GReinSS

Generative Modeling of Discrete Latent Structures via Dynamic Policy Gradients

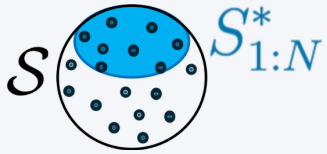
Stefan Ivanovic and **Mohammed El-Kebir**
University of Illinois Urbana-Champaign

NCI Spring School on Algorithmic Cancer Biology — Tutorial

Ivanovic et al., ICML 2026

A recurring statistical inference problem in computational biology

States $S_{1:N} \sim \Pr^*(\mathcal{S})$



Measurements $X_{1:N}$ generated from $S_{1:N}$

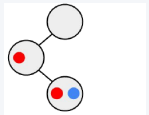
X_i :

$\mathcal{S}$ is typically **large and combinatorial** — graphs, strings, sets, …

S_i^* :

Rather than directly observing the latent **state** S we care about, we observe some indirect **measurement** X generated from it.

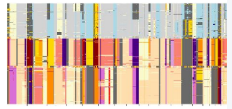
Phylogenies



State: tumor evolution tree
Measurement: DNA-seq

[Ivanovic & El-Kebir, RECOMB/Genome Res. 2023]

CNA profiles



State: copy-number profile
Measurement: read depth + BAF

[Ivanovic & El-Kebir, Genome Biol. 2025]

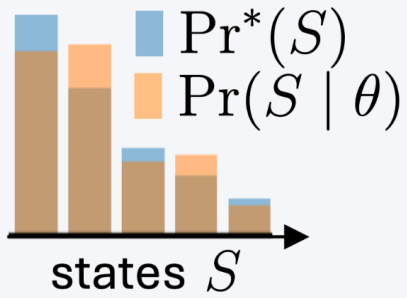
RNA isoforms

State: spliced transcript
Measurement: aligned short reads

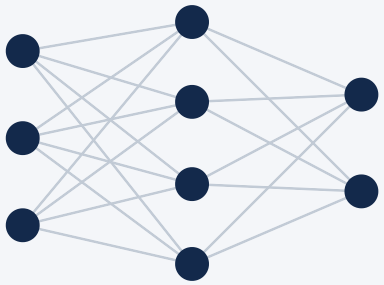
[Ivanovic et al., ICML 2026]

A learning and an inference problem

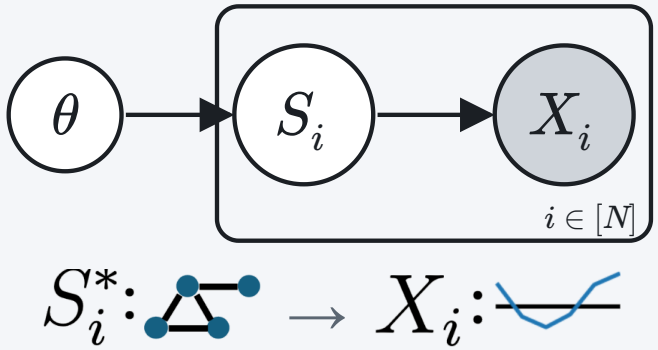
Approximate $\Pr^*(S)$ as $\Pr(S \mid \theta)$



Parameters θ: a linear map or neural network



Generative model with given $\Pr(X \mid S)$



Problem 1 (Learning). *Given (i)* $X_{1:N}$ *and (ii)* $\Pr(X \mid S)$*, find* θ *maximizing* $\Pr(X_{1:N} \mid \theta) = \prod_i \Pr(X_i \mid \theta)$*, where* $\Pr(X_i \mid \theta) = \sum_S \Pr(X_i \mid S)\ \Pr(S \mid \theta)$

Problem 2 (Inference). *Given (i)* $X_{1:N}$*, (ii)* $\Pr(X \mid S)$*, and (iii)* θ*, find* $\hat{S}_{1:N}$*, where* $\hat{S}_i = \arg\max_S \Pr(X_i \mid S)\ \Pr(S \mid \theta)$

