.3) bridges the gap between molecules designed *in silico* and those that can be made in practice, proposing synthesis routes and finding synthesizable analogs; and **(4)** EVALUATOR (Section 7.2.4) functions as a quality control layer, assessing the key properties that define a successful drug, including binding affinity, drug-likeness, and safety-related properties. Acting in concert, these components allow MADDIE to emulate the iterative, collaborative nature of a real-world campaign within a single autonomous system [61].

Each agent (ORCHESTRATOR and the four worker agents) maintains its own *workspace*, each is used to store relevant artifacts required for their tasks (Table 6). The workspaces are dynamically changing and growing continuously, with more artifacts (e.g., molecules) generated as the campaign progresses. Each worker agent operates within its own workspace, with no access to the others. This prevents any alteration of artifacts from worker agents outside their own workspace, preventing error propagation throughout MADDIE. Meanwhile, ORCHESTRATOR alone has access to each worker agent’s workspace, granting it a global view required to lead the campaign.

7.1 ORCHESTRATOR

At the center of MADDIE is ORCHESTRATOR, an agent responsible for the decision-making of the entire drug discovery campaign. ORCHESTRATOR decides how a campaign proceeds: which worker agent to call next and which candidate molecules to focus on. Each decision is conditioned on two sources of information: **(1)** the existing candidate molecules generated throughout the campaign, which captures the progress of the campaign, and **(2)** the profiles of the worker agents (Appendix E.2), which describe the capabilities that each worker agent can contribute to the campaign. Together, these information allow ORCHESTRATOR to reason jointly about what the campaign needs and which worker agent is best suited to provide it. Furthermore, this information enables dynamic decision-making for ORCHESTRATOR, choosing which drug discovery step to focus on next and calling the worker agent dedicated to that step. This design makes MADDIE’s campaign non-linear and adaptive, mirroring the iterative, feedback-driven nature of real-world drug discovery [61].

Operationally, ORCHESTRATOR makes two kinds of decisions. It can choose to advance the campaign by issuing a *delegate* action to a worker agent, or to collect information (e.g., the properties of existing candidate molecules) by issuing a *process*

action. **(1)** Using a *delegate* action, ORCHESTRATOR sends a worker agent a prompt consisting of a task description and references to the artifacts stored in each worker agent’s workspace (Section 7.2). In return, it receives a structured report of what the worker agent has done. These reports, along with the information gathered from *process* action(s) and the profiles of worker agents will determine which worker agent should be called next via *delegate*. **(1)** Using a *process* action, ORCHESTRATOR reads the artifacts produced from the worker agent it has just called. What it takes from them depends on the worker agent: the molecules generated by HITIDENTIFIER or modified by LEADOPTIMIZER, the property values computed by EVALUATOR, or the synthesis routes found by SYNTHESIZER. From these artifacts it updates its view of which candidate molecules meet the requirements and which of them still fall short.

With these two actions, ORCHESTRATOR drives the campaign forward (through *delegate*) while continually grounding its decision (through *process*). Carrying these actions out over many steps rests on two elements of its design: **(1)** the memory that keeps its own record tractable and **(2)** the satisfying candidate repositories in which it keeps the molecules worth pursuing, stored in the workspace.

Memory module

A defining feature of ORCHESTRATOR is its capability to sustain a long-horizon campaign for MADDIE (Section 4.3.3). This is enabled by a dedicated memory module that dynamically stores the most relevant information. The module holds the records of the campaign itself—the information of the artifacts produced, the reports the worker agents return and ORCHESTRATOR’s own prior decisions—kept in full. Other information is stored and pruned to the most recent entries, such as system-level messages generated throughout the campaign (e.g., ORCHESTRATOR calling a worker agent with incorrect format). When the memory module exceeds what the underlying LLM can hold in its context, the module compresses its records by summarizing them, selectively storing and pruning information based on its recency and relevance. The module thus provides ORCHESTRATOR a compact, decision-relevant context, allowing it to condition each decision on the salient history of the campaign while remaining within the effective reasoning capacity of the underlying LLM.

Satisfying candidate repositories

Over the course of a campaign, ORCHESTRATOR maintains two repositories of molecules. The first repository $\mathcal{P}$ holds the candidate molecules that satisfy every property evaluated by EVALUATOR. Concretely, these are molecules from HITIDENTIFIER and LEADOPTIMIZER that pass each property requirement. The second repository $\mathcal{S}$ holds candidate molecules that both satisfy every property and are also synthesizable as verified by SYNTHESIZER. This set has two sources: **(1)** molecules from $\mathcal{P}$ that are synthesizable, and **(2)** their synthesizable analogs (Section 7.2.3) found by SYNTHESIZER that satisfy every property as evaluated by EVALUATOR. ORCHESTRATOR takes both repositories into account when choosing each subsequent action. These repositories serve several purposes: $\mathcal{P}$ tracks which candidate molecules are worth considering for synthesis assessments, while $\mathcal{S}$ tracks which of them are both satisfying and also synthesizable. They also inform ORCHESTRATOR on which worker

Table 6: Artifacts required and produced by each worker agent, along with their respective tools.

Worker agent	Input	Output	Tools
HITIDENTIFIER	Target pocket P .	Candidate molecules.	conDitar [10] for SBDD; DiffSMol [8] for LBDD; AutoDock [62] for predicting binding affinity.
EVALUATOR	Candidate molecules.	Property values.	UniDock [6], QED [38], SAScore [39], ADMET-AI [7] and Lilly filter [43].
LEADOPTIMIZER	Candidate molecules that fail one or more requirements.	Optimized candidate molecules.	GeLLMO-C [9]; MolOpt algorithms [63]; RDKit and EVALUATOR-backed oracles for the target properties.
SYNTHESIZER	Candidate molecules.	Synthesis routes or synthesizable analogs.	MEEA* [64] and PrexSyn [65].

agent it should call next. For instance, a large $\mathcal{P}$ means there are many candidate molecules with satisfying properties, which informs ORCHESTRATOR that it is better to call SYNTHESIZER to verify their synthesizability next.

ORCHESTRATOR is allowed a budget of actions per campaign, and stops once $\mathcal{S}$ holds the requested number of molecules or this budget is exhausted. At each step ORCHESTRATOR is informed of how many actions remain, and once a few of them are left it stops searching for new molecules and delegates those in $\mathcal{P}$ to SYNTHESIZER, ensuring that the candidates accumulated so far can still enter $\mathcal{S}$ before the budget runs out.

The combination of these capabilities in ORCHESTRATOR is what distinguishes MADDIE from prior approaches. Together they allow MADDIE to operate autonomously, to accelerate drug discovery process (Section 4.5.2), and to achieve strong performance in our benchmark (Table 2). Critically, they allow MADDIE to complete long-horizon campaigns on varying targets (Section 4.3.3).

7.2 Worker Agents

Each worker agent carries out one specialized subtask in a drug discovery campaign, each critical for a successful campaign. Each worker agent maintains a static profile, containing a brief summary of its capabilities and what subtask they support (Appendix E.2). These profiles are used by ORCHESTRATOR during its decision-making.

Each worker agent receives from ORCHESTRATOR a prompt containing a task description with references to the artifacts in its workspace, executes the subtask with its own tools and stores the artifacts it produces in its workspace. The worker agent then issues a *report* by providing a structured summary of what it has done back to ORCHESTRATOR. Table 6 lists the artifacts each worker agent requires and produces in its workspace and the tools used. The subsections below describe the strategy with which each worker agent approaches its subtask.

7.2.1 HITIDENTIFIER

HITIDENTIFIER serves as the entry point of MADDIE’s campaign, supplying the candidate molecules from which the rest of the campaign proceeds. It is the only worker agent that conditions the molecule generation directly on the target structure, enabling MADDIE to propose molecules likely to bind the target. Similar to hit identification step in drug discovery, HITIDENTIFIER prioritizes binding affinity, a key property for molecules to have the intended therapeutic effects [66]. Every molecule MADDIE proposes is derived (e.g., optimized) from what HITIDENTIFIER finds, emphasizing its importance.

HITIDENTIFIER first receives a subtask from ORCHESTRATOR containing the references to artifacts in its workspace (e.g., target structure), a number of molecules to return and a budget of iterations to find them. HITIDENTIFIER then finds these molecules over a series of iterations using two strategies: structure-based drug design (SBDD) [67] and ligand-based drug design (LBDD) [68]. At each iteration, it selects a strategy, invokes the corresponding tool, and accumulates the resulting molecules alongside those from previous iterations. The molecules are then scored using their predicted binding affinities, stored in its workspace, and accumulated to guide the selection of the next strategy. HITIDENTIFIER stops once the molecules satisfying the constraints reaches or exceeds the requested number from ORCHESTRATOR or it exhausts the budget of iterations.

Structure-based drug design

SBDD generates molecules conditioned on the three-dimensional structure of the target pocket. It enables HITIDENTIFIER to design molecules by exploiting the binding pocket’s spatial configurations, offering a mechanistic approach to molecular discovery grounded in pocket-ligand interactions. SBDD further provides a principled starting point for ligand-based modeling, or LBDD, and thus complements each other.

Ligand-based drug design

LBDD instead generates molecules conditioned on the shape of one that already binds to the pocket. Where SBDD generates molecules from the binding pocket, LBDD expands around the known binding molecules. Since it relies on different prior information (e.g., shape-conditioned instead of pocket-conditioned), LBDD broadens HITIDENTIFIER’s exploration when combined with SBDD and thus, further increases the likelihood to find more and stronger binding molecules.

7.2.2 LEADOPTIMIZER

LEADOPTIMIZER is the molecular-refinement component of MADDIE. It complements the molecular generative stages as done by HITIDENTIFIER (e.g., generates candidates from a binding pocket) and SYNTHESIZER (e.g., generates synthesizable analogs from existing candidates). Specifically, their candidate molecules are used as starting points by LEADOPTIMIZER to produce derivatives with improved properties. LEADOPTIMIZER addresses the different property requirements a campaign requires through two different families of optimizer: **(1)** pretrained controllable multi-property

optimizer [9] and (2) oracle-guided single-property optimizer [63], each is distinct and complements each other. They provide both adaptability and control over the property improvements to whatever a campaign demands, enabling LEADOPTIMIZER to transform molecules in MADDIE to those with satisfying properties.

LEADOPTIMIZER performs a subtask by first formulating a plan and then selecting the most suitable optimizer. Concretely, after ORCHESTRATOR identifies which candidate molecules are needed to improve, what properties require improvement, and the number of derivatives to produce for each molecule, LEADOPTIMIZER turns the request into a structured plan with objectives. It contains both properties to be improved and the required values or ranges. The plan then guides LEADOPTIMIZER to select the best optimizer from one of two families, based on which properties to optimize, how many of them, and the performance of that optimizer, when such records are stored by LEADOPTIMIZER. Once an optimizer is selected, LEADOPTIMIZER applies it to each candidate molecule, producing molecule derivatives that are then validated, de-duplicated, stored in its workspace, and reported back to ORCHESTRATOR. If the initial attempt doesn’t achieve the requested property improvement(s) or produce enough valid derivatives, LEADOPTIMIZER may run follow-up attempts including retrying or switching optimizers, subject to an execution budget.

Pretrained controllable multi-property optimizer

The pretrained controllable multi-property optimizer family is well suited when the subtask requires improving multiple properties while attempting to preserve other desirable molecular properties. This family offers LEADOPTIMIZER greater control over which properties to maintain and improve. In contrast to using multiple single-property optimizers (the second family), a pretrained multi-property optimizer provides more balanced improvement (e.g., improving while maintaining) and efficiency (e.g., one instead of multiple runs). However, because this family is often pretrained on specific properties [9], it limits their adaptability for the wide range of properties a drug discovery campaign may require, necessitating LEADOPTIMIZER to use another family of optimizers.

Oracle-guided single-property optimizer

Oracle-guided single-property optimizers refine molecules using an *in silico* property predictor available to LEADOPTIMIZER (Table 6). This family is primarily used when the properties to optimize are not supported by the first family. LEADOPTIMIZER has access to more than one optimizer in this family. To select the best one, it uses its available property predictors, prior optimization results, and historical performance records of each tool as guidance (Appendix E.3).

