RETRO-R1: LLM-based Agentic Retrosynthesis

Wei Liu[1,3,4], **Jiangtao Feng**[2,3], **Hongli Yu**[2], **Yuxuan Song**[2,5],
Yuqiang Li[3,4], **Shufei Zhang**[3], **Lei Bai**[3], **Wei-Ying Ma**[2], **Hao Zhou**[2,3] *
[1]School of Computer Science, Shanghai Jiao Tong University
[2]Institute for AI Industry Research (AIR), Tsinghua University
[3]Shanghai Artificial Intelligence Laboratory [4]Shanghai Innovation Institute
[5]Department of Computer Science and Technology, Tsinghua University
liuwei@sii.edu.cn, yuhl25@mails.tsinghua.edu.cn
{liyuqiang, zhangshufei, bailei}@pjlab.org.cn
{fengjiangtao, songyuxuan, maweiying, zhouhao}@air.tsinghua.edu.cn

Abstract

Retrosynthetic planning is a fundamental task in chemical discovery. Due to the vast combinatorial search space, identifying viable synthetic routes remains a significant challenge–even for expert chemists. Recent advances in Large Language Models (LLMs), particularly equipped with reinforcement learning, have demonstrated strong human-like reasoning and planning abilities, especially in mathematics and code problem solving. This raises a natural question: Can the reasoning capabilities of LLMs be harnessed to develop an AI chemist capable of learning effective policies for multi-step retrosynthesis? In this study, we introduce RETRO-R1, a novel LLM-based retrosynthesis agent trained via reinforcement learning to design molecular synthesis pathways. Unlike prior approaches, which typically rely on single-turn, question-answering formats, RETRO-R1 interacts dynamically with plug-in single-step retrosynthesis tools and learns from environmental feedback. Experimental results show that RETRO-R1 achieves a 55.79% pass@1 success rate, surpassing the previous state of the art by 8.95%. Notably, RETRO-R1 demonstrates strong generalization to out-of-domain test cases, where existing methods tend to fail despite their high in-domain performance. Our work marks a significant step toward equipping LLMs with advanced, chemist-like reasoning abilities, highlighting the promise of reinforcement learning for enabling data-efficient, generalizable, and sophisticated scientific problem-solving in LLM-based agents.

1 Introduction

Retrosynthetic planning, a fundamental problem in organic chemistry, starts from a target molecule and works backward to find a pathway to transform it into commercially available starting materials. As a crucial task in drug discovery, retrosynthetic planning poses a significant computational challenge due to its vast and discrete combinatorial search space [1]. Successfully navigating this space requires not only deep chemical knowledge of reactions and reagents but also sophisticated strategic reasoning to evaluate a multitude of possible intermediate compounds and multi-step reaction pathways [2]. Moreover, retrosynthesis is inherently a long-horizon reasoning task, demanding higher-order chemical logic: decisions made in the early stages of the pathway can profoundly affect downstream options and ultimately determine the feasibility of the entire synthesis route [3].

Building on advances in single-step retrosynthesis prediction [4, 5], machine learning techniques have increasingly been applied to the more complex task of planning complete multi-step retrosynthesis

*Corresponding author. This work was done while Wei Liu was a research intern at Institute for AI Industry Research (AIR), Tsinghua University. Jiangtao Feng was the project leader.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

pathways [6]. These approaches often formulate the problem as a search over a vast graph of molecules and reactions, leveraging sophisticated algorithms guided by learned models [7, 8]. Researchers typically follow two main strategies: (1) enhancing single-step models to favor reactions that are more suitable for retrosynthetic planning [9, 10], or (2) training deep neural networks to serve as scoring functions for reaction feasibility or molecular complexity, which are then integrated into search algorithms such as Monte Carlo Tree Search (MCTS) [1, 11] or A* [12].

Recently, Large Language Models (LLMs) have exhibited remarkable capabilities in complex planning and reasoning tasks involving long sequences and intricate dependencies [13, 14]. Their ability to generate coherent text over extended contexts, follow complex instructions [15], leverage external tools [16], and perform multi-step logical deductions [17] has driven breakthroughs in domains such as complex question answering [18, 19, 20], math problem solving [21, 22, 23], code generation [24, 25, 26], and agent-based simulations [27, 28]. A major enabler of these capabilities has been the integration of Reinforcement Learning (RL), including techniques such as reinforcement learning from human feedback (RLHF) [29], proximal policy optimization (PPO) [30], dynamic sampling policy optimization [31], and attention-guided policy optimization [32]. These emerging strengths—especially in handling long-range dependencies and executing sequential reasoning—are highly relevant to the challenges of multi-step retrosynthetic planning. They suggest that LLMs can serve as powerful, flexible, and context-aware agents for navigating and reasoning through complex chemical synthesis pathways.

Inspired by recent advances in both machine learning for chemistry and general-purpose LLM reasoning, we propose RETRO-R1 , a novel LLM-based agent for multi-step retrosynthetic planning. Trained via end-to-end reinforcement learning, RETRO-R1 is capable of making complex decisions to select optimal reaction candidates and adapt to dynamic changes in the environment. We employ Proximal Policy Optimization (PPO) [30] and introduce modifications to Generalized Advantage Estimation (GAE) [33] to effectively support multi-turn interactions and operate over structured state representations. To emulate the iterative, knowledge-accumulating process of a human chemist, our framework enables the agent to conduct multiple planning attempts while incrementally constructing a global synthesis graph. This design allows for efficient exploration of the solution space and facilitates the discovery of high-quality synthesis routes within a practical computational budget.

Our key contributions are summarized as follows:

- We introduce a novel framework for training an LLM agent in end-to-end multi-step retrosynthetic planning, formulating the task as a sequential decision-making problem with environment feedback.
- We adapt Proximal Policy Optimization (PPO) to agentic learning with frequent environment interaction, enabling effective training in structured planning settings.
- We design an iterative planning strategy that builds a global synthesis graph across multiple sequential searches, leveraging memory to avoid redundant computations and emulate a chemist's comprehensive exploration process.
- We demonstrate that our framework achieves state-of-the-art or highly competitive results on benchmark retrosynthetic planning datasets, exhibiting strong generalization and discovering shorter synthesis routes than existing methods.

2 Background

Single-step retrosynthesis aims to identify a set of reactants $\mathcal{S} \subset \mathcal{M}$ from the space of all molecules $\mathcal{M}$ that can be used to synthesize a given target molecule $t \in \mathcal{M}$. A single-step retrosynthesis model, often denoted as B, can be described as $B(\cdot) : t \rightarrow \{R_i, \mathcal{S}_i, c(R_i)\}_{i=1}^{k}$. For a given target t, the model proposes up to k candidates. Each candidate consists of a reaction R_i, its corresponding set of reactants $\mathcal{S}_i = \{r_j\}_{j=1}^{|\mathcal{S}_i|}$, and an associated cost $c(R_i)$. In this work, we directly use the single-step retrosynthesis models from prior studies [12, 9, 10].

Retrosynthetic planning seeks to discover a complete reaction pathway for a target molecule $t \in \mathcal{M}$, starting from a given set of available initial molecules (i.e., building blocks) $\mathcal{I} \subset \mathcal{M}$. A reaction pathway, denoted by τ, is a set of chemical reactions $\tau = \{R_1, R_2, \ldots, R_N\}$, where each reaction R_i transforms a set of reactants $\mathcal{S}_i$ into a product p_i. Note that while a reaction may yield multiple products, we typically focus on a single desired product. A valid pathway τ must satisfy the following

