ICML 2020 Presentation

Retro*: Learning Retrosynthetic Planning with Neural Guided A* Search

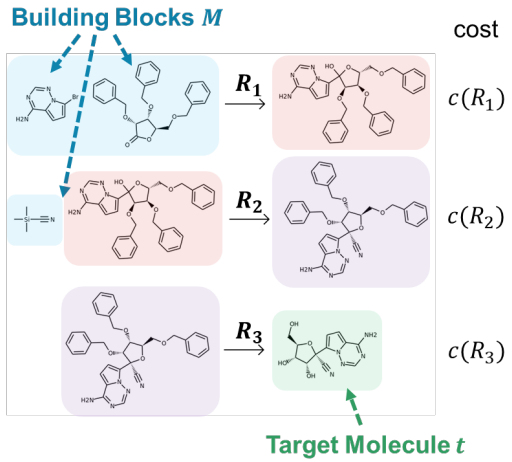
Binghong Chen[1], Chengtao Li[2], Hanjun Dai[3], Le Song[1,4]

[1]Georgia Institute of Technology, [2]Galixir, [3]Google Research, Brain Team,
[4]Ant Financial

Too Long; Didn't Watch

Our paper

- proposes the **optimal retrosynthetic planning** problem and,
- an A*-like algorithm which **learns from experience** as solution,
- with **state-of-the-art** performance on a real-world benchmark dataset.



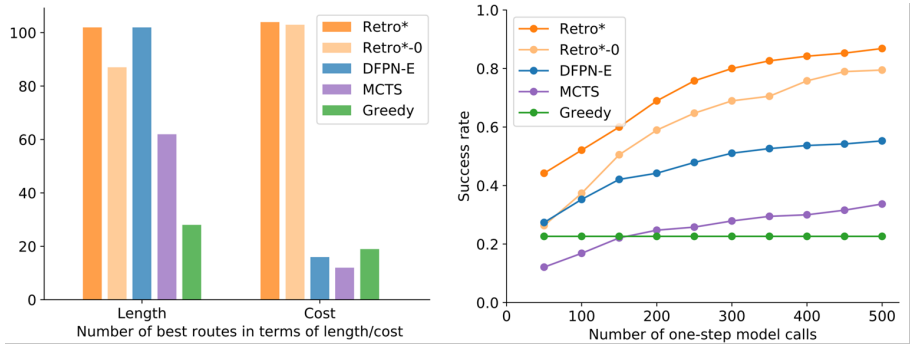
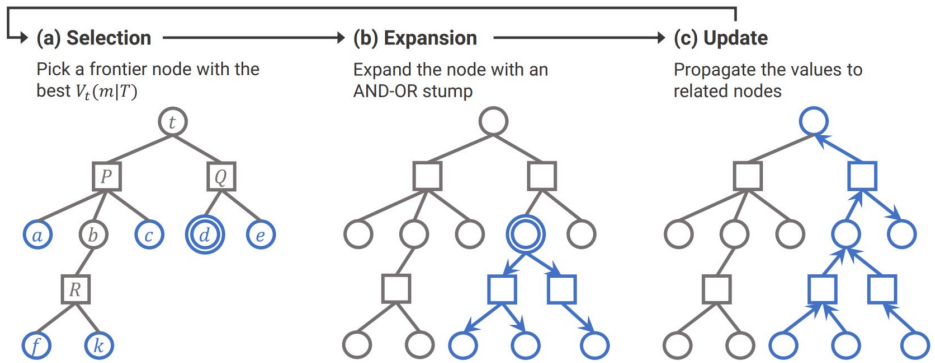


Figure: Counts of the best solutions among all algorithms in terms of length/cost.

Figure: Influence of time on performance.



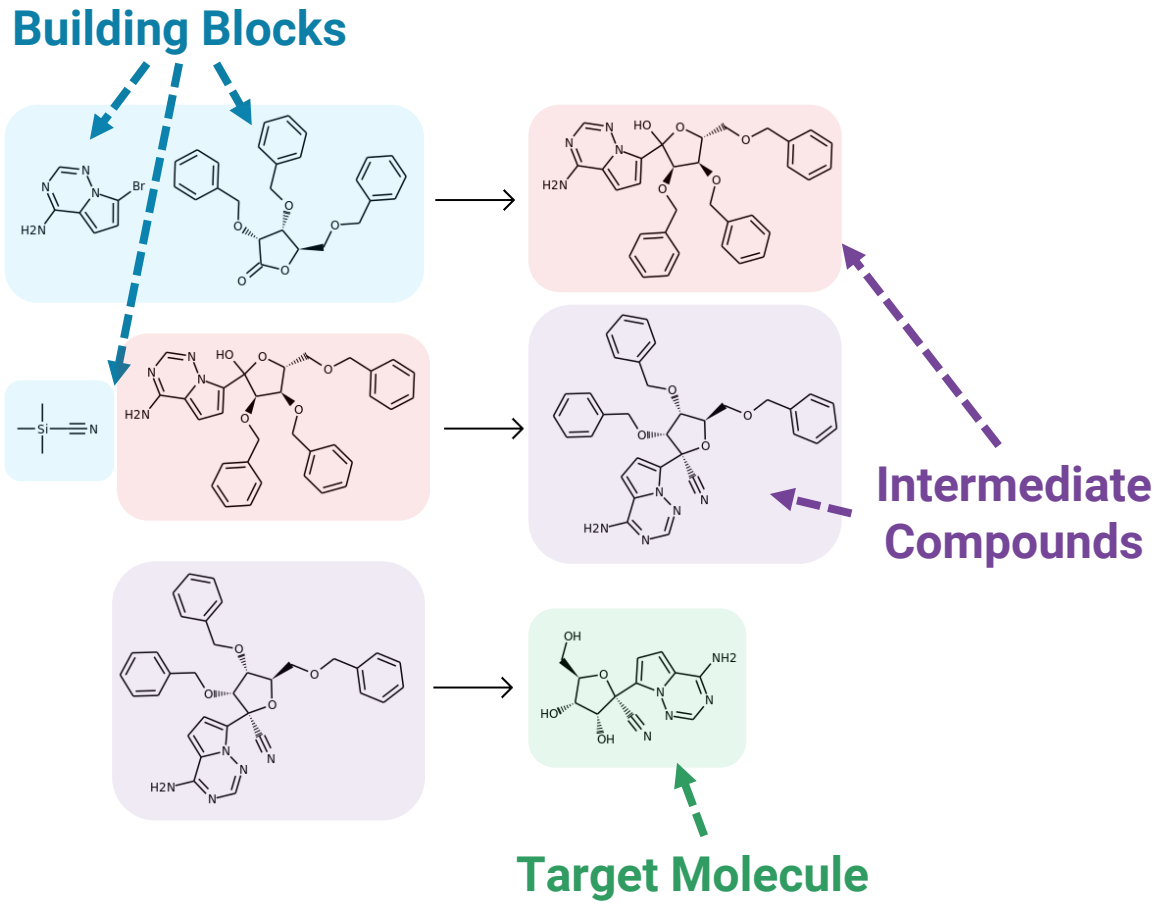
Background: Retrosynthesis Problem

Task: predict synthesis routes for target molecules.

Challenge: combinatorial search space.

