NExt ApplicationS of Quantum Computing

<NE|AS|QC>

Implementation of QRL algorithm on real architecture (D5.5)

Document Properties

Contract Number	951821
Contractual Deadline	M34 (31/06/2023)
Dissemination Level	Public
Nature	Report
Edited by :	Vedran Dunjko (ULEI)
Authors	Simon Marshall (ULEI), Vedran Dunjko (ULEI)
Reviewers	Gonzalo Ferro Costa Andres Gomez Tato
Date	22/05/2023
Keywords	Quantum algorithms
Status	Internal review version
Release	1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 951821

History of Changes

Release	Date	Author, Organization	Description of Changes
0.1	13/04/2023	Vedran Dunjko (ULEI)	First draft: structure of the document
0.2	15/05/2023	Simon Marshall (ULEI)	Details and introduction added
0.3	18/05/2023	Vedran Dunjko (ULEI)	Text refinement
1.0	22/05/2023	Simon Marshall, Vedran Dunjko (ULEI)	Final version for review

Table of Contents

1. Executive Summary

This report is the second deliverable of Task 5.2 - QRL for the inventory management part of the NEASQC project. It describes how we successfully deployed a QRL agent on a real quantum computer to tackle a simplified version of the challenge of inventory management. We describe the setup that solved this problem and examines some interesting conclusions drawn from comparing the real device noise to simulators.

We begin by introducing the reader to quantum reinforcement learning as developed within the NEASQC project, and to the inventory management problem we apply our learning algorithm to in Section 2. Section 3 highlights the key insights we gained from our experiments, particularly the importance of noise while training on a simulator and the relative unimportance of the specific noise model. Section 4 goes into the specifics of our methods and results, going into detail, both about how we achieved this learning problem and the conclusions we draw from our experiments.

2. Description of the quantum algorithm and task

2.1. Quantum algorithm for reinforcement learning

Parameterized quantum circuits (PQC) have emerged as a useful tool in quantum machine learning. A PQC consists of a quantum circuit ansatz and specified parametrized gates. Some of the parameters are used to input classical data into the system and others can be trained to solve the problem at hand.

Recently, the Leiden group, also as a part of the NEASQC project, has focused on the design of reinforcement learning (RL) algorithms based on PQCs . RL refers to a class of problems in which an agent learns to take actions in an environment to maximize some reward signal. This promising approach involves training a PQC to output a policy, i.e., a mapping from states to actions, which maximizes the expected reward (or so-called Q-values), i.e. an approximation of the value of each action in each state. By using gradient-based optimization techniques to update the parameters of the PQC, this approach holds the potential to significantly speed up the learning process and achieve superior performance compared to classical reinforcement learning algorithms. This defines “quantum reinforcement learning” for the purposes of this report.

In particular, in [JER21] we have demonstrated the effectiveness of this general framework in deploying a PQC for reinforcement learning and we have proven the theoretical advantages over classical models. Here, we have built on this work by deploying a similar RL algorithm for PQCs to the inventory management use case. For details on the Quantum RL machinery we use, we refer the reader to [JER21], and to the deliverable D5.2 of the NEASQC problem