Existing techniques

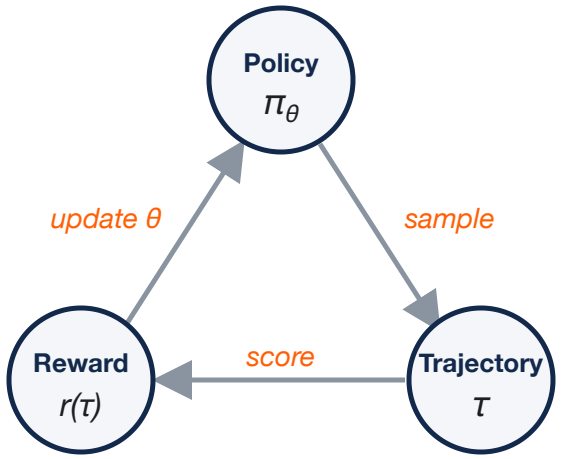
Family	Method	Why it struggles
Exact inference	**Expectation–Maximization**	E-step exact only for **special structure** (e.g. **HMMs**) — **intractable** for a combinatorial state space
Variational	**Variational inference**	maximizes an *ELBO* bound, not the likelihood — needs a **tractable posterior** over combinatorial states
	Variational autoencoders	learn *artificial* continuous latents — **not** the mechanistic state you want
Search	**Local search**	ignores the **shared** model across observations
Reinforcement learning	**Naive policy gradient**	collapses to the single **highest-reward** state
	GFlowNets	aim to **sample in proportion to a fixed reward** — not to maximize a likelihood

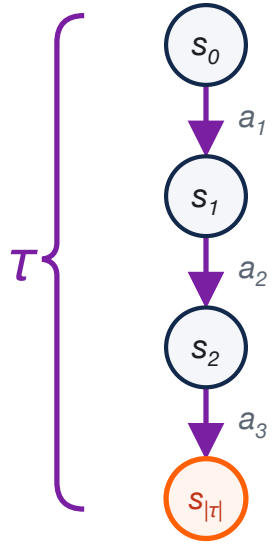
Gap: none of these directly maximize $\Pr(X_{1:N} \mid \theta)$ over a *discrete, combinatorial* state space.

Primer on reinforcement learning (RL) — fixed rewards

Key question: What actions should an agent take to maximize a reward signal?

Concept	In RL
State s_t	current situation
Action a_t	transitions $s_t \to s_{t+1}$
Policy π_θ	picks the next action
Trajectory τ	path $s_0 \to \cdots \to s_{\lvert\tau\rvert}$
Reward $r(\tau)$	scores terminal state
Objective	$\max_\theta \mathbb{E}_\tau[r(\tau)]$





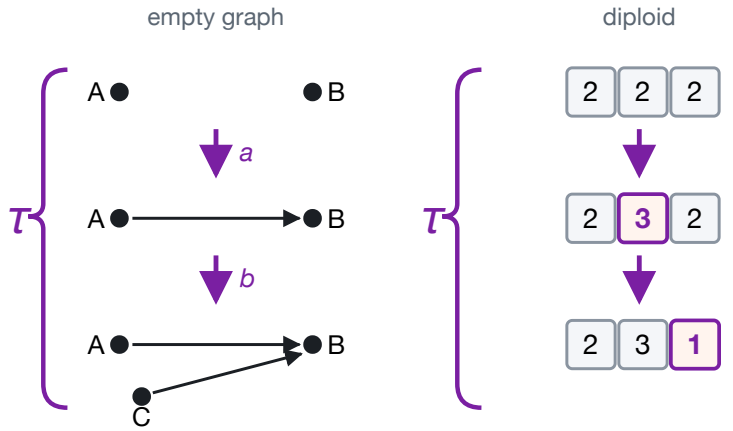
a trajectory τ

Policy gradient (REINFORCE): $\nabla_\theta \mathbb{E}_{\tau\sim\Pr(\tau|\theta)}[r(\tau)] = \mathbb{E}_\tau\left[r(\tau)\,\nabla_\theta \log \Pr(\tau \mid \theta)\right]$ — *gradient through* $\log \Pr(\tau \mid \theta)$ *only, not the fixed reward* $r(\tau)$.

