NOTE:

- Use consistent mathematical notation throughout
- Include clear figures and diagrams
- Provide code snippets for key implementations
- Reference related work appropriately
- Writing style: maintain an academic tone while ensuring readability. Use precise technical language but explain complex concepts clearly. Include examples and visualizations to aid understanding.

NOTE:

Maybe there is not enough natural stochasticiy in the environment's reward-state transition dynamics to make this interesting. Maybe this could remidied by:

- Making the chain length random between episodes?
- Making the reward for reaching a terminal state random following a pre-determined distribution (by sampling the rewards from pre-determined distributions corresponding to each terminal state, we can still make assumptions about the desired behavior — bigger mean rewards ⇒ higher desired sample rates for trained model).

TODO:

Add title page.

TODO:

Add an abstract.

BSc Thesis

Valdemar H. Lorenzen

*Melih Kandemir

Abstract

Abstract goes here.

Contents

1 Introduction

Many real-world applications present a fundamental challenge that current reinforcement learning (RL) methods struggle to address effectively: the problem of delayed and sparse rewards.

> **Delayed and Sparse Rewards:** Learning scenarios where meaningful feedback signals (rewards) are provided only far after a long sequence of actions, and where most actions yeild no immediate feedback.
>
> *Example: In drug discovery, the effectiveness of a designed molecule can only be evaluated after its complete synthesis, with no intermediate feedback during the design process.*

Consider, for instance, the process of drug design, where a reinforcement learning agent must make a series of molecular modifications to create an effective compound. The value of these decisions — the drug's efficacy — can only be assessed once the entire molecule is complete. Similarly, in robotics tasks like assembly or navigation, success often depends on precise sequences of actions where feedback is only available upon having completed the entire task.

Traditional reinforcement learning algorithms face two critical limitations in such environments:

1. **Credit Assignment:** When rewards are delayed, the algorithm struggles to correctly attribute success or failure to specific actions in a long sequence. This is analogous to trying to improve a chess strategy when only knowing the game's outcome, without understanding which moves were actually decisive.
2. **Exploration Efficiency:** With sparse rewards, random exploration becomes highly inefficient. An agent might need to execute precisely the right sequence of actions to receive any feedback at all, making random exploration about as effective as searching for a needle in a haystack.

This thesis investigates a novel approach to addressing these challenges through the comparison of two promising methodologies: **Generative Flow Networks** (GFlowNets) [1] and **Bayesian Exploration Networks** (BEN) [2]. These approaches represent fundamentally different perspectives on handling uncertainty and exploration in reinforcement learning:

1. GFlowNets frame the learning process as a flow network, potentially offering more robust learning in situations with multiple viable solutions.
2. BENs leverages Bayesian uncertainty estimation to guide exploration more efficiently, potentially making better use of limited feedback.

By comparing these approaches, we aim to understand their relative strengths and limitations in environments with delayed and sparse rewards, ultimately contributing to the development of more efficient and practical reinforcement learning algorithms. Our investigation focuses specifically on examining these methods in carefully designed environments that capture the essential characteristics of delayed and sparse reward scenarios while remaining tractable for systematic analysis.

1.1 Research Objectives and Contributions

This thesis aims to advance our understanding of efficient learning in sparse reward environments through three primary objectives:

1. **Comparative Analysis:** Conduct a rigorous empirical comparison between GFlowNets and Bayesian Exploration Networks in standardized environments with delayed rewards.
2. **Hypothesis Testing:** Investigate whether BEN's Bayesian exploration strategy leads to more efficient learning compared to GFlowNets in highly delayed reward scenarios, particularly during early training stages.
3. **Algorithmic Understanding:** Analyze the underlying mechanisms that drive performance differences between these approaches, focusing on their handling of uncertainty and exploration.

The contributions of this work include:

- A comprehensive empirical evaluation using the n-chain environment with varying degrees of reward delay.
- Insights into the relative strengths and limitations of Bayesian and flow-based approaches to exploration.
- Implementation and analysis of both algorithms with comparisons.

1.2 Thesis and Structure

The remainder of this thesis is structured as follows:

> *NOTE:*
>
> Check that these titles correctly correspond to their reference.

Section 2: Preliminaries provides the theoretical foundations of reinforcement learning and explores existing approaches to handling sparse rewards. This chapter establishes the mathematical framework and notation used throughout the thesis.

Section 3: Theoretical Framework presents our hypothesis and analytical approach. We develop the mathematical foundations for comparing GFlowNets and BEN, with particular attention to their theoretical guarantees and limitations.

Section 4: Experimental Design details our testing methodology, including environment specifications, evaluation metrics, and implementation details. This chapter ensures reproducibility and clarity in our experimental approach.