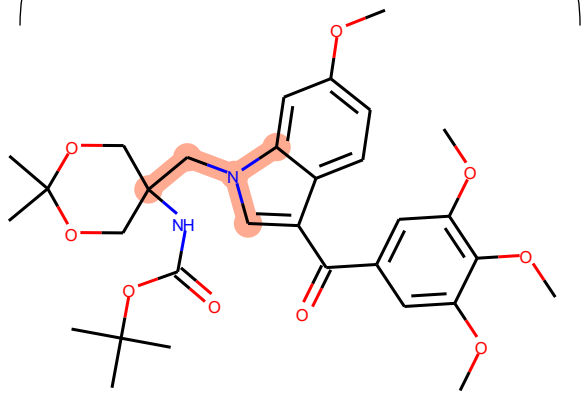
Sub-problems:

- One-step retrosynthesis
- Retrosynthetic planning



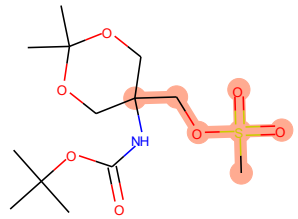
Background: One-step Retrosynthesis

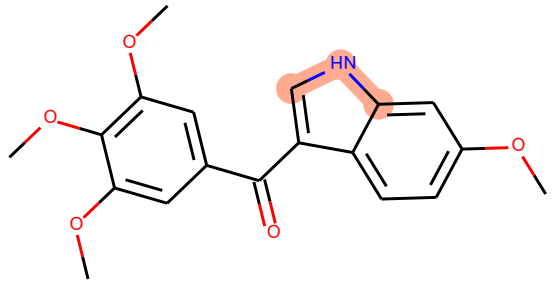
Given



Product t

Predict Reactants





Reactants

$$B(\cdot): \quad t \rightarrow \{R_i, \mathcal{S}_i, c(R_i)\}_{i=1}^{k}$$

$\mathcal{S}_i$: the i-th set of predicted reactants.

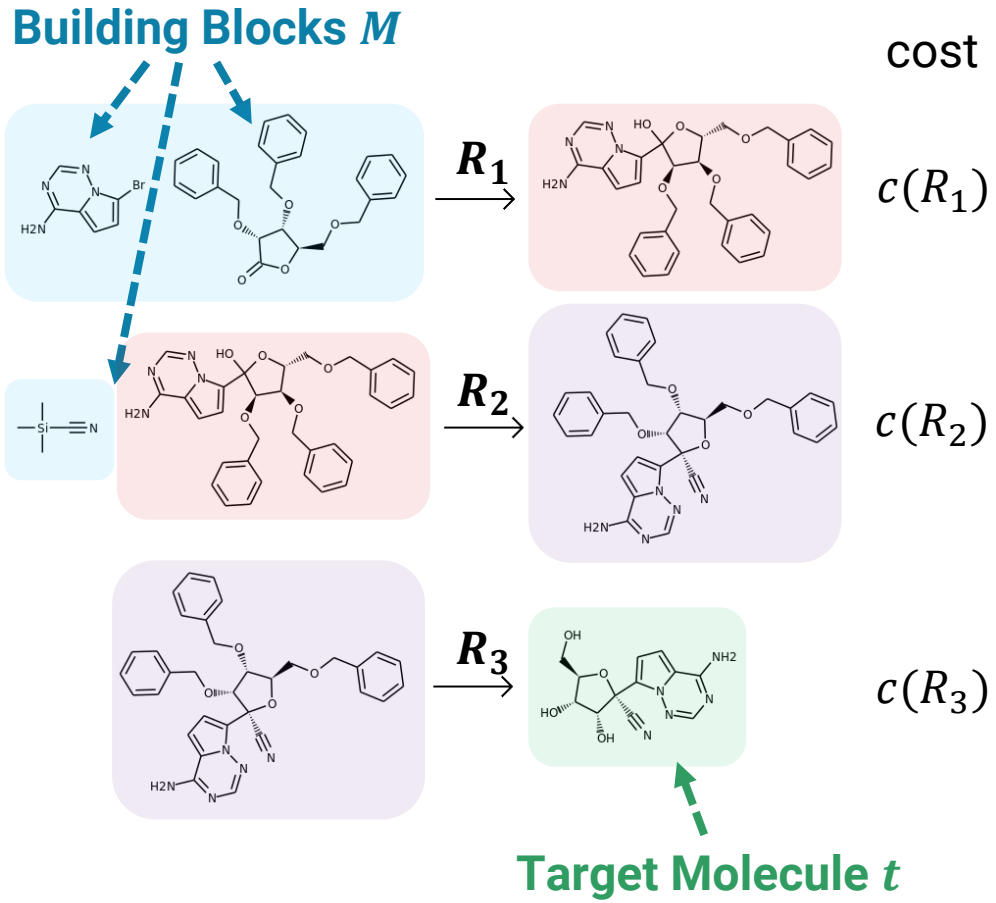
$c(R_i)$: cost of the i-th reaction.

Background: Retrosynthetic Planning

Plan a synthesis route from the reaction candidates produced by one-step model B.

Motivation: find *better routes*

- shorter with higher yields,
- more chemically sound,
- more economically efficient,
- more environmentally friendly,
- … (you name it)



Problem: Optimal Retrosynthetic Planning

Given:

a target molecule t, a set of building blocks M, a one-step model B.

Optimal Planning:

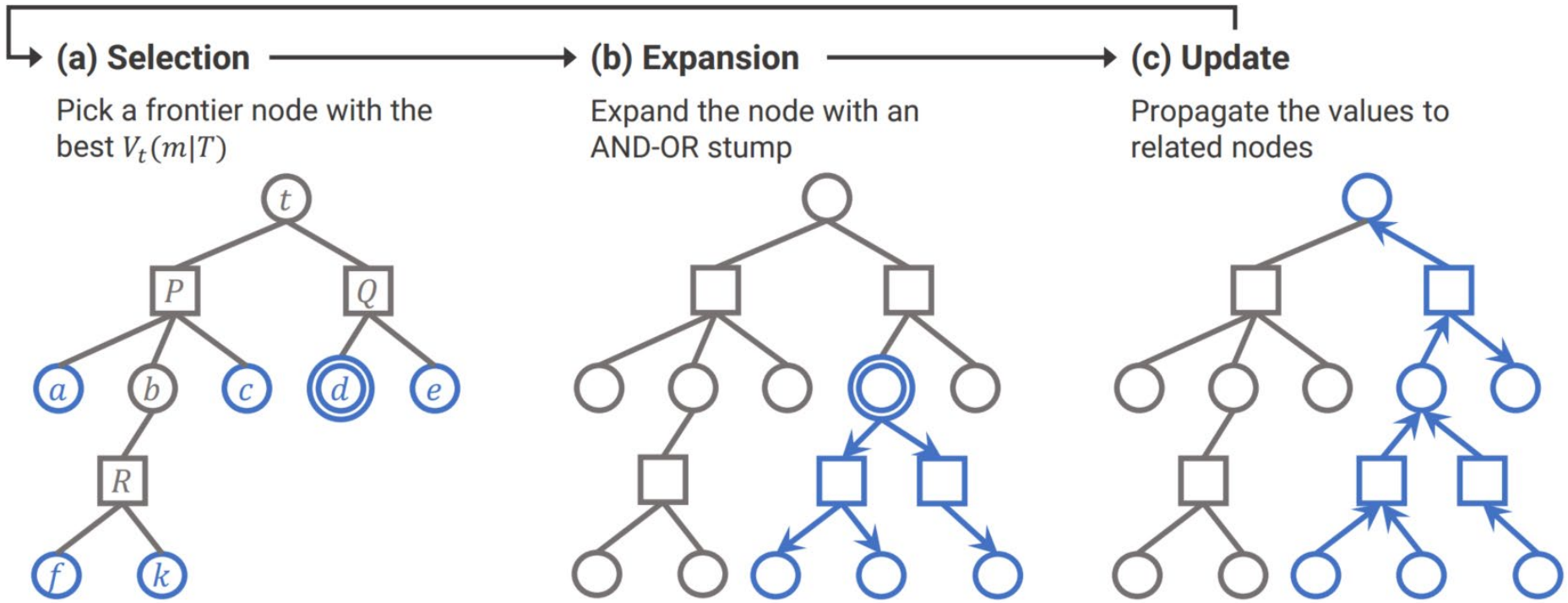
$$\text{Minimize} \quad c(R_1) + c(R_2) + \cdots + c(R_k)$$

Where $R_1, R_2, \ldots, R_k$ is a series of possible reactions predicted with B that start with molecules in M and ultimately lead to synthesis of t.