7.2.3 SYNTHESIZER

SYNTHESIZER investigates the synthesis of molecules, assessing whether the candidate molecules in MADDIE can be made in practice. This is critical for *in silico* drug discovery, as synthesizability is a major bottleneck to development of AI-generated molecules [69]. SYNTHESIZER assesses synthesizability based on whether it can generate a synthesis route for a given molecule, grounding its assessments in interpretable

reaction chemistry rather than relying solely on estimated synthetic accessibility scores [39]. SYNTHESIZER uses two complementary strategies to generate synthesis routes: (1) retrosynthesis planning [64], and (2) synthesizable analog generation [65]. This two-strategy approach mirrors conventional practice in drug discovery, where molecules that are difficult to synthesize directly are used to identify analogs from established libraries of synthesizable molecules.

Retrosynthesis planning

SYNTHESIZER uses retrosynthesis planning to generate synthesis routes by starting from a given candidate molecule and working through the reverse logic of synthesis reactions to reach purchasable starting materials. Using this approach, SYNTHESIZER can identify synthesis routes to a specific molecule, mimicking the retrosynthetic analyses performed by chemists in conventional drug discovery. If this planning process can identify a route for a given molecule, SYNTHESIZER uses the route as the evidence to conclude that the molecule is synthesizable and terminates. However, if no route is found, SYNTHESIZER investigates further using the synthesizable analog generation strategy.

Synthesizable analog generation

In synthesizable analog generation, SYNTHESIZER identifies analogs by constructing synthesis routes to molecules that are structurally similar to the given candidate molecule. Unlike retrosynthesis planning, the route generation starts from purchasable starting materials and works through the forward logic of synthesis reactions to produce analogs. This forward-synthesis approach naturally complements retrosynthesis planning, and can uncover synthesis routes that retrosynthesis planning may struggle to identify. In some cases, the product of the generated synthesis route is not an analog, but the given candidate molecule itself. When this occurs, SYNTHESIZER concludes that the candidate molecule is synthesizable. Otherwise, SYNTHESIZER concludes that the candidate molecule is unsynthesizable and provides the analogs and their routes to ORCHESTRATOR as potential alternatives to the candidate molecule. Thus, SYNTHESIZER goes beyond synthesizability assessment, converting unsynthesizable molecules into potentially promising new leads to further the drug discovery campaign.

7.2.4 EVALUATOR

EVALUATOR serves as the quality control mechanism in MADDIE. Every decision made by ORCHESTRATOR regarding the candidate molecules is grounded in EVALUATOR’s assessments, thus shaping the trajectory of MADDIE’s campaign. EVALUATOR performs a comprehensive assessment of each candidate molecule and supplies the property values on which the rest of a campaign depends. The properties EVALUATOR supports include binding affinity, drug-likeness, synthesizability, ADMET and structural liabilities. Each is critical to a successful drug and computed with its own respective tool (Table 6). Its synthesizability assessment differs from SYNTHESIZER since EVALUATOR uses a rapid heuristic scorer [39] rather than

finding an entire synthesis route from purchasable starting molecules to the candidate molecule. The score complements the synthesis route—it can convey information on the ease of synthesis [39] as a quantitative number that can help in prioritizing molecules.

Acknowledgements

This project was made possible, in part, by support from the National Science Foundation grant no. IIS-2435819 (X.N., D.A.-A.), the National Library of Medicine grant no. R01LM014385 (X.N.), and the National Institute of Aging grant no. R01AG082044 (X.N.). Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agency.

Conflict of interests

The authors claim no conflict of interests.

Ethics Statement

MADDIE is an autonomous system for drug discovery, enabling users to generate molecules that meet the specified property requirements prompted in natural language. We acknowledge that not all generated molecules are safe, and MADDIE can potentially be used to generate harmful molecules. We take a step to minimize the potential negative impact and design MADDIE to operate purely *in silico*, with no capability for the system to execute synthesis and produce physical molecules. Nonetheless, we highly encourage supervision by responsible domain experts when using MADDIE. All users of MADDIE are advised to exercise discretion and to comply with applicable safety guidelines, regulations, and established professional best practices.

Code Availability

All the source code and implementation for MADDIE is available in <https://github.com/ninglab/MADDIE>.

Data Availability

The preprocessed 3D structures of the protein targets, their annotations, and the associated FDA-approved drugs are available in <https://github.com/ninglab/MADDIE>.

Author Contributions

X.N. conceived the research and conducted the project administration. X.N. investigated the research, obtained funding and resources for the research, and supervised

the student and postdoctoral authors. Y.L., R.A. and X.N. designed the computational methodologies of MADDIE. Y.L. and R.A. designed and implemented the computational methodologies of ORCHESTRATOR. R.G. designed and implemented the computational methodologies of HITIDENTIFIER. A.M. designed and implemented the computational methodologies of LEADOPTIMIZER. F.B. designed and implemented the computational methodologies of SYNTHESIZER. Y.L. designed and implemented the computational methodologies of EVALUATOR. Y.L. conducted the main experiments. R.A., A.M., F.B. and R.G. conducted the baseline experiments. R.A. and A.M. conducted data curation. Y.L. and R.A. conducted formal analysis and result analysis. D.A.-A. contributed medicinal chemistry expertise to the result analysis. Y.L. conducted visualization; R.A. and A.M. contributed to visualization. R.A. drafted the original manuscript. Y.L., A.M., F.B., R.G. and X.N. conducted the manuscript reviewing and editing. All the authors reviewed the final paper.

References

- [1] Wouters, O.J., McKee, M., Luyten, J.: Estimated research and development investment needed to bring a new medicine to market, 2009-2018. *JAMA* **323**(9), 844–853 (2020) <https://doi.org/10.1001/jama.2020.1166>
- [2] Sertkaya, A., Beleche, T., Jessup, A., Sommers, B.D.: Costs of drug development and research and development intensity in the us, 2000-2018. *JAMA network open* **7**(6), 2415445 (2024)
- [3] Mulcahy, A., Rennane, S., Schwam, D., Dickerson, R., Baker, L., Shetty, K.: Use of clinical trial characteristics to estimate costs of new drug development. *JAMA Network Open* **8**(1), 2453275–2453275 (2025) <https://doi.org/10.1001/jamanetworkopen.2024.53275>
- [4] Singh, N., Vayer, P., Tanwar, S., Poyet, J.-L., Tsaioun, K., Villoutreix, B.O.: Drug discovery and development: introduction to the general public and patient groups. *Frontiers in Drug Discovery* **Volume 3 - 2023** (2023) <https://doi.org/10.3389/fddsv.2023.1201419>
- [5] Trott, O., Olson, A.J.: 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) <https://doi.org/10.1002/jcc.21334>
- [6] Yu, Y., Cai, C., Wang, J., Bo, Z., Zhu, Z., Zheng, H.: Uni-dock: Gpu-accelerated docking enables ultralarge virtual screening. *Journal of Chemical Theory and Computation* **19**(11), 3336–3345 (2023) <https://doi.org/10.1021/acs.jctc.2c01145>
- [7] Swanson, K., Walther, P., Leitz, J., Mukherjee, S., Wu, J.C., Shivnaraine, R.V., Zou, J.: Admet-ai: a machine learning admet platform for evaluation of large-scale

- chemical libraries. *Bioinformatics* **40**(7), 416 (2024)
- [8] Chen, Z., Peng, B., Zhai, T., Adu-Ampratwum, D., Ning, X.: Generating 3d small binding molecules using shape-conditioned diffusion models with guidance. *Nature machine intelligence* **7**(5), 758–770 (2025)
 - [9] Dey, V., Hu, X., Ning, X.: Large language models for controllable multi-property multi-objective molecule optimization. In: Christodoulopoulos, C., Chakraborty, T., Rose, C., Peng, V. (eds.) *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 20996–21023. Association for Computational Linguistics, Suzhou, China (2025). <https://doi.org/10.18653/v1/2025.findings-emnlp.1145>
 - [10] Gao, R., Peng, J., Chen, Z., Baker, F.N., Kombo, D.C., Jr., J.L.K., Scholte, A.A., Li, Y., LaMarche, M.J., Iconaru, L.I., Biemann, H.-P., Hong, M., Ning, X.: Generating Developable 3D Molecules via Pocket-Conditioned Diffusion and Property-Aware Optimization (2026). <https://arxiv.org/abs/2607.12349>
 - [11] Averly, R., Baker, F.N., Watson, I.A., Ning, X.: LIDDIA: Language-based intelligent drug discovery agent. In: Christodoulopoulos, C., Chakraborty, T., Rose, C., Peng, V. (eds.) *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12004–12028. Association for Computational Linguistics, Suzhou, China (2025). <https://doi.org/10.18653/v1/2025.emnlp-main.603>
 - [12] Liu, S., Lu, Y., Chen, S., Hu, X., Zhao, J., Lu, Y., Zhao, Y.: DrugAgent: Automating AI-aided Drug Discovery Programming through LLM Multi-Agent Collaboration (2025). <https://arxiv.org/abs/2411.15692v2>
 - [13] Pan, Q., Xu, D., Yang, Q., Yao, J.X., Yuan, S., Zhu, Z., Li, J., Ji, J.: Frogent: An end-to-end full-process drug design multi-agent system. *arXiv preprint arXiv:2508.10760* (2025)
 - [14] Gao, B., Huang, Y., Liu, Y., Xie, W., Ma, W.-Y., Zhang, Y.-Q., Lan, Y.: Pharmagents: Building a virtual pharma with large language model agents. *arXiv preprint arXiv:2503.22164* (2025)
 - [15] Wang, E., Schmidgall, S., Jaeger, P.F., Zhang, F., Pilgrim, R., Matias, Y., Barral, J., Fleet, D., Azizi, S.: Txgemma: Efficient and agentic llms for therapeutics. *arXiv preprint arXiv:2504.06196* (2025)
 - [16] Cao, H., Liu, S., Zhang, F., Liu, Z., Li, H., Feng, B., Bai, S., Chen, L., Xie, K., Li, Y.: Mozi: Governed autonomy for drug discovery llm agents. *arXiv preprint arXiv:2603.03655* (2026)
 - [17] Kantorow, J., Mani, N., Mohanraj, N.R., Zong, X.: F.a.d.e. (fully agentic drug engine): A conversational ai platform for drug discovery. *bioRxiv* (2026) <https://doi.org/10.1101/2026.01.01.500000>

- [18] Vichentijevikj, I., Mishev, K., Simjanoska Misheva, M.: Prompt-to-pill: Multi-agent drug discovery and clinical simulation pipeline. *Bioinformatics Advances* **6**(1), 323 (2026) <https://doi.org/10.1093/bioadv/vbaf323>
- [19] Ock, J., Meda, R.S., Badrinarayanan, S., Aluru, N.S., Chandrasekhar, A., Barati Farimani, A.: Large language model agent for modular task execution in drug discovery. *Journal of Chemical Information and Modeling* **66**(4), 2055–2068 (2026) <https://doi.org/10.1021/acs.jcim.5c02454>
- [20] Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A., et al.: Openai gpt-5 system card. arXiv preprint arXiv:2601.03267 (2025)
- [21] D. Segall, M.: Multi-parameter optimization: Identifying high quality compounds with a balance of properties. *Current Pharmaceutical Design* **18**(9), 1292–1310 (2012) <https://doi.org/10.2174/138161212799436430>
- [22] Bohacek, R.S., McMartin, C., Guida, W.C.: The art and practice of structure-based drug design: a molecular modeling perspective. *Medicinal research reviews* **16**(1), 3–50 (1996)
- [23] Zuo, X., Chen, Z.: From gene to therapy: a review of deciphering the role of abcd1 in combating x-linked adrenoleukodystrophy. *Lipids in health and disease* **23**(1), 369 (2024)
- [24] McNaughton, A.D., Sankar Ramalaxmi, G.K., Kruel, A., Knutson, C.R., Varikoti, R.A., Kumar, N.: Cactus: Chemistry agent connecting tool usage to science. *ACS Omega* **9**(46), 46563–46573 (2024) <https://doi.org/10.1021/acsomega.4c08408>
- [25] Ghareeb, A.E., Chang, B., Mitchener, L., Yiu, A., Szostkiewicz, C.J., Shved, D., Gyimesi, G.J., Laurent, J.M., Wright, S.M., Razzak, M.T., White, A.D., Finne-mann, S.C., Hinks, M.M., Rodriques, S.G.: A multi-agent system for automating scientific discovery. *Nature* **655**(8122), 497–505 (2026) <https://doi.org/10.1038/s41586-026-10652-y>
- [26] Peng, X., Luo, S., Guan, J., Xie, Q., Peng, J., Ma, J.: Pocket2Mol: Efficient molecular sampling based on 3D protein pockets. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (eds.) *Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research*, vol. 162, pp. 17644–17655. PMLR, Baltimore, Maryland, USA (2022)
- [27] Guan, J., Qian, W.W., Peng, X., Su, Y., Peng, J., Ma, J.: 3d equivariant diffusion for target-aware molecule generation and affinity prediction. In: *International Conference on Learning Representations* (2023)

