BYU QGars - YQuantum 2026

Kai Sandberg - Will Ott - Cayman Quinn - Peyton Cook

We had an amazing time learning to apply two quantum algorithms to real insurance problems. This is our collective gained knowledge over the last 24 hours.

1 Integer Linear Programming in Insurance

Marbles. Imagine many marbles of different colors. You are arranging them into gift bags; each recipient needs a blue or red marble, and optionally some other colors as well.

This analogy helped us understand, to some degree, the confusing world of insurance. From the company's perspective, they have several coverages (marbles) of several families (colors) to distribute into discounted packages. Abiding by some rules of mixing coverages into bundles, the company aims to maximize profit by choosing which packages to offer to their audience. This can be formulated mathematically as follows.

Let n and m be indices labeling coverages and packages. Let M_0 be an $n \times m$ matrix, where $M_{ij} \in \{0, 1\}$ indicates whether coverage i is in bundle j. The goal is to find an M_0 that maximizes profit subject to some restraints; that is, minimize

$$-\sum_{i,j}(M \odot C)_{ij}$$

subject to certain conditions on M_{ij}, where C_{ij} encodes the average revenue from the ith coverage being in the jth bundle. This is a textbook ILP, or integer linear programming method. This turns out to be NP-hard, as the coefficients need to be 0 or 1. The proceeding quantum algorithms aim to solve this combinatorial problem with the high dimensionality of quantum computing.

2 Quantum Approximate Optimization Algorithm

QAOA is a sneaky beast. We spent the first 9 hours understanding how the algorithm worked. The idea is simple enough: construct a matrix whose eigenvalues are the objective values of the feasible set and eigenvectors corresponding to the lowest one. We begin by encoding the ILP constraints into the objective function by introducing a large scalar penalty λ. The domain of the objective for the quantum computer is all points in

$$\{0, 1\}^{\otimes n},$$

many of which are not feasible. By using nonnegative slack variables, we can ensure that the objective function does not converge to a false minimum by adding a punishment proportional to λ for such points.

Mathematically,

$$f(y) = -\sum c_i M_i + \lambda F(x),$$

where $F(x) = 0$ represents the feasible set. Expanding,

$$f(y) = \sum_i Q_{ii} y_i + \sum_{i<j} Q_{ij} y_i y_j$$

up to a constant. This form is useful after making the substitution

$$y_i = \frac{I - Z_i}{2}.$$

Associating $y_i y_j$ with ZZ and y_i with Z gates, the coefficients Q of this objective function are used to construct precisely the matrix whose eigenvalues are the objective values of the feasible points. This matrix is called a cost Hamiltonian H_c, though seemingly, the term's only relation to energy is the objective to minimize.

To implement H_c on a circuit, we use the unitary matrix

$$U(\gamma) = e^{i\gamma H_c},$$

as H_c is not necessarily Hermitian. After applying on an equal superposition of states, the parameter γ determines the magnitude of the phase picked up by the basis eigenstates; indeed,

$$U(\gamma)|E\rangle = e^{i\gamma E}|E\rangle.$$

Finally, this phase is converted into a magnitude after applying

$$U(\beta) = e^{i\beta \sum_i X_i}.$$

To see this concretely, consider

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$

where $|0\rangle$ and $|1\rangle$ have corresponding eigenvalues, or equivalently, objective function values. Applying $U(\gamma)$ gives a relative phase:

$$\frac{1}{\sqrt{2}}\left(e^{i\gamma E_0}|0\rangle + e^{i\gamma E_1}|1\rangle\right).$$

This relative phase is what allows the amplitudes to change after applying an X-based mixing gate. A weakness in QAOA is periodicity; E_0 and $E' = E_0 + 2\pi$ apply the same phase though their objective values differ. This can be addressed by applying $U(\gamma)$ and $U(\beta)$ successively.

3 DQI

Decoded quantum interferometry is a more recent method to minimize an objective function. Constraints are encoded into a collection of XOR operations and implemented through phase and CNOT gates. The most difficult part is finding the matrix B, which contains information on the XOR-translated constraints. Our DQI implementation worked, but took too long for our laptops (or even Selene) to run in a reasonable amount of time. Instead, we implemented a DQI-inspired QAOA approach, taking ideas from parity and syndrome qubits.