Maximum entropy GFlowNets with soft Q-learning

Sobhan Mohammadpour[12] **Emmanuel Bengio**[3] **Emma Frejinger**[4] **Pierre-Luc Bacon**[24]

[1] MIT [2] Mila - Quebec AI Institute [3] Valence Labs
[4] Department of Computer Science and Operations Research, University of Montreal

Abstract

Generative Flow Networks (GFNs) have emerged as a powerful tool for sampling discrete objects from unnormalized distributions, offering a scalable alternative to Markov Chain Monte Carlo (MCMC) methods. While GFNs draw inspiration from maximum entropy reinforcement learning (RL), the connection between the two has largely been unclear and seemingly applicable only in specific cases. This paper addresses the connection by constructing an appropriate reward function, thereby establishing an exact relationship between GFNs and maximum entropy RL. This construction allows us to introduce maximum entropy GFNs, which, in contrast to GFNs with uniform backward policy, achieve the maximum entropy attainable by GFNs without constraints on the state space.

1 INTRODUCTION

Generative Flow Networks (GFNs) have recently emerged as a scalable method for sampling discrete objects from high-dimensional unnormalized distributions. They transform the complexity of navigating such spaces into a sequential decision-making problem, wherein each sequence of actions yields a unique object. Although the notion of framing "inference" as a control problem has been previously explored (Fleming and Mitter, 1981; Buesing et al., 2020), Bengio et al. (2021) observe that a naive approach based on soft Q-learning (SQL; Haarnoja et al., 2017) and maximum entropy reinforcement learning (RL) tends to favor large objects disproportionately. To counter this, they introduced GFNs, a novel RL-inspired method.

While Bengio et al. (2021) established the exact equivalence between GFNs and SQL for tree-structured problems, GFNs have primarily been explored outside the theoretical confines of RL. This is due to the ambiguity over how this connection could be applicable more generally. Our work aims to bridge this knowledge gap by developing a valid GFN-like method purely from an RL standpoint. The crucial insight is that the sampling bias, identified by Bengio et al. (2021), can be mitigated by formulating a suitable reward function. Combined with the smooth Bellman equation, this leads to samples from the target distribution. We illustrate that this reward function can be efficiently obtained as the solution to an auxiliary dynamic programming problem, the number of paths leading to a specific node. Our resulting approaches, termed generative SQL, can further be interpreted as an instantiation of the concept of general value functions (Sutton et al., 2011; White, 2015).

By leveraging our newly forged theoretical connection, we find the backward policy of a GFN whose forward policy corresponds to generative SQL, which we refer to as the maximum entropy backward and can be calculated using the same dynamic program used for generative SQL. We prove that maximum entropy GFNs, GFNs that use the maximum entropy backward, indeed achieve the upper bound of entropy possible for a GFN—a claim hinted at in prior research under stringent conditions (Zhang et al., 2022).

Additionally, we reveal that applying Path Consistent Learning (PCL; Nachum et al., 2017) on our proposed reward yields “trajectory-balance” for maximum entropy GFNs.

The principal contributions of this paper are as follows:

1. We propose a reward function with the property that, once incorporated within the smooth

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

Bellman equations, leads to policies capable of sampling from the given target distribution.

2. We provide a formulation backward policy for GFNs with the same policy as the solution of the smooth Bellman equations.

3. We show that GFNs constructed in this manner have a unique solution- unlike traditional GFNs- and provably reach the maximum entropy in the general case.

4. We demonstrate through experiments that maximum entropy GFNs enhance the exploration of intermediate states and achieve better results in a hard graph-building environment.

2 BACKGROUND AND NOTATION

We provide a list of notation in Table 4 and a list of abbreviations in Table 5 in the appendix.

A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{S}_0, \mathcal{T}, \mathcal{A}, \mathbb{A}, T, \mathbb{P}_0)$ where $\mathcal{S}$ is the set of states, $\mathcal{S}_0 \subset \mathcal{S}$ is the set of initial states, and $\mathcal{T} \subset \mathcal{S}$ is the set of terminal states. Furthermore, $\mathcal{A}$ is the set of actions, the action mask $\mathbb{A} : \mathcal{S} \to P(\mathcal{A})$, where P is the power set function, is a function that defines the set of actions available at the state s, and $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the deterministic transition function that defines the next state given the current state and actions. The initial state of the trajectories is sampled from $\mathbb{P}_0 : \Delta(\mathcal{S}_0)$ where $\Delta(S_0)$ denotes the set of all distributions over the set $\mathcal{S}_0$. We deviate the notation from that of Sutton and Barto (2018) or Bengio et al. (2023) to both find a notation that helps us bridge the two topics and, at the same time, reflect the underlying assumptions and implementations better.

In the context of GFNs, Bengio et al. (2021) assume that there is a unique initial state $\mathcal{S}_0 = \{s_0\}$. Also central to the definition of GFNs is the parent function $\text{Parent} : \mathcal{S} \to P(\mathcal{S} \times \mathcal{A})$. It returns the set of state action pairs (s, a) that reach a state s'. It can be thought of as the generalized inverse of T and is defined as

$$\text{Parent}(s') = \{(s, a) \in \mathcal{S} \times \mathcal{A} | a \in \mathbb{A}(s) \wedge T(s, a) = s'\}.$$

Furthermore, GFNs assume that the transition function T is acyclic (Bengio et al., 2021), which means it is impossible to reach a state from itself. We refer to MDPs with acyclic transition functions as acyclic MDPs for brevity. An essential consequence of the acyclicity assumption is that any regularized dynamic program (DP) converges and has a unique solution (Mensch and Blondel, 2018). For acyclic MDPs, we define the marginal state distribution $\mu_\pi : \mathcal{S} \to \mathbb{R}$ as the probability of passing through a state using the policy π. The marginal $\mu_\pi(s)$ can be calculated with dynamic programming for deterministic acyclic MDPs using the following recursion:

$$\mu_\pi(s') = \sum_{(s,a) \in \text{Parent}(s')} \mu(s)\pi(a|s).$$

We note that $\mu(s) = \mathbb{P}_0(s)$ for all $s \in \mathcal{S}_0$ and $\sum_{s \in \mathcal{T}} \mu_\pi(s) = 1$. We call a state "reachable" if there exists a sequence of actions leads to that state from a state in $\mathcal{S}_0$ whose initial probability $\mathbb{P}_0$ is greater than zero. If this property holds globally, we say that MDP is reachable.

Given an unnormalized distribution on the terminal states $\tilde{p} : \Delta(\mathcal{T})$ called the target, our goal is to find a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ such that the probability of reaching a terminal state $t \in \mathcal{T}$ is proportional to $\tilde{p}(t)$. For simplicity, we assume that $\tilde{p}$ is zero for all non-terminal states. We denote the normalized version of $\tilde{p}$ as p and write the normalizing constant of the target or the partition function as Z.

2.1 GENERATIVE FLOW NETWORKS

GFNs enforce that for all terminal states $t \in \mathcal{T}$, the probability of ending the trajectory in t, i.e., $\mu(t)$, is proportional to $\tilde{p}$. In other words, the policy samples in proportion to $\tilde{p}$. They are built around the idea that the MDP is acyclic, i.e., no state can reach itself and that only one initial state s_0 exists. Concretely, any Markovian policy π that samples terminal states in proportion to $\tilde{p}$ fits the detailed balance (DB) constraints

$$F(s)\pi(a|s) = q(s, a|s')F(s'), \qquad \text{(DB)}$$

for all transition triplets $s' = T(s, a)$, and where $q : \mathcal{S} \to \Delta(\mathcal{S} \times \mathcal{A})$ is the backward policy and $F : \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the state flow function and is assumed to be equal to $\tilde{p}(t)$ for all terminal states $t \in \mathcal{T}$. The state flow and marginal are linked $F(s)/F(s_0) = \mu(s)$. Any π and $\tilde{p}$ uniquely identify the backward policy and the state flow function. Furthermore, for any π, q, and F that fit (DB), π samples in proportion to $\tilde{p}$ (Bengio et al., 2023).

Detailed balance (DB) is not the only formulation possible for a GFN; Trajectory Balance (TB; Malkin et al., 2022) is obtained by multiplying the (DB) constraint over a trajectory i.e.

$$Z \prod_{t=0}^{T-1} \pi(a_t|s_t) = \tilde{p}(s_T) \prod_{t=0}^{T-1} q(s_t, a_t|s_{t+1}), \qquad \text{(TB)}$$

where Z is shorthand for $F(s_0)$. Additional formulations for GFNs, including Bengio et al. (2021)'s original constraint, are included in Appendix C.

Bengio et al. (2021) propose minimizing the residual of a GFN constraint like (DB), which can admit an infinite number of solutions (see Appendix C). On the other hand, for any strictly concave function, there is only a unique GFN that maximizes that function. One example of a strictly concave function is the flow entropy, defined as

$$\mathbb{H}(\pi) = \mathbb{E}\left[\sum_{t=0}^{T-1} H(\pi(\cdot|s_t))\right] = \sum_{s\in\mathcal{S}} \mu_\pi(s) H(\pi(\cdot|s)),$$

where $H(\pi) = -\sum_{a\in\mathbb{A}(s)} \pi(a|s)\log\pi(a|s)$ is the entropy. In Section 3.1, we not only show that flow entropy is strictly concave, we show it is equal to the entropy of the policy. Flow entropy was first introduced by Zhang et al. (2022) where they showed that for a restrictive class of MDPs setting the backward to be uniform on the parents (i.e., $q(s,a|s') = 1/|\text{Parent}(s')|$) maximizes the flow entropy. However, Zhang et al. (2022) restrict the class of MDPs they analyze so much as to exclude the drug design MDP of Bengio et al. (2021). Concretely, Zhang et al. (2022) look at MDPs where the state is a vector of values and placeholders, and each action sets a placeholder to a value. For instance, the state could be $(\star,\star,0)$, for the placeholder $\star$, and the action "set element 0 to 1" would yield the state $(1,\star,0)$. While many MDPs used on GFNs add elements to placeholders, this definition is much more restrictive. Zhang et al. (2022)'s construction can be relaxed to having MDPs with finite and fixed horizons where the transition function is layered, meaning that there exists a partitioning of the state space $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_n$ where $\mathcal{L}_0 = \mathcal{S}_0$, $\mathcal{L}_n = \mathcal{T}$ and the parents of any set in layer $\mathcal{L}_i$ is in $\mathcal{L}_{i-1}$ for $i \geq 1$. Furthermore, they assume that the number of actions in layer $\mathcal{L}_i$ is $n-i$. These assumptions are restrictive; for instance, adding the constraint that the number of ones is always less than the number of zeros invalidates the assumptions.

