Generative Flow Networks

Connections with Reinforcement Learning

Tristan Deleu

Entalpic

Probabilistic inference

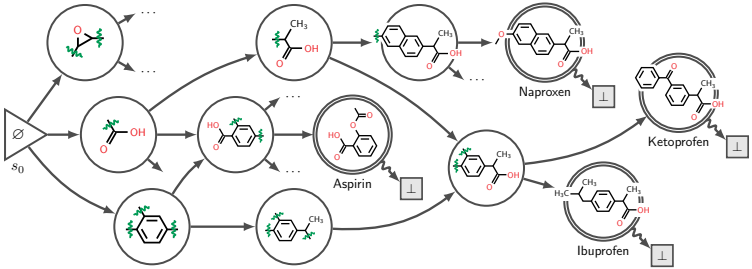
- **Objective:** Sampling from the target distribution

$$P^\star(x) = \frac{\exp(-\mathcal{E}(x))}{Z}$$

- States are organized as a **pointed DAG** $\mathcal{G}$
- **Forward transition probability:** Distribution over the children of s

$$P_F(\cdot \mid s)$$

- **Sampling** from $P_F^\top$: follow P_F at random until we reach $\perp$.



Terminating state distribution

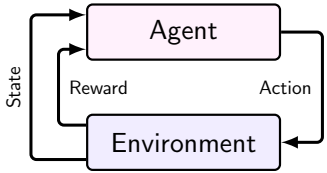
$$P_F^\top(x) \triangleq \sum_{\tau : x \to \perp \in \tau} P_F(\tau) = \sum_{\tau : s_0 \rightsquigarrow x} \left[\prod_{t=0}^{T_\tau} P_F(s_{t+1} \mid s_t) \right]$$

Outline

- Introduction to Reinforcement Learning
- Connection between GFlowNets and Maximum Entropy RL
- KL-regularized RL
- Intermediate rewards

Reinforcement learning

- **Idea:** An **agent** learns to achieve a goal by interacting with an **environment**.
- The agent takes actions in the environment, and receives rewards. The goal is to **maximize the sum of reward**.



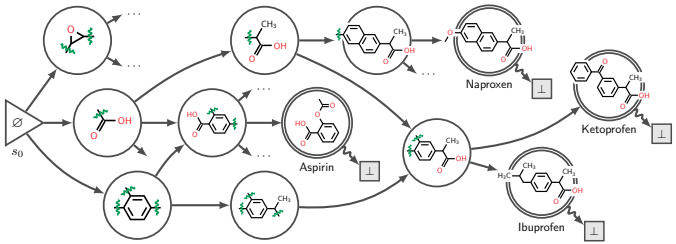
Markov Decision Process

- The environment is represented as a **Markov Decision Process** (MDP)
  - A **state** space $\mathcal{S}$
  - An **action** space $\mathcal{A}$
  - A **transition** function $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$
  - A reward function

$$r(s_t, s_{t+1})$$

- The agent takes actions following a **policy**

$$\pi(s_{t+1} \mid s_t)$$



Objective

$$\pi^{\star}_{\mathrm{RL}} = \arg\max_{\pi} \mathbb{E}_{\tau} \left[\sum_{t=0}^{T} r(s_t, s_{t+1}) \right]$$

Value-based methods

- How to find the **optimal policy** $\pi^\star_{\mathrm{RL}}$?

Optimal policy

$$\pi^\star_{\mathrm{RL}}(s' \mid s) = \begin{cases} 1 & \text{if } s' = \underset{s'' \in \mathrm{Ch}_{\mathcal{G}}(s)}{\arg\max}\ Q^\star(s, s'') \\ 0 & \text{otherwise} \end{cases}$$

where $Q^\star(s, s')$ satisfies the **Bellman optimality equations**

Bellman optimality equations

$$Q^\star(s, s') = r(s, s') + V^\star(s')$$
$$V^\star(s) = \max_{s'' \in \mathrm{Ch}_{\mathcal{G}}(s)} Q^\star(s, s'')$$

- **Idea:** Find the optimal $Q^\star(s, s')$, and derive the optimal policy from it.