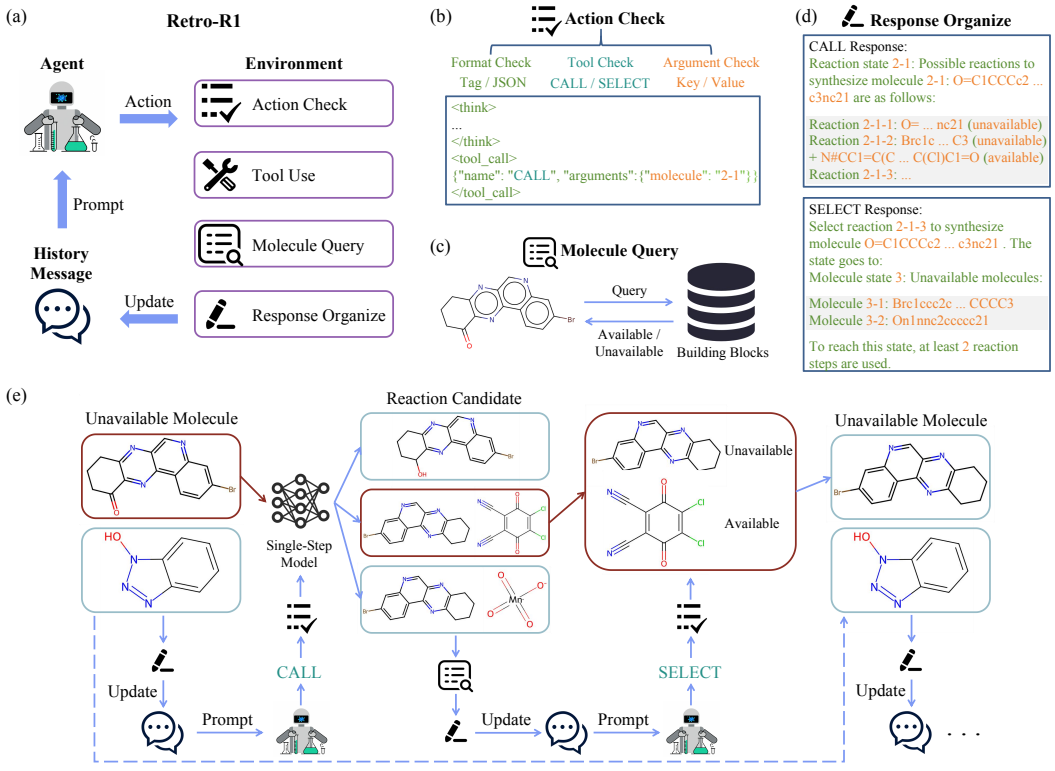

Figure 1: Overview of RETRO-R1 framework. **a** The workflow of RETRO-R1 . The agent interacts with the environment by using Actions. The environment automatically executes four types of operations: Action Check, Tool Use, Molecule Query, and Response Organize. The history message is organized in the form of a multi-turn dialogue, containing all the interaction history between the agent and the environment. The interaction continues until an exit condition is met. **b** Action Check is composed of three parts: Format Check, Tool Check, and Argument Check. Format Check verifies if the model's output correctly uses the designated <think> and <tool_call> tags and ensures the content of <tool_call> is in a valid JSON format. Tool Check validates whether the type of action is permissible in the current state, while Argument Check validates the parameters provided. **c** Molecule Query searches within the building blocks for a target molecule, returning a result of either "available" or "unavailable". **d** Response Organize structures the text response according to a specific format. If any of the other operations encounter an error, this module will return the cause of the error. Otherwise, it will provide the appropriate result corresponding to the type of action. **e** One molecular state transition consists of two interactions between the agent and the environment. Initially, the environment notifies the agent which molecules are currently "unavailable". The agent selects one of these molecules and uses the 'CALL' action to address it. In response, the environment invokes a Single-Step Model to generate several reaction candidates. The agent then uses the 'SELECT' action to pick one of these reactions, prompting the environment to update the molecular state.

conditions: (1) **Reactant Availability:** Every reactant $r \in \mathcal{S}_i$ for any reaction $R_i \in \tau$ must either be a member of the initial building block set $\mathcal{I}$ or be the product p_j of another reaction $R_j \in \tau$ (where $j \neq i$); (2) **Intermediate Connectivity:** The product p_i of any reaction $R_i \in \tau$ must either be the final target molecule t or serve as a reactant in another reaction $R_j \in \tau$ (where $j \neq i$); (3) **Target Uniqueness:** There must be one and only one reaction in τ whose product is the target molecule t; (4) **Chemical Feasibility:** All reactions proposed in the pathway must be chemically viable.

3 Methodology

3.1 Overview

We introduce RETRO-R1 , a novel LLM-based agent for multi-step retrosynthetic planning. As shown in Fig. 1, the retrosynthetic planning process is formulated as a sequential decision-making task within a structured search space. A molecular state $\mathbf{M}_i = \{m_{i1}, m_{i2}, \ldots, m_{in}\}$ contains all the

unavailable molecules. The initial molecule node $\mathbf{M}_1 = \{t\}$ contains only the target molecule. The goal molecule node $\mathbf{M}_e = \varnothing$ contains no molecules. One molecular state transition consists of two interactions between the agent and the environment.

The rest of this section is organized as follows. In Sec. 3.2, the workflow of RETRO-R1 is described. In Sec. 3.3, we detail the reinforcement learning algorithm to train the agent end-to-end. In Sec. 3.4, we describe the iterative planning strategy that enables comprehensive exploration and searches for the shortest route. We implement the framework based on previous works verl [34] and Agent-R1 [35].

3.2 Agent Workflow

We propose a novel workflow that leverages a large language model (LLM) agent trained with reinforcement learning (RL) to navigate the retrosynthesis search space through multi-turn interactions with the environment and strategic tool use. The LLM acts as the policy $\pi_\theta(a|s)$, mapping the current state representation to a probability distribution over possible actions.

State s: The state provided to the LLM agent is the tokens of the history message, which consists of the question, the agent's previous actions, and the environment's responses. As shown in Fig. 1d, the final message from the environment can contain one of the following: the molecules that are currently unavailable, reaction candidates for a selected molecule, or an error message caused by the last interaction.

Action a: The agent takes action by generating texts with specific tags and tool call requests in JSON format, as shown in Fig. 1b. Two actions are designed:

- **CALL**: Selects a molecule from the current molecular state and uses the single-step model to propose potential reactions. This action is only valid if the last message from the environment enumerated the unavailable molecules.
- **SELECT**: Selects a reaction candidate to update the molecular state. This is done by removing the reaction's product and adding its unavailable reactants to the state. This action is only valid if the last message provides a list of reaction candidates.

Environment: The environment drives the retrosynthetic planning process. It manages the molecular state, interacts with the agent, checks for termination conditions, and calculates the final reward. In its interactions with the agent, it executes four operations: Action Check, Tool Use, Molecule Query, and Response Organize, as shown in Fig. 1.

- **Action Check:** Verifies whether the agent's action is valid. It is composed of three parts: Format Check, Tool Check, and Argument Check. Format Check verifies if the model's output correctly uses the designated <think> and <tool_call> tags and ensures the content of <tool_call> is in a valid JSON format. Tool Check validates whether the type of action is permissible in the current state, while Argument Check validates the parameters provided.
- **Tool Use:** For the CALL action, the single-step model is called for the selected molecule, and a set of reactions is proposed. The reactions with top-k probabilities form the reaction candidates. For the SELECT action, the molecular state is updated with set operations, where the product is removed and the unavailable reactants are added.
- **Molecule Query:** Searches within the building blocks for a target molecule. The result is either "available" or "unavailable".
- **Response Organize:** Structures the text response according to a specific format. If any of the other operations encounter an error, the cause of the error is returned. Otherwise, it will provide the corresponding result. For the CALL action, the reaction candidates are listed. For the SELECT reaction, the unavailable molecules are enumerated. Each item is assigned a unique identifier to facilitate selection by the agent.

Termination: A retrosynthetic planning process terminates successfully when the molecular state becomes an empty set. Conversely, the process terminates in failure if the molecular state is not empty by the time any of the following conditions are met: (1) the number of agent-environment interactions exceeds the maximum threshold; (2) the token count of the history message surpasses the LLM's context limit; (3) the CALL action fails to generate any reaction candidates, indicating an unsynthesizable molecule.

Reward: To guide the agent to successfully and efficiently discover the shortest possible synthesis routes, we have empirically designed a reward function based on prior work [36, 31, 37] and extensive experimentation. This function is composed of two components: (1) A success component that evaluates the outcome of the planning process: The agent receives a reward of +0.9 for successfully finding a synthesis path and a penalty of -1.0 for failure; (2) An efficiency component for successful runs that penalizes excessively long searches: Since the number of agent-environment interactions l in our design determines the upper bound of the reaction route length, minimizing l concurrently reduces both search time and route length. Therefore, for successful runs, if $l > 40$, a penalty of $(40 - l) \times 0.02$ is applied. No penalty is incurred otherwise.

3.3 Agentic Reinforcement Learning

We train the LLM policy $\pi_\theta(a|s)$ with Proximal Policy Optimization (PPO) [30], a robust policy gradient algorithm that optimizes a clipped surrogate objective to enable stable updates. Both the policy network (actor) π_θ and the state value function (critic) $V_\phi(s)$ are initialized with the original LLM. The implementation of PPO incorporates specific modifications tailored to our framework's multi-turn interactions with the environment and structured state representations.

Actor and Critic Updates with Environment Masking. The state s provided to the LLM agent includes a sequence of tokens representing the question, the agent's actions, and the environment's responses. To ensure that the agent's learning focuses solely on its own decision-making rather than predicting or replicating environment outputs, we define a token mask $I(s_t)$ where $I(s_t) = 1$ if s_t is an agent token and $I(s_t) = 0$ otherwise. The loss function for the policy network is written as:

$$\mathcal{L}^{\text{CLIP}}(\theta) = -\mathbb{E}_{x\sim\mathcal{D}, s\sim\pi_{\text{old}}(\cdot|x;E)} \left[\frac{1}{\sum_{t=1}^{|s|} I(s_t)} \sum_{t=1}^{|s|} I(s_t) \min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{1}$$

where $\mathcal{D}$ is the training set, x is a question sampled from $\mathcal{D}$, E is the environment, $r_t(\theta) = \frac{\pi_\theta(s_t|s_{<t})}{\pi_{\text{old}}(s_t|s_{<t})}$ is the ratio of the probability under the current policy θ to the probability under the policy used for data collection θ_{old}, $\hat{A}_t$ is the advantage estimate at timestep t, and ϵ is a small clipping parameter. This means the gradients are only computed with respect to the agent tokens, preventing the policy from being trained to regenerate the environment's information.

The loss function for the state value function is a squared error loss:

$$\mathcal{L}^{\text{VF}}(\phi) = I(s_t)(V_\phi(s_t|s_{<t}) - V_t^{target})^2 + (1 - I(s_t))V_\phi^2(s_t|s_{<t}) \tag{2}$$

which means the prediction target for environment tokens is set to zero. The value function is thus trained to estimate the value of agent tokens, without assigning value to the deterministic information provided by the environment.

