Quantum online planning for POMDPs with Bayesian Networks

Gilberto Cunha[1,3,4] and Luís S. Barbosa[1,2,3,5]

[1]Universidade do Minho, [2]International Iberian Nanotechnology Laboratory, [3]INESC TEC,
[4]grncacad@gmail.com, [5]lsb@di.uminho.pt

September 5, 2022

1 Introduction

One of the main goals of quantum computing research is the discovery of possible advantages over classical computing techniques for relevant applications. Proven advantages have been presented for problems in a wide range of fields, such as cryptography [PAB+20], chemistry [CRO+19] and machine learning [HKP21, HBC+21, BWP+17]. Specific to the fields of artificial intelligence (AI) and decision theory, a large set of contributions have driven and fostered research in the overlap with quantum computation. New insights in game theory have been demonstrated by discovering Nash equilibria related to unique quantum strategies in [Mey99]. Furthermore, a quantum advantage for static decision-making problems using Bayesian inference has been proven in [dOB21], via the application of utility functions with quantum operators in an entangled configuration. Also, in social sciences, accurate predictions on human decision-making have been verified by introducing quantum probabilities, using quantum-like Bayesian networks [MW16] as the underlying probabilistic model.

Although some static decision-making processes achieve satisfying results, Reinforcement Learning (RL) agents excel in learning within a set of more complex decision-making domains, such as possibly ever-changing and even partially observable environments. Following its success in other fields, RL has also been targeted by the scientific community as a potential candidate for quantum computing applications and yielded seemingly promising results for the field [DTB17, JTN+21]. In [PDM+14], the authors find that using quantum walks in projective simulation agents yields a quadratic speedup over classical methods. Moreover, as experimentally shown in [SAH+21], RL agents' learning time can be mitigated when interacting with the environment through both classical and quantum communication channels. Perhaps seemingly unrelated at first glance, inference on sparse Bayesian Networks using quantum rejection sampling has been proven to yield a quadratic advantage [LYC14] over classical rejection sampling. Fortunately, RL environments, which are formally described using Markov decision processes (MDPs), can be translated into dynamic Bayesian Networks (DBNs), enabling the quantum Bayesian inference speedup to be adapted to the field of RL.

Founded on the computational separation between the classical and the quantum Bayesian inference algorithms, this work presents a quantum-classical online planning algorithm for model-based RL agents in partially observable environments. The environment is considered to be in the infinite horizon setting, while the planning is performed using a lookahead method over a finite horizon H. The proposed algorithm holds a computational advantage over the classical alternative, which is proven analytically via the computational complexity. The proof entails a detailed analysis of the sample complexity of the algorithm by guaranteeing an ϵ-approximate action value function with a $1-\delta$ confidence interval, as well as a per-sample complexity comparison between the proposed method and its classical alternative.

2 Reinforcement Learning

2.1 An introduction to partially observable Reinforcement Learning problems

Reinforcement Learning (RL) is a sub-field of artificial intelligence concerned with the problem of rational decision making, which achieves this goal by rewarding good decisions and punishing bad ones. In this formulation, there are two main objects of study that interact with each other:

- *Environment*: Representation of a world with different possible states. Agents are always in some state of the environment and transition between them in a probabilistic manner according to their actions.
- *Agent*: Represents a decision-maker that lives in a world represented by the aforementioned *environment* and wants to achieve a certain goal. Its purpose is to take decisions in that *environment* in order to reach that goal.

Each interaction between these two objects involves the following steps (see figure 1):

1. The *agent*, while in the current state "state_t" of the *environment*, takes an action "action_t"
2. As a reaction to action "action_t", the *environment* changes to state "state_{t+1}"
3. In the new state "state_{t+1}", the *environment* sends the *agent* an observation "observation_{t+1}" with information about state "state_{t+1}" and a *reward* "reward_{t+1}" that evaluates the decision made by the *agent* in step 1

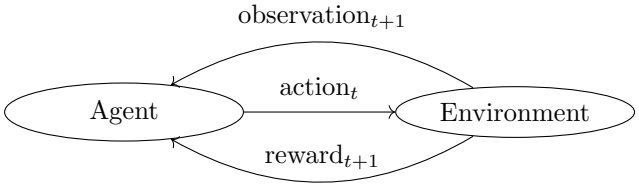


Figure 1: Interaction at time-step t between the *agent* and the *environment*

Given the observation sent to the *agent*, two types of *environments* can be distinguished. If that observation completely defines the state of the *environment* (hence the *agent* always knows in which state it is in), the *environment* is said to be *fully-observable*. However, if that observation only contains partial information about the state, meaning the *agent* can't be certain about its current state, then the *environment* is said to be *partially observable*. In the partially observable case, there is a an added layer of uncertainty the *agent* must deal with, an therefore it is modelled in a different way from a *fully observable environment*.

Since this work concerns itself with partially observable environments, it is useful to give them a formal definition. These types of environments are usually modelled with *partially observable Markov decision processes* (POMDPs), which are a tuple [Lit09] $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \Omega, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \gamma \rangle$ where:

- $\mathcal{S}$ represents the set of states of the environment, $\mathcal{A}$ represents a set of actions the agent can take and Ω represents the set of possible observations the agent can receive
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is a probabilistic transition function:

$$\begin{aligned} &\mathcal{P}^a_{ss'} = P\left(S_{t+1} = s' | S_t = s, A_t = a\right) \\ &\text{where} \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} = 1, \forall s \in \mathcal{S},\, a \in \mathcal{A} \end{aligned} \tag{1}$$

- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ is a reward function:

$$\mathcal{R}^a_{ss'} = \mathbb{E}\left(R_{t+1} | S_{t+1} = s', S_t = s, A_t = a\right) \tag{2}$$

- $\mathcal{Z} : \Omega \times \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ is the sensor model, representing conditional distributions of observations given states and actions:

$$\mathcal{Z}^o_{sa} = P\left(O_{t+1} = o | S_{t+1} = s, A_t = a\right) \tag{3}$$

- $\gamma \in [0, 1]$ is a discount factor to define a bounded return G_t:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Solving a *Reinforcement Learning* problem entails the need to teach the *agent* how to behave in its *environment* in order to maximize its expected return, the so called *maximum expected utility* (MEU) principle [RN09]. It is a complex decision-making problem that involves making good sequences of decisions in a possibly changing *environment*.

Solving a POMDP comes with the added challenge of the agent not knowing in what state it is while taking an action. Nevertheless, while the agent might not know exactly in which state it is, it can have some *beliefs* about in what states it could be. For example, in an environment with two states it could believe that it is 40% likely to be in state 1 and 60% likely to be in state 2. These beliefs can be formally defined as a probability distribution over states that represent the likelihood of being in a certain state for the agent, the *belief states* $b(s)$:

$$b(s) = P(S_t = s) \tag{4}$$

As the decision making process goes on, the agent will be receiving more and more observations about the environment. These observations are important for the agent to update its beliefs in light of this new information and in light of its past actions. Therefore, the definition of a belief state in equation 4 is somewhat incomplete, because an agent's beliefs should depend on the history of observations $o_{1:t}$ and actions $a_{1:t-1}$ taken so far:

$$b(s) = P\left(S_t = s | O_{1:t} = o_{1:t}, A_{1:t-1} = a_{1:t-1}\right) \tag{5}$$

Belief states allow the agent to weight its decisions according to the likelihood of being in each of the possible states instead of considering only its current state. Due to the Markov Property, belief-states in one time-step can be calculated from the belief states in the previous step in a recursive fashion [RN09]. Consider that, at time-step t, an agent with a belief state $b(s)$ takes an action a, leading it to a new unknown state of the POMDP, which sends him an observation o. The agent can update its belief state $b(s)$ to a new belief state $b'(s')$ using the belief-update rule $\tau(b, a, o)$ [Lit09]:

$$\tau\left(b, a, o\right)\left(s'\right) \equiv b'(s') \propto \mathcal{Z}^{o}_{s'a} \sum_{s \in \mathcal{S}} \mathcal{P}^{a}_{ss'} b(s) \tag{6}$$

This surprising results helps tremendously in the calculation of the probabilities of equation 5. It states that the only thing needed for an agent to always know a belief state is to define it at the beginning of the decision making process. With an initial belief state defined, the agent can therefore take the best actions according to that belief state and use the new observations from the environment to update those beliefs.

Finally, to look at a possible solution to partially observable RL problems, one important notion is that of a *value function*. Value functions describe the value of a certain scenario to a RL agent, and generally there are two types of value functions:

- A state value function $V(b_t)$ that represents the value of belief-state b_t to the agent. It has a recursive definition given by the following equation:

$$V(b_t) = \max_{a_t \in \mathcal{A}} \left(\mathbb{E}\left(R_{t+1} | b_t, a_t\right) + \gamma \sum_{o_{t+1} \in \Omega} P\left(o_{t+1} | b_t, a_t\right) V\left(\tau(b_t, a_t, o_{t+1})\right) \right) \tag{7}$$

- A state-action value function $Q(b_t, a_t)$ that represents the value of a belief-state b_t and action a_t pair to the agent. It can be defined recursively using the following equation:

$$Q(b_t, a_t) = \mathbb{E}\left(R_{t+1} | b_t, a_t\right) + \gamma \sum_{o_{t+1} \in \Omega} P\left(o_{t+1} | b_t, a_t\right) \max_{a_{t+1} \in \mathcal{A}} Q\left(\tau(b_t, a_t, o_{t+1}), a_{t+1}\right) \tag{8}$$

Notice that the two definitions in equations 7 and 8 allow the value functions to be defined in terms of each other:

$$V(b_t) = \max_{a_t \in \mathcal{A}} Q(b_t, a_t) \tag{9}$$

$$Q(b_t, a_t) = \mathbb{E}\left(R_{t+1} | b_t, a_t\right) + \gamma \sum_{o_{t+1} \in \Omega} P\left(o_{t+1} | b_t, a_t\right) V\left(\tau(b_t, a_t, o_{t+1})\right) \tag{10}$$

2.2 Lookahead algorithm for partially observable problems

The introduction of subsection 2.1 gives enough technical background to look at one possible solution to partially observable RL problems: an approximate *lookahead* solution method.

This algorithm entails building a lookahead tree as in figure 2: starting with the current belief-state as the root (belief) node of the tree, enumerate every possible action the agent can take to create the child (observation) nodes.

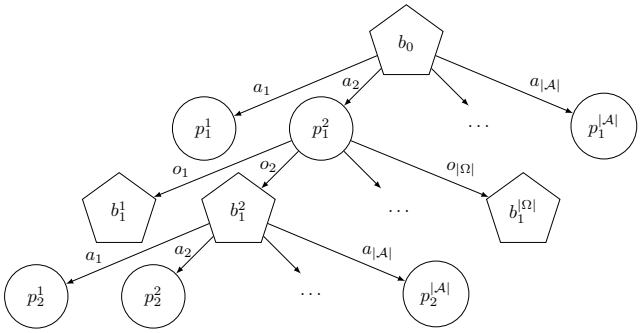


Figure 2: A two-step ($H = 2$) lookahead tree. Belief nodes are pentagon-ally shaped and observation nodes are circle shaped. For diagram simplicity, only one of the observation nodes and one of its belief nodes is expanded.

Then, for every observation node, enumerate all possible observations the agent can receive as a response to create this node's child (belief) nodes [1]. This procedure is repeated until the desired horizon (or depth) H of the tree is reached.

Using this tree, the goal is to calculate the Q-values for each action that can be taken from the root node, and to choose the action with the highest Q-value. To do this, the following steps should be followed:

1. Calculate the expected reward at every leaf (observation) node of the tree. Assume these are the Q-values for these nodes.
2. Using equation 9, use the previously calculated Q-values to calculate the V-values of its parent belief nodes.
3. Using equation 10, use the previously calculated V-values to calculate the Q values of its parent observation nodes.
4. Repeat steps 2 and 3 until the root belief node is reached.
5. Select the action with the highest Q-value.

Using these steps, a near-optimal action can be extracted from the lookahead tree to approximately solve partially observable RL algorithms. The only matter left to solve is how to extract the probability distributions in equations 9 and 10. As the next section details, in this work these probability distributions are approximated using *Bayesian Networks*, which can be used to model POMDPs.

3 Bayesian Networks

3.1 Introduction to Bayesian Networks

A *Bayesian Network* (BN) is a probabilistic graphical model encoding knowledge over a set of variables X whose values are unknown (denoted as *random variables*). Random variables can take *discrete* or *continuous* values, and are be connected by an edge if there is a probabilistic dependency between them. Moreover, there can not be any cyclic dependency between them. In this work, random variables are denoted by a capital letter, while their values are denoted by a lower case one (e.g., random variable X can take value x).

As such, a BN is a *directed acyclic graph* (DAG), where each node X_i contains a *conditional probability table* (CPT) of its values conditioned on its parent random variable values (see fig. 3), mathematically written as $P(X_i|parents(X_i))$.

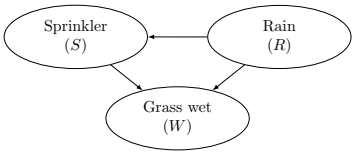
This representation of random variables and their dependencies allows to compactly represent *joint probability distributions* (JPDs) [RN09], which can be written as follows:

$$P(x_1, x_2, \ldots, x_N) = \prod_{i=1}^{N} P(x_i|parents(X_i)) \tag{11}$$

Furthermore, as discussed in subsection 3.2, it is possible to infer any probability distribution $P(\mathcal{Q}|\mathcal{E} = e)$ from a *Bayesian Network*, where $\mathcal{Q}$ is a set of *query variables*, $\mathcal{E}$ is a set of evidence variables and e is denoted as the *evidence*.

These characteristics make *Bayesian Networks* a very useful tool with applications in various scientific domains, ranging from performance evaluation [ZD03], to health monitoring [KHV06] or even aircraft engineering [LML+17].

[1] Recall from the belief-update rule of equation 6 that a belief-state, action and observation can be used to update a belief-state. This belief-update should be performed at every child belief node to calculate its respective belief-state.



R	S	W	$P(W\|S,R)$
False	False	False	0.6
False	False	True	0.4
False	True	False	0.1
False	True	True	0.9
True	False	False	0.01
True	False	True	0.99
True	True	False	0.01
True	True	True	0.99

R	S	$P(S\|R)$
False	False	0.6
False	True	0.4
True	False	0.99
True	True	0.01

R	$P(R)$
False	0.9
True	0.1

Figure 3: A Bayesian Network over three binary random variables and their CPTs.

Even more relevant to this work, as inferring a conditional probability distribution from a BN can be viewed as an application of *Bayes' Theorem*, BNs will be useful both for modelling environments and agents' decisions under uncertain situations.

3.2 Rejection Sampling

Once a Bayesian Network has been constructed, some sort of algorithm is still needed to infer a *posterior* probability distribution $P(\mathcal{Q}|\mathcal{E}=e)$ from a given evidence e, since these probability distributions can not always be directly extracted from the BN. This extraction is referred in the literature as *inference*, and algorithms that perform it are called *inference algorithms*.

Inference algorithms can be grouped into two big categories: *exact inference* and *approximate inference* algorithms. The first kind can calculate the exact posterior probabilities at a high computational cost, while the second only obtains some approximation value with the advantage of being computationally more tractable [MR02].

Given that exact inference algorithms are generally too computationally expensive, seeking approximate methods is essential for these networks to be applicable to a wider variety of applications. The presented approximate inference algorithms rely on *randomly sampling* (also denoted *Monte Carlo* algorithms) from a known probability distribution modelled by the *Bayesian Network*.