GReinSS: Generative Reinforcement Learning of Structured States

Policy gradient (REINFORCE): $\nabla_\theta \, \mathbb{E}_{\tau \sim \Pr(\tau|\theta)}[r(\tau)] = \mathbb{E}_\tau \left[r(\tau) \, \nabla_\theta \log \Pr(\tau \mid \theta) \right]$

Concept	In RL	In GReinSS
State s_t	current situation	*same as RL*
Action a_t	transitions $s_t \to s_{t+1}$	*same as RL*
Policy π_θ	picks the next action	*same as RL*
Trajectory τ	path $s_0 \to \cdots \to s_{\lvert\tau\rvert}$	*same, terminal* $S(\tau)$
Reward $r(\tau)$	scores terminal state	**use** $\Pr(X \mid S)$**???**
Objective	$\max_\theta \mathbb{E}_\tau[r(\tau)]$	$\max_\theta \log \Pr(X_{1:N} \mid \theta)$



actions extend τ; we infer its terminal state $S(\tau)$

Key question: How to set rewards $r(\tau)$ such that

$$\nabla_\theta \, \mathbb{E}_{\tau \sim \Pr(\tau|\theta)}[r(\tau)] = \mathbb{E}_\tau \left[r(\tau) \, \nabla_\theta \log \Pr(\tau \mid \theta) \right] = \nabla_\theta \log \Pr(X_{1:N} \mid \theta)?$$

Dynamic rewards

Train with a policy gradient using the **dynamically rescaled reward**
$r(\tau) = \sum_{i=1}^{N} \Pr(X_i \mid \tau) / \Pr(X_i \mid \theta)$

$$\nabla_\theta \underbrace{\log \Pr(X_{1:N} \mid \theta)}_{\text{log-likelihood}}$$

what we optimize

$$=$$

$$\underbrace{\mathbb{E}_\tau\Big[r(\tau)\, \nabla_\theta \log \Pr(\tau \mid \theta)\Big]}_{\text{policy gradient}}$$

how we optimize it

Theorem 1 (Unbiased policy gradient). *With the dynamically changing reward* $r(\tau) = \sum_{i=1}^{N} \Pr(X_i \mid \tau) / \Pr(X_i \mid \theta)$, *the policy gradient* $\mathbb{E}_\tau\Big[r(\tau)\, \nabla_\theta \log \Pr(\tau \mid \theta)\Big]$ *is an unbiased estimator of* $\nabla_\theta \log \Pr(X_{1:N} \mid \theta)$.

Intuition behind dynamic rewards — why the denominator $\Pr(X_i \mid \theta)$?

$\Pr(X \mid S)$	S_1	S_2	S_3
X_1	.5		
X_2		.3	.2

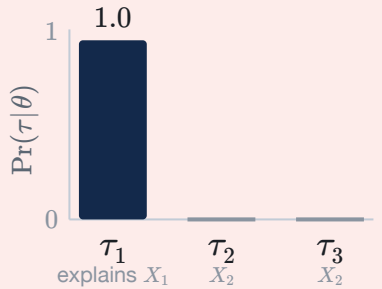
Example: Two measurements X_1 and X_2. States $\mathcal{S} = \{S_1, S_2, S_3\}$.

