Implementation of Markov Decision Processes into quantum algorithms for reinforcement learning

1st M.P. Cuéllar
Dept. Computer Science and Artificial Intelligence
University of Granada
Granada, Spain
manupc@decsai.ugr.es

2nd M.C. Pegalajar
Dept. Computer Science and Artificial Intelligence
University of Granada
Granada, Spain
mcarmen@decsai.ugr.es

3rd L.G.B. Ruiz
Dept. Software Engineering
University of Granada
Granada, Spain
bacaruiz@ugr.es

4th G. Navarro
Dept. Computer Science and Artificial Intelligence
University of Granada
Granada, Spain
gnavarro@ugr.es

5th C. Cano
Dept. Computer Science and Artificial Intelligence
University of Granada
Granada, Spain
ccano@decsai.ugr.es

6th L. Servadei
School of Computation, Information and Technology
Technical University of Munich
Munich, Germany
lorenzo.servadei@tum.de

***Abstract*—In this work, we propose a methodology to implement classic Markov Decision Processes in a Quantum Computing paradigm, as a first step to achieve systems running Quantum Reinforcement Learning where both agent and environment are expressed as quantum programs. To do so, we analyze the interaction cycle between the agent and the environment in classic reinforcement learning and create a method to map a Markov Decision Process with discrete state space, action set, and rewards, into a quantum program.**

***Index Terms*—Reinforcement Learning, Quantum Computing, Quantum Machine Learning, Markov Decision Process**

I. Introduction

Classic Reinforcement Learning (RL) [1] is a type of Machine Learning where an intelligent agent learns by interaction with an unknown environment. A cycle of a RL agent/environment interaction (see Figure 1) is as follows: At each time step t the environment is in a state $s(t)$ which is fully observable by the agent. Then, the agent selects an action $a(t)$ over an available action set and performs the action over the environment, which evolves from state $s(t)$ to $s(t+1)$ and returns a reward scalar value $r(t+1)$ to the agent as a feedback of its performance. The goal in RL is to learn a policy $\pi(a|s)$ to select the best action a at every state s that maximizes the long term accumulated reward, i.e. $\sum_t \gamma r(t)$, where $\gamma \in [0, 1]$ is a *discount factor*. Applications of RL are varied and some are described in [1].

This work was supported by the project QUANERGY (Ref. TED2021-129360B-I00), Ecological and Digital Transition R&D projects call 2022, Government of Spain.

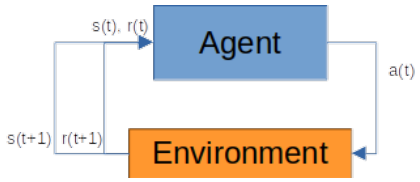

Fig. 1. Agent/environment interaction in a Reinforcement Learning cycle.

Classic RL is built upon the first-order Markov assumption, so that the environment can be modelled with a Markov Decision Process (MDP) [2]. An MDP is defined as a tuple $\langle S, A, P, R \rangle$, where $S = \{s_1, s_2, ..., s_{|S|}\}$ is a set of states, $A = \{a_1, a_2, ..., a_{|A|}\}$ is a set of actions, $P : S \times A \to \mathcal{P}(S)$ is a probabilistic transition function where $P(s_i, a_k, s_j)$ is the probability to evolve from state s_i to state s_j after executing action a_k, and $R : S \times A \times S \to \mathbb{R}$ is a reward function where $R(s_i, a_k, s_j)$ is a scalar value containing the reward obtained after reaching state s_j by means of executing action a_k in state s_i.

Quantum Reinforcement Learning (QRL) [3] attempts to adapt classic Reinforcement Learning methods, or to develop new techniques, for Quantum Computing (QC). As in Quantum Machine Learning [4], four different scenarios combining quantum and classical cases are studied, that arise from the possibilities of a combination of classical/quantum agents operating in classical/quantum environments. Nowadays, the QRL literature has been mostly focused on the case where the agent is implemented as a quantum program (usually by

means of Variational Quantum Circuits, VQC) working over a classical environment as in [3], [5]–[8] to mention just a few, since real quantum environments are difficult to find. However, in the last few years, there have been a few proposals to create quantum environments such as the quantum Tic-Tac-Toe [9].

