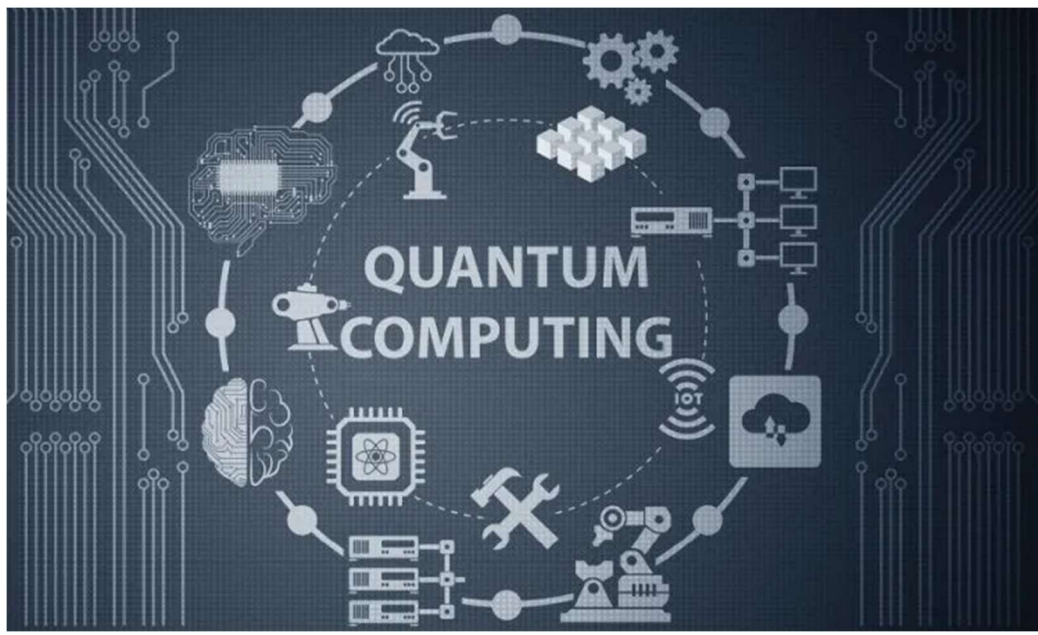


Quantum control implementation on a Quantum Computer

Edition 1.1 (July 7, 2022)

Author: Christophe PARISEL

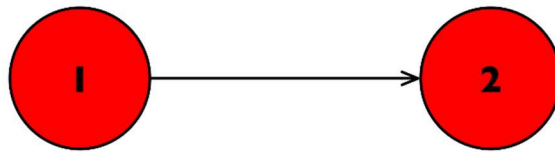


Permalink: https://github.com/labyrinthinesecurity/quantumComputing/blob/main/3_reachability.pdf

Quantum controls is a new and promising branch of **IT security**: it is expected to provide not only new kinds of controls that could complement the baseline found in compliance frameworks like CIS, but also provide a quadratical improvement in execution time.

In an article called *Introducing a new quantum control for IT security*[1], I provided a detailed design of one of the first quantum controls: by performing a random walk over all edges of a directed graph simultaneously, it may be used in a lot of situations where **asserting reachability** is important. So, it is not only useful for IT security (network access control, IAM, provable security), but for other branches of engineering especially networking.

In this short technical paper, we detail an end-to-end implementation of the quantum reachability control on the simplest network topology: the 2-vertices directed graph:



Its Laplacian matrix is simply:

$$\begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}$$

Because the graph is directed, its matrix is not Hermitian: one needs to turn it into a Hermitian operator using an eigenbasis transformation described in *Quantum centrality testing on directed graphs via PT-symmetric quantum walks* [3].

Pseudo-Hermitian to Hermitian transformation

(The Python program is available here [5]).

We first calculate the eigenvectors φ_0 and φ_1 of the Laplacian:

$$\varphi_0 = (0, 1) \quad \varphi_1 = \left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \right)$$

We calculate the outer products and sum them (equation 19 in the paper):

$$\varphi_0 \times \varphi_0 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad \varphi_1 \times \varphi_1 = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \quad \sum_{i=0}^1 \varphi_i \times \varphi_i = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{3}{2} \end{pmatrix}$$

The (matrix) square root of this sum is the metric vector ν .

