AI in Mathematics
Lecture 12
Reinforcement Learning

Bar-Ilan University
Nebius Academy | Stevens Institute of Technology
June 17, 2025

About This Course

~~1 week: Intro~~

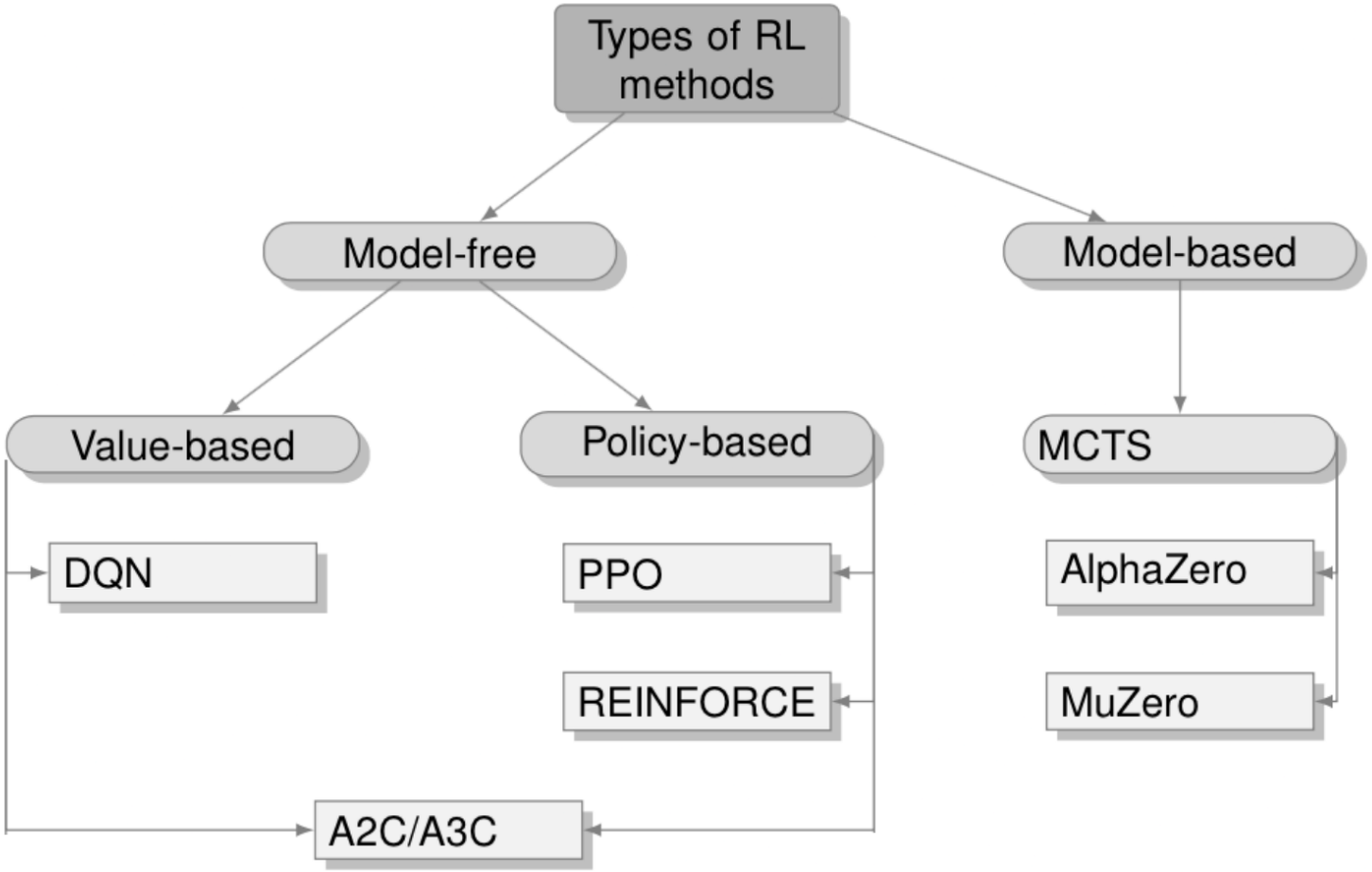
~~2 weeks: Classic ML~~

~~2 weeks: Deep Learning in Mathematics~~

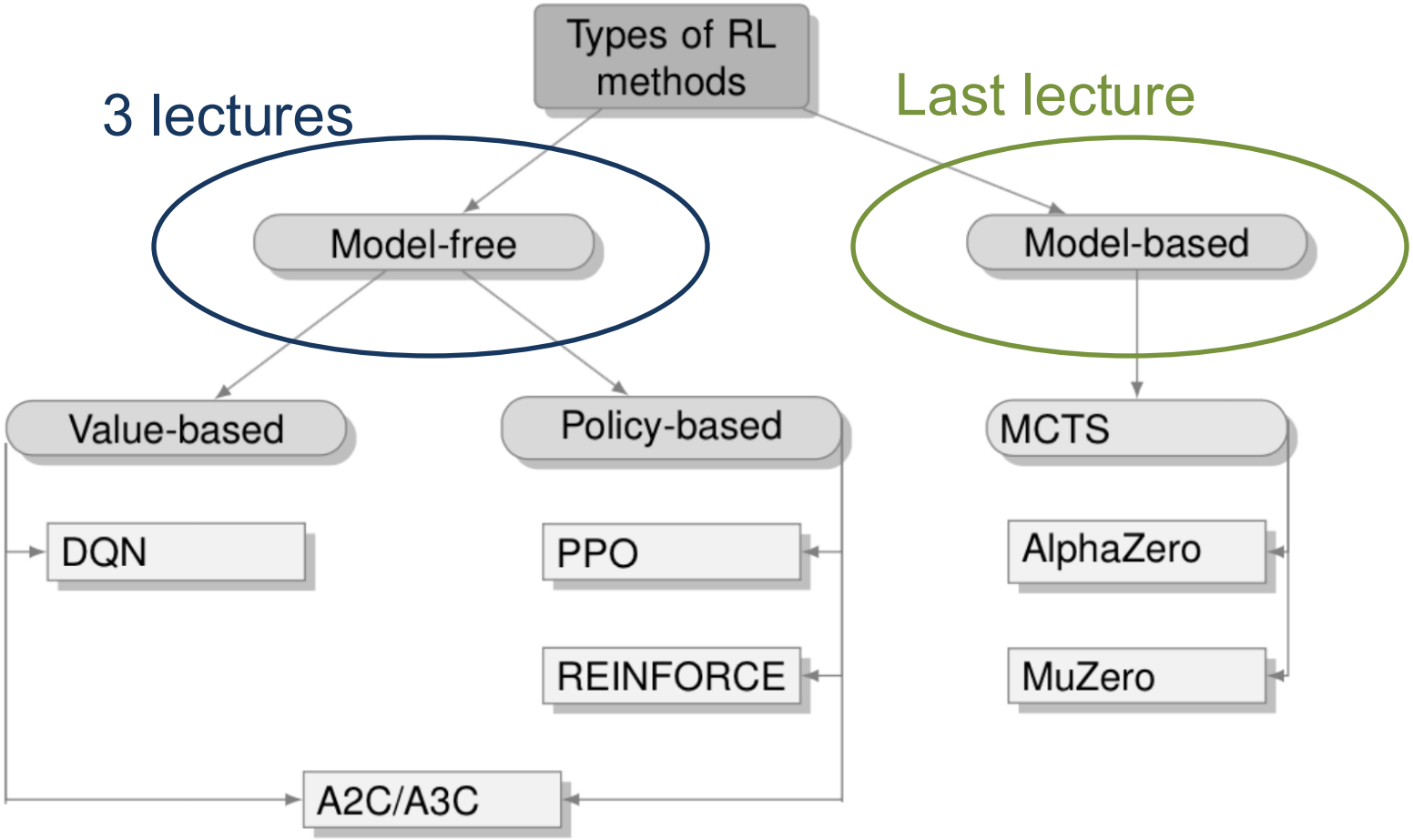
~~4 weeks: Math as an NLP problem (LLMs etc.)~~

4 weeks: Reinforcement Learning (RL) in Math

RL Algorithms



RL Algorithms



Policy Gradient

$$\mathcal{J}(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$$

$$Q(s_t, a_t) = \sum_{t' \geq t} \gamma^{t'-t} R(s_{t'}, a_{t'})$$

$$\nabla_\theta \mathcal{J}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_{\theta_k} \ln \pi_{\theta_k}(a|s)\, Q(s,a)\right]$$

Variance problem

$$\nabla_\theta \mathcal{J}(\pi_\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \ln \pi_\theta(a|s)\,(Q(s,a) - b(s))]$$

$b(s)$ is some function of the state that allows us to reduce variance.

One good solution is to use $b(s) = V^{\pi_\theta}(s) = \mathbb{E}_a Q(s,a)$. It is exactly what A2C will do.

So $V^{\pi_\theta}(s)$ represents how good or promising this state is on average, under the future path defined by π.