Note: Practical constraint (efficiency)

- The number of calls to one-step model B should be limited.

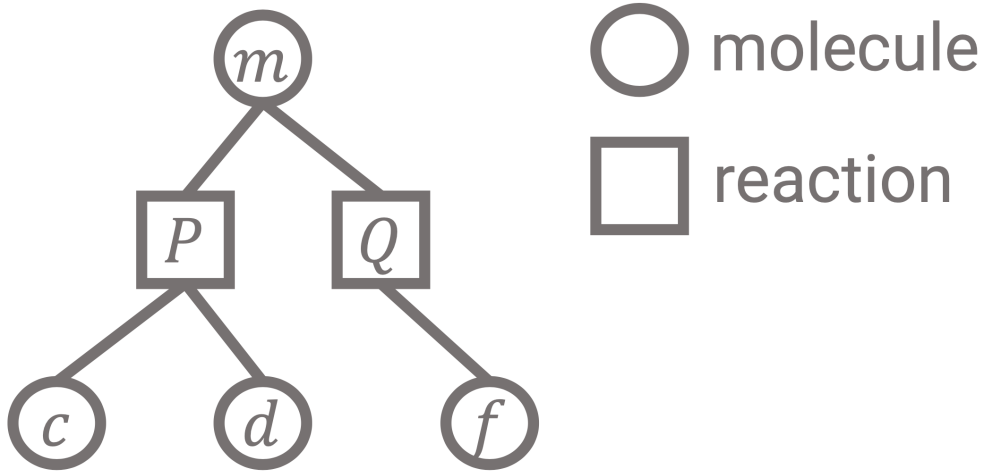
Retro*: Contributions



- First algorithm to learn from previous planning experience.
- State-of-the-art performance on a real-world benchmark dataset.
- Able to induce a search algorithm that guarantees finding the optimal solution.

Retro*: AND-OR Tree Representation

- Each molecule is encoded as an `OR` node (like m), requiring at least one of its children to be ready.
- Each reaction is encoded as an `AND` node (like P), requiring all children to be ready.
- All building blocks are ready.
- Solution found when the root is ready.

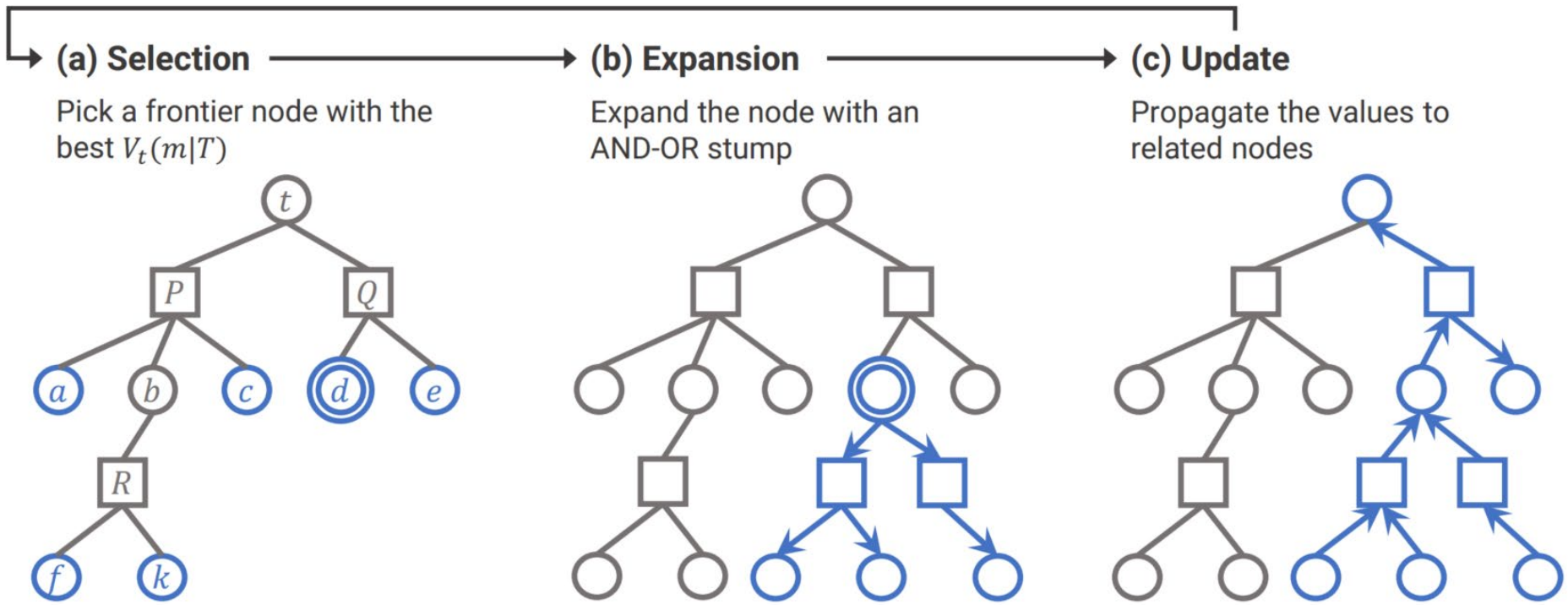


Example:

Reaction $\boldsymbol{P}$: $molecule\ \boldsymbol{c} + molecule\ \boldsymbol{d} \rightarrow molecule\ \boldsymbol{m}$.

Reaction $\boldsymbol{Q}$: $molecule\ \boldsymbol{f} \rightarrow molecule\ \boldsymbol{m}$.

Retro*: Algorithm Framework



Key Idea: Prioritize the synthesis of the molecules in the current *best plan*.
Definition of $V_t(m|T)$: under the current search tree T, the cost of the current best plan containing m for synthesizing target t.

Retro*: Computing $V_t(m|T)$ via Tree-DP

By decomposing $V_t(m|T)$ into simpler components in a recursive fashion, we can compute its value efficiently via tree-structured dynamic programming.

(1) Boundary case: $\boldsymbol{V_m \equiv V_m(m|\emptyset)}$, the cost of synthesizing frontier node m.

