Retrosynthetic Planning with Dual Value Networks

Guoqing Liu [*1] **Di Xue** [*2] **Shufang Xie** [3] **Yingce Xia** [1] **Austin Tripp** [4]
Krzysztof Maziarz [1] **Marwin Segler** [1] **Tao Qin** [1] **Zongzhang Zhang** [2] **Tie-Yan Liu** [1]

Abstract

Retrosynthesis, which aims to find a route to synthesize a target molecule from commercially available starting materials, is a critical task in drug discovery and materials design. Recently, the combination of ML-based single-step reaction predictors with multi-step planners has led to promising results. However, the single-step predictors are mostly trained offline to optimize the single-step accuracy, without considering complete routes. Here, we leverage reinforcement learning (RL) to improve the single-step predictor, by using a tree-shaped MDP to optimize complete routes. Specifically, we propose a novel online training algorithm, called Planning with Dual Value Networks (PDVN), which alternates between the planning phase and updating phase. In PDVN, we construct two separate value networks to predict the synthesizability and cost of molecules, respectively. To maintain the single-step accuracy, we design a two-branch network structure for the single-step predictor. On the widely-used USPTO dataset, our PDVN algorithm improves the search success rate of existing multi-step planners (e.g., increasing the success rate from 85.79% to 98.95% for Retro*, and reducing the number of model calls by half while solving 99.47% molecules for RetroGraph). Additionally, PDVN helps find shorter synthesis routes (e.g., reducing the average route length from 5.76 to 4.83 for Retro*, and from 5.63 to 4.78 for RetroGraph).

1. Introduction

Retrosynthesis is one of the fundamental problems in organic chemistry, widely used in important applications such as drug discovery and materials design. Given a target molecule, the goal of retrosynthesis is to identify a series of chemically valid reactions starting from the target molecule until reaching commercially available building block molecules in a backward and recursive manner. There are many theoretically-possible transformations that can be applied at each step. In addition, each intermediate molecule could be broken into several reactants in one reaction. As a result, the search space of retrosynthesis is enormous and makes the problem challenging even for experienced chemists.

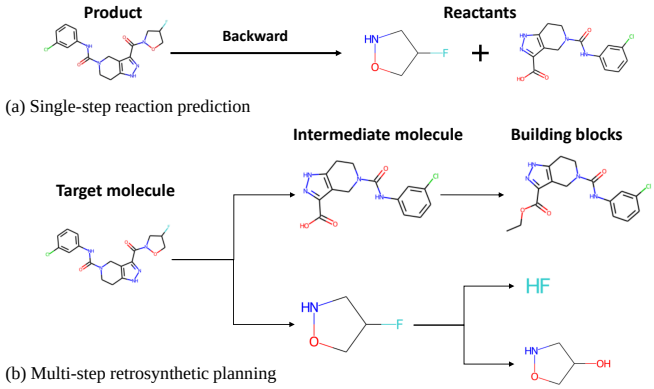

Figure 1. a) The single-step reaction predictor, which predicts potential ways to break a molecule into reactants at each step. b) The multi-step planner, which searches for a complete route by iteratively calling the predictor. The goal of retrosynthesis is to find a synthesis route ending up in the building block molecules for a target molecule.

Retrosynthesis has drawn much attention in the machine learning (ML) community in recent years (Segler et al., 2018; Dong et al., 2021). As shown in Fig. 1, current ML-based retrosynthesis consists of 1) a single-step reaction predictor to predict a set of potential reactions given a molecule; 2) a multi-step planner to search for a complete synthesis route by iteratively calling the predictor. Many algorithms have been proposed for the single-step predictor through supervised learning (SL) based on existing real-world reaction datasets (Lowe, 2012), such as template-based methods (Segler & Waller, 2017; Coley et al., 2017; Dai et al., 2019) and template-free methods (Liu et al., 2017;

[*]Equal contribution [1]Microsoft Research AI4Science [2]National Key Laboratory for Novel Software Technology, Nanjing University [3]Renmin University of China [4]University of Cambridge. Correspondence to: Tao Qin <taoqin@microsoft.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