Inference as control

- Treat probabilistic inference as a **control problem** in a Markov Decision Process
  - $\mathcal{G}$ is the structure of the MDP
  - The reward function defined such that

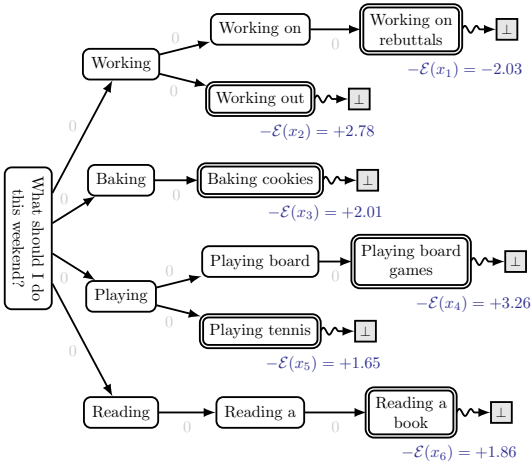
$$\sum_{t=0}^{T} r(s_t, s_{t+1}) = -\mathcal{E}(s_T)$$

e.g., sparse reward received upon termination.

- Find the **optimal policy**

$$\pi^{\star}_{\mathrm{RL}} = \arg\max_{\pi} \mathbb{E}_{\tau}\left[\sum_{t=0}^{T} r(s_t, s_{t+1})\right]$$

- This finds a **mode** of the target distribution $P^{\star}(x) \propto \exp(-\mathcal{E}(x))$



Maximum-entropy reinforcement learning

- **Idea:** Add a **regularizer** to encourage diversity

Maximum-entropy reinforcement learning

$$\pi^\star_{\text{MaxEnt}} = \arg\max_\pi \mathbb{E}_\tau \left[\sum_{t=0}^{T} r(s_t, s_{t+1}) + \mathcal{H}\big(\pi(\cdot \mid s_t)\big) \right]$$

where

$$\mathcal{H}\big(\pi(\cdot \mid s)\big) = -\sum_{s' \in \text{Ch}_\mathcal{G}(s)} \pi(s' \mid s) \log \pi(s' \mid s)$$

is the **entropy** of the policy π.

- Higher entropy $\Rightarrow$ more *uniform* policy.

	RL	MaxEnt RL
Bellman equations	$Q^\star(s,s') = r(s,s') + V^\star(s')$ $V^\star(s) = \max_{s'' \in \text{Ch}_\mathcal{G}(s)} Q^\star(s,s'')$	$Q^\star_{\text{soft}}(s,s') = r(s,s') + V^\star_{\text{soft}}(s')$ $V^\star_{\text{soft}}(s) = \log \sum_{s'' \in \text{Ch}_\mathcal{G}(s)} \exp\big(Q^\star_{\text{soft}}(s,s'')\big)$
Optimal policy	$\pi^\star_{\text{RL}}(s' \mid s) = \arg\max_{s' \in \text{Ch}_\mathcal{G}(s)} Q^\star(s,s')$	$\pi^\star_{\text{soft}}(s' \mid s) = \text{softmax}\big(Q^\star_{\text{soft}}(s,s')\big)$

Maximum-entropy reinforcement learning

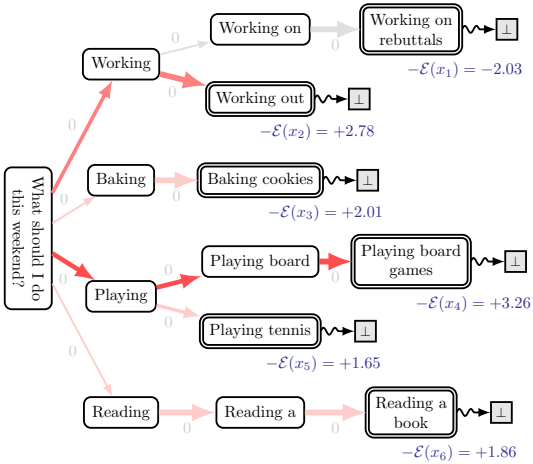
- **Objective:**

$$\pi^{\star}_{\text{MaxEnt}} = \arg\max_{\pi} \mathbb{E}_{\tau}\left[\sum_{t=0}^{T} r(s_t, s_{t+1}) + \mathcal{H}\big(\pi(\cdot \mid s_t)\big)\right]$$

Theorem

$$\pi^{\star}_{\text{MaxEnt}}(\tau) \propto \exp\left(\sum_{t=0}^{T} r(s_t, s_{t+1})\right)$$

- **Recall:** The reward function is defined such that $\sum_{t=0}^{T} r(s_t, s_{t+1}) = -\mathcal{E}(s_T)$.
- If we know the optimal policy $\pi^{\star}_{\text{MaxEnt}}$, then we can sample from $P^{\star}(x) \propto \exp(-\mathcal{E}(x))$ by following $\pi^{\star}_{\text{MaxEnt}}$.