Generalized Advantage Estimation with Unit Parameters. We utilize Generalized Advantage Estimation (GAE) [33] to compute the advantage estimates $\hat{A}_t$ used in the policy and value updates. GAE typically uses discount factor $\gamma \in [0, 1)$ and a weighting factor $\lambda \in [0, 1)$. In our specific setting, to avoid the influence of environment tokens on the return accumulation, we set both $\gamma = 1$ and $\lambda = 1$, and the GAE estimate simplifies to the Monte Carlo return (or TD(1) return). We further set the value of the terminal state $V_\phi(s_T)$ as zero, and then the advantage estimate $\hat{A}_t$ for a state s_t is calculated based on the sum of undiscounted rewards from time step t to the end of the sequence T, minus the value function's estimate $V_\phi(s_t)$:

$$\hat{A}_t = \sum_{k=t}^{T-1} R_k - V_\phi(s_t) \tag{3}$$

This configuration results in high-variance but unbiased estimates of the advantage, focusing the agent on maximizing the total undiscounted reward accumulated over an entire planning episode.

By incorporating environment masking and leveraging Monte Carlo returns via GAE($\gamma = 1, \lambda = 1$), our PPO implementation is adapted to effectively train an LLM agent that solves the problem by interacting with the environment.

3.4 Iterative planning strategy

While a single sequential search can yield a potential synthesis route, real-world chemical synthesis planning often involves exploring multiple alternative pathways and consolidating knowledge gained from different approaches. To emulate a chemist's comprehensive exploration process, we design an iterative planning strategy that builds a global synthesis graph across multiple sequential searches. A memory mechanism is implemented to record all the intermediate information and avoid redundant computations.

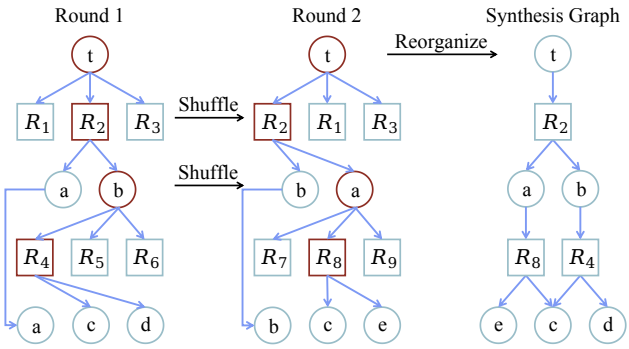

Figure 2: The iterative planning process of RETRO-R1 . t is the target molecule. A circle refers to a molecule, while a square refers to a reaction. Multiple rounds of search are conducted to explore the chemical space, with each round following the process detailed in Fig. 1e. The order of molecules or reactions is used in Response Organize (Fig. 1d). When a state is revisited, the order is shuffled to encourage exploration. Finally, the complete search history is reorganized into an AND-OR graph, where the shortest route is found.

As illustrated in Fig. 2, multiple rounds of search are conducted for the targeted molecule t to explore the chemical space. Each state is recorded by the memory mechanism with a default order. For example, the reaction candidates are ranked in descending order of the probabilities predicted by the single-step model. This order is shuffled if the state is revisited to encourage the agent to explore diverse selections. Finally, the complete search history is reorganized into an AND-OR graph. We implemented an algorithm to search for the shortest route following the prior work [38].

4 Experiments

4.1 Experimental Setup

This section describes the components of our experiments, including the dataset (building blocks, training and testing datasets), single-step models, baselines, and evaluation metrics.

Dataset. Following the common practice [12], we use 23,081,633 commercially available molecules from eMolecules [2] as the building blocks. For the training dataset, the single-step reactions come from USPTO reaction dataset [39]. Based on these reactions and building blocks, 299,202 multi-step synthesis routes are constructed in the prior work [12]. The length of the routes varies from 1 to 14. For baselines, all routes are used in the training. For our method, only 11,366 routes with lengths longer than 4 are used, which shows the data efficiency of our method. For the testing datasets, we use Retro*-190, which are 190 challenging target molecules from [12], each with a reference route, and ChEMBL-1000 [10], which is an out-of-distribution dataset used to test the generalizability of methods, consisting of 1000 molecules without reference routes.

Single-step Model. For a fair comparison with previous work, our framework utilizes a template-based MLP as the single-step retrosynthesis model [4]. It predicts probabilities over 381,302 USPTO reaction templates [39, 40] based on molecule Morgan fingerprints. We evaluate performance using four distinct weight versions (V1∼V4) from previous methods, which were trained on different reaction datasets (USPTO or iteratively on routes found by other planners) [12, 9, 11, 10]. While versions V2∼V4 trained on automatically found routes often yield higher metrics, they risk proposing chemically unrealistic reactions [41]. For fair comparison and better chemical intuition, we adopt a two-branch policy structure akin to [10]: V1 proposes the top-k reaction candidates, and the others rescore and rerank these same candidates. This ensures all compared methods operate within the same candidate search space ($k = 10$), focusing on high-probability, V1-proposed candidates. Our RETRO-R1 agent is trained using the V1 weights but is compatible with any other weights during

[2] https://downloads.emolecules.com/

testing. Weights V1∼V3 are downloaded from public codebases [3], while V4 is retrained following instructions from the PDVN codebase [4] as it is not publicly released.

Baseline. We compare our method against representative baselines on this task. We mark the weights of the single-step model in brackets for all baselines. Greedy Depth First Search (DFS) (V1) always prioritizes the reaction with the highest likelihood [12]. Retro* (V1) [12] is an A*-like algorithm where the past cost is calculated with the probability of the single-step model and the future cost is estimated with a neural value network. Retro*-0 (V1) [12] is its non-learning version where the future cost is set to 0. Retro*+ (V2) and Retro*+-0 (V3) [9] improve Retro* and Retro*-0, respectively, with self-improved single-step retrosynthetic models. EG-MCTS (V2) [11] improves Retro*+ with experience-guided Monte Carlo tree search. PDVN (V4) [10] trains a single-step model with an online training algorithm to improve Retro*-0.

Metric. The first metric is the **Pass@1 Success Rate** metric, inspired by the previous LLM works [36, 25]. This metric assesses a "no-regret" search strategy: when the single-step model proposes k reactions at a given molecular state, the method must select only one, and the other $k-1$ options at that node are permanently discarded. A Pass@1 search process is considered a failure under two conditions: (1) The route length exceeds the maximum limit of 30 (which is sufficient for our datasets); (2) The single-step model fails to propose any reactions for a molecule, resulting in a search dead end. Additionally, following previous work [12, 10], we use the **Success Rate** within a limit of 500 single-step model calls (i.e., iteration limit) for overall feasibility. To evaluate route quality, we measure the **Route Length** and the number of **Shorter Routes** found compared to the reference routes. These two metrics are calculated based on the results after the methods exhaust the 500 iteration limit. Since some baselines are originally designed to terminate once one route is found, we modified their code to ensure a fair comparison.

4.2 Implementation Details

Our framework is built upon verl [34] and Agent-R1 [35], with the implementation publicly accessible at https://github.com/GenSI-THUAIR/Retro-R1. We employ 'Qwen2.5-7B-Instruct-1M'[42, 43] as the policy network for reinforcement learning to achieve superior performance on our long-sequence tasks without additional continued pretraining or supervised fine-tuning. Efficient rollouts are conducted using vLLM[44] with a sampling temperature of 1.0 and a maximum sequence length of 32,768. For training, we apply a clipping ratio of 0.2 in the $\mathcal{L}^{\text{CLIP}}$ objective and incorporate a KL-divergence penalty with a coefficient of 0.001 into the token-level reward. Additionally, an entropy loss with a coefficient of 0.001 is utilized to prevent the model from generating overly chaotic outputs. Both the policy and value networks use Adam [45] as the optimizer. The learning rates of the policy network and value function are 10^{-6} and 10^{-5}, respectively. The rollout batch size is set to 128, while the mini-batch size for gradient updates is 64. During the initial 3 training steps, updates to the policy network are omitted to facilitate warm-up of the randomly initialized critic head.

4.3 Main Results

We evaluate the performance of our proposed RETRO-R1 against several baseline methods on two retrosynthetic planning testsets: retro*-190 is an in-domain dataset from USPTO reaction dataset [39] and ChEMBL-1000 is an out-of-domain dataset from ChEMBL database [46]. Results are summarized in Tab. 1 and Fig. 3. RETRO-R1 demonstrates strong performance on both retro*-190 and ChEMBL-1000 datasets, achieving state-of-the-art or highly competitive results compared to various baselines.

RETRO-R1 makes high-quality decisions. We evaluate the quality of decisions with the pass@1 success rate and success rate at small iteration limits ($N \leq 100$). An optimal policy should always make decisions to convert a molecule to precursors that are simpler or closer to the building blocks. Since the building blocks contain large amounts of molecules, it is possible to achieve a high success rate in one round without regret. We modify the codebase of all baselines to compute this score. The success rate at small iteration limits shows the efficiency of the methods. An optimal policy should find the route quickly without inefficient exploration. As shown in Tab. 1, for the pass@1 success rate, RETRO-R1 outperforms all baselines on both Retro*-190 (55.79%) and ChEMBL-1000 (73.30%) by

[3] https://github.com/binghong-ml/retro_star, https://github.com/junsu-kim97/self_improved_retro

[4] https://github.com/DiXue98/PDVN

Table 1: Results on retro*-190 and ChEMBL-1000. Pass@1 is short for Pass@1 Success Rate. RETRO-R1 is tested in ten random runs, and the means and standard deviations are reported. All baselines are deterministic, meaning they produce the same results on repeated runs; therefore, standard deviations are not reported. Shorter routes refer to the number of molecules whose routes found at the iteration limit of 500 are shorter than the reference routes. This metric is not applicable to ChEMBL-1000 because it lacks reference routes. The best results are marked in bold, and the second-best results are marked with underlines.