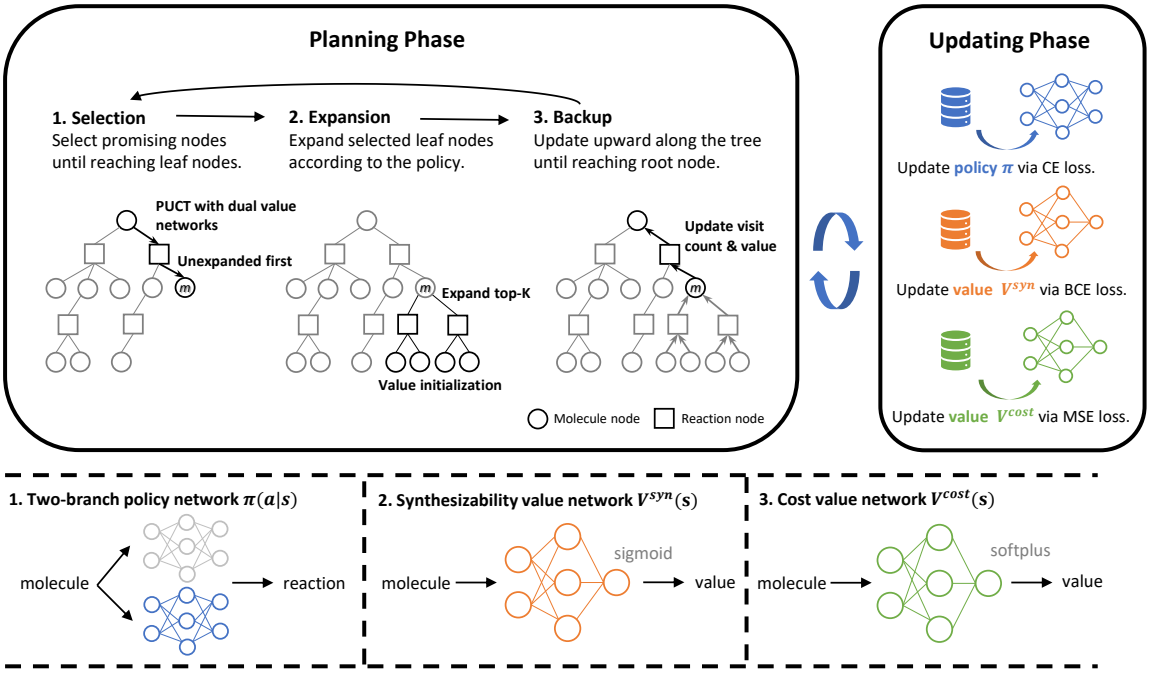

Figure 2. An illustration of our PDVN algorithm. The PDVN algorithm has three modules: 1) a two-branch policy network; 2) a synthesizability value network that predicts if a molecule can be synthesized; 3) a cost value network that predicts the required synthesis cost if synthesizable. PDVN is initialized with an offline SL model and alternates between two phases: 1) *Planning phase*: simulate synthesis experiences on the tree-shaped MDP under the guidance of the policy network and dual value networks. 2) *Updating phase*: extract useful training targets from the generated experiences and update all three networks. Finally, the single-step model trained by PDVN is plugged into popular multi-step planners to enhance their performance.

Tetko et al., 2020). Researchers have also developed several search algorithms for multi-step planning, such as 3N-MCTS (Segler et al., 2018), Retro* (Chen et al., 2020), and RetroGraph (Xie et al., 2022).

However, in most ML-based methods today, the single-step predictor is usually trained offline to optimize the single-step accuracy, without considering complete synthesis routes. In this work, we leverage reinforcement learning (RL) to improve the single-step predictor, or the policy network in the terminology of RL (Sutton & Barto, 2018; Liu et al., 2021), to optimize complete routes. To do this, we use a tree-shaped Markov Decision Process (MDP) to formulate the retrosynthesis problem. Then, we propose a novel online training algorithm, called *Planning with Dual Value Networks (PDVN)*, which alternates between two phases:

1. *Planning phase:* Given a batch of training target molecules, we generate the simulated experiences by planning with the dual value networks.
2. *Updating phase:* We carefully extract useful training targets from the generated experiences, and update the policy network and dual value networks.

Since desirable routes in retrosynthesis should be both synthesizable and low-cost, we construct two separate networks to predict the synthesizability and synthesis cost of molecules. To retain the single-step accuracy, we design a two-branch policy network structure. The first branch is a fixed, pre-trained single-step model that provides a set of valid reactions (e.g., the top 50 candidates). The second branch is a learnable single-step model that optimizes the probability distribution over valid reactions to optimize complete routes.

To demonstrate the effectiveness of our PDVN algorithm, we conduct extensive experiments on the widely used USPTO dataset (Lowe, 2012; Chen et al., 2020). The results show that using the single-step model trained by PDVN largely improves the success rate and route quality of existing multi-step planners. For the Retro* planner (Chen et al., 2020), PDVN increases the search success rate from 85.79% to 98.95%. For the RetroGraph planner (Xie et al., 2022), PDVN reduces the number of model calls by half when solving 99.47% molecules, and achieves the state-of-the-art on the USPTO dataset. Additionally, PDVN effectively reduces the length of found synthesis routes. We also find that PDVN can bring performance gains when addressing hard target molecules from ChEMBL and GDB17 datasets. For the ChEMBL-1000 dataset, the number of molecules solved

increased from 762 to 835. For the GDB17-1000 dataset, the number of molecules solved increased from 95 to 269. Case studies show that PDVN can reliably find chemically valid synthesis routes.

2. Related Work

Single-Step Retrosynthesis. Denote the space of all molecules as $\mathcal{S}$. The single-step predictor takes a product molecule $s \in \mathcal{S}$ as input and predicts a set of possible reactants that can be used to synthesize s. A single-step model can be learned from existing real-world datasets of chemical reactions. Current single-step models roughly fall into two categories, i.e., template-based and template-free. Template-based methods predict reactants with reaction templates that encode chemical reaction cores. The key is to rank template candidates and select an appropriate one to apply. Recent works (Segler & Waller, 2017; Coley et al., 2017; Dai et al., 2019; Chen & Jung, 2021) address the problem of template selection by using a classification neural network. On the other hand, inspired by the recent progress of seq2seq models (Sutskever et al., 2014) and Transformers (Vaswani et al., 2017), template-free methods (Liu et al., 2017; Tetko et al., 2020) cast single-step retrosynthesis as a translation task, where SMILES string[1] of a product molecule is translated to these of the reactants. As more single-step models are developed, the single-step accuracy continues to increase. Some recent benchmark papers (Hassen et al., 2022; Tu et al., 2022) show that the single-step models need to be developed and tested for the multi-step domain.

Multi-Step Retrosynthesis. Multi-step retrosynthetic planning aims to search for the whole synthesis route, by iteratively calling the single-step model. (Segler et al., 2017; 2018) used a Monte Carlo Tree Search (MCTS) algorithm to plan the synthesis routes of small organic molecules. (Kishimoto et al., 2019) propose a DFPN-E method that combines Depth-First Proof-Number Search (DFPN) with heuristic edge initialization. Recently, (Chen et al., 2020) propose Retro*, a neural-based A*-like algorithm, which employs AND-OR search trees and adopts the best-first search strategy on the AND-OR tree. To reduce the duplication of molecules in the tree-based search method, (Xie et al., 2022) propose a graph-based search algorithm named RetroGraph, to further improve the performance of A*-like search algorithms.

(Kim et al., 2021) propose a self-improving procedure, called Retro*+, which trains a single-step model to imitate successful trajectories found by itself. Despite its achievements, Retro*+ only maximizes the success rate and leverages successful simulated experiences to improve the single-step model. Additionally, the A*-like search algorithm used in Retro*+ is based on the best-first search and may fail to effectively balance exploration and exploitation when generating experiences. (Yu et al., 2022) propose GRASP, a goal-driven actor-critic method for finding routes with a specific prescribed goal such as building block materials. Unlike GRASP, which focuses on goal-driven retrosynthesis, our work focuses more on general retrosynthetic planning.

