PROVENANCE-TRACE POLICY OPTIMIZATION FOR AGENTIC RETROSYNTHESIS PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic reinforcement learning (RL) offers a promising approach to retrosynthetic planning. Existing methods train LLM agents to select reactions and explore synthesis routes through tool-mediated interaction, using outcome-based rewards to encourage successful planning. However, trajectory-level rewards do not explicitly distinguish how individual search decisions contribute to planning. Equally rewarded trajectories may resolve different precursor subgoals, while candidate discovery can support reaction selections in later turns. To address this challenge, we propose Provenance-Trace Policy Optimization (PTPO), which converts structured search evidence into local supervision for search decisions. Specifically, PTPO aggregates terminal precursor assessments using the AND–OR synthesis structure and computes local advantages by comparing decisions made for the same molecule under identical ordered candidate views. Through provenance tracing, PTPO further propagates these advantages to the earlier turns that revealed the selected candidates. These local advantages complement trajectory-level supervision through separate turn- and trajectory-level PPO objectives, with each advantage channel normalized independently. Credit estimation reuses the sampled trajectories without a learned critic or additional rollouts. Experiments show that a 4B agent trained with PTPO achieves robust planning performance. On both in-distribution and out-of-distribution retrosynthesis benchmarks, PTPO consistently outperforms outcome-only reward RL agents across all evaluated search budgets.

1 INTRODUCTION

Constructing viable synthesis routes is a crucial and challenging task in organic chemistry (Coley et al., 2019b; Klucznik et al., 2018), requiring reasoning over extended search horizons. In retrosynthetic planning, selected reactions need to yield promising decompositions, and all required precursors must connect to available starting materials (Corey & Wipke, 1969; Chen et al., 2020). Such planning plays a central role in translating molecular designs into realizable compounds, supporting applications across synthetic chemistry and drug discovery (Blakemore et al., 2018). However, the proliferation of reaction alternatives and precursor branches leads to a combinatorial expansion of the search space as routes deepen, making exhaustive exploration computationally prohibitive (Segler et al., 2018). Identifying complete routes within practical search budgets therefore poses a fundamental decision-making challenge.

Learning-based approaches to retrosynthesis typically employ single-step models to propose candidate precursors for a given target molecule. One line of work selects and applies reaction templates that encode local structural transformations, using neural predictors to rank applicable rules (Segler & Waller, 2017), with graph neural networks further capturing molecular context for template selection (Dai et al., 2019; Chen & Jung, 2021). A complementary line directly generates reactant representations from the target structure without explicit template matching, commonly through sequence-to-sequence architectures (Liu et al., 2017; Schwaller et al., 2020). To construct complete synthesis routes, these predictors are integrated with search procedures such as Monte Carlo tree search (Segler et al., 2018) or A*-inspired search (Chen et al., 2020), which recursively expand precursor candidates until all terminal molecules in a synthesis route belong to a predefined inventory of available starting materials. Subsequent work improves single-step predictors using successful planning trajectories (Kim et al., 2021) and route-level reinforcement learning (Liu et al., 2023).

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

Recent work leverages LLMs to navigate multi-step retrosynthesis through iterative generation and retrieval-augmented search, including decision programs that generate, evaluate, and refine complete synthesis routes (Wang et al., 2025a). However, these inference-time procedures rely on predefined search operators and evaluation criteria, without directly optimizing the LLM policy from route-completion outcomes. As an alternative, RL-based agents such as Retro-R1 (Liu et al., 2025) and RetroAgent (Zhu et al., 2026) learn long-horizon retrosynthesis policies from sampled environment interactions, using outcome-based rewards to guide policy optimization. While this enables end-to-end adaptation and allows search behavior to evolve from experience, a critical challenge remains: terminal rewards do not explicitly disentangle the quality of individual reaction choices from candidate availability and downstream precursor exploration, an entanglement that becomes more pronounced as the search branches and trajectories grow longer (Appendix F.7).

Prior work on long-horizon RL highlights the difficulty of attributing delayed outcomes to earlier decisions (Wang et al., 2025b; Qian et al., 2025; Feng et al., 2025b). In retrosynthesis, AND–OR dependencies further couple a reaction's outcome to the resolution of all required precursor branches (Chen et al., 2020; Liu et al., 2023). Thus, equally rewarded trajectories may contain decompositions with different precursor outcomes. Repeated choices for the same molecule can also occur under different candidate observations and interaction histories, making decision quality difficult to separate from candidate availability. Moreover, a prediction request may reveal a productive reaction selected only after another branch is explored. Temporal proximity therefore need not identify the earlier turn that supplied the selected candidate, further complicating credit assignment.

To address this challenge and limitation, we aim to extract comparable reaction outcomes from sampled searches and attribute them to both reaction selection and enabling information acquisition. Given the synthesis dependencies and candidate provenance available in the planning environment, we propose Provenance-Trace Policy Optimization (PTPO), a critic-free RL framework for multi-turn retrosynthetic planning. To learn from intermediate search evidence, our method introduces two components: (i) **matched-context reaction evaluation** to aggregate terminal precursor assessments following the AND–OR search structure and compare reaction selections under identical ordered candidate views, and (ii) **provenance-based credit tracing** to propagate the resulting advantages to earlier turns revealing the selected candidates. These signals are derived from observed search evidence, enabling local comparisons without a learned critic or additional trajectory sampling. Together, these components complement trajectory-level rewards with local supervision through separate sequence- and turn-level PPO objectives with per-channel normalization. Our results demonstrate that search itself provides effective supervision for agentic reinforcement learning in retrosynthesis. Incorporating precursor outcomes and candidate provenance into credit assignment, PTPO improves planning success beyond trajectory-level rewards alone.

Related Work

Multi-Step Retrosynthesis. Multi-step retrosynthetic planning has evolved from reaction prediction with explicit search to learning policies and search guidance from planning experience. Combining single-step prediction with tree search, neural-guided planners prioritize exploration using learned policies or cost estimates Segler et al. (2018); Chen et al. (2020). Graph-based planning improves through intermediate sharing and goal-constrained bidirectional search Xie et al. (2022); Yu et al. (2024). Using planning experience, Retro*+ Kim et al. (2021) trains its reaction predictor to imitate successful routes. EG-MCTS Hong et al. (2023) learns a guidance network from accumulated search experience. PDVN Liu et al. (2023) learns reaction policies and separate value functions for synthesizability and synthesis cost using a tree-shaped MDP. However, these components remain within prescribed search procedures governing how predictions and value estimates guide planning.

LLM Agents and Reinforcement Learning for Retrosynthesis. Building on policy optimization Schulman et al. (2017), reinforcement learning has become a primary means of turning LLMs into capable agents. One line of work strengthens reasoning through improved policy optimization and reward design Shao et al. (2024); Zheng et al. (2025); Tajwar et al. (2026); Zhao et al. (2025b;a), while another incorporates external tools and environment feedback into policy learning, supporting code execution Feng et al. (2025a), information retrieval Jin et al. (2025); Song et al. (2025), and multi-turn decision-making Wang et al. (2025b), with dedicated reward design for tool selection and application Qian et al. (2025). In retrosynthesis, LLMs have similarly emerged as

flexible planners that combine chemical reasoning with tool interaction. Chemical language models such as BatGPT-Chem Yang et al. (2025) support single-step reaction prediction, and LLM-SynPlanner Wang et al. (2025a) generates and refines complete routes through retrieval-augmented evolutionary search. Bringing agentic RL to multi-step planning, Retro-R1 Liu et al. (2025) uses PPO to optimize molecule and reaction selection while accumulating synthesis information across successive planning attempts, and RetroAgent Zhu et al. (2026) uses GSPO Zheng et al. (2025) to learn tool-mediated search, providing explicit tools for inspecting and modifying structured search memory. These memory mechanisms let later reaction selection and tool use draw on earlier search evidence. However, such memory mainly serves as context for action selection, while task supervision remains centered on episode outcomes, so converting long interaction histories into fine-grained learning signals remains challenging.

2 PRELIMINARY

2.1 PROBLEM SETUP

Given a target molecule x and available building blocks $\mathcal{B}$, an LLM agent interacts with planning tools to construct a synthesis route whose terminal precursors belong to $\mathcal{B}$. The environment maintains an AND–OR search graph and remaining budget; observations expose reaction candidates and search status. Molecule nodes are OR nodes, resolvable through any solved reaction child; a reaction r is an AND node requiring all precursors in $\mathcal{P}(r)$ to be solved. At turn t, $\pi_\theta(a_t \mid h_t)$ generates a textual action to query candidates or select decompositions, conditioned on $h_t = (o_1, a_1, o_2, \ldots, a_{t-1}, o_t)$. The environment updates the graph and returns o_{t+1}. Calls within a turn are generated before their observations become available. A complete trajectory is $\tau = (o_1, a_1, \ldots, o_T, a_T, o_{T+1})$, with $o_1 = x$. Success requires a certified route to $\mathcal{B}$. Planning efficiency is characterized by successful route length $L(\tau)$ and search cost $N(\tau)$, which counts single-step reaction predictor calls.

2.2 AGENTIC REINFORCEMENT LEARNING

Outcome-based RL supervises the agent through terminal planning rewards. For each target x, group-based methods sample K complete trajectories $\{\tau_i\}_{i=1}^K$ from $\pi_{\theta_{\text{old}}}(\cdot \mid x)$ with rewards $R_i = R(\tau_i)$. For example, outcome-supervised GRPO Shao et al. (2024) normalizes each reward using the group mean and standard deviation. This group-relative credit assignment can be expressed as

$$A_i^{\text{group}} = \text{GroupComputation}\big(R_i, \{R_j\}_{j=1}^K\big). \tag{1}$$

For each agent trajectory, GRPO assigns the same advantage to all generated tokens across turns, providing relative supervision that favors higher-reward trajectories. Estimating these signals directly from group reward statistics avoids a separate value model, reducing memory overhead during training. This design supports large models and rollout groups, making GRPO a practical approach to scalable, outcome-supervised learning for LLM agents.

3 TRAINING LLM AGENTS WITH PTPO

Challenge: Learning from structured search evidence. Agentic retrosynthesis requires reaction selection and precursor exploration. Outcome-based rewards encourage success and efficiency but compress search evidence. This compression creates three ambiguities. First, differing candidate views can confound reaction comparisons. Second, identical rewards obscure distinct precursor outcomes. Third, temporal proximity need not identify candidate sources. The key challenge is therefore to compare reactions fairly and credit both selection and discovery.

Our method: Provenance-Trace Policy Optimization (PTPO). PTPO converts structured search evidence into policy learning signals through two complementary mechanisms: (i) matched-context reaction evaluation compares selections made under identical ordered candidate views using terminal precursor assessments, and (ii) provenance-based credit tracing attributes these comparisons to the turns that revealed the selected candidates. These mechanisms operate on the same sampled trajectories used for outcome-based learning. We first construct decision groups by matching candidate views and record candidate provenance, then estimate trajectory-level and PT advantages and optimize them through separate PPO objectives with per-channel normalization.

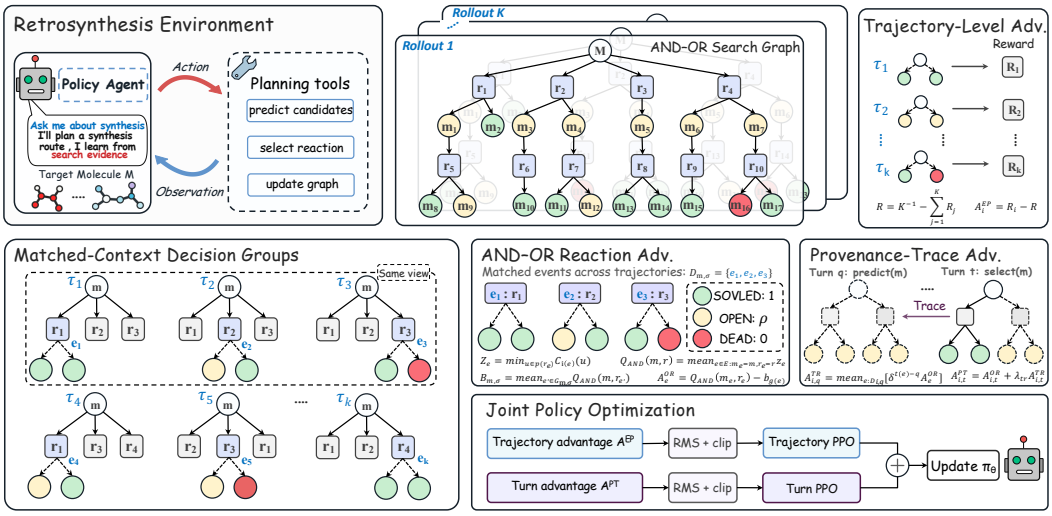


Figure 1: Overview of PTPO. For the sampled trajectories, we first construct matched-context decision groups by aggregating selections with identical candidate views for the same molecule into the same groups. We then define the reaction value as the average minimum terminal precursor score across its occurrences. Based on these groups and provenance links, we provide turn-level advantages for reaction selection and candidate discovery, respectively.

3.1 Matched-Context Decision Group Construction

Candidate Matching. A reaction selection is meaningful relative to the candidates available when it is generated. Grouping decisions solely by the product molecule overlooks this context: selecting the best reaction from one candidate set need not imply the same decision quality as selecting from another. We therefore construct comparable decision groups and record the provenance links needed for subsequent credit tracing.