- [28] Gu, S., Xu, M., Powers, A., Nie, W., Geffner, T., Kreis, K., Leskovec, J., Vahdat, A., Ermon, S.: Aligning target-aware molecule diffusion models with exact energy optimization. In: Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., Zhang, C. (eds.) *Advances in Neural Information Processing Systems*, vol. 37, pp. 44040–44063 (2024). <https://doi.org/10.52202/079017-1398>
- [29] Gao, B., Huang, Y., Liu, Y., Xie, W., He, B., Tan, H., Ma, W.-Y., Zhang, Y.-Q., Lan, Y.: Cidd: Collaborative intelligence for structure-based drug design empowered by llms. In: Belgrave, D., Zhang, C., Lin, H., Pascanu, R., Koniusz, P., Ghassemi, M., Chen, N. (eds.) *Advances in Neural Information Processing Systems*, vol. 38, Main Conference, pp. 13651–13699 (2025). <https://doi.org/10.52202/085713-0460>
- [30] Hu, X., Tuo, H., Ji, J., Li, A., Xing, Q., Yang, B.: Steering large language models through the DMTA cycle: Structure-based drug design via knowledge-driven bi-level thompson sampling. In: Zhang, T., Dudik, M., Jaggi, M., Agarwal, A., Li, S., Schuurmans, D., Zhu, J., Berkenkamp, F., Dong, H., Bietti, A. (eds.) *Proceedings of the 43rd International Conference on Machine Learning. Proceedings of Machine Learning Research*, vol. 306, pp. 45703–45730 (2026)
- [31] Hu, X., Li, A., Xing, Q., Ji, J., Tuo, H., Yang, B.: Empowering llms for structure-based drug design via exploration-augmented latent inference. In: *Proceedings of the ACM Web Conference 2026. WWW '26*, pp. 4244–4255. Association for Computing Machinery, New York, NY, USA (2026). <https://doi.org/10.1145/3774904.3792605>
- [32] Adams, K., Coley, C.W.: Equivariant shape-conditioned generation of 3d molecules for ligand-based drug design. In: *The Eleventh International Conference on Learning Representations* (2023)
- [33] Long, S., Zhou, Y., Dai, X., Zhou, H.: Zero-shot 3d drug design by sketching and generating. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) *Advances in Neural Information Processing Systems*, vol. 35, pp. 23894–23907 (2022). <https://doi.org/10.52202/068431-1735>
- [34] Zhavoronkov, A., Ivanenkov, Y.A., Aliper, A., Veselov, M.S., Aladinskiy, V.A., Aladinskaya, A.V., Terentiev, V.A., Polykovskiy, D.A., Kuznetsov, M.D., Asadulaev, A., *et al.*: Deep learning enables rapid identification of potent ddr1 kinase inhibitors. *Nature biotechnology* **37**(9), 1038–1040 (2019)
- [35] Francoeur, P.G., Masuda, T., Sunseri, J., Jia, A., Iovanisci, R.B., Snyder, I., Koes, D.R.: Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of Chemical Information and Modeling* **60**(9), 4200–4215 (2020) <https://doi.org/10.1021/acs.jcim.0c00411>
- [36] Buniello, A., Suveges, D., Cruz-Castillo, C., Llinares, M., Cornu, H., Lopez, I., Tsukanov, K., Roldán-Romero, J., Mehta, C., Fumis, L., McNeill, G., Hayhurst,

- J., Martinez Osorio, R., Barkhordari, E., Ferrer, J., Carmona, M., Uniyal, P., Falaguera, M., Rusina, P., Smit, I., Schwartzentruber, J., Alegbe, T., Ho, V., Considine, D., Ge, X., Szyszkowski, S., Tsepilov, Y., Ghoussaini, M., Dunham, I., Hulcoop, D., McDonagh, E., Ochoa, D.: Open targets platform: facilitating therapeutic hypotheses building in drug discovery. *Nucleic Acids Research* **53**(D1), 1467–1475 (2025) <https://doi.org/10.1093/nar/gkae1128>
- [37] Berman, H.M., J. Westbrook, Z.F., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E.: The protein data bank. *Nucleic Acids Research* **28**(1), 235–242 (2000) <https://doi.org/10.1093/nar/28.1.235>
- [38] Bickerton, G.R., Paolini, G.V., Besnard, J., Muresan, S., Hopkins, A.L.: Quantifying the chemical beauty of drugs. *Nature chemistry* **4**(2), 90–98 (2012)
- [39] Ertl, P., Schuffenhauer, A.: Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics* **1**(1), 8 (2009)
- [40] Xu, C., Cheng, F., Chen, L., Du, Z., Li, W., Liu, G., Lee, P.W., Tang, Y.: In silico prediction of chemical ames mutagenicity. *Journal of Chemical Information and Modeling* **52**(11), 2840–2847 (2012) <https://doi.org/10.1021/ci300400a>
- [41] Gayvert, K.M., Madhukar, N.S., Elemento, O.: A data-driven approach to predicting successes and failures of clinical trials. *Cell chemical biology* **23**(10), 1294–1301 (2016)
- [42] Wang, S., Sun, H., Liu, H., Li, D., Li, Y., Hou, T.: Admet evaluation in drug discovery. 16. predicting herg blockers by combining multiple pharmacophores and machine learning approaches. *Molecular Pharmaceutics* **13**(8), 2855–2866 (2016) <https://doi.org/10.1021/acs.molpharmaceut.6b00471>
- [43] Bruns, R.F., Watson, I.A.: Rules for identifying potentially reactive or promiscuous compounds. *Journal of Medicinal Chemistry* **55**(22), 9763–9772 (2012) <https://doi.org/10.1021/jm301008n>
- [44] Blakeman, A., Thomas, A., Jhunjhunwala, A., Gupta, A., Khattar, A., Rajfer, A., Renduchintala, A., Asif, A., Vavre, A., Miranda, A.F., et al.: Nemotron 3 ultra: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. arXiv preprint arXiv:2606.15007 (2026)
- [45] Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al.: Qwen3 technical report. arXiv preprint arXiv:2505.09388 (2025)
- [46] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C.H., Gonzalez, J.E., Zhang, H., Stoica, I.: Efficient memory management for large language model serving with pagedattention. In: *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles* (2023)

- [47] Lin, X., Ning, Y., Zhang, J., Dong, Y., Liu, Y., Wu, Y., Qi, X., Sun, N., Shang, Y., Wang, K., et al.: Llm-based agents suffer from hallucinations: A survey of taxonomy, methods, and directions. arXiv preprint arXiv:2509.18970 (2025)
- [48] Darvesh, S.: Butyrylcholinesterase as a diagnostic and therapeutic target for alzheimer’s disease. *Current Alzheimer Research* **13**(10), 1173–7 (2016) <https://doi.org/10.2174/1567205013666160404120542>
- [49] Pajk, S., Knez, D., Košak, U., Zorović, M., Brazzolotto, X., Coquelle, N., Nachon, F., Colletier, J.-P., Živin, M., Stojan, J., Gobec, S.: Development of potent reversible selective inhibitors of butyrylcholinesterase as fluorescent probes. *Journal of Enzyme Inhibition and Medicinal Chemistry* **35**(1), 498–505 (2020) <https://doi.org/10.1080/14756366.2019.1710502> . PMID: 31914836
- [50] Smith, T.M., Tharakan, A., Martin, R.K.: Targeting adam10 in cancer and autoimmunity. *Frontiers in Immunology* **Volume 11 - 2020** (2020) <https://doi.org/10.3389/fimmu.2020.00499>
- [51] Bagaria, J., Bagyinszky, E., An, S.S.A.: Genetics, functions, and clinical impact of presenilin-1 (psen1) gene. *International Journal of Molecular Sciences* **23**(18) (2022) <https://doi.org/10.3390/ijms231810970>
- [52] Hung, L.-W., Sanbonmatsu, K.Y., Williams, R.F., Chen, J.C.-H.: Acetylcholinesterase: Structure, dynamics, and interactions with organophosphorus compounds. *Protein Science* **34**(10), 70297 (2025) <https://doi.org/10.1002/pro.70297> <https://onlinelibrary.wiley.com/doi/pdf/10.1002/pro.70297>
- [53] Jasiecki, J., Wasąg, B.: Butyrylcholinesterase protein ends in the pathogenesis of alzheimer’s disease—could bche genotyping be helpful in alzheimer’s therapy? *Biomolecules* **9**(10) (2019) <https://doi.org/10.3390/biom9100592>
- [54] De Strooper, B., Iwatsubo, T., Wolfe, M.S.: Presenilins and γ -secretase: structure, function, and role in alzheimer disease. *Cold Spring Harbor perspectives in medicine* **2**(1), 006304 (2012)
- [55] Sherrington, R., Rogaev, E., Liang, Y.a., Rogaeva, E., Levesque, G., Ikeda, M., Chi, H., Lin, C., Li, G., Holman, K., et al.: Cloning of a gene bearing missense mutations in early-onset familial alzheimer’s disease. *Nature* **375**(6534), 754–760 (1995)
- [56] Gounder, M., Ratan, R., Alcindor, T., Schöffski, P., Van Der Graaf, W.T., Wilky, B.A., Riedel, R.F., Lim, A., Smith, L.M., Moody, S., et al.: Nirogacestat, a γ -secretase inhibitor for desmoid tumors. *New England Journal of Medicine* **388**(10), 898–912 (2023)
- [57] Sato, C., Takagi, S., Tomita, T., Iwatsubo, T.: The c-terminal pal motif

- and transmembrane domain 9 of presenilin 1 are involved in the formation of the catalytic pore of the γ -secretase. *Journal of Neuroscience* **28**(24), 6264–6271 (2008) <https://doi.org/10.1523/JNEUROSCI.1163-08.2008> <https://www.jneurosci.org/content/28/24/6264.full.pdf>
- [58] Yamasaki, A., Eimer, S., Okochi, M., Smialowska, A., Kaether, C., Baumeister, R., Haass, C., Steiner, H.: The gxgd motif of presenilin contributes to catalytic function and substrate identification of γ -secretase. *Journal of Neuroscience* **26**(14), 3821–3828 (2006) <https://doi.org/10.1523/JNEUROSCI.5354-05.2006> <https://www.jneurosci.org/content/26/14/3821.full.pdf>
- [59] Jiang, L., Lin, J., Zhao, S., Wu, J., Jin, Y., Yu, L., Wu, N., Wu, Z., Wang, Y., Lin, M.: Adamts5 in osteoarthritis: Biological functions, regulatory network, and potential targeting therapies. *Frontiers in Molecular Biosciences* **Volume 8 - 2021** (2021) <https://doi.org/10.3389/fmolb.2021.703110>
- [60] Kim, J.Y., Cho, H., Yoo, J., Kim, G.W., Jeon, Y.H., Lee, S.W., Kwon, S.H.: Pathological role of hdac8: Cancer and beyond. *Cells* **11**(19) (2022) <https://doi.org/10.3390/cells11193161>
- [61] Hughes, J.P., Rees, S., Kalindjian, S.B., Philpott, K.L.: Principles of early drug discovery. *British journal of pharmacology* **162**(6), 1239–1249 (2011)
- [62] Trott, O., Olson, A.J.: 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) <https://doi.org/10.1002/jcc.21334> <https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.21334>
- [63] Gao, W., Fu, T., Sun, J., Coley, C.: Sample efficiency matters: a benchmark for practical molecular optimization. In: *Advances in Neural Information Processing Systems*, vol. 35, pp. 21342–21357 (2022)
- [64] Zhao, D., Tu, S., Xu, L.: Efficient retrosynthetic planning with mcts exploration enhanced a* search. *Communications Chemistry* **7**(1), 52 (2024)
- [65] Luo, S., Coley, C.W.: Efficient and programmable exploration of synthesizable chemical space. *Transactions on Machine Learning Research* (2026)
- [66] Sawada, R., Sakajiri, Y., Shibata, T., Yamanishi, Y.: Predicting therapeutic and side effects from drug binding affinities to human proteome structures. *iScience* **27**(6), 110032 (2024) <https://doi.org/10.1016/j.isci.2024.110032>
- [67] Batool, M., Ahmad, B., Choi, S.: A structure-based drug discovery paradigm. *International Journal of Molecular Sciences* **20**(11) (2019) <https://doi.org/10.3390/ijms20112783>
- [68] Shim, J., MacKerell, A.D. Jr.: Computational ligand-based rational design: role of

