Quantum Walk Optimisation Algorithm

Autumn 2022

Anthony Gambale

1 Introduction

Overview
For a given combinatorial optimisation problem, its set of possible solutions S formulated as bit strings and a quality function $q : S \to \mathbb{R}$, the quantum walk optimisation algorithm will choose an element $s \in S$ with a relatively high value under q. It does this theoretically at least as fast as the QAOA, which exhibits a $\mathcal{O}(\sqrt{n})$ speedup over classical algorithms in time complexity.

Indexing Unitary
Let $\mathcal{M}$ be the number of elements in S, and $[\mathcal{M}] = \{1, 2, ..., \mathcal{M}\}$. Given a function $id : S \to [M]$ who assigns an index (integer ordering) to every possible solution in S, the QWOA requires a quantum circuit $\hat{U}_{\#}$ that computes id on a bit string in a quantum register. This circuit is known as the indexing unitary. Its inverse maps integers in $[\mathcal{M}]$ back to solutions in S, and is called the unindexing unitary.

The indexing unitary of QWOA allows for symmetry breaking to be a part of the algorithm, unlike the QAOA which requires all possible qubit states to be searched.

Preparing the Initial State
We initialise the qubits into a superposition of states in S. We do this by putting them into superposition of the first $\mathcal{M}$ integers, then applying the un-indexing unitary, as follows.

$$|\psi_0\rangle = U_{\#}^{\dagger} \frac{1}{\sqrt{\mathcal{M}}} \sum_{k=0}^{\mathcal{M}-1} |k\rangle$$

For example, take $S = \{011, 110, 101\}$, representing ways to choose 2 objects from a group of 3, for some hypothetical subset problem. With an indexing function that maps $011, 110, 101$ to $0, 1, 2$ respectively, we get

$$
\begin{aligned}
|\psi_0\rangle &= \hat{U}_{\#}^{\dagger} \frac{1}{\sqrt{3}} \sum_{k=0}^{2} |k\rangle \\
&= \hat{U}_{\#}^{\dagger} \frac{1}{\sqrt{3}} (|0\rangle + |1\rangle + |2\rangle) \\
&= \hat{U}_{\#}^{\dagger} \frac{1}{\sqrt{3}} (|000\rangle + |001\rangle + |010\rangle) \\
&= \frac{1}{\sqrt{3}} (\hat{U}_{\#}^{\dagger} |000\rangle + \hat{U}_{\#}^{\dagger} |001\rangle + \hat{U}_{\#}^{\dagger} |010\rangle) \\
&= \frac{1}{\sqrt{3}} (|011\rangle + |110\rangle + |101\rangle)
\end{aligned}
$$

Resulting in a superposition of the states in S. Since the quantum Fourier transform puts its input qubits into a superposition of these first $\mathcal{M}$ integers, the state preparation unitary is $\hat{U}_{\#}^{\dagger} \hat{\mathcal{F}}_{\mathcal{M}}$.

Quality Operator
The next operator in the QWOA is the quality operator, $\hat{U}_Q$. We start with a diagonal matrix $\hat{Q}$ that scales potential solutions in S by their quality value $q(s)$, with S an eigenbasis of $\hat{Q}$. In order to time evolve the state of our qubits to increase this function, we use $\hat{U}_Q = e^{i\gamma\hat{Q}}$ as our quantum circuit, since it is a solution to the Schrodinger equation, $\hat{Q}\,|\psi\rangle = i(\partial/\partial t)\,|\psi\rangle$.

Quantum Walk (Mixing) Operator
The mixing operator in the QWOA, $\hat{U}_W$, relies on a graph which has the potential solutions in S as vertices, and connects them to form paths. For example, in figure 1, a graph with sets of integers at its vertices can be seen in image (a). These sets represent possible solutions to a hypothetical subset problem. The QWOA uses the adjacency matrix of this graph to evolve qubit states, moving the qubits from representing one vertex of this graph into a superposition of adjacent vertices.