Section 5: Results and Analysis presents our findings, including both quantitative performance metrics and qualitative analysis of learning behaviors. We examine how each algorithm handles the exploration-exploitation trade-off and adapts to varying levels of reward sparsity.

Section 6: Future Research ...

Section 7: Conclusion summarizes our findings, discusses their implications for the field, and suggests directions for future research.

2 Preliminaries

NOTE:

- Fundamentals of reinforcement learning
  - Markov Decision Processes
  - Q-learning and temporal difference methods
- Sparse reward challenges
- Survey of existing approaches
  - GFlowNets
  - Deep exploration networks (BEN)
  - Comparison of methodologies

2.1 Flow Networks

GFlowNets [1] rely on the concept of flow networks. A flow network is represented as a directed acyclic graph $G = (\mathcal{S}, \mathcal{A})$, where $\mathcal{S}$ represents the state space and $\mathcal{A}$ represents the action space.

> **Flow Network:** A directed acyclic graph with a single source node (initial state) and one or more sink nodes (terminal states), where flow is conserved at each intermediate node.
>
> *Example: In molecular design, states represent partial molecules and actions represent adding molecular fragments.*

2.1.1 States and Trajectories

We distinguish severel types of states:

- An initial state $s_0 \in \mathcal{S}$ (the source);
- Terminal states $x \in \mathcal{X} \subset \mathcal{S}$ (sinks);
- Intermediate states that form the pathways from source to sinks.

A trajectory τ represents a complete path through the network, starting at s_0 and ending at some terminal state x. Formally, we write a trajectory as an ordered sequence $\tau = (s_0 \to s_1 \to \ldots \to s_n = x)$, where each transition $(s_t \to s_{t+1})$ corresponds to an action in $\mathcal{A}$.

2.1.2 Flow Function and Conservation

The *trajectory flow function* $F : \mathcal{T} \to \mathbb{R}_{\geq 0}$ assigns a non-negative value to each possible trajectory. From this flow function, two important quantities are derived:

1. **State flow**: For any state s, its flow is the sum of flows through all trajectories passing through it:

$$F(s) = \sum_{s \in \tau} F(\tau). \tag{1}$$

2. **Edge flow**: For any action (edge) $s \to s'$, its flow is the sum of flows through all trajectories using that edge:

$$F(s \to s') = \sum_{\tau=(\ldots\to s\to s'\to\ldots)} F(\tau). \tag{2}$$

These flows must satisfy a conservation principle known as the *flow matching constraint*:

> **Flow Matching:** For any non-terminal state s, the total incoming flow must equal the total outgoing flow:
>
> $$F(s) = \sum_{(s''\to s)\in\mathcal{A}} F(s''\to s) = \sum_{(s\to s')\in\mathcal{A}} F(s\to s'). \tag{3}$$

2.1.3 Markovian Flow

The flow function induces a probability distribution over trajectories. Given a flow function F, we define $P(\tau) = \frac{1}{Z}F(\tau)$, where $Z = F(s_0) = \sum_{\tau\in\mathcal{T}} F(\tau)$ is the *partition function* [3] — i.e., the total flow through the network.

> **Markovian Flow:** A flow is *Markovian* when it can be factored into local decisions at each state. This occurs when the following exist [3]:
>
> 1. Forward policies $P_F(-|s)$ over children of each non-terminal state s.t.
>
> $$P(\tau = (s_0 \to \ldots \to s_n)) = \prod_{t=1}^{n} P_F(s_t|s_{t-1}). \tag{4}$$
>
> 2. Backward policies $P_B(-|s)$ over parents of each non-initial state s.t.
>
> $$P(\tau = (s_0 \to \ldots \to s_n)|s_n = x) = \prod_{t=1}^{n} P_B(s_{t-1}|s_t). \tag{5}$$

The Markovian property allows us to decompose complex trajectory distributions into simple local decisions, making learning tractable while maintaining the global flow constraints [CITATION NEEDED].

2.2 GFlowNets

GFlowNets [1] are an approach to learning policies that sample from desired probability distributions. They frame the learning process as discovering a flow function that makes the probability of generating any particular object proportional to its reward.

Given a reward function $R : \mathcal{X} \to \mathbb{R}_{\geq 0}$ defined over the set of terminal states $\mathcal{X}$, GFlowNets aim to approximate a Markovian flow F on the graph G s.t. $F(x) = R(x)$ for all $x \in \mathcal{X}$.