		Success Rate (%) at Iteration Limit N						
Method	Pass@1 (%)	$N = 50$	$N = 100$	$N = 200$	$N = 300$	$N = 400$	$N = 500$	Shorter routes
Performance on Retro*-190								
Greedy DFS	19.47	19.47	19.47	19.47	19.47	19.47	19.47	8
retro*	20.53	38.95	50.00	68.95	74.74	77.89	79.47	64
retro*-0	19.47	27.37	37.89	55.79	62.63	72.11	74.21	57
retro*+	30.00	55.79	68.42	79.47	83.16	84.21	84.21	81
retro*+-0	25.26	47.89	61.05	76.84	82.11	85.79	86.32	79
EG-MCTS	46.84	58.42	64.74	69.47	75.26	78.42	81.05	68
PDVN	42.63	64.74	72.11	**82.63**	**87.89**	**91.58**	**92.11**	**90**
RETRO-R1	**55.79**	**73.21±0.80**	**77.21±1.47**	82.58±1.46	84.47±0.82	85.89±0.57	86.95±0.52	87
Performance on ChEMBL-1000								
Greedy DFS	38.10	38.10	38.10	38.10	38.10	38.10	38.10	–
retro*	47.70	65.60	69.00	71.60	73.20	74.00	74.40	–
retro*-0	38.10	64.60	67.20	70.30	71.80	72.70	73.40	–
retro*+	53.50	70.50	73.90	76.20	78.00	78.80	79.40	–
retro*+-0	49.00	69.40	73.60	75.60	76.70	77.60	78.50	–
EG-MCTS	59.70	71.20	74.20	76.90	78.30	79.00	79.30	–
PDVN	60.00	73.90	76.40	78.10	79.60	80.30	80.60	–
RETRO-R1	**73.30**	**77.88±0.35**	**80.21±0.18**	**81.79±0.23**	**82.62±0.29**	**83.28±0.25**	**83.72±0.17**	–

a large margin. The success rate at $N \leq 100$ of RETRO-R1 is also the highest, showcasing the high quality of the decisions made by RETRO-R1 .

RETRO-R1 shows strong abilities of retrosynthetic planning on the in-domain testset. We consider the success rate and shorter routes on Retro*-190. According to Tab. 1, RETRO-R1 achieves at least top-2 performances across all metrics. Although PDVN has a higher Success Rate at large iteration limits, its number of shorter routes is very close to that of RETRO-R1 , indicating RETRO-R1 's strong ability in discovering high-quality synthetic routes. Notably, RETRO-R1 uses only a subset of the training set with 11,366 items in total, while PDVN uses all 299,202 items, which is more prone to overfitting on the dataset.

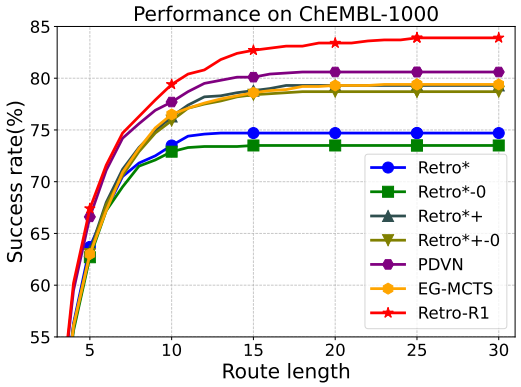

Figure 3: The curves of Success Rate with respect to Route length on ChEMBL-1000. All methods exhaust the 500 iteration limit to find the shortest routes. A point (x, y) on a method's curve indicates that the method achieves a success rate of y when the maximum route length is limited to x.

RETRO-R1 shows strong generalizability on out-of-domain testset.. We report the results on the large, out-of-domain dataset ChEMBL-1000 in Tab. 1. Since no reference routes are provided in this dataset, we report the curves of success rate with respect to route length in Fig. 3. RETRO-R1 consistently outperforms all baselines across all metrics, showing that it learns generalizable chemical knowledge. Notably, RETRO-R1 can find hard routes with length > 20, while other baselines have no improvements when the route length gets very large.

4.4 Ablation Study

We conduct ablation studies on our framework to investigate the contribution of reinforcement learning, context shuffle in planning, and the choice of the single-step model.

Table 2: Ablation Study on Retro*-190. Pass@1 is short for Pass@1 Success Rate. RETRO-SFT refers to the agent trained with supervised fine-tuning on the reference routes of the training set. RETRO-R1 wos refers to planning without context shuffle. All methods are trained with the single-step model with weights V1 and tested with the weights in brackets. RETRO-R1 (V1∼V4) are tested in ten random runs, and the means and standard deviations are reported. The best results are marked in bold. The second-best results are marked with underlines.

		Success Rate (%) at Iteration Limit N					
Method	Pass@1 (%)	$N = 50$	$N = 100$	$N = 200$	$N = 300$	$N = 400$	$N = 500$
RETRO-SFT (V1)	50.00	63.16	67.89	74.21	77.89	78.42	78.42
RETRO-R1 wos (V1)	50.00	61.74	63.16	63.16	63.16	63.16	63.16
RETRO-R1 (V1)	50.00	64.63±1.68	71.32±1.23	78.11±1.68	80.89±1.81	82.95±1.40	84.32±1.22
RETRO-R1 (V2)	52.63	68.53±1.10	74.32±1.79	79.74±1.44	82.16±1.52	84.00±0.95	85.47±1.08
RETRO-R1 (V3)	<u>55.26</u>	<u>68.84±1.43</u>	<u>75.32±1.04</u>	<u>79.95±1.53</u>	<u>83.32±1.29</u>	<u>84.74±0.88</u>	<u>85.84±0.64</u>
RETRO-R1 (V4)	**55.79**	**73.21±0.80**	**77.21±1.47**	**82.58±1.46**	**84.47±0.82**	**85.89±0.57**	**86.95±0.52**

We use all items from the training set to construct an SFT dataset. In this dataset, reference routes are formatted into multi-turn dialogues as described in Sec. 3.2 and Fig. 1, with the text between the <think> and </think> tags set to empty. We use this dataset to fine-tune the 'Qwen2.5-7B-Instruct-1M' model. The resulting model is designated as RETRO-SFT." Other methods presented in Tab. 2 are based on the same model trained using only reinforcement learning. RETRO-R1 wos refers to planning without context shuffle.

As shown in Tab. 2, RETRO-R1 (V1) outperforms RETRO-SFT (V1) in almost all metrics, showcasing the improvements brought by agentic reinforcement learning. Comparing RETRO-R1 wos (V1) and RETRO-R1 (V1), it is obvious that the success rate of RETRO-R1 wos (V1) barely increases with respect to the iteration limit. In fact, it is hard to reach the $N = 500$ iteration limit because of the lack of diversity, and we have to stop the planning process if the synthesis graph has no changes for 5 rounds. The comparison of RETRO-R1 (V1∼V4) shows that any single-step model can serve as a plug-in for our framework without training. A strong single-step model can improve RETRO-R1 's performance with no cost.

5 Related Work

5.1 Retrosynthetic planning

Retrosynthetic planning aims to search for a multi-step reaction pathway for a target molecule. Early computational approaches relied heavily on expert systems encoded with predefined reaction rules and human-designed heuristics to guide the search [3]. More recent methods leverage the power of machine learning to improve the single-step models and search algorithms. 3N-MCTS [1] combines Monte Carlo Tree Search (MCTS) with three networks and use rollout to estimate the score function of a searching state. DFPN-E [47] combines depth-first proof-number search with heuristic edge initialization. Retro* [12] proposes a neural-based A*-like algorithm with a value network predicting the synthetic cost of the molecule. Retro*+ [9] proposes a self-improving procedure to train the single-step model. EG-MCTS [11] uses experience collected in planning to train a value network predicting the cost of the reaction. MEEA* [48] incorporates the exploratory behavior of MCTS into A* by providing a look-ahead search. PDVN [10] constructs two separate value networks to predict the synthesizability and cost of molecules. RetroGraph [38] and DreamRetroer [49] plan a group of target molecules together efficiently. In addition to these traditional methods, researchers have also utilized large language models to solve retrosynthesis tasks. BatGPT-Chem [50] is a large language model tailored for single-step retrosynthesis prediction, trained with extensive instructional data from an expansive chemical database. Llamole [51] integrates a base LLM with a graph encoder to serve as the cost function, based on which the A* search is conducted. [52] proposes a retrosynthetic planning agent based on large language models and knowledge graphs to explore reaction routes with reactions in literature.

5.2 Reinforcement Learning for LLMs

Reinforcement learning has emerged as a powerful paradigm for fine-tuning Large Language Models (LLMs), moving beyond static supervised training to optimize model behavior based on feedback from humans or environments. PPO [30] is one of the most influential algorithms due to its efficiency

and stability. ReMax [53] introduces techniques to reduce the variance of PPO. GRPO [54] uses group sampling strategies to replace value models. REINFORCE++ [55] simplifies GRPO and enhances its training. DAPO [31] further improves the performance with strategies such as Clip-Higher and Overlong Reward Shaping. [32] introduces attention signals as the indicators of the reasoning pattern and proposes to dynamically identify and strengthen key reasoning steps in RL. For tool-integrated reasoning, ReTool [26] modifies PPO to help the policy LLM collaborate with a code sandbox.