In image (b), the effect of the indexing unitary is shown. It moves the relevant rows and columns of the adjacency matrix, the ones corresponding to valid bit strings, to the front, preventing invalid bit strings from being visited by the walk.

Finally, if the graph is connected circulantly, its adjacency matrix is diagonalisable by the quantum Fourier matrix, giving $\hat{\mathcal{F}}_{\mathcal{M}}^{\dagger} \hat{\lambda} \hat{\mathcal{F}}_{\mathcal{M}}$ to be the adjacency matrix. To time evolve quantum state by this matrix, we use the following unitary

$$\hat{U}_W = \hat{U}_{\#}^{\dagger} \hat{\mathcal{F}}_{\mathcal{M}}^{\dagger} e^{i\hat{\lambda}t} \hat{\mathcal{F}}_{\mathcal{M}} \hat{U}_{\#}$$

This unitary is highly efficient to compute on a quantum computer, because it breaks an arbitrary unitary (the original adjacency matrix) into a Fourier transform and a diagonal matrix, both of which have existing efficient implementations in quantum hardware (3).

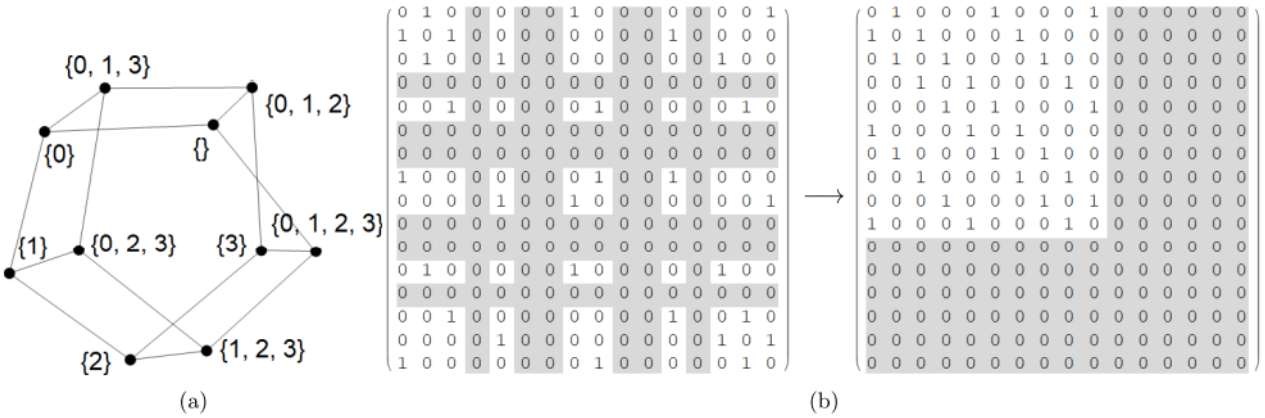

Figure 1: Connectivity graph of QWOA mixing circuit, adjacency matrix and the role of indexing (3).

Parameterisation

The introduction of variables γ and t into these unitaries allows for parameterisation of this circuit. These parameters restrict the number of possible solutions that the quantum computer can search through to a subset of S. With good parameters, the circuit can be directed towards high quality solutions. We use a classical optimisation algorithm to select these parameters.

This leads to the generalised, trotterised approach to the QWOA

$$|\gamma, t\rangle = \hat{U}_W(t_p)\hat{U}_Q(\gamma_p)...\hat{U}_W(t_1)\hat{U}_Q(\gamma_1)\,|\psi\rangle$$

For p layers of repeated circuit. Figure 2 details how this circuit looks at one layer (p=1), with the initial state preparation at the beginning.