[1]Symbolic representation for describing the structures of molecules using ASCII strings.

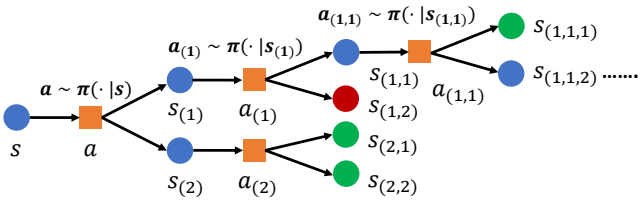

Figure 3. An illustrative example of the tree-shaped MDP. Starting from the target molecule, chemists recursively choose reactions (denoted by orange rectangles) to break down the product molecules (denoted by blue circles) into reactants, until reaching building block molecules (denoted by green circles) or dead-end molecules (denoted by red circles). In this example, the route is not synthesizable, as there is a red dead-end leaf node $S_{(1,2)}$.

3. Method

In this section, we first formulate the retrosynthesis problem using a tree-shaped MDP (Section 3.1). Then, we introduce the Planning with Dual Value Networks (PDVN) algorithm, which alternately performs the planning phase (Section 3.2) and the updating phase (Section 3.3). Finally, to retain single-step retrosynthesis accuracy, we introduce a two-branch policy network structure (Section 3.4). For an overview of our PDVN algorithm, refer to Fig. 2.

3.1. Retrosynthesis MDP

The task of retrosynthesis can be modeled as a tree-shaped MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, 2^{\mathcal{S}}, \mathcal{P}, c)$. $\mathcal{S}$ denotes the state space whose element $s \in \mathcal{S}$ represents a molecule. Terminal states in $\mathcal{S}$ can be classified into (1) building blocks $\mathcal{S}_{\text{bb}}$ which are commercially available molecules, and (2) dead-end molecules $\mathcal{S}_{\text{dead}}$ to which no reactions are available. The element of the action space $a \in \mathcal{A}$ represents the chemical reaction that transforms a product molecule into reactant molecules. The deterministic transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to 2^{\mathcal{S}}$ represents the single-step chemical reaction, whose inputs are the product molecule and the reaction to take, and the output is the set of reactants. Unlike a traditional MDP where the trajectory of each episode is a single path, a trajectory of such an MDP forms a tree, since reactions usually have multiple reactants and thus cause branches (as illustrated in Fig. 3). The cost function

$c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ consists of the cost of performing a certain reaction $c_{\text{rxn}}(s, a)$ and the cost of reaching dead-end molecules $c_{\text{dead}}(s, a)$ as follows:

$$c(s,a) = c_{\text{rxn}}(s,a) + \underbrace{c_{\text{dead}} \cdot \sum_{s' \in \mathcal{P}(s,a)} \mathbb{1}(s' \in \mathcal{S}_{\text{dead}})}_{c_{\text{dead}}(s,a)}. \quad (1)$$

For example, (Schreck et al., 2019) set $c_{\text{rxn}}(s, a) = 1$ for all the reactions, $c_{\text{dead}} = 100$ for all dead-end molecules. Minimizing such cost function aims to generate routes that are synthesizable (i.e., no dead-end molecules in the route) and as short as possible. We note that our cost function is completely general, and can be trivially extended to for example account for building block or reagent prizes, or other route criteria such as convergence.

Given a policy function $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, the value function represents the expected total cost of the generated routes for molecule s:

$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{(s',a') \in \tau} c(s', a')\right], \quad (2)$$

where τ is a tree-structured trajectory starting from state s.

Dual Value Networks. As we can see, the desirable routes in retrosynthesis should be 1) synthesizable and 2) as low-cost as possible. Instead of using one value network to capture both desiderata, we use the law of total expectation[2] to decompose the function in Eqn. 2 into two value functions of different kinds. Specifically, let one random variable X represent the total cost of the route: $\sum_{(s',a') \in \tau} c(s', a')$[3], the other random variable Y represent whether the route has no dead ends: $\mathbb{1}(\sum_{(s',a') \in \tau} c_{\text{dead}}(s', a') = 0)$, then the value function in Eqn. 2 can be rewritten as follows:

$$\begin{aligned}
&\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]] \\
&= P_\pi(Y = 1) \cdot \mathbb{E}[X|Y = 1] + P_\pi(Y = 0) \cdot \mathbb{E}[X|Y = 0] \\
&\approx P_\pi(Y = 1) \cdot \mathbb{E}\left[\sum_{(s',a') \in \tau} c_{\text{rxn}}(s', a')|Y = 1\right] + \\
&P_\pi(Y = 0) \cdot c_{\text{dead}} \quad \text{(omit the coefficient of } c_{\text{dead}} \text{ here,} \\
&\quad \text{since } c_{\text{dead}} \text{ is a relatively large penalty constant).}
\end{aligned} \quad (3)$$

One value network $V^{\text{syn}}(s)$, called synthesizability value network, aims to approximate the probability term $P_\pi(Y = 1)$. $V^{\text{syn}}(s)$ represents the probability of generating a synthesizable route for molecule s following policy π. The other value network $V^{\text{cost}}(s)$, called cost value network, aims to approximate the term $\mathbb{E}[\sum_{(s',a') \in \tau} c_{\text{rxn}}(s', a')|Y = 1]$. $V^{\text{cost}}(s)$ represents the expected total reaction costs given the synthesizable route. Note that both value functions also satisfy the Bellman equation according to their mathematical definitions:

$$\begin{aligned}
V_\pi^{\text{syn}}(s) &= \mathbb{E}_{a \sim \pi}\left[\prod_{s' \in \mathcal{P}(s,a)} V_\pi^{\text{syn}}(s')\right], \\
V_\pi^{\text{cost}}(s) &= \mathbb{E}_{a \sim \pi}\left[c_{\text{rxn}}(s,a) + \sum_{s' \in \mathcal{P}(s,a)} V_\pi^{\text{cost}}(s')|Y = 1\right].
\end{aligned} \quad (4)$$

In a tree-shaped MDP, the Bellman equation is based on the product/sum over all children reactant nodes.