(2) Define *reaction number* $\boldsymbol{rn(\cdot\,|T)}$: minimum estimated cost needed for a molecule/reaction to happen in the current tree.

$$rn(R|T) = c(R) + \sum_{m \in ch(R)} rn(m|T)$$

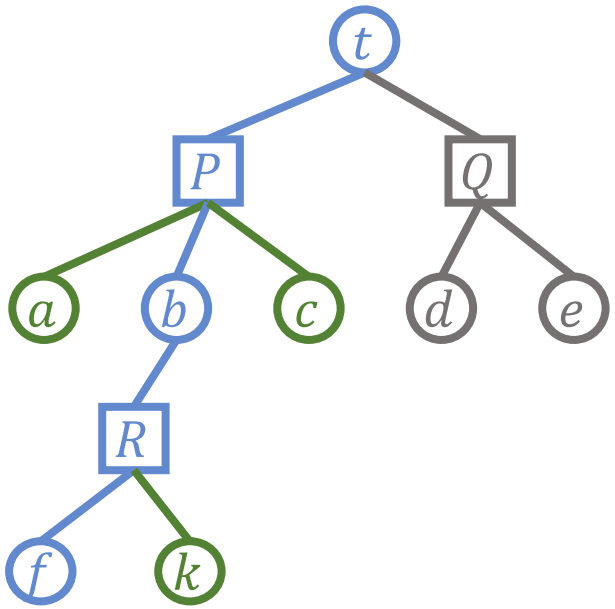
$$rn(m|T) = \begin{cases} V_m, & m \in \mathcal{F}(T) \\ \min_{R \in ch(m)} rn(R|T), & \text{otherwise} \end{cases}$$

(3) Compute $\boldsymbol{V_t(m|T)}$ with $\boldsymbol{rn(\cdot\,|T)}$.

$$V_t(m|T) = \sum_{r \in \mathcal{A}(m|T) \cap \mathcal{V}^r(T)} c(r) + \sum_{m' \in \mathcal{V}^m(T), pr(m') \in \mathcal{A}(m|T)} rn(m'|T)$$

Retro*: Example for Computing $V_t(m|T)$

We learn $V_m \equiv V_m(m|\emptyset)$, the cost of synthesizing m.



$$rn(t|T) = \min\big(rn(P|T), rn(Q|T)\big)$$

$$rn(Q|T) = c(Q) + V_d + V_e$$

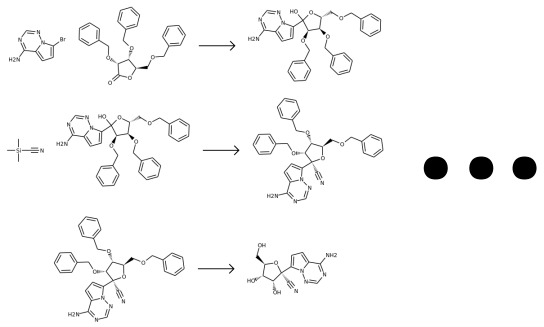
$$V_t(f|T) = \underbrace{c(P) + c(R)}_{g_t(f|T)} + \underbrace{V_a + V_c + V_f + V_k}_{h_t(f|T)}$$

A^* algorithm!

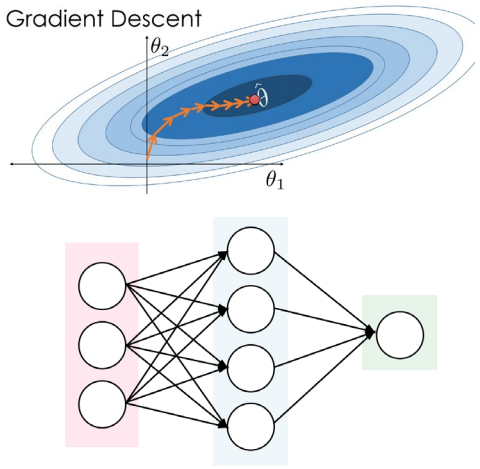
A^* Admissibility: guarantee finding an optimal solution if V_m is a lower-bound!

Note: 0 is the lower-bound of V_m for any molecule m if $c(R) = -\log \mathrm{Prob}(R) \geq 0$.

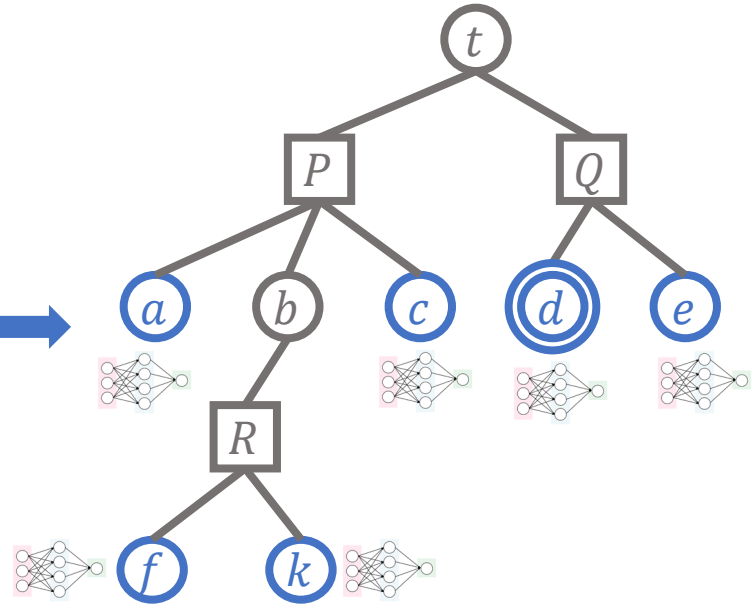
Retro*: Learning to Plan



Collect planning data



Learn V_m



Plan with learned model

Retro*: Training Objective

Dataset: $\mathcal{R}_{train} = \{rt_i = (m_i, v_i, R_i, B(m_i))\}$ each tuple contains target molecule m_i, best synthesis cost v_i, expert reaction R_i, and one-step retrosynthesis candidates $B(m_i)$.

Optimize

$$\min_{V_{(\cdot)}} \quad \mathbb{E}_{rt_i \sim \mathcal{R}_{train}} \Big[\mathcal{L}_{reg}(rt_i) + \lambda \mathbb{E}_{R_j \sim B(m_i) \setminus \{R_i\}} \left[\mathcal{L}_{con}(rt_i, R_j) \right] \Big]$$

