Replenishment of Retail Stores
TCS Quantum Computing Challenge

Phase 1 Submission

Affiliation: Bloq

Nikhil Londhe, Grishma Prasad, Jay Patel, and Pratik Kumar De

January 22, 2024

Contents

1 Replenishment of Retail Stores **1**

2 Methodology **1**
2.1 Variational Quantum Eigensolver (VQE) 1
2.2 Quantum Reinforcement Learning 2

1 Replenishment of Retail Stores

For a large retailer, replenishing the store estate with the right products at the right time requires striking the right balance between stock availability and supply chain cost. The challenge has multiple dimensions, including the rate of sale, range, delivery frequency, pack sizes, sales forecast accuracy, seasonality, distribution center capacity and throughput limitations, physical variations between stores, and store access limitations.

Currently, for a retailer with over a thousand stores and tens of thousands of different SKUs, granularity is traded off against speed of calculation, and hence, stores and SKUs are grouped. This results in a sub-optimal solution. There is immense potential to identify quantum-based solution approaches to address the need of SKU based service levels with the lowest possible working capital and cost to serve.

2 Methodology

Our challenge is using Quantum computing methods to tackle the problem of optimally replenishing retail stores. This problem has two levels of optimization. Firstly, we need to optimize at the store level by finding the optimal replenishment quantity the store will order from the distribution center (DC) to meet the demand while maximizing profit. This involves considering factors such as the stock at the store, the inventory holding cost, and the forecasted demand. Secondly, we need to optimize the DC level, where the DC receives replenishment requests from all the stores and decides how much it will replenish each store based on its stock, lead time for stores, and the expected profit. We focus on store-level optimization and have developed strategies that we discuss below. We use two methods: Variational Quantum Eigensolver (VQE) and Quantum Reinforcement Learning.

2.1 Variational Quantum Eigensolver (VQE)

Variational Quantum Eigensolver (VQE) is a quantum computing algorithm designed to find the ground state energy of a given quantum system. It employs a variational approach by parameterizing a trial wave function and iteratively optimizing its parameters to minimize the system's energy. VQE is particularly suited for near-term quantum devices with limited qubits and coherence times, making it a promising candidate for optimization problems and quantum chemistry simulations. The algorithm leverages classical optimization techniques with a quantum processor, combining classical and quantum computing strengths to tackle complex problems.

Our approach is to perform VQE for product and store combinations to get optimal replenishment quantities for each of the 26 weeks from mid-May to October. Since there are around 16314 unique combinations of products and stores for which we are given the forecasted demands, we decided to look for a way to categorize these combinations to reduce the number of optimizations we need to run. The objective function is given below (for a given product i and store j for week k)

$$P_{ijk} = D_{ijk} \times (P_r, i + P_c, i) - X_{ijk} \times P_c, i - (OH_{ijk} + X_{ijk}) \times P_h, i.$$

Where,

- P_{ijk}: Weekly profit for ith product in jth store for week k.
- D_{ijk}: Demand quantity for ith product in jth store for week k.
- OH_{ijk}: On-Hand quantity for ith product in jth store for week k.
- X_{ijk}: Replenishment for ith product in jth store for week k.
- $P_{r,i}$: Profit per unit for product i
- $P_{c,i}$: Procurement cost per unit for product i.
- $P_{h,i}$: Holding cost per unit per week for product i.

We note that the products can be categorized based on the profit per unit, on-hold cost per unit per day, and purchasing cost per unit. E.g., there are about 30 products with profit = 1.398, holding cost = 0.5592, and purchasing cost = 5.592. We found 33 such categories. Similarly, the stores can be categorized in terms of lead times. The lead time is either 1,2,3 or 4 days for a given store. Our objective function also includes the forecasted demand (D), which is given in decimals from close to 0 to around 1.7. We decided to bin the forecasted demands into 10 bins. We map each forecasted demand to the upper limit of the bin to which it belongs. Since we also need to consider the store stock (OH), we will need to further categorize. For the 1st week in May, there are about 344 optimizations to be done based on the categories created with Product (profit, purchasing cost, holding cost), store (lead time), forecasted demand (after our mapping), and store stock (OH). The number of optimizations for future weeks will depend on categories created because of different store stocks. We expect that there will be many similar optimizations week after week, so instead of running all the optimizations, we will leverage the results we get from prior optimizations.

The number of qubits will depend on the maximum possible replenishment based on the constraint and the number of binary variables as a result of the processing of our objective function. For example, for max replenishment of 15 units, we found that after using some converters and specifically the integer to binary converter, we require 15 qubits. Based on our data analysis, we conclude that the maximum replenishment per item and store would not exceed 15; thus, for this solution, a quantum system with a 15 qubits should be sufficient. We are planning to use Pennylane primarily. The qiskit-optimization package has a way to write an objective function along with its constraints into a quadratic program using the docplex library. We can then do some processing of our quadratic program.

To construct the Hamiltonian, we can use the *to_ising* method. We will use this Hamiltonian to implement VQE using Pennylane. We then need an appropriate parameterized circuit or ansatz, and Pennylane has a lot of known options like UCCSD, real amplitudes, QAOA ansatz or we could take known circuits from other frameworks, e.g., *TwoLocal*, *QAOAAnsatz*. The optimization will try converging to the state corresponding to the least energy value with respect to the Hamiltonian. So, we will minimize the negative of the objective function mentioned above. With quantum optimization techniques like VQE, we can take advantage of quantum mechanical properties and also use sophisticated classical optimizers, which can speed up the solution for complex problems like finding optimal replenishment in the retail supply chain.