For a target x, we sample K complete trajectories from $\pi_{\theta_{\text{old}}}$ with independent decoding and a shared environment seed ξ_x, which fixes candidate subsets and presentation orders within the group. Each successfully committed selection $e = (i, t, k)$ records its trajectory, turn, within-turn selection index, product m_e, canonical reaction r_e, ordered candidate signature σ_e, and producer turn q_e. Eligible selections form $\mathcal{E}$ and require $q_e < t$: the selected candidate must have appeared in an earlier observation. A candidate first returned within the selection turn is therefore ineligible. We define a decision group as $\mathcal{G}_{m,\sigma} = \{e \in \mathcal{E} : m_e = m, \sigma_e = \sigma\}$, where the signature preserves the ordered candidate-to-reaction mapping. Selections are compared within the same group, while complete search histories may differ. Each eligible selection defines a provenance link $(i, q_e) \to e$, identifying the earlier turn that revealed the selected candidate. These links identify the producer turns eligible to receive supported reaction credit through provenance tracing (Eq. 3).

3.2 Multi-Granularity Advantage Estimation

Matching candidate views establishes a common basis for reaction comparison, but terminal rewards alone cannot distinguish all intermediate outcomes. In particular, equally rewarded trajectories may resolve different precursor subproblems. We therefore combine overall planning performance with precursor-level search evidence and propagate reaction credit along recorded provenance links to the turns that supplied the selected candidates.

Trajectory-Level Advantages. To preserve the overall task objective, successful trajectories receive $R_i = 1-\alpha_L(1-L^*/L_i)-\alpha_N(1-N^*/N_i)$, where L_i and N_i denote route length and predictor calls, and L^*, N^* are their separate minima over successful trajectories for x, computed only when successes exist. This relative reward distinguishes efficient plans even when every sampled trajectory succeeds. The two reference minima need not come from the same trajectory, allowing route length and search cost to be assessed against their respective group-level optima. Unsuccessful tra-

jectories receive $R_i = -\min(\eta, \kappa E_i) - \phi \mathbf{1}[C_i = 0]$, where E_i counts turns without parseable calls and C_i counts all parseable calls. The nonnegative coefficients control efficiency preferences and format penalties, with $\alpha_L + \alpha_N < 1$ ensuring that every success outranks every failure. We obtain the trajectory-level advantage by centering within the shared-seed group, $A_i^{\mathrm{EP}} = R_i - K^{-1} \sum_{j=1}^{K} R_j$.

AND–OR Reaction Advantages. Our key observation is that a target-level outcome can obscure precursor subproblems already resolved during search. We assess each precursor at trajectory termination using $c_i(u) = 1$ for SOLVED, $c_i(u) = 0$ for DEAD, and $c_i(u) = \rho_{\mathrm{open}} \in (0, 1)$ for OPEN. SOLVED indicates a route to available building blocks; DEAD indicates infeasibility under the environment's candidate and depth constraints. Other unresolved molecules remain OPEN, receiving a fixed score that preserves the distinction between incomplete search and exhausted continuations.

Since a reaction requires all its precursors, we evaluate each occurrence by the minimum precursor score and aggregate repeated occurrences of the same product–reaction pair. Let $\mathcal{E}_{m,r}$ collect eligible selections of r for m, and let $i(e)$ denote the trajectory containing e. The reaction assessment and matched-context advantage are

$$Q_{\mathrm{AND}}(m, r) = \frac{1}{|\mathcal{E}_{m,r}|} \sum_{e \in \mathcal{E}_{m,r}} \min_{u \in \mathcal{P}(r)} c_{i(e)}(u), \qquad A_e^{\mathrm{OR}} = Q_{\mathrm{AND}}(m_e, r_e) - b_{g(e)}, \tag{2}$$

where $g(e) = (m_e, \sigma_e)$ and $b_g = \sum_r n_g(r) Q_{\mathrm{AND}}(m, r) / \sum_r n_g(r)$ is the occurrence-weighted baseline for $g = (m, \sigma)$. Here, $n_g(r)$ counts selections of r in that group. Reaction evidence is pooled across candidate views, whereas the baseline compares only selections made under one matched view. Groups require at least two distinct selected reactions to support reaction comparison; unselected candidates contribute no outcome evidence. We average all supported selection advantages per turn to obtain $A_{i,t}^{\mathrm{OR}}$, assigning zero to empty averages. This provides differentiated feedback among equally rewarded trajectories when precursor assessments differ.

Provenance-Trace Advantages. Evaluating reaction selections does not directly train the information-gathering turns that enabled them. To connect these decisions, PTPO traces each supported selection back to its recorded producer. Let $\mathcal{D}_{i,q}$ contain supported selections in trajectory i whose candidates were supplied by turn q. The trace component and combined PT advantage are

$$A_{i,q}^{\mathrm{TR}} = \frac{1}{|\mathcal{D}_{i,q}|} \sum_{e \in \mathcal{D}_{i,q}} \delta^{t(e)-q} A_e^{\mathrm{OR}}, \qquad A_{i,t}^{\mathrm{PT}} = A_{i,t}^{\mathrm{OR}} + \lambda_{\mathrm{tr}} A_{i,t}^{\mathrm{TR}}, \tag{3}$$

where $t(e)$ is the selection turn, $\delta \in (0, 1]$ controls trace decay, and λ_{tr} weights the trace component. Empty traces are zero. Each supported selection credits one verified producer; linked selections are averaged, including zero advantages. Tracing uses individual selection advantages, allowing selections within the same turn to credit different producers. The provenance link determines where credit is assigned; temporal distance only attenuates its magnitude.

3.3 Joint Trajectory and Turn-Level Policy Optimization

Trajectory-level and PT advantages provide complementary planning signals and can differ in sign within a trajectory. We preserve their granularity through separate normalization and clipped objectives. Target groups, including all-failure groups, are retained when terminal rewards differ or any PT advantage is nonzero. Within each optimizer batch, the EP normalization counts each trajectory once, while PT normalization counts each turn with nonzero PT advantage once. For channel $c \in \{\mathrm{EP}, \mathrm{PT}\}$, we use $s_c = \max(\mathrm{RMS}(A^c), f_c)$ and $\widehat{A}^c = \mathrm{clip}(A^c / s_c, -a_{\max}, a_{\max})$, taking $\mathrm{RMS}(\varnothing) = 0$. The floor f_c prevents excessive amplification of small signals.

EP and PT advantages use trajectory- and turn-level ratios, respectively, recomputed at each PPO step. For generated token positions V, define the length-normalized ratio $\varrho(V) = \exp\big(|V|^{-1} \sum_{v \in V} \log[\pi_\theta(y_v \mid H_v) / \pi_{\theta_{\mathrm{old}}}(y_v \mid H_v)]\big)$, where H_v is the full token history. We use $\varrho_i = \varrho(V_i)$ for all generated tokens in trajectory i and $\varrho_{i,t} = \varrho(V_{i,t})$ for those in turn t; tool observations are excluded. Writing the clipped surrogate as $\ell_\epsilon(\varrho, A) = \min\{\varrho A, \mathrm{clip}(\varrho, 1 - \epsilon^-, 1 + \epsilon^+) A\}$,

with clipping widths $\epsilon = (\epsilon^-, \epsilon^+)$, the objective is

$$\mathcal{J}_{\mathrm{PTPO}}(\theta) = \mathbb{E}\left[\frac{1}{B}\sum_{i=1}^{B} \ell_{\epsilon_{\mathrm{seq}}}(\varrho_i, \widehat{A}_i^{\mathrm{EP}}) + \lambda_{\mathrm{pt}} w_{\mathcal{S}} \sum_{(i,t)\in\mathcal{S}} \ell_{\epsilon_{\mathrm{turn}}}(\varrho_{i,t}, \widehat{A}_{i,t}^{\mathrm{PT}}) - \beta_{\mathrm{KL}}\mathcal{D}_{\mathrm{KL}}\right], \quad (4)$$

where B counts retained trajectories, $\mathcal{S}$ contains their turns with nonzero PT advantages, and $w_{\mathcal{S}} = \mathbf{1}[|\mathcal{S}| \geq M_{\min}]/\max(1, |\mathcal{S}|)$ suppresses insufficiently supported turn-level updates. The expectation is over retained training batches; λ_{pt} weights the PT contribution, and $\epsilon_{\mathrm{seq}}, \epsilon_{\mathrm{turn}}$ specify the respective lower and upper clipping widths. The reference-policy KL penalty $\mathcal{D}_{\mathrm{KL}}$ averages over generated tokens within each trajectory and then over trajectories. These separate objectives allow intermediate reaction outcomes and their information dependencies to influence learning alongside overall planning performance.

4 Experiment

4.1 Experimental Setting

Datasets. Following prior work (Chen et al., 2020), we use a building-block set of 23,081,633 commercially available molecules from eMolecules.[1] For PTPO training, we extract 20,626 target molecules from USPTO-Full (Chen et al., 2020), retaining targets with routes deeper than 4 reaction steps, with a mean depth of 5.93. For in-distribution evaluation, we use USPTO-190 (Chen et al., 2020), comprising 190 challenging targets. For out-of-distribution evaluation, we use ChEMBL-1000 (Liu et al., 2023), which contains 1,000 drug-like molecules.

Baselines. We compare with both search- and LLM-based baselines. We include Greedy DFS, Retro*, and Retro*-0 (Chen et al., 2020); Retro*+ and Retro*+-0 (Kim et al., 2021); EG-MCTS (Hong et al., 2023); and PDVN (Liu et al., 2023). These baselines cover likelihood-guided depth-first search, neural-guided best-first search, self-improved reaction prediction, experience-guided Monte Carlo tree search, and dual value estimation, enabling comparisons across different approaches to reaction selection and search control. Our primary LLM-based baselines are Retro-R1 (Liu et al., 2025) and RetroAgent (Zhu et al., 2026). Retro-R1 uses `Qwen2.5-7B-Instruct-1M` with a two-action interface and depth-first search without backtracking, while RetroAgent uses the same backbone as PTPO.

Single-step Model. Following prior work (Zhu et al., 2026), we generate single-step reaction candidates with a template-based MLP using pretrained DESP weights (Yu et al., 2024). The network processes a 2048-bit, chirality-aware Morgan fingerprint through two 2048-unit hidden layers and outputs a softmax distribution over 381,302 USPTO reaction templates. For each queried molecule, RDChiral (Coley et al., 2019a) applies and verifies the top-scoring templates, while the agent receives only the resulting precursor sets. We shuffle the returned candidates and the reported frontier during both training and evaluation, encouraging assessment based on content rather than position. We include Retro-R1 (Liu et al., 2025) with V1 weights as a primary baseline, and exclude V4 because its weights were not publicly available, and V2 and V3 due to concerns about chemically unrealistic reaction predictions (Zhu et al., 2026; Genheden & Bjerrum, 2022).

Experiment Setup. We directly optimize Qwen3-4B-Instruct-2507 (Team, 2025) with reinforcement learning, without continued pretraining or supervised fine-tuning. Experiments use the slime framework (Zhu et al., 2025) on 8 H200 GPUs. For each target, we independently sample 8 trajectories with a shared environment seed to match candidate subsets and orderings. Training trajectories are limited to 90 assistant turns, with a total context threshold of 56,000 tokens. Only assistant-generated tokens contribute to the policy loss. The provenance-trace weight and discount factor are 0.3 and 0.6, respectively, and unresolved precursors receive an assessment of 0.5. We clip normalized advantages to $[-3, 3]$, enable the local objective only when at least 64 turns provide local credit within an optimizer step, and set the KL coefficient to 10^{-3}. Following Liu et al. (2025), we report Pass@1, the fraction of targets solved within a single trajectory, and success rate at budgets $N \in \{50, 100, 200, 300, 400, 500\}$. More implementation details are provided in Appendix D.

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

Table 1: Pass@1 and success rate (%) at different predictor-call budgets N on USPTO-190 and ChEMBL-1000. Best in **bold**, second best underlined.

	Method	Pass@1	Success Rate (%) at Budget N					
		(%)	N=50	N=100	N=200	N=300	N=400	N=500
USPTO-190	Greedy DFS	19.47	19.47	19.47	19.47	19.47	19.47	19.47
	Retro*	20.53	38.95	50.00	68.95	74.74	77.89	79.47
	Retro*-0	19.47	27.37	37.89	55.79	62.63	72.11	74.21
	Retro*+	30.00	55.79	68.42	79.47	83.16	84.21	84.21
	Retro*+-0	25.26	47.89	61.05	76.84	82.11	85.79	86.32
	EG-MCTS	46.84	58.42	64.74	69.47	75.26	78.42	81.05
	PDVN	42.63	64.74	72.11	82.63	**87.89**	**91.58**	**92.11**
	Retro-R1 (V1)	50.00±0.00	64.63±1.68	71.32±1.23	78.11±1.68	80.89±1.81	82.95±1.40	84.32±1.22
	RETROAGENT	53.26±0.69	58.42±1.34	70.53±1.93	78.63±1.27	81.58±0.83	83.68±0.37	85.05±0.80
	PTPO	**67.89**±1.90	**70.74**±1.42	**80.32**±1.09	**84.74**±1.34	86.32±0.83	87.26±1.41	87.58±1.27
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 (V1)	68.52±1.58	73.92±0.37	78.00±0.46	80.36±0.35	81.36±0.27	81.98±0.28	82.42±0.22
	RETROAGENT	73.82±0.49	76.36±0.35	79.62±0.15	81.96±0.30	83.22±0.26	83.94±0.30	84.40±0.38
	PTPO	**78.04**±0.60	**79.92**±0.30	**82.44**±0.10	**84.20**±0.40	**84.98**±0.40	**85.48**±0.30	**86.00**±0.30

4.2 EXPERIMENTAL RESULTS

Main Results. As shown in Table 1, PTPO achieves the highest Pass@1 among the evaluated baselines on both benchmarks. Compared with RetroAgent, which uses the same 4B backbone and structured planning environment, PTPO improves Pass@1 by 14.63 percentage points on USPTO-190 and 4.22 percentage points on ChEMBL-1000, reaching 67.89% and 78.04%, respectively. It also exceeds Retro-R1 (V1) by 17.89 and 9.52 percentage points. These results demonstrate improved single-trajectory planning success under both in-distribution and out-of-distribution evaluation. Furthermore, PTPO leads all baselines at budgets $N \in \{50, 100, 200\}$ on USPTO-190 and across all evaluated budgets on ChEMBL-1000. For example, at $N = 50$, PTPO achieves success rates of 70.74% and 79.92%, improving over RETROAGENT by 12.32 and 3.56 percentage points, respectively. Its margins over RETROAGENT generally decrease as the prediction budget increases, reaching 2.53 and 1.60 percentage points at $N = 500$. On USPTO-190, PDVN retains an advantage at $N \geq 300$, attaining 92.11% at $N = 500$ compared with 87.58% for PTPO.

