Implementation of Markov Decision Processes into quantum algorithms for reinforcement learning

Manuel Pegalajar

University of Granada

manupc@decsai.ugr.es

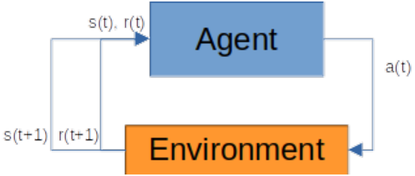
1st Workshop on Quantum Artificial Intelligence.
Naples, 27-28 July 2023

Content

Overview of Reinforcement Learning

Reinforcement learning

Scope: Online learning by interaction with an unknown environment.

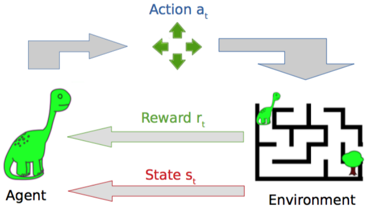


Sketch of Reinforcement Learning dynamics:

1. Initially, at any time stamp t, the environment is in state s(t).
2. The agent perceives an observation of s(t) and select/performs an action a(t) in the environment.
3. The environment changes its internal state from s(t) to s(t+1) according to state dynamics and the action a(t) received.
4. The environment returns an observation (projection) of s(t+1) to the agent, together with a reward r(t).
5. Update t= t+1 and start again.

Reinforcement learning

Experience: Tuple containing the information of an agent-environment interaction, i.e. $(s(t), a(t), r(t), s(t+1))$.
Trajectory (τ): Sequence of experiences starting at a given time t, i.e. $(s(t), a(t), r(t), s(t+1), a(t+1), r(t+1), s(t+2), \ldots)$



Goal in Reinforcement Learning: To learn the agent's action selection policy $\pi(a|s)$ that maximizes the total accumulated reward (**return**) of any given trajectory:

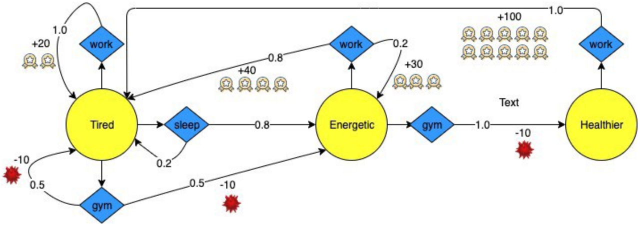
$$R(\tau) = \sum_{t=\tau}^{\infty} r(t); \ \ or \ \ R(t) = r(t) + \gamma R(t+1)$$

Relevant concepts: The environment is unknown, but stochastic. It can be modelled using a Markov Decision Process **(MDP)** $< S, A, P, R >$:

- S is the state space.
- A is the available action set to the agent.
- P is the transition probability function containing $p(s'|s, a)$, i.e. the probability to evolve to state s' from state s by executing action a.
- R is the reward function containing scalar values to obtain a reward $r(s, a, s')$ being at state s, then executing action a and evolving to state s'.

An example MDP with

- $S = \{Tired, Energetic, Healthier\}$
- $A = \{Work, Gym, Sleep\}$



Optimal policy (deterministic):

- $\pi(Tired) = Sleep$
- $\pi(Energetic) = Gym$
- $\pi(Healthier) = Work$

The expected return of $V^*(Tired) = 1376.41$ with $\gamma = 0.99$.

Reinforcement learning

Relevant concepts: Consider that at time t the environment is at state s. The agent selects action a and then the environment evolves to state s'. Then:

- $V^\pi(s)$ is the value of a given state s. It is the expected return that will be obtained starting from state s with policy π.

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')]$$

- $Q^\pi(s,a)$ is the value of the state-action pair (s,a). It is the expected return that will be obtained starting from state s, then executing action a, and following policy π.

$$Q^\pi(s,a) = \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s',a')]$$

Some (old-ish?) classic methods to solve Reinforcement Learning:

- **Value Iteration**: Iterative algorithm. It computes the optimal policy based on Dynamic Programming. Requires prior full knowledge about the underlying MDP.
- **Q-Learning**: Iterative algorithm. It computes an approximation of the Q function $Q(s, a)$ for all pairs (s, a). No prior knowledge about the MDP is required (more realistic case).

More recent: Deep Reinforcement Learning (several approaches), etc.