- conformational sampling and force fields in model development. *MedChemComm* **2**(5), 356–370 (2011) <https://doi.org/10.1039/c1md00044f>
- [69] Blass, B.E.: *Basic Principles of Drug Discovery and Development*, Second edition edn. Academic Press, London (2021)
- [70] Stelzer, G., Rosen, N., Plaschkes, I., Zimmerman, S., Twik, M., Fishilevich, S., Stein, T.I., Nudel, R., Lieder, I., Mazor, Y., Kaplan, S., Dahary, D., Warshawsky, D., Guan-Golan, Y., Kohn, A., Rappaport, N., Safran, M., Lancet, D.: The genecards suite: From gene data mining to disease genome sequence analyses. *Current Protocols in Bioinformatics* **54**(1), 1–30113033 (2016) <https://doi.org/10.1002/cpbi.5>
- [71] Zdrazil, B., Felix, E., Hunter, F., Manners, E.J., Blackshaw, J., Corbett, S., de Veij, M., Ioannidis, H., Lopez, D.M., Mosquera, J., Magarinos, M., Bosc, N., Arcila, R., Kizilören, T., Gaulton, A., Bento, A., Adasme, M., Monecke, P., Landrum, G., Leach, A.: The chembl database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods. *Nucleic Acids Research* **52**(D1), 1180–1192 (2024) <https://doi.org/10.1093/nar/gkad1004>
- [72] Avram, S., Wilson, T.B., Curpan, R., Halip, L., Borota, A., Bora, A., Bologa, C., Holmes, J., Knockel, J., Yang, J., Oprea, T.: Drugcentral 2023 extends human clinical data and integrates veterinary drugs. *Nucleic Acids Research* **51**(D1), 1276–1287 (2023) <https://doi.org/10.1093/nar/gkac1085>
- [73] Kass-Hout, T.A., Xu, Z., Mohebbi, M., Nelsen, H., Baker, A., Levine, J., Johanson, E., Bright, R.A.: Openfda: an innovative platform providing access to a wealth of fda’s publicly available data. *Journal of the American Medical Informatics Association* **23**(3), 596–600 (2016) <https://doi.org/10.1093/jamia/ocv153>
- [74] Myung, Y., de Sá, A.G.C., Ascher, D.B.: Deep-pk: deep learning for small molecule pharmacokinetic and toxicity prediction. *Nucleic Acids Research* **52**(W1), 469–475 (2024) <https://doi.org/10.1093/nar/gkae254>
- [75] Sterling, T., Irwin, J.J.: Zinc 15 – ligand discovery for everyone. *Journal of Chemical Information and Modeling* **55**(11), 2324–2337 (2015) <https://doi.org/10.1021/acs.jcim.5b00559>
- [76] Morgan, H.L.: The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *Journal of Chemical Documentation* **5**(2), 107–113 (1965) <https://doi.org/10.1021/c160017a018> . Accessed 2023-06-06
- [77] Lilienfeld, S.: Galantamine — a novel cholinergic drug with a unique dual mode of action for the treatment of patients with alzheimer’s disease. *CNS Drug Reviews* **8**(2), 159–176 (2002) <https://doi.org/10.1111/j.1527-3458.2002.tb00221>

x <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1527-3458.2002.tb00221.x>

[78] Birks, J., Chong, L., Grimley Evans, J.: Rivastigmine for alzheimer’s disease. Cochrane Database of Systematic Reviews (9) (2015) <https://doi.org/10.1002/14651858.CD001191.pub4>

A Dataset and Annotations

We provide the protocols when curating our dataset here.

Protein targets. First, we curate a list of targets from LIDDIA [11], Cross-Docked2020 [35] and OpenTargets [36], ensuring that these targets don’t exist in CrossDocked2020 training set. We then annotate the targets with disease associations through cross-referencing with OpenTargets [36] and GeneCards [70]. For the 3D target structures, we first identify their experimentally validated structures with a small molecule ligand from RCSB Protein Data Bank (<https://www.rcsb.org/>) [37] and extract their binding pockets using the ligands as references. The annotations for the protein targets are provided in Table A1. The disease categories column indicates whether the target is associated with autoimmune disease (A), cancer (CA), cardiovascular disease (CV), Diabetes (D), Infectious Disease (I), or Neurological Conditions (N).

Approved FDA drugs. We curate a set of approved drugs associated with the targets, starting from ChEMBL [71] phase 4 target-mechanism records and DrugCentral [72] records. The drugs are then mapped to FDA Drugs@FDA active ingredients using openFDA [73], filtered to currently marketed products (as of July 31, 2026), and collapsed to one active-family representative per target. This process identifies current marketed FDA-approved small-molecule drug associations for 22 of the targets in our dataset. Table A2 lists the associated drugs with targets in our dataset.

B Property Requirements

We use our curated FDA-approved drugs to find the thresholds used in the experiments. We set the thresholds according to two criteria: **(1)** at least 80% of reference drugs pass each individual property requirement and **(2)** at least 40% of reference drugs pass all the requirements simultaneously. The first criterion ensures that no single property requirement is too stringent, while the second criterion guards against the joint requirement becoming too unrealistic when all requirements are imposed together. We calculate the thresholds in two stages. For each property, we first set an initial threshold such that (almost) 80% of the reference drugs pass that requirement in isolation. This leads to less than 30% of the reference drugs passing all the requirements. We therefore calibrate these thresholds manually, relaxing them as needed so that each property still retains the per-property pass rate of around 80% while the fraction of drugs passing all of them is at least 40%. We provide both the final requirements used in our experiment and the pass rates for the FDA-approved drugs in Table A3.

Table A1: Protein targets in the MADDIE dataset.

Gene name	PDB ID	Ligand ID	Disease category	Source	
				LIDDIA	CD2020
ABCC8	6PZ9	BJX	D	—	—
ABCD1	7VX8	ATP	N	—	✓
ABO	5BXC	DA8	CV, I	—	✓
ACE	3L3N	LSW	CV	—	✓
ACHE	7D9O	H0L	A, N	—	—
ADAM10	8ESV	BAT	N	—	—
ADAMTS5	3HYG	099	CA, N	—	✓
ADRB1	7BU7	P0G	CV	✓	—
AKR1B10	4GQ0	QAP	CA, CV	—	✓
AKT1	4GV1	0XZ	CA	—	✓
BACE2	2EWY	DBO	D, N	—	✓
BAZ2A	7QYO	GKI	CA, N	—	✓
BCHE	4TPK	3F9	N	✓	—
CD38	6VUA	ROJ	CA, N	—	✓
CDK5	1UNG	ALH	N	✓	—
CDK6	5L2I	LQQ	CA, N	—	✓
CYP3A5	7SV2	MWY	CA, I	✓	—
DPP7	3N0T	OPY	CA	—	✓
DRD2	6CM4	8NU	N	✓	—
DYRK2	5LXC	7AA	CA	—	✓
FKBP1A	6YF1	OP8	CA	—	✓
FLT3	4RT7	P30	CA	✓	—
GRIN2A	5KCJ	6RM	N	—	—
GRK4	4YHJ	AN2	CV	—	✓
GSTP1	20GS	CBD	CA, N	—	✓
HDAC8	4RN0	L6G	CA, N	—	✓
HDGFL2 (HRP2)	7HHD	A1A3V	CA, N	—	—
HTR2A	7WC8	92S	N	✓	—
IDH2	4JA8	1K9	CA	—	✓
IL1B	8RZB	A1H4A	A, CV	—	—
IPMK	5W2I	ADP	CA, N	—	✓
KIT	4U0I	0LI	CA	✓	—
MAP3K14	4IDV	13V	A	—	✓
MET (HGFR)	2RFN	AM7	CA	✓	—
MME	6SUK	FT8	CV, N	—	✓
NOS2	3EJ8	HEC	A, CV	—	✓
NQO1	10QV	SWF	CA, N	—	✓
NR1H4	5Q0Q	9LD	A, D	—	✓
P2RY12 (P2Y12)	4NTJ	AZJ	CV	✓	—
PAK4	5VEF	M77	CA, N	—	✓
PSEN1	7C9I	FTO	N, CV	✓	—
PTGIS	3B6H	MXD	CV	—	✓
PTGS1 (COX1)	2OYE	IM8	CV	✓	—
QPCT	2AFX	1BN	CA, N	—	✓
RET	6I82	H6W	CA	—	—
SIRT3	4JT9	1NS	N	—	✓
TBK1	4IWQ	1FV	I, N	—	✓
TLR7	6LVY	EWU	A, CA, I	—	—
TNKS (TNKS1)	4TOS	355	CA	—	✓
TNKS2	5AEH	8IR	CA	—	✓

CD2020: CrossDocked2020. ✓: target belongs to the source subset; —: does not.

C Baseline Selection Discussion

Out of several multi-agent systems for drug discovery discussed in Section 2, we found several of them do not provide the code for the system, or only provide some parts of the code. This makes evaluation on these systems incredibly difficult for our experiments. We decide to use baselines that provide their implementation publicly, notably

Table A2: FDA-approved drug associations for targets in the MADDIE dataset.

Target	Associated FDA-approved drugs
ABCC8	Glimepiride, Glipizide, Glyburide, Nateglinide, Repaglinide
ACE	Benazepril, Captopril, Enalapril, Enalaprilat, Fosinopril, Lisinopril, Moexipril, Perindopril, Quinapril, Ramipril, Trandolapril
ACHE	Benzgalantamine, Donepezil, Echothiophate, Galantamine, Neostigmine, Pralidoxime, Pyridostigmine, Rivastigmine
ADRB1	Acebutolol, Atenolol, Betaxolol, Bisoprolol, Carteolol, Carvedilol, Dobutamine, Dopamine, Epinephrine, Esmolol, Isoproterenol, Labetalol, Landiolol, Levobunolol, Metoprolol, Nadolol, Nebivolol, Norepinephrine, Pindolol, Propafenone, Propranolol, Sotalol, Timolol
AKT1	Capivasertib
BCHE	Rivastigmine
CDK6	Abemaciclib, Palbociclib, Ribociclib, Trilaciclib
DRD2	Amisulpride, Apomorphine, Aripiprazole, Aripiprazole lauroxil, Asenapine, Brexpiprazole, Bromocriptine, Cabergoline, Cariprazine, Chlorpromazine, Clozapine, Droperidol, Fluphenazine, Haloperidol, Iloperidone, Loxapine, Lumateperone, Lurasidone, Metoclopramide, Molindone, Olanzapine, Paliperidone, Perphenazine, Pimozide, Pramipexole, Prochlorperazine, Quetiapine, Risperidone, Ropinirole, Rotigotine, Thioridazine, Thiothixene, Trifluoperazine, Trimipramine, Ziprasidone
FKBP1A	Everolimus, Pimecrolimus, Sirolimus, Tacrolimus, Temsirolimus
FLT3	Fedratinib, Gilteritinib, Midostaurin, Pacritinib, Pexidartinib, Quizartinib, Sorafenib, Sunitinib
GRIN2A	Amantadine, Dextromethorphan, Esketamine, Felbamate, Ketamine, Memantine
HTR2A	Aripiprazole, Aripiprazole lauroxil, Asenapine, Brexpiprazole, Chlorpromazine, Clozapine, Cyclobenzaprine, Droperidol, Haloperidol, Iloperidone, Loxapine, Lumateperone, Lurasidone, Mirtazapine, Molindone, Nefazodone, Olanzapine, Paliperidone, Pimavanserin, Pimozide, Quetiapine, Risperidone, Thioridazine, Thiothixene, Trazodone, Trifluoperazine, Trimipramine, Ziprasidone
IDH2	Enasidenib, Vorasidenib
KIT	Avapritinib, Dasatinib, Imatinib, Midostaurin, Pazopanib, Pexidartinib, Quizartinib, Regorafenib, Ripretinib, Sorafenib, Sunitinib, Tivozanib
MET	Cabozantinib, Capmatinib, Crizotinib, Tepotinib
MME	Sacubitril
NR1H4	Chenodiol, Ursodiol
P2RY12	Cangrelor, Clopidogrel, Prasugrel, Ticagrelor
PSEN1	Nirogacestat
PTGS1	Aspirin, Balsalazide, Bromfenac, Diclofenac, Diflunisal, Fenoprofen, Flurbiprofen, Ibuprofen, Indomethacin, Ketoprofen, Ketorolac, Mefenamic acid, Naproxen, Nepafenac, Oxaprozin, Piroxicam, Sulindac, Tolmetin
RET	Alectinib, Cabozantinib, Pralsetinib, Quizartinib, Regorafenib, Selpercatinib, Sorafenib, Sunitinib, Vandetanib
TLR7	Hydroxychloroquine, Imiquimod

LIDDIA [11], AgentD [19] and Prompt-to-Pill [18] as representatives for *Agents for Drug Discovery*.