3.2. Planning with Dual Value Networks

To learn optimal policies that lead to desirable routes on the retrosynthesis MDP, we propose an algorithm named Planning with Dual Value Networks (PDVN), which alternates between the planning phase and the updating phase.

The planning phase aims to generate valuable simulated experiences on tree-shaped MDP with MCTS-based planning utilizing dual value networks. First, we initialize an empty search tree with a target molecule as the root node. In each iteration, a tree search process is executed from the current root node, utilizing the dual value networks and policy network. After the process completes, search probabilities based on the visit count of each reaction from the current root node are returned. One reaction is chosen according to this search probabilities, and the reactant nodes associated with the chosen reaction are pushed into a stack. In the next iteration, a molecule is popped from the top of the stack and serves as a new root node. The planning phase will conclude when either all the leaf nodes have been converted to building block molecules, or a dead-end leaf node is encountered.

In particular, each tree search process runs a predetermined number of simulations (e.g., 500 steps), and each simulation comprises three sequential steps: 1) Selection, 2) Expansion, and 3) Backup. We omit the step of rollout evaluation, and instead use networks to initialize the value of the newly expanded nodes, as this practice can effectively reduce the variance and computation efforts.

Selection. Starting from the current root node, we alternate between selecting a reaction node and a child molecule node, until we encounter a leaf molecule node. To select reaction nodes, we propose a modified version of the PUCT rule (Rosin, 2011) that considers dual value networks. The new rule includes an estimate of synthesizability $R(s, a)$, an estimate of cost $Q(s, a)$, the policy $\pi(a|s)$, and the visit

[2] $\text{E}[X] = \text{E}[\text{E}[X|Y]]$.

[3] Cost function $c(s, a)$ is defined in Eqn. 1.

count $N(s,a)$; its detailed equation is derived from Eqn. 3:

$$a = \arg\max_{a'} -U(s,a') + C\,\pi(a'|s)\,\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a')},$$
$$U(s,a') = R(s,a')\cdot Q(s,a') + (1-R(s,a'))\cdot c_{\text{dead}}, \tag{5}$$

where C is the exploration coefficient. To select a child molecule node[4], we prioritize molecules that have not been expanded; if none are available, we choose ones that have not been solved. If molecule nodes are either all expanded or all solved, we randomly select one of them.

Expansion. When a leaf molecule node is encountered, we expand the search tree by adding reaction nodes and their corresponding reactant nodes. Specifically, we select the top 50 predictions of the policy network to append reaction nodes[5], and then use RDKit[6] to obtain the corresponding reactant nodes. For new molecule nodes, the visit count is initially set to zero; the dual value networks assign values for the initial V^{syn} and V^{cost}.

Backup. At the end of each simulation, the nodes visited, both the reaction and molecule nodes, form a path $T = (s_0, a_0, \ldots, s_l, a_l, \ldots, s_L)$, where s_0 is the root of the current simulation and s_L is the leaf node before expansion. During the backup step, we first calculate the current values of each molecule node s_l ($0 \le l \le L-1$) on path T by:

$$V_T^{\text{syn}}(s_l) = V_T^{\text{syn}}(s_{l+1}) \cdot \prod_{s' \in \mathcal{P}(s_l,a_l)\setminus\{s_{l+1}\}} V^{\text{syn}}(s'),$$
$$V_T^{\text{cost}}(s_l) = c_{\text{rxn}}(s_l,a_l) + V_T^{\text{cost}}(s_{l+1}) + \sum_{s' \in \mathcal{P}(s_l,a_l)\setminus\{s_{l+1}\}} V^{\text{cost}}(s'). \tag{6}$$

The above update rule is derived from the Bellman equation for a tree-shaped MDP (i.e., Eqn. 4). Then, we update the average value $V^{\text{syn}}(s_l)$, $V^{\text{cost}}(s_l)$ by $V^{\text{syn}}(s_l) = (V^{\text{syn}}(s_l)\cdot N(s_l) + V_T^{\text{syn}}(s_l))/(N(s_l)+1)$, $V^{\text{cost}}(s_l) = (V^{\text{cost}}(s_l)\cdot N(s_l) + V_T^{\text{cost}}(s_l))/(N(s_l)+1)$, and the visit count by $N(s_l) = N(s_l)+1$. Finally, we update the values of reaction nodes as follows:

$$R(s_l,a_l) = \prod_{s' \in \mathcal{P}(s_l,a_l)} V^{\text{syn}}(s'),$$
$$Q(s_l,a_l) = c_{\text{rxn}}(s_l,a_l) + \sum_{s' \in \mathcal{P}(s_l,a_l)} V^{\text{cost}}(s'). \tag{7}$$

[4]Since each reaction node may have multiple child molecule nodes in a tree-shaped MDP

[5]Since 50 is widely used in previous work (e.g., 3N-MCTS, Retro*, RetroGraph, Retro*+).

[6]https://www.rdkit.org/, open-source cheminformatics software.

Note that $V_T^{\text{syn}}(s_l)$ and $V_T^{\text{cost}}(s_l)$ denote the values calculated from current path T, while $V^{\text{syn}}(s_l)$ and $V^{\text{cost}}(s_l)$ denote the average values over all visited paths.

3.3. Training on Generated Experiences

After completing the Planning phase, we have a search tree along with the statistics gathered during planning. This tree contains valuable information for updating the networks, regardless of whether it solves the target molecule, helping to benefit future planning. During PDVN training, three neural networks (i.e., policy network, synthesizability value network, and cost value network) play different roles. In this subsection, we carefully design the process of extracting data and the objective function for training each network.