2.2 SOFT Q-LEARNING

Given a reward function over transitions $R : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$, terminal reward $R_\mathcal{T} : \mathcal{T} \to \mathbb{R}$, along with the discount factor $\gamma \in (0,1]$, we can derive two fundamental functions: the state value function $V(s)$, and the state-action value function $Q(s,a)$. The state value function $V(s)$ represents the maximum discounted reward obtainable from the state s, while the state-action value function $Q(s,a)$ represents the maximum discounted reward from state s when taking the action a. The Bellman equation (Bellman, 1954) articulates the relationship between the V and Q functions. This equation can be formulated as:

$$Q(s,a) = R(s,a) + \gamma V(T(s,a)), \tag{1a}$$
$$V(s) = \max_{a\in\mathbb{A}(s)} Q(s,a). \tag{1b}$$

We assume that the value of the V function is the terminal reward at the terminal states, i.e., $V(t) = R_\mathcal{T}(t)$ for all $t\in\mathcal{T}$.

If we augment the reward with the entropy or an i.i.d. Gumbel term to the maximum in (1b), we obtain the soft (smooth) Bellman equation (Rust, 1987; Todorov, 2006; Ziebart et al., 2008; Peters et al., 2010; Rawlik et al., 2012; Fosgerau et al., 2013; Van Hoof et al., 2015; Fox et al., 2016; Nachum et al., 2017; Haarnoja et al., 2017; Geist et al., 2019; Garg et al., 2023)

$$Q(s,a) = R(s,a) + \gamma V(T(s,a)) \tag{2a}$$
$$V(s) = \max_{\pi\in\Delta(\mathbb{A}(s))} \mathbb{E}_{a\sim\pi}[Q(s,a) - \tau\log\pi_a], \tag{2b}$$

at temperature τ. If γ is less than one, the soft Bellman equation will have a unique solution (Geist et al., 2019). However, the existence of a unique solution is not guaranteed in the undiscounted case, i.e., where $\gamma = 1$. Mai and Frejinger (2022, Remark 2) give a sufficient condition for the existence of a unique solution. In the acyclic case, Mensch and Blondel (2018) showed that the soft Bellman equation has a unique solution. We note that there cannot be more than one solution to the equations as entropy is strictly concave.

Given a Q function, the value function V is equal to $V(s) = \tau\log\sum_{a\in\mathbb{A}(s)}\exp(Q(s,a)/\tau)$, and the policy is $\pi(a|s) = \frac{\exp(Q(s,a)/\tau)}{\sum_{a'\in\mathbb{A}(s)}\exp(Q(s,a')/\tau)} = \exp\left(Q(s,a)/\tau - V(s)/\tau\right)$. Taking the log on both sides of this expression is the basis for path consistency learning (PCL) (Nachum et al., 2017), which aims to enforce a temporal consistency between the policy and value function over multiple steps. For deterministic MDPs, path consistency learning enforces

$$V(s_i) + \sum_{t=i}^{j-1} \tau\gamma^{t-i}(\tau\log\pi(a_t|s_t) - R(s_t,a_t)) = \gamma^{j-i}V(s_j) \tag{PCL}$$

instead of the soft Bellman equation over a sub trajectory $s_i, a_i, \dots, s_{j-1}, a_{j-1}, s_j$. We note that the relationship between (TB) and (DB) is reminiscent of (PCL) and (2).

In the remainder of this paper, we assume $\tau = 1$ and $\gamma = 1$ for clarity.

A key property we use in our proofs is that the probability of a trajectory can be determined using the value function of its starting and ending states, along with the rewards accrued throughout that trajectory. This is further demonstrated in the following proposition.

Proposition 1. *(Fosgerau et al., 2013) in deterministic entropy regularized MDPs with no discounting we have* $\log \mathbb{P}(s_i, a_i, \ldots, s_{j-1}, a_{j-1}, s_j) = \sum_{t=i}^{j-1} \log \pi(a_t|s_t) = V(s_j) + \sum_{t=i}^{j-1} R(s_t, a_t) - V(s_i)$.

By Proposition 1, the likelihood of trajectories that share initial and terminal states is proportional to the exponent of their reward difference. If there are no intermediate rewards, they have the same probability. This property is important for our proofs.

3 FROM SOFT Q-LEARNING TO MAXIMUM ENTROPY GFNs

In this section, we derive a policy that reaches terminal states in proportion to an unnormalized distribution $\tilde{p}$ by constructing an appropriate reward under the soft Bellman equations. As a reminder, we assume an acyclic deterministic MDP with only one initial state. The acyclicity assumption reflects that we are building complex discrete objects by accumulation of primitive parts. As for the assumption of a single initial state, itc reflects the fact that we start over every time. These assumptions mirror the assumptions of GFNs. All proofs are in the appendix.

If we set $R(s, a) = 0$ and $R_{\mathcal{T}}(s_T) = \tilde{p}(s_T)$, the return for a trajectory $s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T$ is $\tilde{p}(s_T)$. By Proposition 1, the likelihood of taking the trajectory is proportional to $\exp \tilde{p}(s_T)$. Thus, we can estimate the probability of sampling any terminal state, given the number of trajectories that lead to that state.

Proposition 2. *(Proposition 1 in Bengio et al., 2021) For a terminal state $t \in \mathcal{T}$ and the numer of trajectories starting at s_0 that terminate in t, $n(t)$, the probability of a trajectory terminating in t, if we set the terminal reward $R_{\mathcal{T}}(t)$ to $\tilde{p}(t)$ and have no intermediate reward, is proportional to $n(t)\exp(\tilde{p}(t))$.*

We include the proof since it is more concise than the one in Bengio et al. (2021). In the next proposition, we modify our reward function such that SQL samples terminal states in proportion to $\tilde{p}$.

Proposition 3. *For terminal reward $R_{\mathcal{T}}(t) = \log \tilde{p}(t) - \log n(t)$, the probability of reaching the state t is proportional to $\tilde{p}(t)$.*

Definition 1. We call soft Q-Learning with the rewards $R(s, a) = 0$ and terminal rewards $R_{\mathcal{T}}(s) = \log \tilde{p}(s) - \log n(s)$ generative soft Q-learning or GSQL.

In Appendix D we show that it is possible to calculate $n(s)$ for certain MDPs with combinatorial structures but in general $n(s)$ can be calculated using DP.

Theorem 1. *The number of trajectories $n(s)$ satisfies*

$$n(s) = \sum_{(s',a') \in Parent(s)} n(s'), \tag{3}$$

and $n(s_0) = 1$.

For most MDPs, the recursion of Theorem 1 is not tractable as the size of the DP table grows exponentially with respect to the horizon. Instead, we advocate learning n. Using the inverted MDP defined below, we show that we can leverage entropy regularized RL methods to learn n.

Definition 2. The inverse of an MDP

$$(\mathcal{S}, \mathcal{S}_0, \mathcal{T}, \mathcal{A}, \mathbb{A}, T)$$

is an MDP

$$(\mathcal{S}, \mathcal{T}, \mathcal{S}_0, \mathcal{S} \times \mathcal{A}, \text{Parent}, \bar{T})$$

where the set of actions is the set of state action pairs of the original MDP, the action mask function is the set of parents, and the inverted dynamics $\bar{T}$ undoes the action such that $\bar{T}(s', (s, a)) = s$. We omit $\mathbb{P}_0$ because it does not affect the calculations.

Note that the inverse MDP is implicitly present in the definition of GFNs as the backward policy is defined on the inverted MDP and that the inverse of the inverted MDP is the original MDP. Every trajectory $s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T$ in the original MDP has a corresponding trajectory $s_T, (s_{T-1}, a_{T-1}), s_{T-1}, \ldots, (s_0, a_0), s_0$ in the inverted MDP. Using the inverted MDP, the dependence of $n(s)$ on the parents becomes a dependence on the children, giving rise to a formulation that uses a Bellman equation.

It is convenient to learn $\log n$ both for numerical purposes and for the synergy it has with Soft Q-learning. In the log space, the sum in (3) is replaced by a log-sum-exp. Indeed, as the following proposition shows, $\log n$, denoted as l, fits the soft Bellman equation.

Proposition 4. *Let $l(s) = \log n(s)$, then l is the value function of the soft Bellman equation in the inverted MDP with the rewards set to zero for all states and transitions.*

Proposition 4 allow us to use standard entropy regularized RL tools to learn n. This allows us to assert that learning n is **not** harder than learning the GFN. Indeed, as shown in the experiment section, it is often easier. Any constraint like the soft Bellman

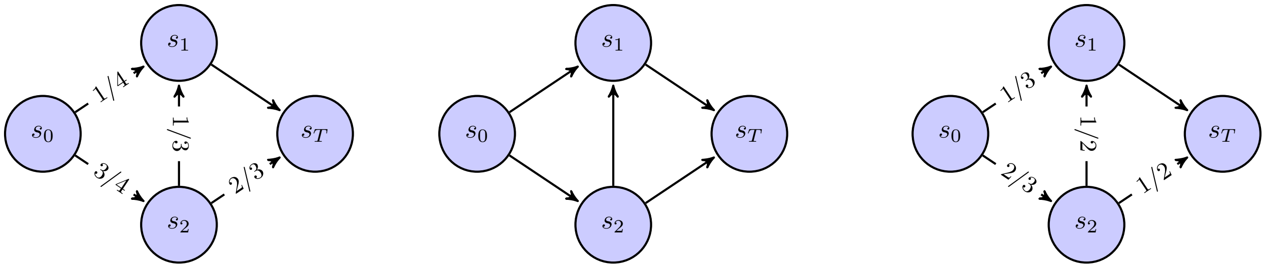


Figure 1: Comparison of maximum entropy and uniform backward. Left: uniform backward policy, middle: MDP, right: maximum entropy gflownet. The numbers are the probabilities of the policies at state s_0 and s_2.

equation and PCL can be used. If PCL is used, let $\log q((s,a)|s') = l(s) - l(s')$, the PCL equation becomes $l(s_j) + \sum_{t=i}^{j-1} \log q((s_t,a_t)|s_{t+1}) = l(s_i)$. We revisit this q in Section 3.2. Notice how the value of the initial state in the trajectories is on the right-hand side, not the left-hand side, as we are using a consistency equation on the inverted MDP.

The next proposition shows that the value initial state s_0 of GSQL equals the logarithm of total flow $\log F(s_0)$ or $\log Z$. We begin with a lemma that shows that the value of every state s is the log-sum-exp of all trajectories that pass through s. Using trajectories instead of terminal states allows us to not run into issues when a state is the descendent of multiple states that are not the descent of each other.

Lemma 1. *In finite acyclic MDPs with $R(s,a) = 0$, $V(s)$ is equal to the log-sum-exp of the reward of all trajectories that start at s.*

Equipped with Lemma 1, we can show the following proposition for GSQL.