> **GFlowNet:** [3] defines a GFlowNet as any learning algorithm that discovers flow functions matching terminal state rewards, consisting of:
>
> 1. A model that outputs:
>   - Initial state flow $Z = F(s_0)$;
>   - Forward action distributions $P_F(-|s)$ for non-terminal states.
> 2. An objective function that, when globally minimized, guarantees $F(x) = R(x)$ for all terminal states.
>
> *Example: In molecular design, this ensures that high-reward molecules are genered more frequently, while maintaining diversity through exploration of multiple pathways.*

The power of GFlowNets lies in their ability to handle situations where multiple action sequences can lead to the same terminal state — a common scenario in real-world applications like molecular design or image synthesis. Unlike traditional RL methods that focus on finding a single optimal path, GFlowNets learn a distribution over all possible paths proportional to their rewards.

2.2.1 Learning Process

The learning process of GFlowNets involves iteratively improving both flow estimates and the policies. The forward policy of a GFlowNet can sample trajectories from the learned Markovian flow F by sequentially selecting actions according to $P_F(-|s)$. When the training converges to a global minimum of the objective function, this sampling process guarantees that $P(x) \propto R(x)$. That is, the probability of generating any terminal state x is proportional to its reward $R(x)$. This property makes GFlowNets particularly well-suited for:

1. **Diverse Candidate Generation:** Rather than converging to a single solution, GFlowNets maintain a distribution over solutions weighted by their rewards.
2. **Multi-Modal Exploration:** The flow-based approach naturally handles problems with multiple distinct solutions of similar quality.
3. **Compositional-Structure Learning:** By learning flows over sequences of actions, GFlowNets can capture and generalize compositional patterns in the solution space.

To achieve this, GFlowNets employ various training objectives, with *trajectory balance* [3] being one such particularly effective objective.

2.2.2 Trajectory Balance

Trajectory balance focuses on ensuring consistency across entire trajectories, instead of matching flows at every state (which can be computationally expensive).

> **Trajectory Balance:** A principle that ensures the probability of generating a trajectory matches its reward by maintaining consistency between forward generation and backward reconstruction probabilities.

Consider a Markovian flow F that induces a distribution P over trajectories according to $P(\tau) = \frac{1}{Z}F(\tau)$. The forward policy P_F and backward policy P_B must satisfy the following *trajectory balance constraint* [3]

$$Z\prod_{t=1}^{n} P_F(s_t|s_{t-1}) = F(x)\prod_{t=1}^{n} P_B(s_{t-1}|s_t). \tag{6}$$

That is to say, the probability of constructing a trajectory forward should match the probability of reconstructing it backward, scaled by the appropriate rewards.

2.2.3 Trajectory Balance as an Objective

To convert the trajectory balance function into a training objective, we introduce a parametrized model with parameters θ that outputs:

1. A forward policy $P_F(-|s;\theta)$;
2. A backward policy $P_B(-|s;\theta)$;
3. A scalar estimate Z_θ of the partition function.

For any complete trajectory $\tau = (s_0 \to \ldots \to s_n = x)$, we define the *trajectory balance loss* as

$$\mathcal{L}_{\text{TB}}(\tau) = \left(\log \frac{Z_\theta \prod_{t=1}^{n} P_F(s_t|s_{t-1};\theta)}{R(x)\prod_{t=1}^{n} P_B(s_{t-1}|s_t;\theta)}\right)^2. \tag{7}$$

This loss captures how well our model satisfies the trajectory balance constraint. When the loss approaches zero, our model has learned to generate samples proportional to their rewards. In practice, we compute this loss in the log domain to avoid numerical stability, as suggested by [3]:

$$\mathcal{L}_{\text{TB}}(\tau) = \left(\log Z_\theta + \log\sum_{t=1}^{n} P_F(s_t|s_{t-1};\theta) - \log R(x) - \log\sum_{t=1}^{n} P_B(s_{t-1}|s_t;\theta)\right)^2. \tag{8}$$

[3] also remarks that a simplificatoin of Equation 7 occurs in tree-structured state spaces (when G is a directed tree), where each state has exactly one parent. In such cases, the backward policy becomes deterministic ($P_B = 1$), reducing the loss function to

$$\mathcal{L}_{\text{TB}}(\tau) = \left(\log \frac{Z_\theta \prod_{t=1}^{n} P_F(s_t|s_{t-1};\theta)}{R(x)}\right)^2, \tag{9}$$

which can be exploited for the n-chain environment.

The model is trained by sampling trajectories from a training policy π_θ — typically a tempered version of $P_F(-|-;\theta)$ to encourage exploration — and updating parameters using stochastic gradient descent: $\theta \leftarrow \theta - \alpha\mathbb{E}_{\tau\sim\pi_\theta}\nabla_\theta\mathcal{L}_{\text{TB}}(\tau)$.

2.3 Contextual Reinforcement Learning

TODO:

Define a regular MDP

> **Contextual MDP:** A Markov Decision Process augmented with a context variable that determines the specific dynamics of the environment. This allows us to model uncertainty about the true environment through uncertainty about the context.