Policy Network. The policy network $\pi(a|s)$ aims at predicting the reactions that lead to desirable routes. Instead of imitating the visitation frequency $N(s,a)/\sum_b N(s,b)$ from MCTS simulations as in AlphaZero (Silver et al., 2017; 2018), we extract pairs of (molecule, reaction) from successful routes with minimal cost in the search tree. Specifically, for each molecule node in the search tree, we first determine if there are any reactions leading to a successful route. If more than one reaction meet the required condition, we will select the one with the lowest cost. We use cross-entropy (CE) loss to update the policy network.

Synthesizability Value Network. The synthesizability value network $V^{\text{syn}}(s)$ aims to predict the probability of solving molecule s. We train $V^{\text{syn}}(s)$ by using all the molecules in the search tree. First, we run a recursive algorithm to check if each molecule node in the search tree is solved or not. For solved molecule nodes, we set the training target as 1. For unsolved molecules s, we use $0.8 \times V^{\text{syn}}(s)$ as the training target, where 0.8 is a slight penalty since the molecule is not solved in the search tree. For dead-end molecules, we set the training target as 0. We use binary cross-entropy (BCE) loss to update the synthesizability value network.

Cost Value Network. The cost value function $V^{\text{cost}}(s)$ aims to estimate the minimal cost or length of synthesizing the molecule. $V^{\text{cost}}(s)$ is trained only on solved molecules in the search tree. Specifically, we first use a recursive algorithm to obtain the lengths of the shortest successful routes on those solved molecules, which we use as the training target for $V^{\text{cost}}(s)$. We minimize the mean squared error (MSE) loss to update the cost value network.

3.4. Two-Branch Policy Network Structure

The above subsections focus on optimizing policies to generate desirable routes. In this subsection, we focus more on how to retain single-step accuracy.

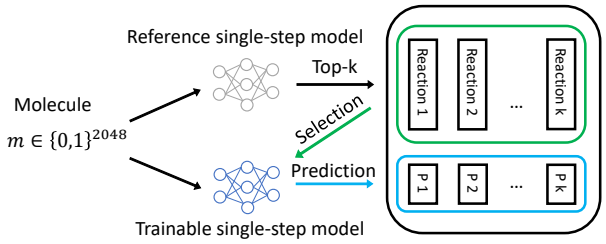

Figure 4. An illustration of the two-branch policy network. The reference single-step model provides a realistic subset of reactions for the input molecule, denoted by Reaction 1 . . . , Reaction k . The learnable single-step network optimizes a probability distribution over the selected reactions, i.e., P_i is the probability of Reaction i.

A natural way to design a policy network is to directly use a single-step model. However, as the training proceeds, such policy may choose unrealistic reactions that are not likely to happen in practice. Chemists often question the feasibility of the routes generated by AI-based retrosynthesis software (Genheden & Bjerrum, 2022). To this end, we design a two-branch policy network structure, as illustrated in Fig. 4.

Specifically, the proposed policy network consists of two separate branches. The first branch, called the reference single-step model, is inherited from a single-step model trained by offline SL, and frozen during RL training. Following (Segler et al., 2018; Chen et al., 2020), we use a template-based MLP model as the single-step model, which is a multi-class classification network[7]. Since the model is pre-trained using the real-world reaction dataset, this branch provides a realistic subset of reactions (e.g., the top-50 candidates from a total of $\sim 380K$ classes). The second branch is initialized by the SL model, but has learnable parameters that can be optimized to generate synthesizable and cost-effective routes from the realistic subset of reactions. The training objective refers to the Policy Network part in section 3.3. These two branches work together to generate both realistic and desirable synthesis routes.

4. Experiments

In this section, we aim to answer the following questions: **Q1:** On the widely-used USPTO dataset, can our PDVN algorithm significantly improve the performance of existing multi-step planners? **Q2**: On more test target molecules from ChEMBL and GDB-17, can PDVN algorithm still bring performance gains? **Q3**: Are the proposed dual value networks necessary in our algorithm? **Q4**: Does PDVN algorithm helps find chemically sound routes?

[7]Template-based approaches use reaction templates to predict reactants from a product. First, the template-based model predicts the reaction templates, and then the template is applied to a product to find a match via subgraph isomorphism. If a proper isomorphism is found, the product is transformed according to the template.

4.1. Experimental Setup

Our algorithm requires specifying (1) a set of building block molecules $\mathcal{S}_{bb}$, (2) a training target molecule dataset $\mathcal{D}_{\text{train}}$, (3) a test target molecule dataset $\mathcal{D}_{\text{test}}$, (4) a retrosynthetic planning algorithm, (5) a single-step model.

Dataset. (1) For the building block molecules $\mathcal{S}_{bb}$, we follow common practice (Chen et al., 2020), and use the commercially available molecules (about $23.1M$) from *eMolecules*[8]. (2) For the training target molecule dataset $\mathcal{D}_{\text{train}}$, we follow the procedure from (Chen et al., 2020; Kim et al., 2021) and construct synthesis routes based on the publicly available reaction dataset extracted from the United States Patent Office (USPTO) (Lowe, 2012). Specifically, they take each molecule that has appeared in USPTO reaction data and analyze if it can be synthesized by existing reactions within USPTO training data. After processing, 299202 training routes are obtained. Following (Kim et al., 2021), we use the root molecules of these training routes as training target molecules to generate simulated experiences in the Planning phase. (3) For the test target molecules $\mathcal{D}_{\text{test}}$, we use the 190 challenging target molecules that were widely used in previous work (Chen et al., 2020; Kim et al., 2021; Han et al., 2022; Xie et al., 2022; Tripp et al., 2022). Besides, we introduce two novel test datasets, i.e., ChEMBL-1000 and GDB17-1000, to assess the generalizability of the trained model.