In this work, we propose a method to migrate a classic MDP to a QC paradigm, as a first step to create true quantum environment simulators that enable the execution of quantum agents in quantum environments. To do so, we first analyze the components of the underlying MDP that rules the environment evolution according to an agent action, and propose a methodology to create quantum programs that emulate the agent/environment interaction cycle.

This work is structured as follows: Section II describes the approach. After that, Section III show an example of the method as a proof of concept, and Section IV concludes.

II. Implementation of the MDP cycle in Quantum Computing scenarios

Our starting hypothesis is that all the state space S, action space A, and possible rewards in R, can be expressed as discrete sets. In this context, our goal is to find a deterministic policy $\pi(s)$ that returns the best action $a = \pi(s)$ for each environment state. The proposed general method to implement the RL cycle in QC, containing the underlying MDP, can be devised as follows (Figure 2):

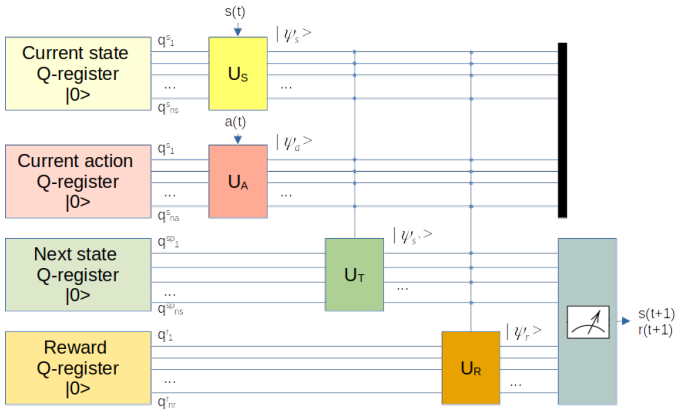

Fig. 2. Scheme of the implementation of an MDP with a quantum program.

- At time t, the environment is at a given state $s(t) = s_i \in S$, provided as an input. Therefore, a quantum state preparation method is required to encode the cycle's initial state s_i into its corresponding quantum state $|\psi_s\rangle$ in a quantum register with n_s qubits. The operator to create $|\psi_s\rangle$ from $|0\rangle$ is U_S, i.e. $|\psi_s\rangle = U_S|0\rangle$.
- A deterministic action $a(t) = a_k \in A$, selected by either a classical or quantum agent, is provided as input and applied over the environment. The action a_k is encoded into the state of a quantum register with n_a qubits $|\psi_a\rangle$ using the unitary transformation U_A so that $|\psi_a\rangle|\psi_s\rangle = U_A|0\rangle|\psi_s\rangle$.
- Once the input state s_i and action a_k are encoded into quantum states $|\psi_s\rangle, |\psi_a\rangle$, the transition function $P(s_i, a_k)$ is executed to evolve the environment from the known state $s(t) = s_i$ to a new state $s(t+1) = s_j$ with probability $P(s_i, a_k, s_j)$. To do so, a target quantum register to store the next state, $|\psi_{s'}\rangle$, is required. A controlled unitary transformation U_T is in charge of calculating the superposition of target environment states as $|\psi_{s'}\rangle$, so that $|\psi_{s'}\rangle|\psi_a\rangle|\psi_s\rangle = U_T|0\rangle|\psi_a\rangle|\psi_s\rangle$.
- The reward $r(t+1) = R(s_i, a_k, s_j)$ is finally calculated. We could think of the reward function as a controlled quantum operator U_R that evolves a quantum register $|0\rangle$ of n_r qubits allocated for rewards to a quantum state $|\psi_r\rangle$ encoding $r(t+1)$. Formally, it is written as $|\psi_r\rangle|\psi_{s'}\rangle|\psi_a\rangle|\psi_s\rangle = U_R|0\rangle|\psi_{s'}\rangle|\psi_a\rangle|\psi_s\rangle$.
- Return of $s(t+1), r(t+1)$ to the agent. In the case the agent is classical, this can be performed with measurement operators over the target state quantum register $|\psi_{s'}\rangle$ and reward register $|\psi_r\rangle$.

A. Current state and action encoding

To achieve an optimal use of qubit resources using a dense encoding technique, all environment states, actions and rewards are stored in the amplitudes of their corresponding quantum registers. This means that the size of n_s qubits used for representing the environment states, n_a qubits to represent the available actions, and n_r qubits to represent rewards can hold $2^{n_s}, 2^{n_a}$ and 2^{n_r} different possible values, respectively.