Proof

$$
\begin{aligned}
\pi^\star_{\text{MaxEnt}}(\tau) &= \prod_{t=0}^{T} \pi^\star_{\text{MaxEnt}}(s_{t+1} \mid s_t) \\
&= \prod_{t=0}^{T} \exp\left(r(s_t, s_{t+1}) + V^\star_{\text{soft}}(s_{t+1}) - V^\star_{\text{soft}}(s_t)\right) \\
&= \exp\left(\sum_{t=0}^{T} \left(r(s_t, s_{t+1}) + V^\star_{\text{soft}}(s_{t+1}) - V^\star_{\text{soft}}(s_t)\right)\right) \\
&= \exp\left(\sum_{t=0}^{T} r(s_t, s_{t+1}) + V^\star_{\text{soft}}(\bot) - V^\star_{\text{soft}}(s_0)\right) \\
&= \frac{1}{\exp\left(V^\star_{\text{soft}}(s_0)\right)} \exp\left(\sum_{t=0}^{T} r(s_t, s_{t+1})\right) \\
&\propto \exp\left(\sum_{t=0}^{T} r(s_t, s_{t+1})\right)
\end{aligned}
$$

Theorem

$$\pi^\star_{\text{MaxEnt}}(\tau) \propto \exp\left(\sum_{t=0}^{T} r(s_t, s_{t+1})\right)$$

Soft Bellman optimality equations

$$Q^\star_{\text{soft}}(s, s') = r(s, s') + V^\star_{\text{soft}}(s')$$

$$V^\star_{\text{soft}}(s) = \log \sum_{s'' \in \text{Ch}_\mathcal{G}(s)} \exp\left(Q^\star_{\text{soft}}(s, s'')\right)$$

Soft optimal policy

$$\pi^\star_{\text{MaxEnt}}(s' \mid s) = \exp\left(Q^\star_{\text{soft}}(s, s') - V^\star_{\text{soft}}(s)\right)$$

Biased sampling

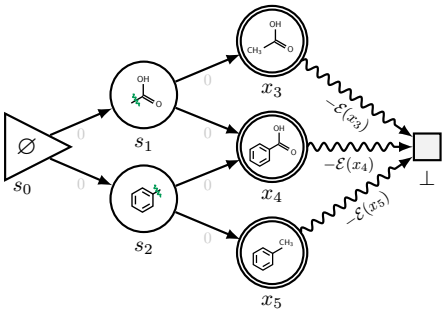
- If $\sum_t r(s_t, s_{t+1}) = -\mathcal{E}(s_T)$, then this is *almost* matching the target distribution $P^\star(x) \propto \exp(-\mathcal{E}(x))$.

$$\pi^\star_{\text{MaxEnt}}(\tau) \propto \exp\left(\sum_{t=0}^{T} r(s_t, s_{t+1})\right)$$

- We are interested in a distribution over **outcomes**, not a distribution over **how** these objects are constructed.
- **Terminating state distribution:**

$$\pi^{\star\top}_{\text{MaxEnt}}(x) = \sum_{\tau : x \to \perp \in \tau} \pi^\star_{\text{MaxEnt}}(\tau) \propto n(x) \exp(-\mathcal{E}(x))$$

- Sampling is **biased** towards terminating states with multiple paths leading to them.



x	τ	$\pi^\star_{\text{MaxEnt}}(\tau)$	$\pi^{\star\top}_{\text{MaxEnt}}(x)$
x_3	$s_0 \to s_1 \to x_3$	$\exp(-\mathcal{E}(x_3))/Z'$	$\exp(-\mathcal{E}(x_3))/Z'$
x_4	$s_0 \to s_1 \to x_4$	$\exp(-\mathcal{E}(x_4))/Z'$	$2\exp(-\mathcal{E}(x_4))/Z'$
	$s_0 \to s_2 \to x_4$	$\exp(-\mathcal{E}(x_4))/Z'$	
x_5	$s_0 \to s_2 \to x_5$	$\exp(-\mathcal{E}(x_5))/Z'$	$\exp(-\mathcal{E}(x_5))/Z'$