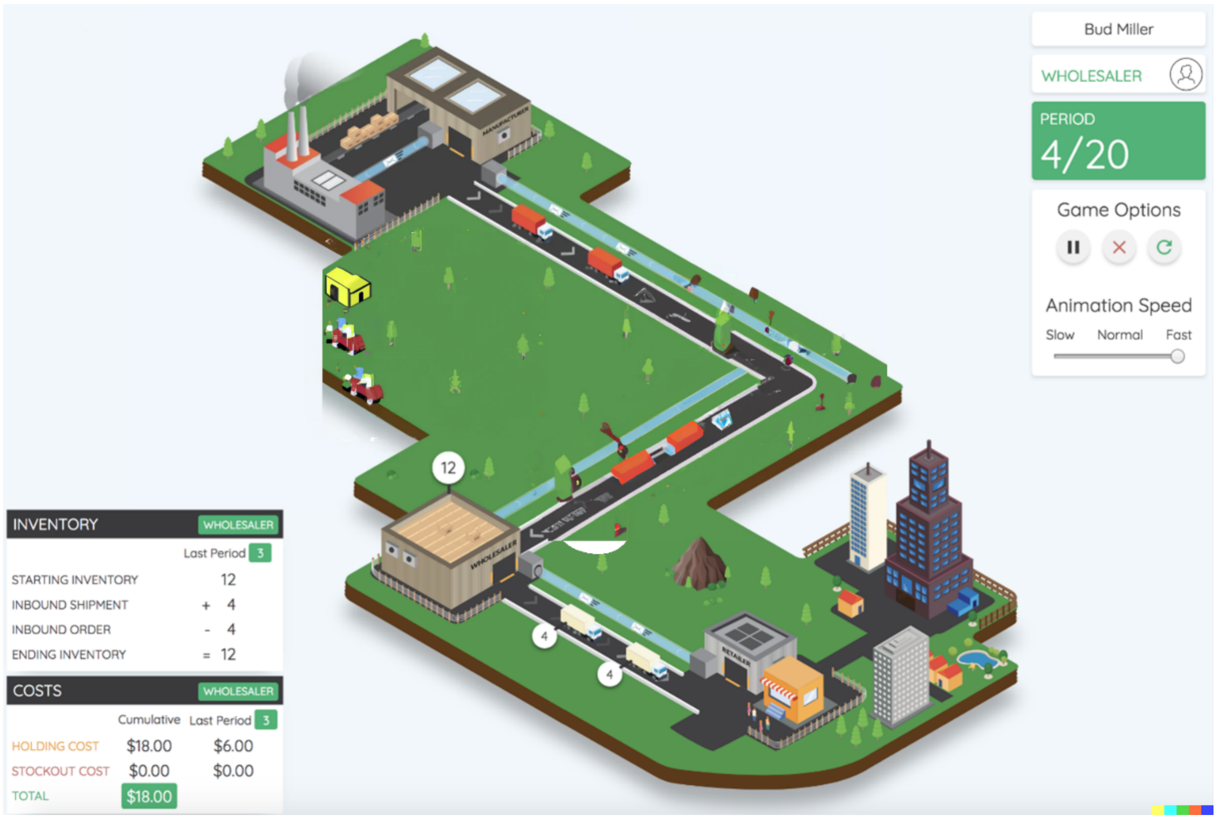
2.2. Test problem

A first step in the deployment of new technologies consists of tests in simpler settings, where the performance can be fully assessed. Here we describe the task we used toward larger-scale inventory management problems. One commonly studied problem in inventory management is the task of balancing supplies as a middle-person in a long supply chain, originally formulated as trading crates of beer which gives rise to the problem's traditional name: “the beer game”. The beer game can be thought of as a reinforcement learning environment, where multiple agents interact with each other in a supply chain. The agents form a chain of agents, each able to trade with the agent above and below them in the chain. The chain begins with the factory where the items for the inventory are produced and ends with the retailer where inventory is consumed. Between these two are some number of “middle-persons”, who must balance their inventory to never run out (and lose profit) or to stock too much (and waste capital).

The goal of the agents is to maximize their profit by balancing their inventory levels and meeting customer demand. Each agent receives a reward based on their profit, which is determined by the difference between their revenue from selling beer and their costs of holding and ordering inventory.

The agents can take actions by placing orders for beer from the agent upstream and deciding how much beer to keep in their inventory. However, the agents only have partial observability of the system, as they do not know the exact demand or inventory levels of the other agents. The details of the setting are provided later.

The challenge of the beer game is to learn a policy that maximizes profit while dealing with the uncertainties and delays inherent in the supply chain. The agents must learn to adapt their inventory levels and ordering decisions based on the feedback they receive from their rewards, as well as the observed outcomes of their actions.



Bud Miller

WHOLESALER

PERIOD 4/20

INVENTORY WHOLESALER

		Last Period 3
STARTING INVENTORY		12
INBOUND SHIPMENT	+	4
INBOUND ORDER	-	4
ENDING INVENTORY	=	12

COSTS WHOLESALER

	Cumulative	Last Period 3
HOLDING COST	$18.00	$6.00
STOCKOUT COST	$0.00	$0.00
TOTAL	$18.00	

Figure 1: Illustration of the beer game from [ORO17] modified with DALL.E 2. Two warehouses are shown between the factory (top) and distributor/end consumer (bottom). At each stage of the game an individual warehouse/agent decides how much beer to order from the agent above them in the supply chain and sends the beer ordered by the agent below.

3. Simulations v.s. real device implementations

We have successfully implemented a quantum reinforcement learning (QRL) method to solve the inventory management problem from the previous section. This was done in simulations, but we have also trained the QRL agent using a simulator and then deployed it on a real device -- at present the full training on the device is beyond our budget of QC time. Nonetheless, this enables us to suggest how correct the claims of the effectiveness of QRL in solving RL problems are, compared to literature e.g. [CHE20, SKO22, LOC20] when one considers realistic noise models and actual device implementations. In particular, we present a clear comparison of models trained noiselessly and with noise, and to see how the actual deployments of these algorithms differ.

Our findings show that a QRL model trained with noise present is able to adapt well to different forms of noise and perform nearly equally well on both simulated and real device noise. However, a QRL model trained on a noiseless simulator performs poorly on both simulated and real device noise, indicating a clear gap between these noisy cases. This suggests that QML statements made when trained on a noisy simulator may carry over to real devices, while QML statements made on a noiseless simulator may not. These findings are detailed in the next section.