Target-Level Planning Efficiency. Figure 2 compares the cumulative fraction of ChEMBL-1000 targets solved under different single-step model call budgets. The plotted results characterize the selected checkpoints, while Table 1 reports averages over ten independent experiments. PTPO's curve separates from Retro-R1 and RetroAgent within the first few calls and remains higher thereafter, indicating that PTPO solves more targets under the same call budget. Its success rate plateaus at 77.60%, compared with 70.30% for Retro-R1 and 70.20% for RetroAgent. Equivalently, for success levels reached by all three methods, PTPO generally requires fewer calls, indicating greater efficiency in reaching comparable target coverage. The persistent plateau gap further indicates that additional calls within the evaluated range do not close the performance difference between PTPO and either baseline.

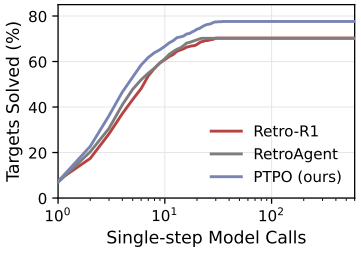


Figure 2: Cumulative targets solved versus single-step model call budgets.

4.3 Ablation Study

Table 2: Core component ablations of PTPO on USPTO-190. Except for the base model, each variant changes only the indicated component. Best in **bold**, second best underlined.

Variant	Pass@1 (%)	Success Rate (%) at Budget N					
		$N = 50$	$N = 100$	$N = 200$	$N = 300$	$N = 400$	$N = 500$
PTPO	**67.89**±1.90	**70.74**±1.42	**80.32**±1.09	**84.74**±1.34	**86.32**±0.83	**87.26**±1.41	**87.58**±1.27
Base model (no RL)	6.10±2.10	18.80±1.60	18.90±1.60	18.90±1.60	18.90±1.60	18.90±1.60	18.90±1.60
w/o candidate matching ($g(e)=m_e$)	54.39±2.65	63.51±3.08	71.75±2.13	77.72±1.32	80.35±0.80	81.93±1.22	82.28±1.10
w/o PT advantages ($\lambda_{\text{pt}}=0$)	46.50±0.80	48.20±0.60	59.80±1.60	67.90±1.60	71.20±1.60	73.30±1.10	73.90±0.60
w/o EP advantages ($A^{\text{EP}}=0$)	25.80±1.60	48.10±0.80	57.40±1.60	60.70±0.60	63.00±1.80	63.70±1.90	63.70±1.90
w/o provenance tracing ($\lambda_{\text{tr}}=0$)	62.28±2.60	68.42±2.29	75.79±0.91	82.63±0.91	84.39±1.10	85.09±1.32	85.26±1.05
w/o turn-level PT ratio (ϱ_i for A^{PT})	38.42±0.53	54.91±1.99	65.09±1.61	72.28±1.32	73.33±0.80	73.51±1.10	73.51±1.10

As shown in Table 2, without reinforcement learning, the base model reaches 6.10% Pass@1, saturating at 18.90% at N=100. Removing PT advantages (λ_{pt}=0) reduces Pass@1 to 46.50%, leaving only trajectory-level advantages. Removing trajectory-level EP advantages (A^{EP}=0) yields 25.80% Pass@1 and 63.70% at N=500, substantially above the base model but below full PTPO, showing PT advantages independently aid learning while benefiting from trajectory-level signals. Grouping solely by molecule (w/o candidate matching), excluding ordered candidate signatures, gives 54.39%, while disabling credit propagation to candidate-revealing turns (w/o provenance tracing) gives 62.28%. Except for removing EP advantages, a trajectory-level ratio for PT advantages (w/o turn-level PT ratio) causes the largest Pass@1 drop (29.47 points) by weakening turn-level attribution. At N=500, variants without provenance tracing, candidate matching, PT advantages, and the turn-level PT ratio trail by 2.32, 5.30, 13.68, and 14.07 points, respectively. These results support PT advantages and turn-level PT ratios as complementary contributors to PTPO.

4.4 Analysis

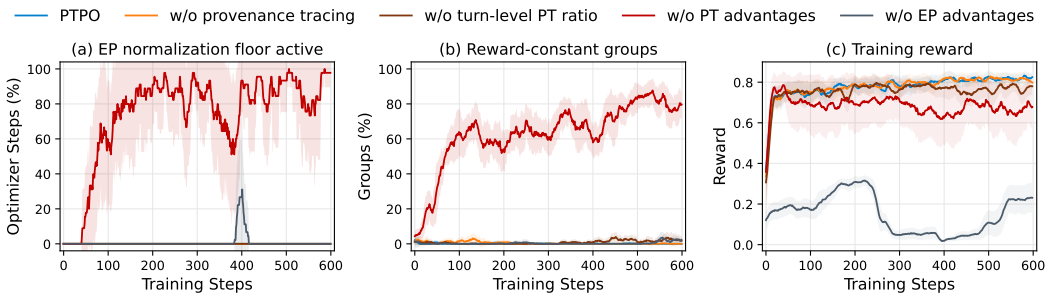


Figure 3: Training dynamics of PTPO and ablations. (a) EP normalization-floor activation fraction; (b) reward-constant group fraction after dynamic sampling; (c) mean reward before dynamic sampling, with midpoint curves and mixed-group bounds.

Training Dynamics. Figure 3 compares PTPO with its ablations. Without PT advantages, reward-constant groups after dynamic sampling exceed 50% within 70 training steps and average 82% over the final 100 steps (Figure 3(b)). These groups have $A^{\text{EP}} = 0$ and contribute no EP policy-gradient signal. Under the shared normalization configuration, the EP floor f_{EP} is active in 89% of optimizer steps during the final 100 steps (Figure 3(a)), with normalized EP advantages having RMS below one before clipping. Variants retaining both channels maintain low rates of reward-constant groups and EP floor activation, reaching training rewards of approximately 0.77–0.82, versus 0.67 without PT advantages (Figure 3(c)). Rewards are measured before dynamic sampling. Without EP advantages, reward-constant groups also remain rare, yet reward drops from approximately 0.3 to below 0.1 around steps 250–300 and only partially recovers later. We observe two failure patterns under the evaluated configuration: diminished EP learning signals with a lower reward plateau without PT, and reward collapse despite persistent within-group reward variation without EP.

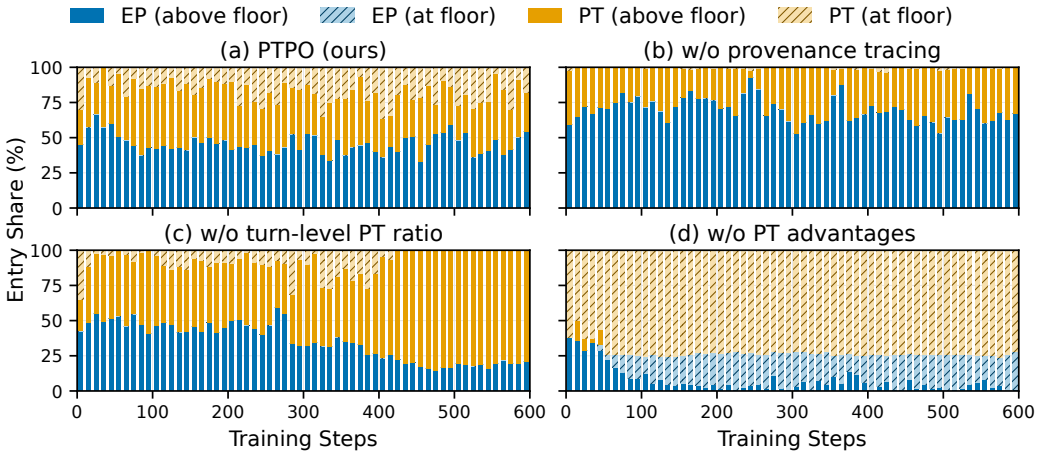


Figure 4: Live and clamped credit mass in PTPO and ablations.

Advantage Entries. Figure 4 reports advantage-entry shares and normalization-floor status, not gradient contributions. Both PTPO channels sustain substantial above-floor shares; EP remains above-floor (Figure 4(a)). Removing tracing cuts PT's share from roughly half to one-third (Figure 4(b)), consistent with crediting earlier selected-candidate producers. With trajectory-level PT ratios, PT's share approaches 80% (Figure 4(c)); the loss formulation changes, but not the advantage estimator. Without PT advantages, only EP remains active: its above-floor entry share falls from roughly one-third to under 2% (Figure 4(d)); nearly all remaining EP and computed but unused PT entries use floor normalization. Identical within-group rewards yield $A^{\mathrm{EP}} = 0$; such groups' high prevalence coincides with EP RMS below the normalization floor (Figure 3). PT compares joint precursor outcomes under identical candidate signatures without requiring terminal-reward differences.

Computational Budget. PTPO retains the core operations of group-based RL: multi-turn rollout generation, old- and reference-policy probability evaluation, and clipped policy updates. PTPO adds candidate matching for decision group construction and PT advantage estimation through AND–OR reaction assessment and provenance tracing, reusing precursor assessments and candidate provenance recorded during rollouts without a learned critic or additional trajectory sampling. As shown in Figure 5, rollout generation, policy updates, and probability evaluation dominate the measured runtime, taking 376.9 s, 49.4 s, and 23.5 s, respectively. Candidate matching takes 0.036 s, while AND–OR reaction assessment and provenance tracing together require 0.093 s. EP advantage estimation adds 0.0003 s. Total credit computation takes approximately 0.13 s, or 0.03% of the total profiled time. Under the profiled training configuration, PTPO thus derives PT advantages for reaction selection and information acquisition at negligible computational cost relative to core RL operations.

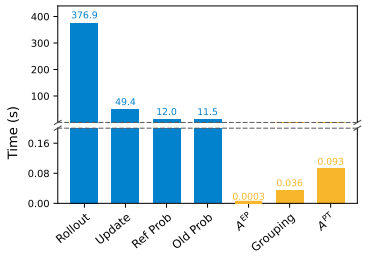


Figure 5: Time breakdown per training step. PTPO requires only 0.13 s for credit computation.

5 Conclusion

In this work, we propose Provenance-Trace Policy Optimization (PTPO), adopting a provenance-based credit assignment strategy in place of outcome-only credit assignment for retrosynthetic agent RL. By incorporating AND–OR synthesis dependencies and candidate provenance into policy optimization, PTPO can compare joint precursor outcomes under identical candidate views and attribute credit to both reaction selections and the earlier turns that revealed the selected candidates. Experimental results suggest that PTPO provides an effective pathway for improving planning success and generalization across in-distribution and out-of-distribution retrosynthesis benchmarks.

LLM USAGE STATEMENT

This work used LLM assistance for literature search, language editing, and coding that was subsequently inspected and refined by hand. All LLM-assisted code and writing were validated by the authors, who take full responsibility for the final manuscript and released code.

REPRODUCIBILITY STATEMENT

We have made extensive efforts to ensure reproducibility. The benchmarks used in our experiments, USPTO-190 and ChEMBL-1000, are publicly available, with data usage and evaluation protocols described in the main text and appendix. Code is available at `https://anonymous.4open.science/r/PTPO-B5D4/`. Model configurations, training hyperparameters, reward definitions, and evaluation metrics are documented in the main text and appendix.

ETHICS STATEMENT

This study focuses on computational retrosynthetic planning using public reaction and molecular datasets. The reported experiments evaluate model training and automated synthesis-route construction. The planning metrics measure computational route completion and do not establish experimental feasibility or chemical safety. We recognize the potential dual use of automated synthesis planning and emphasize responsible applications in chemical research. Any practical use of the generated routes requires independent assessment of chemical feasibility, safety, and appropriate handling by qualified experts. We are committed to transparent reporting and responsible development of agentic retrosynthesis systems.

REFERENCES

David C Blakemore, Luis Castro, Ian Churcher, David C Rees, Andrew W Thomas, David M Wilson, and Anthony Wood. Organic synthesis provides opportunities to transform drug discovery. *Nature chemistry*, 10(4):383–394, 2018.

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

Shuan Chen and Yousung Jung. Deep retrosynthetic reaction prediction using local reactivity and global attention. *JACS Au*, 1(10):1612–1620, 2021.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In Inês Lynce, Nello Murano, Mauro Vallati, Serena Villata, Federico Chesani, Michela Milano, Andrea Omicini, and Mehdi Dastani (eds.), *ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025)*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pp. 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL `https://doi.org/10.3233/FAIA251160`.

Connor W. Coley, William H. Green Jr., and Klavs F. Jensen. Rdchiral: An rdkit wrapper for handling stereochemistry in retrosynthetic template extraction and application. *J. Chem. Inf. Model.*, 59(6):2529–2537, 2019a. doi: 10.1021/ACS.JCIM.9B00286. URL `https://doi.org/10.1021/acs.jcim.9b00286`.