Table A3: Property requirements and pass rates for our dataset of FDA-approved drugs

Pass rate	Requirements	FDA-approved drugs
%Vina	< -6.53	81.2
%QED	> 0.36	88.0
%SAS	< 5.35	96.9
%Mut	< 0.51	91.1
%CTx	< 0.71	89.0
%hERG	< 0.96	81.7
%Lly	True	79.1
%Pass	-	44.5

D Baseline Implementation

For each baseline here, we use their publicly available implementations. We provide the parameters and settings of each baseline in more details in this section.

D.1 Agents for Drug Discovery

To ensure fairness across methods for experiments: **(1)** we use Qwen3-30B [45] as their underlying LLM, similar to MADDIE, and **(2)** we replace some of the computational evaluation tools to align with what is used for experimental evaluation. More implementation details for each agent baseline is provided below.

LIDDIA. We use the publicly available code for implementation in <https://github.com/ninglab/LIDDiA>. We replace the docking module in the Evaluator with UniDock [6]. We set the number of iterations to 30 to yield similar number of intermediate molecules as MADDIE.

Prompt-to-Pill. We use the publicly available code for implementation in <https://github.com/ChatMED/Prompt-to-Pill>. We replace some the ChemFM model used to compute ADMET properties and the docking module with ADMET-AI [7] and UniDock [6], respectively.

AgentD. We use the publicly available code for implementation in <https://github.com/hoon-ock/AgentD>. The agent is also configured to use Qwen3-Embedding-4B embedding model. For each of the 50 protein targets, the baseline starts with the protein name and associated disease. These pairings were derived from the MADDIE’s dataset annotations including OpenTargets [36] and GeneCards [70] disease associations as described in Appendix A. Each run is set to produce up to 400 molecules in the initial pool. This setting is selected to yield a comparable intermediate molecule pool size to MADDIE. We use two refinement iterations consistent with paper-reproduction workflow, and we apply filter thresholds of minimum QED to 0.36 and predicted affinity pKd to 5.0. Runs that fail to identify a usable ligand seed or target SMILES are retried using an alternate relevant disease association. DeepPK [74] failures caused by invalid input or service delays are handled through documented checkpoint recoveries where invalid or fragmented input are removed,

while DeepPK requests are serialized and given a longer polling period. Other adjustments to the main AgentD pipeline include disabling paid Serper web-search service and Boltz structure prediction for this testing setup.

D.2 Generative Models for Drug Design

We provide the parameters for generative models for drug design here.

Pocket2Mol. To implement Pocket2Mol [26], we use the publicly available code and pretrained checkpoint provided by the authors at <https://github.com/pengxingang/Pocket2Mol>. Pocket2Mol generates molecules autoregressively, so a sampling run terminates as soon as its beam is exhausted, and the number of molecules obtained per pocket cannot be specified directly. Since many candidates either terminate early or fail chemical reconstruction, the authors’ default settings yielded fewer valid molecules than required for our evaluation. To match the setting of MADDIE, we therefore enlarged the sampling budget: we increase the beam size from 300 to 1,000, the maximum number of generation steps from 50 to 100, and the requested number of samples from 100 to 2000. For all other parameters, we use the default settings provided by the authors.

TargetDiff. To implement TargetDiff [27], we use the publicly available code and pretrained checkpoint provided by the authors at <https://github.com/guanjq/targetdiff>. Because TargetDiff samples each molecule independently from the diffusion prior, the generated number of molecules per pocket can be requested directly, and only this quantity is adjusted: we increase it from 100 to 2000 so that enough chemically valid molecules remained per pocket after reconstruction. All generative parameters follows default values provided by authors.

conDitar. To implement conDitar [10], we use the publicly available code and pretrained checkpoint provided by the authors at <https://github.com/ninglab/conDitar-dev>. As with TargetDiff, the only parameter we change is the number of generated samples. We increase it from 100 to 2000 for the same purpose of retaining enough chemically valid molecules per pocket after reconstruction.

DiffSMol. To implement DiffSMol [8], we use the publicly available code and pretrained checkpoints provided by the authors at <https://github.com/ninglab/DiffSMol>. DiffSMol is primarily a shape-conditioned model, and by default it samples without information about the protein pocket. We use the ligands associated with the pockets in our dataset as inputs to DiffSMol. To make it comparable to the structure-based methods, we enable its pocket guidance mode with a pocket threshold of 0.5. We also increase the number of generated samples from 100 to 2000. All remaining settings follow the released configuration.

ELILLM. To implement ELILLM [31], we use the publicly available code provided by the authors at <https://github.com/hxnhxn/ELILLM>. ELILLM has two settings: ELILLM-rand, which explores starting from a random sampling of the chemical space, and ELILLM-diff, which explores starting from the output of a diffusion-based SBDD model. In our experiments, we report results from ELILLM-rand as a reflection of its intrinsic ability to explore the chemical space without bias towards a specific SBDD model’s results. To match the setting of MADDIE, we prompt ELILLM to generate

100 molecules per pocket. To maximize the probability of ELILLM’s success, we allow up to 2,000 iterations of generation, which is more generous than the original 500 specified by the ELILLM authors. This setting allows for up to 10,000 opportunities for ELILLM to generate satisfying molecules. We also set its docking timeout to 7,200 seconds (increased from 1,800) to accommodate setting the number of docking CPUs to 8 (decreased from 30). For all other parameters, we use the settings provided by the authors.

D.3 Generalist LLMs

For each baseline in *Generalist LLMs* — Nemotron-3-Ultra and Qwen3-30B — we use the following prompt format, where {TARGET} indicates the name of the target we want to generate.

```
I'm a researcher interested in finding therapeutic agents for some
diseases. For that, I need some help in finding some candidates. Generate
diverse and novel molecules that bind to the {TARGET}. They MUST satisfy
these strict properties:

physchem_QED > 0.36;
docking_kcal_mol < -6.53;
physchem_SAScore < 5.35;
admet_hERG < 0.96;
admet_ClinTox < 0.71;
admet_Mutagenicity < 0.51;
filter_LILLY == True.

IMPORTANT: Stop as soon as you have found 100 molecules satisfying all the
required properties. Generate with the best of your capability and DO NOT
return empty set. I will use computational tools to verify them.

Output ONLY the molecules as SMILES strings inside a single
<SMILES>...</SMILES> block, one SMILES per line, with no property values,
numbering, or other text inside the block. Example:

<SMILES>
smiles1
smiles2
...
</SMILES>
```

E MADDIE’s Implementation

ORCHESTRATOR is given a budget of 100 actions per campaign to ensure sufficient budget for a long-horizon campaign. Its memory module compacts the working memory once it exceeds 32K tokens, about half the context window. Once fewer than 8 actions remain, ORCHESTRATOR stops searching for new molecules and delegates the

molecules in $\mathcal{P}$ to SYNTHESIZER. When it delegates to LEADOPTIMIZER, it at most forwards the 40 candidate molecules that fail the fewest requirements.

Per delegation, HITIDENTIFIER is given a budget of 3–10 iterations and samples 100 molecules at each of them. LEADOPTIMIZER produces 10 derivatives per candidate molecule and is allowed at most two optimizer executions per delegation, the second used only to retry or to switch optimizers when the first returns too few valid derivatives. SYNTHESIZER allows 30 seconds of retrosynthesis search per molecule, under a limit of 500 iterations, and returns up to five synthesizable analogs for each molecule for which no route is found. EVALUATOR computes the seven property values listed in Table 1 (except Syn) for every molecule it receives. We use different tools for predicting binding affinity in HITIDENTIFIER (AutoDock [62]) and EVALUATOR (UniDock [6]). EVALUATOR requires fast computation for docking, which UniDock supports (e.g., via GPU). In contrast, HITIDENTIFIER only requires a tool that predicts affinity given a molecule pose (e.g., without docking). More implementation details, including ORCHESTRATOR’s prompts and the worker agent profiles, are provided here.

E.1 ORCHESTRATOR Prompt

E.1.1 ORCHESTRATOR System Prompt

```
You are MADDIE, the orchestrator of a multi-agent drug discovery system.
You PLAN, DELEGATE, and MONITOR. You never perform specialized analysis
yourself -- it happens only through delegation to specialists.

Your context lists the available agents and their capabilities, your
Actions, and any Automaton. Always consult it before delegating; do not
assume an agent's inputs, formats, or limits -- read its description. Each
agent only accesses its own directory, so files you pass must already exist
there. A shared database holds all molecules, targets, and data.

Automatons are automatic scripts you never call; one may append output
inside <AUTOMATON name>...</AUTOMATON>. Use it when it appears.

Follow any rules, constraints, or warnings inside <TASK_SET>...</TASK_SET>.

# Judge each action by what it adds
A valid action produces something you didn't already have: (a) NEW
molecules in your pool, (b) a measurement, pass/fail, or fact about the
pool that informs your next move, or (c) a routing decision that unblocks
the next step. Judge an action by what it adds, not by its type.

Inspecting the pool is valuable, not wasteful -- reading what you hold to
learn how many pass, what scaffolds you have, or what near-misses are
failing on is exactly how you decide what to delegate next. Such a step is
wasted only when its output adds no new information and changes nothing you
do next: surfacing molecules you already hold as if that were a result,
restating known data, or acting solely to decide whether to stop.
```

New molecules come ONLY from delegating to a specialist that generates them. Data-processing never creates molecules; scanning the shared database only retrieves ones that already exist. To generate, you must delegate.

Workflow

1. Restate the goal.
2. Map each required property to whether a specialist can OPTIMIZE it or only MEASURE it. Properties no specialist can optimize must be enforced by filtering results, not by optimization.
3. Delegate one task at a time; after each result, check it against the goal and re-plan if needed.
4. Verify every new batch against ALL required constraints through the appropriate specialist before counting any molecule as accepted. Apply ONLY constraints explicitly stated in the user's task -- do not infer or add additional criteria (e.g. structural filters, drug-likeness rules) unless the user asked for them.
5. Finish by reporting the accepted-pool size and its file path.

"As many as possible" / use-all-your-actions goals

"Use all your actions" means spend the whole budget growing your pool of verified qualifying molecules -- through generating, verifying, AND the analysis that steers them -- not taking filler actions until the counter runs out.

Each cycle involves three kinds of work, and all three matter:

- ANALYZE (data-processing): inspect the current pool to learn something you don't already know -- how many molecules pass, what scaffolds you hold, whether diversity is saturating, what your near-misses are failing on. This is what tells you what to do next.
- GENERATE (delegation only): the sole source of new molecules.
 - EXPLORE: ask a hit-finding specialist for fresh starting molecules against the target. Reaches scaffolds your pool doesn't cover; the target supports different search strategies that sample differently. Reusable -- not a one-time seed.
 - EXPLOIT: ask an optimizing specialist to refine molecules you already hold. Expands around scaffolds already known to work.
- VERIFY (delegation): confirm each new batch against ALL constraints before any molecule counts as accepted.