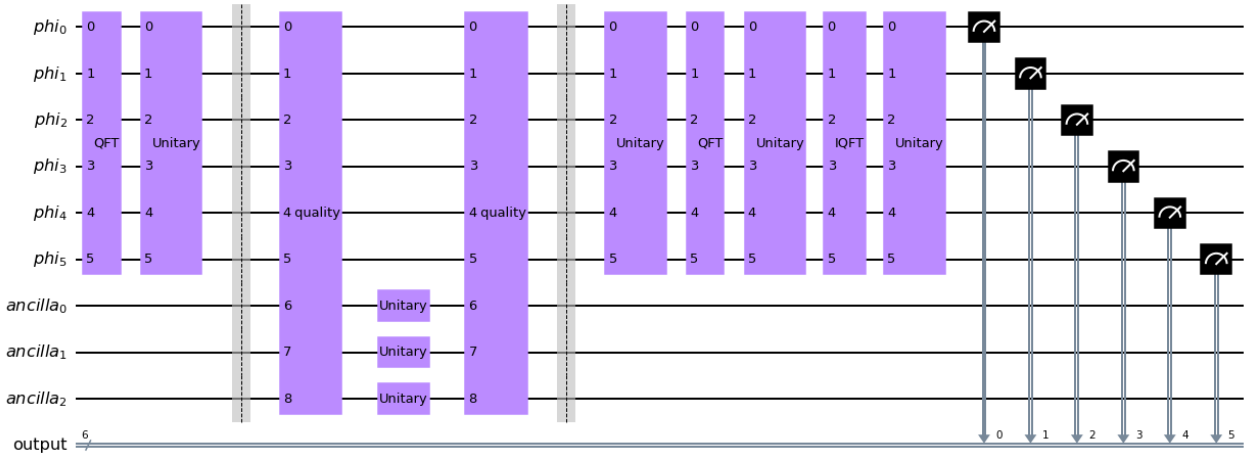

Figure 2: High-level view of QWOA circuit. Initial state preparation is QFT followed by unindexing unitary. Quality unitary uses a circuit that implements the quality value, with rotation by parameters between. This is an efficient method given by Marsh and Wang. Quantum walk circuit is given as per the formulation above, with indexing, QFT, evolution by eigenvalues of adjacency matrix, inverse QFT and inverse indexing.

2 Exploring the Indexing Unitary

The Quantum Walk Optimisation Algorithm's capacity for indexing allows the exploration of solution spaces defined by arbitrarily complex constraints. This is a very big improvement on the QAOA, where all constraints in the allowed solution bit strings must be made within the cost hamiltonian.

Take, for example, the 6 permutations on 3 objects.

$$[0,1,2],[0,2,1],[1,0,2],[1,2,0],[2,0,1],[2,1,0]$$

To encode these as bit strings, we use $3\log_2(3) = 6$ bit registers as follows

$$S = \{000110, 001001, 010010, 011000, 100001, 100100\}$$

The first 2 bits of each string represents the first number in the permutation list. The next 2 bits represent the middle number, and the last 2 bits the last number. So for example, the string 001001 matches the permutation $[00_2, 10_2, 01_2] = [0,2,1]$.

Indexing Method
We index the permutations using a weighted sum of the values of their respective Lehmer code. The ith value of the Lehmer code for a permutation π is determined by the number of values in the permutation after the ith one who are smaller than it in value, or

$$d_i = |\{j : j > i \wedge \pi_j < \pi_i\}|$$

We simply index a permutation π by the following weighted sum

$$id(\pi) = \sum_{i=1}^{n} d_i(n-i)$$

Using this function, we build an indexing unitary $\hat{U}_{\#}$, such that the entry at column i, row $id(i)$ equals 1. Columns x where x does not represent a valid permutation are mapped to integers outside the index range.

QWOA Circuit
We created a QWOA circuit using this indexing unitary, to test first and foremost if it would force the circuit to produce only valid permutation outputs, and secondly to analyze the contour plot for depth 1.

In this circuit, we use circulant connectivity of only distance 1 on the valid indices, as shown in figure 3.

This connectivity graph only connects nodes 6, 7, 8, 9, 18, 24, 33, and 36. These values are the base 10 representation of the bit strings denoted above in the set S.