Since environment states are fully observable in a MDP, the quantum state representing $|\psi_s\rangle$ contains an amplitude whose squared value equals 1.0, and therefore a simple basis encoding technique [4] containing X gates can be used as the encoding operator U_S. The same situation occurs for actions and U_A. Assuming a mapping from environment states, $s_i \in S$ to basis quantum states, i.e. $s_i \mapsto |i\rangle$, and also for actions $a_k \mapsto |k\rangle$, the unitary operators U_S and U_A can be generalized and implemented using parameterized circuits containing $R_x(\theta)$ gates, where $\theta \in \{0, \pi\}$ depending on the binary representation of s_i, a_k.

B. Implementation details of transition probabilities

The unitary transformation U_T is in charge of calculating the superposition of target states $s_j \in S$ into $|\psi_{s'}\rangle$, so that the register of the next state $|\psi_{s'}\rangle$ contains the probabilities of measurement $P(s_i, a_k, s_j)$. Thus, U_T can be implemented as a sequence of instances of the Q-Sample encoding method [4] controlled by the values of the quantum registers $|\psi_a\rangle|\psi_s\rangle$. Section III provides an example of such instances.

C. Implementation details of reward computation

Similarly to states and actions, we assume a mapping from the set of rewards $r_l \in R$ to basis states as $r_l \mapsto |l\rangle$ in the reward quantum register. After all values s_i, a_k, s_j are known, the reward function $R(s_i, a_k, s_j)$ is implemented into U_R as a controlled operation that sets a probability amplitude of a basis vector in the reward quantum register to value 1.0. This is implemented in a circuit as a sequence of multiple controlled NOT gates that set the correct amplitude. The idea behind

this operation is to entangle the reward quantum state with $|\psi_{s'}\rangle |\psi_a\rangle |\psi_s\rangle$ so that the correct reward value is obtained once $|\psi_{s'}\rangle$ collapses after measurement.

III. Proof of Concept and Implementation

Due to space limitations, we limit the experimentation in this work to show the RL cycle mapping from the MDP in Figure 3 to a quantum program. We also performed a toy example experiment to train a classic agent using the Q-Learning procedure with $\gamma = 0.99$ in a maximum number of $T = 200$ steps and learning rate $\alpha = 0.2$ to test convergence. The source code for this experimentation is freely available at *https://github.com/manupc/MDPQuantum*.

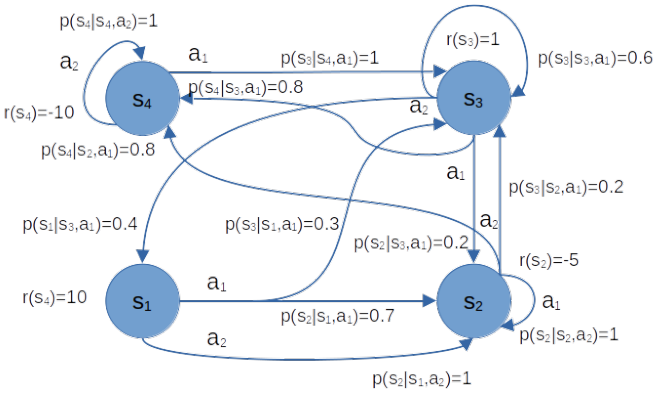

Fig. 3. Example MDP.

The MDP in Figure 3 contains four states $S = \{s_1, s_2, s_3, s_4\}$, two possible actions for each state $A = \{a_1, a_2\}$, and a discrete set with four rewards that depend on the final transition state only $\{r(s_1) = 10, r(s_2) = -5, r(s_3) = 1, r(s_4) = -10\}$. Therefore, the number of qubits required to represent the states and rewards is $n_s = n_r = 2$. The proposed mapping from states and rewards to basis vectors is $s_1 \mapsto |00\rangle, s_2 \mapsto |01\rangle, s_3 \mapsto |10\rangle, s_4 \mapsto |11\rangle$ and $r(\cdot,\cdot,s_1) \mapsto |00\rangle, r(\cdot,\cdot,s_2) \mapsto |01\rangle, r(\cdot,\cdot,s_3) \mapsto |10\rangle, r(\cdot,\cdot,s_4) \mapsto |11\rangle$, respectively. It is assumed that the initial state to execute the environment is s_1. With respect to actions, a number of $n_a = 1$ qubit is required under the mapping $a_1 \mapsto |0\rangle, a_2 \mapsto |1\rangle$.