In a Contextual Markov Decision Process (CMDP), we work in an infinite-horizon, discounted setting where a context variable $\varphi \in \Phi \subseteq \mathbb{R}^d$ indexes specific MDPs. Formally, we describe this as

$$\mathcal{M}(\varphi) := \langle \mathcal{S}, \mathcal{A}, P_0, P_S(s, a, \varphi), P_R(s, a, \varphi), \gamma \rangle. \tag{10}$$

where the context φ parametrizes both:

- A transition distribution $P_S(s, a, \varphi)$ determining how states evolve;
- A reward distribution $P_R(s, a, \varphi)$ determining the rewards received.

The agent has complete knowledge of the following aspects of the environment:

- The state space $\mathcal{S} \subset \mathbb{R}^n$;
- The action space $\mathcal{A}$;
- The initial state distribution P_0;
- The discount factor γ.

However, the agent does not know the true context φ^* that determines the actual dynamics and rewards.

2.3.1 Policies and Histories

In contextual RL, an agent follows a *context-conditioned policy* $\pi : \mathcal{S} \times \Phi \to \mathcal{P}(\mathcal{A})$, selecting actions according to $a_t \sim \pi(s_t, \varphi)$. As the agent interacts with the environment, it accumulates a history of experiences $h_t := \{s_0, a_0, r_0, s_1, a_1, r_1, ..., a_{t-1}, r_{t-1}, s_t\}$. This history belongs to a state-action-reward product space $\mathcal{H}_t$ and follows a context-conditioned distribution $P_t^\pi(\varphi)$ with density

$$p_t^\pi(h_t|\varphi) = p_0(s_0) \prod_{i=0}^{t} \pi(a_i|s_i, \varphi) p(r_i, s_{i+1}|s_i, a_i, \varphi). \tag{11}$$

2.3.2 Optimization Objective

The agent's goal in a CMDP is to find a policy that optimizes the expected discounted return

$$J^\pi(\varphi) = \mathbb{E}_{\tau_\infty \sim P_\infty^\pi(\varphi)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]. \tag{12}$$

An optimal policy $\pi^*(\cdot, \varphi)$ belongs to the set $\Pi^*_\Phi(\varphi) := \arg\max_{\pi \in \Pi_\Phi} J^\pi(\varphi)$. With this, we define the optimal Q-function $Q^*(h_t, a_t, \varphi)$.

> **Optimal Q-Function:** For an optimal policy π^*, *the optimal Q-function* $Q^* : \mathcal{S} \times \mathcal{A} \times \Phi \to \mathbb{R}$ satisfies the Bellman equation
>
> $$\mathcal{B}^*[Q^*](s_t, a_t, \varphi) = Q^*(s_t, a_t, \varphi), \tag{13}$$
>
> where $\mathcal{B}^*$ is the optimal Bellman operator defined as
>
> $$\mathcal{B}^*[Q^*](s_t, a_t, \varphi) := \mathbb{E}_{r_t, s_{t+1} \sim P_{R,S}(s_t, a_t, \varphi)} \left[r_t + \max_{a' \in \mathcal{A}} Q^*(s_{t+1}, a', \varphi) \right]. \tag{14}$$

2.3.3 The Learning Challenge

When an agent has access to the true MDP $\mathcal{M}(\varphi^*)$, finding an optimal policy becomes a *planning problem*. However, in real-world scenarios, agents typically lack access to the true transition dynamics and reward functions. This transforms the task into a *learning problem*, where the agent must balance:

1. *Exploration*: learning about the environment's dynamics through interaction;
2. *Exploitation*: using current knowledge to maximize rewards.

This tension — known as the exploration/exploitation dilemma — remains one of the core challenges in reinforcement learning. As we'll see in the next section, Bayesian approaches offer a principled framework for addressing this challenge.

2.4 Bayesian Reinforcement Learning

In the Bayesian approach to RL, rather than viewing uncertainty as a problem to be eliminated, it becomes an integral part of the decision-making process — something to be reasoned about systematically.

> **Bayesian Epistemology:** A framework that characterizes uncertainty through probability distributions over possible worlds. In reinforcement learning, this means maintaining distributions over possible MDPs, updated as new evidence arrives.

2.4.1 From Prior to Posterior

The Bayesian learning process begins with a *prior distribution* P_Φ representing our initial beliefs about the true context φ^* before any observations. As the agent interacts with the environment, it accumulates a history of experiences h_t and updates these beliefs through Bayesian inference, forming a *posterior distribution* $P_\Phi(h_t)$.

This history-dependent posterior in Bayesian RL differentiates it from traditional RL approaches.