Multi-Step Planner. We use two popular multi-step planners: 1) Retro* (Chen et al., 2020), an efficient retrosynthetic planning algorithm built upon the AND-OR search tree and best-first search; RetroGraph (Xie et al., 2022), which proposes to use AND-OR graph to handle the duplicated nodes and searches within the retrosynthetic paths. Note that Retro*-0 denotes a variant of Retro* not relying on the pretrained value function as the heuristic (Chen et al., 2020). These two planners are based on the A* algorithm and can be combined with any single-step model. We plug the single-step model trained by PDVN into these two planners to see if there are any improvements on the test target molecule dataset.

Single-Step Model. Following (Segler et al., 2018; Chen et al., 2020; Kim et al., 2021), we use a template-based single-step model, which is a 2-layer MLP with ELU activation. The output layer has $\sim 380K$ units and each corresponds to a distinct template. Although simple, this model is adopted in the most widely used retrosynthesis implementations (e.g., Retro*, ASKCOS, and AIZynthfinder).

[8]https://downloads.emolecules.com/free/2022-11-01/

Table 1. Performance summary on the USPTO 190 test dataset. The evaluation metrics include the success rates under different numbers of model calls (N), the average number of model calls used, the average number of reaction nodes (T) and molecule nodes (M) visited, under the computation budget of 500 model calls.

	Success rate [%] ↑								
Algorithm	50	100	200	300	400	500	# model calls ↓	# T nodes ↓	# M nodes ↓
Greedy DFS	-	38.42	40.53	44.21	45.26	46.84	300.56	-	-
DFPN-E	-	50.53	58.42	64.21	68.42	75.26	208.12	3123.33	4635.08
MCTS-rollout	-	43.68	47.37	54.74	58.95	62.63	254.32	-	-
Retro*-0	27.37	38.42	58.42	67.37	75.26	79.47	209.86	3905.62	5565.37
Retro*	40.00	55.79	70.53	76.84	82.11	85.79	158.81	2632.84	3685.31
Retro*+-0	56.84	67.37	83.16	92.11	94.74	96.32	97.95	1444.52	2139.3
Retro*+	63.16	74.21	83.16	86.84	90.00	90.53	98.91	1157.74	1708.17
PDVN+Retro*-0	**86.32**	**93.68**	**97.37**	**97.89**	**98.95**	**98.95**	**30.94**	**773.56**	**995.22**
RetroGraph	69.47	88.42	97.89	98.95	99.47	99.47	45.13	674.23	500.44
PDVN+RetroGraph	**93.16**	**96.84**	**97.89**	**99.47**	**99.47**	**99.47**	**20.24**	**486.87**	**417.54**

To train this model, we follow (Chen et al., 2020) and use the offline training dataset comprising $\sim 1.3M$ reactions extracted from USPTO published up to September 2016.

Dual value networks. For the synthesizability value network, we use a 2-layer MLP where the output layer is a sigmoid layer. The cost value network uses the same network architecture but the output layer is a softplus layer as the cost of non-building blocks is positive. The input to both networks is the molecular Morgan fingerprint of radius 2, which is a 2048-bit vector.

Training. For the two-branch policy network, the reference model is pre-trained offline by SL and then frozen during PDVN training. The learnable branch has the same network architecture as the reference model and is initialized by the reference model. Instead of training the reference model from scratch, we load the model checkpoint provided by (Chen et al., 2020). The parameters of the dual value networks are randomly initialized. During the Planning phase, the batch size of sampled target molecules is 1024. We set $c_{rxn}(s, a) = 0.1$ and $c_{dead} = 5$. During the Updating phase, the Adam optimizer (Kingma & Ba, 2014) with a mini-batch of size 128 and a learning rate of 0.001 is used for all models. We iterate the training target molecule dataset $\mathcal{D}_{\text{train}}$ three times.

4.2. USPTO Results

To answer **Q1**, we investigate the effectiveness of PDVN in terms of planning efficiency and route quality. We train the single-step model by PDVN and load the trained model to two state-of-the-art retrosynthesis planners, i.e., Retro* and RetroGraph. The results against other baselines are summarized in Table 1 and Table 2.

Table 2. The average length of the routes on the USPTO 190 test dataset. The results are averaged over the 138 molecules that can be solved by all methods.

Algorithm	Avg length ↓
Retro*-0	5.83
Retro*	5.76
Retro*+-0	6.16
Retro*+	5.77
PDVN+Retro*	**4.83**
RetroGraph	5.63
PDVN+RetroGraph	**4.78**

We can observe that the success rates of both Retro* and RetroGraph increase significantly when combined with the PDVN trained model, and the average number of model calls is reduced by more than half. With the model calls limit $N = 500$, the success rate of Retro* is improved to 98.95% while Retro*+, which imitates the successful routes given by Retro*, achieves a success rate of 96.32%. Notably, PDVN improves the success rate for fewer model calls limit, especially $N = 50$. Previous best result with $N = 50$ is 69.47% by RetroGraph, PDVN significantly improves it to 93.16%.

On the other hand, we use the average route length to represent route quality. To make a fair comparison, we only consider the molecules that can be solved by all the planners. As shown in Table 2, PDVN outperforms the baselines by a large margin: the trained model reduces the average route length on the 138 selected molecules from 5.83 to 4.83 for Retro*-0 and from 5.63 to 4.78 for RetroGraph.

Table 3. Number of solved target molecules on ChEMBL-100 dataset and GDB17-1000 datasets.

Algorithm	ChEMBL-1000	GDB17-1000
Retro*-0	751	75
Retro*	762	95
Retro*+-0	811	150
Retro*+	818	154
PDVN+Retro*-0	**835**	**269**
RetroGraph	852	215
PDVN+RetroGraph	**860**	**371**

Table 4. Ablation study on dual value networks.