Proposition 5. *The value function of the initial state is equal to the logarithm of the partition function, i.e., $V(s_0) = \log \sum_{s\in\mathcal{T}} \tilde{p}(s) = \log Z$.*

3.1 A different definition of flow entropy

This subsection shows that flow entropy equals entropy over trajectories and that GSQL achieves maximum entropy. We define the trajectory entropy as follows:

Definition 3. The trajectory entropy H is the entropy over a set of trajectories and is defined as

$$H(\pi) = \mathbb{E}\left[\sum_{t=0}^{T-1} \log \pi(a_t|s_t)\right] \tag{4}$$

for Markovian policies.

The definition is trivially extendable to non-Markovian policies as it is the entropy of the distribution over trajectories. The following proposition shows that $\mathbb{H}$ and H are, in fact, equal.

Proposition 6. *The trajectory entropy and flow entropy are equal for Markovian policies, i.e., $\mathbb{H}(\pi) = H(\pi)$.*

Theorem 2. *GSQL achieves the maximum entropy a policy sampling in proportion to the target can achieve.*

The uniqueness of the maximum entropy GFN is derived from the following concavity proof, which is the same as the proof given by Hoda et al. (2010) except that we define the flow for acyclic graphs, not trees.

Lemma 2. *For any concave function $H : \Delta \to \mathbb{R}$, the function $\sum_{s\in\mathcal{S}} F(s) H\left(\frac{F(s,\cdot)}{F(s)}\right)$ where $F(s,\cdot)$ is the vector of outgoing flows at state s and $F(s)$ is the sum of all outgoing flows, is concave and maximizing any such function yields a unique GFN.*

Given that the policy entropy is the flow entropy, we can specialize Lemma 2.

Proposition 7. *Markovian entropy and flow entropy is a special case of Lemma 2 and thus is strictly concave, and thus the maximum entropy GFN is unique.*

3.2 The backward of GSQL

As mentioned in the introduction, for a fixed policy and $\tilde{p}$, we can uniquely identify the backward policy q. The following identifies the backward policy of GSQL.

Theorem 3. *The backward policy q of the policy of GSQL is*

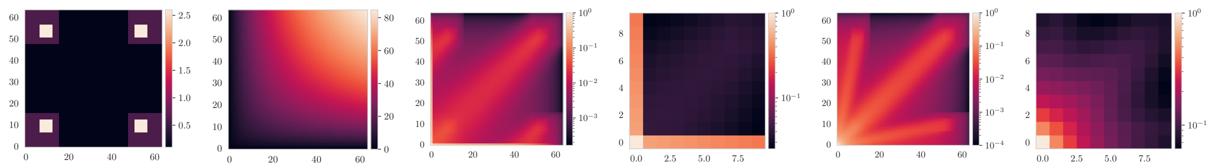
$$q(s,a|s') = n(s)/n(s'). \tag{5}$$

The backward proposed here can be used to train maximum entropy GFNs.

Remark 1. The backward policy q gives a uniform distribution over all backward trajectories leading to s_0, given $s_T \in \mathcal{T}$.

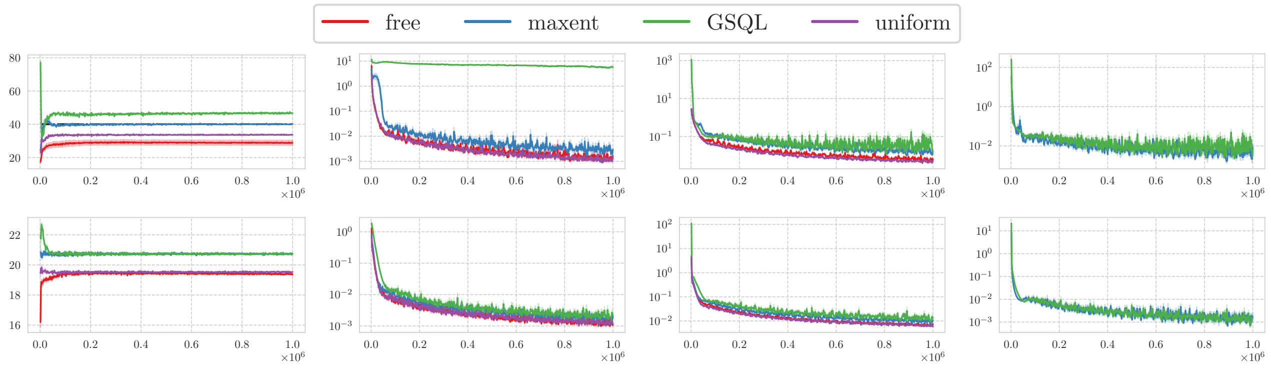
Remark 2. For a sub trajectory $s_i, a_i, \ldots, s_{j-1}, a_{j-1}, s_j$, $\prod_{t=i}^{j-1} q(s_t, a_t|s_{t+1}) = n(s_i)/n(s_j)$.

Remark 3. We find the uniform backward if the number of paths to all parent nodes is equal. This



(a) $\tilde{p}$ (b) l (c) uniform μ (d) uniform μ zoomed (e) maxent μ (f) maxent μ zoomed

Figure 2: From left to right, target, $l =$, and marginal of the uniform backward and maximum entropy GFNs for the 64^2 grid. Note the log scale colors for μ and the non-smooth partitioning of the flow around the bottom and left edges with the uniform backward policy.



(a) entropy (b) KL (c) policy loss (d) n loss

Figure 3: Various metrics for the hypergrid experiments. The top row is for the 64^2 grid, and the bottom row is for the 8^4 grid. The black line shows the maximum entropy attainable by a GFN. Log y-scale was used.

can happen if the MDP is layered and the number of parents is the same for all nodes in all layers. Thus, the maximum entropy backward subsumes the uniform backward.

Note that the uniform backward is not always maximum entropy. For instance, as shown in Appendix D, it is not for any of the MDPs we analyze.

One major difference between GSQL and maximum entropy GFNs is what happens if n is not learned to high fidelity. If $n(s)$ is greater than zero, then maximum entropy GFNs will still be GFNs yet GSQL does not have this property.

Lastly, unlike the uniform backward policy, the maximum entropy backward relaxes the reachability assumption. This is because n is zero for unreachable states. However, this may require sampling backward trajectories similar to Zhang et al. (2022), which is beyond the scope of this work.

3.3 Remarks on PCL

Nachum et al. (2017) showed that if (PCL) holds for all sub-trajectories of a certain length, then the soft Bellman equation holds and vice versa. We extend the result to full trajectories.

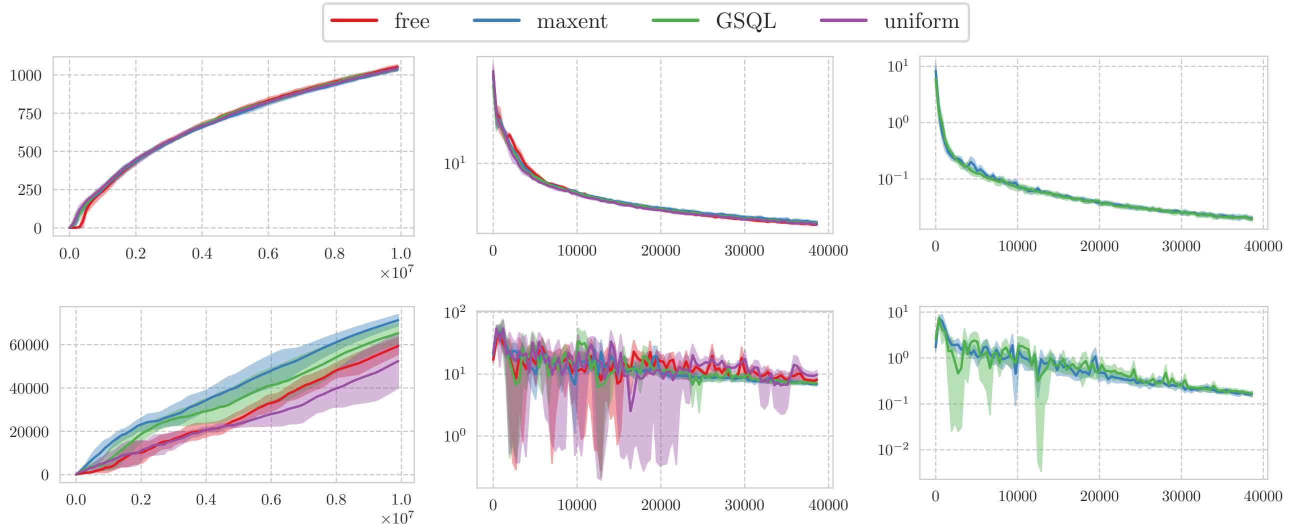
Proposition 8. *If PCL holds for all sub-trajectories reaching a terminal state* $(s_i, a_i, \dots, s_{T-1}, a_{T-1}, s_T)$*, all trajectories starting at an initial state* $(s_0, a_0, \dots, s_{i-1}, a_{i-1}, s_i)$*, or all full trajectories* $(s_0, a_0, \dots, s_{T-1}, a_{T-1}, s_T)$*, then it holds for all sub-trajectories. We call these conditions the terminal, initial, and trajectory PCL conditions, respectively.*

We can use PCL to calculate n using Proposition 4.

Proposition 9. *Given n, maximum entropy GFNs with (TB) and GSQL with trajectory PCL have the same residual and gradient.*

4 EXPERIMENTS

This section is divided into three subsections, each focusing on MDPs that do not follow the assumptions of Zhang et al. (2022). They help illustrate that it is not hard to find MDPs where the uniform backward is not maximum entropy. We compare maximum entropy GFNs and GSQL with GFNs with uniform and learned backward policies.



(a) number of modes found

(b) policy loss

(c) n loss

Figure 4: Experiment statistics. Confidence intervals show the IQM. From top to bottom, the rows belong to the sEH and QM9 experiments.

Appendix E presents all experiment details, pseudocode, and further experiments.

4.1 A simple MDP

In the MDP presented in Figure 1, there are three paths from the initial state s_0 to the destination s_T: s_0, s_1, s_T, s_0, s_2, s_1, s_T, and s_0, s_2, s_T. The uniform backward gives $q(s_1|s_T) = q(s_2|s_T) = q(s_0|s_1) = q(s_2|s_1) = 1/2$; thus, the paths have probabilities 0.25, 0.25, and 0.5, respectively, yielding an entropy of $3/2 \log 2$. The maximum entropy GFN yields the backward $q(s_1|s_T) = 2/3$ and $q(s_2|s_T) = 1/3$ since there are 2 paths from s_1 but only one path from s_2. Furthermore, $q(s_2|s_1) = q(s_0|s_1) = 1/2$, since only one path per parent exists. The entropy is $\log 3$, which is the maximum entropy.