> **History-Conditioned Policies:** Unlike traditional RL policies that map states to actions, Bayesian policies operate on entire histories, defining a set of history-conditioned policies $\Pi_{\mathcal{H}} := \{\pi : \mathcal{H} \to \mathcal{P}(\mathcal{A})\}$, where $\mathcal{H} := \{\mathcal{H}_t | t \geq 0\}$ denotes the set of all histories.

Where the prior P_Φ represents our initial uncertainty (the special case where $h_t = \emptyset$), the posterior $P_\Phi(h_t)$ captures our refined beliefs after observing interactions with the environment. This allows us to reason about future outcomes by *marginalizing* across all possible MDPs according to our current uncertainty.

2.4.2 The Bayesian Perspective on Transitions

The power of the Bayesian approach stems from how it handles state transitions. Instead of committing to a single model of the environment, it maintains a distribution over possible transitions through the *Bayesian state-reward transition distribution*

$$P_{R,S}(h_t, a_t) := \mathbb{E}_{\varphi \sim P_\Phi(h_t)}\left[P_{R,S}(s_t, a_t, \varphi)\right]. \tag{15}$$

This distribution lets us reason about future trajectories using the *prior predictive distribution* P_t^π with density

$$p_t^\pi(h_t) = p_0(s_0) \prod_{i=0}^{t} \pi(a_i|h_i) p(r_i, s_{i+1}|h_i, a_i). \tag{16}$$

> **Belief Transitions:** The evolution of beliefs about the environment can itself be viewed as a transition system. This leads to the concept of a *Bayes-adaptive MDP* (BAMDP) [4].

The belief transition distribution $P_{\mathcal{H}}(h_t, a_t)$ captures how our beliefs evolve with new observations, with density

$$p_{\mathcal{H}}(h_{t+1}|h_t, a_t) = p(s_{t+1}, r_t|h_t, a_t). \tag{17}$$

This formulation leads to the definition of the Bayes-adaptive MDP:

$$\mathcal{M}_{\text{BAMDP}} := \langle \mathcal{H}, \mathcal{A}, P_0, P_{\mathcal{H}}(h, a), \gamma \rangle. \tag{18}$$

2.4.3 Natural Resolution of the Exploration Dilemma

An interesting aspects of the Bayesian framework is how it naturally resolves the exploration-exploitation dilemma. Rather than treating exploration as a separate mechanism, it emerges naturally from the optimization of expected returns under uncertainty

$$J_{\text{Bayes}}^\pi := \mathbb{E}_{h_\infty \sim P_\infty^\pi}\left[\sum_{i=0}^{\infty} \gamma^i r_i\right]. \tag{19}$$

A Bayes-optimal policy achieves perfect balance between exploration and exploitation because:

1. It accounts for uncertainty through the posterior at each timestep;
2. It considers how this uncertainty will evolve in the future;

3. It weights future information gain by the discount factor γ.

> **Conditionality Principle:** Bayesian decisions only condition on observed data, never on unknown quantities. This principle automatically prevents the pathological exploration-exploitation trade-offs that plague frequentist approaches.

NOTE:

Describing the conditionality principle would probably require describing frequentist reinforcement learning.

2.4.4 The Optimal Bayesian Q-Function

For a Bayes-optimal policy π^*, we can define the optimal Bayesian Q-function as $Q^*(h_t, a_t) := Q^{\pi^*_{\text{Bayes}}}(h_t, a_t)$. This Q-function satisfies the optimal Bayesian Bellman equation

$$Q^*(h_t, a_t) = \mathcal{B}^*[Q^*](h_t, a_t), \tag{20}$$

where $\mathcal{B}^*[Q^*]$ is the optimal Bayesian Bellman operator

$$\mathcal{B}^*[Q^*](h_t, a_t) := \mathbb{E}_{h_{t+1} \sim P_{\mathcal{H}}(h_t, a_t)} \left[r_t + \gamma \max_{a'} Q^*(h_{t+1}, a') \right]. \tag{21}$$

2.5 Bayesian Exploration Networks

TODO:

Probably explain model-free reinforcement learning vs model-based approaches.

NOTE:

- In model-free BRL, the goal is to characterise uncertainty in the optimal Bayesian Bellman operator instead of the reward-state transition distribution
- Given samples from the true reward-state distribution $r_t, s_{t+1} \sim P^*_{R,S}(s_t, a_t)$ we use *bootstrapping* to estimate the optimal Bayesian Bellman operator

$$b_t = \beta_\omega(h_{t+1}) := r_t + \gamma \max_{a'} Q_\omega(h_{t+1}, a') \tag{22}$$