A2C

$$\nabla_\theta \mathcal{J}(\pi_\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \ln \pi_\theta(a|s)\, A(s,a)]$$

where the *advantage* function is defined as

$A(s,a) = Q(s,a) - V^{\pi_\theta}(s),\ V^{\pi_\theta}(s) = \mathbb{E}_a Q(s,a).$

To approximate A, we will use

$$A_t(s,a) = r_t + \gamma\, V(s_{t+1}) - V(s_t)$$

This is a **one-step temporal difference (TD) estimate** of the advantage — efficient and requires minimal storage.

1. The **value function** V(s) is learned using a neural network.
2. The **advantage estimate** A_t uses only the **immediate reward** and **next state value** — no need to store full trajectories.

A2C

Collect Trajectories

1. Interact with the environment using the current policy π_θ.
2. Collect (s_t, a_t, r_t, s_{t+1}) for multiple time steps (usually in parallel environments).

Use a neural network (critic) to predict $V_\phi(s) \approx V^\pi(s)$.

Use 1-step TD estimate:

$$A_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

Update Policy (Actor):

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta \ln \pi_\theta(a_t \mid s_t) \cdot A_t$$

Update Critic using the loss function:

$$L_{critic} = \left(r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t) \right)^2$$

A2C (Detailed)

Collect Trajectories
(Shared Step)

- Interact with the environment using the current policy π_θ.
- Collect transitions (s_t, a_t, r_t, s_{t+1}) over multiple steps (e.g., from parallel environments)
- Use a neural network (Critic) to predict $V_\phi(s) \approx V^\pi(s)$.
- Compute 1-step TD Advantage estimate:

$$A_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

What Actually Happens:

We **collect a batch** of transitions (s_t, a_t, r_t, s_{t+1}) from multiple environments or steps.

Then compute:

- **Batch of value predictions** $V(s_t), V(s_{t+1})$
- **Batch of advantages**

$$A_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

These batches are then used to perform **gradient updates**

A2C (Detailed)

Actor Update

Policy Improvement
The actor updates the policy parameters θ to maximize expected return.

Gradient ascent step:

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta \ln \pi_\theta(a_t \mid s_t) \cdot A_t$$

Critic Update

The critic learns to estimate the **state value** $V^\pi(s) \approx V_\phi(s)$

Loss function:

$$L_{critic} = \left(r_t + \gamma V\phi(s_{t+1}) - V_\phi(s_t)\right)^2$$

Gradient descent step:

$$\phi \leftarrow \phi - \eta \cdot \nabla_\phi L_{critic}$$

In both cases we can use stochastic gradient descent to update parameters.

Reducing Variance: Generalized Advantage Estimation (GAE)

TD residual: $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

GAE: $\hat{A}_t = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}$

$\lambda = 1$ close to REINFORCE approach

$\lambda = 0$ correspond to TD(0) approach

Importance Sampling

It is an idea coming from off-policy algorithms.

If we collect data with policy $\pi_{\theta_{old}}$ how do we train a new policy π_θ?

$$\mathbb{E}_{a\sim\pi_\theta(a|s)}[f(a)] = \sum_a \pi_\theta(a|s) f(a)$$

$$= \sum_a \pi_\theta(a|s) \frac{\pi_{\theta_{old}}(a|s)}{\pi_{\theta_{old}}(a|s)} f(a)$$

$$= \mathbb{E}_{a\sim\pi_{\theta_{old}}(a|s)} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} f(a) \right]$$

Importance Ratio

$$r(\theta) = \frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)}$$

This idea allows us to sample trajectories less often and reuse them with a correct approximation under the new policy.

This cheating only works if $r(\theta)$ is close to 1.

PPO (Proximal Policy Optimization)

PPO implements the idea of importance sampling and carefully applies it by clipping the loss function with respect to the importance ratio.

$$L_t^{clip}(\theta) = \min(r_t(\theta)\hat{A}_t,\ clip(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon)\hat{A}_t)$$

Where $r_t(\theta) = \frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)}$,

$\hat{A}_t = \sum_{l=0}^{\infty}(\gamma\lambda)^l\ \delta_{t+l}$ – GAE.

Value function update

We minimize the squared error between the predicted value and a fixed target:

$$L_t^V = \left(V_\phi(s_t) - R_t \right)^2$$

Where the target return is:

$$R_t = \hat{A}_t + V_\phi^{old}(s_t)$$

Reminder: $V_\phi(s_t)$ approximates $V^\pi(s_t) \approx \mathbb{E}_\pi[\sum_{i=0}^{\infty} \gamma^i r_{t+i}]$.