Parameters: $\theta \equiv (\Pr(S_1 \mid \theta), \Pr(S_2 \mid \theta), \Pr(S_3 \mid \theta))$

Objective: $\max_\theta \Pr(X_1, X_2 \mid \theta) = \max_\theta \mathbb{E}_\tau[r(\tau)]$

Fixed rewards

$$r(\tau) = \Pr(X_i \mid \tau)$$



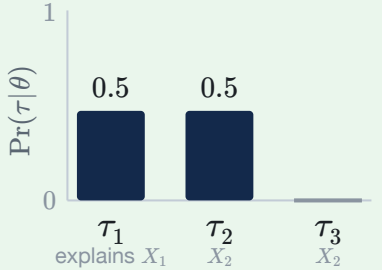
$\theta^\star = (1, 0, 0)$: $\Pr(X_1 \mid \theta) = .5,\ \Pr(X_2 \mid \theta) = 0$

$\mathbb{E}_\tau[r]$ is linear in the policy ⇒ all mass on the best, τ_1

$\Pr(X_1, X_2 \mid \theta) = .5 \times 0 = \mathbf{0}$ ✗

Dynamic rewards

$$r(\tau) = \Pr(X_i \mid \tau) / \Pr(X_i \mid \theta)$$



$\theta^\star = (.5, .5, 0)$: $\Pr(X_1 \mid \theta) = .25,\ \Pr(X_2 \mid \theta) = .15$

$\max \mathbb{E}_\tau[r] = \max \Pr(X_1, X_2 \mid \theta)$ ⇒ balances τ_1, τ_2

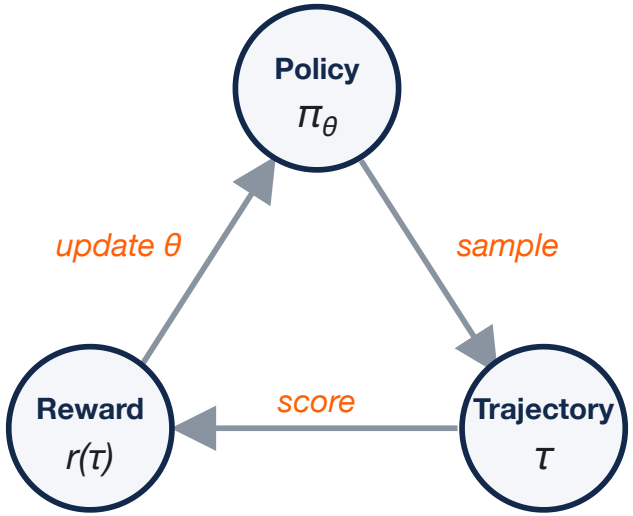
$\Pr(X_1, X_2 \mid \theta) = .25 \times .15 = 0.0375$ ✓

GReinSS training loop

The **sample → score → update** cycle of RL, run with a reward that changes as θ learns:

$$r(\tau) = \sum_{i=1}^{N} \frac{\Pr(X_i \mid \tau)}{\boxed{\Pr(X_i \mid \theta)}} \qquad \Pr(X_i \mid \theta) = \mathbb{E}_\tau\big[\Pr(X_i \mid \tau)\big]$$

Dynamic reward — the boxed denominator is **re-estimated by sampling** each iteration.



Repeat until convergence:

1. **Sample** a batch $\tau_1, \ldots, \tau_M \sim \Pr(\tau \mid \theta)$
2. **Score** each: $\Pr(X_i \mid \tau_j)$
3. **Estimate** $\Pr(X_i \mid \theta) \approx \frac{1}{M} \sum_j \Pr(X_i \mid \tau_j)$ *(same batch)*
4. **Reward** $r(\tau_j) = \sum_i \Pr(X_i \mid \tau_j) / \Pr(X_i \mid \theta)$
5. **Policy-gradient** step along $\mathbb{E}_\tau\big[r(\tau)\, \nabla_\theta \log \Pr(\tau \mid \theta)\big]$

You supply only two things:

- a **generator** for S (action-by-action)
- the likelihood $\Pr(X \mid S)$

→ NOTEBOOK · Demo 1: Set reconstruction

Recover binary **sets** from noisy real-valued measurements. *Trains live in ~10 s.*

Problem. $S_i^* \subseteq \mathcal{U}$, observe $X_{i,j} \sim \mathcal{N}(1, \sigma^2)$ if $j \in S_i^*$, else $\mathcal{N}(0, \sigma^2)$.