Connor W Coley, Dale A Thomas III, Justin AM Lummiss, Jonathan N Jaworski, Christopher P Breen, Victor Schultz, Travis Hart, Joshua S Fishman, Luke Rogers, Hanyu Gao, et al. A robotic platform for flow synthesis of organic compounds informed by ai planning. *Science*, 365(6453): eaax1566, 2019b.

Elias James Corey and W Todd Wipke. Computer-assisted design of complex organic syntheses: Pathways for molecular synthesis can be devised with a computer and equipment for graphical communication. *Science*, 166(3902):178–192, 1969.

Hanjun Dai, Chengtao Li, Connor W. Coley, Bo Dai, and Le Song. Retrosynthesis prediction with conditional graph logic network. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8870–8880, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html.

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. *CoRR*, abs/2504.11536, 2025a. doi: 10.48550/ARXIV.2504.11536. URL https://doi.org/10.48550/arXiv.2504.11536.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025b. URL http://papers.nips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

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

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.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025. doi: 10.48550/ARXIV.2503.09516. URL https://doi.org/10.48550/arXiv.2503.09516.

Junsu Kim, Sungsoo Ahn, Hankook Lee, and Jinwoo Shin. Self-improved retrosynthetic planning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5486–5495. PMLR, 2021. URL http://proceedings.mlr.press/v139/kim21b.html.

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.

Bowen Liu, Bharath Ramsundar, Prasad Kawthekar, Jade Shi, Joseph Gomes, Quang Luu Nguyen, Stephen Ho, Jack Sloane, Paul Wender, and Vijay Pande. Retrosynthetic reaction prediction using neural sequence-to-sequence models. *ACS central science*, 3(10):1103–1113, 2017.

Guoqing Liu, Di Xue, Shufang Xie, Yingce Xia, Austin Tripp, Krzysztof Maziarz, Marwin H. S. Segler, Tao Qin, Zongzhang Zhang, and Tie-Yan Liu. Retrosynthetic planning with dual value networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22266–22276. PMLR, 2023. URL https://proceedings.mlr.press/v202/liu23as.html.

Wei Liu, Jiangtao Feng, Hongli Yu, Yuxuan Song, Yuqiang Li, Shufei Zhang, Lei Bai, Wei-Ying Ma, and Hao Zhou. Retro-r1: Llm-based agentic retrosynthesis. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego,*

CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/6697768644aeabfd163726c983f75795-Abstract-Conference.html.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/97c5b2707228e7e3fb67e4ecc2e0e607-Abstract-Conference.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Philippe Schwaller, Riccardo Petraglia, Valerio Zullo, Vishnu H Nair, Rico Andreas Haeuselmann, Riccardo Pisoni, Costas Bekas, Anna Iuliano, and Teodoro Laino. Predicting retrosynthetic pathways using transformer-based models and a hyper-graph exploration strategy. *Chemical science*, 11(12):3316–3325, 2020.

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.

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.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053, 2019. URL http://arxiv.org/abs/1909.08053.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *CoRR*, abs/2503.05592, 2025. doi: 10.48550/ARXIV.2503.05592. URL https://doi.org/10.48550/arXiv.2503.05592.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff G. Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning. *CoRR*, abs/2602.02710, 2026. doi: 10.48550/ARXIV.2602.02710. URL https://doi.org/10.48550/arXiv.2602.02710.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Haorui Wang, Jeff Guo, Lingkai Kong, Rampi Ramprasad, Philippe Schwaller, Yuanqi Du, and Chao Zhang. Llm-augmented chemical synthesis and design decision programs. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025a. URL https://proceedings.mlr.press/v267/wang25ag.html.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: understanding self-evolution in LLM agents via multi-turn reinforcement learning. *CoRR*, abs/2504.20073, 2025b. doi: 10.48550/ARXIV.2504.20073. URL https://doi.org/10.48550/arXiv.2504.20073.

Shufang Xie, Rui Yan, Peng Han, Yingce Xia, Lijun Wu, Chenjuan Guo, Bin Yang, and Tao Qin. Retrograph: Retrosynthetic planning with graph search. In Aidong Zhang and Huzefa Rangwala (eds.), *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pp. 2120–2129. ACM, 2022. doi: 10.1145/3534678.3539446. URL https://doi.org/10.1145/3534678.3539446.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Yifei Yang, Runhan Shi, Zuchao Li, Shu Jiang, Bao-Liang Lu, Qibin Zhao, Yang Yang, and Hai Zhao. Batgpt-chem: A foundation large model for chemical engineering. *Research*, 8:0827, 2025.

Kevin Yu, Jihye Roh, Ziang Li, Wenhao Gao, Runzhong Wang, and Connor W. Coley. Double-ended synthesis planning with goal-constrained bidirectional search. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/cd091a4d8e97157d32940428f902c7b0-Abstract-Conference.html.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025a. URL http://papers.nips.cc/paper_files/paper/2025/hash/9837dc00ff67d176373268ed48042d49-Abstract-Conference.html.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *CoRR*, abs/2505.19590, 2025b. doi: 10.48550/ARXIV.2505.19590. URL https://doi.org/10.48550/arXiv.2505.19590.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *CoRR*, abs/2507.18071, 2025. doi: 10.48550/ARXIV.2507.18071. URL https://doi.org/10.48550/arXiv.2507.18071.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark W. Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

Yanqiao Zhu, Jingru Gan, Xiaoqi Sun, Fang Sun, Yidan Shi, Md Mofijul Islam, Chao Shang, Wenhao Gao, Connor W. Coley, Yizhou Sun, and Wei Wang. Retroagent: Harnessing llms to search over structured memory for agentic retrosynthesis planning. *CoRR*, abs/2607.14512, 2026. doi: 10.48550/ARXIV.2607.14512. URL https://doi.org/10.48550/arXiv.2607.14512.

Zilin Zhu, Chengxing Xie, X Lv, et al. slime: An llm post-training framework for rl scaling, 2025.

Appendix

A Dataset Details **15**

B Baseline Details **15**
B.1 Search-Based Baselines 15
B.2 LLM-Agent Baselines 15

C Evaluation Details **15**

D Implementation Details **16**

E PTPO Algorithm **16**

F More Experimental Results **17**
F.1 Training Dynamics 17
F.2 Search Tree Structure 19
F.3 Same-Backbone Comparison 20
F.4 Analysis of Rollout Termination Modes 20
F.5 Route Length 21
F.6 Template Overlap 21
F.7 Terminal Outcomes versus Reaction-Level Certification 21

G Prompts Used in PTPO **22**

H Tool Environment Used in PTPO **22**

I Multi-turn Interaction Experiments **23**

J Case Study **28**

K Limitations and Future Work **29**

A Dataset Details

We conduct experiments on two widely used retrosynthesis benchmarks, covering both in-distribution and out-of-distribution evaluation settings:

- **USPTO-190** (Chen et al., 2020): A standard in-distribution benchmark containing 190 challenging target molecules extracted from USPTO-Full, together with reference synthesis routes.
- **ChEMBL-1000** (Liu et al., 2023): An out-of-distribution benchmark containing 1,000 drug-like molecules sampled from ChEMBL without reference routes, used to evaluate the generalization of retrosynthesis planners beyond the training distribution.

For reinforcement learning, we use 20,626 target molecules from USPTO-Full whose reference-route depths fall between 5 and 30. Molecules appearing in either evaluation set are excluded from the training pool. We evaluate on all 190 USPTO-190 targets and all 1,000 ChEMBL-1000 targets.

B Baseline Details

We compare PTPO with two groups of baselines based on different planning paradigms:

B.1 Search-Based Baselines

- **Greedy DFS**: A depth-first search baseline that always prioritizes the reaction assigned the highest probability by the single-step retrosynthesis predictor.
- **Retro*** (Chen et al., 2020): A neural A*-like best-first search algorithm in which the accumulated cost is derived from single-step reaction probabilities and the future cost is estimated by a learned value network.
- **Retro*-0** (Chen et al., 2020): The non-learning variant of Retro* that sets the future-cost estimate to zero and therefore performs search without a learned value function.
- **Retro*+** (Kim et al., 2021): An enhanced Retro* planner using a self-improved single-step reaction model trained from successful routes discovered during planning.
- **Retro*+-0** (Kim et al., 2021): The corresponding zero-value variant of Retro*+, combining the self-improved single-step model with search without learned future-cost guidance.
- **EG-MCTS** (Hong et al., 2023): An experience-guided Monte Carlo tree search method that learns a guidance network from previously explored synthesis routes to improve action evaluation.
- **PDVN** (Liu et al., 2023): A reinforcement-learning-based planner that jointly improves the single-step reaction policy and learns dual value networks for estimating synthesizability and route cost.

B.2 LLM-Agent Baselines

- **Retro-R1** (Liu et al., 2025): A retrosynthesis agent trained with reinforcement learning. It interacts with a two-action planning interface and follows depth-first search without explicit backtracking.
- **RetroAgent** (Zhu et al., 2026): A retrosynthesis agent trained with GSPO that directly controls a tool-augmented AND–OR search environment. Its structured memory exposes explored routes, frontier molecules, and alternative decompositions to the policy.

C Evaluation Details

We evaluate PTPO using the standalone evaluation pipeline with the same tool server as training, with greedy decoding (temperature 0). Each rollout runs for up to 120 turns and a 120K-token budget, more generous than training (90 turns, 56K tokens) to allow extra exploration. Following the iterative protocol, each experiment issues successive rollouts for a target until a route is found or the search budget is exhausted; the budget counts unique molecules submitted to the single-step predictor, predictions retrieved from the cross-rollout cache incur no additional cost, and Pass@1 is taken from the first rollout of each experiment. We run 10 independent experiments per target and report the mean and standard deviation, with stochasticity coming entirely from shuffled prediction

and frontier orderings in the tool server, not from LLM sampling. A rollout terminates when the root molecule is solved (completed route), when maximum turns are reached (out of turns), when no successful expansion occurs for five consecutive turns (no progress), when the frontier is exhausted (dead frontier), when the token budget is exhausted (out of context), or when a single turn exceeds the response limit of 8,192 tokens due to repetition (out of response). Reaching the maximum route depth of 30 reaction steps blocks only the offending branch and does not by itself end the rollout.

D Implementation Details

As shown in Table 3, we summarize the detailed hyperparameter configurations used throughout our experiments. We initialize the policy from Qwen3-4B-Instruct-2507 (Team, 2025) and train it using the slime framework (Zhu et al., 2025) with Megatron (Shoeybi et al., 2019) and SGLang (Zheng et al., 2024).

Table 3: Hyperparameter settings for PTPO.

Category	Setting	Value
Model	Policy backbone	Qwen3-4B-Instruct-2507
Model	Numerical precision	bfloat16
Model	Context length	65,536 tokens
Model	Maximum response length	8,192 tokens
Rollout	Training sampling temperature	1.0
Rollout	Group size K	8 trajectories per target
Rollout	Rollout batch	32 targets $\times$ 8 trajectories
Rollout	Maximum training episode	90 turns / 56K tokens
Rollout	Candidate reactions	Shuffle top 50; expose at most 20
PTPO	Relative reward weights	$\alpha_L = 0.35,\ \alpha_N = 0.25$
PTPO	Failure penalties	$\kappa = 0.025,\ \eta = 0.20,\ \phi = 0.25$
PTPO	Open-precursor score	$\rho_{\text{open}} = 0.5$
PTPO	Provenance trace	$\lambda_{\text{tr}} = 0.3,\ \delta = 0.6$
PTPO	Local-objective weight	$\lambda_{\text{local}} = 0.25$
PTPO	RMS normalization floors	$f_{\text{EP}} = 0.05,\ f_{\text{LOCAL}} = 0.10$
PTPO	Advantage clipping	$a_{\max} = 3.0$
PTPO	Minimum local support	$M_{\min} = 64$ turns
PTPO	Shared environment seeds	1 seed per trajectory group
Optimization	Optimizer	Adam, $\beta_1 = 0.9,\ \beta_2 = 0.98$
Optimization	Learning rate / weight decay	1×10^{-6} constant / 0.01
Optimization	Global batch size	128 trajectories
Optimization	Maximum training iterations	600
Optimization	Sequence clipping	$\epsilon_{\text{seq}} = 3.5 \times 10^{-4}$
Optimization	Turn-level clipping	$\epsilon_{\text{turn}} = 4\epsilon_{\text{seq}} = 1.4 \times 10^{-3}$
Optimization	TIS configuration	2 updates per rollout; clip range $[0.5, 2.0]$
Optimization	KL regularization	$\beta_{\text{KL}} = 0.001$; low-variance KL
Infrastructure	RL framework	`slime`
Infrastructure	Training backend	Megatron
Infrastructure	Rollout backend	SGLang
Infrastructure	Training hardware	8$\times$ NVIDIA H200 GPUs
Infrastructure	Tool-server workers	16 independent instances

E PTPO Algorithm

Algorithm 1 summarizes PTPO through two procedures. The credit estimation procedure computes trajectory-level advantages, evaluates reaction selections within matched candidate contexts that provide a common basis for comparison among the available alternatives, and traces their advantages to the corresponding producer turns, thereby linking downstream feedback to the earlier decisions responsible for candidate generation. Building on this credit assignment, the optimization procedure retains informative rollout groups and updates the policy using separately normalized trajectory-level and provenance-trace objectives, thus combining trajectory feedback with credit assigned to producer turns. This normalization reduces sensitivity to differences in the numerical scales of the two advantage channels, while reference-policy KL regularization helps limit policy drift.

Algorithm 1 Provenance-Trace Policy Optimization.