The first algorithm, named *direct sampling*, can be used to extract a sample from the JPD encoded by the *Bayesian Network*. If applied repeatedly while collecting all samples, an approximation of this JPD can be obtained.

Getting a sample from the network involves iterating over its random variables in topological order [2]. For the root nodes, a value is sampled according to the probabilities of each values' occurrence. Those values are then propagated to their children, so that a value for those child nodes can be sampled according to the values previously sampled from their parents. This procedure is described in algorithm 1.

Algorithm 1 Direct sampling algorithm. Returns a sample from the BN's joint probability distribution.

Require:
$\mathcal{B}$, a BN representing the distribution $P(X_1, X_2, \ldots, X_N)$ with N nodes in topological order.

$i \leftarrow 0$ ▷ Counter to traverse all nodes
Initialize r ▷ Initialize a sample with N empty values
while $i \neq N$ **do** ▷ Traverse nodes in topological order
  $r[i] \leftarrow x_i \sim P(X_i \mid parents(X_i))$ ▷ Randomly sample from X_i
  $i \leftarrow i + 1$ ▷ Go to next node
end while
return r

For the case of the BN represented in figure 3, the algorithm involves the following steps:

1. Sample from $P(\text{Rain})$. Suppose it returns False
2. Sample from $P(\text{Sprinkler}|\text{Rain} = \text{False})$. Suppose it returns True
3. Sample from $P(\text{Grass Wet}|\text{Rain} = \text{False}, \text{Sprinkler} = \text{True})$. Suppose it returns True
4. The sample gathered is Rain=False, Sprinkler=True, Grass Wet=True

[2] A topologically ordering of a DAG is one where for any edge $\langle X_i, X_j \rangle$ from X_i to X_j in the graph, X_i comes first in the ordering. This sorting can be done using *Kahn's algorithm* and should be established only once after constructing the BN to prevent its usage every time inference is to be performed.

Direct sampling therefore allows the extraction of a sample $x_1, x_2, \ldots, x_n \sim P(X_1, X_2, \ldots, X_N)$ represented in the BN. By repeating this algorithm and collecting a total of n_T samples, each occurring $n(x_1, x_2, \ldots, x_N)$ times, the joint probability distribution can be approximated by:

$$P(x_1, x_2, \ldots x_N) \approx \frac{n(x_1, x_2, \ldots, x_N)}{n_T} \tag{12}$$

As an example, if the sample Rain=False, Sprinkler=True, Grass Wet=True occurred 456 times out of a total of 1000 samples, its probability would be approximated by $\frac{456}{1000} = 0.456$.

A single sample can be extracted using *direct sampling* in $\mathcal{O}(NM)$ time [RN09], where N is the total number of nodes in the BN and M is the maximum number of parents of any node in the network. The time complexity is linear with respect to both of these parameters, which is a large benefit when compared to the exponential cost of *exact inference* methods. Nonetheless, this algorithm does not allow sampling from a posterior $P(\mathcal{Q}|\mathcal{E} = e)$, given that it samples from the joint distribution. To solve this issue, another approximate sampling algorithm, namely *rejection sampling*, can be used.

Rejection sampling follows naturally from *direct sampling*. To sample from a probability distribution $P(\mathcal{Q}|\mathcal{E} = e)$, first repeatedly sample from the joint distribution using *direct sampling* and only accepting samples with evidence $\mathcal{E} = e$, discarding any sample that does not satisfy that condition. Using all the accepted samples, a count for every possible value q of the query variables $\mathcal{Q}$ is stored and turned into an approximate probability as in equation 12. This whole procedure is captured by algorithm 2.

Algorithm 2 Rejection sampling algorithm. Returns a conditional probability distribution.

Require:
$\mathcal{B}$, a BN representing the distribution $P(X_1, X_2, \ldots, X_N)$ with nodes topologically ordered;
$\mathcal{Q}$, the set of query variables;
$\mathcal{E}$, the set of evidence variables;
e, the values for the query variables;
n, the number of samples to generate;

```
i ← 0                                          ▷ Counter for the number of samples gathered
Initialize r                                   ▷ Dictionary of counts over all possible values of Q, initially zero
while i ≠ n do
    x ← direct sampling (B)                    ▷ Get a sample from the BN's JPD
    if x is compatible with E = e then
        r [q] ← r [q] + 1                      ▷ Increase count. q are the values of Q in x
        i ← i + 1
    end if
end while
r ← normalize (r)                              ▷ Turn counts into probabilities
return r
```

Due to the rejection of samples in this algorithm, it naturally follows that it is more costly to gather a single sample with *rejection sampling* rather than *direct sampling*. For *rejection sampling*, the time complexity to gather one sample is $\mathcal{O}\left(NMP(e)^{-1}\right)$ [LYC14], since the lower the probability $P(e)$ of the evidence taking value e, the more likely the sample is to be rejected.

Using this algorithm, it is possible to infer from the network any probability distribution involving the random variables of its nodes. If the *Bayesian Network* is modelling an environment, then this inference can be used to extract probabilistic information about it, which is useful to help an agent reason about which scenarios it is likely to encounter and help it make decisions.

3.3 Dynamic Decision Networks

A *Dynamic Decision Network* (DDN) is a type of *Dynamic Bayesian Network*, a collection of Bayesian Networks representing time-slices whose state random variables are connected (see figure 4). As a model that can represent both MDPs and POMDPs, DDNs are able to not only represent a system, but also to make decisions on that system, hence requiring *states*, *observations*, *rewards* and *actions* to be incorporated in its structure. As such, a Dynamic Decision Network must have the following types of nodes for each time-slice:

- *State Nodes*: nodes that represent the state of the environment. For POMDPs, these nodes can not be directly observed by the agent to make decisions.

- *Observation Nodes*: nodes that represent the observations of the environment that allow the agent to infer its belief-state.
- *Reward Nodes*: nodes that allow the environment to send a reward after each transition.
- *Action Nodes*: nodes that represent the actions an agent can choose from, therefore introducing decision making in these Bayesian models. These nodes are root nodes, since their values are not dependent on any random variable, otherwise the decision making process would be biased [RN09].

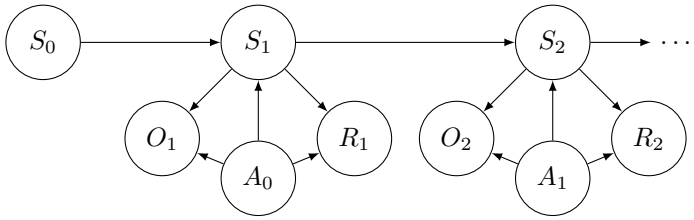


Figure 4: The simplest *Dynamic Decision Network*.

For the action nodes, their values should be encoded directly after choosing the desired action to take. Another important point is that each state, action, observation and reward in a time-slice does not need to be represented by a single node as in figure 4. Each of them can have multiple nodes interconnected in order to represent that state, action, reward or observation, as if they were Bayesian Networks of their own, but connected to each other.

To show that DDNs can model POMDPs, notice from the definition of a POMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \Omega, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \gamma \rangle$ that all of its elements are present in a DDN, except for the discount factor γ, which must be separately defined:

- The sets $\mathcal{S}$, $\mathcal{A}$ and Ω are represented by all the possible values the random variables S_t, A_t and O_t can take, respectively.
- The transition function $\mathcal{P}$, whose values can be obtained by inferring $P(S_{t+1}|S_t, A_t)$, is represented by the edges $\langle S_t, S_{t+1} \rangle$ and $\langle A_t, S_{t+1} \rangle$.
- The reward function $\mathcal{R}$, whose values can be inferred from $\mathbb{E}(R_{t+1}|S_t, A_t)$, is represented by the edges $\langle S_t, R_t \rangle$ and $\langle A_{t-1}, R_t \rangle$.
- The sensor model $\mathcal{Z}$, whose values can be inferred from $P(O_t|S_t, A_{t-1})$, is represented by the edges $\langle S_t, O_t \rangle$ and $\langle A_{t-1}, O_t \rangle$.

As such, given that DDNs are a way of modeling a POMDP, the probability distributions that need to be calculated for the lookahead algorithm described in subsection 2.2 can be extracted by performing rejection sampling over these networks, allowing agents to leverage this structure to make rational decisions in their environment.

4 Quantum Bayesian Inference

As already stated in section 3, Bayesian Networks are a powerful tool for representing probability distributions in a compact way and can be used to solve Reinforcement Learning problems. Nonetheless, inference is required to use Bayesian Networks in order to solve these problems, because it allows the retrieval of conditional probability distributions.

This section presents a quantum version of the rejection sampling inference algorithm described in subsection 3.2, which holds a quadratic speedup over its classical counterpart due to the use of Amplitude Amplification.

Quantum rejection sampling inference works by encoding the Bayesian Network in a quantum operator $\mathcal{B}$ and using Amplitude Amplification to amplify quantum states with evidence $\mathcal{E} = e$. To further explain the workings of this algorithm, this section in broken into two subsections, the first one explaining how to construct the quantum gate $\mathcal{B}$, and the second detailing how to build the evidence phase flip operator $\mathcal{S}_e$ that performs the phase flip on the states with the right evidence, similar to the oracle operator $\mathcal{S}_f$.

4.1 Quantum Embeddings of Bayesian Networks

The problem of encoding a Bayesian Network in a quantum circuit can be easily understood. A Bayesian Network encodes a probability distribution $P(X_1, X_2, \ldots, X_N)$ over its random variables, and encoding it in a quantum circuit means being able to construct a quantum state $|\psi\rangle = \mathcal{B}|0\rangle^{\otimes N}$ whose the qubits represent the random variables. Once the circuit is properly encoded, sampling from the Bayesian Network joint distribution in the quantum setting is equivalent to performing measurements in the qubits of the circuit. Nevertheless, for this method to work, the Bayesian Network can only have *discrete* random variables, due to the encoding scheme used for encoding the Bayesian Network into a quantum circuit. Additionally, for simplicity, this discussion only considers the binary random variables case, where each random variable can only take two values and is encoded in a single qubit. For a description of this process for arbitrary discrete random variables, the reader is referred to [BNN+21]. Hence, to successfully encode the Bayesian Network, the following equation must hold:

$$|\langle x_1 x_2 \ldots x_N|\psi\rangle|^2 = P(X_1 = x_1, X_2 = x_2, \ldots, X_N = x_N) \tag{13}$$

This can be achieved by encoding each of the values of the CPTs in the Bayesian Network into the quantum circuit. To understand how this is done in [LYC14], consider a quantum circuit with N qubits q_i, each representing a binary random variable X_i. As previously seen in section 3, each entry of the CPT encodes a probability $P(X_i = x_i|\text{parents}(X_i) = x_p)$ of the occurrence of a value x_i in a random variable X_i, given a value x_p for the random variable's parents, $\text{parents}(X_i)$. In the quantum setting, for each random variable X_i, a controlled $R_Y(\theta)$ rotation gate is applied for each possible value x_p its parent random variables can take, where the control qubits of this gate are the qubits q_i representing random variables $\text{parents}(X_i)$. This ensures that any random variables that share an edge in the Bayesian Network are entangled in the quantum circuit, expressing the information they share.

Consider the Bayesian Network from figure 3 as an example. To encode it in a quantum circuit, the following gates have to be applied (see figure 5):

- For the variable Rain R, with no parents, an uncontrolled rotation gate is applied.
- For Sprinkler S, with R as a parent, two rotation gates are applied, controlled by the qubit representing R. The first one represents the case when $|R\rangle = |0\rangle$ [3], and the second one when $|R\rangle = |1\rangle$.
- For WetGrass W, which has two parents, four rotation gates, controlled by both $|R\rangle$ and $|S\rangle$, are needed. The first one for when $|RS\rangle = |00\rangle$, the second for when $|RS\rangle = |01\rangle$, the third for $|RS\rangle = |10\rangle$ and a final one for $|RS\rangle = |11\rangle$.

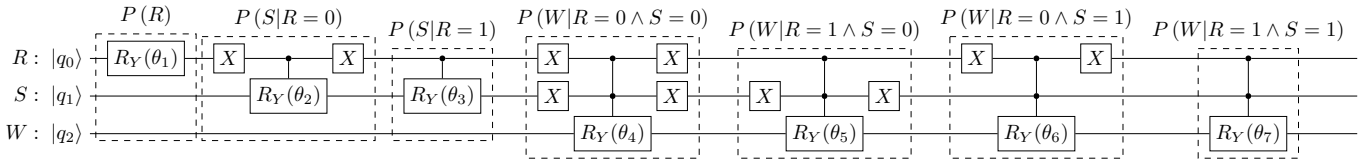


Figure 5: Quantum circuit encoding the Bayesian Network in figure 3.

Finally, the only missing step for the encoding is knowing the angle for each rotation gate. Consider the controlled rotation gate for a binary random variable X_i with value x_p for its parents. The angle of rotation θ is given by:

$$\theta = 2\arctan\left(\sqrt{\frac{P(X_i = 1|\text{parents}(X_i) = x_p)}{P(X_i = 0|\text{parents}(X_i) = x_p)}}\right) \tag{14}$$

As [LYC14] details, this encoding can be done with a complexity $\mathcal{O}(N2^M)$, where N represents the number of random variables in the Bayesian Network and M represents the greatest number of parents of any random variable. Given that this encoding allows the sampling from the joint probability distribution, it can be directly compared to direct sampling, explained in subsection 3.2, which has a complexity of $\mathcal{O}(NM)$. The quantum encoding is therefore

[3] when $|R\rangle = |0\rangle$, an X gate must be applied *before* and *after* the control operation on qubit $|R\rangle$. This is because a controlled gate, by definition, only changes the target state when its control qubit is in state $|1\rangle$, but in this case the opposite effect is wanted. Therefore, the first X gate flips state $|R\rangle$ before the control operation, and the second X gate returns the $|R\rangle$ state to the way it was before the first X gate was applied, leaving it unchanged.

generally slower than direct sampling, due to the at most M qubit uniformly controlled rotations, each resulting in a decomposition of $\mathcal{O}\left(2^M\right)$ elementary gates [BVMS05].

The fact that the encoding phase is more expensive in the quantum case has a very profound implication for applying this algorithm to RL. In the lookahead algorithm showed in subsection 2.2, there are three probability distributions that need to be extracted, as in equation 10, in order to be able to calculate a near-optimal action: $P\left(R_{t+1}|\mathbf{b}_t, a_t\right)$, $P\left(o_{t+1}|\mathbf{b}_t, a_t\right)$ and $P\left(s_{t+1}|o_{t+1}, \mathbf{b}_t, a_t\right)$. Notice that the first two probability distributions are conditioned on the state (or belief-state) and the action, which are root nodes of a DDN when performing inference[4], and therefore their values can be encoded directly into the CPTs of the DDN and encoded into a quantum circuit using the procedure mentioned in this chapter. As a consequence, these probability distributions do not need amplitude amplification to be extracted and therefore are more expensive to obtain using this quantum algorithm, since quantum encoding is more computationally expensive than classical direct sampling. Therefore, even in the quantum setting, only the belief-update probability distribution should be extracted using quantum rejection sampling, while the other two should be extracted using classical rejection sampling. Given that the belief update probability distribution only exists in the partially observable RL setting, it can also be concluded that this quantum algorithm does not hold a quantum speedup for totally observable RL problems.