6 Conclusion

In this work, we propose a novel framework for multi-step retrosynthetic planning that leverages an RL-trained LLM agent RETRO-R1 , a well-designed workflow, and an iterative planning strategy inspired by human chemists. In the sequential decision process, the agent learns to select molecules and reactions to decompose target molecules into available precursors. Evaluated on retro*-190 and ChEMBL-1000, RETRO-R1 achieves a competitive planning success rate on the in-domain testset and state-of-the-art on the out-of-domain testset, demonstrating a strong generalizability. RETRO-R1 outperforms all baselines on pass@1 success rate and route length, showcasing the ability to make high-quality decisions and find better synthesis routes. These results underscore the potential of leveraging LLMs and RL within an agentic framework for complex scientific problems.

Limitations and Future Work. Despite the promising results, our RETRO-R1 has several limitations. Firstly, the LLM cannot propose a reaction by itself and the performance inherently relies on the single-step retrosynthesis model. Secondly, although we only focus on the reactions with high probabilities, we still cannot guarantee that these reactions are feasible in practice, and predicting reaction conditions is very difficult. Future work will focus on designing a strong agent or multi-agent system that can propose reactions and analyze them using computational chemistry tools. The reward function should integrate more explicit chemical knowledge, which is essential for developing more powerful and versatile AI chemists.

Acknowledgments and Disclosure of Funding

This work is supported by the National Key R&D Program of China (2022ZD0160501), the National Natural Science Foundation of China (Grant No. 62376133), and the Beijing Nova Program (20240484682).

References

[1] Marwin HS Segler, Mike Preuss, and Mark P Waller. Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698):604–610, 2018.

[2] James Law, Zsolt Zsoldos, Aniko Simon, Darryl Reid, Yang Liu, Sing Yoong Khew, A Peter Johnson, Sarah Major, Robert A Wade, and Howard Y Ando. Route designer: a retrosynthetic analysis tool utilizing automated retrosynthetic rule generation. *Journal of chemical information and modeling*, 49(3):593–602, 2009.

[3] Tomasz Klucznik, Barbara Mikulak-Klucznik, Michael P McCormack, Heather Lima, Sara Szymkuć, Manishabrata Bhowmick, Karol Molga, Yubai Zhou, Lindsey Rickershauser, Ewa P Gajewska, et al. Efficient syntheses of diverse, medicinally relevant targets planned by computer and executed in the laboratory. *Chem*, 4(3):522–532, 2018.

[4] Marwin HS Segler and Mark P Waller. Neural-symbolic machine learning for retrosynthesis and reaction prediction. *Chemistry–A European Journal*, 23(25):5966–5971, 2017.

[5] Kaipeng Zeng, Bo Yang, Xin Zhao, Yu Zhang, Fan Nie, Xiaokang Yang, Yaohui Jin, and Yanyan Xu. Ualign: pushing the limit of template-free retrosynthesis prediction with unsupervised smiles alignment. *Journal of Cheminformatics*, 16(1):80, 2024.

[6] Jingxin Dong, Mingyi Zhao, Yuansheng Liu, Yansen Su, and Xiangxiang Zeng. Deep learning in retrosynthesis planning: datasets, models and tools. *Briefings in Bioinformatics*, 23(1):bbab391, 2022.

[7] Zipeng Zhong, Jie Song, Zunlei Feng, Tiantao Liu, Lingxiang Jia, Shaolun Yao, Tingjun Hou, and Mingli Song. Recent advances in deep learning for retrosynthesis. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 14(1):e1694, 2024.

[8] Guillaume Gricourt, Philippe Meyer, Thomas Duigou, and Jean-Loup Faulon. Artificial intelligence methods and models for retro-biosynthesis: A scoping review. *ACS Synthetic Biology*, 13(8):2276–2294, 2024.

[9] Junsu Kim, Sungsoo Ahn, Hankook Lee, and Jinwoo Shin. Self-improved retrosynthetic planning. In *International Conference on Machine Learning*, pages 5486–5495. PMLR, 2021.

[10] Guoqing Liu, Di Xue, Shufang Xie, Yingce Xia, Austin Tripp, Krzysztof Maziarz, Marwin Segler, Tao Qin, Zongzhang Zhang, and Tie-Yan Liu. Retrosynthetic planning with dual value networks. In *International Conference on Machine Learning*, pages 22266–22276. PMLR, 2023.

[11] Siqi Hong, Hankz Hankui Zhuo, Kebing Jin, Guang Shao, and Zhanwen Zhou. Retrosynthetic planning with experience-guided monte carlo tree search. *Communications Chemistry*, 6(1):120, 2023.

[12] Binghong Chen, Chengtao Li, Hanjun Dai, and Le Song. Retro*: learning retrosynthetic planning with neural guided a* search. In *International conference on machine learning*, pages 1608–1616. PMLR, 2020.

[13] Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, et al. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*, 2025.

[14] Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, et al. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*, 2025.

[15] Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. *arXiv preprint arXiv:2404.15846*, 2024.

[16] Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535, 2024.

[17] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[18] Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Mohamed Amin, Le Hou, Kevin Clark, Stephen R Pfohl, Heather Cole-Lewis, et al. Toward expert-level medical question answering with large language models. *Nature Medicine*, pages 1–8, 2025.

[19] Jenish Maharjan, Anurag Garikipati, Navan Preet Singh, Leo Cyrus, Mayank Sharma, Madalina Ciobanu, Gina Barnes, Rahul Thapa, Qingqing Mao, and Ritankar Das. Openmedlm: prompt engineering can out-perform fine-tuning in medical question-answering with open-source large language models. *Scientific Reports*, 14(1):14156, 2024.

[20] Antoine Louis, Gijs van Dijck, and Gerasimos Spanakis. Interpretable long-form legal question answering with retrieval-augmented large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 22266–22275, 2024.

[21] Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, et al. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*, 2024.

[22] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024.

[23] Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, et al. Internlm-math: Open math large language models toward verifiable reasoning. *arXiv preprint arXiv:2402.06332*, 2024.

[24] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024.

[25] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

[26] Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.

[27] Chen Gao, Xiaochong Lan, Nian Li, Yuan Yuan, Jingtao Ding, Zhilun Zhou, Fengli Xu, and Yong Li. Large language models empowered agent-based modeling and simulation: A survey and perspectives. *Humanities and Social Sciences Communications*, 11(1):1–24, 2024.

[28] Onder Gurcan. Llm-augmented agent-based modelling for social simulations: Challenges and opportunities. *arXiv preprint arXiv:2405.06700*, 2024.

[29] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[31] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[32] Yang Li, Zhichen Dong, Yuhan Sun, Weixun Wang, Shaopan Xiong, Yijia Luo, Jiashun Liu, Han Lu, Jiamang Wang, Wenbo Su, et al. Attention illuminates llm reasoning: The preplan-and-anchor rhythm enables fine-grained policy optimization. *arXiv preprint arXiv:2510.13554*, 2025.

[33] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[34] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[35] Jie Ouyang, Ruiran Yan, Yucong Luo, Mingyue Cheng, Qi Liu, Zirui Liu, Shuo Yu, and Daoyu Wang. Training powerful llm agents with end-to-end reinforcement learning, 2025.

[36] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[37] Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

[38] Shufang Xie, Rui Yan, Peng Han, Yingce Xia, Lijun Wu, Chenjuan Guo, Bin Yang, and Tao Qin. Retrograph: Retrosynthetic planning with graph search. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2120–2129, 2022.

[39] Daniel Mark Lowe. *Extraction of chemical structures and reactions from the literature*. PhD thesis, 2012.

[40] Connor W Coley, William H Green, and Klavs F Jensen. Rdchiral: An rdkit wrapper for handling stereochemistry in retrosynthetic template extraction and application. *Journal of chemical information and modeling*, 59(6):2529–2537, 2019.

[41] Samuel Genheden and Esben Bjerrum. Paroutes: towards a framework for benchmarking retrosynthesis route predictions. *Digital Discovery*, 1(4):527–539, 2022.

[42] An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. Qwen2.5-1m technical report. *arXiv preprint arXiv:2501.15383*, 2025.

[43] Qwen Team. Qwen2.5-1m: Deploy your own qwen with context length up to 1m tokens, January 2025.

[44] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[45] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[47] Akihiro Kishimoto, Beat Buesser, Bei Chen, and Adi Botea. Depth-first proof-number search with heuristic edge cost and application to chemical synthesis planning. *Advances in Neural Information Processing Systems*, 32, 2019.

[48] Dengwei Zhao, Shikui Tu, and Lei Xu. Efficient retrosynthetic planning with mcts exploration enhanced a* search. *Communications Chemistry*, 7(1):52, 2024.

[49] Xuefeng Zhang, Haowei Lin, Muhan Zhang, Yuan Zhou, and Jianzhu Ma. A data-driven group retrosynthesis planning model inspired by neurosymbolic programming. *Nature Communications*, 16(1):192, 2025.

[50] Yifei Yang, Runhan Shi, Zuchao Li, Shu Jiang, Bao-Liang Lu, Yang Yang, and Hai Zhao. Batgpt-chem: A foundation large model for retrosynthesis prediction. *arXiv preprint arXiv:2408.10285*, 2024.