Require: Policy π_θ, reference policy π_{ref}, training set $\mathcal{D}$, group size K, and hyperparameters Ω
Ensure: Optimized policy π_θ

procedure CREDIT($\pi_{\theta_{\text{old}}}, x, K$)
  $\tau_{1:K} \leftarrow \text{ROLLOUT}(\pi_{\theta_{\text{old}}}, x, K; \xi_x)$
  $\mathcal{E} \leftarrow \text{ELIGIBLESELECTIONS}(\tau_{1:K})$ ▷ Require $q_e < t(e)$
  $\{(L_i, N_i, E_i, C_i)\}_{i=1}^K \leftarrow \text{STATISTICS}(\tau_{1:K})$
  $\mathcal{I}^+ \leftarrow \{i : \tau_i \text{ is successful}\}$
  if $\mathcal{I}^+ \neq \varnothing$ **then**
    $(L^*, N^*) \leftarrow (\min_{i\in\mathcal{I}^+} L_i, \min_{i\in\mathcal{I}^+} N_i)$
    $R_i \leftarrow 1 - \alpha_L(1 - L^*/L_i) - \alpha_N(1 - N^*/N_i), \quad i \in \mathcal{I}^+$
  end if
  $R_i \leftarrow -\min(\eta, \kappa E_i) - \phi \mathbf{1}[C_i = 0], \quad i \notin \mathcal{I}^+$
  $A_i^{\text{EP}} \leftarrow R_i - \frac{1}{K}\sum_{j=1}^K R_j, \quad i = 1, \ldots, K$
  $\{c_i(u)\} \leftarrow \text{TERMINALSCORES}(\tau_{1:K}; \rho_{\text{open}})$
  $(\mathcal{G}, \mathcal{E}_{m,r}) \leftarrow \text{MATCHEDGROUPS}(\mathcal{E})$
  $\mathcal{G}^+ \leftarrow \{g \in \mathcal{G} : |\{r : n_g(r) > 0\}| \geq 2\}$
  $\mathcal{E}^+ \leftarrow \{e \in \mathcal{E} : g(e) \in \mathcal{G}^+\}$
  $(Q_{\text{AND}}, \{A_e^{\text{OR}}\}_{e\in\mathcal{E}^+}) \leftarrow \text{AND_OR}(\mathcal{E}_{m,r}, \mathcal{G}^+, \{c_i(u)\})$ ▷ Eq. 2
  $A_{i,t}^{\text{OR}} \leftarrow \text{TURNMEAN}(\mathcal{E}^+, \{A_e^{\text{OR}}\})$
  $A_{i,t}^{\text{TR}} \leftarrow \text{TRACE}(\mathcal{E}^+, \{A_e^{\text{OR}}\}, \delta)$
  $A_{i,t}^{\text{PT}} \leftarrow A_{i,t}^{\text{OR}} + \lambda_{\text{tr}} A_{i,t}^{\text{TR}}$ ▷ Eq. 3
  return $\tau_{1:K}, A^{\text{EP}}, A^{\text{PT}}$
end procedure

procedure PTPO($\pi_\theta, \pi_{\text{ref}}, \mathcal{D}$)
  for $\ell = 1, \ldots, I$ **do**
    $\theta_{\text{old}} \leftarrow \theta, \quad \mathfrak{B} \leftarrow \varnothing$
    for each target x in a minibatch sampled from $\mathcal{D}$ **do**
      $(\tau, A^{\text{EP}}, A^{\text{PT}}) \leftarrow \text{CREDIT}(\pi_{\theta_{\text{old}}}, x, K)$
      if $A^{\text{EP}} \not\equiv 0$ **or** $A^{\text{PT}} \not\equiv 0$ **then**
        $\mathfrak{B} \leftarrow \mathfrak{B} \cup \{(\tau, A^{\text{EP}}, A^{\text{PT}})\}$
      end if
    end for
    if $\mathfrak{B} = \varnothing$ **then continue**
    $(\tau_{1:B}, A^{\text{EP}}, A^{\text{PT}}) \leftarrow \text{FLATTEN}(\mathfrak{B})$
    $\mathcal{S} \leftarrow \{(i,t) : A_{i,t}^{\text{PT}} \neq 0\}$
    $s_{\text{EP}} \leftarrow \max\big(\text{RMS}(\{A_i^{\text{EP}}\}_{i=1}^B), f_{\text{EP}}\big)$
    $s_{\text{PT}} \leftarrow \max\big(\text{RMS}(\{A_{i,t}^{\text{PT}}\}_{(i,t)\in\mathcal{S}}), f_{\text{PT}}\big)$
    for $c \in \{\text{EP}, \text{PT}\}$ **do**
      $\widehat{A}^c \leftarrow \text{clip}(A^c/s_c, -a_{\max}, a_{\max})$
    end for
    $\widehat{\mathfrak{B}} \leftarrow (\tau_{1:B}, \widehat{A}^{\text{EP}}, \widehat{A}^{\text{PT}}, \mathcal{S})$
    $\theta \leftarrow \text{PPOUPDATE}(\theta, \theta_{\text{old}}, \widehat{\mathfrak{B}}, \pi_{\text{ref}}; \Omega)$ ▷ Eq. 4
  end for
  return π_θ
end procedure

F MORE EXPERIMENTAL RESULTS

F.1 TRAINING DYNAMICS

Figure 6 presents the training dynamics of PTPO and its ablations. In panels (a) and (b), PTPO and the variants retaining both EP and PT advantages rapidly improve the training solve rate and reward, whereas the variant without PT advantages levels off at about 68% solve rate and a reward of about 0.67. Without EP advantages, the solve rate initially reaches approximately 80%, then falls below 20% around steps 250–300 and partially recovers later; training reward exhibits a similar decline. This decline coincides with shorter episodes in panel (c). Between roughly steps 50 and 300, PTPO

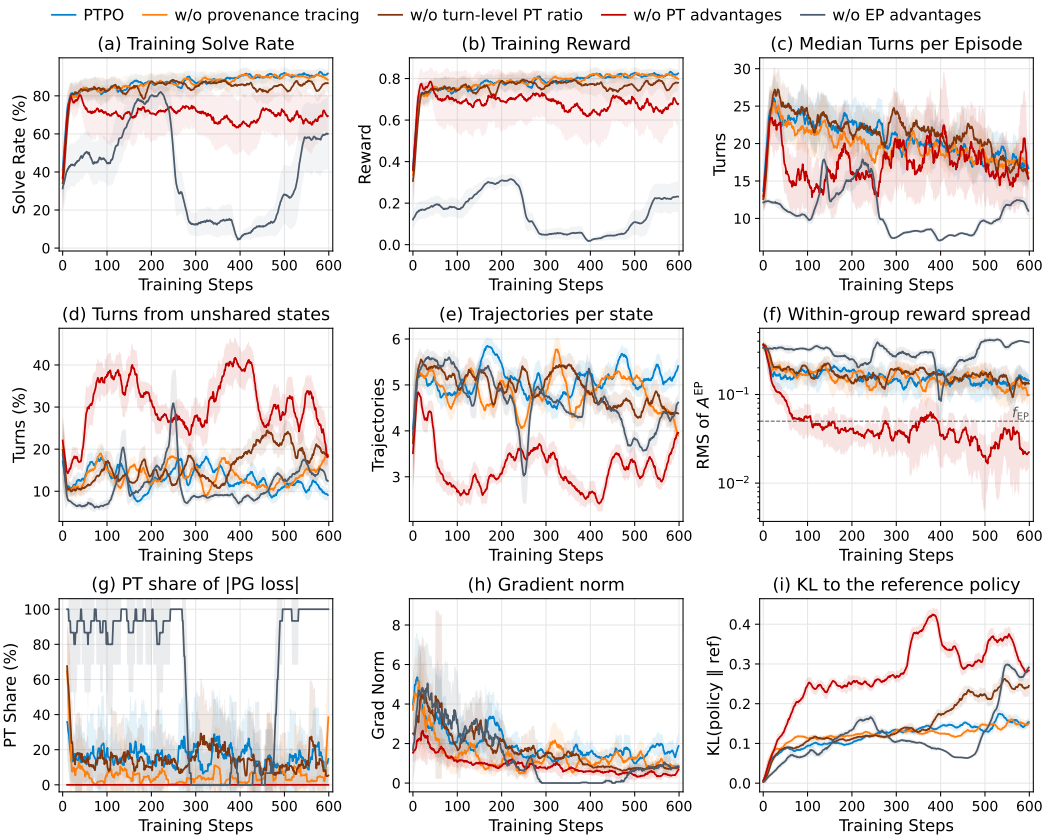


Figure 6: PTPO and ablation training dynamics (15-step moving averages). (a–c) Solve rate, reconstructed reward (solve rate times mean solved-trajectory reward), and median episode turns across all groups before dynamic sampling. (d,e) Among eight trajectories per target: fraction of turns with unshared source states and mean trajectories per state; states use search-graph and five-turn budget-bucket keys, distinct from Section 3.2 matched-context groups. (f) Training-batch reward spread. (g) PT absolute policy-gradient loss share; (h) gradient norm; (i) low-variance reference-policy KL estimate. (g–i) average two optimizer steps per training step. Bands: (a,b) bounds from logged group outcome counts; (c–i) ± 1 s.d. within the smoothing window.

runs longer episodes than the variant without PT advantages (median 22.5 vs. 16.0 turns); later the two become similar.

Panels (d) and (e) measure how often trajectories on the same target reach the same search state. PTPO keeps about 11–14% of turns in states that no other trajectory reached and about five trajectories per state, whereas without PT advantages 25–31% of turns start from unshared states and about three trajectories share each state. Panel (f) shows the diagnostic within-group reward spread, $\mathrm{RMS}(A^{\mathrm{EP}})$. Without PT advantages, it falls below the EP floor f_{EP} by about step 100 and remains below it in most later steps, matching Figure 3(a). For variants retaining both channels, it remains around 0.13–0.14. Without EP advantages, the plotted reward spread remains mostly above the floor despite the performance decline. Thus, persistent reward variation does not by itself ensure sustained task improvement or establish the magnitude of PT gradients.

Panels (g–i) examine optimization dynamics. The PT share of the absolute policy-gradient loss persists in PTPO (about 15% late in training) but is lower without provenance tracing (about 5%). Without EP advantages, this share approaches 100% during substantial portions of training, but falls near zero during much of the low-performing middle phase. Gradient norms also approach zero during this phase and increase again later. These loss shares describe relative contributions, not gradient magnitudes. The variant without the turn-level PT ratio shows greater late-stage KL divergence than PTPO, while removing PT advantages produces higher divergence over most of

training. The EP-disabled variant also exhibits a pronounced late increase in KL. Overall, these runs exhibit distinct failure patterns: diminished EP learning signals with a lower reward plateau without PT, and a marked performance decline despite persistent reward variation, accompanied by weak gradient activity, without EP.

F.2 Search Tree Structure

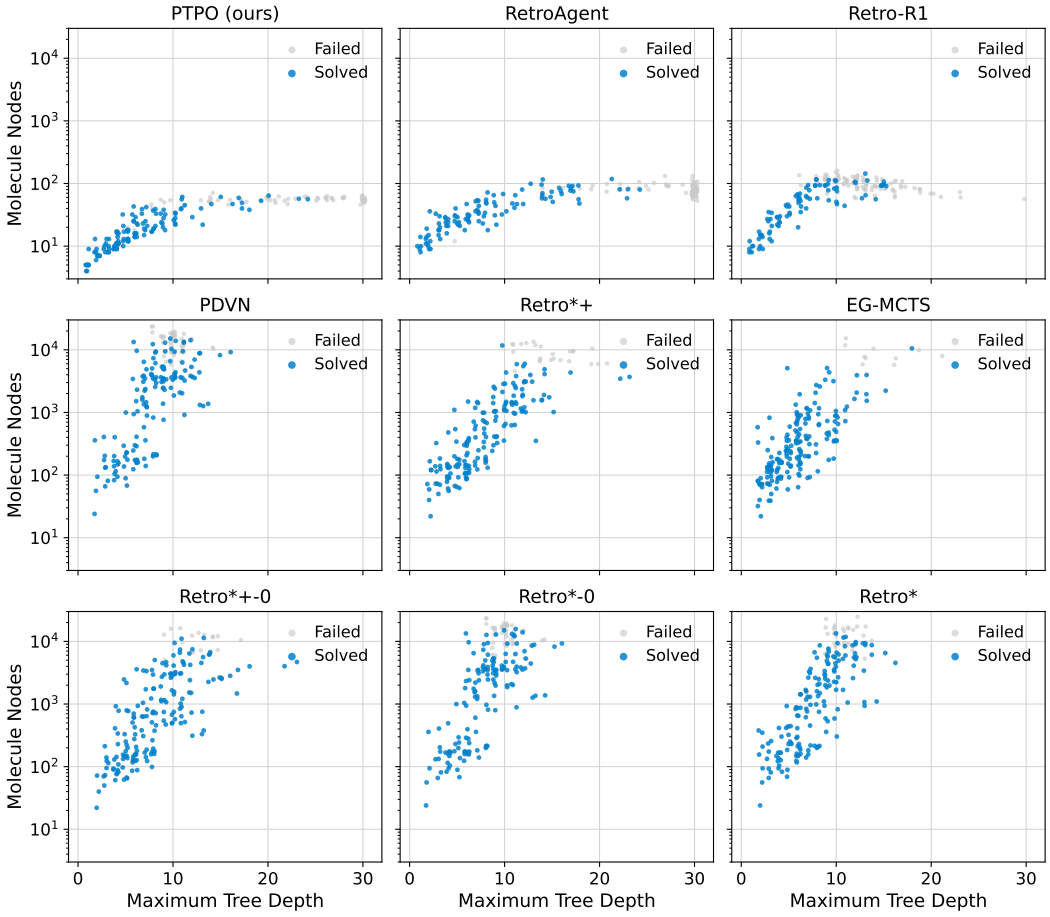


Figure 7: Search tree structures of PTPO and baselines on USPTO-190. Blue and gray points denote solved and failed runs, respectively. The vertical axis uses a logarithmic scale.