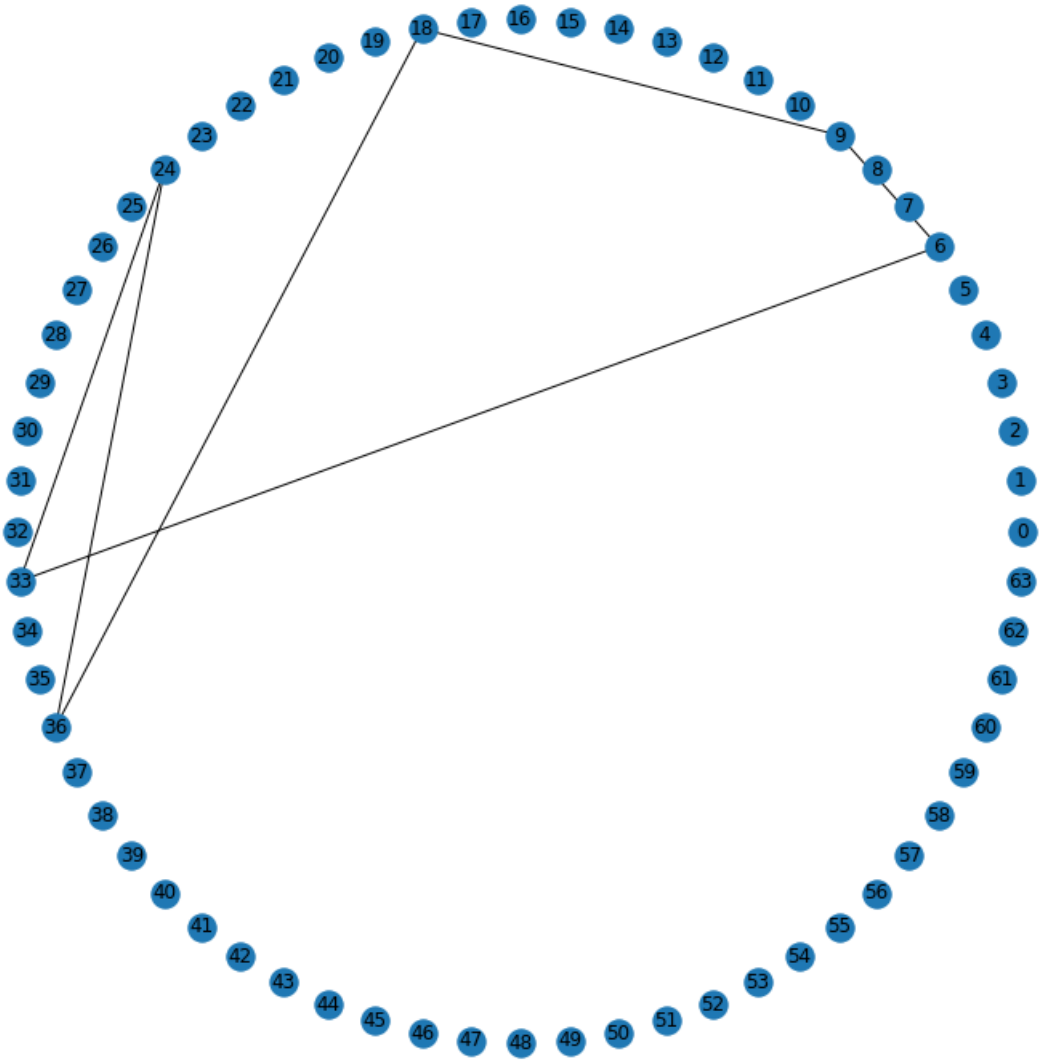

Figure 3: Connectivity graph used in $\hat{U}_W$ operator.

The quality function to optimise here simply requires the first 3 bits of the bitstring to be set. This simple quality circuit works for our purposes.

As seen in figure 4, on 512 shots, the circuit gives an output clearly in favour of the bit string 011000, representing the permutation $[1, 2, 0]$, as seen in figure 4. This shows that indeed, the indexing circuit restricts the QWOA algorithm from selecting other invalid integers with higher quality, like 111000, 111100, etc. This restrition would need to be encoded into the cost hamiltonian to work for QAOA, and would be more difficult to implement and balance with other cost function terms.