[51] Gang Liu, Michael Sun, Wojciech Matusik, Meng Jiang, and Jie Chen. Multimodal large language models for inverse molecular design with retrosynthetic planning. *arXiv preprint arXiv:2410.04223*, 2024.

[52] Qinyu Ma, Yuhao Zhou, and Jianfeng Li. Automated retrosynthesis planning of macromolecules using large language models and knowledge graphs. *Macromolecular Rapid Communications*, page 2500065, 2025.

[53] Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023.

[54] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[55] Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.

[56] Yu Dong, Xiaodi Qiu, Neil Shaw, Yueyang Xu, Yuna Sun, Xuemei Li, Jun Li, and Zihe Rao. Molecular basis for the inhibition of β-hydroxyacyl-acp dehydratase hadab complex from mycobacterium tuberculosis by flavonoid inhibitors. *Protein & cell*, 6(7):504–517, 2015.

[57] Yanru Qu, Keyue Qiu, Yuxuan Song, Jingjing Gong, Jiawei Han, Mingyue Zheng, Hao Zhou, and Wei-Ying Ma. Molcraft: Structure-based drug design in continuous parameter space. *arXiv preprint arXiv:2404.12141*, 2024.

[58] Sunghwan Kim, Paul A Thiessen, Evan E Bolton, Jie Chen, Gang Fu, Asta Gindulyte, Lianyi Han, Jane He, Siqian He, Benjamin A Shoemaker, et al. Pubchem substance and compound databases. *Nucleic acids research*, 44(D1):D1202–D1213, 2016.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in abstract and Sec. 1 accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Sec. 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the framework in detail in Sec. 3 and the implementation settings in Sec. 4.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All the code are available with the link in Sec. 4.2. We do not use new dataset.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All details are specified in Sec. 4.1 and Sec. 4.2

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In Tab. 1 and Tab. 2, we report the standard deviation of our methods.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All details about compute resources are specified in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper has no problem of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited all the original paper for the codebases and datasets we used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We have written the documentation alongside the codebase in Github URL.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We describe the use of LLMs in Sec. 3 and Sec. 4.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Two-branch policy network

In Sec. 4.1, we describe the two-branch policy network structure from the previous work PDVN to ensure all methods have the same search space and the probabilities of proposed reactions are high according to weights V1 trained on real reactions. The structure is shown in Fig. 4. The reference single-step model, which is weights V1 in this work, proposes top-k candidate reactions for the given molecule mol1. The augmented single-step models, which are weights V2, V3, and V4 in this work, recompute probabilities for these reactions and rerank them. In this way, all single-step models provide the same candidates. The only difference is the probabilities assigned to them. This helps the retrosynthesis planner focus on reactions that are possible to happen in practice.

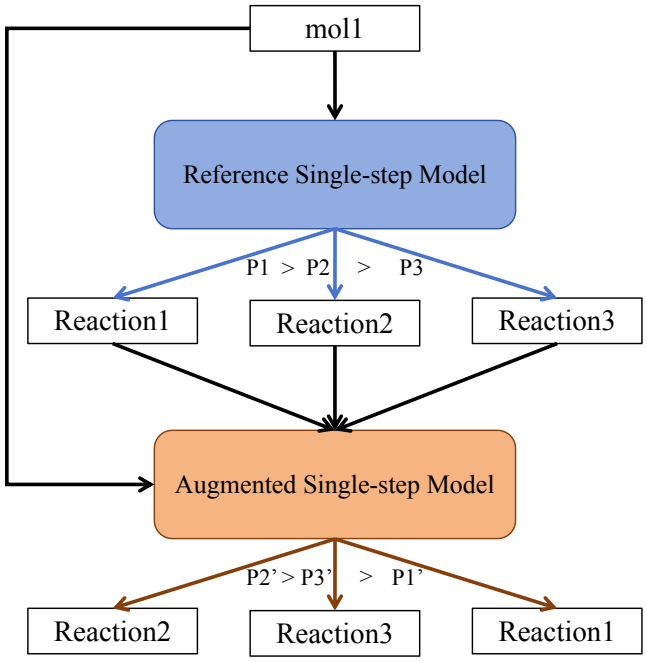

Figure 4: The illustration of the two-branch policy network structure from the previous work PDVN. The reference single-step model, which is weights V1 in this work, proposes top-k candidate reactions for the given molecule mol1. The augmented single-step models, which are weights V2, V3, and V4 in this work, recompute probabilities for these reactions and rerank them accordingly.

B Prompt and examples of interaction

In this section, we give the prompts and some examples of interaction between the agent and the environment. In Fig. 5, we give the system prompt and user prompt to synthesize molecule C[C@H](c1ccccc1)N1C[C@]2(C(=O)OC(C)(C)C)C=CC[C@@H]2C1=S. In Fig. 6, we give two cases where the agent wrongly takes actions and the corresponding responses of the environment. In Fig. 7, we give two cases where the agent correctly takes actions.

C Curves in training

Fig. 8 plots six key metrics recorded during training: (a) **Pass@1 Success Rate:** This measures the percentage of molecules in a training batch solved in a single round planning. As the training progresses, this metric steadily increases to 95%, which means our reinforcement learning framework makes full use of the data efficiently. (b) **Average Reward:** This is the average rule-based reward (defined in Sec. 3.2) per training batch. As expected, this curve is highly correlated with the success

```
# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>
{"name": "CALL", "description": "Perform single step retrosynthesis for a
    molecule and return several possible reactions to synthesize it. Note
    that the reactions may be incorrect and each reactant is marked as '
    available' or 'unavailable'. The unavailable molecules have to be
    synthesized further.", "parameters": {"type": "object", "properties": {"
    molecule": {"type": "string", "description": "The ID for the molecule to
    be synthesized. For example, 0-0 ."}}, "required": ["molecule"]}}
{"name": "SELECT", "description": "Given several reactions to synthesize a
    molecule, use this tool to select one from them.", "parameters": {"type":
     "object", "properties": {"reaction": {"type": "string", "description": "
    The ID for the selected reaction. For example, 0-0-0 ."}}, "required": ["
    reaction"]}}
</tools>

For each function call, return a json object with function name and arguments
     within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

```
You are a professional organic chemist, skilled at finding pathways to
    synthesize novel molecules. Given a target molecule, your task is to find
     a pathway where the target can be eventually synthesized using
    ingredient molecules, and each step of the pathway is a viable chemical
    reaction. The maximum number of reaction steps you can use is 30.
You can use the tools provided to you to find the pathway. You can use the
    tool as many times as you want.
You must first conduct reasoning inside <think>...</think>. Then use the tool
     call <tool_call>...</tool_call> to call the tool.

Output format for tool call:
<think>
...
</think>
<tool_call>
...
</tool_call>

The target molecule is C[C@H](c1ccccc1)N1C[C@]2(C(=O)OC(C)(C)C)C=CC[C@@H]2C1=
    S. Now start to search for the pathway.
Molecule state 0: Unsolved molecules:
Molecule 0-0: C[C@H](c1ccccc1)N1C[C@]2(C(=O)OC(C)(C)C)C=CC[C@@H]2C1=S
Now, 0 steps are used.
Please select the molecule with the highest synthesis difficulty from the
    unsolved molecules. Use the tool 'CALL' to synthesize it.
```

Figure 5: The system prompt (upper figure) and user prompt (bottom figure) to synthesize molecule C[C@H](c1ccccc1)N1C[C@]2(C(=O)OC(C)(C)C)C=CC[C@@H]2C1=S.

```
ENV: <tool_response>
    Select reaction 9-1-1 to synthesize molecule N#C[C@@]1(CO)C=CC[C@@H]1C(=O)
        O . The state goes to:
    Molecule state 8: Unsolved molecules:
    Molecule 8-1: CCOC(=O)[C@H]1CC=C[C@@]1(C#N)CO
    To reach this state, at least 7 reaction steps are used.
    </tool_response>
Agent: <think>
    ...
    </think>
    <tool_call>
    {"name": "CALL", "arguments": {"molecule": "7-1"}}
    </tool_call>
ENV: <tool_response>
    Invalid arguments for tool 'CALL': Parameter molecule has invalid value,
        should be 8-1 ~ 8-1 .
    </tool_response>
```

```
ENV: <tool_response>
    Reaction state 9-1: Possible reactions to synthesize molecule 9-1: CC[C@H
        ](c1ccccc1)N1C[C@]2(C=O)C=CC[C@@H]2C1=O are as follows:
    Reaction 9-1-1: CC[C@H](c1ccccc1)N1C[C@]2(CO)C=CC[C@@H]2C1=O (unavailable)
    Reaction 9-1-2: CCOC(=O)[C@@]12C=CC[C@@H]1C(=O)N([C@H](CC)c1ccccc1)C2 (
        unavailable)
    ...
    </tool_response>
Agent: <think>
    ...
    </think>
    <tool_call>{"name": "SELECT", "arguments": {"reaction": "9-1-6"}}
ENV: <tool_response>
    Invalid tool call format. Please use the format:
    <tool_call>
    {"name": "tool_name", "arguments": {params_json}}
    </tool_call> .
    </tool_response>
```

Figure 6: Two examples of failed interaction. The error messages from the environment are shown. In the upper figure, the agent takes the action CALL with an illegal molecule identifier. In the bottom figure, the agent misses the </tool_call> tag and a newline character after <tool_call>.

rate, as the first term (success component) of the reward function is dominant. (c) **Penalty Score:** This is the second term (efficiency component) of the reward, which penalizes the agent for failing to complete tasks within the interaction limit. The penalty is high in the early stages when the agent frequently fails or takes incorrect actions, but it decreases as the agent improves, eventually applying only to hard data items. (d) **Average Critic Value:** This shows the average value predicted by the critic network across all tokens generated by the actor network in a training batch. The value fluctuates significantly in the early stages but stabilizes as training progresses. This stabilization is crucial, as an unstable critic can misguide the actor. As noted in Eq. 2, we set the prediction target for environment tokens to zero to aid this stability. (e) **Actor Entropy:** This measures the uncertainty (entropy) of the actor network when generating the next tokens. The observed decrease in entropy indicates that the actor's policy is converging from exploration to a more deterministic, optimal strategy. (f) **Average Interaction Turns:** This plots the average number of turns used to solve the tasks. The decreasing trend demonstrates that the agent is learning to find solutions more efficiently as training progresses.