4. Results

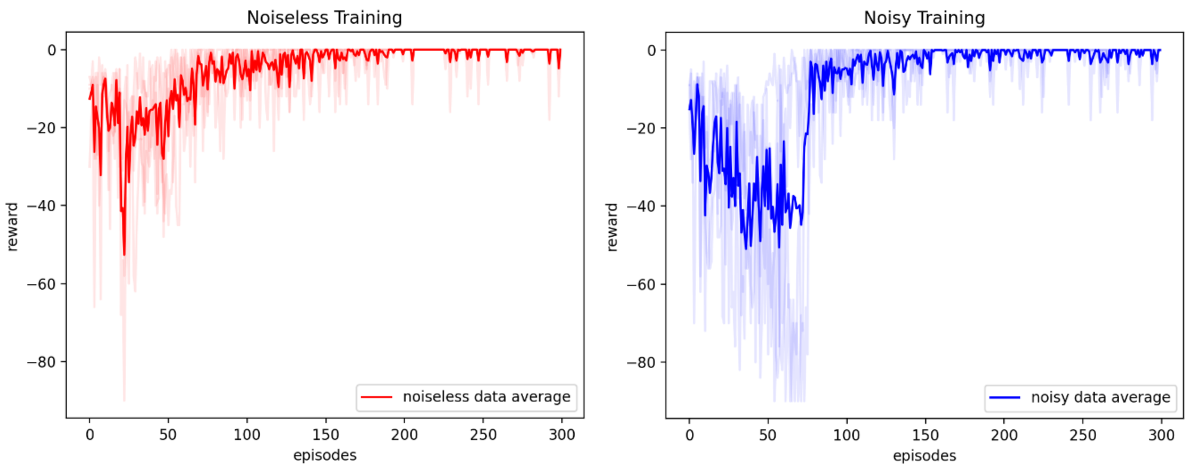


Figure 2a & 2b: The reward for our model during training on the inventory management task. Figure 2a is for the noiseless case, 2b includes noise in the simulator. Convergence in the noiseless case is quicker but potentially less robust (see figure 3).

Methods:
The experimental setup involved using the previously described QRL algorithm which we developed for previous papers. We utilized a hardware-efficient ansatz with 3 qubits and had a depth of 3. We trained two models, one noiselessly and one with noise.

We used the in-built simulator in TensorFlow Quantum during training - this was convenient as the original work in [JER21] already developed all the QRL machinery in that language. In the noisy training run, we added noise using phase flip and amplitude damping channels. The rates were set to approximately equal levels seen in the real device although the noise profile was largely different (amplitude damping with $\gamma = 0{,}001$ and phase flip with $p = 0.0014$). During deployment, we tested the model on a real device (IBM Cairo) and a different noisy simulator (this time provided by IBM) to make a fair comparison.

The inventory management environment was based on the game described and consisted of a middle-person, a factory that always met demand, and a retailer that purchased beer based on a preset amount. Our QRL model was in charge of ordering stock for the middleman. The agent could not see the last order, making it challenging for the model to learn under uncertainty about what was last ordered. We conducted experiments with episode lengths ranging from 5 to 30 trading steps ultimately settiling on 10 trading steps to not overwhelm the quantum computer during deployment. Reward was assigned negatively, with -1 point awarded for every unsold piece of inventory held and -2 points awarded for every missed sale (having demand but not enough stock to fufill this demand)

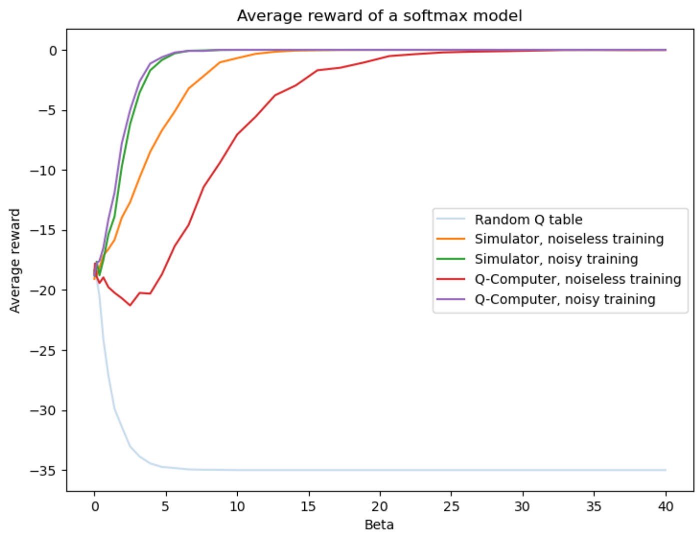


Figure 3: Deployment performance of models on quantum simulator (different from the simulator used in training) and real quantum device against beta (softmax inverse temperature). When detereministic both models in both settings perform optimally, although when the policy is made stochastic (by increasing the temperature) the more robust noiseless case prevails.