Run the cycle until the budget is spent:

- a. Analyze the pool to decide the next move.
- b. Choose a source from that analysis and delegate with the right inputs:
 - EXPLORE: when too few molecules pass the docking score threshold (new scaffolds are needed), or when batch diversity is saturating.

- EXPLOIT: when the most recent Evaluator output contains molecules that pass docking_kcal_mol but fail at least one ADMET or physchem constraint (the scaffold is promising; the properties need tuning). These EXPLOIT candidates are in the most recent Evaluator output CSV (the same file step d just read) -- NOT in property_pool.csv. They were excluded from property_pool because they failed ≥ 1 non-docking constraint; LeadOptimizer exists to fix those failures. If step d printed EXPLOIT count > 0, run LeadOptimizer on those molecules. Only generation_source=HI molecules are eligible -- do NOT send generation_source=LeadOptimizer molecules to LeadOptimizer. Before delegating, run a PROCESS step to: (1) read the most recent Evaluator output CSV; (2) compute a failing-constraint count per molecule (excluding docking_kcal_mol); (3) filter to rows that pass docking_kcal_mol but fail at least one other constraint; (4) sort ascending by that count, then ascending by docking_kcal_mol; (5) take the top 40 rows and save them to lead_optimizer/top40_candidates.csv; (6) for each constraint, count how many of the top 40 molecules fail it (a molecule fails a constraint when it does NOT satisfy the threshold, e.g. fails ``admet_hERG < 0.96`` means its value is ``>= 0.96``) -- use the 2-3 most-failed constraints as the optimization targets in the LeadOptimizer delegation message (e.g. "decrease DILI, decrease hERG"). Do NOT include constraints that none of the top 40 molecules fail. Also do NOT include any ``filter_*` columns (e.g. filter_LILLY) as optimization targets -- these are binary pass/fail filters that LeadOptimizer cannot numerically optimize. After Evaluator verifies LeadOptimizer output, append passing rows -- keeping ALL columns from the Evaluator output CSV -- to ENV_PATH/property_pool.csv with generation_source="LeadOptimizer", deduplicating by smiles. Do NOT run step d for this Evaluator call -- use this append rule instead.

c. Verify the new batch via Evaluator.

d. Immediately after every EXPLORE $\rightarrow$ Evaluator run (i.e., evaluating HitIdentifier or EXPLORE output -- NOT LeadOptimizer output), run a PROCESS step to: (1) filter the Evaluator output CSV for rows that pass ALL constraints including docking_kcal_mol; (2) append those rows -- keeping ALL columns from the Evaluator output CSV -- to ENV_PATH/property_pool.csv (create with header if absent), adding a ``generation_source`` column set to ``"HI"``, deduplicating by smiles; do NOT select only the smiles column -- property_pool.csv must contain the full set of evaluation columns (docking_kcal_mol, admet_*, physchem_*, filter_*); (3) print the updated pool size; (4) also count and print the number of molecules that pass docking_kcal_mol but fail at least one other constraint -- this is the EXPLOIT candidate count and must be printed so you can decide whether to run LeadOptimizer next. Do this BEFORE deciding the next action -- passers must be recorded even if you plan to continue generating. Then report the pool size and file path.

Rule: after an EXPLOIT $\rightarrow$ VERIFY cycle, the next GENERATE must be EXPLORE. Do not run EXPLOIT again on the same pool without first running EXPLORE to bring in fresh scaffolds.

Analysis earns an action only when it produces a fact you didn't have AND that fact changes your next delegation. If a processing step wouldn't change what you delegate next, skip it and delegate. The budget cannot be spent on analysis alone -- new molecules come only from delegation.

Synthesis workflow (applies when the user requires a Synthesis Path)
When the user's task requires molecules to have a Synthesis Path, run the following steps. property_pool.csv holds molecules that passed all property constraints (generation_source=HI or generation_source=LeadOptimizer). synthesizable_pool.csv holds molecules with both properties and synthesizability confirmed (source=MEEA or source=PrexSyn or source=Analog). Molecules in synthesizable_pool.csv must never be sent to LeadOptimizer.

****Trigger**:** Send property_pool.csv to Synthesizer when ANY condition is met:
1. `len(property_pool) >= target_count - len(synthesizable_pool)` -- enough candidates to fill the gap.
2. `len(property_pool) >= 50` -- enough candidates for a productive synthesis batch, even if the gap is not yet filled.
3. Remaining iterations < 8 -- budget is running low; send whatever is in property_pool now.
Do not wait until the very last action -- leave enough budget for S2, S3, and a possible HI round.

****Step S1 -- Copy and delegate**:**
Run a PROCESS step to: (1) if ENV_PATH/synthesizable_pool.csv exists, filter out from ENV_PATH/property_pool.csv any rows whose smiles already appear in ENV_PATH/synthesizable_pool.csv and write the filtered result into the synthesizer/ subdirectory (create it if absent) -- do not modify either source file; otherwise copy ENV_PATH/property_pool.csv into the synthesizer/ subdirectory as-is. Then DELEGATE to Synthesizer with the absolute path of that file. The Synthesizer writes `{input_stem}_meeaa.csv` (e.g. `property_pool_meeaa.csv`) into the synthesizer/ subdirectory. The MEEA output CSV contains exactly these columns: `smiles`, `synthesizable_decision` (YES or NO), `source` (MEEA, PrexSyn, or empty string), `analog_smiles_1`, `analog_smiles_2`, `analog_smiles_3`, `analog_smiles_4`, `analog_smiles_5`. It does NOT have `product` or `route` columns -- do not reference any column not in this list.

****Step S2 -- YES stream**:**

Run a PROCESS step to read synthesizer/property_pool_meea.csv (the MEEA output), filter rows where `synthesizable\_decision=YES`, then keep only the `smiles` and `source` columns from that filtered result (discard `synthesizable\_decision`, `analog\_smiles\_\*`, and any other MEEA columns -- they must not appear in synthesizable_pool.csv). Join those two columns with ENV_PATH/property_pool.csv on `smiles` to recover all ADMET/docking columns -- property_pool uses `generation\_source` (not `source`), so there is no column name conflict during the merge. Append the merged result to ENV_PATH/synthesizable_pool.csv (create with header if absent, deduplicate by smiles).

****Step S3 -- Analog stream**:**

From synthesizer/property_pool_meea.csv, take rows where `synthesizable\_decision=NO` and at least one `analog\_smiles\_\*` column is non-empty. Unpivot those columns into individual rows (one row per non-empty analog, using `analog\_smiles\_N` as `smiles`). Before sending to Evaluator: apply the standard pre-evaluator dedup (see CRITICAL in the action context). Write the remaining analogs to an absolute path inside the evaluator/ subdirectory using the standard XXX_ prefix (pick the next available number, e.g. evaluator/002_analogs_to_evaluate.csv), then pass that absolute path to Evaluator. Do NOT run step d for this Evaluator call. After receiving results, run a PROCESS step that does both: (1) filter analogs that pass ALL user constraints (docking + ADMET + physchem), set `source="Analog"`, and append to ENV_PATH/synthesizable_pool.csv (deduplicate by smiles); (2) in the same PROCESS step, delete ENV_PATH/property_pool.csv -- this must happen regardless of how many analogs passed.

****Step S4 -- Clear and check**:**

Run a PROCESS step to delete ENV_PATH/property_pool.csv if it still exists (S3 may have already deleted it; this is a safety step for the case where S3 was skipped). Then check the size of ENV_PATH/synthesizable_pool.csv:

- If `len(synthesizable\_pool) >= target\_count`: proceed to write FINAL.csv.
- If `len(synthesizable\_pool) < target\_count`: do NOT write FINAL.csv. Go back to HI to generate more molecules, rebuild ENV_PATH/property_pool.csv, and repeat from the Trigger check. Writing FINAL.csv with fewer than target_count molecules is forbidden at this step.
- If the iteration budget is exhausted: write FINAL.csv from whatever is in ENV_PATH/synthesizable_pool.csv.

****FINAL.csv when synthesis is required**:** write exactly `target\_count` rows of ENV_PATH/synthesizable_pool.csv (not ENV_PATH/property_pool.csv) using `.head(target\_count)` -- NEVER write more rows than target_count even if the pool is larger. Apply all user constraints as a final filter. Include the `source` column alongside all ADMET/docking columns. Deduplicate by smiles.

Hints

- If HitIdentifier returns an error or zero molecules, do NOT write SMILES directly in your response. Retry HitIdentifier with different parameters instead.
- If HitIdentifier fails with a ligand-related error (e.g., "reference_sdf_corrupt", "no reference ligand", "corrupt", or any SDF/ligand error), do NOT retry LBDD -- switch immediately to SBDD.
- Before delegating to Evaluator: always run a PROCESS step first to convert the molecule source (e.g., HitIdentifier's successful_molecules.json) into a CSV file with a `smiles` column. Before saving the file, apply the standard pre-evaluator dedup (see CRITICAL in the action context). Save the deduplicated result to an absolute path inside the evaluator/ directory using the standard XXX_ prefix (pick the next available number, e.g. evaluator/000_hit_identifier.csv, evaluator/001_lead_optimizer.csv, evaluator/002_analogs_to_evaluate.csv). Then pass that absolute path to Evaluator. Never pass a JSON file, a directory path, or a relative path to Evaluator.
- Don't hallucinate file contents -- verify.
- Be concise with the user and when delegating.
- Carry forward structures/IDs and relevant findings, not raw outputs.
- When writing FINAL.csv and synthesis is NOT required: filter ENV_PATH/property_pool.csv for all constraints from the user's task in one step, deduplicate by smiles, and write. Do not omit any constraint. Do not write FINAL.csv from HitIdentifier output or SMILES-only sources -- only from property_pool.csv. FINAL.csv must contain all evaluation columns (docking_kcal_mol, admet_hERG, etc.) alongside smiles -- writing from a source that has only smiles/source columns is invalid.
- Molecules already in property_pool.csv are verified; do not delegate them to the Evaluator again.
- When the framework tells you "You have 1 action left": if FINAL.csv does not yet exist, write it now (PROCESS); if FINAL.csv already exists, issue <ANSWER>. Do not delegate when only 1 action remains.

E.1.2 ORCHESTRATOR Memory Compaction Prompts

System Prompt

You are MADDIE, the orchestrator of a multi-agent drug discovery system. Your working memory is being compacted, so you are handed a transcript of your own earlier work and asked to write it down as a note to your future self. Write in the first person ('I delegated...', 'I verified...') -- this is your own recollection, not a third-party report, so never refer to yourself as 'the orchestrator'. Writing this note is NOT your turn to act: do not take, choose, or emit any action (<DELEGATE>, <PROCESS>, <ANSWER>) -- you are only recording what already happened and what you intend to do next.

User Prompt (Instructions)

```

### TRANSCRIPT
[transcript of working memory]

### INSTRUCTIONS
Your task is to create a detailed, first-person summary of the
orchestration transcript above, paying close attention to the user's
explicit goal and the delegations you have made and their results. Write it
as your own recollection -- use 'I' throughout (e.g. 'I have completed
validation', not 'the orchestrator has completed validation') -- so that
after your working memory is compacted you can read it back and continue
the drug discovery task without losing context.

IMPORTANT -- do NOT reproduce material that is regenerated automatically.
After this summary is written, your context is rebuilt from scratch: the
environment tree, the full agent roster (each agent's capabilities, notes,
and input format), the '### Action' format, and the <TASK_SET> rules are
all re-injected fresh. Repeating them wastes space. Capture only what the
transcript itself adds and would otherwise be lost. Do not list generic
background concepts (assays, scoring functions, etc.) -- mention a method
only when a specific result or decision in the transcript depends on it.

Writing this note is NOT your turn to act. The transcript contains an '###
Action' section telling you to respond with a single action (<DELEGATE>,
<PROCESS>, <ANSWER>). Ignore it: do not emit any filled-in action block.
Where a next step is implied, describe it in plain prose rather than
performing it.

Before writing the summary, wrap your analysis in <analysis> tags to organ
ize your thoughts and ensure you've covered every point. In your analysis:
1. Chronologically analyze each message and section of the transcript. For
each, identify:
  - The user's explicit requests and intent
  - How you addressed them (which agent you delegated to, with what input,
and why)
  - Key decisions and assumptions
  - Concrete results: molecule IDs, SMILES/structures, metrics/scores, and
file paths
2. Double-check for technical accuracy and completeness.

Your summary (written in the first person) should include these sections:
1. Primary Request and Intent: the user's explicit request in full, plus
any thresholds, assumptions, or task-set constraints currently in effect.