Algorithm	Success rate	Avg length
PDVN+Retro*-0	**98.95**	**4.83**
SingleValue+Retro*-0	95.26	5.05
PDVN w/o Cost+Retro*-0	95.79	5.16
PDVN+RetroGraph	**99.47**	**4.78**
SingleValue+RetroGraph	96.32	4.93
PDVN w/o Cost+RetroGraph	96.32	5.00

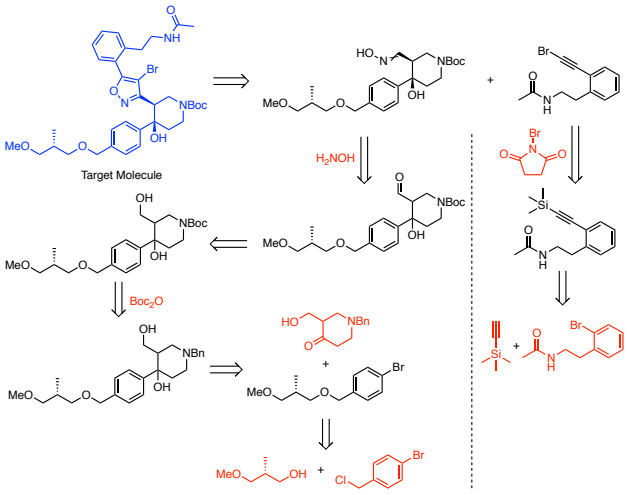

Figure 5. Case study of an exemplary route predicted with PDVN. The arrow represents the single-step chemical reaction, and the molecules at the end of the synthesis route are building block molecules.

4.3. Results on ChEMBL and GDB-17 Datasets

To answer **Q2**, we follow (Tripp et al., 2022) and construct two more test datasets from the ChEMBL dataset and GDB17 dataset, i.e., ChEMBL-1000 and GDB17-1000. They are 1000 molecules randomly chosen from a subset of the ChEMBL dataset and GDB17 dataset. To create a subset of molecules equally or more challenging than the USPTO 190 test dataset, we preprocess the CHEMBL dataset and GDB17 dataset by using the script from (Brown et al., 2019), keeping only molecules whose molecular weight, Bertz coefficient, logP, and TPSA were larger than the mean of the respective values in the USPTO 190 test dataset, and removing all known building block molecules.

The results are reported in Table 3, and we can see that PDVN still brings performance gains to the baselines. For ChEMBL-1000, PDVN+Retro*-0 and PDVN+RetroGraph can solve 24 and 8 more molecules than Retro*+-0 and RetroGraph, respectively. The GDB17-1000 dataset is much harder but the improvement is more obvious. Retro*+-0 and RetroGraph can solve 154 and 215 molecules, and our method enables both planners to additionally solve more than half of what they can originally do, solving as many as 269 and 371 molecules, respectively.

4.4. Ablation Study on Dual Value Networks

In order to verify the necessity of the proposed dual value networks, we design two variants of PDVN using only one value network for comparison. For the first variant, which we call SingleValue, we use a single value network $V^{\text{single}}(s)$ to estimate the overall objective of retrosynthesis in Eqn. 2. The backup step of $V^{\text{single}}(s)$ is similar to that of the cost value $V^{\text{cost}}(s)$ described in Eqn. 6. The PUCT rule is also modified to use the single value $V^{\text{single}}(s)$. For the second variant, which is called PDVN w/o Cost, we remove the cost value network in PDVN, and use only the synthesizability value network during PDVN training.

As shown in Table 4, both variants have lower success rates than PDVN, which implies that 1) it is beneficial to decompose the total cost into dual value networks; 2) cost value network benefits the training, along with synthesizability value network. Besides, we observe that PDVN w/o Cost can achieve a slightly higher success rate than SingleValue for the Retro*-0 planner, but the average length of found routes is longer. That may suggest that it is important to take the cost term into consideration.

4.5. Qualitative Case study

A potential risk of reinforcement learning is exploitation of the environment, i.e. the single step SL model with its known imperfections. We performed a qualitative analysis of the routes given by Retro* using different models. The analysis indicated that PVDN leads to routes of similar chemical plausibility as the SL model, with usually fewer steps. As an example, we choose molecule 25 from the USPTO 190 test dataset. With the computation budget of 500 model calls, Retro* cannot solve it using the SL model, however, with PDVN it is able to provide a route. As shown in Fig. 5, our method identifies a route closely related to the

hold-out original route, albeit with one synthesis step less.

5. Conclusion

In this work, we introduce PDVN, a novel policy learning framework for retrosynthesis. PDVN improves the single-step predictor to not only predict valid reactions, but also predict reactions that lead to synthesizable and low-cost synthesis routes. Experiments on the widely-used USPTO dataset demonstrate that PDVN significantly enhances both the search success rate and route quality of existing multi-step planners (e.g., Retro*, RetroGraph), achieving state-of-the-art performance on the USPTO dataset. For future work, one potential direction is to extend our PDVN algorithm to other single-step models, such as template-free ones, which have shown great single-step accuracy and generalizability. We anticipate that our algorithm will help to further accelerate the discovery of molecules and materials for health care, agriculture, and energy storage.

6. Acknowledgments

We would like to thank Elise van der Pol for proofreading the manuscript and offering invaluable feedback, as well as Sarah Lewis and Megan Stanley for their insightful discussions. Di Xue and Zongzhang Zhang acknowledge funding from the National Science Foundation of China (62276126) and the Fundamental Research Funds for Central Universities (14380010).

References

Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. Guacamol: benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59(3):1096–1108, 2019.

Chen, B., Li, C., Dai, H., and Song, L. Retro*: learning retrosynthetic planning with neural guided A* search. In *International Conference on Machine Learning*, pp. 1608–1616. PMLR, 2020.

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

Coley, C. W., Rogers, L., Green, W. H., and Jensen, K. F. Computer-assisted retrosynthesis based on molecular similarity. *ACS Central Science*, 3(12):1237–1245, 2017.

Dai, H., Li, C., Coley, C., Dai, B., and Song, L. Retrosynthesis prediction with conditional graph logic network. In *Advances in Neural Information Processing Systems*, volume 32, pp. 8870–8880, 2019.

Dong, J., Zhao, M., Liu, Y., Su, Y., and Zeng, X. Deep learning in retrosynthesis planning: datasets, models and tools. *Briefings in Bioinformatics*, 23(1), 09 2021. ISSN 1477-4054.