4.2 Hypergrid

We now focus on the hypergrid domain from Bengio et al. (2021). The state is a vector that starts at 0. At each step, the agent can increase one of the components of the vector by one, i.e., move one step in one of the directions or terminate the episode. The agent is in a bounded box, and the target function is given by

$$0.1 + 0.5\prod_i \mathbb{I}[0.25 < |s_i - 0.5|] + 2\prod_i \mathbb{I}[0.3 < |s_i - 0.5| < 0.4], \quad (6)$$

for the indicator function $\mathbb{I}$ and the ratio of the current position and the maximum allowed position in component i, s_i. In Figure 2, for a 64×64 grid, we show the target, the logarithm of the number of paths l, and the marginal distribution of uniform backward policy and the maximum entropy GFNs. First, note that l gets very big. We cannot store $n = \exp(l)$ as a 64-bit integer as n for the top right corner is $(2 \times 63)!/63!/63!$. Second, notice how the maximum entropy GFN distributes the flow more evenly and does not accumulate flow on the bottom and left edges.

We test the models on the 64^2 and 8^4 hypergrids to reach terminal states in proportion to (6). The results can be seen in Figure 3. In the 64^2 experiments, GSQL failed to reach all modes but worked fine on the 8^4 grid, whereas maximum entropy GFN successfully learns to sample in proportion to the target. The failure of GSQL is minor as it can be alleviated with added exploration, yet it illustrates the difference between GSQL and maximum entropy GFNs. Indeed, maximum entropy GFNs, unlike GSQL, are GFNs regardless of the quality of the estimated n. We note that since GSQL fails to learn the target distribution in the 64^2 experiment, the fact that its entropy is higher than the maximum entropy is irrelevant.

4.3 Molecule design

In this section, we examine the sEH task of Bengio et al. (2021) and the QM9 task of Jain et al. (2023). We use a tree-building MDP in the sEH (Jin et al., 2018) and a graph-building MDP in the QM9 experiments. In the tree-building environment, every state is a tree, and each action either adds a node and connects it to an existing node or sets an attribute on each edge. In the sEH experiments, each node

Table 1: KL divergence of the policy of different models on small sEH task of Shen et al. (2023). The cells show the average KL divergence between the row and column policies.

		free	uniform	maxent		GSQL	
	n			known	learned	known	learned
free		0.32 ± 0.06	12.61 ± 0.23	11.03 ± 0.23	11.27 ± 0.23	11.13 ± 0.23	11.29 ± 0.22
uniform		10.69 ± 0.37	$\mathbf{0.08 \pm 0.01}$	0.38 ± 0.01	0.41 ± 0.01	0.39 ± 0.01	0.40 ± 0.02
maxent	known	10.02 ± 0.37	0.45 ± 0.01	$\mathbf{0.05 \pm 0.01}$	$\mathbf{0.09 \pm 0.01}$	$\mathbf{0.08 \pm 0.01}$	$\mathbf{0.10 \pm 0.01}$
	learned	9.83 ± 0.36	0.53 ± 0.03	$\mathbf{0.09 \pm 0.01}$	$\mathbf{0.05 \pm 0.01}$	$\mathbf{0.10 \pm 0.01}$	$\mathbf{0.07 \pm 0.01}$
GSQL	known	10.22 ± 0.37	0.56 ± 0.02	$\mathbf{0.10 \pm 0.01}$	$\mathbf{0.12 \pm 0.01}$	$\mathbf{0.09 \pm 0.01}$	$\mathbf{0.12 \pm 0.01}$
	learned	9.94 ± 0.37	0.52 ± 0.01	$\mathbf{0.11 \pm 0.01}$	$\mathbf{0.09 \pm 0.01}$	$\mathbf{0.11 \pm 0.01}$	$\mathbf{0.08 \pm 0.01}$

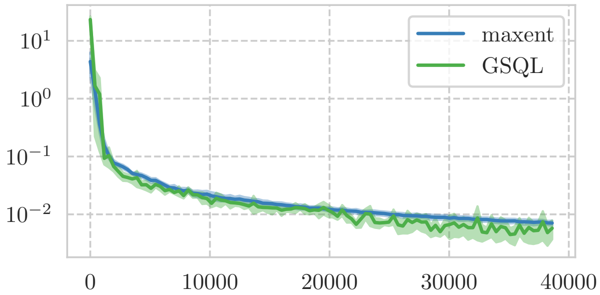


Figure 5: MSE of the learned n and the ground truth in the sEH experiments.

represents a fragment (collection of atoms), and edge features show the place of connection of two fragments. In the graph-building environment, every state is a connected graph (every node is reachable from every other node), and actions either add a new node and edge or set the attribute on edge.

The graph-building environment is more expressive than the tree-building environment. However, unlike the tree-building environment, the graph-building environment can lead to molecules that cannot exist and are invalid.

The tree-building MDP is much more structured, and we can calculate n for each state directly to verify that we are, in fact, learning (see Appendix D). Figure 5 shows the mean squared error between the predicted $\log n$ and the real $\log n$. The difference between the learned n and the ground truth is small.

We show the KL divergence between the policies found using different methods in Table 1 when trained using the small sEH MDP of Shen et al. (2023). Notice how GFNs with free backward policy have a higher KL divergence among themselves than the other policies. Indeed, this results from the GFN constraints having infinite feasible solutions for this MDP. As expected, GSQL and maximum entropy GFNs find close policies. We also note that they are close regardless of whether n is given or learned. Lastly, as shown by the higher KL divergence between the GFNs with uniform backward policy and maximum entropy GFNs, the other two find different policies.

Next, we focus on the full sEH task; we show the GFN and n losses in the top row of Figure 4b and Figure 4c. Not only n is learnable, but it is also is easier to learn than the policy constraint as the reward is always zero. As shown in the top row of Figure 4a, GFNs with free backward policy start finding modes with a short lag. We show some statistics after training in Table 2. The top K statistics are the same for all policies thus we can infer that all models are GFNs regardless of the backward but not using a free backward policy helps at the initial phase of the training and using maximum entropy backward policy policy increases the entropy. We refer to Appendix E for a definition of the top K statistics.

Our last set of experiments focuses on the QM9 experiments. While we calculated n for the sEH experiments, we do not do so here as it is prohibitively expensive to do via the general recursion, and we could not find a combinatorial structure to exploit. As shown in the second row of Figure 4b and Figure 4c, it is harder than the sEH experiment, but the n objective is still easier than the policy objective. The bottom row of Figure 4a shows that maxent GFNs outperform GSQL, and GSQL outperforms the other two GFNs. Table 3 shows some statistics. We note that we put a hard limit of 100,000 modes as the cost of finding modes grows quadratically. In this experiment, maxent GFN and GSQL backward find more modes. Again, the key takeaway is that as we deviate more from the assumptions of Zhang et al. (2022), the performance of GFNs with uniform backward worsen.

5 CONCLUSION

This work showed an equivalence between entropy-regularized RL and GFNs. Concretely, we showed

Table 2: sEH experiment results.

model	n	entropy	diverse top K reward	top K diversity	top K reward	modes with $\tilde{p} \geq 1$	modes with $\tilde{p} \geq 0.875$
free	none	77.92 ± 0.02	1.03	0.46	1.03	$\mathbf{1049.90 \pm 5.66}$	3378.00 ± 24.30
uniform	none	77.27 ± 0.02	1.03	0.46	1.03	1042.90 ± 5.21	3318.30 ± 31.15
GSQL	learned	$\mathbf{78.42 \pm 0.02}$	1.03	0.46	1.03	1043.90 ± 7.18	$\mathbf{3446.50 \pm 40.27}$
	known	$\mathbf{78.39 \pm 0.01}$	1.03	0.46	1.03	1036.60 ± 7.74	3379.90 ± 29.55
maxent	learned	$\mathbf{78.41 \pm 0.03}$	1.03	0.46	1.03	1035.70 ± 5.56	3417.60 ± 32.82
	known	$\mathbf{78.41 \pm 0.02}$	1.03	0.46	1.03	1042.50 ± 5.98	$\mathbf{3344.00 \pm 44.67}$

Table 3: QM9 experiment results.

model	entropy	diverse top K reward	top K diversity	top k reward	modes with $\tilde{p} \geq 1.125$	modes with $\tilde{p} \geq 1$
free	93.41 ± 0.24	**1.20**	0.83	**1.20**	$60\,465 \pm 2157$	100 000
uniform	89.96 ± 7.61	1.19 ± 0.01	0.81 ± 0.04	1.19 ± 0.01	$52\,865 \pm 4481$	100 000
GSQL	$\mathbf{98.59 \pm 0.30}$	1.19	**0.84**	1.19	$63\,909 \pm 2730$	100 000
maxent	$\mathbf{98.22 \pm 0.11}$	**1.20**	0.83	**1.20**	$\mathbf{71\,339 \pm 1468}$	100 000

that using the number of trajectories to each terminal state, we can create a reward such that the probability of reaching terminal states of entropy-regularized RL is proportional to an unnormalized distribution. We use the equivalence to show the equivalence of (TB) and (PCL) for a specific class GFNs and entropy regularized RL with our proposed reward.

Building on top of our proposed extension of entropy regularized RL, we introduced maximum entropy GFNs, subsuming the uniform backward policy of Zhang et al. (2022). A benefit of our model is that it automatically falls back to the uniform backward policy whenever the uniform backward policy is maximum entropy. We showed that we can learn n using the soft Bellman equation on the inverted MDP, as calculating it via the recursion may be too expensive, and a combinatorial structure may be hard to identify, as is the case with the graph-building environment used in the QM9 experiments. We then verified that n is indeed learnable empirically.

While we mirror the assumptions of Bengio et al. (2021), our analysis is limited to undiscounted MDPs with deterministic transitions and only one initial state. Solving the same problem with stochastic transitions may be of interest (e.g. Pan et al., 2023) but is more complex. For instance, while the reachability of the terminal states guarantees a solution in the deterministic case, a solution may not exist in the stochastic formulation of Pan et al. (2023). Jiralerspong et al. (2023)'s formulation for GFNs in stochastic MDPs overcome the feasibility issue of Pan et al. (2023)'s formulation and maximum entropy GFNs may help relax the tree assumption of Jiralerspong et al. (2023) yet maintain their equilibrium results. However, it is unclear how to define the corresponding inverse MDP properly.

Independently, Tiapkin et al. (2023) provide an alternative way of obtaining GFNs from soft Q-learning given a fixed backward policy. Tiapkin et al. (2023) method can be used in conjunction with the maximum entropy backward policy introduced here. Limiting ourselves to maximum entropy makes our proposed method simpler and does not conceptually depend on GFNs. With the equivalence results, many RL algorithms like Munchausen Vieillard et al. (2020) are now directly applicable to GFNs. Future work will look at the improvements those methods can bring.