Results:
The results showed that the noiseless case had quicker training, with clear and strong convergence early on in all training runs. In contrast, the noisy case was more spread out, with some training runs taking much longer to converge to a good policy. When we deployed the trained weights in the real device and set the agent to maximize the expected reward, all learned models performed optimally and thus equally well. This result is better than expected, but consistent with the relatively simple nature of the game.
This only means that the noisy system still has the correct action as the most likely one. However, when a softmax was applied to choose the action, as is often done during training to encourage the agent to explore actions it suspects will be fruitful, the model trained on the noisy simulator outperformed the noiseless model in all instances. This is because now the distribution of actions is not greedy, and differences between the learned action probabilities become directly influential in the actions of the agent.

Interestingly, the noisy model also had a much closer performance on the simulator and real quantum computer, despite the differences in the noise profiles used during training and the noise observed in the real device.

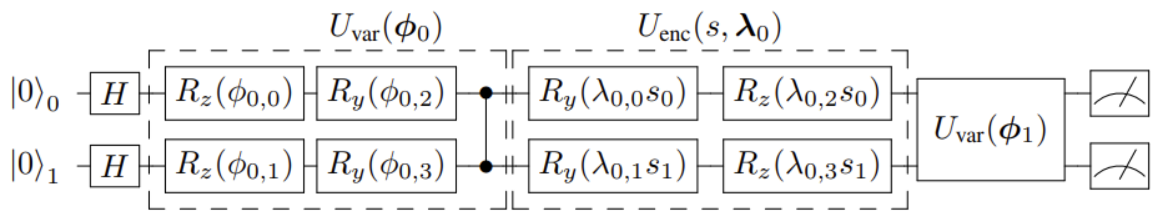


Figure 4: bThe PQC architecture for n = 2 qubits and depth Denc = 1 comprises alternating layers of encoding unitaries, denoted as Uenc(s, λi), which take a state vector s = (s0, . . . , sd−1) and scaling parameters λi (from a vector λ ∈ R^|λ| of dimension |λ|) as input. Additionally, it includes variational unitaries denoted as Uvar(φi), which take rotation angles φi (from a vector φ ∈ [0, 2π]^|φ| of dimension |φ|) as input.

5. Conclusion

In this deliverable, we have applied quantum reinforcement learning (QRL) to the inventory management problem, which involves balancing inventory levels in a long supply chain to maximize profit. We have trained a QRL model using a parameterized quantum circuit (PQC) ansatz and gradient-based optimization and evaluated its performance on both a simulator and a real quantum device. Our findings suggest that QRL can be effective in solving reinforcement learning problems, but the performance of the model can be sensitive to the presence of noise in the training and execution.

We have demonstrated that a QRL model trained with noise present is able to adapt to various forms of noise and perform nearly equally well on both simulated and real device noise. However, a QRL model trained on a noiseless simulator performs poorly on both simulated and real device noise, indicating a clear gap between either noisy case. These results suggest that QML statements made when trained on a noisy simulator may apply to real devices, while QML statements made on a noiseless simulator may be more robust.
Overall, our work contributes to the growing body of research on quantum machine learning and reinforcement learning, demonstrating the potential of QRL in real-world applications. Our results highlight the importance of considering the impact of noise in the training data and the need for further research to develop robust and scalable QRL algorithms.

6. Acronyms and Abbreviations

Term	Definition
PQC	Parametrized quantum circuit

Table 1: Acronyms and Abbreviations

7. List of Figures

8. List of Tables

9. Bibliography

[JER21] Jerbi, S., Gyurik, C., Marshall, S., Briegel, H., & Dunjko, V. (2021). *Parametrized Quantum Policies for Reinforcement Learning.* Advances in Neural Information Processing Systems (pp. 28362-28375). Curran Associates, Inc.
[ORO17] Oroojlooyjadid, A., Nazari, M., Snyder, L. V., & Takác, M. (2017). A deep q-network for the beer game with partial information. CoRR, abs/1708.05924.
[CHE20] Chen, S. Y. C., Yang, C. H. H., Qi, J., Chen, P. Y., Ma, X., & Goan, H. S. (2020). Variational quantum circuits for deep reinforcement learning. *IEEE Access*, *8*, 141007-141024.
[SKO22] Skolik, A., Jerbi, S., & Dunjko, V. (2022). Quantum agents in the gym: a variational quantum algorithm for deep q-learning. *Quantum*, *6*, 720.
[LOC20] Lockwood, O., & Si, M. (2020, October). Reinforcement learning with quantum variational circuit. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (Vol. 16, No. 1, pp. 245-251).