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

Han, P., Zhao, P., Lu, C., Huang, J., Wu, J., Shang, S., Yao, B., and Zhang, X. Gnn-retro: Retrosynthetic planning with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 4014–4021, 2022.

Hassen, A. K., Torren-Peraire, P., Genheden, S., Verhoeven, J., Preuss, M., and Tetko, I. V. Mind the retrosynthesis gap: Bridging the divide between single-step and multi-step retrosynthesis prediction. In *NeurIPS 2022 AI for Science: Progress and Promises*, 2022.

Kim, J., Ahn, S., Lee, H., and Shin, J. Self-improved retrosynthetic planning. In *International Conference on Machine Learning*, pp. 5486–5495. PMLR, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kishimoto, A., Buesser, B., Chen, B., and Botea, A. Depth-first proof-number search with heuristic edge cost and application to chemical synthesis planning. In *Advances in Neural Information Processing Systems*, volume 32, pp. 7226–7236, 2019.

Liu, B., Ramsundar, B., Kawthekar, P., Shi, J., Gomes, J., Luu Nguyen, Q., Ho, S., Sloane, J., Wender, P., and Pande, V. Retrosynthetic reaction prediction using neural sequence-to-sequence models. *ACS Central Science*, 3 (10):1103–1113, 2017.

Liu, G., Zhao, L., Zhang, P., Bian, J., Qin, T., Yu, N., and Liu, T.-Y. Demonstration actor critic. *Neurocomputing*, 434:194–202, 2021. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.12.116.

Lowe, D. M. *Extraction of chemical structures and reactions from the literature*. PhD thesis, University of Cambridge, 2012.

Rosin, C. D. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3): 203–230, 2011.

Schreck, J. S., Coley, C. W., and Bishop, K. J. M. Learning retrosynthetic planning through simulated experience. *ACS Central Science*, 5(6):970–981, 2019.

Segler, M., Preuß, M., and Waller, M. P. Chemical synthesis planning with tree search and deep neural network policies. *arXiv preprint arXiv:1702.00020*, 2017.

Segler, M. H. and Waller, M. P. Neural-symbolic machine learning for retrosynthesis and reaction prediction. *Chemistry–A European Journal*, 23(25):5966–5971, 2017.

Segler, M. H. S., Preuss, M., and Waller, M. P. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, 555:604–610, 2018.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, pp. 3104–3112, 2014.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tetko, I. V., Karpov, P., Van Deursen, R., and Godin, G. State-of-the-art augmented NLP transformer models for direct and single-step retrosynthesis. *Nature Communications*, 11(1):1–11, 2020.

Tripp, A., Maziarz, K., Lewis, S., Liu, G., and Segler, M. Re-evaluating chemical synthesis planning algorithms. In *NeurIPS 2022 AI for Science: Progress and Promises*, 2022.

Tu, H., Shorewala, S., Ma, T., and Thost, V. Retrosynthesis prediction revisited. In *NeurIPS 2022 AI for Science: Progress and Promises*, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6000–6010, 2017.

Xie, S., Yan, R., Han, P., Xia, Y., Wu, L., Guo, C., Yang, B., and Qin, T. Retrograph: Retrosynthetic planning with graph search. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2120–2129, 2022.

Yu, Y., Wei, Y., Kuang, K., Huang, Z., Yao, H., and Wu, F. GRASP: Navigating retrosynthetic planning with goal-driven policy. In *Advances in Neural Information Processing Systems*, volume 35, pp. 10257–10268, 2022.

A. Implementation Details

A.1. Network architecture

The goal of the PDVN training is to optimize the parameters of the policy network and dual value networks. The inputs of these networks are binary strings of length 2048, which are the Morgan fingerprints of molecules of radius 2. The policy network has two sub-networks inside it, i.e., the reference single-step model and the learnable single-step model. The two sub-networks share the same MLP structure, and they are both initialized with the parameters from the SL trained model by (Chen et al., 2020). The dual value networks also share the same MLP structure but the output activation functions are different. The hyper-parameters of these neural networks are listed below.

Table 5. Hyper-parameters of neural networks.

Single-step model input units	2048
Dual value networks input units	2048
Single-step model hidden units	512
Dual value networks hidden units	512
Cost value network output activation	softplus
Synthesizability value network output activation	sigmoid

A.2. PDVN planning phase

To generate experiences for training, we design an MCTS-based planner based on dual value networks. Our planner resembles the online MCTS planner that conducts a fixed number of simulations at the root node in each iteration. We use a queue to store the simulation roots. Within each simulation, we alternately select reaction and molecule nodes until a leaf molecule node is encountered. We use the PUCT rule to select a reaction (in Eqn. 5), where a parameter C balances the trade-off between exploitation and exploration. Besides, we set a maximum route depth to avoid too-long synthesis routes. To avoid circular routes, we further eliminate those reactions that appeared in their ancestors.

Table 6. Hyper-parameters for PDVN planning.

C (PUCT)	1.0
α (Synthesizability penalty)	0.8
MCTS depth	15
Number of simulations	100
c_{dead}	5.0
$c_{\mathrm{rxn}}(s, a)$	0.1

A.3. PDVN training

The training of the PDVN algorithm iterates over the whole training target molecule dataset $\mathcal{D}_{\mathrm{train}}$ for three epochs. For each update, we uniformly sample a batch of training target molecules from $\mathcal{D}_{\mathrm{train}}$ to generate the training data, and update the networks. To speed up the data generation process, we implement a parallel version of MCTS planners to run MCTS planning for multiple training target molecules simultaneously. The whole training process takes about 18 hours on a server with four NVIDIA TITAN Xp and 48 CPU cores (using 15 parallel workers).

Table 7. Hyper-parameters for PDVN training.

Training dataset size	299202
Batch size	1024
Optimizer	Adam
Learning rate	1e-3
Dropout rate	0.1
Mini-batch size	128
SL epochs	8