```
ENV: <tool_response>
    Select reaction 6-1-4 to synthesize molecule N#C[C@H]1CC=C[C@@]1(C#N)CCl .
         The state goes to:
    Molecule state 7: Unsolved molecules:
    Molecule 7-1: N#C[C@H]1CC=C[C@@]1(C#N)CO
    Molecule 7-2: O=S(Cl)Cl
    To reach this state, at least 7 reaction steps are used.
    </tool_response>
Agent: <think>
    The molecules with the highest synthesis difficulty are 7-1 and 7-2. We
        should use the tool 'CALL' to synthesize molecule 7-1, as it is likely
         the more challenging one.
    </think>
    <tool_call>
    {"name": "CALL", "arguments": {"molecule": "7-1"}}
    </tool_call>
```

```
ENV: <tool_response>
    Reaction state 1-1: Possible reactions to synthesize molecule 1-1: C[C@H](
        c1ccccc1)N1C[C@]2(C(=O)OC(C)(C)C)C=CC[C@@H]2C1=O are as follows:
    Reaction 1-1-1: C=CC[C@@H]1C(=O)N([C@H](C)c2ccccc2)C[C@@]1(C=C)C(=O)OC(C)(
        C)C (unavailable)
    Reaction 1-1-2: C[C@H](c1ccccc1)N1C(=O)[C@H]2CC=C[C@@]2(C(=O)OC(C)(C)C)C1=
        O (unavailable)
    Reaction 1-1-3: C[C@@H](NC[C@]1(C(=O)OC(C)(C)C)C=CC[C@@H]1C(=O)Cl)c1ccccc1
         (unavailable)
    Reaction 1-1-4: C[C@H](c1ccccc1)N1C(=O)[C@H]2CC=C[C@@]2(C(=O)OC(C)(C)C)C1=
        S (unavailable)
    Reaction 1-1-5: CN(C)C(OC(C)(C)C)OC(C)(C)C (available) + C[C@H](c1ccccc1)
        N1C[C@]2(C(=O)O)C=CC[C@@H]2C1=O (unavailable)
    Reaction 1-1-6: CC(C)(C)O (available) + C[C@H](c1ccccc1)N1C[C@]2(C(=O)O)C=
        CC[C@@H]2C1=O (unavailable)
    ...
    </tool_response>
Agent: <think>
    Among the reactions, the possible ones are 1-1-5 and 1-1-6, where the
        reactants are either available or easy to synthesize. We should use
        the tool 'SELECT' to choose the best reaction.
    </think>
    <tool_call>
    {"name": "SELECT", "arguments": {"reaction": "1-1-5"}}
    </tool_call>
```

Figure 7: Two examples of the successful actions taken by the agent. In the upper figure, the agent selects a more challenging molecule to synthesize first. In the bottom figure, the agent selects a reaction whose reactants are available or easy to synthesize.

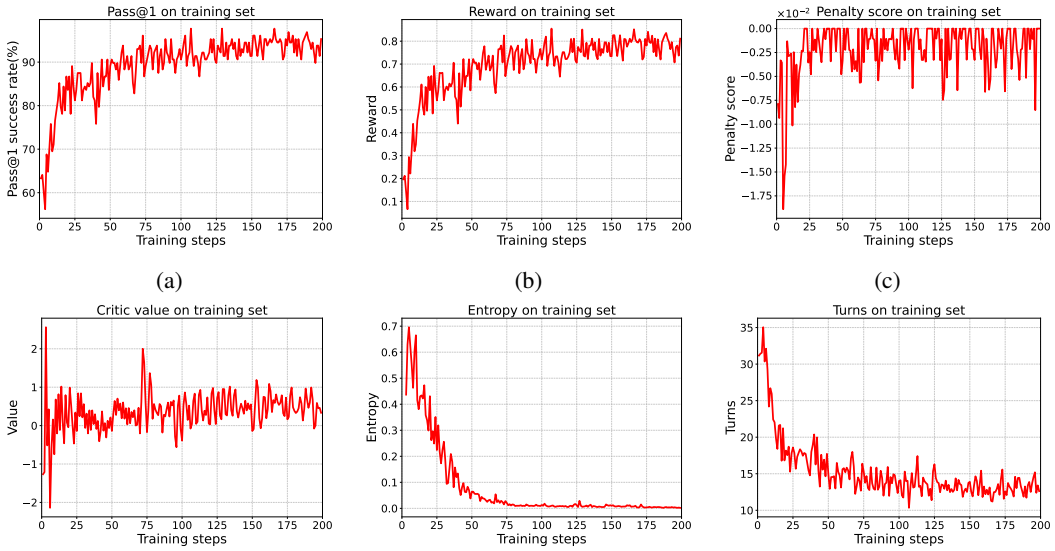

Figure 8: The curves of metrics with respect to the training steps. (a) The pass@1 success rate on the training batch. (b) The average reward described in Sec. 3.2 on the training batch. (c) The average penalty score on the training batch. (d) The average value of all generated tokens predicted by the critic network on the training batch. (e) The entropy of the actor network on the training batch. (f) The average interaction turns between the agent and the environment in the training batch.

D Discussion on computational cost

We use 32×A800 80G GPU to train RETRO-R1 . The model converges after 190 training steps, requiring approximately 70 hours. When planning with 500 iteration limit, it takes around 20 minutes per molecule on a single A800 80G GPU. While the LLM-based approach exhibits higher computational demands compared to traditional methods, this expenditure is justified given the inherent complexity and critical importance of retrosynthesis planning. Traditional methods, often constrained by smaller models, are incapable of effectively leveraging the extensive corpus of chemical literature. Their reliance on limited, task-specific datasets precludes the acquisition of a comprehensive understanding of foundational chemical principles. Consequently, progress in this domain has stagnated; while existing methods may report gains in specific metrics, they fundamentally lack generalization capacity. This work demonstrates that Large Language Models, optimized via reinforcement learning fine-tuning on small amounts of data, can rapidly surpass traditional approaches, emerging as domain-specific experts with robust generalization capabilities. Furthermore, the inherent capacity of LLMs to articulate their reasoning process in natural language (i.e., "chain of thought") significantly enhances the interpretability, rendering the outputs readily analyzable by chemists. We believe this work represents a significant step forward. In the future, LLM-based agents are poised to solve an increasing number of complex chemical problems, driving comprehensive advancements across the entire field of chemistry.

E More experiment results

E.1 Performance comparison across datasets

To further compare the generalizability of methods, we visualize the success rate (at an iteration limit of 500) across two testing sets in Fig. 9: the in-domain Retro*-190 and the out-of-domain ChEMBL-1000. The red area highlights the performance gap. The baseline Retro*-0 only uses the single-step model with V1 weights, which is trained on the real reaction dataset, and its planning algorithm is training-free. Consequently, it exhibits comparable performance on both testing sets. Retro*+-0 and PDVN improve Retro*-0 by only augmenting the single-step model (to weights V3 and V4, respectively) on the training set. However, they do not evaluate the cost of molecules or