4.2 The evidence phase flip operator

Having explained the state preparation circuit $\mathcal{B}$, the missing part of the algorithm is the oracle operator, which, in this particular example, is the evidence phase flip operator $\mathcal{S}_e$. In quantum Bayesian inference, the good states are states where the evidence qubits $\mathcal{E}$ matches the evidence value e of the conditional distribution $P\left(\mathcal{Q}|\mathcal{E}=e\right)$ that is to be inferred.

In this sense, the Amplitude Amplification technique turns the joint probability distribution of the BN into the aforementioned conditional distribution by decreasing the amplitudes of states with incorrect evidence $\mathcal{E} \neq e$ and amplifying the amplitudes of states with the right evidence $\mathcal{E}=e$. Formally, the resulting state $|\psi\rangle = \mathcal{B}\left|0\right\rangle^{\otimes N}$ of the Bayesian Network quantum encoding can be expressed as:

$$|\psi\rangle = \sqrt{P(e)}\left|\mathcal{Q}, e\right\rangle + \sqrt{1-P(e)}\left|\mathcal{Q}, \bar{e}\right\rangle \tag{15}$$

Where $|\mathcal{Q}, e\rangle$ represents the quantum states with the correct evidence e, $|\mathcal{Q}, \bar{e}\rangle$ the quantum states with the wrong evidence and $P(e)$ the probability of occurrence of evidence e. As such, the effect of $\mathcal{S}_e$ on this quantum state is given by:

$$\mathcal{S}_e\left|\psi\right\rangle = -\sqrt{P(e)}\left|\mathcal{Q}, e\right\rangle + \sqrt{1-P(e)}\left|\mathcal{Q}, \bar{e}\right\rangle \tag{16}$$

Let $e = e_1 e_2 \ldots e_k$ represent the evidence bit string, with $e_i \in \mathbb{B}$. Constructing this operator requires the help of two other operators:

- The operator $X_i = \left(\bigotimes_{j=1}^{i-1} \mathbb{1}\right) \otimes X \otimes \left(\bigotimes_{j=i+1}^{k} \mathbb{1}\right)$, which flips the i-th evidence qubit by applying an X gate to it and leaving every other qubit unchanged.
- The controlled phase operator $P_{1\ldots k}$, which flips the phase of a quantum state if all evidence qubits are in state $|1\rangle$.

With the above two operators defined, the evidence phase flip operator $\mathcal{S}_e$ can be defined as follows:

$$\mathcal{S}_e = \mathcal{F} P_{1\ldots k} \mathcal{F},\ \mathcal{F} = \prod_{i=1}^{k} X_i^{1-e_i} \tag{17}$$

Consider once more the example of figure 3 with evidence qubits R and W and values 1 and 0, respectively, in order to infer the probability distribution $P\left(S|R=1, W=0\right)$. The evidence phase flip operator for this specific case is represented in figure 6

Notice from equation 17 that the quantum operator $\mathcal{F}$ only flips the evidence qubits with evidence $e_i = 0$, otherwise it leaves them unchanged. This allows for the operator $\mathcal{S}_e$ to behave as expressed by equation 16. To show this, let

[4] Although state nodes are not always root nodes, when performing inference, one always starts with the current state as a root node, removing the dependency of the previous time-steps. This is done because the belief-state (the CPT for that state node) already perfectly captures the past dependencies in full, and therefore previous nodes are removed for memory and time saving computational purposes. As such, the state nodes of time-slice t of the DDN are always root nodes of that DDN at time t.

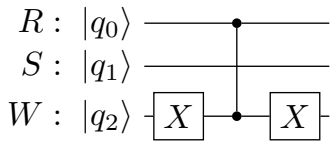


Figure 6: Quantum circuit for the evidence phase flip operator $\mathcal{S}_e$ for the Bayesian Network in figure 3 with evidences $R = 1$ and $W = 0$.

$|\mathcal{Q}\rangle|\tilde{e}\rangle = \mathcal{F}|\mathcal{Q}\rangle|\bar{e}\rangle$ (where $|\tilde{e}\rangle \neq |1\rangle^{\otimes k}$) represent the application of the quantum operator $\mathcal{F}$ on a superposition of quantum states whose evidence does not match e. Then:

$$\begin{aligned}
\mathcal{S}_e|\psi\rangle &= \mathcal{F}P_{1\dots k}\mathcal{F}\left(\sqrt{P(e)}|\mathcal{Q}\rangle|e\rangle + \sqrt{1-P(e)}|\mathcal{Q},\bar{e}\rangle\right) \\
&= \mathcal{F}P_{1\dots k}\left(\sqrt{P(e)}|\mathcal{Q}\rangle|1\rangle^{\otimes k} + \sqrt{1-P(e)}|\mathcal{Q},\tilde{e}\rangle\right) \\
&= \mathcal{F}\left(-\sqrt{P(e)}|\mathcal{Q}\rangle|1\rangle^{\otimes k} + \sqrt{1-P(e)}|\mathcal{Q},\tilde{e}\rangle\right) \\
&= -\sqrt{P(e)}|\mathcal{Q},e\rangle + \sqrt{1-P(e)}|\mathcal{Q},\bar{e}\rangle
\end{aligned} \tag{18}$$

This operator, together with the Bayesian Network encoding, leads to an Amplitude Amplification operator $G = \mathcal{B}\mathcal{S}_0\mathcal{B}^\dagger\mathcal{S}_e$ which allows the quantum inference in Bayesian Networks to be quadratically faster than in the classical case, provided that the Bayesian Network has a small maximum number of parents M:

	Classical	Quantum
Complexity	$\mathcal{O}\left(NMP(e)^{-1}\right)$	$\mathcal{O}\left(N2^M P(e)^{-\frac{1}{2}}\right)$

Table 1: Comparison between classical and quantum complexity for rejection sampling inference in Bayesian Networks.

Another important mention is that this algorithm can also be applied to Dynamic Bayesian Networks, as [BN21] show, by also encoding observation nodes in the quantum circuit and considering a finite number of time-slices of the network at a time, performing inference over those finite slices. The only difference with respect to performing inference in a regular Bayesian Network is their relative size. Moreover, the same concept can be applied to Dynamic Decision Networks by also encoding action and reward nodes into qubits, allowing this inference to be performed in the context of solving Reinforcement Learning problems, such as by using it along with the lookahead algorithm presented in subsection 2.2.

5 Complexity Analysis

In subsection 2.2, a lookahead algorithm to solve partially observable RL problems is detailed in a general form, where the model for the environment is not imposed. As a way to propose a model for the environment, subsection 3.3 explains DDNs, Bayesian Networks that model POMDPs and leverage inference algorithms to extract probability distributions. This work presents two inference algorithms: classical and quantum rejection sampling, which, merged with the lookahead algorithm, yield two algorithms to solve partially observable RL problems. This section is devoted to a time complexity analysis of both the classical and quantum-classical lookahead algorithms in order to compare them and check for potential speedups in the quantum approach.

Both of these lookahead algorithms, however, are approximate algorithms, and probabilistic at that. To perform these algorithms, samples have to be extracted and used to approximate probability distributions, which in turn are used to calculate a near-optimal action. As such, to determine the computational complexity of the algorithms, one should also determine their *sample complexity*: the total number of samples needed to execute the algorithm. This section therefore presents results concerning both the *sample complexity* and the *computational complexity* of the algorithms. The proofs for these results are in appendices A and B.

5.1 Sample Complexity

The goal of the sample complexity analysis is to determine the number of samples[5] needed for the algorithm to be guaranteed to have an error at least as small as ϵ with a confidence interval of $1 - \delta$. The choice of error is arbitrary,

[5] which corresponds to the total amount of times that the DDN has to be called

but in RL it is common to use the *regret* [LXP+21] as the error, as in the following definition:

Definition 5.1. *(Lookahead error). Let* $a_t^{\mathcal{L}} = \text{argmax}_{a_t \in \mathcal{A}} Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t)$ *be the near optimal action chosen by the lookahead algorithm* $\mathcal{L}$ *at time* t*. Also let* $\mathbf{b}_{l_t}$ *be the belief-state approximation of* $\mathbf{b}_t$ *due to approximate sampling over* t *time-steps. The lookahead error* ϵ_t *at time* t *is defined as the regret:*

$$\epsilon_t = \left| \max_{a_t \in \mathcal{A}} Q^\star(\mathbf{b}_t, a_t) - Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) \right| \tag{19}$$

Notice that in performing the lookahead algorithm, there are three distinct sampling operations to be performed, and therefore each should have their own number of samples:

- Belief sampling: using rejection sampling to update the belief-state. Suppose each of these sampling operations at time-step t uses l_t samples.
- Observation sampling: using rejection sampling to calculate the observation distribution in equation 10. Suppose each of these sampling operations at time-step t uses m_t samples.
- Reward sampling: using rejection sampling to calculate the reward distribution in equation 10. Suppose each of these sampling operations at time-step t uses n_t samples.

Moreover, to bound the error of the whole algorithm, bounding the error of each sampling operation is necessary, according to some confidence interval. This confidence interval is local for each sampling operation and not to be confused with the global confidence interval of the whole lookahead algorithm. Suppose that the confidence interval for every sampling operation at time t is given by $1 - \sigma_t$. The following two assumptions impose restrictions on these numbers of samples and local confidence interval (used mostly to simplify the calculations of the error bounds):

Assumption 5.2. *Both the number of samples* n_t, m_t *and* l_t *and the confidence interval* $1 - \sigma_t$ *are fixed, such that they do not change over time:*

$$n_t \equiv n, \quad m_t \equiv m, \quad l_t \equiv l, \quad \sigma_t \equiv \sigma$$

Assumption 5.3. *Let* $\Gamma = (1-\gamma)^{-1}$*. The contribution to the lookahead error of each sampling operation is considered to be the same at time-step* $t = 0$*, such that the number of samples can be related to each other according to the following expressions:*

$$m = (\gamma\Gamma)^2 n, \quad l = \left(\frac{1}{4} \frac{\mathcal{R} + \gamma\Gamma\Omega}{\mathcal{S} - 1} \left(\frac{\mathcal{S}}{\Gamma\left(1 - \gamma^H\right)} \frac{(\gamma\mathcal{S})^H - 1}{\gamma\mathcal{S} - 1} - 1 \right) \right)^2 n$$

Finally, consider that there is a stopping time T for the execution of the algorithm. That is, that the algorithm terminates after making T decisions (or T time-steps). The following theorem presents the necessary conditions for the algorithm to be ϵ-optimal with a confidence of $1 - \delta$:

Theorem 5.4. *Consider a stopping time* T*, horizon* H *and assumptions 5.2 and 5.3 for the lookahead algorithm. A near optimal action is guaranteed to be taken at each time-step of the decision making process with a regret of at most* ϵ *and confidence interval of at least* $1 - \delta$ *if all of the following conditions are met:*

$$n \geq \frac{1}{2} \log\left(\frac{2}{\sigma}\right) \left(\frac{r_{max}}{\epsilon - 2\gamma^H \Gamma r_{max}} \left(8\Gamma + 4\mathcal{S}^T\right) \right)^2$$

$$\sigma \leq \delta \left(\mathcal{A}\Omega\right)^{-H} \left(2 + \mathcal{A} \left(T + 4 + \mathcal{A}\Omega \frac{H-1}{\left(\mathcal{A}\Omega - 1\right)^2} \right) \right)^{-1}$$

$$H > \log_\gamma \left(\frac{\epsilon}{2\Gamma r_{max}} \right)$$

5.2 Computational Complexity

To calculate the computational complexity of the lookahead algorithm, both the classical and quantum-classical versions, one must multiply the total number of samples needed by the computational complexity necessary to compute each sample.

Recalling the distinction between classical and quantum rejection sampling, captured in table 1, the former has a complexity of $\mathcal{O}\left(NMP(e)^{-1}\right)$. Now utilizing this complexity to analyze the whole lookahead algorithm, one must be careful to note that the evidences e vary for each sampling operation along the lookahead-tree:

- For reward sampling, let $c_n \equiv P(e)^{-1}$. It represents the average inverse reward probability $P\left(r_{t+1}|\mathbf{b}_t, a_t\right)^{-1}$ along every observation node. Since the reward distribution is conditioned on the belief-state and action, c_n is the average of the inverse probabilities for every action and belief-state along the lookahead tree.
- For observation sampling, let $c_m \equiv P(e)^{-1}$. It represents the average inverse observation probability $P\left(o_{t+1}|\mathbf{b}_t, a_t\right)^{-1}$ along every observation node. This distribution is also conditioned on the belief-state and action, hence $c_m = c_n$.
- For belief-state sampling, let $c_l \equiv P(e)^{-1}$. It represents the average inverse belief update probability $P\left(s_{t+1}|o_{t+1}, \mathbf{b}_t, a_t\right)$ along every belief node. This is conditioned on the previous belief-state, action and observation, hence c_l is the average of the inverse probabilities for each belief-state, action and observation in the lookahead tree.

Given this analysis, the computational complexity of the classical lookahead algorithm is characterized in theorem 5.5:

Theorem 5.5. *The computational complexity of the classical lookahead algorithm is given by:*

$$\mathcal{O}\left(NM\mathcal{A}^{H-1}\Omega^{H-1}\left(\mathcal{A}c_n + \left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2 c_l\right)\left(\frac{\mathcal{S}^T}{\epsilon}\right)^2 \log\left(\left(T + \frac{H}{\mathcal{A}\Omega}\right)\frac{\mathcal{A}^{H+1}\Omega^H}{\delta}\right)\right)$$

In the quantum-classical case only belief-state sampling uses quantum rejection sampling, while the other two operations still use classical rejection sampling, and therefore only belief-state sampling needs to be re-evaluated in this discussion. The complexity for belief-state sampling for the whole algorithm is of the order $\mathcal{O}\left(N2^M q_l\right)$, where $q_l \equiv P(e)^{-\frac{1}{2}}$. In this scenario, c_l is no longer adequate to represent $P(e)^{-\frac{1}{2}}$, because the latter is an average of the inverse square rooted probabilities $P\left(o_{t+1}, \mathbf{b}_t, a_t\right)^{-\frac{1}{2}}$ and not of the inverse probabilities $P\left(o_{t+1}, \mathbf{b}_t, a_t\right)^{-1}$.