  - We refer to $\beta_\omega(h_{t+1})$ as the bootstrap function
  - Interprete bootstrapping as making a change of variables under the mapping $\beta_{\omega(\cdot, h_t, a_t)} : \mathbb{R} \times \mathcal{S} \to \mathbb{R}$
- Bootstrapped samples b_t have distribution $P^*_B(h_t, a_t; \omega)$ which is the *pushforward* distribution over next period's possible updated Q-values satisfying

$$\mathbb{E}_{b_t \sim P_B^*(h_t, a_t; \omega)}[f(b_t)] = \mathbb{E}_{r_t, s_{t+1} \sim P_{R,S}^*(s_t, a_t)}\left[f\left(r_t + \gamma \max_{a'} Q_\omega(h_{t+1}, a')\right)\right] \tag{23}$$

(for any measurable function $f : \mathbb{R} \to \mathbb{R}$)

- Refer to $P_B^*(h_t, a_t; \omega)$ as the Bellman distribution

- When predicting b_t given an observation h_t, a:
  - Two sources of uncertainty:
    - Firstly, even if $P_B^*(h_t, a_t; \omega)$ is known, there's natural stochasticity due to the environment's reward-state transition dynamics that prevents b_t from being determined (*aleatoric uncertainty*)
      - Aleatoric uncertainty *cannot* be reduced with more data
    - Secondly, in a learning problem, the Bellman distribution $P_B^*(h_t, a_t; \omega)$ cannot be determined a priori and must be inferred from observations of b_t (*epistemic uncertainty*)
      - Epistemic uncertainty *can* be reduced with more data as the agent explores
- We introduce a model of the process $b_t \sim P_B^*(h_t, a_t; \omega)$ which characterises the aleatoric uncertainty in the optimal Bellman operator
- [2] choose a parametric model $P_B(h_t, a_t, \varphi; \omega)$
  - density $p(b_t | h_t, a_t, \varphi; \omega)$
  - parametrised by $\varphi \in \Phi$
- The space of models $P_B(h_t, a_t, \varphi; \omega)$ can be interpreted as a hypothesis space over the true Bellman distribution $P_B^*(h_t, a_t; \omega)$, with each hypothesis indexed by a parameter $\varphi \in \Phi$
- $\mathcal{D}(h_t) := \{(b_i, h_i, a_i)\}_{i=0}^{t-1}$ denotes the dataset of bootstrapped samples
  - the agent updates its belief in φ by inferring a posterior $P_\Phi(\mathcal{D}_\omega(h_t))$ when it has observed $\mathcal{D}_\omega(h_t)$
  - This posterior $(P_\Phi(\mathcal{D}_\omega(h_t)))$ characterises the epistemic uncertainty over the hypothesis space, which is used to obtain the predictive optimal Bellman distribution:

$$P_B(h_t, a_t; \omega) = \mathbb{E}_{\varphi \sim P_\Phi(\mathcal{D}_\omega(h_t))}[P_B(h_t, a_t, \varphi; \omega)] \tag{24}$$

  - Taking expectations over the variable b_t using $P_B(h_t, a_t; \omega)$, the predictive optimal Bellman operator is derived:

$$B^+[Q_\omega](h_t, a_t) := \mathbb{E}_{b_t \sim P_B(h_t, a_t; \omega)}[b_t], \tag{25}$$

    which integrates both the aleatoric epistemic uncertainty in b_t to make a Bayesian prediction of the optimal Bellman operator at each timestep t

Bayesian Exploration Networks (BENs) represent an approach to model-free Bayesian RL that addresses the challenge of efficient exploration under uncertainty by incorporating both aleatoric and epistemic uncertainty in the model. The innovation of BENs lies in their three-component architecture that separates different types of uncertainty:

1. A *recurrent Q-network* that approximates Q-values while maintaining a history of past interactions;
2. An *aleatoric network* that models inherent randomness in the environment;
3. An *epistemic network* that captures uncertainty in our knowledge of the environment.

This three-component architecture allows BEN to:
1. Maintain a history-dependent view of the environment;
2. Model both inherent randomness and knowledge uncertainty separately;
3. Learn Bayes-optimal policies through principled exploration.

We will examine each component in detail.

2.5.1 Recurrent Q-Network

TODO:

- If mentioning QBRL, explain what it is.

At its core, BEN uses a recurrent neural network (RNN) to approximate the optimal Bayesian Q-function. Unlike approaches based on QBRL that only consider the current state (and a context variable) [2], BEN's Q-network processes the entire history of interactions. We denote the output at timestep t as $q_t = Q_\omega(h_t, a_t) = Q_\omega\left(\hat{h}_{t-1}, o_t\right)$, where h_t represents the history up to time t, a_t is the action, $\hat{h}_{t-1}$ is the recurrent encoding of previous history, and o_t contains the current observation tuple $\{r_{t-1}, s_t, a_t\}$.

By conditioning on history rather than just current state, BEN can capture how uncertainty evolves over time, making it capable of learning Bayes-optimal policies.