Derman et al. (2021) pointed out that entropy-regularized RL is equivalent to a particular robust reinforcement learning formulation. However, it is unclear how such a result would apply meaningfully here as the uncertainty set that Derman et al. (2021) proposes would give non-zero reward to transitions, something we cannot have with GFNs.

Current GFN environments are too structured. Most non-toy environments have many of the features Zhang et al. (2022) require. For instance, the MDPs we analyze are all layered but do not have the same number of parents. In the sEH experiment, the bulk of the high-reward molecules are in the final layer. We expect as the MDPs used become more and more complex, maximum entropy GFNs shine more as they did in QM9 experiment compared to the sEH experiment.

Acknowledgments

We thank Nikolay Malkin for comments about uniform backward policy and Gabriele Farina for comments about dilated entropy. This research was supported by compute resources provided by Calcul Quebec (`calculquebec.ca`), the BC DRI Group, the Digital

Research Alliance of Canada (alliancecan.ca), and Mila (mila.quebec).

References

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. (2021). Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394.

Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. (2023). Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55.

Buesing, L., Heess, N., and Weber, T. (2020). Approximate inference in discrete distributions with monte carlo tree search and value functions. In Chiappa, S. and Calandra, R., editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 624–634. PMLR.

Deleu, T., Góis, A., Emezue, C., Rankawat, M., Lacoste-Julien, S., Bauer, S., and Bengio, Y. (2022). Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence*, pages 518–528. PMLR.

Derman, E., Geist, M., and Mannor, S. (2021). Twice regularized mdps and the equivalence between robustness and regularization. *Advances in Neural Information Processing Systems*, 34:22274–22287.

Fleming, W. H. and Mitter, S. K. (1981). Optimal control and nonlinear filtering for nondegenerate diffusion processes. Technical Report ADA117069, Brown University, Lefschetz Center for Dynamical Systems, Providence, RI. Approved for public release.

Fosgerau, M., Frejinger, E., and Karlstrom, A. (2013). A link based network route choice model with unrestricted choice set. *Transportation Research Part B: Methodological*, 56:70–80.

Fox, R., Pakman, A., and Tishby, N. (2016). Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, page 202–211. AUAI Press.

Garg, D., Hejna, J., Geist, M., and Ermon, S. (2023). Extreme q-learning: Maxent rl without entropy. In *The Eleventh International Conference on Learning Representations*.

Geist, M., Scherrer, B., and Pietquin, O. (2019). A theory of regularized markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169. PMLR.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (2004). *Fundamentals of convex analysis*. Springer Science & Business Media.

Hoda, S., Gilpin, A., Peña, J., and Sandholm, T. (2010). Smoothing Techniques for Computing Nash Equilibria of Sequential Games. *Mathematics of OR*, 35(2):494–512.

Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pages 492–518. Springer.

Jain, M., Raparthy, S. C., Hernández-García, A., Rector-Brooks, J., Bengio, Y., Miret, S., and Bengio, E. (2023). Multi-objective gflownets. In *International Conference on Machine Learning*, pages 14631–14653. PMLR.

Jin, W., Barzilay, R., and Jaakkola, T. (2018). Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR.

Jiralerspong, M., Sun, B., Vucetic, D., Zhang, T., Bengio, Y., Gidel, G., and Malkin, N. (2023). Expected flow networks in stochastic environments and two-player zero-sum games. *arXiv preprint arXiv:2310.02779*.

Kim, M., Yun, T., Bengio, E., Zhang, D., Bengio, Y., Ahn, S., and Park, J. (2023). Local search gflownets. *arXiv preprint arXiv:2310.02710*.

Landrum, G., Tosco, P., Kelley, B., Ric, Cosgrove, D., sriniker, gedeck, Vianello, R., NadineSchneider, Kawashima, E., N, D., Jones, G., Dalke, A., Cole, B., Swain, M., Turk, S., AlexanderSavelyev, Vaucher, A., Wójcikowski, M., Take, I., Probst, D., Ujihara, K., Scalfani F., V., Godin, G., Lehtivarjo, J., Walker, R., Pahl, A., Berenger, F., jasondbiggs, and strets123 (2023). Rdkit.

Madan, K., Rector-Brooks, J., Korablyov, M., Bengio, E., Jain, M., Nica, A. C., Bosc, T., Bengio, Y., and Malkin, N. (2023). Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pages 23467–23483. PMLR.

Mai, T. and Frejinger, E. (2022). Undiscounted recursive path choice models: Convergence properties and algorithms. *Transportation Science*, 56(6):1469–1482.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. (2022). Trajectory balance: Improved credit assignment in gflownets. In *Advances in Neural Information Processing Systems*.

Mensch, A. and Blondel, M. (2018). Differentiable dynamic programming for structured prediction and attention. In *International Conference on Machine Learning*, pages 3462–3471. PMLR.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. (2017). Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, 30.

Pan, L., Zhang, D., Jain, M., Huang, L., and Bengio, Y. (2023). Stochastic generative flow networks. *arXiv preprint arXiv:2302.09465*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Peters, J., Mulling, K., and Altun, Y. (2010). Relative entropy policy search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):1607–1612.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7.

Rawlik, K., Toussaint, M., and Vijayakumar, S. (2012). On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*.

Rust, J. (1987). Optimal replacement of gmc bus engines: An empirical model of harold zurcher. *Econometrica: Journal of the Econometric Society*, pages 999–1033.

Shen, M. W., Bengio, E., Hajiramezanali, E., Loukas, A., Cho, K., and Biancalani, T. (2023). Towards understanding and improving gflownet training. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23.

Shwartz, A. (2001). Death and discounting. *IEEE Transactions on Automatic Control*, 46(4):644–647.

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

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768.

Tiapkin, D., Morozov, N., Naumov, A., and Vetrov, D. (2023). Generative flow networks as entropy-regularized rl. *arXiv preprint arXiv:2310.12934*.

Todorov, E. (2006). Linearly-solvable markov decision problems. *Advances in neural information processing systems*, 19.

Van Hoof, H., Peters, J., and Neumann, G. (2015). Learning of non-parametric control policies with high-dimensional state features. In *Artificial Intelligence and Statistics*, pages 995–1003. PMLR.

Vieillard, N., Pietquin, O., and Geist, M. (2020). Munchausen reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4235–4246.

White, A. (2015). Developing a predictive approach to knowledge.

Zhang, D., Malkin, N., Liu, Z., Volokhova, A., Courville, A., and Bengio, Y. (2022). Generative flow networks for discrete probabilistic modeling. In *International Conference on Machine Learning*, pages 26412–26428. PMLR.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, volume 8, pages 1433–1438.

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. No
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes
   (b) Complete proofs of all theoretical results. Yes
   (c) Clear explanations of any assumptions. Yes

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). No
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). No

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Not Applicable
   (b) The license information of the assets, if applicable. Not Applicable
   (c) New assets either in the supplemental material or as a URL, if applicable. Yes
   (d) Information about consent from data providers/curators. Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable

A PROOFS FOR SECTION 2 (BACKGROUND AND NOTATION)

Proposition 1. *(Fosgerau et al., 2013) in deterministic entropy regularized MDPs with no discounting we have* $\log \mathbb{P}(s_i, a_i, \ldots, s_{j-1}, a_{j-1}, s_j) = \sum_{t=i}^{j-1} \log \pi(a_t|s_t) = V(s_j) + \sum_{t=i}^{j-1} R(s_t, a_t) - V(s_i)$.

Proof. This can be verified easily as

$$\log \mathbb{P}(s_i, a_i, \ldots, s_{j-1}, a_{j-1}, s_j) = \sum_{t=i}^{j-1} \log \pi(a_t|s_t) \tag{7}$$

$$= \sum_{t=i}^{j-1} Q(s_t, a_t) - V(s_t) \tag{8}$$

$$= \sum_{t=i}^{j-1} R(s_t, a_t) + V(s_{t+1}) - V(s_t) \tag{9}$$

simplifying (9) yields

$$\log \mathbb{P}(s_i, a_i, \ldots, s_{j-1}, a_{j-1}, s_j) = V(s_j) + \sum_{t=i}^{j-1} R(s_t, a_t) - V(s_i). \tag{10}$$

Note that for whole trajectories $r = s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T$, (10) is equal to

$$\log \mathbb{P}(s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T) = R_{\mathcal{T}}(S_T) + \sum_{t=0}^{T-1} R(s_t, a_t) - V(s_0). \tag{11}$$

□

B PROOFS FOR SECTION 3 (FROM SOFT Q-LEARNING TO MAXIMUM ENTROPY GFNs)

Proposition 2. *(Proposition 1 in Bengio et al., 2021) For a terminal state $t \in \mathcal{T}$ and the numer of trajectories starting at s_0 that terminate in t, $n(t)$, the probability of a trajectory terminating in t, if we set the terminal reward $R_{\mathcal{T}}(t)$ to $\tilde{p}(t)$ and have no intermediate reward, is proportional to $n(t)\exp(\tilde{p}(t))$.*

Proof. The probability of taking any of the trajectories ending in t is proportional to $\exp(\tilde{p}(t))$, and there are $n(t)$ trajectories; thus, the probability of reaching the state t is proportional to $n(t)\exp(\tilde{p}(t))$. □

Proposition 3. *For terminal reward $R_{\mathcal{T}}(t) = \log \tilde{p}(t) - \log n(t)$, the probability of reaching the state t is proportional to $\tilde{p}(t)$.*

Proof. If $R_{\mathcal{T}}(t) = \log \tilde{p}(t) - \log n(t)$, then the probability of any trajectory ending in t is proportional to $\tilde{p}(t)/n(t)$, and thus the probability of reaching t is $\tilde{p}(t)$. □

Theorem 1. *The number of trajectories $n(s)$ satisfies*

$$n(s) = \sum_{(s',a') \in Parent(s)} n(s'), \tag{3}$$

and $n(s_0) = 1$.