Figure 7 compares the search tree structures of PTPO and eight baselines, where each point represents a search run with its maximum explored depth and total number of molecule nodes. Blue and gray points indicate solved and failed runs, respectively, with node counts plotted on a logarithmic scale. Comparing across methods reveals three key findings.

(i) LLM-based planners construct substantially smaller search trees. As shown in the top row, PTPO, RetroAgent, and Retro-R1 generally explore fewer than 200 molecule nodes. In contrast, PDVN, EG-MCTS, and the Retro* variants frequently construct trees containing hundreds to thousands of nodes, with some runs reaching 10^4 or more. This separation is also visible across overlapping depth ranges, showing that methods reaching comparable maximum depths can differ substantially in search tree size.

(ii) PTPO exhibits more compact successful searches than other LLM-based planners. Solved runs in the PTPO panel are concentrated at shallow depths with only tens of molecule nodes, while the successful-search distributions of RetroAgent and Retro-R1 extend toward higher node counts over overlapping depth ranges. PTPO also solves some runs whose maximum explored depth ex-

ceeds 20, showing that compact search can accommodate deeper exploration when needed. These patterns are consistent with selective reaction exploration guided by structured search feedback.

(iii) Greater search depth or tree size alone does not guarantee success. The failed PTPO and RetroAgent runs extend to depths near 30 while retaining relatively small node counts. For several conventional search baselines, failed runs instead occur at moderate depths with approximately 10^4 molecule nodes. These contrasting patterns show that unsuccessful planning can involve either deep exploration or extensive branching.

These results show that PTPO achieves successful planning with substantially smaller search trees than conventional search methods and generally fewer molecule nodes than the LLM-based RetroAgent and Retro-R1, while retaining the capacity for deeper exploration.

F.3 Same-Backbone Comparison

Table 4: Same-backbone comparison on USPTO-190. Best reported values are in **bold**.

Method	Base model	Pass@1	$N = 50$	$N = 100$	$N = 200$
Base model (no RL)	`Qwen3-4B-Instruct-2507`	$6.10_{\pm 2.10}$	$18.80_{\pm 1.60}$	$18.90_{\pm 1.60}$	$18.90_{\pm 1.60}$
Retro-R1V1	`Qwen3-4B-Instruct-2507`	$45.05_{\pm 0.29}$	$50.63_{\pm 3.08}$	$60.32_{\pm 1.96}$	$71.37_{\pm 3.10}$
RetroAgent	`Qwen3-4B-Instruct-2507`	$53.26_{\pm 0.69}$	$58.42_{\pm 1.34}$	$70.53_{\pm 1.93}$	$78.63_{\pm 1.27}$
PTPO	`Qwen3-4B-Instruct-2507`	$\mathbf{67.89}_{\pm 1.90}$	$\mathbf{70.74}_{\pm 1.42}$	$\mathbf{80.32}_{\pm 1.09}$	$\mathbf{84.74}_{\pm 1.34}$

Table 4 examines whether PTPO's gains persist when the LLM backbone is held fixed, controlling for backbone identity and parameter count. All methods use `Qwen3-4B-Instruct-2507`, including the base model without RL and the RL-trained Retro-R1 and RetroAgent baselines. With the shared 4B backbone, PTPO achieves a Pass@1 of $67.89 \pm 1.90\%$, compared with $6.10 \pm 2.10\%$ without RL, and delivers relative improvements of 50.70% and 27.47% over Retro-R1 and RetroAgent, respectively. It also leads all same-backbone baselines at every reported search budget, exceeding RetroAgent by 12.32, 9.79, and 6.11 percentage points at $N = 50$, 100, and 200, respectively. These results show that PTPO's advantage persists with the backbone held fixed and against agents already trained with RL.

F.4 Analysis of Rollout Termination Modes

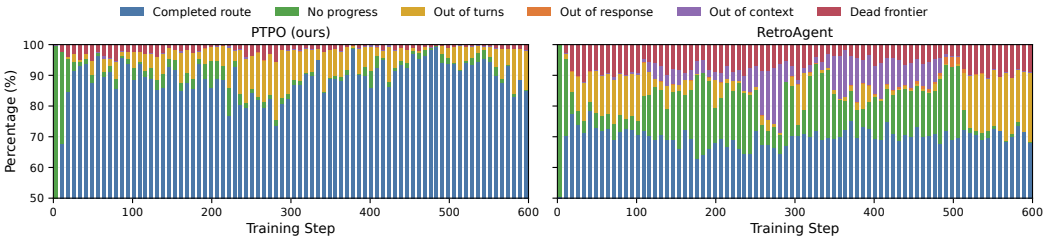


Figure 8: Rollout termination modes across training.

Figure 8 compares the distributions of episode termination modes for PTPO and RetroAgent throughout training. PTPO maintains a higher proportion of completed routes across most checkpoints, reaching approximately 85% at the final displayed checkpoint compared with 68% for RetroAgent—a gap of about 17 percentage points. This improvement is accompanied by fewer *No progress* and *Dead frontier* outcomes and consistently rare *Out of context* terminations. In contrast, RetroAgent exhibits more frequent stalled searches and substantial context-budget exhaustion during parts of training. The predominant remaining failure mode for PTPO is *Out of turns*, indicating that its unsuccessful episodes more often exhaust the interaction budget than stall or exceed the context limit. Overall, the higher completion rate and reduced incidence of these failure modes support the effectiveness of PTPO in completing multi-step retrosynthetic plans under constrained interaction budgets.

F.5 Route Length

Table 5 compares the route-length distributions of PTPO and RetroAgent across all 190 USPTO-190 targets, with 0 denoting an unsolved target. PTPO solves 164 targets versus 160 for RetroAgent. Among solved targets, PTPO achieves median and mean route lengths of 7 and 8.24, versus 9 and 10.62 for RetroAgent. The difference is particularly pronounced in the long-route tail: PTPO returns routes exceeding 15 steps for 15 targets versus 43 for RetroAgent, a 65.1% reduction. Meanwhile, the 4–6-step bin contains 53 PTPO targets versus 35 RetroAgent targets. PTPO thus combines slightly higher target coverage with fewer very long routes.

Table 5: Route-length distributions on USPTO-190. Each bin reports the number of target molecules among all 190 targets; 0 denotes an unsolved target. Median and mean route lengths are reported over solved targets only, excluding zeros.

	Route length (steps)							Solved targets	
Method	0	1–3	4–6	7–9	10–12	13–15	> 15	Median	Mean
PTPO	26	22	53	41	22	11	15	**7**	**8.24**
RetroAgent	30	19	35	31	18	14	43	9	10.62

F.6 Template Overlap

Table 6: Reaction-template overlap for 155 USPTO-190 targets solved by both planners. Bins count target molecules; bold counts mark modal bins. For fingerprint sets X (PTPO) and Y (RetroAgent), Jaccard similarity is $|X \cap Y|/|X \cup Y|$; containment, $|X \cap Y|/\min(|X|, |Y|)$, normalizes by the smaller fingerprint, addressing size asymmetry (medians: 8 versus 12 templates) that can lower Jaccard similarity.

	Template overlap							Summary	
Metric	0	$(0, 0.1]$	$(0.1, 0.25]$	$(0.25, 0.5]$	$(0.5, 0.75]$	$(0.75, 1)$	1	Median	Mean
Jaccard	14	14	65	61	1	0	0	0.211	0.218
Containment	14	0	13	56	49	17	6	0.500	0.498

Table 6 compares reaction-template overlap on the 155 targets solved by both planners. The median and mean Jaccard similarities are 0.211 and 0.218, respectively, with 93 targets (60.0%) having similarity at most 0.25. However, the median fingerprint sizes differ: 8 templates for PTPO and 12 for RetroAgent. Because this size asymmetry can lower Jaccard similarity, we also report containment, which normalizes the shared templates by the smaller fingerprint. Its median is 0.500 and its mean is 0.498, indicating that, at the median, half of the smaller fingerprint is shared. Fourteen targets have no shared templates, while complete containment occurs for only six targets and identical fingerprints for none. These results reveal differences in reaction-template composition between the planners beyond fingerprint-size asymmetry.

F.7 Terminal Outcomes versus Reaction-Level Certification

We test whether a trajectory's terminal outcome reflects the quality of its committed reactions using existing USPTO-190 evaluation rollouts of PTPO and RetroAgent (one run each, at most 120 turns per rollout), with no additional runs. A *committed reaction* is a reaction node created by a successful `expand`; repeated expansions are rejected and create none. A committed reaction is *certified* if its final status is SOLVED, i.e., every precursor recursively reaches the building-block set, and *uncertified* if it remains OPEN at rollout end. An uncertified reaction is not necessarily poor: its precursors may remain unexplored, illustrating the downstream-exploration dependence that terminal feedback cannot separate. We condition on outcome because long rollouts are mostly failures, and pooling would let the solved/failed mix drive the trend.

Figure 9(a) shows frequent disagreement between terminal outcomes and reaction-level certification. Among solved rollouts, 38.4% (PTPO) and 99.4% (RetroAgent) commit at least one never-certified

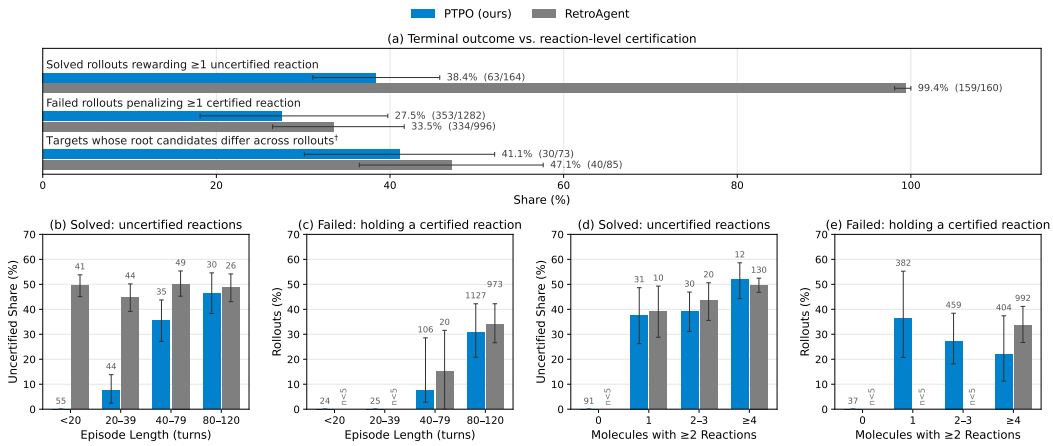


Figure 9: Terminal outcome versus reaction-level certification on USPTO-190 evaluation rollouts. (a) Overall disagreement: solved rollouts with a never-certified reaction sharing success; failed rollouts with a fully certified reaction sharing failure; and †targets with at least two rollouts whose first query returned different candidate or building-block-candidate counts (equal counts can hide different sets, so this is a lower bound). (b–e) Disagreement by episode length (b, c) and branching (d, e): (b, d) mean share of uncertified committed reactions in solved rollouts; (c, e) share of failed rollouts with a certified reaction. Branching counts molecules with at least two committed reactions; in (d), the share is zero without branching. Error bars show 95% bootstrap intervals over targets; numbers give rollout counts; bins with fewer than five rollouts are empty.

reaction that shares the success; these reactions constitute 32.4% and 50.4% of all reactions committed in solved rollouts. Among failed rollouts, 27.5% and 33.5% contain at least one fully certified reaction that shares the failure. Candidate availability also varies independently of agent choices: for at least 41.1% and 47.1% of targets with multiple rollouts, the first query returns different candidate sets across rollouts because the environment shuffles and truncates predictor proposals.

Panels (b–e) relate these disagreements to episode length and branching. In PTPO's solved rollouts, the uncertified share rises from 0% below 20 turns to 46.3% at 80–120 turns (panel b); RetroAgent's solved rollouts all branch and remain at 45–50% across lengths. In failed rollouts, the share containing a certified reaction rises with length for both systems, reaching 30.6% (PTPO) and 34.0% (RetroAgent) at 80–120 turns (panel c). After branching, the uncertified share in solved rollouts tends to rise from about 38–39% with one branching molecule to about 50% with four or more (panel d). Failed rollouts show no increasing trend with branching (panel e). Episode length correlates with target difficulty, so these trends cannot be attributed to length alone.

G Prompts Used in PTPO

The prompts we use for PTPO agents are presented in Figure 10 and Figure 11. These prompt templates are constructed using Python-style string formatting, where placeholders enclosed in curly braces ({}) represent semantic slots. These placeholders, such as {target_smiles} and {max_route_depth}, are dynamically populated at runtime through Python formatted string literals. The corresponding tool schemas are incorporated into the prompt through the Qwen3 chat template.