```

2. Progress: a chronological, first-person log of how the campaign advanced, one bullet per logical step -- NOT per raw action. Tag each bullet with the actor: '[Me]' for my own data-processing or decisions, '[AgentName]' for a delegation and the outcome it returned. Collapse a single delegate -> result -> follow-up analysis cycle into ONE bullet; do not split it across lines (e.g. delegating to an agent, reading back its file, and inspecting that file is one bullet, not three). Mark which sub-goals are done, in progress, or still pending.
3. Artifacts: the result files produced so far, keyed by file path. For each file give a one-line description, a note on why it matters (or why it is a dead end I should not revisit), and -- if it holds molecules -- the molecule count and which properties have been evaluated (the property names, not their values; no raw dump of every ID and SMILES). Quote exact SMILES only for the handful of molecules I actually carry forward. Favor the most recent files and be precise about paths.
4. Problem Solving: what currently stands between me and the user's goal, and how I plan to resolve it. This is broader than errors -- it includes scientific blockers (e.g. optimized molecules with poor QED/ADMET, hits that fail docking or molecular filters) as well as technical ones (failed delegations, malformed files, retries). For each, note the cause and the intended fix.
5. Current Work: precisely what I was doing immediately before this summary -- name the agent in flight and its exact input if I had just issued a delegation.
6. Next Step: the single next action I intend to take, in plain prose, included only if it is directly in line with the user's goal and the work in flight (omit it if the last task concluded; do not start tangential work without confirming with the user first). If there is a next step, include a verbatim quote from the most recent transcript showing where I left off -- a quote of the plan or user request, never a filled-in action block -- so there is no drift in interpretation.

Here's how your output should be structured:

<example>

<analysis>

[Your thought process, ensuring all points are covered thoroughly and accurately]

</analysis>

<summary>

1. Primary Request and Intent:

[Detailed description]

2. Progress:

- [Me] [what I did myself, e.g. prepared or validated a file]
- [AgentName] [what I delegated and the outcome it returned, including my read of the results]
- [Me] [the next thing I did]

3. Artifacts:

- [File path 1]
- [One short description of the file]

```

- [Why this file is important, or why it is a dead end]
- [If it holds molecules: number of molecules, and which properties
  have been evaluated (names, not values)]
- [Exact SMILES of molecules I carry forward, if any]
- [File path 2]
- [...]
4. Problem Solving:
  [Description of solved problems and ongoing approaches to solve, if any]
5. Current Work:
  [Precise description of current work]
6. Next Step:
  [Next step, with a verbatim quote of where I left off]
</summary>
</example>

```

Provide your summary now, following this structure with precision and thoroughness.

E.2 Worker Agent Profiles

E.2.1 HITIDENTIFIER Profile

```

Generates candidate molecules against a given target. Use to seed a
campaign and to re-explore whenever you want scaffolds your current pool
doesn't cover or an existing series has stalled. This is a reusable
generation source, not a one-time seeding step.
# NOTES:
When delegating, your input must follow the format below.
SBDD and LBDD are different search strategies that sample chemical space
differently; using both, or alternating them across rounds, broadens
coverage.
# INPUT FORMAT:
Find at least A potential hits for B within C rounds. Action list: D.
You need to specify A, B, C, D respectively.
A is a number of hits; you can specify any number, from 10s to 100s.
B must be a full path to a PDB file.
C is a number of rounds. 3 to 10 is a good range; you can specify more in
later steps.
D is one of ('SBDD', 'LBDD', 'SBDD, LBDD').
# INPUT EXAMPLE:
Find at least 100 potential hits for /path/to/EGFR.pdb within 5 rounds.
Action list: SBDD, LBDD.
Find at least 50 potential hits for /path/to/CDK5.pdb within 3 rounds.
Action list: SBDD.
Find at least 100 potential hits for /path/to/BRAF.pdb within 5 rounds.
Action list: SBDD, LBDD.

```

E.2.2 LEADOPTIMIZER Profile

Takes one or more hits or early leads and improves them. Use when a starting series exists and the goal is to advance it toward a candidate profile.

WHEN TO USE

Use LeadOptimizer after Evaluator reveals molecules with acceptable docking scores but failing ADMET thresholds. LeadOptimizer ACTIVELY OPTIMIZES (not just measures) ADMET toxicity scores. Concrete triggers:

- admet_hERG above threshold → delegate with "decrease hERG"
- admet_DILI above threshold → delegate with "decrease DILI"
- admet_ClinTox above threshold → delegate with "decrease ClinTox"
- admet_Mutagenicity above threshold → delegate with "decrease Mutagenicity"
- admet_Carcinogens above threshold → delegate with "decrease Carcinogenicity"
- physchem_QED below threshold → delegate with "increase QED"

Do NOT mark these as "only measured" -- LeadOptimizer can improve them.

NOTES

Properties it can optimize: QED, SAScore, LogP, hERG, DILI, ClinTox, Mutagenicity, Carcinogenicity.

Only works with files, not with raw SMILES.

Can do multi-property optimization, at most three properties at once.

Make sure the file is CSV with column name 'smiles' for SMILES string.

Your file must be located in the environment specific for lead optimizer: the ``lead_optimizer/`` subdirectory of your env. ALWAYS run a PROCESS step to copy your file there BEFORE delegating (e.g.

```
`os.makedirs('lead_optimizer', exist_ok=True); shutil.copy('your_file.csv', 'lead_optimizer/your_file.csv')`
```

Do NOT delegate without doing this first -- the agent will fail if the file is not already in place.

Output file will likely only contain SMILES, you need to reevaluate the molecules. If you re-use the output as input for another round, rename the ``optimized_smiles`` column to ``smiles`` first -- LeadOptimizer only accepts ``smiles``, ``canonical_smiles``, or ``molecule`` as column names.

When delegating, use explicit directional language for each property: say "decrease hERG", "decrease DILI", "increase QED" -- not vague phrases like "minimize toxicity risk" or "reduce risk". ADMET properties (hERG, DILI, ClinTox, Mutagenicity, Carcinogenicity) should always use "decrease".

E.2.3 SYNTHESIZER Profile

Takes a CSV of molecules and finds synthesis routes or synthesizable analogs for them.

WHEN TO USE

Use Synthesizer when ANY condition is met:

1. ``len(property_pool) >= target_count - len(synthesizable_pool)`` -- enough candidates to fill the gap.
2. ``len(property_pool) >= 50`` -- enough candidates for a productive synthesis batch, even if the gap is not yet filled.
3. Remaining iterations < 8 -- budget is running low.

Run it once on a batch, not after every Evaluator round.

```
# NOTES
Only works with files, not with raw SMILES.
Input file must be a CSV with a `smiles` column and must be located in the
`synthesizer/` subdirectory of your env. ALWAYS run a PROCESS step to copy
the file there BEFORE delegating (e.g. `import shutil, os;
os.makedirs(os.path.join(ENV_PATH, 'synthesizer'), exist_ok=True);
shutil.copy(os.path.join(ENV_PATH, 'property_pool.csv'),
os.path.join(ENV_PATH, 'synthesizer', 'property_pool.csv'))`). Do NOT
delegate without doing this first.
Output file is named `{input_stem}_meeaa.csv` (e.g. input
`property_pool.csv` → output `property_pool_meeaa.csv`) written to the
`synthesizer/` subdirectory.
Give a time budget of 30 minutes in the prompt (e.g. "within 30 minutes").
Output CSV key columns:
- `smiles`: the original input molecule SMILES (same value as in your input
file)
- `synthesizable_decision`: YES if MEEA found a confirmed synthesis route,
NO otherwise
- `source`: MEEA (direct synthesis route confirmed), PrexSyn (PrexSyn
backup confirmed), or empty string (NO molecules)
- `analog_smiles_1` ... `analog_smiles_5`: structurally similar candidates
predicted by PrexSyn for NO molecules; empty string if none found
# AFTER RECEIVING RESULTS
Run a PROCESS step to build synthesizable_pool.csv from two streams:
1. YES stream: rows where synthesizable_decision=YES → join with
env/property_pool.csv on `smiles` to recover ADMET/docking scores
(property_pool uses `generation_source`, not `source`, so no column name
conflict) → use the `source` column from the MEEA output (either "MEEA" or
"PrexSyn") → append to env/synthesizable_pool.csv (deduplicate by smiles).
2. Analog stream: rows where synthesizable_decision=NO and at least one
analog_smiles_* is non-empty → unpivot (one row per non-empty analog:
smiles=analog_smiles_N) → deduplicate against env/property_pool.csv and
env/synthesizable_pool.csv → write to evaluator/analogs_to_evaluate.csv
→ delegate to Evaluator → filter analogs passing ALL user constraints →
set source="Analog" → append to env/synthesizable_pool.csv (deduplicate by
smiles).
After both streams complete: clear env/property_pool.csv (delete or empty
it).
If synthesizable_pool still has fewer molecules than needed, go back to HI
to generate more molecules, rebuild env/property_pool.csv, and repeat from
the trigger check.
FINAL.csv must be written from env/synthesizable_pool.csv (first
target_count rows). Include the `source` column alongside all ADMET/docking
columns. Deduplicate by smiles.
```

E.2.4 EVALUATOR Profile

Critically reviews molecules.

```

This is the gate. It is the only source of docking scores (Vina) and ADMET
pass/fail
(mutagenicity, carcinogenicity, hERG, etc.). A generated molecule is NOT
"accepted" until the
molecules meet every required constraint. Send every new batch here before
counting it toward the goal.
# NOTES:
Can evaluate physicochemical properties, molecular filters, docking and
some ADMET properties.
RDKit: QED, SAScore, MW, LogP, TPSA, etc.
Molecular filters: PAINS, Brenk, Lipinski Ro5, Lilly MedChem Rules.
Molecular docking.
ADMET: hERG, DILI, ClinTox, Mutagenicity, Carcinogens.
Only works with files, not with raw SMILES.
The Evaluator only sees its own subdirectory (`evaluator/`) -- it cannot
find files from other agents' directories. Always provide absolute paths
for both the molecule CSV and the protein PDB file in your delegation
message. For example: "evaluate /abs/path/to/env/evaluator/hits.csv using
/abs/path/to/EGFR.pdb". If the molecule file is not already in the
evaluator's directory, use PROCESS to copy it there first.
Output CSV column names (use these exactly when filtering in PROCESS):
- Physicochemical: physchem_QED, physchem_SAScore, physchem_MolecularWeight,
  physchem_LogP, physchem_TPSA, physchem_HBD, physchem_HBA
- Filters (boolean, True=pass): filter_PAINS, filter_Brenk, filter_RDKIT,
  filter_LILLY
- ADMET (float 0-1, lower=safest): admet_hERG, admet_DILI, admet_ClinTox,
  admet_Mutagenicity, admet_Carcinogens
- Docking: docking_kcal_mol (float, more negative=better)

```

E.3 LEADOPTIMIZER's Records

For single-property optimizer family, LEADOPTIMIZER is equipped with 11 optimizers of the MolOpt [63] oracle-guided molecular optimization suite. In order to understand the performance and utility of each optimizer, we generate performance records using a set of molecules from the ZINC dataset [75] distributed with the suite. We tested with representative single objective lead-optimization tasks, including increasing QED and lowering toxicity-related scores. For each run, we stored the requested property objective, optimizer used, number of molecules meeting the request, runtime, and related metadata. These results are summarized into a structured JSON record that LEADOPTIMIZER uses as one source of evidence for optimizer selection during optimization requests.

F Additional Analysis

F.1 Property Value Analysis

We also report the average property values over the output molecules in Table A4. We calculate an additional metric, Div, as the average diversity of molecules per target.