The optimal deterministic policy was obtained using the Value Iteration method in classic RL, obtaining the policy $\pi(s_1) = a_1, \pi(s_2) = a_2, \pi(s_3) = a_2, \pi(s_4) = a_1$. Running the value iteration over the classical environment required 0.02 sec. on a desktop computer Intel(R) Core(TM) i5-9600K CPU at 3.70GHz with 32GB RAM. On the other hand, the Q-Learning algorithm over the classical environment took 0.012 sec. and obtained the optimal policy.

The quantum MDP was implemented in IBM's Qiskit with the QASM noise-free simulator. An example of a sub-circuit containing the transition from s_3 and action a_2 to states s_0, s_3 with probabilities 0.4 and 0.6, respectively, is depicted in Figure 4. The Q-Learning method was applied to learn the optimal policy, using a computational time of 226.68 sec. under ideal conditions.

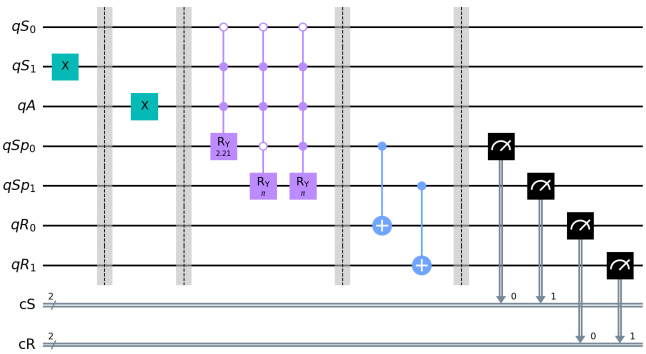

Fig. 4. Example circuit to represent a single transition $P(s_2, a_2)$.

IV. Conclusions

In this work, we have shown a preliminary study towards the creation of quantum environments for reinforcement learning, inspired in the hypotheses of classic RL using MDPs as a first step. We have developed a method to implement a reinforcement learning agent/environment interaction cycle in a quantum computer containing the full transition and reward functions of the MDP. Future works will consider extending the implementation to models with partial observability, and general methods to implement quantum agents working in quantum environments.

Acknowledgements

This article was supported by the project QUANERGY (Ref. TED2021-129360B-I00), Ecological and Digital Transition R&D projects call 2022 funded by MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR.

References

[1] P. Winder, *Reinforcement Learning: Industrial Applications with Intelligent Agents*. O'Reilly, 2020.

[2] M. Putterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[3] D. Dong, C. Chen, H. Li, and T.-J. Tarn, "Quantum reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 5, pp. 1207–1220, 2008.

[4] A. Zeguendry, Z. Jarir, and M. Quafafou, "Quantum machine learning: A review and case studies," *Entropy*, vol. 25, no. 2, 2023. [Online]. Available: https://www.mdpi.com/1099-4300/25/2/287

[5] A. Skolik, S. Jerbi, and V. Dunjko, "Quantum agents in the Gym: a variational quantum algorithm for deep Q-learning," *Quantum*, vol. 6, p. 720, 2022.

[6] S. Jerbi, C. Gyurik, S. C. Marshall, H. J. Briegel, and V. Dunjko, "Parametrized quantum policies for reinforcement learning," 2021.

[7] N. D. Pozza, L. Buffoni, S. Martina, and F. Caruso, "Quantum reinforcement learning: the maze problem," *Quantum Mach. Intell.*, vol. 4, no. 2, pp. 1–10, 2022. [Online]. Available: https://doi.org/10.1007/s42484-022-00068-y

[8] E. Andrés, M. P. Cuéllar, and G. Navarro, "On the use of quantum reinforcement learning in energy-efficiency scenarios," *Energies*, vol. 15, no. 16, 2022. [Online]. Available: https://www.mdpi.com/1996-1073/15/16/6034

[9] S. Sagole, A. Dey, B. Behera, and P. Panigrahi, "Quantum tic-tac-toe: A hybrid of quantum and classical computing," Tech. Rep., 12 2019.