$V_\phi^{old}(s_t)$ has frozen parameters and we do not propagate a gradient through it.

Entropy bonus (Optional)

$$S_{\pi_\theta} = -\sum_a \pi_\theta(a|s_t) \log \pi_\theta(a|s_t)$$

- When $S(\pi_\theta)$ is **close to 0** → the policy is nearly **deterministic** (one action dominates).

- When $S(\pi_\theta)$ is **high** → the policy is more **stochastic**, promoting exploration.

Total Loss

$$L = -L^{clip} + c_1 L^V - c_2 S_{\pi_\theta}$$

This loss is composed of three parts:

- L^{clip}: Clipped surrogate objective for the **actor** (policy update)
- L^V: Value function loss for the **critic**
- $S(\pi_\theta)$: Policy entropy to encourage **exploration**

Algorithm

- **Collect Trajectories**

  Run the current policy π_θ to gather (s_t, a_t, r_t, s_{t+1}) across multiple steps and trajectories.
- **Estimate Advantages**

  Use **Generalized Advantage Estimation (GAE)** to compute $\hat{A}_t$ from collected rollouts.
- **Compute Loss**
- **Update Parameters**

  Use backpropagation to update both **actor** and **critic** networks.

Application

Gukov et al. (2025) – "What Makes Math Problems Hard for Reinforcement Learning: A Case Study"

A paper that advances and deepens the application of reinforcement learning to mathematics, while also delivering concrete results.

Andrews-Curtis Conjecture

Conjecture [J.Andrews and M.Curtis '65]:

Every balanced presentation of the trivial group

$$\langle x_1, \dots, x_n \mid r_1, \dots, r_n \rangle \ (r_i = e \ is \ a \ relation)$$

can be reduced to the trivial presentation

$$\langle x_1, \dots, x_n \mid x_1, \dots, x_n \rangle$$

by a sequence of Andrews-Curtis moves.

AC moves

$$r_i, r_j \longmapsto r_i r_j, r_j$$
$$r_i \longmapsto r_i^{-1}$$
$$r_i \longmapsto x_j^{\pm 1} r_i x_j^{\mp 1}$$

For example:

$$\langle x_1, x_2 | x_1 x_2, x_2 \rangle \to \langle x_1, x_2 | x_1, x_2 \rangle$$

Can be done in three moves.

A simple **invariant** of a presentation:
Total relator length (sum of word lengths).

Result #1

$$AK(n) = \langle x, y \,|x^n = y^{n+1}, xyx = yxy \rangle$$

For every $n \geq 2$, $AK(n)$ is AC-equivalent to the presentation

$$\langle x, y \,|x^{-1}yx = xyx^{-1}y\,, xy^{n-1}x = yxy \rangle$$

of length $n + 11$.

This gives a reduction in length of $AK(n)$ for all $n \geq 5$.

Result #2

$$MS(n, w) = \langle x, y \,|x^{-1}\, y^n x \;=\; y^{n+1}, x \;=\; w \rangle$$

These presentations are AC-trivialisable:
(i) $MS(1, w)$ for all w,
(ii) $MS(n, w^\star)$ for all $n \;>\; 0$ and $w^\star \;=\; y^{-1}xyx^{-1}$
(iii) $MS(2, w_k)$ for $w_k \;=\; y^{-k}x^{-1}yxy$
where $k \;\in \mathbb{Z}$.

Furthermore, for each fixed n > 0, all members of the 1-parameter family, $MS(n, w_k)$, parameterized by $k \;\in \mathbb{Z}$, are AC-equivalent to each other and to $AK(n)$.

THE COMPLEXITY OF BALANCED PRESENTATIONS AND THE ANDREWS–CURTIS CONJECTURE

MARTIN R. BRIDSON

Hard AC presentations

Theorem A. *For $k \geq 4$ one can construct explicit sequences of k-generator balanced presentations $\mathcal{P}_n$ of the trivial group so that*

(1) the presentations $\mathcal{P}_n$ are AC-trivialisable;
(2) the sum of the lengths of the relators in $\mathcal{P}_n$ is at most $24(n+1)$;
(3) the number of (dihedral) AC moves required to trivialise $\mathcal{P}_n$ is bounded below by the function $\Delta(\lfloor \log_2 n \rfloor)$ where $\Delta : \mathbb{N} \to \mathbb{N}$ is defined recursively by $\Delta(0) = 2$ and $\Delta(m+1) = 2^{\Delta(m)}$.