Proof. Let $\mathbb{L}(s)$ be the set of paths from s_0 to s; every trajectory in $\mathbb{L}(s)$ comes from one of the parents of s. Thus, every trajectory in $\mathbb{L}(s)$ is the concatenation of a trajectory that leads to a parent of s, one action that leads to s, and s. Hence, for every trajectory to a parent of s and an action a that leads to s, we have a unique trajectory that leads to s. □

Proposition 4. *Let $l(s) = \log n(s)$, then l is the value function of the soft Bellman equation in the inverted MDP with the rewards set to zero for all states and transitions.*

Proof. Since l is the value function of the soft Bellman equation,

$$l(s') = \log \sum_{(s,a)\in \text{Parent}(s')} \exp(l(\bar{T}(s', (s,a)))) = \log \sum_{(s,a)\in \text{Parent}(s')} \exp l(s), \tag{12}$$

holds since the reward is zero. Taking the exponent of both sides yields

$$n(s') = \exp l(s') = \sum_{(s,a)\in \text{Parent}(s')} \exp l(s) = \sum_{(s,a)\in \text{Parent}(s')} n(s). \tag{13}$$

Furthermore, since the value of the terminal state is zero, the value of the original initial states is zero, i.e., $\exp l(s_0) = \exp 0 = 1 = n(s_0)$. A unique solution is guaranteed by Propostion 2 of Mensch and Blondel (2018). □

Lemma 1. *In finite acyclic MDPs with $R(s,a) = 0$, $V(s)$ is equal to the log-sum-exp of the reward of all trajectories that start at s.*

Proof. We show the lemma by induction on the depth of the topological sort of the states. The result is trivial for terminal states $t \in \mathcal{T}$ as $V(t) = R_{\mathcal{T}}(s)$. If the proposition is true for all states with a depth of at least i, for any state s at depth $i-1$ we have that $V(s) = \log \sum_{a\in\mathbb{A}(s)} \exp Q(s,a) = \log \sum_{a\in\mathbb{A}(s)} \exp V(s')$ where $s' = T(s,a)$. Since we know that $V(s')$ is the logarithm of the sum of the rewards of all trajectories that start at s', $\exp V(s')$ is the sum of the reward of all trajectories that start with s, a. Since the sum is over all actions, $V(s)$ is the logarithm of the sum of the rewards of all trajectories starting at s. □

Proposition 5. *The value function of the initial state is equal to the logarithm of the partition function, i.e., $V(s_0) = \log \sum_{s\in\mathcal{T}} \tilde{p}(s) = \log Z$.*

Proof. For each terminal state $t \in \mathcal{T}$, there are $n(t)$ trajectories, each with reward $\log \tilde{p}(t) - \log n(t)$; thus, the sum-exp of their reward is $n(t)\exp(\log \tilde{p}(t) - \log n(t)) = \tilde{p}(t)$. Taking the sum of the sum-exp of reward over the set of all terminal states results in the sum-exp over the set of all trajectories:

$$V(s_0) = \log \sum_{s_0,a_0,\dots,s_T} \tilde{p}(s_T)/n(s_T) \tag{14}$$

$$= \log \sum_{s_T\in\mathcal{T}} \sum_{s_0,a_0,\dots,s_T} \tilde{p}(s_T)/n(s_T), \tag{15}$$

$$= \log \sum_{t\in\mathcal{T}} \tilde{p}(t) = \log Z. \tag{16}$$

The third equality holds because there are $n(s_T)$ trajectories for the destination s_T, we remove the second summation and the corresponding division. □

Proposition 6. *The trajectory entropy and flow entropy are equal for Markovian policies, i.e., $\mathbb{H}(\pi) = H(\pi)$.*

Proof. We start with the definition of flow entropy, let $\pi(\cdot|s_t)$ be the distribution over actions at state s_t, and $\bar{\mathcal{T}}$ the complement of $\mathcal{T}$ or the set of all non terminal states, we have:

$$\mathbb{H}(\pi) = \mathbb{E}_{s_0,a_0,\dots,s_T}[\sum_{t=0}^{T-1} H(\pi(\cdot|s_t))]. \tag{17}$$

Using the linearity of the expectation, we move the summation out of the expectation to get:

$$= \sum_{s\in\bar{\mathcal{T}}} \mathbb{P}(\cdots s \cdots) H(\pi(\cdot|s)) \tag{18}$$

Since the probability of all trajectories that pass through s is equal to the probability to partial trajectories that end in s, we can further simplify the summation to:

$$= \sum_{s \in \bar{\mathcal{T}}} \mathbb{P}(\cdots s) H(\pi(\cdot|s) \tag{19}$$

$$= \sum_{s \in \bar{\mathcal{T}}} \mathbb{P}(\cdots s) \sum_{a \in \mathbb{A}(s)} \pi(a|s) \log \pi(a|s) \tag{20}$$

Using the Bayes' rule, we have that $\mathbb{P}(\cdots s)\pi(a|s) = \mathbb{P}(\cdots sa)$.

$$= \sum_{s \in \bar{\mathcal{T}}, a \in \mathbb{A}(s)} \mathbb{P}(\cdots sa) \log \pi(a|s) \tag{21}$$

$$= \sum_{s \in \bar{\mathcal{T}}, a \in \mathbb{A}(s)} \mathbb{P}(\cdots sa \cdots) \log \pi(a|s) \tag{22}$$

$$= \mathbb{E}_{s_0, a_0, \ldots, s_T}[\sum_{t=0}^{T-1} \log \pi(a_t|s_t)] \tag{23}$$

$$= H(\pi). \tag{24}$$

□

Theorem 2. *GSQL achieves the maximum entropy a policy sampling in proportion to the target can achieve.*

Proof. Given the elementary property that $H(X, Y) = H(X) + \mathbb{E}[H(Y|X)]$, the trajectory entropy is the sum of entropy of the destination $H(p)$ (recall that p is a distribution proportional to $\tilde{p}$) and the expected conditional entropy on the destinations $\mathbb{E}[H(\pi|s_T)]$ i.e.

$$H(\pi) = H(p) + \mathbb{E}_{s_T \sim p}[H(\pi|s_T)]. \tag{25}$$

Since $H(p)$, and the distribution of s_T is part of the problem, we can maximize the interior of the expectation. The uniform distribution achieves maximum entropy, and our SQL samples uniformly on trajectories conditioned on the destination (as they have the same reward), thus, GSQL achieves maximum entropy. □

Lemma 2. *For any concave function $H : \Delta \to \mathbb{R}$, the function $\sum_{s \in \mathcal{S}} F(s) H\left(\frac{F(s,\cdot)}{F(s)}\right)$ where $F(s, \cdot)$ is the vector of outgoing flows at state s and $F(s)$ is the sum of all outgoing flows, is concave and maximizing any such function yields a unique GFN.*

Proof. The proof uses the dilation or perspective operation (Section 2.2 of Hiriart-Urruty and Lemaréchal, 2004). For a strictly concave function $H(x)$, the dilated function $yH(x/y)$, where $x \in \mathbb{R}^n_{\geq 0}$ and $y \in \mathbb{R}_{\geq 0}$, is also strictly concave. We assume that $yH(x/y)$ is zero when y is zero.

The function H is strictly concave, so the dilated function $F(s)H(F(s, \cdot)/F(s))$ is also strictly concave.

Since the flow entropy is strictly concave with respect to the state and state-action flow vectors, maximizing the flow entropy under the GFN constraints will yield a unique policy as the GFN constraints are linear. □

Proposition 7. *Markovian entropy and flow entropy is a special case of Lemma 2 and thus is strictly concave, and thus the maximum entropy GFN is unique.*

Proof. Since $\pi(a|s) = F(s, a)/F(s)$, the entropy (see Proposition 6), $\sum_{s \in \mathcal{S}} F(s)H(F(s, \cdot)/F(s))$, is strictly concave. □

Theorem 3. *The backward policy q of the policy of GSQL is*

$$q(s, a|s') = n(s)/n(s'). \tag{5}$$

Proof. By the definition of n, q is a distribution as it is both positive and sums to one over the set of all parents. To show that q is the backward policy of GSQL, we start with Proposition 5 and show that (TB) holds for the proposed backward policy.

By Proposition 5 we have that

$$Z = \exp V(s_0). \tag{26}$$

Multiplying both sides by $\exp\left(\log \tilde{p}(s_T) - V(s_0) - \log n(s_T)\right)$ and simplifying the $\exp \log$ on the left hand side yields

$$Z \exp(\log \tilde{p}(s_T) - \log n(s_T) - V(s_0)) = \tilde{p}(s_T)/n(s_T). \tag{27}$$

Since $V(s_T) = \log \tilde{p}(s_T) - \log n(s_T)$ and $n(s_0) = 1$ we get

$$Z \exp(V(s_T) - V(s_0)) = \tilde{p}(s_T) n(s_0)/n(s_T). \tag{28}$$

We then use the fact that $V(s_T) - V(s_0) = \sum_{t=0}^{T-1} V(s_{t+1}) - V(s_t)$ and $n(s_t)/n(s_T) = \prod_{t=0}^{T-1} n(s_t)/n(s_{t+1})$ to get

$$Z \prod_{t=0}^{T-1} \exp(V(s_{t+1}))/\exp(V(s_t)) = \tilde{p}(s_T) \prod_{t=0}^{T-1} n(s_t)/n(s_{t+1}). \tag{29}$$

Using the definition of q and Proposition 1 we get

$$Z \prod_{t=0}^{T-1} \pi(a_t|s_t) = \tilde{p}(s_T) \prod_{t=0}^{T-1} q(a_t, s_t|s_{t+1}), \tag{30}$$

which is (TB). □

Remark 1. The backward policy q gives a uniform distribution over all backward trajectories leading to s_0, given $s_T \in \mathcal{T}$.

Proof. Since $n(s_t)/n(s_{t+1})$ of the trajectories to s_{t+1} come through the action a_t, the uniform distribution over the trajectories needs to follow that ratio as well. □

Remark 2. For a sub trajectory $s_i, a_i, \ldots, s_{j-1}, a_{j-1}, s_j$, $\prod_{t=i}^{j-1} q(s_t, a_t|s_{t+1}) = n(s_i)/n(s_j)$.

Proof. $\prod_{t=i}^{j-1} q(s_t, a_t|s_{t+1}) = \prod_{t=i}^{j-1} n(s_t)/n(s_{t+1}) = n(s_i)/n(s_j)$. □

Proposition 8. *If PCL holds for all sub-trajectories reaching a terminal state* $(s_i, a_i, \ldots, s_{T-1}, a_{T-1}, s_T)$*, all trajectories starting at an initial state* $(s_0, a_0, \ldots, s_{i-1}, a_{i-1}, s_i)$*, or all full trajectories* $(s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T)$*, then it holds for all sub-trajectories. We call these conditions the terminal, initial, and trajectory PCL conditions, respectively.*

Proof. If PCL holds for two trajectories $s_i, a_i, \ldots, s_{T-1}, a_{T-1}, s_T$ and $s_{i+1}, a_{i+1}, \ldots, s_{T-1}, a_{T-1}, s_T$, then it holds for $s_i a_i s_{i+1}$, thus if terminal PCL holds, PCL holds as the soft Bellman equation holds. By symmetry, the same is true for initial PCL.