Theorem 5.6 is analogous to the previous theorem, but characterizes the computational complexity of the quantum-classical lookahead algorithm:

Theorem 5.6. *The computational complexity of the quantum-classical lookahead algorithm is given by:*

$$\mathcal{O}\left(N\mathcal{A}^{H-1}\Omega^{H-1}\left(\mathcal{A}Mc_n + \left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2 2^M q_l\right)\left(\frac{\mathcal{S}^T}{\epsilon}\right)^2 \log\left(\left(T + \frac{H}{\mathcal{A}\Omega}\right)\frac{\mathcal{A}^{H+1}\Omega^H}{\delta}\right)\right)$$

As expected, a partial speedup is obtained in the quantum case for the belief-state sampling operations, while the computational complexity of the rest of the algorithm is similar across both classical and quantum algorithms. However, it is interesting to take a closer look at the scenarios where the portion of the algorithm that uses a quantum subroutine dominates the complexity of the algorithm so much that the contribution of the classical subroutines to the computational complexity of the algorithm is negligible. In these scenarios, detailed in corollaries 5.7 and 5.8, the quantum-classical lookahead has its greatest possible speedup over its classical counterpart:

Corollary 5.7. *Under the assumption that* $\frac{c_n}{q_l} \ll \frac{\left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2}{\mathcal{A}}$*, the computational complexity of the classical lookahead algorithm of theorem B.1 can be re-written as:*

$$\mathcal{O}\left(NM\mathcal{A}^{H-1}\Omega^{H-1}\left(\left(\mathcal{R}+\Omega\right)\frac{\mathcal{S}^{T+H-1}}{\epsilon}\right)^2 c_l \log\left(\left(T + \frac{H}{\mathcal{A}\Omega}\right)\frac{\mathcal{A}^{H+1}\Omega^H}{\delta}\right)\right)$$

Corollary 5.8. *Under the assumption that* $\frac{c_n}{q_l} \ll \frac{\left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2}{\mathcal{A}}$*, and that M is small enough such that M and 2^M have about the same order of magnitude, the computational complexity of the quantum-classical lookahead algorithm of theorem 5.6 can be re-written as:*

$$\mathcal{O}\left(N2^M\mathcal{A}^{H-1}\Omega^{H-1}\left(\left(\mathcal{R}+\Omega\right)\frac{\mathcal{S}^{T+H-1}}{\epsilon}\right)^2 q_l \log\left(\left(T + \frac{H}{\mathcal{A}\Omega}\right)\frac{\mathcal{A}^{H+1}\Omega^H}{\delta}\right)\right)$$

Corollaries 5.7 and 5.8 compares the classical and quantum algorithm's complexity in the best case scenario for the quantum algorithm, where $\frac{c_n}{q_l} \ll \frac{\left((\mathcal{R}+\Omega)\mathcal{S}^{H-1}\right)^2}{\mathcal{A}}$. It remains to be answered as to what RL problems this assumption can be applied to in practice, as this work does not address this question. Nonetheless, it gives a benchmark as to what speedup one can expect from the quantum algorithm in a best case scenario.

What speedups can therefore be expected? Although one might initially think that using quantum rejection sampling would provide a quadratic speedup, this is not always the case. Dividing the classical complexity by the quantum complexity (and assuming that M and 2^M have about the same order of magnitude), this fraction reduces to $\frac{c_l}{q_l} = \frac{\sum_x P^{-1}(x)}{\sum_x P^{-\frac{1}{2}}(x)}$, where the sum over x represents the sum over the different evidences. It just so happens that $\sqrt{\sum_x a_x} \leq \sum_x \sqrt{a_x} \leq \sum_x a_x$, such that:

$$1 \leq \frac{c_l}{q_l} \leq q_l \tag{20}$$

The inequality of equation 20 entails that the complexity of the quantum algorithm relative to the classical one can range between no speedup to a quadratic one, but it is most likely to fall somewhere in between these two extremes. This comes from the notion that, although using quantum rejection sampling yields a quadratic advantage over classical rejection sampling, the same cannot be said about using it sequentially, due to the mathematical fact that the sum of square roots can be larger than the square root of the sum.

6 Conclusion

Quantum computing has found its usefulness in many other scientific fields, including RL. This work studies the potential advantage of quantum computer for solving partially observable RL problems, more specifically using an approximate model-based lookahead algorithm with a Dynamic Decision Network modeling the environment's POMDP. The environment dynamics are extracted classically using rejection sampling and in a quantum setting using quantum rejection sampling, and these two methods are compared when inserted in the whole lookahead algorithm.

It can be concluded from this work, as seen in subsection 4.1, that quantum rejection sampling can only be benefitial for solving partially observable RL problems and not fully observable ones, since quantum rejection sampling can be replaced by a simple bayesian network encoding quantum circuit for the latter, which is more expensive than the classical alternative, direct sampling.

Moreover, a complete analysis of the sample complexity of the algorithm is given, which can be applied for both the classical and quantum algorithms, and the conditions for ϵ-optimality with a $1-\delta$ confidence interval, using the regret as the error definition, are presented in subsection 5.1.

Finally, it can be concluded that, although there is a local quadratic speedup for each sampling operation in the lookahead algorithm in the quantum setting, due to the use of quantum rejection sampling, the same speedup is not observed in the full lookahead algorithm, as subsection 5.2 details. Nonetheless, although the quantum speedup of using the presented quantum-classical lookahead algorithm can range from none to quadratic, it is still factual that a speedup will be observed in most use cases, making the quantum algorithm the most efficient choice in terms of time complexity.

A possible fully quantum algorithm for extracting an action from the lookahead tree might be more beneficial than the presented quantum-classical algorithm and is an interesting for possible future work. Another interesting direction that could be taken is considering a model-free approach for this algorithm, where the DDN for the environment is not known, and has to be learn at the same time as the agent is making decisions, which is a much harder task to solve, and might find quantum speedups in novel ways that this work does not consider.

References

[BN21] Sima E Borujeni and Saideep Nannapaneni. Modeling time-dependent systems using dynamic quantum bayesian networks. *arXiv preprint arXiv:2107.00713*, 2021.

[BNN+21] Sima E Borujeni, Saideep Nannapaneni, Nam H Nguyen, Elizabeth C Behrman, and James E Steck. Quantum circuit representation of bayesian networks. *Expert Systems with Applications*, 176:114768, 2021.

[BVMS05] Ville Bergholm, Juha J. Vartiainen, Mikko Möttönen, and Martti M. Salomaa. Quantum circuits with uniformly controlled one-qubit gates. *Physical Review A*, 71(5), May 2005.

[BWP+17] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[CRO+19] Yudong Cao, Jonathan Romero, Jonathan P. Olson, Matthias Degroote, Peter D. Johnson, Má ria Kieferová, Ian D. Kivlichan, Tim Menke, Borja Peropadre, Nicolas P. D. Sawaya, Sukin Sim, Libor Veis, and Alán Aspuru-Guzik. Quantum chemistry in the age of quantum computing. *Chemical Reviews*, 119(19):10856–10915, aug 2019.

[dOB21] Michael de Oliveira and Luis Soares Barbosa. Quantum Bayesian decision-making. *Foundations of Science*, pages 1–21, 2021.

[DTB17] Vedran Dunjko, Jacob M Taylor, and Hans J Briegel. Advances in quantum reinforcement learning. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 282–287. IEEE, 2017.

[HBC+21] Hsin-Yuan Huang, Michael Broughton, Jordan Cotler, Sitan Chen, Jerry Li, Masoud Mohseni, Hartmut Neven, Ryan Babbush, Richard Kueng, John Preskill, et al. Quantum advantage in learning from experiments. *arXiv preprint arXiv:2112.00778*, 2021.

[HKP21] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Information-theoretic bounds on quantum advantage in machine learning. *Physical Review Letters*, 126(19), may 2021.

[JTN+21] Sofiene Jerbi, Lea M Trenkwalder, Hendrik Poulsen Nautrup, Hans J Briegel, and Vedran Dunjko. Quantum enhancements for deep reinforcement learning in large spaces. *PRX Quantum*, 2(1):010328, 2021.

[KHV06] Ranganath Kothamasu, Samuel H Huang, and William H VerDuin. System health monitoring and prognostics—a review of current paradigms and practices. *The International Journal of Advanced Manufacturing Technology*, 28(9-10):1012–1024, 2006.

[Lit09] Michael L Littman. A tutorial on partially observable markov decision processes. *Journal of Mathematical Psychology*, 53(3):119–125, 2009.

[LML+17] Chenzhao Li, Sankaran Mahadevan, You Ling, Sergio Choze, and Liping Wang. Dynamic Bayesian network for aircraft wing health monitoring digital twin. *Aiaa Journal*, 55(3):930–941, 2017.

[LXP+21] Xu-Hui Liu, Zhenghai Xue, Jingcheng Pang, Shengyi Jiang, Feng Xu, and Yang Yu. Regret minimization experience replay in off-policy reinforcement learning. *Advances in Neural Information Processing Systems*, 34:17604–17615, 2021.

[LYC14] Guang Hao Low, Theodore J. Yoder, and Isaac L. Chuang. Quantum inference on Bayesian networks. *Physical Review A - Atomic, Molecular, and Optical Physics*, 89(6), 2014.

[Mey99] David A. Meyer. Quantum strategies. *Physical Review Letters*, 82(5):1052–1055, feb 1999.

[MR02] Kevin Patrick Murphy and Stuart Russell. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, 2002.

[MW16] Catarina Moreira and Andreas Wichert. Quantum-like bayesian networks for modeling decision making. *Frontiers in psychology*, page 11, 2016.

[PAB+20] Stefano Pirandola, Ulrik L Andersen, Leonardo Banchi, Mario Berta, Darius Bunandar, Roger Colbeck, Dirk Englund, Tobias Gehring, Cosmo Lupo, Carlo Ottaviani, et al. Advances in quantum cryptography. *Advances in optics and photonics*, 12(4):1012–1236, 2020.

[PDM+14] Giuseppe Davide Paparo, Vedran Dunjko, Adi Makmal, Miguel Angel Martin-Delgado, and Hans J Briegel. Quantum speedup for active learning agents. *Physical Review X*, 4(3):031002, 2014.

[RN09] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

[SAH+21] Valeria Saggio, Beate E Asenbeck, Arne Hamann, Teodor Strömberg, Peter Schiansky, Vedran Dunjko, Nicolai Friis, Nicholas C Harris, Michael Hochberg, Dirk Englund, et al. Experimental quantum speed-up in reinforcement learning agents. *Nature*, 591(7849):229–233, 2021.

[Sen92] Eugene Seneta. On the history of the strong law of large numbers and boole's inequality. *Historia Mathematica*, 19(1):24–39, 1992.

[SWW+18] Aaron Sidford, Mengdi Wang, Xian Wu, Lin F Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving discounted markov decision process with a generative model. *arXiv preprint arXiv:1806.01492*, 2018.

[ZD03] JY Zhu and A Deshmukh. Application of Bayesian decision networks to life cycle engineering in Green design and manufacturing. *Engineering Applications of Artificial Intelligence*, 16(2):91–103, 2003.

A Sample Complexity Analysis

The sample complexity of the quantum-classical algorithm counts the total number of samples [6] in order to achieve a certain goal. In this analysis, the chosen goal is to reach an ϵ-optimal approximation of the Q-value of the optimal action, with a confidence interval of $1-\delta$.

To start off, recall the definition of the action value function $Q^\star(\mathbf{b}_t, a_t)$ as in equation 8:

$$Q^\star(\mathbf{b}_t, a_t) = \mathbb{E}\left(r_{t+1}|\mathbf{b}_t\right) + \gamma \sum_{o_{t+1}} P(o_{t+1}|\mathbf{b}_t, a_t) \max_{a_{t+1}\in\mathcal{A}} Q^\star(\mathbf{b}_{t+1}, a_{t+1})$$

$$\mathbf{b}_{t+1}(s_{t+1}) = P(s_{t+1}|o_{t+1}, a_t, \mathbf{b}_t)$$

The analysis of the sample complexity of the algorithm entails many manipulations of the above expressions, and for the sake of compactness and comprehensibility of the proofs for the lemmas and theorems ahead, it is useful to resort to vector notation [7]. As such, let $\mathbf{p}_{r_t} \in \Delta_{\mathcal{R}}$ [8] represent the probability distribution $P(r_{t+1}|\mathbf{b}_t), \forall r_{t+1} \in \mathcal{R}$, let $\mathbf{p}_{o_t} \in \Delta_\Omega$ represent the probability distribution $P(o_{t+1}|\mathbf{b}_t, a_t), \forall o_{t+1} \in \Omega$, let $\mathbf{r} \in \mathbb{R}^{|\mathcal{R}|}$ be a vector whose entries enumerate every possible reward and $\mathbf{V}^\star_{t+1} \in \mathbb{R}^{|\Omega|}$ be a vector whose entries are the values $\max_{a_{t+1}\in\mathcal{A}} Q^\star(\mathbf{b}_{t+1}, a_{t+1})$ for every belief update of $\mathbf{b}_t$ after taking action a_t [9]. The previous equation can be re-written, using this notation, as:

$$Q^\star(\mathbf{b}_t, a_t) = \mathbf{p}_{r_t}^\top \mathbf{r} + \gamma \mathbf{p}_{o_t}^\top \mathbf{V}^\star_{t+1} \tag{21}$$

The lookahead algorithm $\mathcal{L}$ calculates a near-optimal value function $Q^\mathcal{L}$ similarly to that of equation 21, but replacing $\mathbf{p}_{r_t}$ with its empirical estimation $\mathbf{p}_{r_t,n_t}$ using n_t samples, $\mathbf{p}_{o_t}$ with its empirical estimation $\mathbf{p}_{o_t,m_t}$ using m_t samples and $\mathbf{V}^\star_{t+1}$ with its empirical estimation $\mathbf{V}^\mathcal{L}_{t+1}$ calculated recursively. Thus, the lookahead algorithm, starting with root belief node $\mathbf{b}_t$ at time-step t obeys the following recursive equation 22 and stopping case 23 (according to the lookahead horizon H):

$$Q^\mathcal{L}(\mathbf{b}_t, a_t) = \mathbf{p}_{r_t,n_t}^\top \mathbf{r} + \gamma \mathbf{p}_{o_t,m_t}^\top \mathbf{V}^\mathcal{L}_{t+1} \tag{22}$$

$$Q^\mathcal{L}(\mathbf{b}_{t+H}, a_{t+H}) = 0 \tag{23}$$

So far, three sources of error for the lookahead algorithm have been mentioned:

- Approximating $\mathbf{p}_{r_t}$ by its empirical estimation $\mathbf{p}_{r_t,n_t}$, using n_t samples.
- Approximating $\mathbf{p}_{o_t}$ by its empirical estimation $\mathbf{p}_{o_t,m_t}$, using m_t samples.
- The stopping case for the lookahead algorithms in the horizon H limit.

There is yet another source of error, coming from the estimation of the belief state. This estimation, however, differs slightly from the estimations of the two distributions mentioned above. Suppose we have an empirical estimation $\mathbf{b}_{l_t}$ of the belief state $\mathbf{b}_t$, using l_t samples. When performing the belief update to get the next belief state $\mathbf{b}_{l_{t+1}}(s_{t+1}) = P_{l_{t+1}}(s_{t+1}|o_{t+1}, a_t, \mathbf{b}_{l_t})$, this quantity estimates $P(s_{t+1}|o_{t+1}, a_t, \mathbf{b}_{l_t})$ rather than the belief-update $P(s_{t+1}|o_{t+1}, a_t, \mathbf{b}_t)$ of the true belief-state $\mathbf{b}_t$. This happens because the belief-update in this algorithm uses the previous estimation (rather than the true belief state) to calculate the belief state in the next time-step, samples from it afterwards and keeps repeating this process. This sequential sampling process is the source of the belief approximation error.

Definition A.1 introduces the lookahead error using the notion of *regret* [10] [LXP+21]. The remainder of this section serves to provide and analyze an upper bound to this error in order to give theoretical guarantees to the performance of the algorithm.

Definition A.1. *(Lookahead error). Let $a_t^\mathcal{L} = \text{argmax}_{a_t\in\mathcal{A}} Q^\mathcal{L}(\mathbf{b}_{l_t}, a_t)$ be the near optimal action chosen by the algorithm at time t. The lookahead error ϵ_t at time t is defined as the regret:*

$$\epsilon_t = \left| \max_{a_t\in\mathcal{A}} Q^\star(\mathbf{b}_t, a_t) - Q^\star(\mathbf{b}_{l_t}, a_t^\mathcal{L}) \right| \tag{24}$$

[6] which corresponds to the total amount of times that the DDN has to be called

[7] Although vector notation in preferred in this section, the previous notation is sometimes clearer and is still used occasionally.