7.4. **An Example.** Let me close by writing down an explicit presentation to emphasize that the explosive growth in the length of AC-trivialisations begins with relatively small presentations. Here is a balanced presentation of the trivial group that requires more than 10^{10000} AC-moves to trivialise it. We use the commutator convention $[x, y] = xyx^{-1}y^{-1}$.

$$\langle a, t, \alpha, \tau \mid [tat^{-1}, a]a^{-1}, \quad [\tau\alpha\tau^{-1}, \alpha]\alpha^{-1},$$
$$\alpha t^{-1}\alpha^{-1}[a, [t[t[ta^{20}t^{-1}, a]t^{-1}, a]t^{-1}, a]],$$
$$a\tau^{-1}a^{-1}[\alpha, [\tau[\tau[\tau\alpha^{20}\tau^{-1}, \alpha]\tau^{-1}, \alpha]\tau^{-1}, \alpha]]\rangle.$$

Challenge

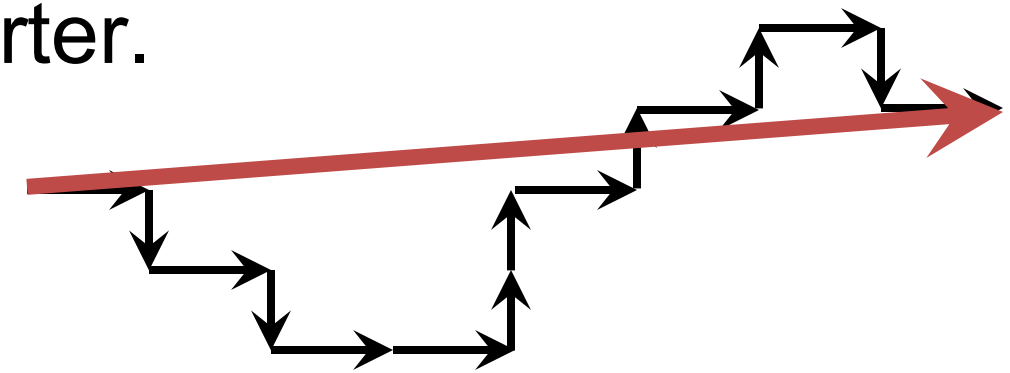
Trivializing These Presentations

We cannot rely on naive search.

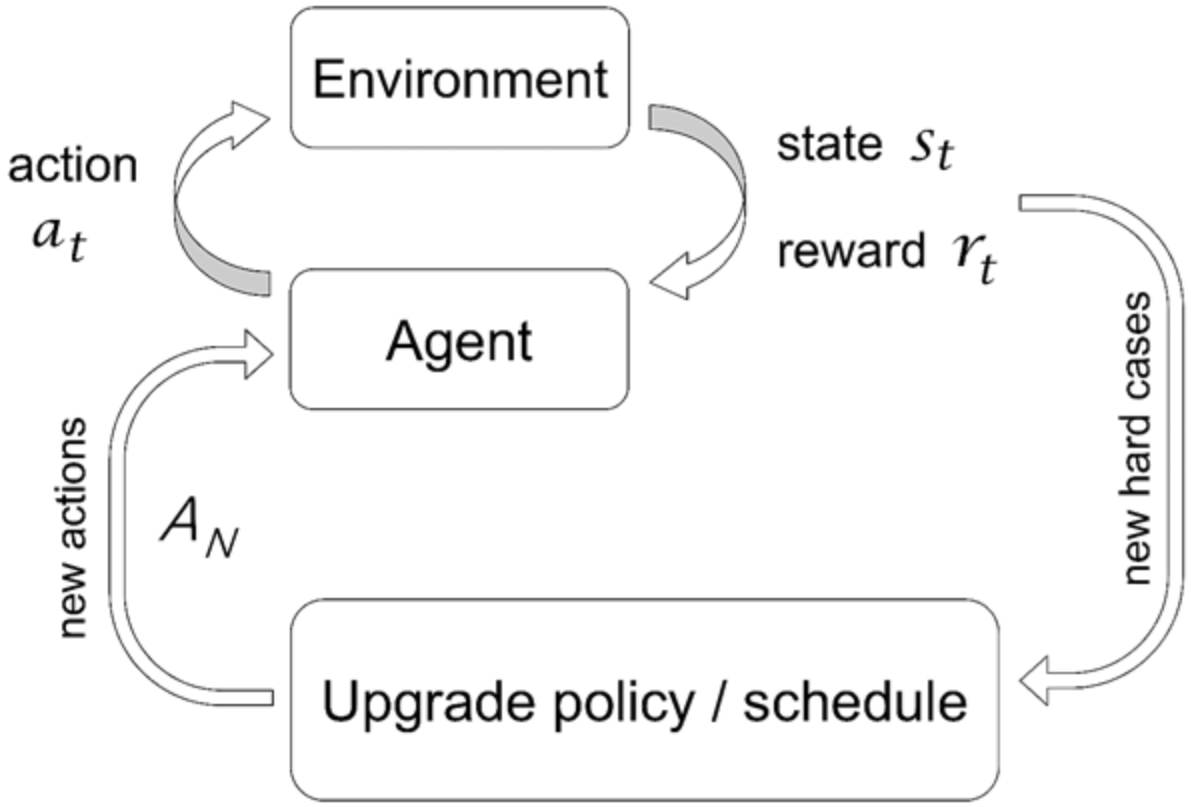
We need to develop clever strategies, structure-aware heuristics, or even machine learning to attack this problem.