Reward correction

Theorem

Let $P_B(s \mid s')$ be an arbitrary **backward policy** (*i.e.*, a distribution over the parents of $s' \neq s_0$ in $\mathcal{G}$). If the reward function satisfies for any trajectory $\tau = (s_0, s_1, \ldots, s_T, \perp)$:

$$\sum_{t=0}^{T} r(s_t, s_{t+1}) = -\mathcal{E}(s_T) + \sum_{t=1}^{T} \log P_B(s_{t-1} \mid s_t),$$

then the terminating state distribution associated with the optimal MaxEnt RL policy satisfies

$$\pi^{\star\top}_{\text{MaxEnt}}(x) \propto \exp(-\mathcal{E}(x))$$

Proof

$$\pi^{\star\top}_{\mathrm{MaxEnt}}(x) = \sum_{\tau : s_0 \rightsquigarrow x} \prod_{t=0}^{T_\tau} \pi^\star_{\mathrm{MaxEnt}}(s_{t+1} \mid s_t)$$

$$= \sum_{\tau : s_0 \rightsquigarrow x} \exp\left[\sum_{t=0}^{T_\tau} \left(r(s_t, s_{t+1}) + V^\star_{\mathrm{soft}}(s_{t+1}) - V^\star_{\mathrm{soft}}(s_t)\right)\right]$$

$$= \sum_{\tau : s_0 \rightsquigarrow x} \exp\left[\sum_{t=0}^{T_\tau} r(s_t, s_{t+1}) + V^\star_{\mathrm{soft}}(\bot) - V^\star_{\mathrm{soft}}(s_0)\right]$$

$$= \sum_{\tau : s_0 \rightsquigarrow x} \exp\left[-\mathcal{E}(x) + \sum_{t=1}^{T_\tau} \log P_B(s_{t-1} \mid s_t) - V^\star_{\mathrm{soft}}(s_0)\right]$$

$$= \exp\left(-\mathcal{E}(x) - V^\star_{\mathrm{soft}}(s_0)\right) \underbrace{\sum_{\tau : s_0 \rightsquigarrow x} \prod_{t=1}^{T_\tau} P_B(s_{t-1} \mid s_t)}_{=1}$$

$$= \frac{1}{\exp(V^\star_{\mathrm{soft}}(s_0))} \exp(-\mathcal{E}(x))$$

Corrected reward

$$\sum_{t=0}^{T} r(s_t, s_{t+1}) = -\mathcal{E}(s_T) + \sum_{t=1}^{T} \log P_B(s_{t-1} \mid s_t)$$

Soft Bellman optimality equations

$$Q^\star_{\mathrm{soft}}(s, s') = r(s, s') + V^\star_{\mathrm{soft}}(s')$$

$$V^\star_{\mathrm{soft}}(s) = \log \sum_{s'' \in \mathrm{Ch}_{\mathcal{G}}(s)} \exp\left(Q^\star_{\mathrm{soft}}(s, s'')\right)$$

Soft optimal policy

$$\pi^\star_{\mathrm{MaxEnt}}(s' \mid s) = \exp\left(Q^\star_{\mathrm{soft}}(s, s') - V^\star_{\mathrm{soft}}(s)\right)$$

Examples

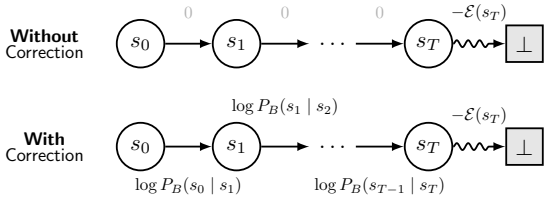
- Corrected reward:

$$\sum_{t=0}^{T} r(s_t, s_{t+1}) = \mathcal{E}(s_T) + \sum_{t=1}^{T} \log P_B(s_{t-1} \mid s_t)$$

- In the case of a sparse reward, we can **add the correction along the way**.
- If the MDP is a **tree**, then $P_B(s \mid s') = 1$: we recover the case with uncorrected rewards.
- If the backward policy is

$$P_B(s_t \mid s_{t+1}) = n(s_t)/n(s_{t+1})$$

then we get a policy **maximizing the entropy** over trajectories.



$$r(s_t, s_{t+1}) = \log P_B(s_t \mid s_{t+1})$$
$$r(s_T, \perp) = -\mathcal{E}(s_T)$$