[8] In this notation, $\mathbf{p} \in \Delta_\mathcal{V}$ means that $\mathbf{p}$ is a probability distribution vector over the space $\mathcal{V}$.

[9] Since the belief update rule is $\mathbf{b}_{t+1}(s_{t+1}) = P(s_{t+1}|o_{t+1}, a_t, \mathbf{b}_t)$ and both a_t and $\mathbf{b}_t$ are fixed, there are $|\Omega|$ different possible $\mathbf{b}_{t+1}$, one for each observation $o_{t+1} \in \Omega$.

[10] The *regret* in RL is a quantity generally used to define an algorithm's error. It is defined as the absolute difference between the utility of the best action that could be taken and the utility of the action that the agent actually took.

To derive the upper bound on the algorithm's error, some lemmas will be useful to resort to in the proofs of this section. One example is lemma A.2, an inequality that relates the difference of the maxima of two vectors and the maximum of their difference:

Lemma A.2. *Let $\mathbf{U}, \mathbf{V} \in \mathcal{W}$ be vectors in some vector space $\mathcal{W}$. Then:* [11]

$$\|\mathbf{U}\|_\infty - \|\mathbf{V}\|_\infty \leq \|\mathbf{U} - \mathbf{V}\|_\infty$$

Another result that is used repeatedly throughout the proofs of this section is a generalization of the triangle's inequality, stated in lemma A.3:

Lemma A.3. *Let $\mathbf{U}, \mathbf{V} \in \mathcal{W}$ be vectors in some euclidean space $\mathcal{W}$. Then:*

$$\|\mathbf{U} + \mathbf{V}\| \leq \|\mathbf{U}\| + \|\mathbf{V}\|$$

Using the two lemmas stated above, lemma A.4 decomposes the lookahead error into three distinct errors, each easier to bound than the original definition of the error:

Lemma A.4. *The lookahead error ϵ_t can be bounded by the following expression:*

$$\epsilon_t \leq \max_{a_t \in \mathcal{A}} \left(\left| Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t) \right| + \left| Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t) \right| \right) + \left| Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) \right|$$

Proof.

$$\begin{aligned}
\epsilon_t &\overset{(A.1)}{=} \left| \max_{a_t \in \mathcal{A}} Q^\star(\mathbf{b}_t, a_t) - Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) \right| \\
&\overset{(A.3)}{\leq} \left| \max_{a_t \in \mathcal{A}} Q^\star(\mathbf{b}_t, a_t) - \max_{a_t \in \mathcal{A}} Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t) \right| + \left| Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) - \max_{a_t \in \mathcal{A}} Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t) \right| \\
&\overset{(A.2)}{\leq} \max_{a_t \in \mathcal{A}} \left| Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t) \right| + \left| Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) \right| \\
&= \max_{a_t \in \mathcal{A}} \left| \left(Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t) \right) + \left(Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t) \right) \right| + \left| Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) \right| \\
&\overset{(A.3)}{\leq} \max_{a_t \in \mathcal{A}} \left(\left| Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t) \right| + \left| Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t) \right| \right) + \left| Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) \right|
\end{aligned}$$

□

Lemma A.4 states that the total lookahead error can be decomposed into three distinct errors:

- $\left| Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t) \right|$: the error between the true action value function and the approximate one, both evaluated at belief state $\mathbf{b}_t$.
- $\left| Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t) \right|$: the error between the approximate action value function evaluated at the true belief state $\mathbf{b}_t$ and at the approximate belief-state $\mathbf{b}_{l_t}$.
- $\left| Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) \right|$: the error between the true and approximate action value functions, both evaluated at the approximate belief $\mathbf{b}_{l_t}$ and the near optimal action $a_t^\star$.

The remainder of this section determines upper bounds to all error terms in order to define an upper bound to the lookahead error. The main result that allows to define these bounds is the Hoeffding's inequality [SWW+18], stated in theorem A.5.

Theorem A.5. *(Hoeffding's Inequality). Let $\mathbf{p} \in \Delta_{\mathcal{V}}$ be a probability vector and $\mathbf{V} \in \mathbb{R}^{|\mathcal{V}|}$ a vector. Let $\mathbf{p}_m \in \Delta_{\mathcal{V}}$ be an empirical estimation of $\mathbf{p}$ using m i.i.d.* [12] *samples and $\delta \in (0, 1)$ be a parameter. Then, with probability at least $1 - \delta$:*

$$\left| \mathbf{p}^\top \mathbf{V} - \mathbf{p}_m^\top \mathbf{V} \right| \leq \|V\|_\infty \sqrt{\frac{1}{2m} \log\left(\frac{2}{\delta}\right)}$$

[11] $\|\cdot\|_\infty$ denotes the l_∞ norm of a vector, which extracts its maximum element. If $u = \|\mathbf{U}\|_\infty$, then $u \geq u', \forall u' \in \mathbf{U}$.
[12] Independent and identically distributed

The lemmas and theorems above already provide the necessary tools to bound two of the error sources that compose the lookahead error as in lemma A.4. Using these previous results, lemma A.6 bounds the error between the true value function and the approximate one, both evaluated at the same belief state and action:

Lemma A.6. *Let $\Gamma = (1-\gamma)^{-1}$ be the effective horizon, $r_{max} = \|\mathbf{r}\|_\infty$ be the maximum reward and $\sigma_t \in (0,1)$ a parameter defining the confidence interval $1-\sigma_t$ for sampling both $\mathbf{p}_{r_t}$ and $\mathbf{p}_{o_t}$ at time-step t. The error $\left|Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t)\right|$ has the following upper bound:*

$$\left|Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t)\right| \leq \frac{r_{max}}{\sqrt{2}} \sum_{k=0}^{H-1} \gamma^k \sqrt{\log\left(\frac{2}{\sigma_{t+k}}\right)} \left(\sqrt{n_{t+k}^{-1}} + \gamma\Gamma\sqrt{m_{t+k}^{-1}}\right) + \gamma^H \Gamma r_{max}$$

Proof. Using the definitions for $Q^\star$ and $Q^{\mathcal{L}}$ in equations 21 and 22, respectively:

$$\left|Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t)\right| \overset{(A.3)}{\leq} \left|\mathbf{r}^\top \mathbf{p}_{r_t} - \mathbf{r}^\top \mathbf{p}_{r_t,n_t}\right| + \gamma \left|\mathbf{p}_{o_t}^\top \mathbf{V}_{t+1}^\star - \mathbf{p}_{o_t,m_t}^\top \mathbf{V}_{t+1}^{\mathcal{L}}\right|$$

Using the identity $\mathbf{p}_{o_t} - \mathbf{p}_{o_t,m_t} \leq |\mathbf{p}_{o_t} - \mathbf{p}_{o_t,m_t}| \Leftrightarrow -\mathbf{p}_{o_t,m_t} \leq |\mathbf{p}_{o_t} - \mathbf{p}_{o_t,m_t}| - \mathbf{p}_{o_t}$:

$$\left|Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t)\right| \overset{(A.3)}{\leq} \left|\mathbf{r}^\top \mathbf{p}_{r_t} - \mathbf{r}^\top \mathbf{p}_{r_t,n_t}\right| + \gamma \left|\mathbf{p}_{o_t}^\top \mathbf{V}_{t+1}^{\mathcal{L}} - \mathbf{p}_{o_t,m_t}^\top \mathbf{V}_{t+1}^{\mathcal{L}}\right| + \gamma \left|\mathbf{p}_{o_t}^\top \mathbf{V}_{t+1}^\star - \mathbf{p}_{o_t}^\top \mathbf{V}_{t+1}^{\mathcal{L}}\right|$$

The terms $\left|\mathbf{r}^\top \mathbf{p}_{r_t} - \mathbf{r}^\top \mathbf{p}_{r_t,n_t}\right|$ and $\left|\mathbf{p}_{o_t}^\top \mathbf{V}_{t+1}^{\mathcal{L}} - \mathbf{p}_{o_t,m_t}^\top \mathbf{V}_{t+1}^{\mathcal{L}}\right|$ can be bounded using Hoeffding's inequality in theorem A.5 and the fact that, by definition, $\|\mathbf{V}_{t+1}^{\mathcal{L}}\|_\infty \leq Q_{\max} = \Gamma r_{\max}$. The other term can be decomposed into a sum:

$$\begin{aligned}\left|Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t)\right| \overset{(A.5)}{\leq}\ & r_{\max}\sqrt{\frac{1}{2}\log\left(\frac{2}{\sigma_t}\right)}\left(\sqrt{n_t^{-1}} + \gamma\Gamma\sqrt{m_t^{-1}}\right) \\ & + \gamma \max_{a_{t+1}\in\mathcal{A}} \sum_{o_{t+1}\in\Omega} P(o_{t+1}|\mathbf{b}_t, a_t) \left|Q^\star(\mathbf{b}_{t+1}, a_{t+1}) - Q^{\mathcal{L}}(\mathbf{b}_{t+1}, a_{t+1})\right|\end{aligned}$$

Let $\boldsymbol{\mu}_{t+1}$ be a belief-state and a'_{t+1} an action such that $\left|Q^\star(\boldsymbol{\mu}_{t+1}, a'_{t+1}) - Q^{\mathcal{L}}(\boldsymbol{\mu}_{t+1}, a'_{t+1})\right| = \max_{a_{t+1},\mathbf{b}_{t+1}} \left|Q^\star(\mathbf{b}_{t+1}, a_{t+1}) - Q^{\mathcal{L}}(\mathbf{b}_{t+1}, a_{t+1})\right|$:

$$\begin{aligned}\left|Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t)\right| &\leq r_{\max}\sqrt{\frac{1}{2}\log\left(\frac{2}{\sigma_t}\right)}\left(\sqrt{n_t^{-1}} + \gamma\Gamma\sqrt{m_t^{-1}}\right) \\ &\quad + \gamma \left|Q^\star(\boldsymbol{\mu}_{t+1}, a'_{t+1}) - Q^{\mathcal{L}}(\boldsymbol{\mu}_{t+1}, a'_{t+1})\right| \sum_{o_{t+1}\in\Omega} P(o_{t+1}|\boldsymbol{b}_t, a_t) \\ &= r_{\max}\sqrt{\frac{1}{2}\log\left(\frac{2}{\sigma_t}\right)}\left(\sqrt{n_t^{-1}} + \gamma\Gamma\sqrt{m_t^{-1}}\right) \\ &\quad + \gamma \left|Q^\star(\boldsymbol{\mu}_{t+1}, a'_{t+1}) - Q^{\mathcal{L}}(\boldsymbol{\mu}_{t+1}, a'_{t+1})\right|\end{aligned}$$

Finally, with the recursive application of this inequality to $\left|Q^\star(\boldsymbol{\mu}_{t+1}, a'_{t+1}) - Q^{\mathcal{L}}(\boldsymbol{\mu}_{t+1}, a'_{t+1})\right|$, using equation 23 as a stopping case to the approximate action value function, the result of this lemma is attained:

$$\begin{aligned}\left|Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t)\right| &\leq \frac{r_{\max}}{\sqrt{2}} \sum_{k=0}^{H-1} \gamma^k \sqrt{\log\left(\frac{2}{\sigma_{t+k}}\right)} \left(\sqrt{n_{t+k}^{-1}} + \gamma\Gamma\sqrt{m_{t+k}^{-1}}\right) \\ &\quad + \gamma^H \left|Q^\star(\boldsymbol{\mu}_{t+H}, a'_{t+H}) - Q^{\mathcal{L}}(\boldsymbol{\mu}_{t+H}, a'_{t+H})\right| \\ &\overset{(23)}{=} \frac{r_{\max}}{\sqrt{2}} \sum_{k=0}^{H-1} \gamma^k \sqrt{\log\left(\frac{2}{\sigma_{t+k}}\right)} \left(\sqrt{n_{t+k}^{-1}} + \gamma\Gamma\sqrt{m_{t+k}^{-1}}\right) \\ &\quad + \gamma^H \left|Q^\star(\boldsymbol{\mu}_{t+H}, a^\star_{t+H})\right| \\ &\leq \frac{r_{\max}}{\sqrt{2}} \sum_{k=0}^{H-1} \gamma^k \sqrt{\log\left(\frac{2}{\sigma_{t+k}}\right)} \left(\sqrt{n_{t+k}^{-1}} + \gamma\Gamma\sqrt{m_{t+k}^{-1}}\right) + \gamma^H \Gamma r_{\max}\end{aligned}$$

□

Lemma A.6 gives a bound on the difference between the true value function and the value function calculated by the lookahead algorithm, both evaluated at the same action and belief-state. Notice, however, that the bound holds for any choice of action and belief-state, because it is independent of both these quantities. Thus, both $\left|Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t)\right|$ and $\left|Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t^{\mathcal{L}})\right|$ can be bounded by this lemma.

Before moving on to the next lemma, it will prove useful to first count the total number of Hoeffding inequalities that were used in the proof of lemma A.6. This count will be useful later on to calculate the confidence interval of the lookahead algorithm's bounds, but more on that later. The bound in lemma A.6 was attained by splitting the error source $\left|Q^\star(\boldsymbol{b}_t, a_t) - Q^{\mathcal{L}}(\boldsymbol{b}_t, a_t)\right|$ in two terms, one where the Hoeffding bounds are applied ($\left|\mathbf{r}^\top \mathbf{p}_{r_t} - \mathbf{r}^\top \mathbf{p}_{r_t,n_t}\right| + \gamma \left|\mathbf{p}_{o_t}^\top \mathbf{V}_{t+1}^{\mathcal{L}} - \mathbf{p}_{o_t,m_t}^\top \mathbf{V}_{t+1}^{\mathcal{L}}\right|$) and a recursive term ($\gamma \left|\mathbf{p}_{o_t}^\top \mathbf{V}_{t+1}^{\star} - \mathbf{p}_{o_t}^\top \mathbf{V}_{t+1}^{\mathcal{L}}\right|$) where the same procedure is applied until a stopping case is found. Let's call these two terms *immediate* and *recursive*, respectively. To bound the immediate term, 2 Hoeffding inequalities were used. For the first time-step in the recursive term, $2\mathcal{A}\Omega$ Hoeffding inequalities are needed (2 Hoeffding inequalities as per the immediate term, but for every action and every observation). As such, a total of $2\sum_{k=0}^{H-1} \mathcal{A}^k \Omega^k$ Hoeffding inequalities are used to derive the bound of lemma A.6.

To bound the full lookahead error, only the error of approximating the belief-state remains to be bounded. As mentioned above, the belief-states for the lookahead algorithm are approximated in a different manner to the reward and observation distributions. The only true belief state that is known to the lookahead algorithm is the prior $\mathbf{b}_0$, and therefore, the belief estimations $\mathbf{b}_{l_t}$ are calculated by performing a belief-update on the previous estimation $\mathbf{b}_{l_{t-1}}$, yielding the distribution $\mathbf{p}_t \in \Delta_{\mathcal{S}}$ with entries $P(s_t|o_t, a_{t-1}, \mathbf{b}_{l_{t-1}})$, and extracting l_t samples from it, creating an empirical estimation distribution $\mathbf{p}_{t,l_t} \equiv \mathbf{b}_{l_t} \in \Delta_{\mathcal{S}}$ with entries $P_{l_t}(s_t|o_t, a_{t-1}, \mathbf{b}_{l_{t-1}})$. As such, Hoeffding's inequality can not be applied to the distributions $\mathbf{b}_t$ and $\mathbf{b}_{l_t}$, because they are not direct estimations of one another. It can be applied, however, to $\mathbf{p}_t$ and $\mathbf{p}_{t,l_t}$, since the latter approximates the former distribution via sampling.