Supermoves

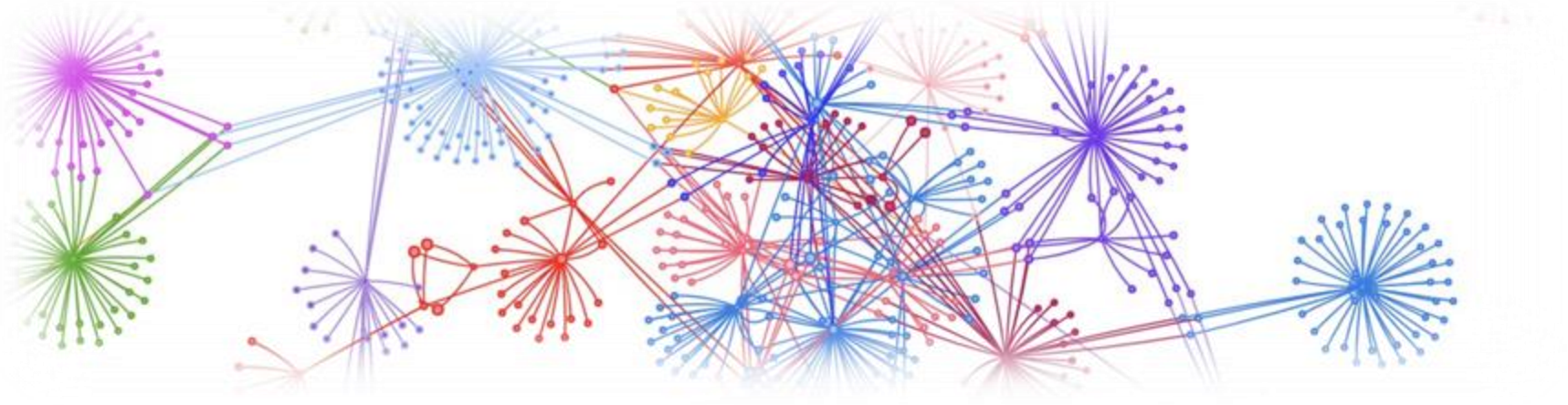
To overcome this challenge, it is natural to introduce **supermoves**—compositions of multiple elementary moves—so that, in terms of supermoves, the distance between presentations becomes significantly shorter.