You'll write:

```
def log_pr_x_given_g(state, obs):
    return -0.5*np.sum((obs-state)**2)/sigma**2
```

…then `train_model_off_policy(...)`.

Shared structure: Each S_i^* = a union of a few reusable **subsets**, so a shared $\Pr(S \mid \theta)$ pools across observations.

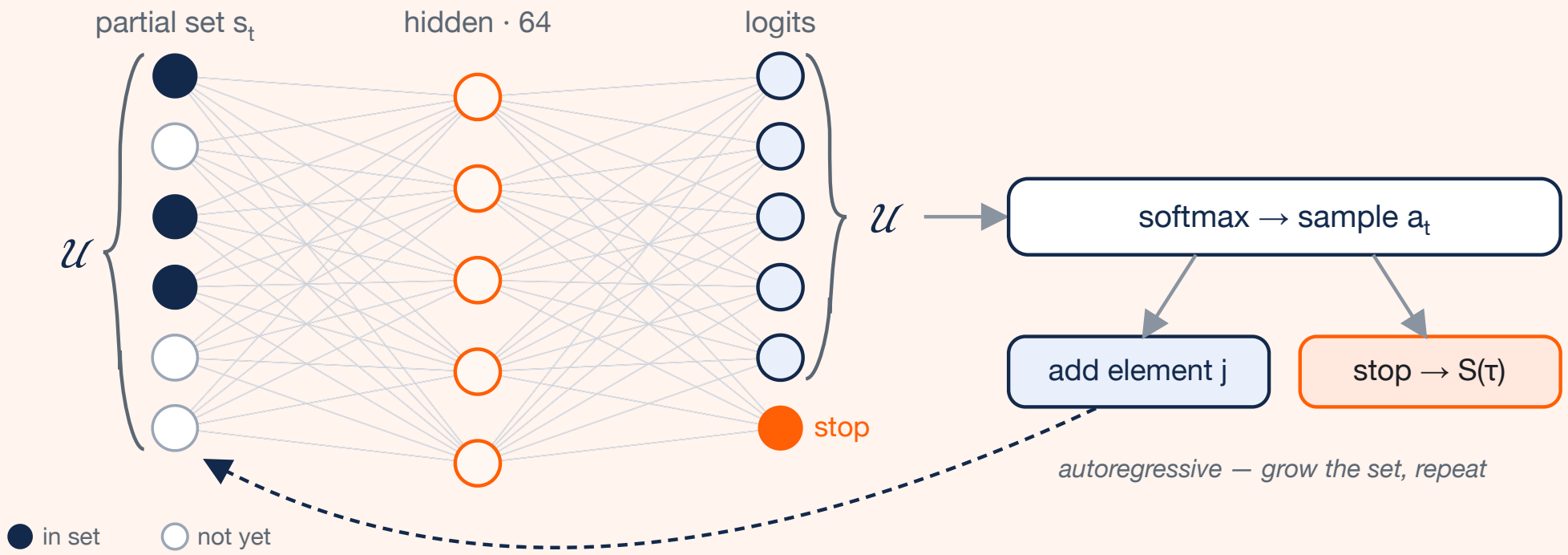
Watch for:

- median $\log \Pr(X_i \mid \theta)$ climbing to ~0
- recovered sets vs. thresholding the noise
- where the **shared model fixes** noisy bits

● **true set** S_i^* ● **noisy observation** X_i — rounding at ½ flips the ◯ circled elements.