Lemma A.7. *Given a distribution $\mathbf{p}_{s_t}$ from a family of distributions $P(\cdot|s_t, \cdot)$ conditioned on the state $s_t \in \mathcal{S}$ [13], the following inequality bounds the error of approximating the belief-state at time t, using l_k samples and a confidence interval of $1 - \sigma_k$ to estimate it at time-step k:*

$$\left|\mathbf{p}_{s_t}^\top \mathbf{b}_t - \mathbf{p}_{s_t}^\top \mathbf{b}_{l_t}\right| \leq \mathcal{S}^t \sum_{k=0}^{t} \frac{1}{\mathcal{S}^k} \sqrt{\frac{1}{2l_k} \log\left(\frac{2}{\sigma_k}\right)}$$

Proof.

$$\left|\mathbf{p}_{s_t}^\top \mathbf{b}_t - \mathbf{p}_{s_t}^\top \mathbf{b}_{l_t}\right| = \left|\sum_{s_t \in \mathcal{S}} P(\cdot|s_t, \cdot) \left(P(s_t|o_t, a_{t-1}, \mathbf{b}_{t-1}) - P_{l_t}(s_t|o_t, a_{t-1}, \mathbf{b}_{l_{t-1}})\right)\right|$$

Using $P(s_t|o_t, a_{t-1}, \mathbf{b}_{t-1}) \leq P(s_t|o_t, a_{t-1}, \mathbf{b}_{l_{t-1}}) + \left|P(s_t|o_t, a_{t-1}, \mathbf{b}_{t-1}) - P(s_t|o_t, a_{t-1}, \mathbf{b}_{l_{t-1}})\right| = P(s_t|o_t, a_{t-1}, \mathbf{b}_{l_{t-1}}) + \left|\mathbf{p}_{s_{t-1}}^\top \mathbf{b}_{t-1} - \mathbf{p}_{s_{t-1}}^\top \mathbf{b}_{l_{t-1}}\right|$, the inequality above can be re-expressed as:

$$\begin{aligned}
\left|\mathbf{p}_{s_t}^\top \mathbf{b}_t - \mathbf{p}_{s_t}^\top \mathbf{b}_{l_t}\right| &\leq \sum_{s_t \in \mathcal{S}} P(\cdot|s_t, \cdot) \left(P(s_t|o_t, a_{t-1}, \mathbf{b}_{l_{t-1}}) - P_{l_t}(s_t|o_t, a_{t-1}, \mathbf{b}_{l_{t-1}}) + \left|\mathbf{p}_{s_{t-1}}^\top \mathbf{b}_{t-1} - \mathbf{p}_{s_{t-1}}^\top \mathbf{b}_{l_{t-1}}\right|\right) \\
&\leq \left|\mathbf{p}_{s_t}^\top \mathbf{p}_t - \mathbf{p}_{s_t}^\top \mathbf{p}_{t,l_t}\right| + \mathcal{S} \left|\mathbf{p}_{s_{t-1}}^\top \mathbf{b}_{t-1} - \mathbf{p}_{s_{t-1}}^\top \mathbf{b}_{l_{t-1}}\right|
\end{aligned}$$

Applying the inequality above in a recursive manner and using Hoeffding's inequality:

$$\begin{aligned}
\left|\mathbf{p}_{s_t}^\top \mathbf{b}_t - \mathbf{p}_{s_t}^\top \mathbf{b}_{l_t}\right| &\leq \sum_{k=0}^{t} \mathcal{S}^{t-k} \left|\mathbf{p}_{s_k}^\top \mathbf{p}_k - \mathbf{p}_{s_k}^\top \mathbf{p}_{k,l_k}\right| \\
&\overset{(A.5)}{\leq} \mathcal{S}^t \sum_{k=0}^{t} \frac{1}{\mathcal{S}^k} \sqrt{\frac{1}{2l_k} \log\left(\frac{2}{\sigma_k}\right)}
\end{aligned}$$

□

This lemma states that the belief-approximation error bound is cumulative over time. This intuitively makes sense, given that the approximation of a belief-state at time-step t is attained by estimating over the previous estimation, and therefore the estimation errors pile up as time progresses.

[13] In the results of this lemma and the lemmas and theorems that follow, the notation $\mathcal{X}$ for a set $\mathcal{X}$ can have two meanings, that depend on the context of use: it can refer to the set itself, or to the size of the set $|\mathcal{X}|$.

A few more bounds are useful to derive the full upper bound of the belief-error. Due to the fact that an inaccurate belief-state is used throughout every time-step of the algorithm, not only does this cause errors in the algorithm's belief-states themselves, but also in calculating other probability distributions that are conditioned on the belief-state (because they should be conditioned on the true belief-state rather than an estimation of it). This implies that both reward sampling and observation sampling also suffer from this belief-state approximation error, and these error contributions should be accounted for. Thus, lemma A.8 bounds the reward sampling error caused by this belief-approximation:

Lemma A.8. *Let $\mathbf{p} \in \Delta_{\mathcal{R}}$ represent the probability distribution $P(r_{t+1}|\mathbf{b}_t)$ and $\hat{\mathbf{p}} \in \Delta_{\mathcal{R}}$ represent the distribution $P(r_{t+1}|\mathbf{b}_{l_t})$. Also, let $\mathbf{p}_{n_t}$ and $\hat{\mathbf{p}}_{n_t}$ be their respective estimations using n_t samples. Then, the following inequality holds:*

$$\left|\mathbf{p}_{n_t}^\top \mathbf{r} - \hat{\mathbf{p}}_{n_t}^\top \mathbf{r}\right| \leq r_{max}\left(2\sqrt{\frac{1}{2n_t}\log\left(\frac{2}{\sigma_t}\right)} + \mathcal{R}\mathcal{S}^t \sum_{k=0}^{t} \frac{1}{\mathcal{S}^k}\sqrt{\frac{1}{2l_k}\log\left(\frac{2}{\sigma_k}\right)}\right)$$

Proof.

$$\begin{aligned}
\left|\mathbf{p}_{n_t}^\top \mathbf{r} - \hat{\mathbf{p}}_{n_t}^\top \mathbf{r}\right| &\leq \left|\left(\mathbf{p}^\top + \left|\mathbf{p}^\top - \mathbf{p}_{n_t}^\top\right|\right)\mathbf{r} + \left(-\hat{\mathbf{p}}^\top + \left|\hat{\mathbf{p}}^\top - \hat{\mathbf{p}}_{n_t}^\top\right|\right)\mathbf{r}\right| \\
&\overset{(A.3)}{\leq} \left|\mathbf{p}^\top \mathbf{r} - \mathbf{p}_{n_t}^\top \mathbf{r}\right| + \left|\hat{\mathbf{p}}^\top \mathbf{r} - \hat{\mathbf{p}}_{n_t}^\top \mathbf{r}\right| + \left|\mathbf{p}^\top \mathbf{r} - \hat{\mathbf{p}}^\top \mathbf{r}\right| \\
&\overset{(A.5)}{\leq} 2r_{\max}\sqrt{\frac{1}{2n_t}\log\left(\frac{2}{\sigma_t}\right)} + \sum_{r_t \in \mathcal{R}} r_t \left|\mathbf{p}_{s_t}^\top \mathbf{b}_t - \mathbf{p}_{s_t}^\top \mathbf{b}_{l_t}\right| \\
&\overset{(A.7)}{\leq} r_{\max}\left(2\sqrt{\frac{1}{2n_t}\log\left(\frac{2}{\sigma_t}\right)} + \mathcal{R}\mathcal{S}^t \sum_{k=0}^{t} \frac{1}{\mathcal{S}^k}\sqrt{\frac{1}{2l_k}\log\left(\frac{2}{\sigma_k}\right)}\right)
\end{aligned}$$

□

Analogously to the lemma A.8, lemma A.9 bounds the observation sampling error due to the belief-state approximation:

Lemma A.9. *Let $\mathbf{p}_{o_t} \in \Delta_\Omega$ be a probability distribution $P(o_{t+1}|\mathbf{b}_t, a_t)$ and $\hat{\mathbf{p}}_{o_t} \in \Delta_\Omega$ be $P(o_{t+1}|\mathbf{b}_{l_t}, a_t)$, such that $\mathbf{p}_{o_t,m_t}$ and $\hat{\mathbf{p}}_{o_t,m_t}$ are their empirical estimations with m_t samples. Also, let $\mathbf{V} \in \mathbb{R}^{|\Omega|}$ be a vector of value functions with entries $V(\mathbf{b}_t)$ and $\mathbf{V}^{\mathcal{L}}$ a similar vector with entries $V^{\mathcal{L}}(\mathbf{b}_{l_t})$. Then, the following inequality holds:*

$$\left|\mathbf{p}_{o_t,m_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V}^{\mathcal{L}}\right| \leq \Gamma r_{max}\left(2\sqrt{\frac{1}{2m_t}\log\left(\frac{2}{\sigma_t}\right)} + \Omega\mathcal{S}^t \sum_{k=0}^{t} \frac{1}{\mathcal{S}^k}\sqrt{\frac{1}{2l_k}\log\left(\frac{2}{\sigma_k}\right)}\right) + \left|\hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V}^{\mathcal{L}}\right|$$

Proof.

$$\begin{aligned}
\left|\mathbf{p}_{o_t,m_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V}^{\mathcal{L}}\right| &\leq \left|\left(\mathbf{p}_{o_t}^\top + \left|\mathbf{p}_{o_t}^\top - \mathbf{p}_{o_t,m_t}^\top\right|\right)\mathbf{V} + \left(-\hat{\mathbf{p}}_{o_t}^\top + \left|\hat{\mathbf{p}}_{o_t}^\top - \hat{\mathbf{p}}_{o_t,m_t}^\top\right|\right)\mathbf{V}^{\mathcal{L}}\right| \\
&\overset{(A.3)}{\leq} \left|\mathbf{p}_{o_t}^\top \mathbf{V} - \mathbf{p}_{o_t,m_t}^\top \mathbf{V}\right| + \left|\hat{\mathbf{p}}_{o_t}^\top \mathbf{V}^{\mathcal{L}} - \hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V}^{\mathcal{L}}\right| + \left|\mathbf{p}_{o_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t}^\top \mathbf{V}^{\mathcal{L}}\right| \\
&\overset{(A.5)}{\leq} 2\Gamma r_{\max}\sqrt{\frac{1}{2m_t}\log\left(\frac{2}{\sigma_t}\right)} + \left|\mathbf{p}_{o_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t}^\top \mathbf{V}\right| + \left|\hat{\mathbf{p}}_{o_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t}^\top \mathbf{V}^{\mathcal{L}}\right| \\
&= 2\Gamma r_{\max}\sqrt{\frac{1}{2m_t}\log\left(\frac{2}{\sigma_t}\right)} + \sum_{o_{t+1} \in \Omega} \hat{V}(\mathbf{b}_{t+1}) \left|\mathbf{p}_{s_t}^\top \mathbf{b}_t - \mathbf{p}_{s_t}^\top \mathbf{b}_{l_t}\right| + \left|\hat{\mathbf{p}}_{o_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t}^\top \mathbf{V}^{\mathcal{L}}\right| \\
&\leq 2\Gamma r_{\max}\sqrt{\frac{1}{2m_t}\log\left(\frac{2}{\sigma_t}\right)} + \Gamma r_{\max}\Omega \left|\mathbf{p}_{s_t}^\top \mathbf{b}_t - \mathbf{p}_{s_t}^\top \mathbf{b}_{l_t}\right| + \left|\hat{\mathbf{p}}_{o_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t}^\top \mathbf{V}^{\mathcal{L}}\right| \\
&\overset{(A.7)}{\leq} \Gamma r_{\max}\left(2\sqrt{\frac{1}{2m_t}\log\left(\frac{2}{\sigma_t}\right)} + \Omega\mathcal{S}^t \sum_{k=0}^{t} \frac{1}{\mathcal{S}^k}\sqrt{\frac{1}{2l_k}\log\left(\frac{2}{\sigma_k}\right)}\right) + \left|\hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V}^{\mathcal{L}}\right|
\end{aligned}$$

Which concludes the proof. The final recursive term $\left|\hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V}^{\mathcal{L}}\right|$ is expanded and used to give the upper bound to the full belief error in lemma A.10. □

Finally, using the results of lemmas A.8 and A.9, lemma A.10 provides an upper bound to the belief approximation error:

Lemma A.10. *The belief error* $\left|Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t)\right|$ *is bounded by the following expression* [14]*:*

$$\left|Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t)\right| \leq r_{max} \sum_{k=0}^{H-1} \gamma^k \left(2\sqrt{\frac{1}{2n_{t+k}} \log\left(\frac{2}{\sigma_{t+k}}\right)} + 2\gamma\Gamma\sqrt{\frac{1}{2m_{t+k}} \log\left(\frac{2}{\sigma_{t+k}}\right)} + (\mathcal{R} + \gamma\Gamma\Omega)\, \mathcal{S}^{t+k} \sum_{j=0}^{t+k} \frac{1}{\mathcal{S}^j} \sqrt{\frac{1}{2l_j} \log\left(\frac{2}{\sigma_j}\right)}\right)$$

Proof.

$$\left|Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t)\right| \leq \left|\mathbf{p}_{n_t}^\top \mathbf{r} - \hat{\mathbf{p}}_{n_t}^\top \mathbf{r}\right| + \gamma \left|\mathbf{p}_{o_t,m_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V}^{\mathcal{L}}\right|$$

Using lemmas A.8 and A.9:

$$\begin{aligned}
\left|Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t)\right| &\leq r_{\max} \left(2\sqrt{\frac{1}{2n_t} \log\left(\frac{2}{\sigma_t}\right)} + 2\gamma\Gamma\sqrt{\frac{1}{2m_t} \log\left(\frac{2}{\sigma_t}\right)} + (\mathcal{R} + \gamma\Gamma\Omega)\, \mathcal{S}^t \sum_{j=0}^{t} \frac{1}{\mathcal{S}^j} \sqrt{\frac{1}{2l_j} \log\left(\frac{2}{\sigma_j}\right)}\right) \\
&\quad + \gamma \left|\hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V} - \hat{\mathbf{p}}_{o_t,m_t}^\top \mathbf{V}^{\mathcal{L}}\right| \\
&\leq r_{\max} \left(2\sqrt{\frac{1}{2n_t} \log\left(\frac{2}{\sigma_t}\right)} + 2\gamma\Gamma\sqrt{\frac{1}{2m_t} \log\left(\frac{2}{\sigma_t}\right)} + (\mathcal{R} + \gamma\Gamma\Omega)\, \mathcal{S}^t \sum_{j=0}^{t} \frac{1}{\mathcal{S}^j} \sqrt{\frac{1}{2l_j} \log\left(\frac{2}{\sigma_j}\right)}\right) \\
&\quad + \gamma \sum_{o_{t+1} \in \Omega} P_{m_t}(o_{t+1} | \mathbf{b}_{l_t}, a_t) \max_{a_{t+1}} \left|Q^{\mathcal{L}}(\mathbf{b}_{t+1}, a_{t+1}) - Q^{\mathcal{L}}(\mathbf{b}_{l_{t+1}}, a_{t+1})\right|
\end{aligned}$$