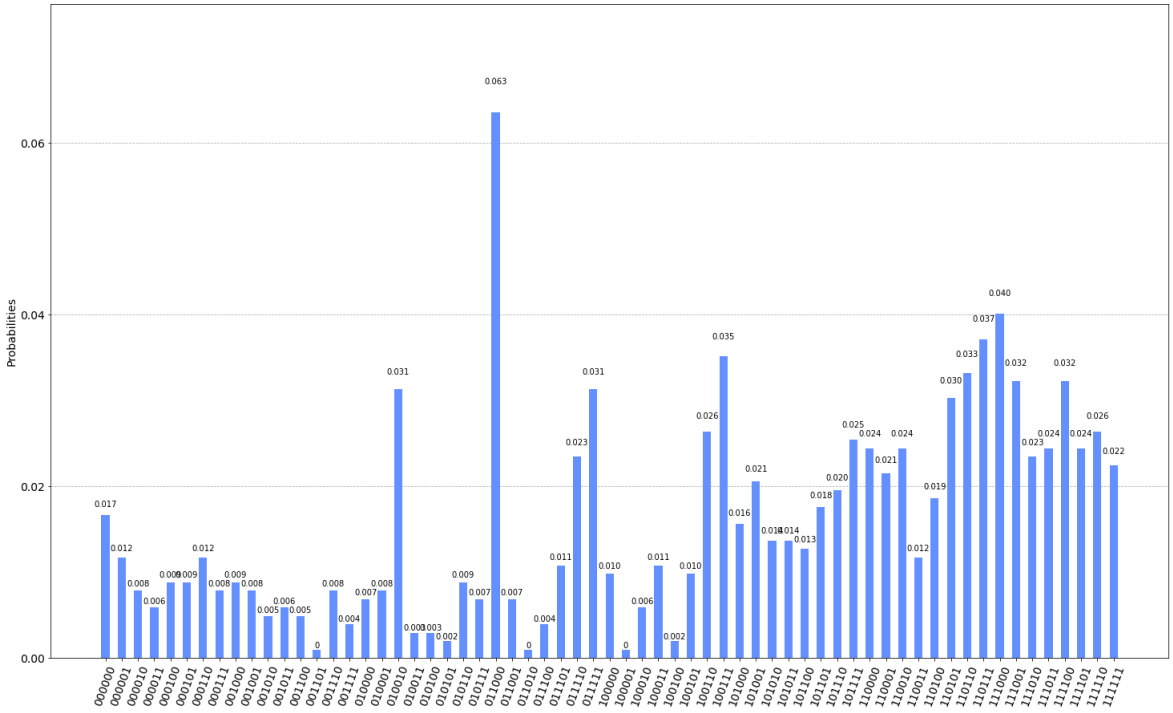

Figure 4: Output of QWOA permutations circuit after 512 shots. Predominant solution is 011000.

References

(1) Bennett, T., Matwiejew, E., Marsh, S., Wang, J. B. (2021). Quantum Walk-Based Vehicle Routing Optimisation. Frontiers in Physics, 9. https://doi.org/10.3389/fphy.2021.730856

(2) Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A., Preda, D. (2001). A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem. Science, 292(5516), 472–475. https://doi.org/10.1126/science.1057726

(3) Marsh, S., Wang, J. B. (2020). Combinatorial optimization via highly efficient quantum walks. Physical Review Research, 2(2). https://doi.org/10.1103/physrevresearch.2.023302

(4) Slate, N., Matwiejew, E., Marsh, S., Wang, J. B. (2021). Quantum walk-based portfolio optimisation. Quantum, 5, 513. https://doi.org/10.22331/q-2021-07-28-513

(5) Farhi E., Gutmann S., Goldstone J. (2014). A Quantum Approximate Optimization Algorithm. arxiv journal. https://arxiv.org/abs/1411.4028