Neural network policy π_θ for the set reconstruction problem

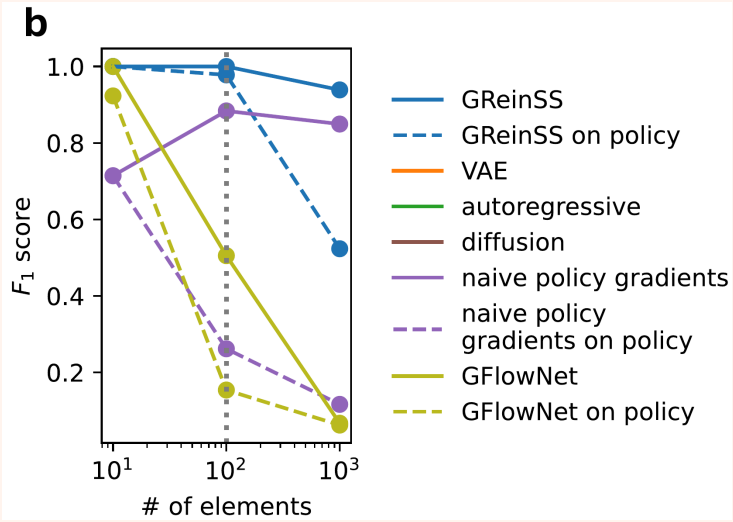
1-hidden-layer MLP — `Linear(` $|\mathcal{U}| \to 64$ `)` → LeakyReLU → `Linear(` $64 \to |\mathcal{U}|+1$ `)` — run once **per step**: read the partial set, then **add an element** or **stop**.



$|\mathcal{U}|$ add-logits **+ one stop**; weights **shared across steps**.
Illegal (already-added) moves masked. $\Pr(S \mid \theta) = \sum_{\tau:\, S(\tau)=S} \prod_t \pi_\theta(a_t \mid s_t)$

→ NOTEBOOK · Demo 2: Off-policy learning at scale

With a **larger, dispersed** state space: sampling $\Pr(\tau \mid \theta)$ rarely hits a terminal state $S(\tau)$ explaining any X_i, so most trajectories earn no reward and learning stalls.



Off-policy sampling: If the policy rarely samples states explaining any X_i, learning stalls — **sample where the data says to look:**

Theorem 2 (Optimal off-policy proposal). *The unbiased, variance-minimizing sampling proposal is $q^\star$:*

$$q^\star(\tau \mid X_{1:N}, \theta) = \frac{1}{N} \sum_{i=1}^{N} \Pr(\tau \mid X_i, \theta)$$

$$\Pr(\tau \mid X_i, \theta) = \frac{\Pr(X_i \mid \tau)\, \Pr(\tau \mid \theta)}{\Pr(X_i \mid \theta)}$$

$q^\star$ **is intractable** → sample a tractable $q \approx q^\star$ (for sets: favor element j by $(X_{i,j} - \frac{1}{2})/\sigma^2$), then **reweight** by $\Pr(\tau \mid \theta)/q(\tau)$.

The off-policy update in full

Exact — unbiased for the data log-likelihood gradient, for *any* proposal q:

$$\nabla_\theta \log \Pr(X_{1:N} \mid \theta) = \mathbb{E}_{\tau \sim q}\left[\, w(\tau)\, r(\tau)\, \nabla_\theta \log \Pr(\tau \mid \theta) \,\right]$$

$$w(\tau) = \frac{\Pr(\tau \mid \theta)}{q(\tau)} = \prod_t \frac{\pi_\theta(a_t \mid s_t)}{q(a_t \mid s_t)} \qquad r(\tau) = \sum_{i=1}^{N} \frac{\Pr(X_i \mid \tau)}{\Pr(X_i \mid \theta)}$$

Set problem — pick an observation i uniformly, then bias each step by its log-odds:

$$q(a_t = \text{add } j \mid s_t) = \text{softmax}_j\left(\log \pi_\theta(\text{add } j \mid s_t) + \tfrac{X_{i,j} - \frac{1}{2}}{\sigma^2} \right)$$

Batch estimate over $\tau_1, \ldots, \tau_M \sim q$ (the denominator $\Pr(X_i \mid \theta)$ in r is re-estimated each batch)**:**

$$\widehat{g} = \frac{1}{M} \sum_{m=1}^{M} w(\tau_m)\, r(\tau_m)\, \nabla_\theta \log \Pr(\tau_m \mid \theta) \qquad \Pr(X_i \mid \theta) \approx \frac{1}{M} \sum_{m=1}^{M} w(\tau_m)\, \Pr(X_i \mid \tau_m)$$

On-policy: $q = \Pr(\tau \mid \theta) \Rightarrow w \equiv 1$. **Self-normalize** (divide by $\sum_m w(\tau_m)$) to cut variance.