H Tool Environment Used in PTPO

The tool interface used by PTPO agents is presented in Figure 12. The <tool_call> </tool_call> block instructs the agent to issue an executable planning action by specifying a tool name and its arguments in JSON format, whereas the <tool_response> </tool_response> block encloses the corresponding observation returned by the planning environment. The agent requests candidate reactions through `predict_templates`, commits one

System Prompt Template for PTPO

You are an expert retrosynthesis planner. Decompose a target molecule into commercially available building blocks through valid reactions.
You search over an AND–OR graph. Molecules (M1, M2, ...) are OR nodes: a molecule can be decomposed in multiple ways, and any one successful decomposition suffices. Reactions (R1, R2, ...) are AND nodes: all reactants must be solved for the reaction to succeed. Status propagates automatically up the tree.
Core loop:
1. `predict_templates(molecule_id)`: get candidate reactions (P1, P2, ...).
2. `expand(molecule_id, prediction_id)`: apply predictions to the graph. Use comma-separated IDs (e.g., "P1, P3") to expand multiple at once.
3. `get_frontier()`: see which OPEN molecules still need decomposition.
4. Repeat until the route is solved (automatic detection).
Maximum route depth: {max_route_depth} reaction steps.

Figure 10: The system prompt template used for PTPO agents.

User Prompt Template for PTPO

Plan a retrosynthesis route for: {target_smiles}. The target is M1. The graph tools (`predict_templates`, `expand`, and `inspect_node`) use molecule IDs (M1, M2, ...). The chemistry tools (`get_molecule_info`, `search_building_blocks`, `calculate_synthetic_accessibility`, and `calculate_scscore`) use SMILES strings. Start with `predict_templates("M1")`.

Figure 11: The user prompt template used for PTPO agents.

or more selected decompositions through `expand`, and examines unresolved molecules through `get_frontier` or `inspect_node`. Auxiliary chemistry tools operate directly on SMILES strings and provide building-block availability and molecular-complexity estimates. Multiple planning actions may be emitted as separate `<tool_call>` blocks within the same turn; all calls are generated before their `<tool_response>` observations become available.

I Multi-turn Interaction Experiments

In this section, we present the prompts, tool interface, and representative interactions between the PTPO agent and the retrosynthesis environment. Figure 11 gives the user prompt used to initialize the planning task, while Figure 12 describes the available tools and the agent–environment communication protocol. Figure 13 presents a successful interaction (Case A) for synthesizing `CC(C)c1ccc(OC2CCC3(CC2)N[C@@H](C(=O)O)C(C)(C)S3)cc1`, where the agent completes the route through two successive reaction selections. Figure 14 presents another successful interaction (Case B) for `O=C1c2cc(OCCC3CCN(Cc4ccc(C5=NCCN5)cc4)CC3)ccc2CCCN1Cc1ccccc1`. The agent completes a four-step route over 11 interaction turns using four predictor calls. Case C in Figure 15 illustrates an unsuccessful interaction that terminates after 13 turns under the no-progress guard. The agent initially selects the only reaction containing a purchasable precursor among 19 candidates. At `M3`, however, it selects a methyl-ester precursor from 16 candidates, despite two alternatives containing building-block precursors. A subsequent selection replaces the methyl ester with an ethyl ester while preserving the main scaffold, after which repeated expansions and queries yield no progress. This case illustrates the limited guidance that terminal-only feedback provides about where progress was lost. PTPO supplements this outcome with comparisons of sampled reaction selections under matched candidate views, using terminal precursor assessments to derive local advantages.

Tool Environment for PTPO

```
# Tools

You may call one or more functions to assist with retrosynthesis
planning. Function signatures are provided within <tools></tools>
XML tags:

<tools>
{"type":"function","function":{"name":"predict_templates","description":"Predict one-
    step retrosynthesis decompositions for a molecule node.","parameters":{"type":"
    object","properties":{"molecule_id":{"type":"string","description":"Molecule node
     ID to decompose, e.g., M1."}},"required":["molecule_id"]}}}

{"type":"function","function":{"name":"expand","description":"Expand a molecule node
    using one or more predictions returned by predict_templates.","parameters":{"type
    ":"object","properties":{"molecule_id":{"type":"string","description":"Molecule
    node ID to expand, e.g., M1."},"prediction_id":{"type":"string","description":"
    One or more prediction IDs, e.g., P1 or P1,P3."}},"required":["molecule_id","
    prediction_id"]}}}

{"type":"function","function":{"name":"get_frontier","description":"Return OPEN leaf
    molecules that still require decomposition.","parameters":{"type":"object","
    properties":{},"required":[]}}}

{"type":"function","function":{"name":"inspect_node","description":"Return full
    structural details for a molecule or reaction node.","parameters":{"type":"object
    ","properties":{"node_id":{"type":"string","description":"Node ID to inspect, e.g
    ., M2 or R1."}},"required":["node_id"]}}}

{"type":"function","function":{"name":"get_molecule_info","description":"Return basic
    molecular information for a list of SMILES strings.","parameters":{"type":"object
    ","properties":{"molecules":{"type":"array","items":{"type":"string"},"
    description":"SMILES strings to analyze."}},"required":["molecules"]}}}

{"type":"function","function":{"name":"search_building_blocks","description":"Check
    whether molecules occur in the available building-block collection.","parameters
    ":{"type":"object","properties":{"molecules":{"type":"array","items":{"type":"
    string"},"description":"SMILES strings to search for."}},"required":["molecules
    "]}}}

{"type":"function","function":{"name":"calculate_synthetic_accessibility","description
    ":"Calculate synthetic-accessibility scores from 1 to 10; lower is easier.","
    parameters":{"type":"object","properties":{"molecules":{"type":"array","items":{"
    type":"string"},"description":"SMILES strings to score."}},"required":["molecules
    "]}}}

{"type":"function","function":{"name":"calculate_scscore","description":"Calculate
    synthetic-complexity scores from 1 to 5; lower is easier.","parameters":{"type":"
    object","properties":{"molecules":{"type":"array","items":{"type":"string"},"
    description":"SMILES strings to score."}},"required":["molecules"]}}}
</tools>

For each function call, return a JSON object containing the function
name and arguments within <tool_call></tool_call> XML tags:

<tool_call>
{"name":"<function-name>","arguments":<args-json-object>}
</tool_call>

Multiple functions may be called in the same turn by emitting separate
<tool_call></tool_call> blocks. Environment observations are returned
within <tool_response></tool_response> XML tags.
```

Figure 12: Tool definitions and invocation protocol for PTPO agents. Graph tools take molecule or reaction node identifiers; auxiliary chemistry tools take SMILES strings directly.

PTPO Agent–Environment Interaction Case A

User:
Plan a retrosynthesis route for the target molecule M1. Query candidate reactions with predict_templates, commit a selected decomposition with expand, and inspect unresolved molecules with get_frontier. Continue until all terminal precursors are available building blocks.

Agent: <tool_call>
{"name":"predict_templates","arguments":{"molecule_id":"M1"}}
</tool_call>
ENV: <tool_response>
{"molecule_id":"M1","predictions":{
"P1":"CC(C)c1ccc(OC2CCC3(CC2)N[C@@H](C(=O)N2C(=O)OC[C@H]2Cc2ccccc2)C(C)(C)S3)cc1 + [OH-]*",
"P2":"CC(C)c1ccc(OC2CCC3(CC2)N[C@@H](C#N)C(C)(C)S3)cc1 + CO* + [OH-]*",
"P3":"CC(C)c1ccc(O)cc1* + CC1(C)SC2(CCC(O)CC2)N[C@H]1C(=O)O",
"P5":"CC(C)(S)[C@@H](N)C(=O)O* + CC(C)c1ccc(OC2CCC(=O)CC2)cc1 <- selected",
"...":"19 predictions total; 12 contain a building-block precursor (*); 4 contain only purchasable precursors"}}
</tool_response>

Agent: <tool_call>
{"name":"expand","arguments":{"molecule_id":"M1","prediction_id":"P5"}}
</tool_call>
ENV: <tool_response>
{"new_molecules":["CC(C)(S)[C@@H](N)C(=O)O",
"CC(C)c1ccc(OC2CCC(=O)CC2)cc1"]}
</tool_response>

Agent: <tool_call>
{"name":"get_frontier","arguments":{}}
</tool_call>
ENV: <tool_response>
{"M3":"CC(C)c1ccc(OC2CCC(=O)CC2)cc1"}
</tool_response>

Agent: <tool_call>
{"name":"predict_templates","arguments":{"molecule_id":"M3"}}
</tool_call>
ENV: <tool_response>
{"molecule_id":"M3","predictions":{
"P1":"CC(C)c1ccc(O)cc1* + O=C1CCC(O)CC1* <- selected",
"P2":"O=C1CCC(O)CC1* + CC(C)c1ccc(Br)cc1*",
"P3":"CC(C)c1ccc(OC2C=CC(=O)CC2)cc1",
"...":"20 predictions total; 6 contain a building-block precursor (*); 5 contain only purchasable precursors"}}
</tool_response>

Agent: <tool_call>
{"name":"expand","arguments":{"molecule_id":"M3","prediction_id":"P1"}}
</tool_call>
ENV: <tool_response>
{"new_molecules":["CC(C)c1ccc(O)cc1","O=C1CCC(O)CC1"],
"route_solved":true}
</tool_response>

Figure 13: The example PTPO-agent interaction with the retrosynthesis environment. The target molecule M1 is CC(C)c1ccc(OC2CCC3(CC2)N[C@@H](C(=O)O)C(C)(C)S3)cc1.

PTPO Agent–Environment Interaction Case B

```
User (task summary):
Plan a retrosynthesis route for target M1:
O=C1c2cc(OCCC3CCN(Cc4ccc(C5=NCCN5)cc4)CC3)ccc2CCCN1Cc1ccccc1

Agent: <tool_call> {"name":"predict_templates","arguments":{"molecule_id":"M1"}} </tool_call>
ENV: <tool_response> {"molecule_id":"M1","predictions":{
"P1":"O=C1c2cc(O)ccc2CCCN1Cc1ccccc1 + CS(=O)(=O)OCCC1CCN(Cc2ccc(C3=NCCN3)cc2)CC1",
"P2":"O=C1CNC(c2ccc(CN3CCC(CCOc4ccc5c(c4)C(=O)N(Cc4ccccc4)CCC5)CC3)cc2)=N1",
"P3":"O=C1c2c(ccc(OCCC3CCN(Cc4ccc(C5=NCCN5)cc4)CC3)c2F)CCCN1Cc1ccccc1",
"P16":"O=C1c2cc(F)ccc2CCCN1Cc1ccccc1 + OCCC1CCN(Cc2ccc(C3=NCCN3)cc2)CC1 <- selected",
"...":"20 predictions total; 2 contain a building-block precursor (*)"
}} </tool_response>

Agent: <tool_call> {"name":"expand","arguments":{"molecule_id":"M1","prediction_id":"P16"}}
     </tool_call>
ENV: <tool_response> {"new_molecules":["O=C1c2cc(F)ccc2CCCN1Cc1ccccc1",
"OCCC1CCN(Cc2ccc(C3=NCCN3)cc2)CC1"]} </tool_response>

Agent: <tool_call> {"name":"get_frontier","arguments":{}} </tool_call>
ENV: <tool_response> {"M3":"OCCC1CCN(Cc2ccc(C3=NCCN3)cc2)CC1",
"M2":"O=C1c2cc(F)ccc2CCCN1Cc1ccccc1"} </tool_response>

Agent: <tool_call> {"name":"predict_templates","arguments":{"molecule_id":"M2"}} </tool_call>
ENV: <tool_response> {"molecule_id":"M2","predictions":{
"P1":"O=C1c2cc(F)ccc2CCCN1Cc1cccc(F)c1",
"P2":"O=C1c2cc(F)ccc2C=CCN1Cc1ccccc1",
"P3":"O=C1NCCCc2ccc(F)cc21* + ClCc1ccccc1* <- selected",
"...":"20 predictions total; 4 contain a building-block precursor (*); 3 contain at least two
     "
}} </tool_response>

Agent: <tool_call> {"name":"expand","arguments":{"molecule_id":"M2","prediction_id":"P3"}} </
     tool_call>
ENV: <tool_response> {"new_molecules":["O=C1NCCCc2ccc(F)cc21","ClCc1ccccc1"]} </tool_response
     >

Agent: <tool_call> {"name":"get_frontier","arguments":{}} </tool_call>
ENV: <tool_response> {"M3":"OCCC1CCN(Cc2ccc(C3=NCCN3)cc2)CC1"} </tool_response>

Agent: <tool_call> {"name":"predict_templates","arguments":{"molecule_id":"M3"}} </tool_call>
ENV: <tool_response> {"molecule_id":"M3","predictions":{
"P1":"c1cc(C2=NCCN2)ccc1CN1CCC(CCOC2CCCCO2)CC1",
"P2":"OCCC1CCNCC1* + BrCc1ccc(C2=NCCN2)cc1",
"P3":"O=C(O)CC1CCN(Cc2ccc(C3=NCCN3)cc2)CC1",
"P11":"O=Cc1ccc(C2=NCCN2)cc1 + CC(C)(C)OC(=O)N1CCC(CCO)CC1* <- selected",
"...":"20 predictions total; 5 contain a building-block precursor (*)"
}} </tool_response>

Agent: <tool_call> {"name":"expand","arguments":{"molecule_id":"M3","prediction_id":"P11"}}
     </tool_call>
ENV: <tool_response> {"new_molecules":["O=Cc1ccc(C2=NCCN2)cc1",
"CC(C)(C)OC(=O)N1CCC(CCO)CC1"]} </tool_response>

Agent: <tool_call> {"name":"get_frontier","arguments":{}} </tool_call>
ENV: <tool_response> {"M6":"O=Cc1ccc(C2=NCCN2)cc1"} </tool_response>

Agent: <tool_call> {"name":"predict_templates","arguments":{"molecule_id":"M6"}} </tool_call>
ENV: <tool_response> {"molecule_id":"M6","predictions":{
"P1":"O=CO* + N#Cc1ccc(C2=NCCN2)cc1*",
"P2":"CCOC(OCC)c1ccc(C2=NCCN2)cc1",
"P3":"CO* + N#Cc1ccc(C2=NCCN2)cc1*",
"P18":"N#Cc1ccc(C=O)cc1* + NCCN* <- selected",
"...":"20 predictions total; 13 contain a building-block precursor (*); 9 contain at least
     two"
}} </tool_response>

Agent: <tool_call> {"name":"expand","arguments":{"molecule_id":"M6","prediction_id":"P18"}}
     </tool_call>
ENV: <tool_response> {"new_molecules":["N#Cc1ccc(C=O)cc1","NCCN"],"route_solved":true} </
     tool_response>
```

Figure 14: The example of a PTPO-agent interaction with the retrosynthesis environment. The target molecule `M1` is `O=C1c2cc(OCCC3CCN(Cc4ccc(C5=NCCN5)cc4)CC3)ccc2CCCN1Cc1ccccc1`.