Regression loss: $\mathcal{L}_{reg}(rt_i) = (V_{m_i} - v_i)^2$

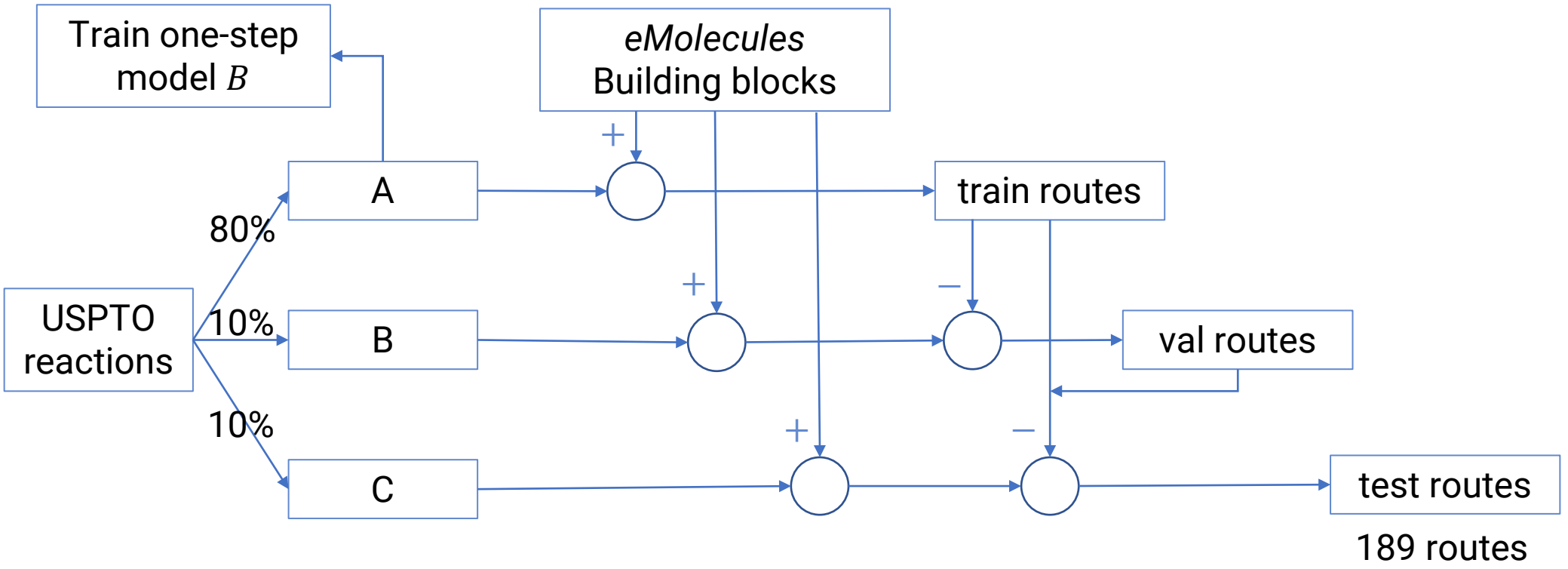
Consistency loss:

$$\mathcal{L}_{con}(rt_i, R_j) = \max \left\{ 0, v_i + \epsilon - c(R_j) - \sum_{m' \in \mathcal{S}_j} V_{m'} \right\}$$

Constraint:

$$c(R_j) + \sum_{m' \in S_j} V_{m'} > v_i + \epsilon$$

Exp: Creating Benchmark Dataset



Exp: Baselines & Evaluation

Baselines:

- **Greedy** - greedy Depth First Search: prioritize the reaction with the highest likelihood.
- **MCTS** - Monte-Carlo Tree Search (*Segler et al.*, 2018).
- **DFPN-E** - a variant of Proof Number Search (*Kishimoto et al.*, 2019).
- **Retro*-0** - obtained by setting V_m to a lower-bound, 0 (ablation study).

Evaluation:

- Time: number of calls to the one-step model ($\approx 0.3s$ per call, occupying $> 99\%$ time).
- Solution quality: total costs of reactions / number of reactions (length).

Exp: Results

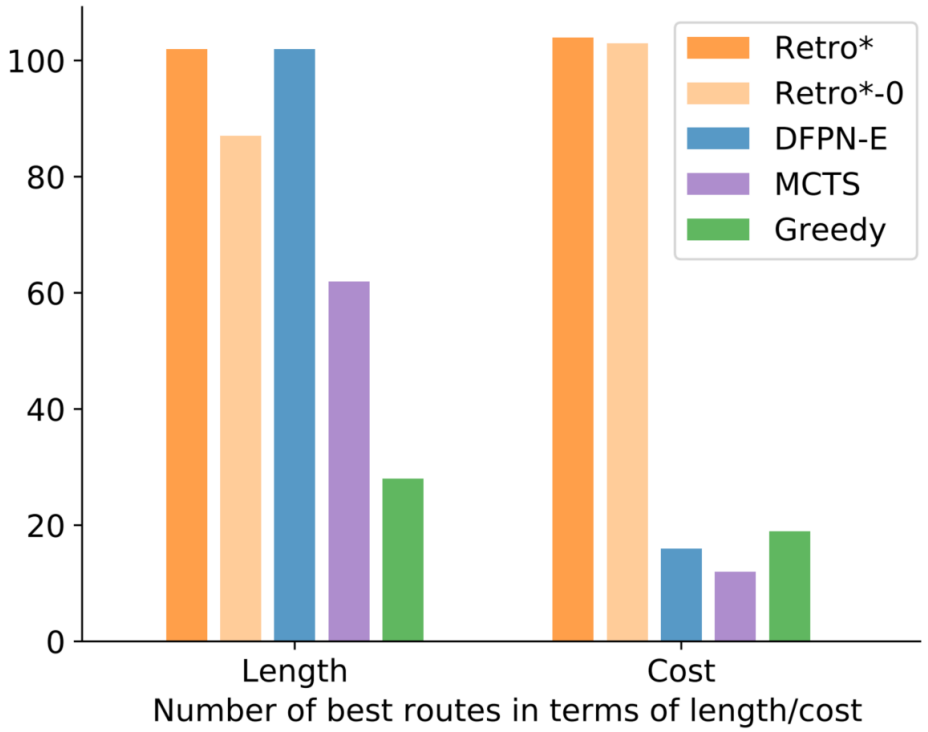


Figure: Counts of the best solutions among all algorithms in terms of length/cost.

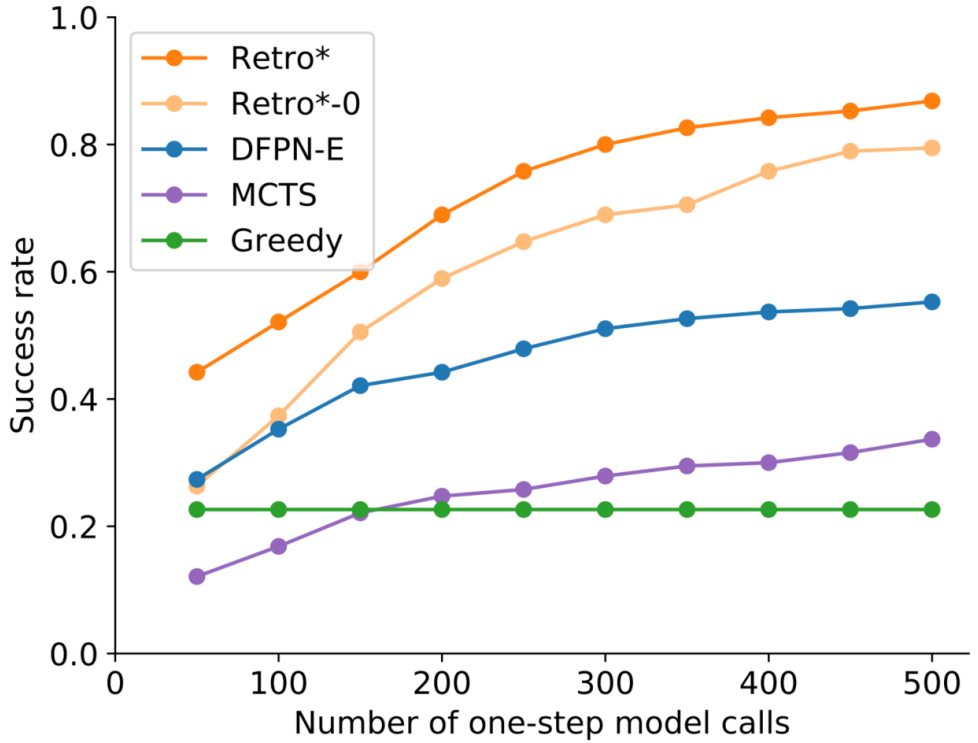


Figure: Influence of time on performance.