→ NOTEBOOK · Demo 3: Graph inference (pre-trained)

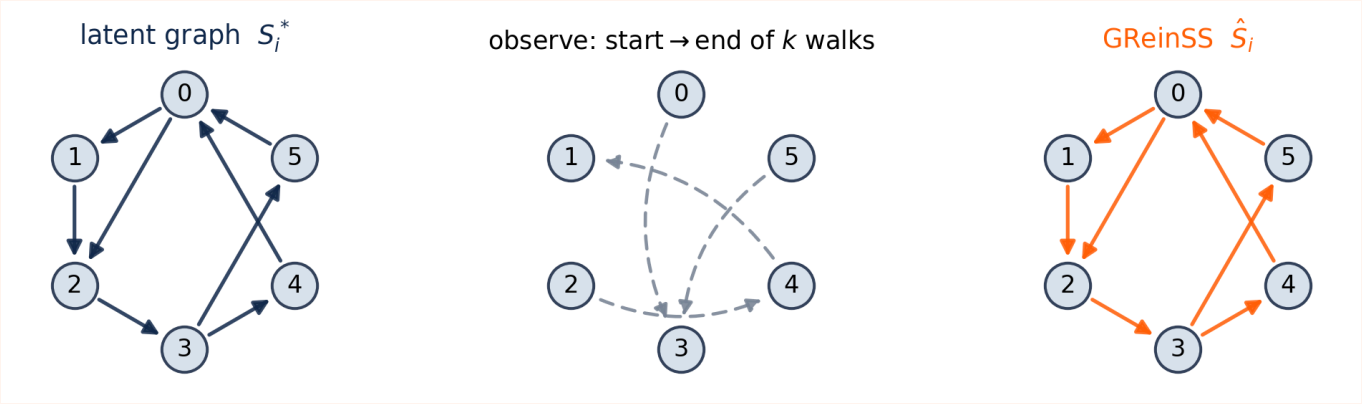
Reconstruct latent **directed graphs** from start/end points of k random walks. *Pre-trained.*

Problem. S_i^* = directed graph (10 nodes, 90 edges); we see only the (v, w) **endpoints** of k walks. $\Pr(X \mid S)$: shifted-Laplacian.

Shared structure: Each S_i^* = a random **thresholded subgraph** of one **Erdős–Rényi** $(p=\frac{1}{2})$ base graph.

Watch for:

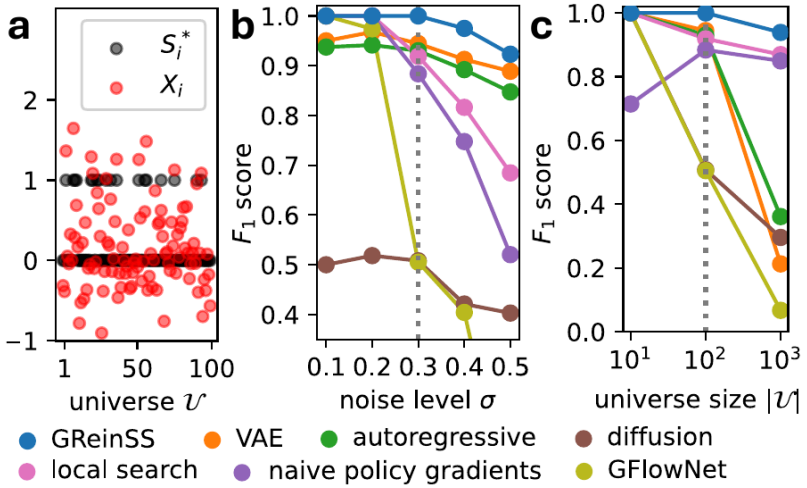
- training likelihood curve (pre-computed)
- a reconstructed graph $\hat{S}_i$ vs. true S_i^*
- per-graph F_1 (median ≈ 0.97)