2.5.2 Aleatoric Network

The aleatoric network models inherent randomness in the environment's behavior — what we might call "known uncertainty." It uses normalizing flows to transform a simple base distribution (such as a standard Gaussian) into a more complex distribution $P_B(h_t, a_t, \varphi; \omega)$, over possible next-state Q-values by applying the transformation $b_t = B(z_{\text{al}}, q_t, \varphi)$, where $z_{\text{al}} \in \mathbb{R} \sim P_{\text{al}}$ is a base variable with a zero-mean, unit variance Gaussian P_{al}, q_t is the Q-value from the recurrent network, and φ and ω represent the network parameters.

> **Aleatoric Uncertainty:** The unpredictability inherent in the environment, even with perfect knowledge of its dynamics. Like rolling a fair die — we know the probabilities perfectly, but can't predict individual outcomes.

2.5.3 Epistemic Network

TODO:

- Explain variational inference?

The epistemic network captures our uncertainty about the environment itself — what we might call "unknown uncertainty." This layer uses normalizing flows for variational inference to learn a tractable approximation P_ψ of the potentially complex target distribution $P_B(h_t, a_t, \varphi; \omega)$ parametrised by $\psi \in \Psi$. We learn ψ by minimizing the KL-divergence between the two distributions $\mathrm{KL}\big(P_\psi \parallel P_\Phi(\mathcal{D}_\omega(h_t))\big)$, which is equivalent to minimising the tractable evidence lower bound $\mathrm{ELBO}(\psi; h, \omega)$ [2]. This flow P_ψ, representing the epistemic uncertainty, characterises the uncertainty in φ.

> **Epistemic Uncertainty:** Uncertainty about the true nature of the environment, which can be reduced through observation and learning. Like uncertainty about whether a die is fair — this can be resolved through repeated observations.

2.5.4 Learning Process

The network is trained by minimizing two objectives:

- The Mean Squared Bayesian Bellman Error (MSBBE) for the Q-network and the aleatoric network;
- The Evidence Lower Bound (ELBO) for the epistemic network.

This dual optimization process ensures that the network learns both optimal value estimation and appropriate uncertainty quantification.

2.5.5 MSBBE as an Objective

TODO:

We use the predictive optimal Bellman operator, but we don't define it
- This would fit into a preliminary section on model-free BRL

The MSBBE is computed as the difference between the predictive optimal Bellman operator $B^+[Q_\omega]$ and Q_ω:

$$\mathrm{MSBBE}(\omega; h_t, \psi) := \parallel B^+[Q_\omega](h_t, a_t) - Q_\omega(h_t, a_t) \parallel_\rho^2, \tag{26}$$

which is minimized to learn the parametrisation ω^*, satisfying the optimal Bayesian Bellman equation for our Q-function approximator, with ρ being an arbitrary sampling distribution with support over $\mathcal{A}$.

This gives rise to a nested optimisation problem, as is common in model-free RL [2], which can be solved using two-timescale stochastic approximation. In this case, we update the epistemic network parameters ψ using gradient descent on an asymptotically faster timescale than the function approximator parameters ω to ensure convergence to a fixed point [2].

2.5.6 ELBO as an Objective

The Evidence Lower Bound (ELBO) serves as the optimization objective for training BEN's epistemic network. While minimizing the KL-divergence $\mathrm{KL}\big(P_\psi \parallel P_\Phi(\mathcal{D}_\omega(h_t))\big)$ directly would give us the most accurate approximation of the true posterior, computing this divergence is typically intractable. Instead, we can derive and optimize the ELBO, which provides a tractable lower bound on the model evidence.

Starting with the definition of the KL-divergence and applying Bayes' rule, [2] derives

$$\begin{aligned}&\mathrm{ELBO}(\psi; h_t, \omega)\\ &\quad := \mathbb{E}_{z_{\text{ep}} \sim P_{\text{ep}}}\left[\sum_{i=0}^{t-1}\Big(B^{-1}(b_i, q_i, \varphi)^2 - \log|\ \partial_b B^{-1}(b_i, q_i, \varphi)\ |\Big) - \log p_\Phi(\varphi)\right],\end{aligned} \tag{27}$$

where $\varphi = t_\psi\big(z_{\text{ep}}\big)$ and:

- z_{ep} is drawn from the base distribution P_{ep} (a standard Gaussian $\mathcal{N}\big(0, I^d\big)$);
- B^{-1} is the inverse of the aleatoric network's transformation;
- $\partial_b B^{-1}$ is the Jacobian of this inverse transformation;
- t_ψ represents the epistemic network's transformation.

> **Jacobian Term:** The term $\partial_b B^{-1}$ accounts for how the epistemic network's transformation changes the volume of probability space. This is important for maintaining proper probability distributions when using normalizing flows.