Exp: Sample Solution

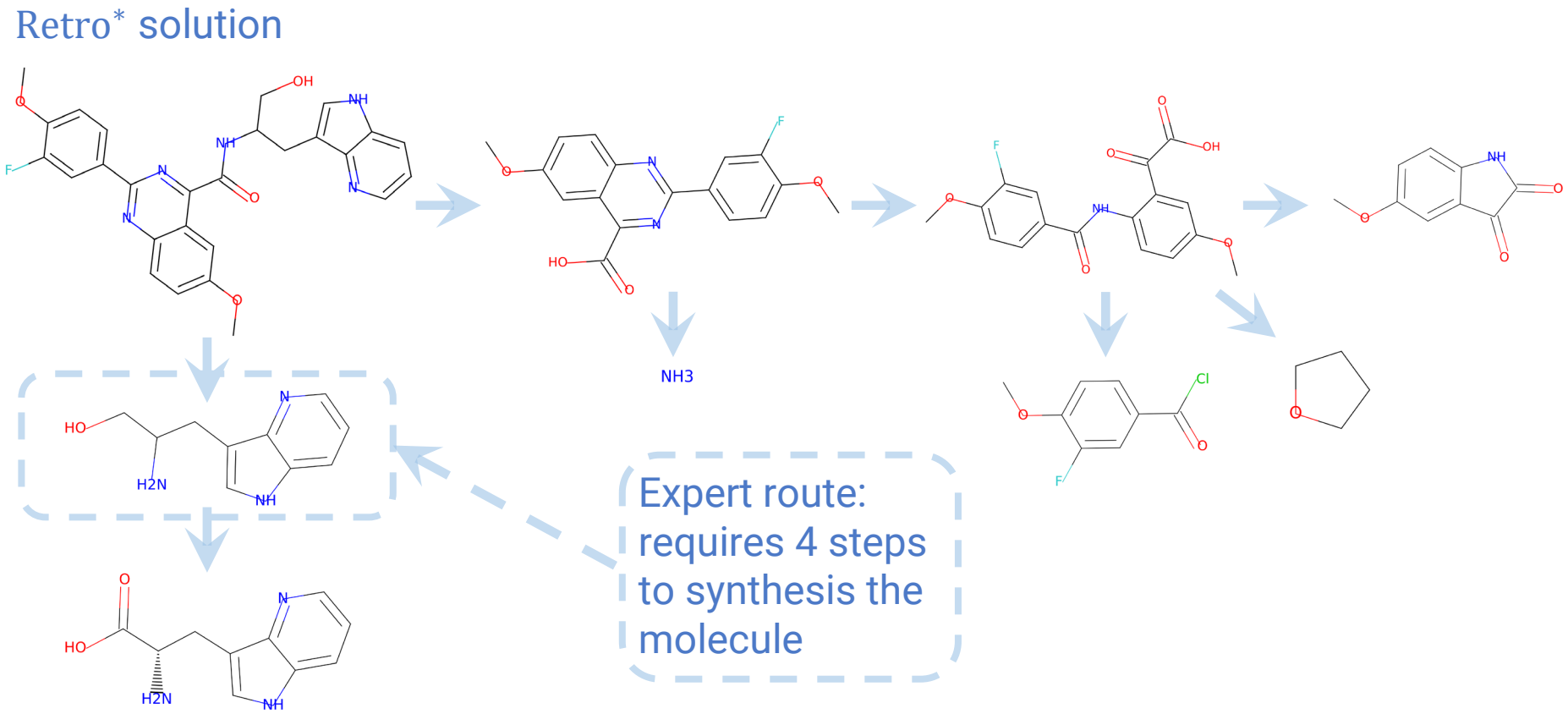


Figure: Sample solution route produced by Retro*. Expert route requires 3 more steps to synthesize one molecule in the route.

Future Work

Learning to plan in theorem proving

- Same search space

Polymer retrosynthesis

- More patterns and constraints
  - Chain reaction
- No polymerization dataset
  - Transfer knowledge from small molecules

Thanks for listening!

For more details, please refer to our paper/full slides/poster:

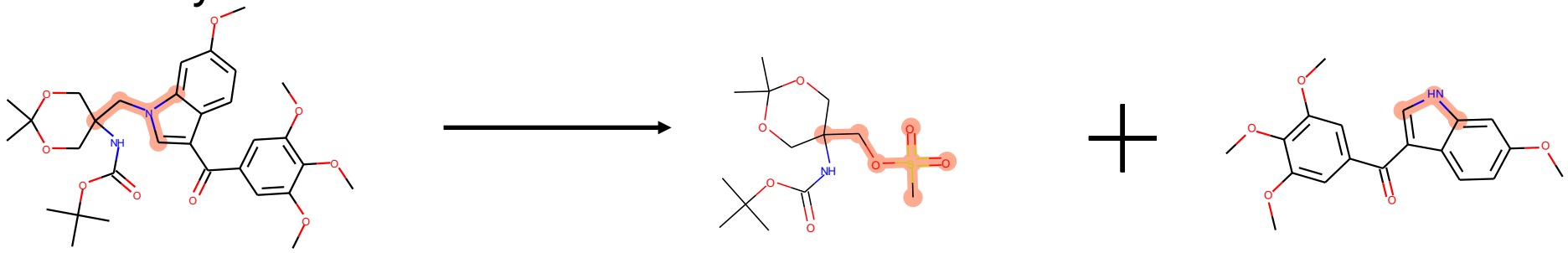
Paper

Full Slides

Poster

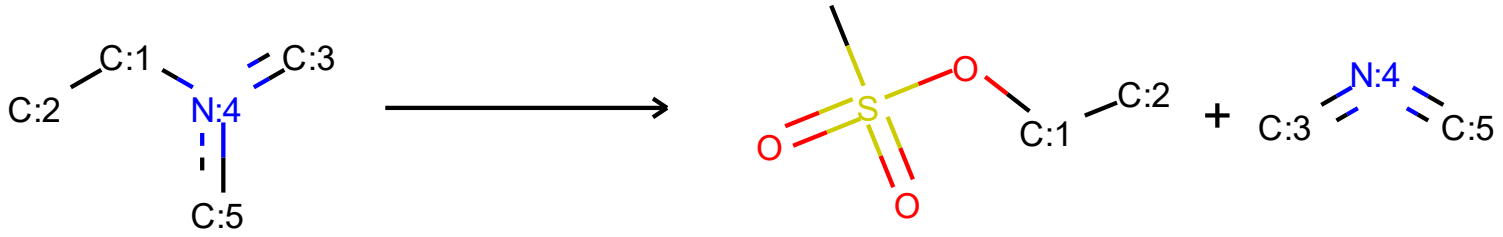
Background: Reaction Template

- Known Retrosynthesis



Product C Reactant A Reactant B

- Template: subgraph (i.e., reaction core highlighted in red) rewriting rules



- Given a product, how to apply template?

RDKit's **runReactants** will produce list of precursors

One-step Retrosynthesis

- Template-based approach
  - Graph Logic Network
  - Dai, Hanjun, et al. "Retrosynthesis Prediction with Conditional Graph Logic Network." *Advances in Neural Information Processing Systems*. 2019.
- Template-free approach
  - Seq2seq models
  - Karpov, Pavel, Guillaume Godin, and Igor V. Tetko. "A transformer model for retrosynthesis." *International Conference on Artificial Neural Networks*. Springer, Cham, 2019.