reactions to alleviate the bias in the single-step model, and thus are significantly overfitted. The performance of Retro*, Retro*+, and EG-MCTS is much lower than RETRO-R1 , suggesting they acquire limited chemical knowledge. RETRO-R1 , however, achieves both high overall performance and a small generalization gap, indicating that it learns transferable chemical knowledge.

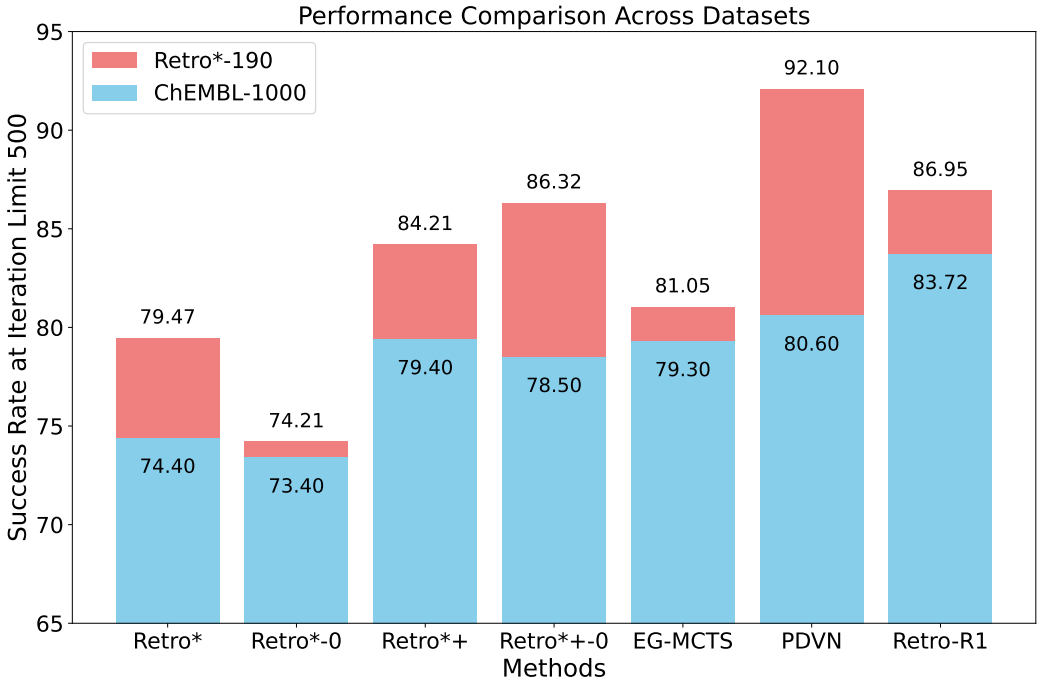

Figure 9: The performance comparison of all methods on Retro*-190 and ChEMBL-1000. Retro*-190 is an in-domain dataset while ChEMBL-1000 is an out-of-domain dataset. RETRO-R1 achieves both high overall performance and a small generalization gap on two testing sets.

E.2 Route diversity

In practice, chemists select the most suitable synthetic route from multiple candidates proposed by a retrosynthetic planning method. Route diversity is therefore a crucial metric for evaluating method performance. To assess the route diversity of RETRO-R1 , we introduce two metrics: the Number of Routes and the Number of Initial Molecules. Two routes are considered distinct if they differ by a single molecule or reaction. The second metric quantifies the size of the unique set of initial molecules aggregated from all feasible routes. This metric measures the algorithm's ability to explore a broad chemical space, rather than being confined to a limited set of precursors. A wider selection of starting materials is highly beneficial for chemists. Using the iterative planning strategy (Sec. 3.4), RETRO-R1 constructs a large-scale synthesis graph for each target molecule at the 500-iteration limit. We extracted all feasible routes from these graphs for the retro*-190 dataset and calculated both metrics. Tab. 3 and Tab. 4 present the distributions of these metrics. The results show that RETRO-R1 finds over 100 distinct routes for most target molecules, involving more than 100 unique initial molecules, thus demonstrating the high route diversity of our approach.

Table 3: Distribution of the Number of Routes on Retro*-190. 0 means no routes are found. The distribution refers to the number of target molecules in the dataset.

Number of Routes	**0**	**1∼10**	**11∼100**	**101∼1000**	**1001∼10000**	**10001∼100000**	**>100000**
Distribution	26	11	26	38	54	23	12

Table 4: Distribution of the Number of Initial Molecules on Retro*-190. 0 means no routes are found. The distribution refers to the number of target molecules in the dataset.

Number of Initial Molecules	**0**	**1∼10**	**11∼100**	**101∼1000**	**1001∼10000**	**>10000**
Distribution	26	14	25	49	50	26

E.3 Case study

The main purpose of this paper is to accelerate the process of drug discovery. To test whether our model can find the synthesis route for a novel molecule, we select a protein target and generate a novel molecule that can bind to it. Specifically, the selected protein is related to the tuberculosis (TB): (3R)-hydroxyacyl-ACP dehydratase HadAB hetero-dimer from Mycobacterium tuberculosis (PDB ID: 4RLU). Dehydration is one of the key steps in the biosynthesis of mycolic acids and is vital to the growth of Mycobacterium tuberculosis (Mtb). Consequently, stalling dehydration cures tuberculosis (TB) [56]. The molecular generation method we use is the state-of-the-art structure-based drug design (SBDD) model MolCRAFT [57]. From the generated molecules, we select one that is not recorded in the PubChem database [58] and has a strong docking affinity with the target protein. The SMILES is O=C1C=CCc2nc3cnc4cc(-c5cccnc5)ccc4c3nc21 and the vina score is -10.829 kcal/mol. The docking pose is shown in Fig. 10.

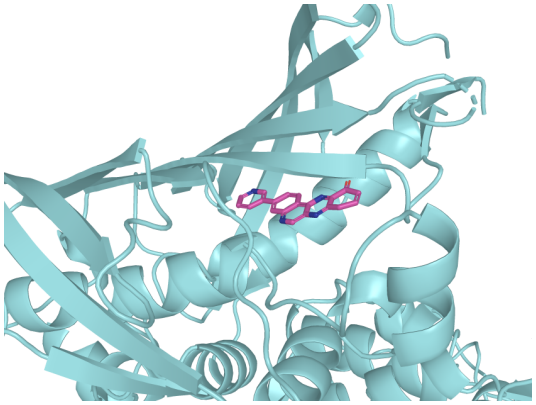

Figure 10: The docking pose of (3R)-hydroxyacyl-ACP dehydratase HadAB hetero-dimer from Mycobacterium tuberculosis (PDB ID: 4RLU) and the generated molecule O=C1C=CCc2nc3cnc4cc(-c5cccnc5)ccc4c3nc21.

The synthesis route found by RETRO-R1 is shown in Fig. 11. The reaction ① is a Suzuki Coupling Reaction. The reaction ② is an Oxidative Dehydrogenation Reaction. The reaction ③ is a DDQ-mediated Dehydrogenative Oxidation Reaction. The reaction ④ is a Quinoxaline Synthesis. All the reactions are very likely to happen in practice. The building blocks are marked with green rectangles.

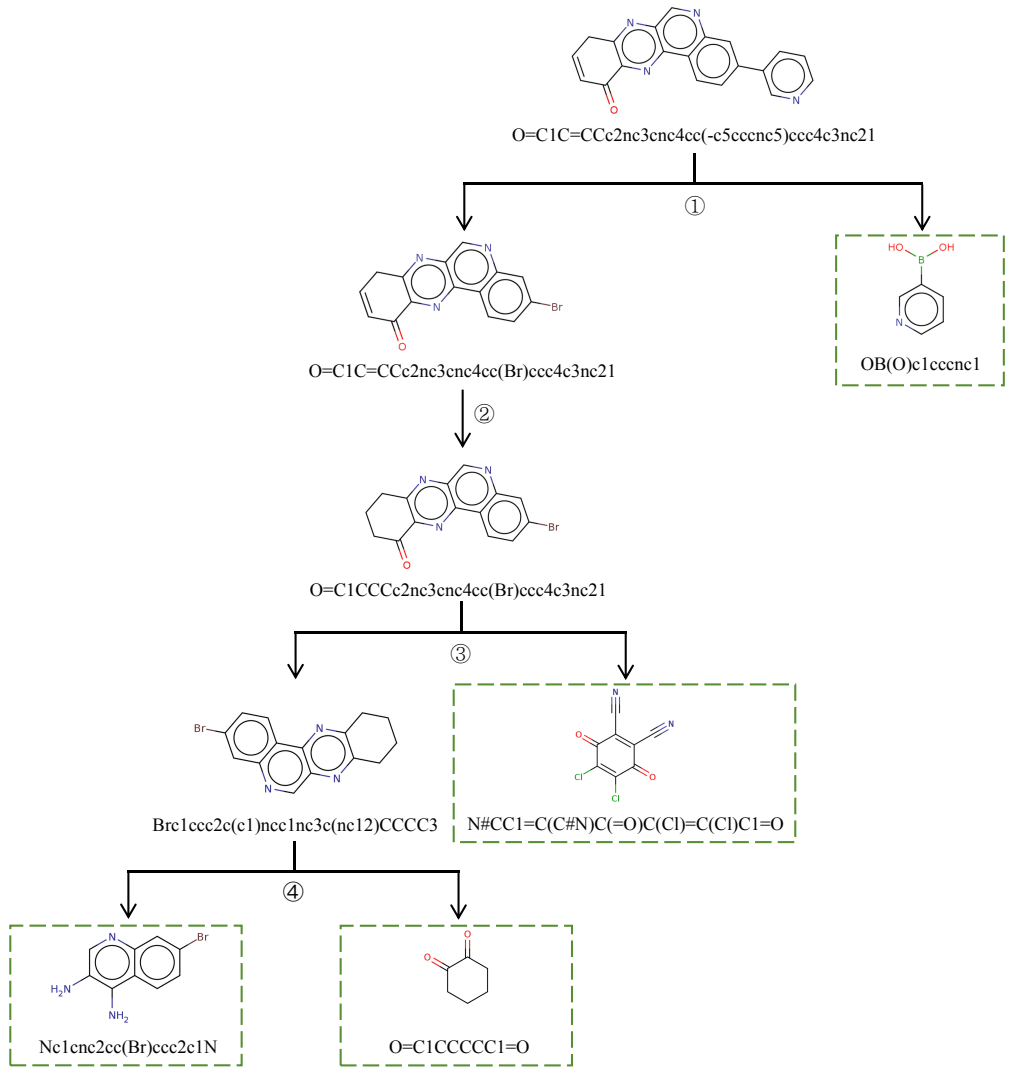

Figure 11: The synthesis route of molecule O=C1C=CCc2nc3cnc4cc(-c5cccnc5)ccc4c3nc21 found by RETRO-R1 . The building blocks are marked with green rectangles.