Assuming trajectory PCL, for a trajectory $s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T$, we define $V(s_i)$ as

$$V(s_0) + \sum_{t=0}^{i-1} \gamma^t \left[\tau \log \pi(a_t|s_t) - R(s_t, a_t)\right]. \tag{31}$$

We first show that this definition is consistent in the sense that if there is another trajectory ${s'}_0, {a'}_0, \ldots, {s'}_{k-1}, {a'}_{k-1}, {s'}_k a'_k s_i$, then $V(s_i)$, defined in (31), is also equal to

$$V(s'_0) + \sum_{t=0}^{k} \gamma^t \left[\tau \log \pi(a'_t|s'_t) - R(s'_t, a'_t)\right]. \tag{32}$$

Since trajectory PCL holds, and

$$V(s'_0) + \sum_{t=0}^{k} \gamma^t \left[\tau \log \pi(a'_t|s'_t) - R(s'_t, a'_t)\right] + \sum_{t=i}^{T-1} \gamma^t \left[\tau \log \pi(a_t|s_t) - R(s_t, a_t)\right] \tag{33}$$

Table 4: Notation

Unnomralized distribution on terminal states	$\tilde{p}$
Normalizing factor	Z or $F(s_0)$
Normalized version of $\tilde{p}$	p
Probability simplex over the set $\mathcal{X}$	$\Delta(\mathcal{X})$
Calligraphic letters are used for sets	$\mathcal{X}, \mathcal{S}, \mathcal{A}$
Set of states	$\mathcal{S}$
Set of initial states	$\mathcal{S}_0 \subset \mathcal{S}$
Initial state (if unique, i.e., $\mathcal{S}_0 = \{s_0\}$)	s_0
Set of actions	$\mathcal{A}$
Function that returns a set	$\mathbb{A}, \mathbb{L}$
Action mask function	$\mathbb{A}$
Set of terminal states	$\mathcal{T} \subset \mathcal{S}$
Non terminal states	$\bar{\mathcal{T}}$
Power set	P
Deterministic transition function	T
Initial state distribution	$\mathbb{P}_0$
Set of trajectories leading to s from a unique starting state s_0	$\mathbb{L}(s)$
Number of trajectories leading to s from a unique starting state s_0	$n(s)$
$\log n(s)$	$l(s)$
Marginal distribution	μ
Discount factor	γ

Table 5: Abvreviations

Markov decision process	MDP
Generalized value function	GVF
Generative flow networks	GFN
Soft Q-learning	SQL
Flow matching	FM
Detailed balance	DB
Trajectory balance	TB
Path consistency learning	PCL

and

$$V(s_0) + \sum_{t=0}^{T-1} \gamma^t \left[\tau \log \pi(a_t|s_t) - R(s_t, a_t)\right] \tag{34}$$

are equal to $V(T)$ subtracting them yields zero. By subtracting (34) and (33) and removing the duplicate part, we are left with the two alternative definitions of $V(s_i)$, which means they must be equal.

Since all PCL holds for all initial trajectories, then PCL holds. □

Proposition 9. *Given n, maximum entropy GFNs with (TB) and GSQL with trajectory PCL have the same residual and gradient.*

Proof. The backward probability of any trajectory $s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T$ is $1/n(s_T)$ by Remark 2. Thus maxent GFNs have the objective $\log Z + \sum_{i=0}^{T-1} \log \pi(a_t|s_t) = \tilde{p}(s_T) - \log n(s_T)$ which is (PCL) for GSQL . □

C GENERATIVE FLOW NETWORKS

In this section, we review a few concepts related to GFN that did not fit in the main text due to to lack of space. While they are not necessary to understand this work, we belive that they can help understand GFNs better.

Table 6: n for MDPs with combinatorial structure. Bars indicate that the variable is fixed.

State space	Action space	$n(s)$
Words	append from right	1
Words of length N	append from either sides	2^{N-1}
DAGs with $\bar{N}$ nodes and E edges	connect two nodes	$E!$
Trees	add node	See (36)

Linearizing the term $F(s)\pi(a|s)$ of (DB) yields the flow action function $F(s,a)$ that has to fit the flow matching constraints (Bengio et al., 2021)

$$\tilde{p}(s) + \sum_{a\in\mathbb{A}(s)} F(s,a) = \sum_{s',a'\in\text{Parent}(s)} F(s',a'), \tag{FM}$$

for each state s where $F : \mathcal{S}\times\mathcal{A} \to \mathbb{R}_{\geq 0}$ is the state action flow function and $\tilde{p}$ is zero at nonterminal states. Any flow function F, the policy π, and the backward policy q that fit in (DB) or (FM) fit in $F(s,a) = F(s)\pi(a|s)$, $F(s,a) = F(s')q(s,a|s')$, $F(t) = \tilde{p}(t)$ and $F(s_0) = Z$ for a transition triplet $s' = T(s,a)$ and terminal state t.

Sub-trajectory balance (STB) (Madan et al., 2023) is the convex combination of (TB) over all sub-trajectories. The sub-TB constraint is defined as

$$F(s_i)\prod_{t=i}^{j-1}\pi(a_t|s_t) = F(s_j)\prod_{t=i}^{j-1} q(s_i, a_i|s_{t+1}), \tag{STB}$$

for all partial trajectory $s_i, a_i, \ldots, s_{j-1}, a_{j-1}, s_j$. When learning with (STB), (Madan et al., 2023) propose taking the weighted sum of the residual of all equations. They propose weighting the samples by γ to the length of the sub-trajectories as it corresponds to the expected loss of a random sub-trajectory length that is sampled from a geometric distribution with parameter γ (Shwartz, 2001). In this work, we assume $\gamma = 1$.

The constraints (FM), (DB), (STB), and (TB) are equivalent in the sense if one of them holds, all of them hold (Bengio et al., 2023; Malkin et al., 2022; Madan et al., 2023).

Multiple solutions for GFNs. Any convex combination of forward policies of GFNs leads to another forward policy of a GFN. Thus, if there are two valid solutions, there are infinite solutions. If two trajectories lead to the same node, then there are infinite solutions as the flow of that node can either flow in one or the other trajectory. An example is the MDP shown in Figure 1.

D CALCULATING $n(s)$OR MDPS WITH COMBINATORIAL STRUCTURES

Calculating $n(s)$ may be possible using the combinatorial structure of the state space. We give a small list in Table 6. The DAG MDP is based on the work of Deleu et al. (2022). The other MDPs are inspired by Bengio et al. (2021). These formulas are sensitive to the definition of the problem; for instance, calculating n for DAGs, where the actions add edges and nodes, is more complicated than where the nodes are fixed.

Similarly, assuming that the graph stays connected makes calculating n more complicated. For instance, we are unaware of any reasonable way that would not require traversing the MDP to calculate n for general connected graph-building environments. This is one of the reasons why we advocate learning n instead of calculating it.

The number of ways to define a tree represented as a directed graph rooted at a node r fits the following DP

$$W_r[s] = \frac{(D_r[s]-1)!}{\prod_{c\in\text{Children}(s)} D_r[c]!} \prod_{c\in\text{Children}(s)} W_r[c], \tag{35}$$

where $D_r[c]$ is the number of children of node c and fits $D_r[s] = 1 + \sum_{c\in\text{Children}} D_r[c]$. We obtain (35) by shuffling all actions that can be used to build the children (it holds since the actions used to build the children are independent in a tree). The total number of trajectories is

$$\sum_{s\in\text{Nodes}} W_s[s]. \tag{36}$$

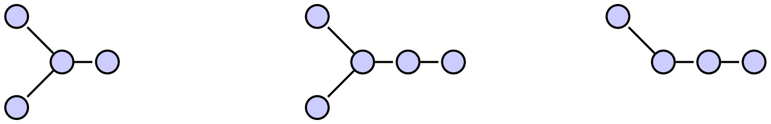


Figure 6: Proof that the uniform backward is not maximum entropy on tree-building environments. Left and right are the parents of the middle tree. Assuming nodes are unique, the left tree has 12 trajectories that reach it, while the right tree has 8.

We note that if attributes are on the edges, $D_r[c]$ is not the number of children but the number of children and attributes.

Lastly, we show in Figure 6 that for tree-building environments, similar to the drug design problem of Bengio et al. (2021), the uniform backward is not maximum entropy. The proposed state has two non-isomorphic parents whose n differ.

E EXPERIMENT DETAILS

The sEH experiments are tree-building environments where we assume that each state is a tree, each node in the tree is a fragment, a group of atoms, and each edge has a feature to describe how two fragments are connected (Jin et al., 2018). The fragments are those of Bengio et al. (2021) or Shen et al. (2023). The reward is Bengio et al. (2021)'s proxy for predicting the binding energy of a molecule to soluble epoxide hydrolase (sEH). We refer to Bengio et al. (2021) for a more detailed explanation.

The QM9 experiments are graph-building environments where we assume that each state is a connected graph (i.e., there is a path between each pair of nodes). Each node represents a carbon, nitrogen, fluorine, or oxygen atom. The main difference with the sEH experiments is that the state graph can have loops and terminal states may be invalid molecules. The target is Jain et al. (2023)'s proxy, a neural network predicting the HUMO-LUMO gap trained on the QM9 dataset (Ramakrishnan et al., 2014).

We transform the rewards into a scalar where zero is the worst quantity, and the reward is below one for most molecules. We divide the output of the proxy of Bengio et al. (2021) by 8. We linearly rescale the QM9 gaps such that the 5% molecules with the lowest gap in Ramakrishnan et al. (2014)'s dataset have a reward greater than one while the rest have a reward between 0 and 1. For the QM9, since the MDP can create invalid molecules, we give a reward of $\exp(-75)$ to invalid molecules.

Metrics. For the top K, diverse top K, and top K diversity, we sample N terminal states. We then calculate the average top K rewards, the average top K rewards under the constraint that the molecules have a similarity of less than 0.5 and one minus the average similarity of the top K molecules. If no K valid solutions exist, we assume the worst (i.e., the reward is zero or the molecule is one of the existing molecules for diversity). For molecules, we use the Tanimoto similarity of RDkit (Landrum et al., 2023).

Calculating (STB) and (PCL). We use dynamic programming to calculate these objectives in a performant manner. Algorithm 1 shows an implementation using functions available in PyTorch (Paszke et al., 2019). The main idea is that we can calculate the matrix $D_{ij} = \sum_{t=i}^{j} x_t$ by subtracting the vector $y_i = \sum_{t=0}^{i} x_t$ from its transposed shifted. Here v is the value and x_i is $\log \pi(a_i|s_i) - \log q(s_i, a_i|s_{i+1})$ for GFNs and $\log \pi(a_i|s_i)$ for SQL. We can avoid reversing the trajectory when learning n by using $x_i = -\log \pi(s_i, a_i|s_{i+1})$.

A practical algorithm A training loop roughly looks like the following

1. Obtain a batch of trajectories. The batch can either be sampled directly or through a more complex sampling scheme like the one proposed by Kim et al. (2023). The batch can also be obtained by obtaining a set of terminal states (potentially from a replay buffer) and using the backward policy to sample a backward trajectory like Zhang et al. (2022).