Let $\boldsymbol{\mu}_{t+1}$ and a'_{t+1} be a belief-state and an action, respectively, such that $\left|Q^{\mathcal{L}}(\boldsymbol{\mu}_{t+1}, a'_{t+1}) - Q^{\mathcal{L}}(\boldsymbol{\mu}_{l_{t+1}}, a'_{t+1})\right| \geq \left|Q^{\mathcal{L}}(\boldsymbol{b}_{t+1}, a_{t+1}) - Q^{\mathcal{L}}(\boldsymbol{b}_{l_{t+1}}, a_{t+1})\right|, \forall \boldsymbol{b}_{t+1}, a_{t+1}$. Then:

$$\begin{aligned}
\left|Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t)\right| &\leq r_{\max} \left(2\sqrt{\frac{1}{2n_t} \log\left(\frac{2}{\sigma_t}\right)} + 2\gamma\Gamma\sqrt{\frac{1}{2m_t} \log\left(\frac{2}{\sigma_t}\right)} + (\mathcal{R} + \gamma\Gamma\Omega)\, \mathcal{S}^t \sum_{j=0}^{t} \frac{1}{\mathcal{S}^j} \sqrt{\frac{1}{2l_j} \log\left(\frac{2}{\sigma_j}\right)}\right) \\
&\quad + \gamma \left|Q^{\mathcal{L}}(\boldsymbol{\mu}_{t+1}, a'_{t+1}) - Q^{\mathcal{L}}(\boldsymbol{\mu}_{l_{t+1}}, a'_{t+1})\right|
\end{aligned}$$

Using the inequality above recursively, and using the stopping case from equation 23:

$$\left|Q^{\mathcal{L}}(\boldsymbol{b}_t, a_t) - Q^{\mathcal{L}}(\boldsymbol{b}_{l_t}, a_t)\right| \leq r_{\max} \sum_{k=0}^{H-1} \gamma^k \left(2\sqrt{\frac{1}{2n_{t+k}} \log\left(\frac{2}{\sigma_{t+k}}\right)} + 2\gamma\Gamma\sqrt{\frac{1}{2m_{t+k}} \log\left(\frac{2}{\sigma_{t+k}}\right)} + (\mathcal{R} + \gamma\Gamma\Omega)\, \mathcal{S}^{t+k} \sum_{j=0}^{t+k} \frac{1}{\mathcal{S}^j} \sqrt{\frac{1}{2l_j} \log\left(\frac{2}{\sigma_j}\right)}\right)$$

□

Once again, let's count the number of Hoeffding inequalities used to prove this bound. Analogously to the count done for lemma A.6, consider the *immediate* and *recursive*. Notice that, for the immediate term, $(t+1)+2$ Hoeffding inequalities are used. The first time-step in the recursive term needs $\mathcal{A}\Omega\left((t+2)+2\right)$ Hoeffding inequalities (it should hold for every action and observation, and the $t+1$ Hoeffding inequalities from the belief-error bound now become $t+2$ because the time-step has increased by one). In total, $\sum_{k=0}^{H-1} \mathcal{A}^k \Omega^k \left(t+k+2\right)$ Hoeffding inequalities are needed to prove this bound.

Finally, with the two error terms of the lookahead already bounded, the full lookahead error can be derived:

Lemma A.11. *The lookahead error* ϵ_t *has the following upper bound:*

$$\epsilon_t \leq r_{max} \sum_{k=0}^{H-1} \gamma^k \left(4\sqrt{\frac{1}{2n_{t+k}} \log\left(\frac{2}{\sigma_{t+k}}\right)} + 4\gamma\Gamma\sqrt{\frac{1}{2m_{t+k}} \log\left(\frac{2}{\sigma_{t+k}}\right)} + (\mathcal{R} + \gamma\Gamma\Omega)\, \mathcal{S}^{t+k} \sum_{j=0}^{t+k} \frac{1}{\mathcal{S}^j} \sqrt{\frac{1}{2l_j} \log\left(\frac{2}{\sigma_j}\right)}\right) + 2\gamma^H \Gamma r_{max}$$

Proof. Using lemma A.4:

$$\epsilon_t \leq \max_{a_t \in \mathcal{A}} \left(\left|Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t)\right| + \left|Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t)\right|\right) + \left|Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t^{\mathcal{L}})\right|$$

Applying lemmas A.6 and A.10 to the previous expression concludes the proof. □

The error bound provided by lemma A.11 contains terms relating to the four different types of error associated with the quantum-classical lookahead algorithm:

[14] In this bound, and throughout the remaining of this section, it is assumed that every sampling operation on the same time-step t (be it reward, observation or belief-state sampling) has the same confidence interval $1 - \sigma_t$.

- *Approximating the reward distribution*: this error term is captured by the n_{t+k} term of the sum.
- *Approximating the observation distribution*: this error term is captured by the m_{t+k} term of the sum.
- *Approximating the belief states*: this error is captured by the l_j term of the bound and increases over time due to repeated approximation of the belief state.
- *The finite horizon of the lookahead*: the lookahead tree has a fixed depth, given by the horizon H, which is finite, introducing a systematic error that can't be compensated by increasing the number of samples, captured by the $2\gamma^H \Gamma R_{\max}$ term of the bound.

Lemma A.11 provides an upper bound on the lookahead error defined in A.1. This upper bound takes into account the use of multiple sampling operations throughout the lookahead tree: reward sampling, observation sampling and belief-state sampling. Given that the bounds used for each sampling operator fail with a probability of σ_t, the bound for the whole lookahead error, naturally, also has a non-zero chance of failure, that depends on the confidence interval of each sampling operation and is yet to be defined. This is because the confidence interval $1-\sigma_t$ only holds locally for each sampling operation, and not for the whole expression presented in lemma A.11. Naturally, if none of the sampling operation bounds fail, the lookahead error bound holds. As such, the probability of failure δ_t of the lookahead error bound is at least as large as the probability that any one of the sampling bounds fails.

One way to relate the confidence interval $1-\delta_t$ of the bound of lemma A.11 to the confidence interval $1-\sigma_t$ of each sampling operation is by using theorem A.12, *Boole's inequality* [Sen92]. It states that the probability of the union of multiple events is at least as small as the sum of the probabilities of each individual event:

Theorem A.12. *(Boole's Inequality). Let $Z_1, Z_2, \ldots, Z_n$ be probabilistic events that occur with probability $P(Z_1), P(Z_2), \ldots, P(Z_n)$:*

$$P\left(\bigcup_{i=1}^{n} Z_i\right) \leq \sum_{i=1}^{n} P(Z_i)$$

In the formulation of theorem A.12, the probabilistic events Z_i could have a higher sampling error than the one provided in Hoeffding's inequality A.5. By attributing this meaning to the probabilistic events, Boole's inequality can be used to relate the sampling bound's confidence interval $1-\sigma_t$ to the confidence interval of the algorithm's upper bound presented in lemma A.11.

Lemma A.13. *The confidence interval $1-\delta_t$ of the algorithm's upper bound given in lemma A.11 and the confidence interval $1-\sigma_t$ of the Hoeffding's bound for each sampling operation obey the following inequality:*

$$\delta_t \leq \sum_{i=0}^{H-1} \mathcal{A}^i \Omega^i \left(2 + \mathcal{A}(t+i+4)\right) \sigma_{t+i}$$

Proof. Recall, from lemma A.4, that the lookahead error can bounded by the following expression:

$$\epsilon_t \leq \max_{a_t \in \mathcal{A}} \left(\left| Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t) \right| + \left| Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t) \right| \right) + \left| Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) \right|$$

The terms $\left|Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t)\right|$ and $\left|Q^\star(\mathbf{b}_{l_t}, a_t^{\mathcal{L}}) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t^{\mathcal{L}})\right|$ can both be bounded using lemma A.6, which needs $2\sum_{i=0}^{H-1} \mathcal{A}^i\Omega^i$ Hoeffding inequalities to be derived. The term $\left|Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t)\right|$ can be bounded using lemma A.10, which requires $\sum_{i=0}^{H-1} \mathcal{A}^i\Omega^i(t+i+2)$ Hoeffding inequalities. However, both the terms $\left|Q^\star(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_t, a_t)\right|$ and
$\left|Q^{\mathcal{L}}(\mathbf{b}_t, a_t) - Q^{\mathcal{L}}(\mathbf{b}_{l_t}, a_t)\right|$ are under a $\max_{a_t \in \mathcal{A}}$ operation in the lookahead error bound presented above, so their number of Hoeffding inequalities should be multiplied by a factor of $\mathcal{A}$, since these bounds should hold for any possible action. As such, the total number of Hoeffding inequalities to bound the lookahead error is given by:

$$2\sum_{i=0}^{H-1} \mathcal{A}^i\Omega^i + 2\mathcal{A}\sum_{i=0}^{H-1} \mathcal{A}^i\Omega^i + \mathcal{A}\sum_{i=0}^{H-1} \mathcal{A}^i\Omega^i(t+i+2) = \sum_{i=0}^{H-1} \mathcal{A}^i\Omega^i\left(2 + \mathcal{A}(t+i+4)\right)$$

Each term of the previous sum represents the total number of Hoeffding inequalities at that particular depth in the lookahead tree (if $i = 2$, the corresponding term represents the number of Hoeffding inequalities at depth 2). Given

that, for a depth i at time-step t, the probability of a sampling operation failing is σ_{t+1}, and using Boole's inequality in theorem A.12:

$$\delta_t \leq \sum_{i=0}^{H-1} \mathcal{A}^i \Omega^i \left(2 + \mathcal{A}\left(t + i + 4\right)\right) \sigma_{t+i}$$

□

The bounds derived in lemmas A.11 and A.13 depend on the definition of four successions:

- n_t, m_t and l_t: the successions that define how the number of samples for each sampling operation varies over time.
- σ_t: the succession defining how the confidence interval of each sampling operation varies over time.

Unfortunately, due to the cumulative nature of the belief-state error, the lookahead error grows exponentially as time progresses (it grows with $\mathcal{S}^t$, and the state size $\mathcal{S}$ of the POMDP can be extremely large by itself!). This effect could be counteracted by an appropriate succession choice for both l_t and σ_t, but this work does not try to answer this question. Instead, a much simpler case is considered: both n_t, m_t, l_t and σ_t are fixed, such that they do not change over time, as stated in assumption A.14.

Assumption A.14. *Both the number of samples n_t, m_t and l_t and the confidence interval $1 - \sigma_t$ are fixed, such that they do not change over time:*

$$n_t \equiv n, \quad m_t \equiv m, \quad l_t \equiv l, \quad \sigma_t \equiv \sigma$$

Moreover, a further simplification is assumed to reduce the degrees of freedom in the choice of the number of samples: the contribution of each sampling operation to the whole lookahead error at time-step $t = 0$ is assumed to be the same, such that the number of samples n, m and l can be related to each other according to assumption A.15:

Assumption A.15. *The contribution to the error bound of lemma A.11 of each sampling operation is considered to be the same at time-step $t = 0$, such that the number of samples can be related to each other according to the following expressions:*

$$m = \left(\gamma\Gamma\right)^2 n, \quad l = \left(\frac{1}{4}\frac{\mathcal{R} + \gamma\Gamma\Omega}{\mathcal{S} - 1}\left(\frac{\mathcal{S}}{\Gamma\left(1 - \gamma^H\right)}\frac{\left(\gamma\mathcal{S}\right)^H - 1}{\gamma\mathcal{S} - 1} - 1\right)\right)^2 n$$

Proof. According to lemma A.11 and assumption A.14, the lookahead error at time $t = 0$ has the following upper-bound:

$$\begin{aligned}\epsilon_0 &\leq r_{\max}\sqrt{\frac{1}{2}\log\left(\frac{2}{\sigma}\right)}\sum_{k=0}^{H-1}\gamma^k\left(4\sqrt{n^{-1}} + 4\gamma\Gamma\sqrt{m^{-1}} + \left(\mathcal{R} + \gamma\Gamma\Omega\right)\mathcal{S}^k\sqrt{l^{-1}}\sum_{j=0}^{k}\frac{1}{\mathcal{S}^j}\right) + 2\gamma^H\Gamma r_{\max} \\ &= S_n + S_m + S_l + 2\gamma^H\Gamma r_{\max}\end{aligned}$$

Where S_n, S_m and S_l represent the sums over the number of samples n, m and l, respectively. For their contributions to be the same, as stated in assumption A.15, all of these sums must be equal ($S_n = S_m = S_l$).

By letting $S_n = S_m$:

$$\begin{aligned}4r_{\max}\sqrt{\frac{1}{2n}\log\left(\frac{2}{\sigma}\right)}\sum_{k=0}^{H-1}\gamma^k &= 4\gamma\Gamma r_{\max}\sqrt{\frac{1}{2m}\log\left(\frac{2}{\sigma}\right)}\sum_{k=0}^{H-1}\gamma^k \\ \sqrt{n^{-1}} &= \gamma\Gamma\sqrt{m^{-1}} \\ m &= \left(\gamma\Gamma\right)^2 n\end{aligned}$$

Finally, by letting $S_n = S_l$ and making use of the geometric series [15]:

$$4r_{\max}\sqrt{\frac{1}{2n}\log\left(\frac{2}{\sigma}\right)}\sum_{k=0}^{H-1}\gamma^k = r_{\max}\sqrt{\frac{1}{2l}\log\left(\frac{2}{\sigma}\right)}\left(\mathcal{R}+\gamma\Gamma\Omega\right)\sum_{k=0}^{H-1}\gamma^k\mathcal{S}^k\sum_{j=0}^{k}\frac{1}{\mathcal{S}^j}$$

$$4\sqrt{n^{-1}}\sum_{k=0}^{H-1}\gamma^k = \sqrt{l^{-1}}\left(\mathcal{R}+\gamma\Gamma\Omega\right)\sum_{k=0}^{H-1}\gamma^k\frac{\mathcal{S}^{k+1}-1}{\mathcal{S}-1}$$

$$4\sqrt{n^{-1}}\Gamma\left(1-\gamma^H\right) = \sqrt{l^{-1}}\left(\mathcal{R}+\gamma\Gamma\Omega\right)\frac{1}{\mathcal{S}-1}\left(\mathcal{S}\sum_{k=0}^{H-1}\left(\gamma\mathcal{S}\right)^k - \sum_{k=0}^{H-1}\gamma^k\right)$$

$$4\sqrt{n^{-1}}\Gamma\left(1-\gamma^H\right) = \sqrt{l^{-1}}\left(\mathcal{R}+\gamma\Gamma\Omega\right)\frac{1}{\mathcal{S}-1}\left(\mathcal{S}\frac{\left(\gamma\mathcal{S}\right)^H-1}{\gamma\mathcal{S}-1} - \Gamma\left(1-\gamma^H\right)\right)$$

$$l = \left(\frac{1}{4}\frac{\mathcal{R}+\gamma\Gamma\Omega}{\mathcal{S}-1}\left(\frac{\mathcal{S}}{\Gamma\left(1-\gamma^H\right)}\frac{\left(\gamma\mathcal{S}\right)^H-1}{\gamma\mathcal{S}-1}-1\right)\right)^2 n$$

□

Given assumptions A.14 and A.15, the bounds in lemmas A.11 and A.13 can be re-written in the following form:

Theorem A.16. *The algorithm's error ϵ_t has the following upper bound, with a confidence interval of $1-\sigma_t$:*

$$\epsilon_t \leq r_{max}\sqrt{\frac{1}{2n}\log\left(\frac{2}{\sigma}\right)}\left(8\Gamma+4\mathcal{S}^t\right)+2\gamma\Gamma r_{max}$$

$$\delta_t \leq \sigma\left(\mathcal{A}\Omega\right)^H\left(t+\mathcal{A}\left(8+\Omega\frac{H-1}{\left(\mathcal{A}\Omega-1\right)^2}\right)\right)$$

Proof. The bound on the error ϵ_t, follows directly from lemma A.11, by substituting m and l using the expressions in assumption A.15.