Path Consistency Learning

- **Recall:** Soft Bellman optimality equations & optimal policy

$$Q^\star_{\text{soft}}(s, s') = r(s, s') + V^\star_{\text{soft}}(s') \qquad \pi^\star_{\text{MaxEnt}}(s' \mid s) = \exp\left(Q^\star_{\text{soft}}(s, s') - V^\star_{\text{soft}}(s)\right)$$

- Unrolling $\log \pi^\star_{\text{MaxEnt}}(s_{t+1} \mid s_t)$ along a trajectory, we get

Path Consistency Learning

$$\sum_{t=0}^{T} \log \pi^\star_{\text{MaxEnt}}(s_{t+1} \mid s_t) = \sum_{t=0}^{T} r(s_t, s_{t+1}) - V^\star_{\text{soft}}(s_0)$$

Trajectory Balance

$$\log \frac{Z \prod_{t=0}^{T} P_F(s_{t+1} \mid s_t)}{\exp(-\mathcal{E}(s_T)) \prod_{t=1}^{T} P_B(s_{t-1} \mid s_t)}$$

- PCL is equivalent to TB, with the reward correction $\sum_{t=0}^{T} r(s_t, s_{t+1}) = -\mathcal{E}(s_T) + \sum_{t=1}^{T} \log P_B(s_{t-1} \mid s_t)$
- There exist other equivalences with MaxEnt RL algorithms (*e.g.*, FL-DB & Soft Q-Learning).

KL-regularized Reinforcement Learning

- We can use **other types of regularizers**. For example if we want the policy to not deviate too much from a **base policy** $\mu(\cdot \mid s)$.

KL-regularized Reinforcement Learning

$$\pi^\star_{\mathrm{RelEnt}} = \arg\max_\pi \mathbb{E}_\tau \left[\sum_{t=0}^{T} r(s_t, s_{t+1}) - \mathrm{KL}\big(\pi(\cdot \mid s_t) \,\|\, \mu(\cdot \mid s_t)\big) \right]$$

where

$$\mathrm{KL}\big(\pi(\cdot \mid s) \,\|\, \mu(\cdot \mid s)\big) = \sum_{s' \in \mathrm{Ch}_{\mathcal{G}}(s)} \pi(s' \mid s) \log \frac{\pi(s' \mid s)}{\mu(s' \mid s)}$$

- The policy $\mu(\cdot \mid s)$ may be a **prior distribution** (*e.g.*, a distribution learned from data)
- This type of objective is very popular recently with **RL-finetuning of LLMs**.

Terminating state distribution

$$\pi^{\star}_{\mathrm{RelEnt}} = \arg\max_{\pi} \mathbb{E}_{\tau}\left[\sum_{t=0}^{T} r(s_t, s_{t+1}) - \mathrm{KL}\big(\pi(\cdot \mid s_t) \,\|\, \mu(\cdot \mid s_t)\big)\right]$$

Theorem

If the reward function satisfies for any trajectory $\tau = (s_0, s_1, \ldots, s_T, \perp)$:

$$\sum_{t=0}^{T} r(s_t, s_{t+1}) = -\mathcal{E}(s_T),$$

then the terminating state distribution associated with the optimal KL-regularized RL policy satisfies

$$\pi^{\star\top}_{\mathrm{RelEnt}}(x) \propto \mu^{\top}(x) \exp(-\mathcal{E}(x))$$

Relative Trajectory Balance

- Modification of the **Trajectory Balance** to account for a prior distribution

Relative Trajectory Balance

$$Z \prod_{t=0}^{T} P_F(s_{t+1} \mid s_t) = \exp(-\mathcal{E}(s_T)) \prod_{t=0}^{T} \pi_{\text{prior}}(s_{t+1} \mid s_t)$$

- If RTB is satisfied for all trajectories, then $P_F^{\top}(x) \propto \pi_{\text{prior}}^{\top}(x) \exp(-\mathcal{E}(x))$.
- This is **equivalent** to a KL-regularized RL algorithm called **Trust-PCL** (without any reward correction needed).