2. Minimize the residual of a constraint on n like PCL or the soft Bellman equation.

Algorithm 1 Psudocode for fast (STB) and (PCL)

```
def subtb(v, x):
    return torch.triu(v[:-1, None] - v[None, 1:] + cross(x))
def cross(x):
    y = torch.cumsum(x, 0)
    return y[None] - shift_right(y)[:, None]
def shift_right(x):
    x = torch.roll(x, 1, dims=0)
    x[0] = 0
    return x
```

Table 7: Hyperparameters used

Learning Rate	$5e-4$
Reward Exponent	96
Batch Size	256
ϵ uniform (exploration)	$1e-3$
Samples	$1e7$
EMA sampling model	0.95
Huber loss δ	0.25
Huber loss β	1

3. Minimize the residual of a GFN or GSQL constraint using PCL, TB, DB, or any other variant.

4. Update the sampling model.

For speed, we do steps 2 and 3 simultaneously by having a single neural network with two heads. We use the exponential moving average of the weights of the model as the sampling model

On the choice of the hyperparameters. We opt for (TB) and trajectory (PCL) to learn n and the policy. Since we use the Huber loss function (Huber, 1992) defined as

$$\text{Huber}(x;\delta) = \begin{cases} 0.5x^2/\delta & \text{if } |x| \leq \beta \\ \beta(|x| - 0.5\beta)/\delta & \text{otherwise,} \end{cases} \tag{37}$$

we add both losses together and do not need to worry about one loss contributing too much more to the loss. The main benefit of using the Huber loss is that the contribution of objectives to the gradient is more balanced than the mean squared error if the magnitude of the objectives is different, as the gradients are bounded. Using the Huber loss allows us to reduce the gradient clipping needed for stable training and improve the performance of all GFNs. We use the same hyperparameters for all models. Table 7 shows the hyperparameters used.

Code and reproducibility. The code is available at `https://github.com/recursionpharma/gflownet`

Quality of the solutions. Table 8 shows the Pearson correlation coefficient of different backward policies. We limit the models to 5 fragments from the 18 fragments of Shen et al. (2023) in the sEH experiment to be able to calculate the total probability of any terminal state. We sample molecules directly from $\tilde{p}$ and use them to calculate the Pearson correlation coefficient. It is worth noting how GFNs with free backward have a lower correlation coefficient than the rest when used with (STB).

Table 9 and Table 10 provide statistics for different parameters. These table have **not** been used for hyper parameter tuning.

Table 8: Pearson correlation coefficient of different backward policies. We sample terminal states with a random walk and then either sample them proportionately to the objective (proportional) or uniformly (uniform). *ST* and *T* mean sub-trajectory and trajectory level constraint.

model	n	loss	ϵ random	Person (proportional)	Person (uniform)
free		ST	0.001	0.82 ± 0.01	0.88 ± 0.01
			0.01	0.82 ± 0.01	0.87
		T	0.001	0.92	0.92
			0.01	0.92	0.91
uniform		ST	0.001	0.89	0.89
			0.01	0.90	0.90
		T	0.001	0.92	0.92
			0.01	0.93	0.91
maxent	known	ST	0.001	0.89	0.90
			0.01	0.89	0.90
		T	0.001	0.93	0.92
			0.01	0.93	0.92
	learned	ST	0.001	0.86	0.88
			0.01	0.85 ± 0.01	0.87
		T	0.001	0.92	0.91
			0.01	0.92	0.92
GSQL	known	ST	0.001	0.88	0.89
			0.01	0.87	0.88
		T	0.001	0.93	0.92
			0.01	0.93	0.91
	learned	ST	0.001	0.83	0.86
			0.01	0.83 ± 0.01	0.86
		T	0.001	0.93	0.92
			0.01	0.93	0.91

Table 9: Statistics for the sEH experiment.

model	π loss	n loss	ϵ random	entropy	diverse top K reward	top K diversity	top K reward	modes with $\tilde{p} \geq 1$	modes with $\tilde{p} \geq .875$
free	ST	none	0.001	72.39 ± 0.75	1.03	0.44	1.03	646.20 ± 11.29	4007.80 ± 31.68
			0.01	68.22 ± 0.61	1.02	0.44	1.02	565.00 ± 10.19	4114.20 ± 45.65
	T	none	0.001	77.92 ± 0.02	1.03	0.46	1.03	1049.90 ± 5.66	3378.00 ± 24.30
			0.01	77.61 ± 0.06	1.03	0.46	1.03	941.30 ± 6.43	3382.20 ± 55.95
uniform	ST	none	0.001	77.22 ± 0.03	1.03	0.44	1.03	668.80 ± 12.22	4059.40 ± 35.78
			0.01	77.22 ± 0.02	1.03	0.44	1.03	649.11 ± 9.53	4316.33 ± 50.56
	T	none	0.001	77.27 ± 0.02	1.03	0.46	1.03	1042.90 ± 5.21	3318.30 ± 31.15
			0.01	77.24 ± 0.02	1.03	0.46	1.03	971.10 ± 7.23	3324.70 ± 25.68
GSQL	ST	ST	0.001	78.59 ± 0.05	1.02	0.43	1.02	488.50 ± 7.00	4076.70 ± 30.72
			0.01	78.60 ± 0.07	1.02	0.43	1.02	507.50 ± 17.67	4305.50 ± 58.98
		known	0.001	78.46 ± 0.02	1.03	0.44	1.03	565.11 ± 10.36	4093.89 ± 26.39
			0.01	78.55 ± 0.03	1.02	0.43	1.02	486.50 ± 14.49	4151.70 ± 72.83
	T	ST	0.001	78.37 ± 0.01	1.03	0.46	1.03	1053.60 ± 4.74	3489.00 ± 30.93
			0.01	78.38 ± 0.02	1.03	0.46	1.03	972.00 ± 6.65	3388.90 ± 20.39
		T	0.001	78.42 ± 0.02	1.03	0.46	1.03	1043.90 ± 7.18	3446.50 ± 40.27
			0.01	78.38 ± 0.02	1.03	0.46	1.03	958.40 ± 7.94	3389.60 ± 46.82
		known	0.001	78.39 ± 0.01	1.03	0.46	1.03	1036.60 ± 7.74	3379.90 ± 29.55
			0.01	78.37 ± 0.02	1.03	0.46	1.03	931.11 ± 6.92	3293.44 ± 24.53
maxent	ST	ST	0.001	78.41 ± 0.02	1.03	0.44	1.03	554.70 ± 9.22	4134.80 ± 20.68
			0.01	78.48 ± 0.03	1.02	0.43	1.02	549.50 ± 7.45	4302.20 ± 34.57
		known	0.001	78.41 ± 0.02	1.03	0.44	1.03	656.50 ± 12.83	3993.30 ± 25.19
			0.01	78.47 ± 0.02	1.03	0.44	1.03	609.70 ± 9.72	4153.50 ± 35.74
	T	ST	0.001	78.42 ± 0.01	1.03	0.46	1.03	1048.50 ± 5.80	3388.60 ± 40.80
			0.01	78.40 ± 0.02	1.03	0.46	1.03	963.90 ± 6.12	3373.70 ± 26.76
		T	0.001	78.41 ± 0.03	1.03	0.46	1.03	1035.70 ± 5.56	3417.60 ± 32.82
			0.01	78.40 ± 0.01	1.03	0.46	1.03	951.60 ± 7.66	3354.00 ± 19.02
		known	0.001	78.41 ± 0.02	1.03	0.46	1.03	1042.50 ± 5.98	3344.00 ± 44.67
			0.01	78.41 ± 0.02	1.03	0.46	1.03	949.78 ± 6.22	3296.00 ± 35.13

Table 10: Statistics for the QM9 experiment.

model	π loss	n loss	ϵ random	entropy	diverse top K reward	top K diversity	top K reward	modes with $\tilde{p} \geq 1.125$	modes with $\tilde{p} \geq 1$
free	ST	none	0.001	89.72 ± 8.93	1.15 ± 0.01	0.80 ± 0.08	1.15 ± 0.01	$36\,941.80 \pm 2828.57$	100 000.00
			0.01	97.78 ± 0.41	1.13	0.89	1.13	$39\,792.20 \pm 1304.05$	100 000.00
	T	none	0.001	93.41 ± 0.24	1.20	0.83	1.20	$60\,465.30 \pm 2157.86$	100 000.00
			0.01	83.35 ± 8.04	1.21 ± 0.01	0.75 ± 0.08	1.21 ± 0.01	$62\,732.10 \pm 1845.51$	100 000.00
uniform	ST	none	0.001	96.82 ± 4.13	1.14 ± 0.01	0.88 ± 0.01	1.14 ± 0.01	$45\,371.30 \pm 2548.19$	100 000.00
			0.01	100.91 ± 0.04	1.14	0.89	1.14	$46\,464.80 \pm 615.89$	100 000.00
	T	none	0.001	89.96 ± 7.61	1.19 ± 0.01	0.81 ± 0.04	1.19 ± 0.01	$52\,865.10 \pm 4481.11$	100 000.00
			0.01	96.89 ± 0.05	1.20	0.82	1.20	$70\,884.20 \pm 824.47$	100 000.00
GSQL	ST	ST	0.001	102.58 ± 0.10	1.14	0.88	1.14	$46\,815.80 \pm 1470.56$	100 000.00
			0.01	102.67 ± 0.05	1.14	0.88	1.14	$45\,283.30 \pm 861.96$	100 000.00
	T	ST	0.001	100.29 ± 2.39	1.07 ± 0.11	0.85 ± 0.01	1.07 ± 0.11	$56\,199.90 \pm 3737.65$	100 000.00
			0.01	99.12 ± 0.29	1.19	0.83 ± 0.01	1.19	$54\,856.40 \pm 1866.60$	100 000.00
		T	0.001	98.59 ± 0.30	1.19	0.84	1.19	$63\,909.30 \pm 2730.65$	100 000.00
			0.01	89.18 ± 8.78	1.21 ± 0.01	0.74 ± 0.08	1.21 ± 0.01	$67\,928.20 \pm 2021.95$	100 000.00
maxent	ST	ST	0.001	102.61 ± 0.04	1.14	0.88	1.14	$47\,538.90 \pm 1432.97$	100 000.00
			0.01	102.48 ± 0.04	1.14	0.88	1.14	$46\,710.10 \pm 482.77$	100 000.00
	T	ST	0.001	98.49 ± 0.30	1.20	0.83 ± 0.01	1.20	$67\,809.20 \pm 2652.35$	100 000.00
			0.01	98.12 ± 0.09	1.20	0.82	1.20	$71\,374.50 \pm 1106.31$	100 000.00
		T	0.001	98.22 ± 0.11	1.20	0.83	1.20	$71\,339.40 \pm 1468.82$	100 000.00
			0.01	98.31 ± 0.11	1.20	0.83	1.20	$71\,962.60 \pm 1064.17$	100 000.00