Existing Planners

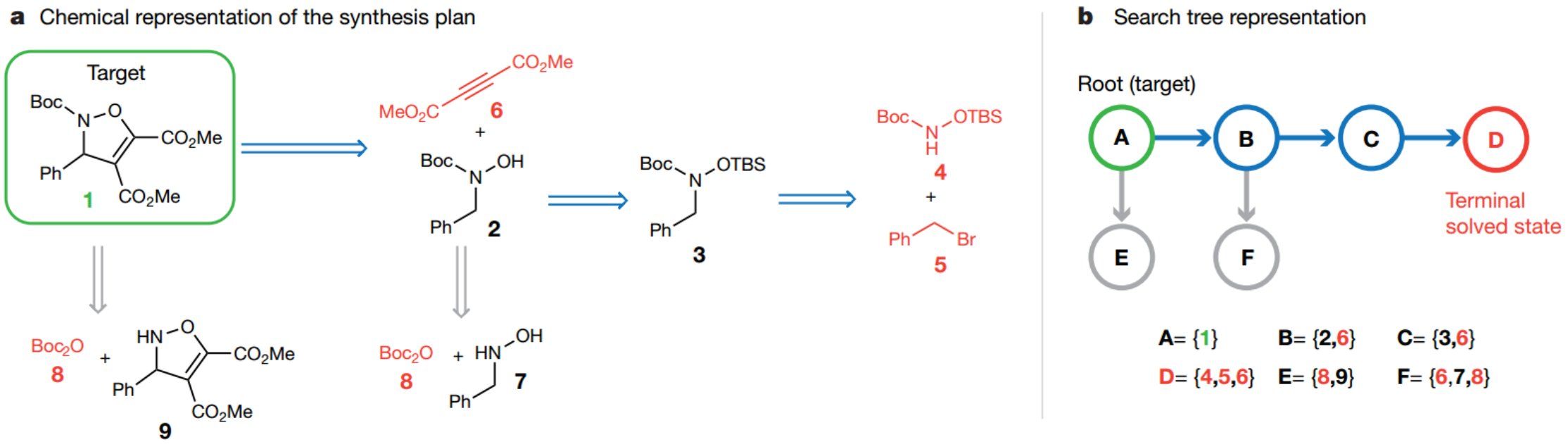
Monte Carlo Tree Search

Segler, Marwin HS, Mike Preuss, and Mark P. Waller. "Planning chemical syntheses with deep neural networks and symbolic AI." *Nature* 555.7698 (2018): 604-610.

Proof Number Search

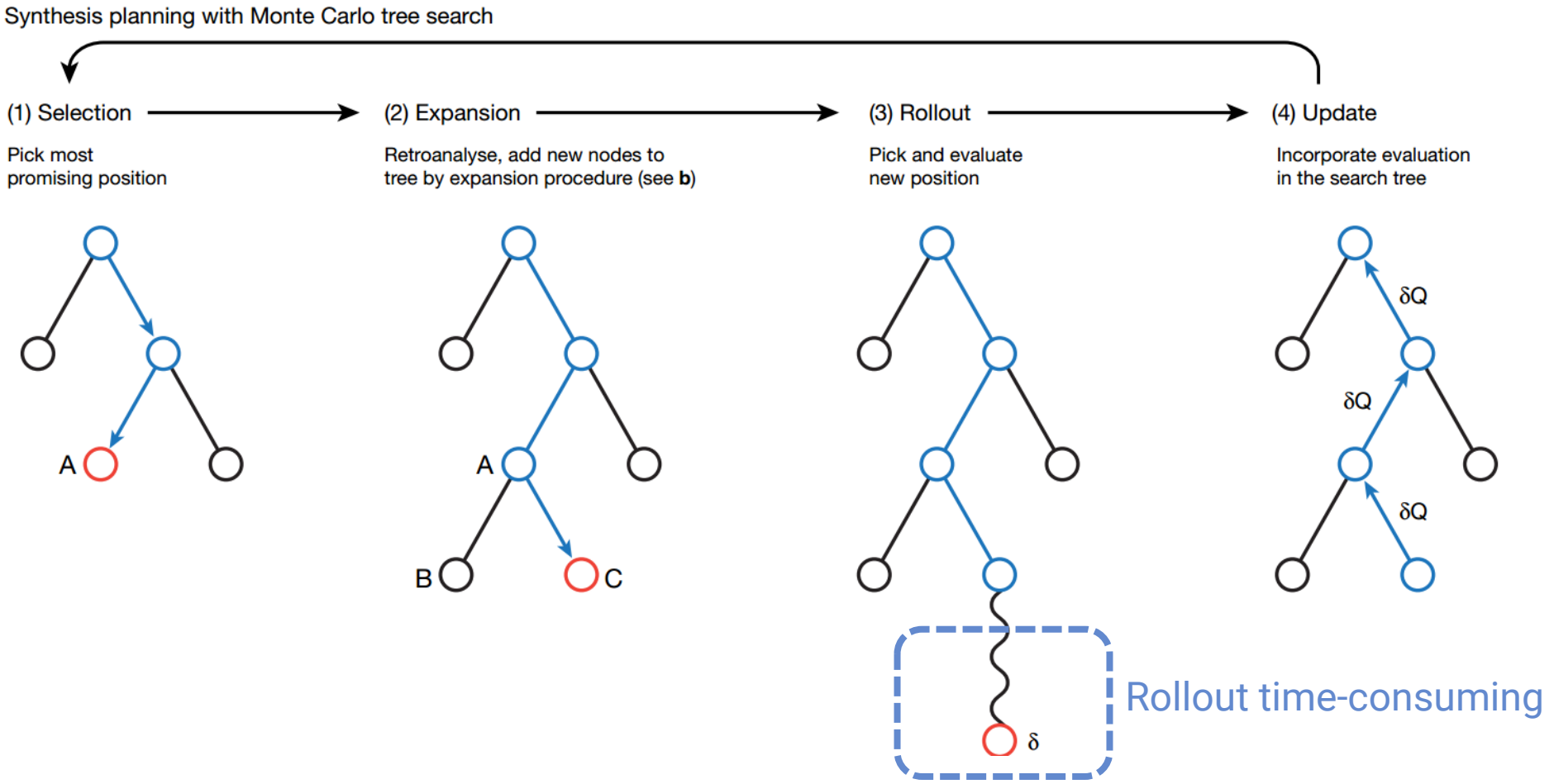
Kishimoto, Akihiro, et al. "Depth-First Proof-Number Search with Heuristic Edge Cost and Application to Chemical Synthesis Planning." *Advances in Neural Information Processing Systems*. 2019.