As for the bound for the confidence δ_t, it follows from lemma A.13 that:

$$\begin{aligned}\delta_t &\leq \sum_{i=0}^{H-1}\mathcal{A}^i\Omega^i\left(2+\mathcal{A}\left(t+i+4\right)\right)\sigma_{t+i}\\ &= \sigma\sum_{i=0}^{H-1}\mathcal{A}^i\Omega^i\left(2+\mathcal{A}\left(t+4\right)+i\mathcal{A}\right)\end{aligned}$$

By using the finite geometric series on the $2+\mathcal{A}\left(t+4\right)$ multiplicative term in the sum:

$$\begin{aligned}\delta_t &\leq \sigma\left(\mathcal{A}\sum_{i=0}^{H-1}i\mathcal{A}^i\Omega^i + \left(2+\mathcal{A}\left(t+4\right)\right)\frac{\left(\mathcal{A}\Omega\right)^H-1}{\mathcal{A}\Omega-1}\right)\\ &\leq \sigma\left(\mathcal{A}\sum_{i=0}^{H-1}i\mathcal{A}^i\Omega^i + \left(2+\mathcal{A}\left(t+4\right)\right)\left(\mathcal{A}\Omega\right)^H\right)\end{aligned}$$

[15] $\sum_{i=0}^{H-1} z^i = \frac{z^H-1}{z-1}$

Analogously applying a variation of the finite geometric series [16] to the sum $\sum_{i=0}^{H-1} i\mathcal{A}^i\Omega^i$, the bound becomes:

$$\begin{aligned}
\delta_t &\leq \sigma\left(\mathcal{A}\frac{(H-1)(\mathcal{A}\Omega)^{H+1} - H(\mathcal{A}\Omega)^H + \mathcal{A}\Omega}{(\mathcal{A}\Omega-1)^2} + (2+\mathcal{A}(t+4))(\mathcal{A}\Omega)^H\right)\\
&\leq \sigma\left(\mathcal{A}\frac{(H-1)(\mathcal{A}\Omega)^{H+1}}{(\mathcal{A}\Omega-1)^2} + (2+\mathcal{A}(t+4))(\mathcal{A}\Omega)^H\right)\\
&\leq \sigma\left(\mathcal{A}^2\Omega\right)^H\left(\mathcal{A}\Omega\frac{(H-1)}{(\mathcal{A}\Omega-1)^2} + \mathcal{A}(t+4) + 2\right)\\
&\leq \sigma(\mathcal{A}\Omega)^H\left(2+\mathcal{A}\left(t+4+\mathcal{A}\Omega\frac{H-1}{(\mathcal{A}\Omega-1)^2}\right)\right)
\end{aligned}$$

□

As can be seen in theorem A.16, both the bounds for ϵ_t and δ_t grow over time. Suppose that the designed algorithm accepts three arguments as input: the maximum error ϵ, the minimum confidence interval $1-\delta$ and the stopping time T for the decision making algorithm. In this scenario, both ϵ_t and δ_t will have a maximum bound for $t = T$. To ensure the error ϵ and the confidence interval $1-\delta$, theorem A.16 can be re-expressed to derive a lower-bound for the number of samples and an upper bound for σ (the probability of failure for the sampling bounds):

$$\begin{aligned}
n &\geq \frac{1}{2}\log\left(\frac{2}{\sigma}\right)\left(\frac{r_{\max}}{\epsilon - 2\gamma^H\Gamma r_{\max}}\left(8\Gamma + 4\mathcal{S}^T\right)\right)^2\\
\sigma &\leq \delta(\mathcal{A}\Omega)^{-H}\left(2+\mathcal{A}\left(T+4+\mathcal{A}\Omega\frac{H-1}{(\mathcal{A}\Omega-1)^2}\right)\right)^{-1}
\end{aligned} \tag{25}$$

The above conditions, however, are not enough to ensure the error ϵ. This is because no matter how many samples one extracts for approximating the probability distributions, the number of samples can't compensate for the error of performing a finite lookahead if the chosen horizon H is too shallow [17]. As such, it should also be assured that the lookahead horizon H is large enough to make the desired error ϵ is attainable. This condition is met when $2\gamma^H\Gamma r_{\max} < \epsilon$, such that:

$$H > \log_\gamma\left(\frac{\epsilon}{2\Gamma r_{\max}}\right) \tag{26}$$

The three bounds presented in equations 25 and 26 present a summary of the whole sample complexity analysis of the lookahead algorithm. They are captured in theorem A.17, the main theorem of this analisys:

Theorem A.17. *Consider a stopping time T, horizon H and assumptions A.14 and A.15 for the lookahead algorithm. A near optimal action is guaranteed to be taken at each time-step of the decision making process with a regret of at most ϵ and confidence interval of at least $1-\delta$ if all of the following conditions are met:*

$$\begin{aligned}
n &\geq \frac{1}{2}\log\left(\frac{2}{\sigma}\right)\left(\frac{r_{max}}{\epsilon - 2\gamma^H\Gamma r_{max}}\left(8\Gamma + 4\mathcal{S}^T\right)\right)^2\\
\sigma &\leq \delta(\mathcal{A}\Omega)^{-H}\left(2+\mathcal{A}\left(T+4+\mathcal{A}\Omega\frac{H-1}{(\mathcal{A}\Omega-1)^2}\right)\right)^{-1}\\
H &> \log_\gamma\left(\frac{\epsilon}{2\Gamma r_{max}}\right)
\end{aligned}$$

B Computational Complexity Analysis

To calculate the computational complexity of the lookahead algorithm, both the classical and quantum-classical versions, one must multiply the total number of samples needed by the computational complexity necessary to compute

[16] $\sum_{i=0}^{H-1} iz^i = \frac{(H-1)z^{H+1} - Hz^H + z}{(z-1)^2}$

[17] Using a finite lookahead introduces an error of $2\gamma\Gamma r_{\max}$, as in theorem A.16.

each sample. Given that the computational complexity for each sample has already been defined in section 4, the only task left is to define the total number of samples that need to be extracted.

In order to do so, it is important to recall that there are three types of sampling operations that the algorithm performs:

- *Reward sampling*: this sampling operation is performed at every observation node and requires n samples.
- *Observation sampling*: performed at every observation node and requires m samples.
- *Belief sampling*: performed at every belief node and requires l samples.

As such, the total number of samples N_s can be defined in terms of the number of belief nodes N_b and the number of observation nodes N_o as:

$$N_s = lN_b + (n+m)\,N_o$$

Where the number of belief nodes can be calculated as:

$$N_b = \sum_{i=0}^{H-1} (\mathcal{A}\Omega)^i = \frac{(\mathcal{A}\Omega)^H - 1}{\mathcal{A}\Omega - 1}$$

And the number of observation nodes as:

$$N_o = \mathcal{A} \sum_{i=0}^{H-1} (\mathcal{A}\Omega)^i = \mathcal{A}\frac{(\mathcal{A}\Omega)^H - 1}{\mathcal{A}\Omega - 1}$$

Such that the total number of samples can be re-expressed as:

$$N_s = \frac{(\mathcal{A}\Omega)^H - 1}{\mathcal{A}\Omega - 1}\,(l + \mathcal{A}\,(n+m))$$

Let C_n, C_m and C_l represent the computational complexity of extracting each of the three types of samples considered. In asymptotic terms, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$, Ω, H and T are all assumed to be very large, the computational complexity of the algorithm can be expressed as:

$$\mathcal{O}\left(\mathcal{A}^{H-1}\Omega^{H-1}\left(lC_l + \mathcal{A}\left(nC_n + mC_m\right)\right)\right)$$

Recalling the assumption on the relation between the number of samples n, m and l in assumption A.15, this complexity becomes:

$$\begin{aligned}
&\mathcal{O}\left(n\mathcal{A}^{H-1}\Omega^{H-1}\left(\left(\frac{1}{4}\frac{\mathcal{R}+\gamma\Gamma\Omega}{\mathcal{S}-1}\left(\frac{\mathcal{S}}{\Gamma\left(1-\gamma^H\right)}\frac{(\gamma\mathcal{S})^H-1}{\gamma\mathcal{S}-1}-1\right)\right)^2 C_l + \mathcal{A}\left(C_n + \gamma^2\Gamma^2 C_m\right)\right)\right)\\
&= \mathcal{O}\left(n\mathcal{A}^{H-1}\Omega^{H-1}\left(\mathcal{A}\left(C_n + C_m\right) + \left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2 C_l\right)\right)
\end{aligned}$$

Using the upper bound on n defined in theorem A.17, this computational complexity can be rewritten in terms of the lookahead error:

$$\mathcal{O}\left(\mathcal{A}^{H-1}\Omega^{H-1}\log\left(\frac{1}{\sigma}\right)\left(\frac{\mathcal{S}^T}{\epsilon}\right)^2\left(\mathcal{A}\left(C_n + C_m\right) + \left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2 C_l\right)\right)$$

Before moving on to expanding the computational complexities C_n, C_m and C_l, the local confidence interval $1-\sigma$ of each sampling operation can also be substituted in the expression using theorem A.17 in order to re-write the complexity according to the whole algorithm's confidence interval $1-\delta$:

$$\mathcal{O}\left(\mathcal{A}^{H-1}\Omega^{H-1}\log\left(\left(T + \frac{H}{\mathcal{A}\Omega}\right)\frac{\mathcal{A}^{H+1}\Omega^H}{\delta}\right)\left(\frac{\mathcal{S}^T}{\epsilon}\right)^2\left(\mathcal{A}\left(C_n + C_m\right) + \left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2 C_l\right)\right)$$

To substitute the computational complexities C_n, C_m and C_l, a distinction has to be made for the classical and quantum-classical algorithms, since they use different inference algorithms to extract probability distributions.

In the classical case, rejection sampling is used for performing inference. As such, all of these computational complexities are of order $\mathcal{O}\left(NMP(e)^{-1}\right)$, as according to table 1. The evidences e, however, vary for each sampling operation along the tree:

- Let $c_n \equiv P(e)^{-1}$ for reward sampling. It represents the average inverse probability, for reward sampling, along every observation node. Since the reward distribution $P\left(r_{t+1}|\mathbf{b}_t, a_t\right)$ is conditioned on the belief-state and action, c_n is the average of every inverse probability $P\left(\boldsymbol{b}_t, a_t\right)^{-1}$ for every action and every belief-state along the lookahead tree.
- Let $c_m \equiv P(e)^{-1}$ for observation sampling. It represents the average inverse probability, for observation sampling, along every observation node. The observation distribution $P\left(o_{t+1}|\boldsymbol{b}_t, a_t\right)$ is also conditioned on the belief-state and action, and therefore $c_m = c_n$.
- Let $c_l \equiv P(e)^{-1}$ for belief-state sampling. It represents the average inverse probability, for belief-state sampling, along every belief node. The belief-state distribution $P\left(s_{t+1}|o_{t+1}, \boldsymbol{b}_t, a_t\right)$ is conditioned on the previous belief-state, action and current observation, hence c_l is the average of every inverse probability $P\left(o_{t+1}, \boldsymbol{b}_t, a_t\right)^{-1}$.

For the quantum-classical case, rejection sampling is still used for reward and observation sampling, and therefore the discussion above is still valid, but quantum rejection sampling is applied for belief-state sampling. Therefore, in the quantum-classical case, the complexity for C_l is of the order $\mathcal{O}\left(N2^M q_l\right)$, where $q_l \equiv P(e)^{-\frac{1}{2}}$. In this scenario, c_l is no longer adequate to represent $P(e)^{-\frac{1}{2}}$, because the latter is an average of the inverse square rooted probabilities $P\left(o_{t+1}, \boldsymbol{b}_t, a_t\right)^{-\frac{1}{2}}$ and not of the inverse probabilities $P\left(o_{t+1}, \boldsymbol{b}_t, a_t\right)^{-1}$.

With this analysis, theorem B.1 characterizes the computational complexity of the classical lookahead algorithm:

Theorem B.1. *The computational complexity of the classical lookahead algorithm is given by:*

$$\mathcal{O}\left(NM\mathcal{A}^{H-1}\Omega^{H-1}\left(\mathcal{A}c_n + \left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2 c_l\right)\left(\frac{\mathcal{S}^T}{\epsilon}\right)^2 \log\left(\left(T + \frac{H}{\mathcal{A}\Omega}\right)\frac{\mathcal{A}^{H+1}\Omega^H}{\delta}\right)\right)$$

Theorem B.2 is analogous to the previous theorem, but characterizes the computational complexity of the quantum-classical lookahead algorithm:

Theorem B.2. *The computational complexity of the quantum-classical lookahead algorithm is given by:*

$$\mathcal{O}\left(N\mathcal{A}^{H-1}\Omega^{H-1}\left(\mathcal{A}Mc_n + \left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2 2^M q_l\right)\left(\frac{\mathcal{S}^T}{\epsilon}\right)^2 \log\left(\left(T + \frac{H}{\mathcal{A}\Omega}\right)\frac{\mathcal{A}^{H+1}\Omega^H}{\delta}\right)\right)$$

Finally, corollaries B.3 and B.4 characterize the classical and quantum algorithm's complexity, respectively, in a scenario where the quantum speedup is the most prevalent. This happens when the complexity of computing the belief-state distributions has a much greater impact than the reward and observation distribution sampling operations for the overall computational complexity, making their contribution to the complexity of the algorithm negligible.

Corollary B.3. *Under the assumption that* $\frac{c_n}{q_l} \ll \frac{\left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2}{\mathcal{A}}$*, the computational complexity of the classical lookahead algorithm of theorem B.1 can be re-written as:*

$$\mathcal{O}\left(NM\mathcal{A}^{H-1}\Omega^{H-1}\left(\left(\mathcal{R}+\Omega\right)\frac{\mathcal{S}^{T+H-1}}{\epsilon}\right)^2 c_l \log\left(\left(T + \frac{H}{\mathcal{A}\Omega}\right)\frac{\mathcal{A}^{H+1}\Omega^H}{\delta}\right)\right)$$

Proof. Given that $q_l \leq c_l$, which implies that $\frac{c_n}{c_l} \leq \frac{c_n}{q_l}$, then $\frac{c_n}{q_l} \ll \frac{\left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2}{\mathcal{A}} \Rightarrow \frac{c_n}{c_l} \ll \frac{\left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2}{\mathcal{A}}$. Thus, the result of this corollary follows directly from theorem B.1. □

Corollary B.4. *Under the assumption that* $\frac{c_n}{q_l} \ll \frac{\left(\left(\mathcal{R}+\Omega\right)\mathcal{S}^{H-1}\right)^2}{\mathcal{A}}$*, and that* M *is small enough such that* M *and* 2^M *have about the same order of magnitude, the computational complexity of the quantum-classical lookahead algorithm of theorem B.2 can be re-written as:*

$$\mathcal{O}\left(N2^M\mathcal{A}^{H-1}\Omega^{H-1}\left(\left(\mathcal{R}+\Omega\right)\frac{\mathcal{S}^{T+H-1}}{\epsilon}\right)^2 q_l \log\left(\left(T + \frac{H}{\mathcal{A}\Omega}\right)\frac{\mathcal{A}^{H+1}\Omega^H}{\delta}\right)\right)$$