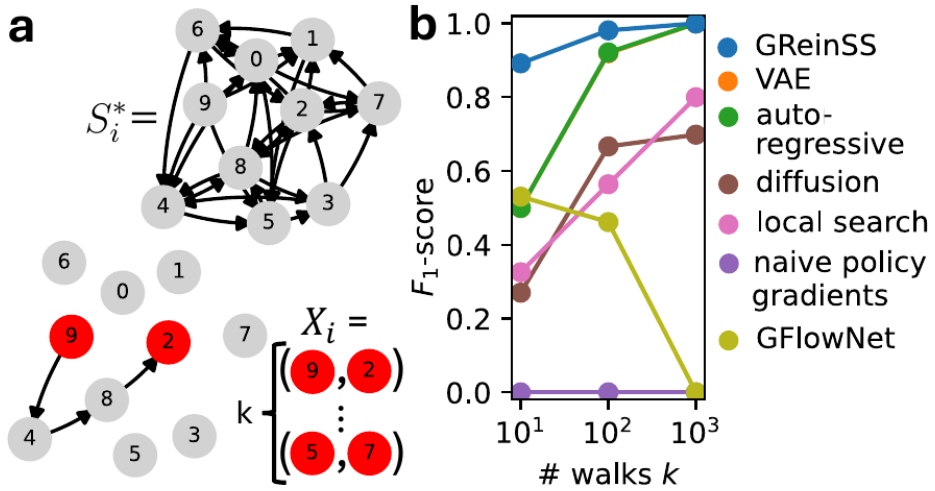
• **latent graph** S_i^* • **GReinSS** $\hat{S}_i$ — dashed = observed (start→end); paths never seen.

Simulation results from [Ivanovic et al. ICML 2026]

Latent sets — noisy measurements



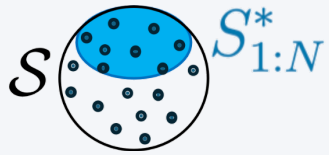
Latent graphs — walk endpoints



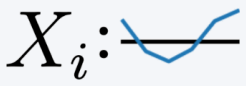
Method	Family	Max. $\prod_i \Pr(X_i \mid \theta)$?
GReinSS	RL	✅
Naive policy gradient	RL	❌
GFlowNets	RL	❌
VAE / autoregressive / diffusion **+ EM**	Variational + EM	≈
Local search	Search	❌

Conclusion

States $S_{1:N} \sim \Pr^*(\mathcal{S})$



Measurements $X_{1:N}$ generated from $S_{1:N}$



$\mathcal{S}$ is typically **large and combinatorial** — graphs, strings, sets, …

Good fit ✓

- latent state is **discrete / combinatorial**
- you can **generate** it incrementally
- you know (or can compute) $\Pr(X \mid S)$
- observations are **indirect** & shared structure exists

Reach elsewhere ✗

- continuous latents → VAE / diffusion
- tractable exact E-step → classic EM
- rewards truly fixed & known → standard RL / GFlowNet

Recipe: ① write a generator for S · ② write $\Pr(X \mid S)$ · ③ `train_model_off_policy` · ④ `simpleInference`. *(optional)* add an off-policy proposal for hard instances.

Thank you — let's build

Code + notebook: `code/README.md`, `tutorial/GReinSS_demo.ipynb`
Recipe: generator for S + likelihood $\Pr(X \mid S)$ → train → infer

Questions? · melkebir@illinois.edu