The ELBO objective breaks down into three key components:

1. A reconstruction term $B^{-1}(b_i, q_i, \varphi)^2$ that measures how well our model can explain the observed Q-values;
2. A volume correction term $\log|\partial_b B^{-1}(b_i, q_i, \varphi)|$ that accounts for the change in probability space;
3. A prior regularization term $\log p_\Phi(\varphi)$ that encourages the approximated posterior to stay close to our prior beliefs.

By minimizing the ELBO, we obtain an approximate posterior that balances accuracy with computational tractability, allowing BEN to maintain and update its uncertainty estimates efficiently during learning.

3 Theoretical Framework

NOTE:

- Hypothesis development
- Problem formulation
  - Mathematical notation and definitions

  - Assumptions and constraints
- Proposed solution approach

4 Experimental Design

NOTE:

- Test environments
  - N-Chain implementation
- Evaluation metrics
  - Sample efficiency (steps needed to reach optimal policy) - measure by objective loss over training?
  - Final performance (average success/reward rate) - measure by difference between sample distribution and true distributions? Otherwise, this probably doesn't make sense for the semi-deterministic n-chain environment, as we know we'll succeed in n steps.
  - Exploration behavior (state coverage over time) - maybe not so interesting for the simple n-chain environment, but not entirely uninteresting either.
- Implementation details
  - Network architectures
  - Training procedures
    - For GFlowNets, mention tempered exploration during training (off-policy training)
  - Hyperparameter selection

4.1 Test Environment

TODO:

- Describe the n-chain environment:
  - a chain of length n
  - the chain has a branch point from which two branches emerge
  - last state is always a terminal state
  - terminal states have a disignated reward
- Describe the action space:
  - actions include a *forward* action;
  - a *left branch* or *right branch* choice at branch point.
-

  NOTE:

The *terminal stay* action is probably inconsequensial.

4.2 Evaluation Metrics

TODO:

- Describe how the sample efficiency is measured.
  - Maybe the objective loss over training (how many steps to convergence)
- Describe the final performance:
  - In my n-chain implementation, a terminal state will always be reached, so average success rate wouldn't make sense as a metric (as success will always be 100%).
  - Instead, for GFlowNets, this should probably be evaluated as the difference from the true distribution (the expected distribution).
  - I.e., with rewards 10 and 5 for two terminal states, the true distribution would be $\frac{1}{3}$ samples with reward 5 and $\frac{2}{3}$ samples with reward 10.
- Describe exploration behavior:
  - State coverage over time.
  - This will probably require a higher number of branches for any interesting metrics.

4.3 Implementation Details

TODO:

- Network architectures
  - GFlowNets:
    - Multi-layer perceptron for state encoding
      - Input: State tensor of shape [batch_size, state_dim]
      - Output: State encoding of of shape [batch_size, hidden_dim]
    - Single layer for forward policy
      - Input: State encoding of shape [batch_size, hidden_dim]
      - Output: Forward policy [batch_size, num_actions]
    - Single layer for backward policy
      - Input: State encoding of shape [batch_size, hidden_dim]
      - Output: Backward policy [batch_size, num_actions]
        - Note: for the n-chain environment, the graph is a directed tree and so each state can have at most one parent, meaning that only one action is possible for each state in the backward policy, as only one action could have lead from the previous state to this state (as actions are represented by edges)
    - Parameter (scalar) for log Z function approximation

- Training procedures
  - For GFlowNets, mention tempered exploration during training (i.e., off-policy training)
    - Mention epsilon parameter for temperature control in guided exploration
- Hyperparameter selection

5 Results and Analysis

NOTE:

- Quantitative results
  - Performance comparisons
  - Statistical analysis
- Qualitative analysis
  - Exploration patterns
  - Learning behavior
- Discussion of findings

6 Future Research

NOTE:

Everything I think I could have done better essentially.

7 Conclusion

NOTE:

- Summary of contributions
- Key insights
- Future work directions

Bibliography

[1] E. Bengio, M. Jain, M. Korablyov, D. Precup, and Y. Bengio, "Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation." [Online]. Available: https://arxiv.org/abs/2106.04399

[2] M. Fellows, B. Kaplowitz, C. S. de Witt, and S. Whiteson, "Bayesian Exploration Networks." [Online]. Available: https://arxiv.org/abs/2308.13049

[3] N. Malkin, M. Jain, E. Bengio, C. Sun, and Y. Bengio, "Trajectory balance: Improved credit assignment in GFlowNets." [Online]. Available: https://arxiv.org/abs/2201.13259

[4] M. O. Duff and A. G. Barto, "Optimal learning: computational procedures for bayes-adaptive markov decision processes," 2002. [Online]. Available: https://api.semanticscholar.org/CorpusID:118819159