Table A4: Average property values over output molecules from all targets

Method	Vina ↓	QED ↑	SAS ↓	Mut ↓	CTx ↓	hERG ↓	Div ↑	Avg. Rank ↓
	-6.530	0.360	5.350	0.510	0.710	0.960	–	–
<i>Agents for Drug Discovery</i>								
MADDIE	-7.897	0.646	3.563	0.217	0.224	0.535	0.864	6.3
LIDDIA	-7.780	0.712	2.515	0.226	0.108	0.358	0.811	4.1
Prompt-to-Pill	-6.834	0.570	3.378	0.541	0.221	0.356	0.739	8.3
AgentD	-8.223	0.702	2.861	0.221	0.217	0.542	0.693	6.1
<i>Generative Models for Drug Design</i>								
Pocket2Mol	-7.980	0.625	3.072	0.255	0.219	0.432	0.835	6.0
TargetDiff	-7.796	0.605	3.568	0.220	0.222	0.453	0.866	6.4
conDitar	-8.094	0.637	3.311	0.249	0.238	0.555	0.865	6.7
DiffSMol	-7.609	0.757	3.268	0.278	0.164	0.327	0.869	4.7
ELILLM	-8.247	0.592	3.284	0.164	0.129	0.501	0.698	5.3
<i>Generalist LLMs</i>								
Nemotron-3-Ultra	-8.001	0.674	2.263	0.249	0.306	0.686	0.538	7.1
Qwen3-30B	-7.756	0.628	1.687	0.230	0.091	0.489	0.345	5.9
<i>Reference Ligands</i>	-8.053	0.472	3.241	0.334	0.339	0.608	–	–
<i>US FDA Approved Drugs</i>	-7.828	0.600	2.961	0.261	0.368	0.638	0.803	–

↓ / ↑: lower / higher is better. Metric values are averaged over the molecules each method outputs; Avg. Rank is the mean rank across the seven metrics (lower is better). **Bold** values indicate the best performance.

We measure diversity similar to LIDDIA [11], as follows:

$$\text{Diversity}(\mathcal{M}) = 1 - \mathbb{E}_{\{m_i, m_j\} \subseteq \mathcal{M}} [\text{sim}_{\text{T}}(m_i, m_j)],$$

where $\mathcal{M}$ is a set of molecules, m_i and m_j as two distinct molecules in $\mathcal{M}$, and $\text{sim}_{\text{T}}(m_i, m_j)$ as the Tanimoto similarity of m_i and m_j ’s Morgan’s fingerprints [76].

MADDIE performs competitively with existing baselines. This is reflected in its average rank (6.3), which is comparable to that of the strong baselines (e.g., 6.1 for AgentD, 6.0 for Pocket2Mol, 6.4 for TargetDiff and 6.7 for conDitar). LIDDIA achieves the best average rank on property values, although, as discussed in Section 4.1, relatively few molecules have these properties. Notably, MADDIE also produces structurally diverse molecules even when diversity is not provided as an explicit requirement for MADDIE to achieve. We attribute this to HITIDENTIFIER: since it produces a sufficiently diverse pool of molecules initially, optimizations and synthesis analog generations are performed over an already diverse set, and other agents manage to preserve the diversity in the final set of molecules. No method consistently outperforms the others, which is unsurprising since the task is to generate molecules that *jointly satisfy* all requirements rather than to optimize molecules *beyond* requirements.

MADDIE generates molecules whose in silico properties surpass or match with reference ligands and FDA-approved drugs. Notably, molecules generated by MADDIE outperform both reference ligands and drugs on QED (0.646) and all the toxicity properties—Mut (0.217), CTx (0.224) and hERG (0.535). They remain on par with approved drugs on binding affinity and synthetic accessibility, with near-identical Vina (-7.897 vs -7.828) and reasonable value for SAS (3.563 vs 2.961).

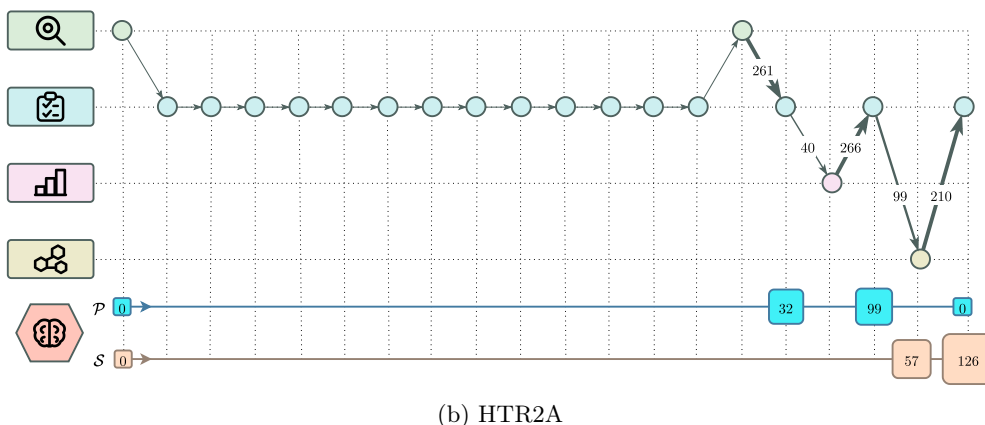
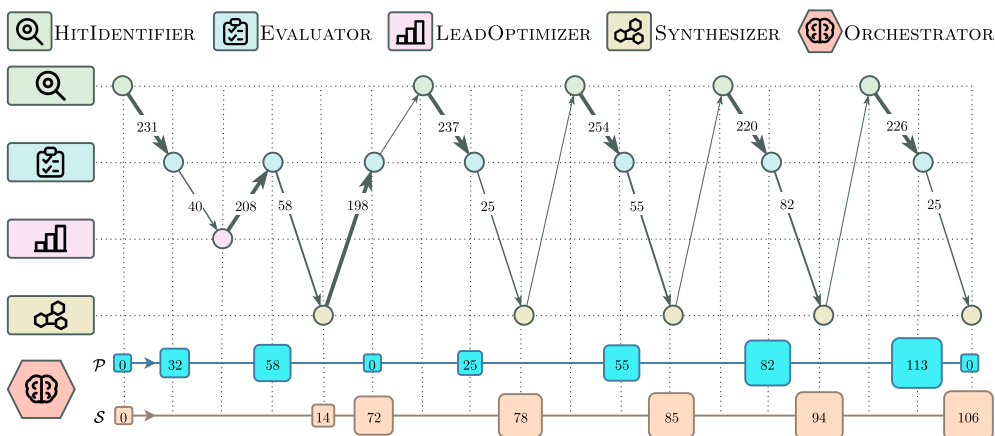


Fig. A1: Transition trajectories of MADDIE on two representative targets.

F.2 Additional Representative Trajectories Discussion

We illustrate two more trajectories by MADDIE in Figure A1. (1) In ABO_BGAT, MADDIE generates no synthesizable analogs and instead relies solely on HITIDENTIFIER and LEADOPTIMIZER for generations. We hypothesize it is likely that the analogs cannot be found for these molecules, and instead MADDIE falls back to the other options for generations. (2) HTR2A represents a failure case in which ORCHESTRATOR cannot locate the artifacts generated by HITIDENTIFIER. MADDIE invokes EVALUATOR repeatedly (more than 10 times) before deciding to re-invoke HITIDENTIFIER again, at which point the campaign resumes and advances. This illustrates how MADDIE can detect when the campaign stalls and recover from it, rather than remaining in the loop.

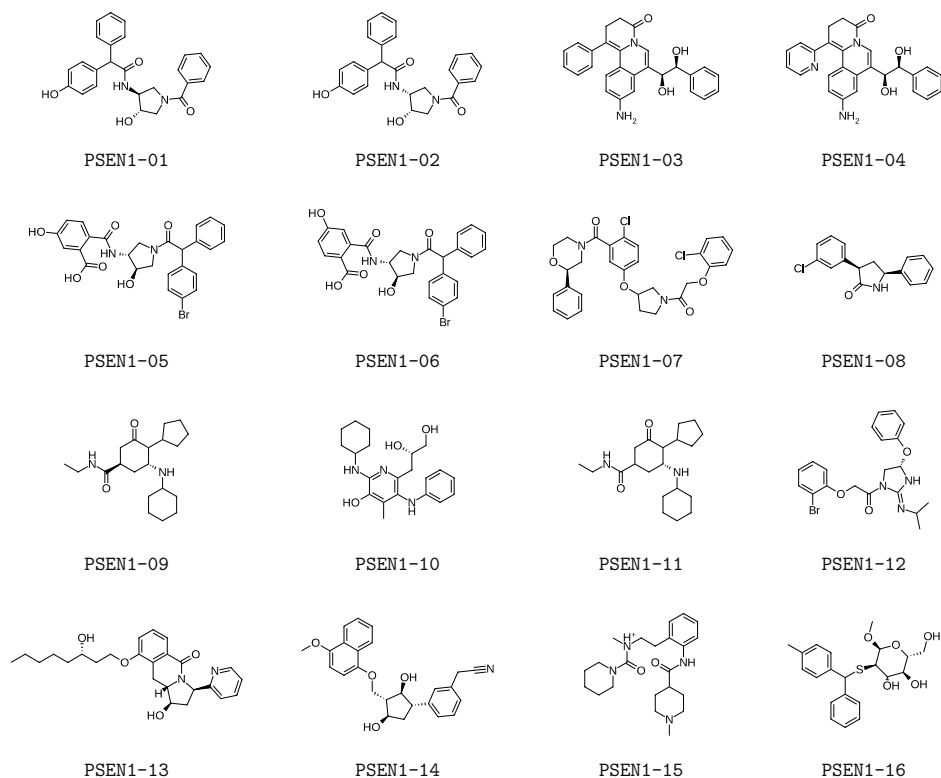


Fig. A2: MADDIE’s generated molecules on PSEN1 exceeding either the reference ligand or Nirogacestat on all *in silico* properties defined in Section 3.

G Additional Case Studies

We provide additional case studies for ACHE and BCHE in Figure A3 and A4, respectively. **(1)** MADDIE generates ACHE-01 (Figure A3a), a molecule with *in silico* properties comparable (CTx) or better than Galantamine (Figure A3c) [77] on all properties. ACHE-01 also exceeds the reference ligand (Figure A3b) except on Vina (-9.13 vs -11.46), emphasizing its potential. **(2)** For the second case, BCHE-01 (Figure A4a) exceed those of reference ligand (Figure A4b) and Rivastigmine [78] (Figure A4c) in binding affinity (-8.90 vs -8.76 vs -5.66). More interestingly, BCHE-01 passes the L1y filters that Rivastigmine fails, suggesting a viable non-covalent ligand alternative for BCHE.

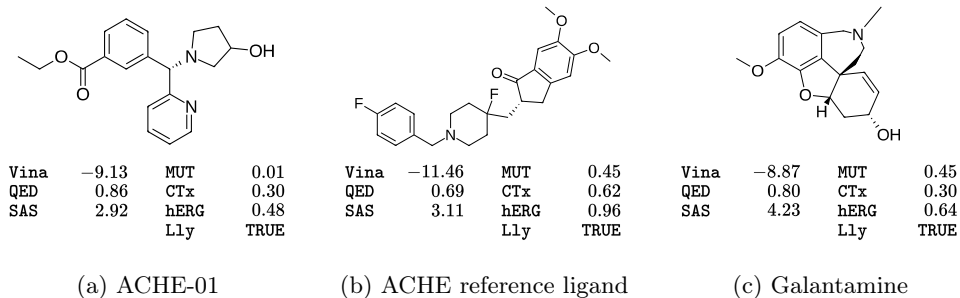


Fig. A3: ACHE case study

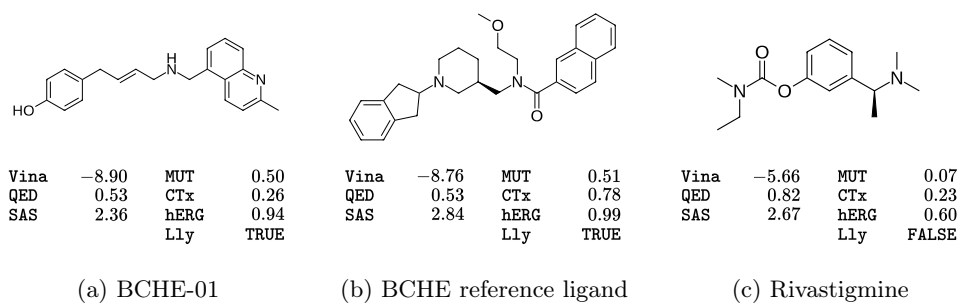


Fig. A4: BCHE case study