PTPO Agent–Environment Interaction Case C (failed episode)

```
User (task summary):
Plan a retrosynthesis route for target M1:
CC1=NC(C)=C(C(=O)OCN2C(=O)c3ccccc3S2(=O)=O)C(c2ccccc2Cl)C1=C(O)OCC1CCCO1

Agent: <tool_call> {"name":"predict_templates","arguments":{"molecule_id":"M1"}} </tool_call>
ENV: <tool_response> {"molecule_id":"M1","predictions":{
"P1":"CC1=NC(C)=C(C(=O)OCNC(=O)c2ccccc2S(=O)(=O)Cl)C(c2ccccc2Cl)C1=C(O)OCC1CCCO1",
"P2":"CC1=NC(C)=C(C(=O)OCN2C(=O)c3ccccc3S2(=O)=O)C(c2ccc(F)cc2Cl)C1=C(O)OCC1CCCO1",
"P6":"O=C1c2ccccc2S(=O)(=O)N1CCl* + CC1=NC(C)=C(C(=O)O)C(c2ccccc2Cl)C1=C(O)OCC1CCCO1 <-
     selected",
"...":"19 predictions total; 1 contains a building-block precursor (*)"
}} </tool_response>

Agent: <tool_call> {"name":"expand","arguments":{"molecule_id":"M1","prediction_id":"P6"}} </
     tool_call>
ENV: <tool_response> {"new_molecules":["O=C1c2ccccc2S(=O)(=O)N1CCl",
"CC1=NC(C)=C(C(=O)O)C(c2ccccc2Cl)C1=C(O)OCC1CCCO1"]} </tool_response>

Agent: <tool_call> {"name":"get_frontier","arguments":{}} </tool_call>
ENV: <tool_response> {"M3":"CC1=NC(C)=C(C(=O)O)C(c2ccccc2Cl)C1=C(O)OCC1CCCO1"} </
     tool_response>

Agent: <tool_call> {"name":"predict_templates","arguments":{"molecule_id":"M3"}} </tool_call>
ENV: <tool_response> {"molecule_id":"M3","predictions":{
"P1":"CC1=NC(C)=C(C(=O)OCCC#N)C(c2ccccc2Cl)C1=C(O)OCC1CCCO1",
"P2":"COc1ccc(C2C(C(=O)O)=C(C)N=C(C)C2=C(O)OCC2CCCO2)c(C)c1 + Cc1cccc(Cl)c1S(=O)(=O)Cl*",
"P3":"COC(=O)C1=C(C)N=C(C)C(=C(O)OCC2CCCO2)C1c1ccccc1Cl <- selected",
"...":"16 predictions total; 2 contain a building-block precursor (*)"
}} </tool_response>

Agent: <tool_call> {"name":"expand","arguments":{"molecule_id":"M3","prediction_id":"P3"}} </
     tool_call>
ENV: <tool_response> {"new_molecules":["COC(=O)C1=C(C)N=C(C)C(=C(O)OCC2CCCO2)C1c1ccccc1Cl"]}
     </tool_response>

Agent: <tool_call> {"name":"get_frontier","arguments":{}} </tool_call>
ENV: <tool_response> {"M4":"COC(=O)C1=C(C)N=C(C)C(=C(O)OCC2CCCO2)C1c1ccccc1Cl"} </
     tool_response>

Agent: <tool_call> {"name":"predict_templates","arguments":{"molecule_id":"M4"}} </tool_call>
ENV: <tool_response> {"molecule_id":"M4","predictions":{
"P1":"COC(=O)C1=C(C)N=C(C)C(=C(O)OCC2CCCO2)C1c1cc(Cl)ccc1Cl",
"P2":"COC(=O)C1=C(C)N=C(C)C(=C(O)OCC2CCCO2)C1c1ccc(F)cc1Cl",
"P4":"CCOC(=O)C1=C(C)N=C(C)C(=C(O)OCC2CCCO2)C1c1ccccc1Cl <- selected",
"...":"9 predictions total; 0 contain a building-block precursor (*)"
}} </tool_response>

Agent: <tool_call> {"name":"expand","arguments":{"molecule_id":"M4","prediction_id":"P4"}} </
     tool_call>
ENV: <tool_response> {"new_molecules":["CCOC(=O)C1=C(C)N=C(C)C(=C(O)OCC2CCCO2)C1c1ccccc1Cl"]}
      </tool_response>

Agent: <tool_call> {"name":"expand","arguments":{"molecule_id":"M3","prediction_id":"P3"}} </
     tool_call>
ENV: <tool_response> {"new_molecules":[]} </tool_response>

[ 4 further turns, abridged: the agent re-queries M1 -- whose pool has been
  filtered from 19 to 18 candidates, none of them now carrying a building
  block -- and re-issues expansions already committed, each returning an
  empty molecule list. One turn produces no parseable tool call and is
  therefore absent from the call log. Five consecutive turns without a
  successful expansion trigger the no-progress guard. ]

TERMINATED: no_progress after 13 turns; 0 of 3 committed reactions reach a
building block on the unresolved branch.
```

Figure 15: The example of a PTPO-agent interaction with the retrosynthesis environment. The target molecule `M1` is `CC1=NC(C)=C(C(=O)OCN2C(=O)c3ccccc3S2(=O)=O)C(c2ccccc2Cl)C1=C(O)OCC1CCCO1`.

J CASE STUDY

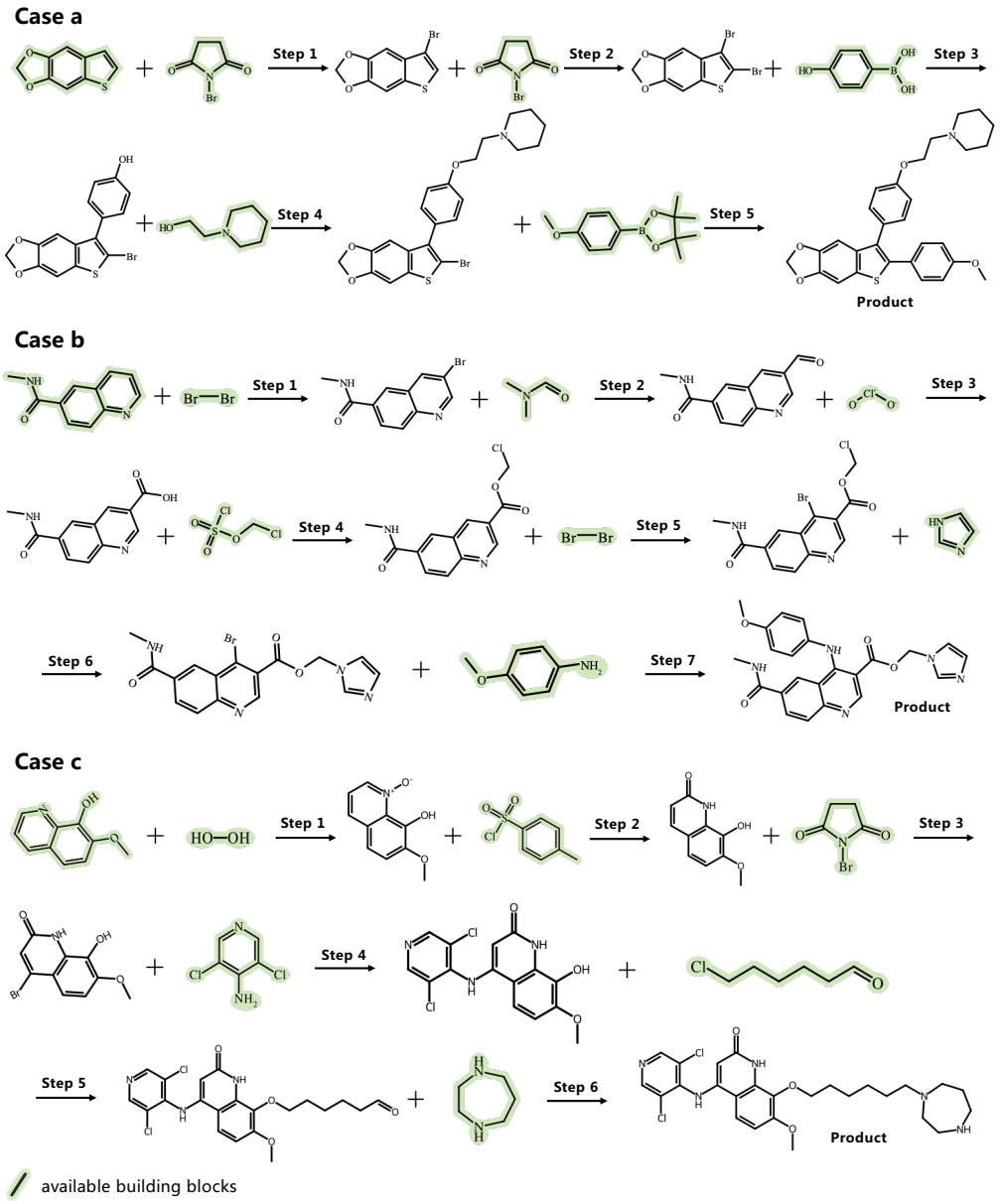


Figure 16: Synthesis routes found by PTPO for target molecules. Routes are presented in the forward synthetic direction. Available building blocks are highlighted in green.

This work aims to support drug discovery through multi-step retrosynthetic planning. To illustrate the routes identified by PTPO for structurally complex targets, we examine three molecules featuring benzothiophene, quinoline, and quinolin-2-one scaffolds, respectively. PTPO performs retrosynthetic planning from target molecules to available building blocks. For ease of interpretation, Figure 16 displays the resulting routes in the forward synthetic direction, from building blocks to targets. Building blocks designated as available in the planning environment are highlighted in green.

Case a PTPO identifies a five-step synthesis route. Steps 1 and 2 are successive brominations of the benzothiophene scaffold using *N*-bromosuccinimide. Step 3 is a Suzuki–Miyaura coupling

that introduces a 4-hydroxyphenyl group. Step 4 forms an ether linkage that installs the piperidine-containing side chain. Step 5 is a second Suzuki–Miyaura coupling at the remaining brominated position, introducing a 4-methoxyphenyl group to obtain the target molecule. This route combines scaffold functionalization with stepwise fragment assembly.

Case b PTPO identifies a seven-step synthesis route. Step 1 brominates the quinoline scaffold. Step 2 replaces the bromine substituent with a formyl group, and Step 3 oxidizes the resulting aldehyde to a carboxylic acid. Step 4 converts the acid into a chloromethyl ester. Step 5 introduces a bromine substituent at a different position on the quinoline ring. In Step 6, *N*-alkylation of imidazole replaces the chlorine atom of the chloromethyl group. Step 7 forms a C–N bond with 4-methoxyaniline at the brominated ring position to obtain the target molecule. Compared with Case a, this route involves a longer sequence of functional-group interconversions before the final coupling step.

Case c PTPO identifies a six-step synthesis route. Step 1 oxidizes the quinoline nitrogen using hydrogen peroxide to form the corresponding *N*-oxide, and Step 2 converts this intermediate into a quinolin-2-one scaffold. Step 3 introduces a bromine substituent using *N*-bromosuccinimide. Step 4 forms a C–N bond with 3,5-dichloropyridin-4-amine. Step 5 installs an aldehyde-terminated alkyl side chain through *O*-alkylation of the phenolic group. Step 6 incorporates 1,4-diazepane through reductive amination of the terminal aldehyde to obtain the target molecule. This route combines heterocyclic scaffold modification with sequential side-chain assembly.

These cases illustrate how PTPO connects individual retrosynthetic disconnections into complete routes whose terminal precursors belong to the available building-block set. They also highlight the dependencies underlying multi-step planning: completing a route requires resolving every precursor requirement introduced by the selected disconnections. The proposed routes remain computational predictions. Experimental validation is needed to establish suitable reaction conditions and assess selectivity, compatibility between successive steps, and overall synthetic feasibility.

K Limitations and Future Work

PTPO learns planning policies within a predefined reaction space and does not directly optimize the underlying reaction predictor. Its local supervision relies on explicit precursor states and candidate provenance, so adapting PTPO to other environments requires corresponding structured feedback. Although credit estimation is lightweight, rollout generation remains the dominant computational expense. Adaptive rollout allocation is therefore a promising direction for improving training efficiency at larger scales. Another direction is to adaptively weight reaction and trace advantages according to the statistical support and uncertainty of the underlying reaction-value estimates.

Extending provenance tracing to agents with evolving memory across planning tasks is promising. PTPO links reaction selections to the tool calls producing their candidates within each rollout. Recent agent memory research explores dynamic memory organization and updating (Xu et al., 2025), as well as information consolidation and retrieval across sessions (Chhikara et al., 2025). Extending PTPO to this setting would require preserving provenance through memory consolidation and retrieval, allowing credit to reach both evidence acquisition and subsequent memory operations.