2.2 Quantum Reinforcement Learning

Markov Decision Process, or MDP in short, is a tool for studying and solving optimization problems. MDP is characterized by states, actions, transition probabilities, and rewards, with the Markov property ensuring that future states depend only on the current state and action. The goal is to find an optimal policy that guides the agent to take actions, maximizing the expected cumulative reward.

Previous attempts have dealt with replenishment problems using MDP[1][4]. The common approach is creating a state of store stock and demand. We will create states similarly but add available capital as an extra dimension. We define the state $\mathcal{S}$ as,

$$\mathcal{S} = (OH, FD, C, T)$$

where OH is the on-hand inventory, FD is the forecasted demand, C is the available capital, and T is the time period (week in our case). OH is matrix of shape $N_p \times N_s$ where N_p is number of products and N_s is the number of stores. OH_{ij} means on-hand stock for ith product in the jth store.

Action is the weekly replenishment we will make, denoted by X. Like OH, X will be a matrix with the shape $N_p \times N_s$. The reward for action will be calculated based on the sales that occur that week and will be calculated from the forecasted demand. We create a skewed distribution for predicting sales, with 0 being the minimum and FD being the maximum. Then, we sample from this distribution to get the sales. An important aspect of MDP is the state space. In our case, it is all the possible replenishments we can make. Figuring out all possible actions (replenishment) requires determining the total number of partitions of all possible products we can buy at any point in time. Partition theory deals with these types of problems, and there is no analytical formula to calculate the number of partitions for any given number; rather, we have to count it manually. Due to this, at any given state, it is generally difficult to know all possible actions. So, we devised our solution to have an MDP for every combination of products and stores. Therefore, our new MDP state s_{ij} will be,

$$s_{ij} = (OH_{ij}, FD_{ij}, C, T)$$

where, definitions as same as before but OH_{ij} and FD_{ij} are scalars now instead of multidimensional matrices. We will classify product/store combinations into high, medium, and low profitability classes. So, we will run MDP for combinations with high first, medium second, and low third. This will ensure that we prioritize products and stores that are highly profitable.

Reinforcement Learning

We will use Quantum Reinforcement Learning (QRL) to tackle the abovementioned MDP. Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to make sequential decisions by interacting with an environment. The agent receives feedback through rewards or punishments based on its actions, guiding it towards learning optimal strategies. RL involves exploring the environment to discover the most rewarding actions while exploiting known strategies. Key components include states, actions, rewards, policies, and value functions[3].

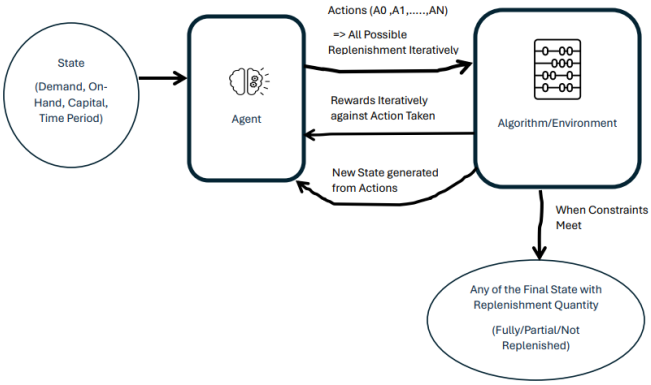

Figure 2.1

QRL[2] is similar to classical RL, wherein you have a quantum state that acts as an agent to interact with the environment. The agent interacts with the environment via actions and gauges them by rewards. A series of actions form a policy, and we want to find one that maximizes the cumulative reward. Popular algorithms like Q-learning and deep reinforcement learning methods, such as Deep Q Networks (DQN), have been successful in applications ranging from the game playing to robotics and autonomous systems.

AWS Resources for Phase 2

- For VQE, we have done an assessment of the number of optimizations. If we use 1000 shots per optimization and assume that there will be many similar optimizations across weeks, the minimum credits will be around **$250**.
- For QRL, we will need at least 5 qubits to implement quantum RL. We need to hold four dimensions of the state in a quantum state using angle encoding and extra qubits for the ancillae. Like VQE, we will need at least **$250** to run QRL for a handful of product store combinations.

We will require at least **$500** of AWS credits for Phase 2 submission.

References

[1] J. Goedhart, R. Haijema, R. Akkerman, and S. de Leeuw. Replenishment and fulfilment decisions for stores in an omni-channel retail network. *European Journal of Operational Research*, 311(3):1009–1022, 2023.

[2] W. Hu and J. Hu. Distributional reinforcement learning with quantum neural networks. *Intelligent Control and Automation*, 2019.

[3] R. Karthikeyan. *Quantum inspired algorithms for learning and control of stochastic systems*. PhD thesis, 2015.

[4] A. G. Zheng Sui and L. Lin. A reinforcement learning approach for inventory replenishment in vendor-managed inventory systems with consignment inventory. *Engineering Management Journal*, 22(4):44–53, 2010.