Quantum Reinforcement Learning

Quantum Reinforcement Learning

Different ways to include Quantum Computing into Reinforcement Learning scenarios:

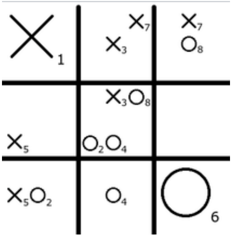
		Environment	
		Classical	Quantum
Agent	Classical	CC	CQ
	Quantum	QC	QQ

Most of the Literature: **QC** (Quantum agent, classical environment).
Opportunities: What is a quantum environment and how it can be modelled?

Quantum Reinforcement Learning

The lack of definition of what a quantum environment could be, and its model, is required to extend classical Reinforcement Learning to a quantum environment. Some research lines:

- **Modelling partial observability**: Partially Observable MDP (POMDP) encompasses a set of (classic) techniques to deal with situations where the agent cannot perceive the environment state s completely. A quantum environment is inherently partially observable.
- **Quantum environment models models**, as for instance the Quantum Tic-Tac-Toe.



Quantum Reinforcement Learning

Our mid-term goal:

To create a methodology to build quantum environments able to be run into quantum computers.

Our proposal in this work:

First attempt to translate a classic MDP implementation to a quantum program.

MDP Quantum Implementation

MDP Quantum implementation

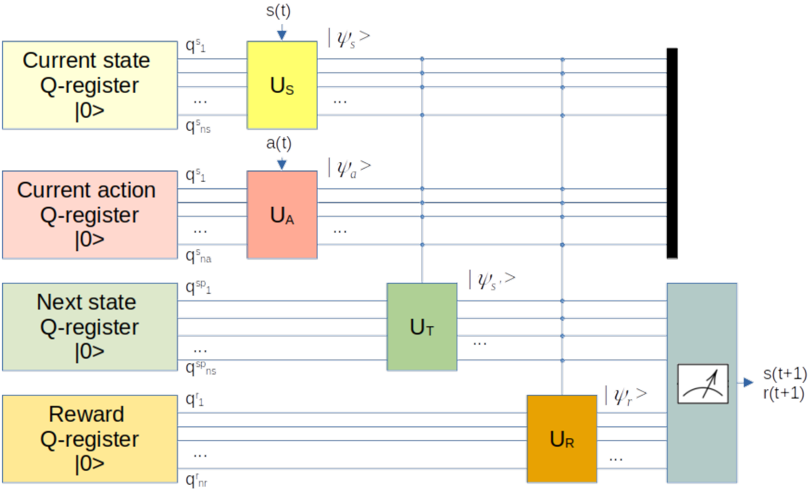
Hypothesis of our approach:

- The state-space is discrete and finite.
- The action set is discrete and finite.
- The rewards can be mapped to a finite discrete set.

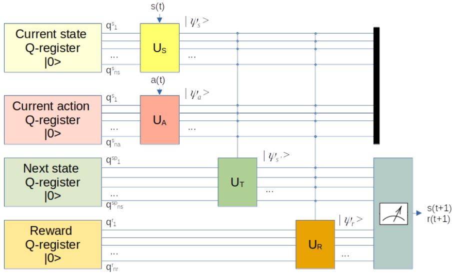
General idea: Four different quantum registers are used to store:

- The current known state s.
- The current known agent action a.
- The next state s' (output)
- The reward r (output)

General scheme:

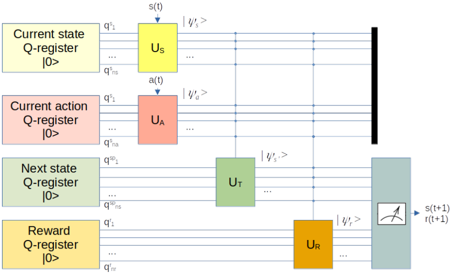


MDP Quantum implementation



- U_s: In charge of encoding state $s(t)$ (basis encoding, Q-Sample...)
- U_a: In charge of encoding agent action $a(t)$ (basis encoding)
- U_T: In charge of implementing the transition function conditioned to $s(t)$ and $a(t)$ with Q-Sample.
- U_R: In charge of implementing the reward function conditioned to $s(t), s(t+1)$ and $a(t)$, using entanglement (for instance, CNOT).