Adaptive / Dynamic Action Space

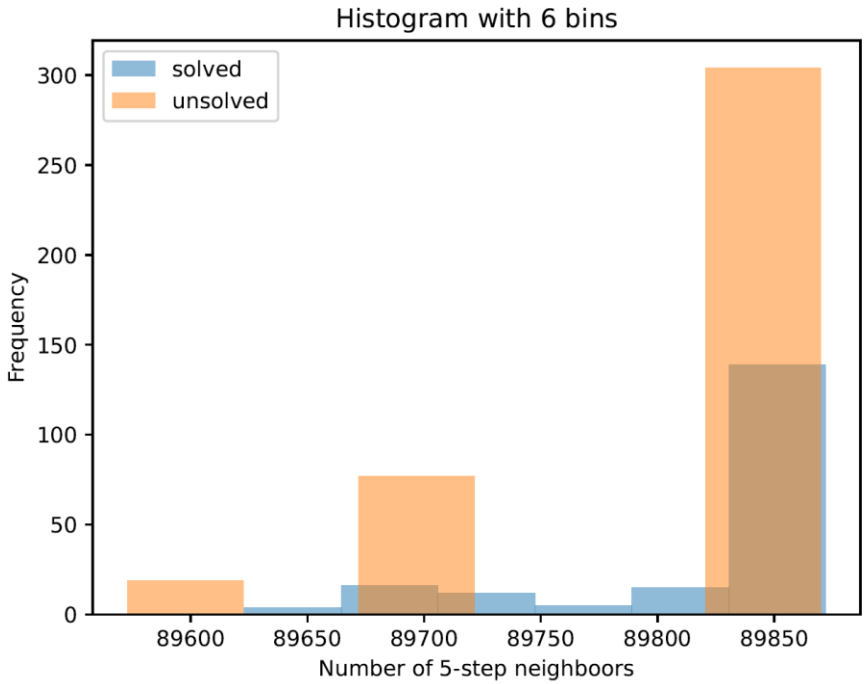


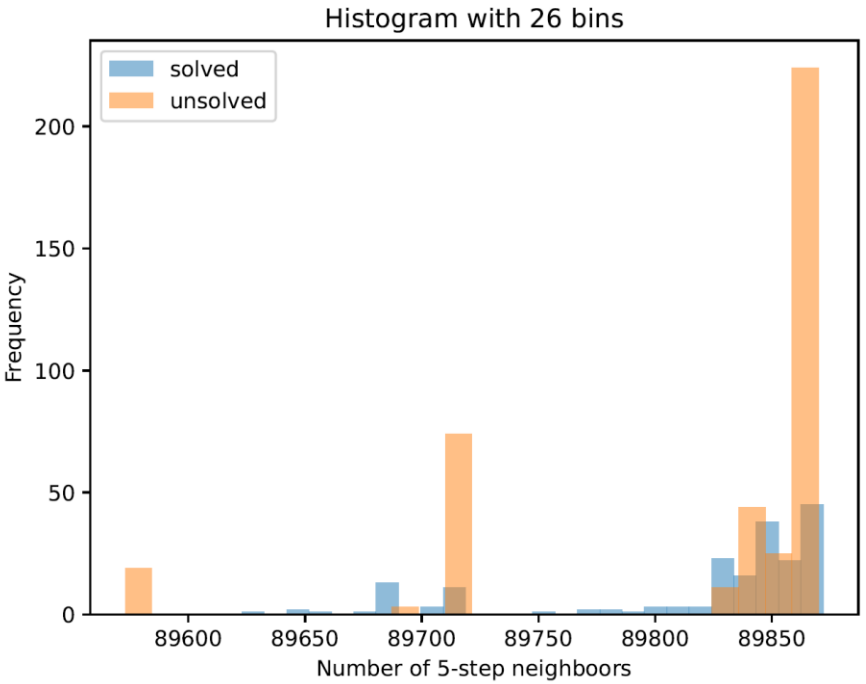
AC graph:



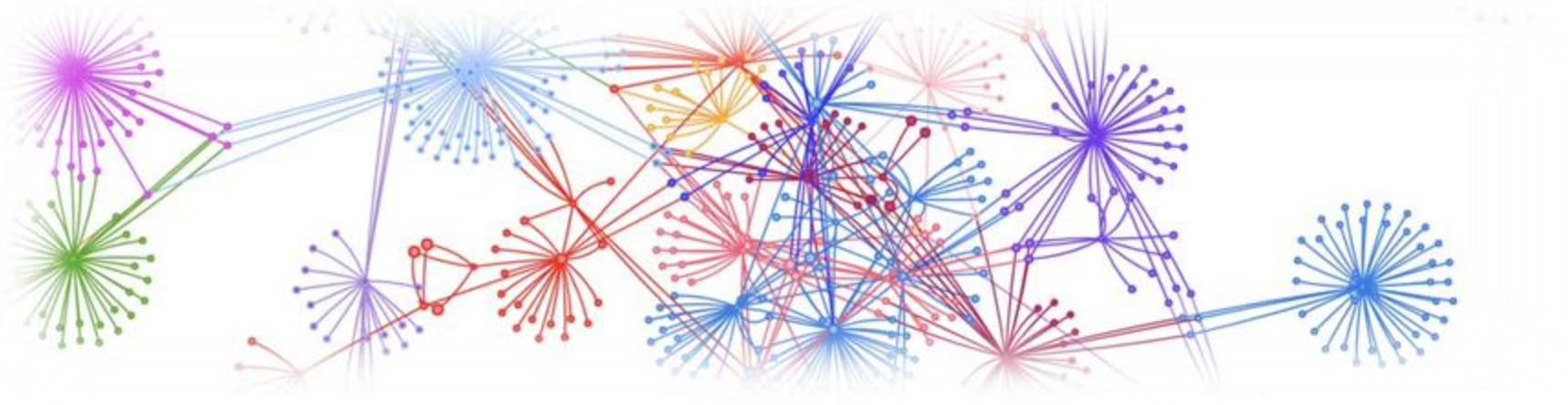
5-step neighborhood sizes:

Min	Max	Mean	Median
72,964	89,872	89,532	89,859





AC graph:



5-step neighborhood sizes:

Min	Max	Mean	Median
72,964	89,872	89,532	89,859

[89,573 -- 89,575] [89,715 -- 89,731] [89,859 -- 89,872] contain **90%** of hard AC-presentations

complement of these three bands contains **77.2%** of easy presentations

54% of easy presentations have neighborhood sizes smaller than 89,573

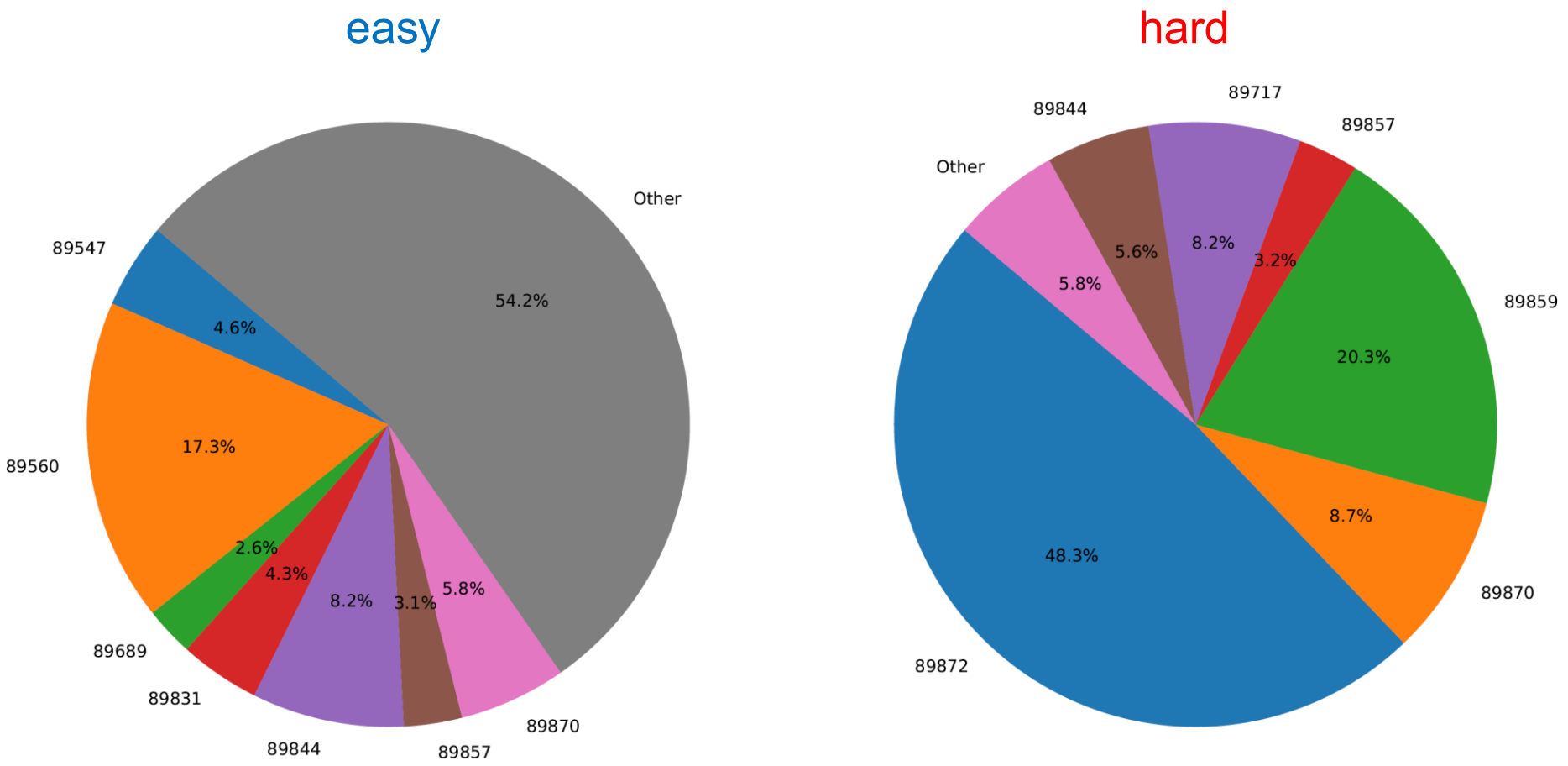


FIGURE 17. Sizes of the 5-step neighborhood of all considered presentations in the Miller–Schupp series. We group neighborhood sizes whose representation is below 2.5%.