Existing Planner – MCTS



Search Tree Representation

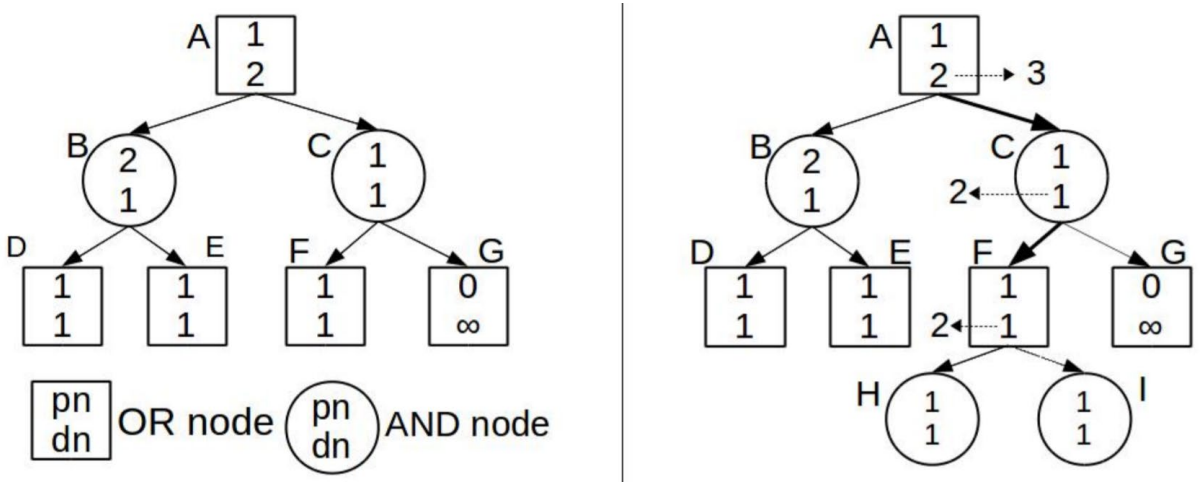
Existing Planner – MCTS



Algorithm framework

Existing Planner – PNS

- Formulate the retrosynthesis problem as a two-player game.
- Optimize for quickly finding one route.
- Require hand-designed criterion during search.

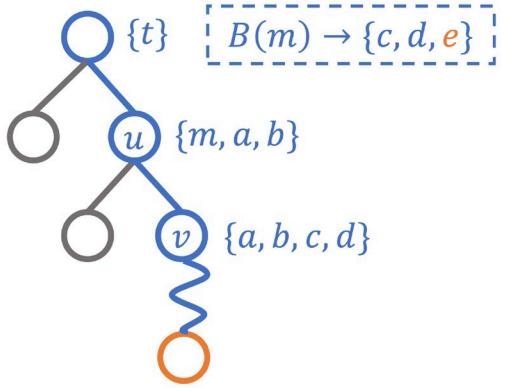


Search Tree Representation

Existing Planners Summary

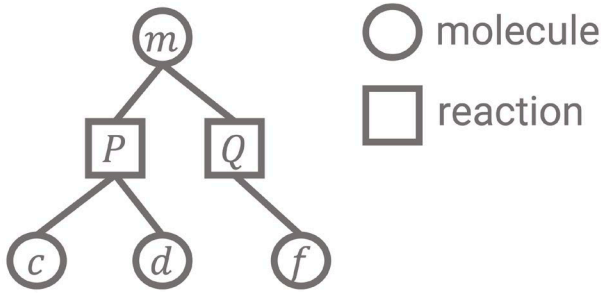
Monte Carlo Tree Search

- Rollout time-consuming and comes with high variance.
- Sparsity in variance estimation.
- Not optimized for total costs.



Proof Number Search

- Formulation mismatch.
- Hand-designed criterion during search, hard to tune and generalize.
- Not optimized for total costs.



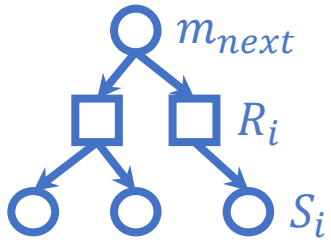
Algorithm Framework

Algorithm 1: $\texttt{Retro}^*(t)$

Initialize $T = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} \leftarrow \{t\}, \mathcal{E} \leftarrow \emptyset$;
while *route not found* **do**
  $m_{next} \leftarrow \text{argmax}_{m \in \mathcal{F}(T)} V_t(m)$;
  $\{R_i, \mathcal{S}_i, c(R_i)\}_{i=1}^{k} \leftarrow B(m_{next})$;
  for $i \leftarrow 1$ **to** k **do**
    Add R_i to T under m_{next};
    for $j \leftarrow 1$ **to** $|\mathcal{S}_i|$ **do**
      Add $\mathcal{S}_{ij}$ to T under R_i;
  Update $V_t(m)$ for m in $\mathcal{F}(T)$;
return route;

(a) Select the most promising frontier node

(b) Expand the node with one-step model



(c) Update current estimate of V function

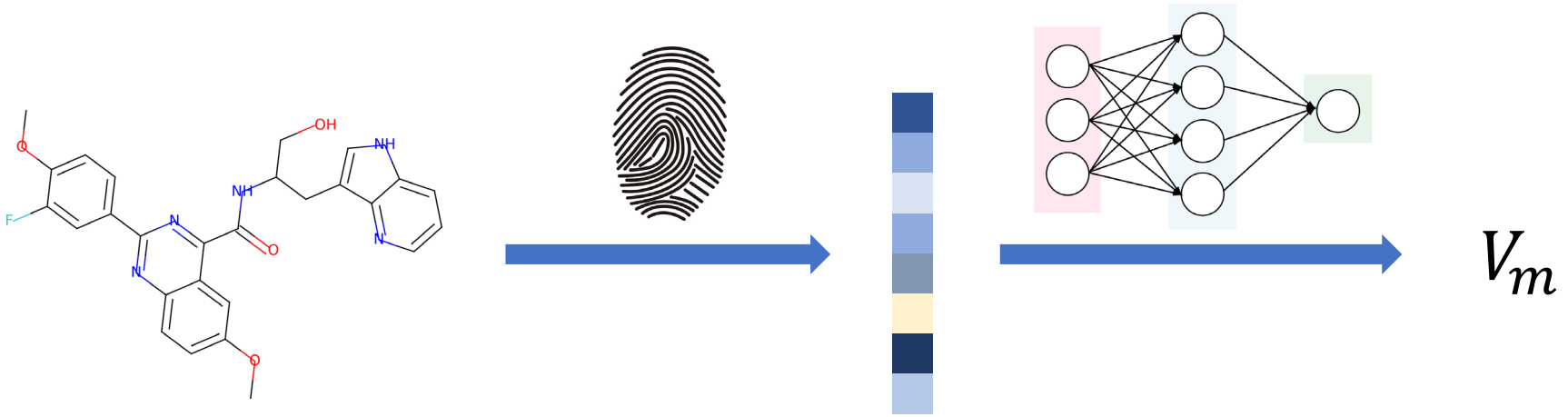
Retro*: Theoretical Guarantees

Theorem 1 *Assuming V_m or its lowerbound is known for all encountered molecules m, Algorithm 1 is guaranteed to return an optimal solution, if the halting condition is changed to "the total costs of a found route is no larger than* $\text{argmin}_{m \in \mathcal{F}(T)} V_t(m)$*".*

Proof
Similar to A^* admissibility proof.

Remark
0 is the lower-bound of V_m for any molecule m if cost is defined as the negative log-likelihood.

Retro*: Representing Value Function V_m



molecule fingerprint value prediction

One-step Model Training

$$B(\cdot): \quad t \rightarrow \{R_i, \mathcal{S}_i, c(R_i)\}_{i=1}^{k}$$

- Template-based MLP model.
- There are ~380K distinct templates.
- Multi-class classification.
- Predicts top-50 reaction templates for each target.
- Apply templates to obtain the corresponding reactants.

- Trained on training reaction set.

- Cost defined as the negative log-likelihood of the predicted reaction.
- Minimize cost = maximize likelihood.

Exp: Results (2)

Algorithm	Retro*	Retro*-0	DFPN-E	MCTS	Greedy DFS
Success rate	86.84%	79.47%	55.26%	33.68%	22.63%
Time	156.58	208.58	279.67	380.02	388.15
Shorter routes	50	52	59	30	11
Better routes	112	102	25	18	26

Performance Table: The number of shorter and better routes are obtained from the comparison against the expert routes, in terms of length and total costs.