Energy difference as intermediate reward

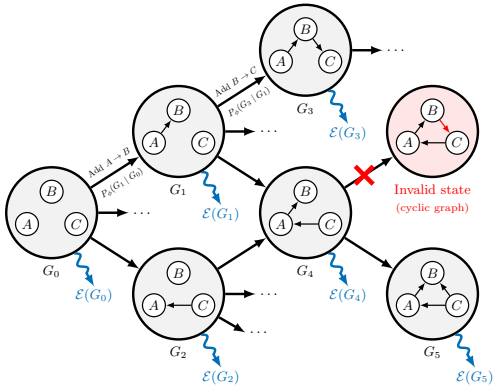
- In cases where **all the states are terminating**, then we can use the difference in energies as an intermediate reward

$$r(s_t, s_{t+1}) = \mathcal{E}(s_t) - \mathcal{E}(s_{t+1})$$

this guarantees that

$$\sum_{t=0}^{T} r(s_t, s_{t+1}) = \underbrace{\mathcal{E}(s_0)}_{\text{constant}} - \mathcal{E}(s_T)$$

- **Examples:** Hypergrid, structure learning.
- **Consequence:** FL-DB does not depend on a flow (value) function anymore. So-called **"modified" DB**.



Diffusion model

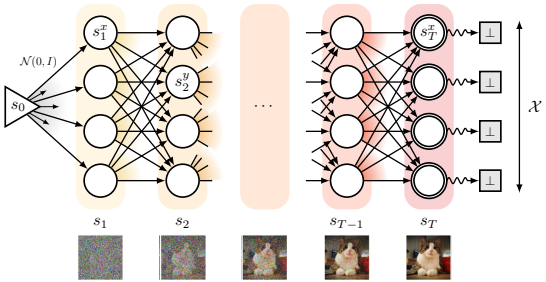
- Noising process $x_T \to x_{T-1} \to \ldots \to x_1$

$$q(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \sqrt{\alpha_t} x_t, (1-\alpha_t) I)$$

- **Objective:** Find a model $p_\theta(x_{t+1} \mid x_t)$ that denoises x_t to x_{t+1}.

$$p_\theta(x_{t+1} \mid x_t) = \mathcal{N}(x_{t+1}; \mu_\theta(x_t, t), \sigma_t^2 I)$$

where we start with $p(x_1) = \mathcal{N}(x_1; 0, I)$.



	Diffusion model	GFlowNet		First step
Denoising	$p_\theta(x_{t+1} \mid x_t)$	$P_F(x_{t+1} \mid x_t)$	$= \mathcal{N}(\mu_\theta(x_t, t), \sigma_t^2 I)$	$\mathcal{N}(0, I)$
Noising	$q(x_{t-1} \mid x_t)$	$P_B(x_{t-1} \mid x_t)$	$= \mathcal{N}(\sqrt{\alpha_t} x_t, (1-\alpha_t) I)$	δ_{s_0}

Steering diffusion models

- **Diffusion sampler:** Steering the generation of images with a diffusion process based on an energy $\mathcal{E}(x)$ *i.e.*, sampling from the distribution $\propto \exp(-\mathcal{E}(x))$.
- Parametrization of the denoiser / forward policy $P_F(x_{t+1} \mid x_t) = \mathcal{N}(\mu_\theta(x_t, t), \sigma_t^2 I)$

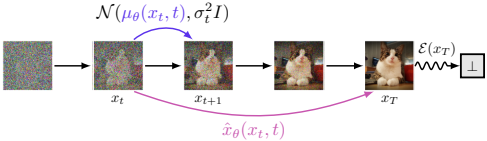
$$\mu_\theta(x_t, t) = a_t x_t + b_t \hat{x}_\theta(x_t, t)$$

- Use the x_T-prediction model as a **proxy**, and evaluate the difference in energies on the predictions

$$r(s_t, s_{t+1}) = \mathcal{E}\big(\hat{x}_\theta(s_t)\big) - \mathcal{E}\big(\hat{x}_\theta(s_{t+1})\big)$$

With careful treatment of edge cases, this guarantees

$$\sum_{t=0}^{T} r(s_t, s_{t+1}) = -\mathcal{E}(x_T)$$



- **Idea #1:** Use Trajectory Balance
- **Idea #2:** Use the difference in energies $r(s_t, s_{t+1}) = \mathcal{E}(x_t) - \mathcal{E}(x_{t+1})$

Conclusion

- There are **strong connections** between GFlowNets and Reinforcement Learning
- All the techniques from RL can be transferred to GFlowNet applications:
  - Replay buffers
  - Target networks
  - Distributional
  - Advanced exploration strategies
- **Takeaways:**
  - **Be careful** of papers claiming the GFlowNets are significantly better than RL
  - Using GFlowNets for autoregressive tasks (*e.g.*, LLMs) is **exactly equivalent** to MaxEnt RL