MDP Quantum implementation

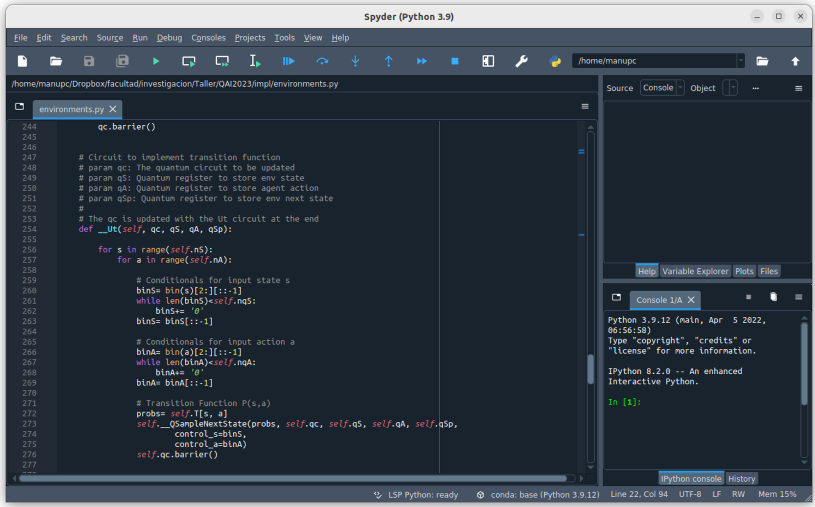


Steps:

- $|\psi_s 000\rangle = U_s |0000\rangle$
- $|\psi_s \psi_a 00\rangle = U_a |\psi_s 000\rangle$
- $|\psi_s \psi_a \psi_{s'}\rangle = U_T |\psi_s \psi_a 00\rangle$
- $|\psi_s \psi_a \psi_{s'} \psi_r\rangle = U_R |\psi_s \psi_a \psi_{s'} 0\rangle$
- Measure registers for qubits of next state and reward.

MDP Quantum implementation

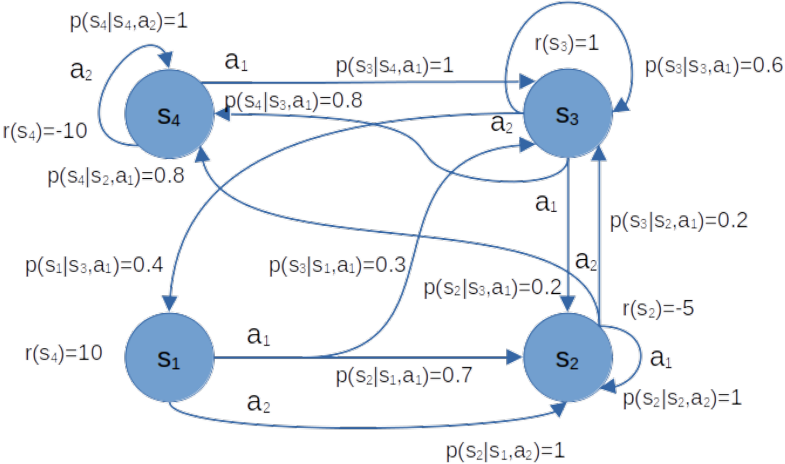
Free implementation provided at:
https://github.com/manupc/MDPQuantum



Sample proof of concept

Proof of concept

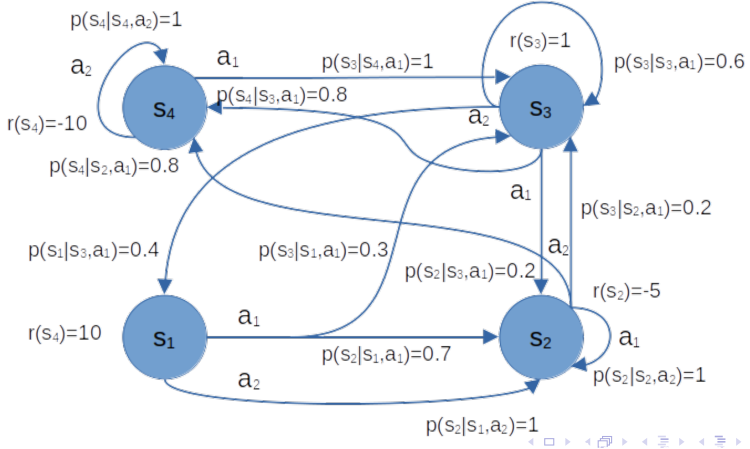
We validate our proposal over a toy MDP as a proof of concept:



Proof of concept

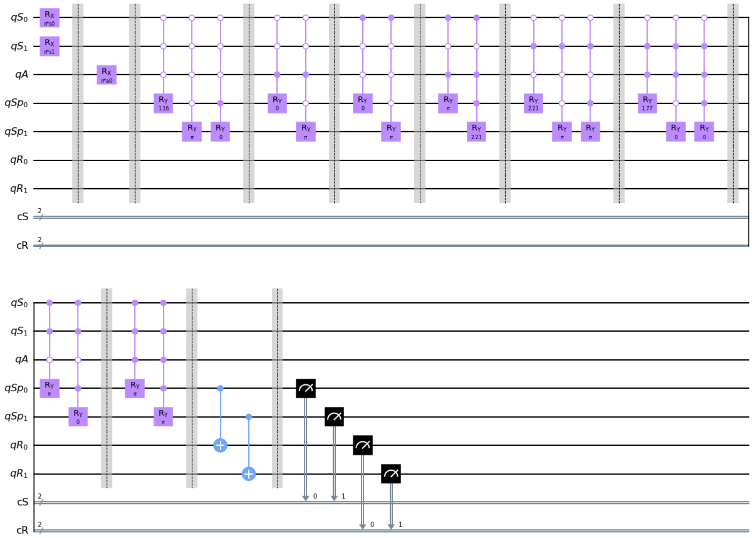
Elements:

- $S = \{s_1, s_2, s_3, s_4\}$ encoded as $S = \{|00\rangle^S, |01\rangle^S, |10\rangle^S, |11\rangle^S\}$
- $A = \{a_1, a_2\}$ encoded as $A = \{|0\rangle^A, |1\rangle^A\}$
- $r(\cdot, \cdot, s') \in \{1, , -5, -10, 10\}$ encoded as $\{|00\rangle^R, |01\rangle^R, |10\rangle^R, |11\rangle^R\}$



Proof of concept

The circuit encoding the toy MDP:



Proof of concept

Experiments:

- **Value Iteration**: Classic Agent - Classic Environment
  - $\gamma = 0.99$.
- **Q-Learning**: Classic Agent - Classic/Quantum Environment
  - $\gamma = 0.99$.
  - Iterations: 200.
  - Exploration: ϵ-greedy with linear decay from 0.5 to 0.001.
  - Learning rate $\alpha = 0.2$.

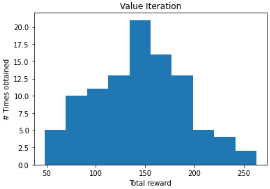
Simulation environment: Qiskit with noise-free QASM simulator.

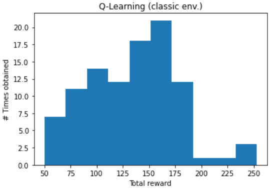
Proof of concept

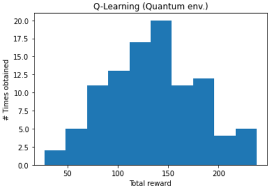
- The optimal policy was obtained in all cases:
  - $\pi(s_1) = a_1; \pi(s_2) = a_2; \pi(s_3) = a_2; \pi(s_4) = a_1$
- Time to learn:
  - Classic environment, Value iteration: 0.02 s.
  - Classic environment, Q-Learning: 0.02 s.
  - Quantum environment, Q-Learning: 226.68 sec.
- Average return in test after 100 environment executions:
  - Classic environment, Value iteration: 144.03.
  - Classic environment, Q-Learning: 134.25.
  - Quantum environment, Q-Learning: 136.54.

Proof of concept

Histograms of Returns after 100 test environment executions:







Conclusions and future work

Conclusions and future work

Our contributions:

- We have developed a method to implement Markov Decision Processes in Quantum programs.
- We have tested experimentally that the implementation corresponds to the same MDP implemented classically.
- A first step to dig into the construction of truw Quantum Environments for Quantum Reinforcement Learning.

TO-DO list (short-term):

- To adapt the methodology to partially observable scenarios.
- To integrate Quantum Environments with Quantum Agents.
- To study other types of models able to create Quantum Environments for Reinforcement Learning.

The End