We left-multiply the Laplacian by η and right-multiply by the inverse of η to obtain the Hamiltonian (equation 9 in the paper):

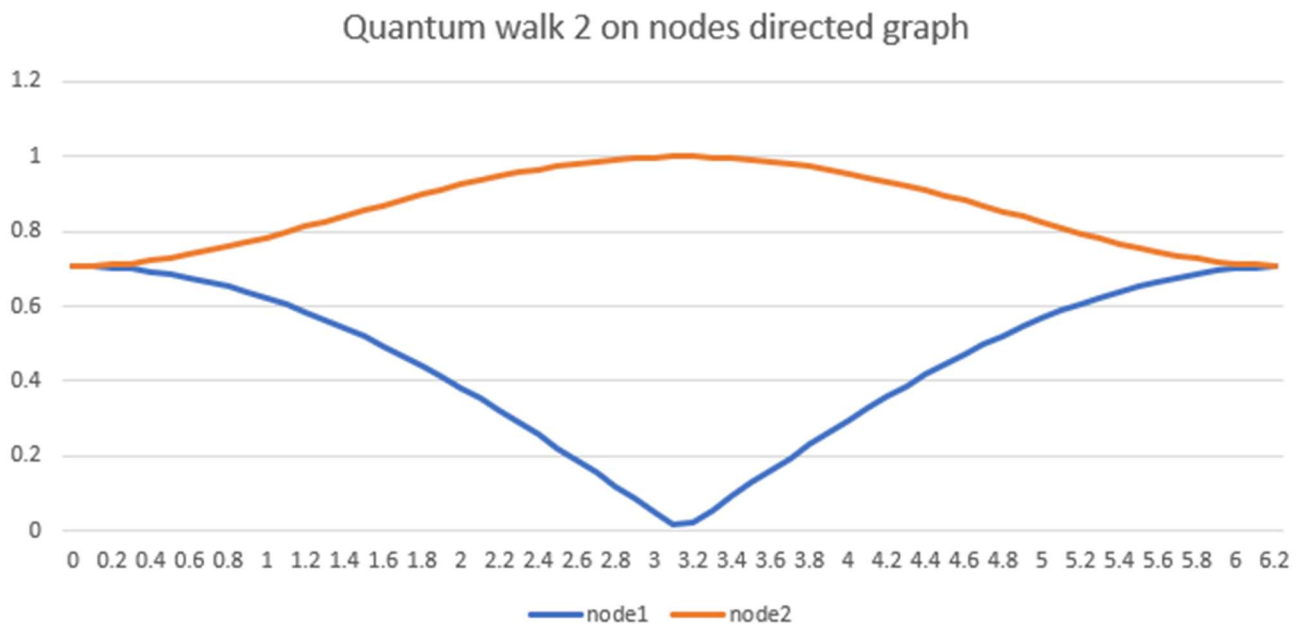
$$H = \eta L \eta^{-1}$$

The resulting Hamiltonian is:

$$\begin{pmatrix} \sqrt{2}+1 & -1 \\ -1 & \sqrt{2}-1 \end{pmatrix}$$

Quantum walks over the graph

If we set the initial state of a quantum walk to a single qubit in quantum superposition, running the Schrödinger equation with the Hamiltonian of our graph yields the following probability distribution:



We see that edge 1 is unreachable from edge 2 because of the null probability at time π modulo π .

Finding null probabilities is the silver bullet of the quantum reachability control: it states that all edges with a null probability are unreachable from some part of the graph.

How to run this control on a quantum circuit?

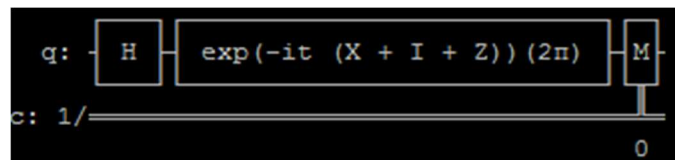
As explained in my article, we need to break down the Hamiltonian into simpler operators. Here is the most natural decomposition using the Pauli basis:

$$Z - X + I\sqrt{2}$$

Where I is the identity operator, Z and X correspond to Pauli- Z and Pauli- X .

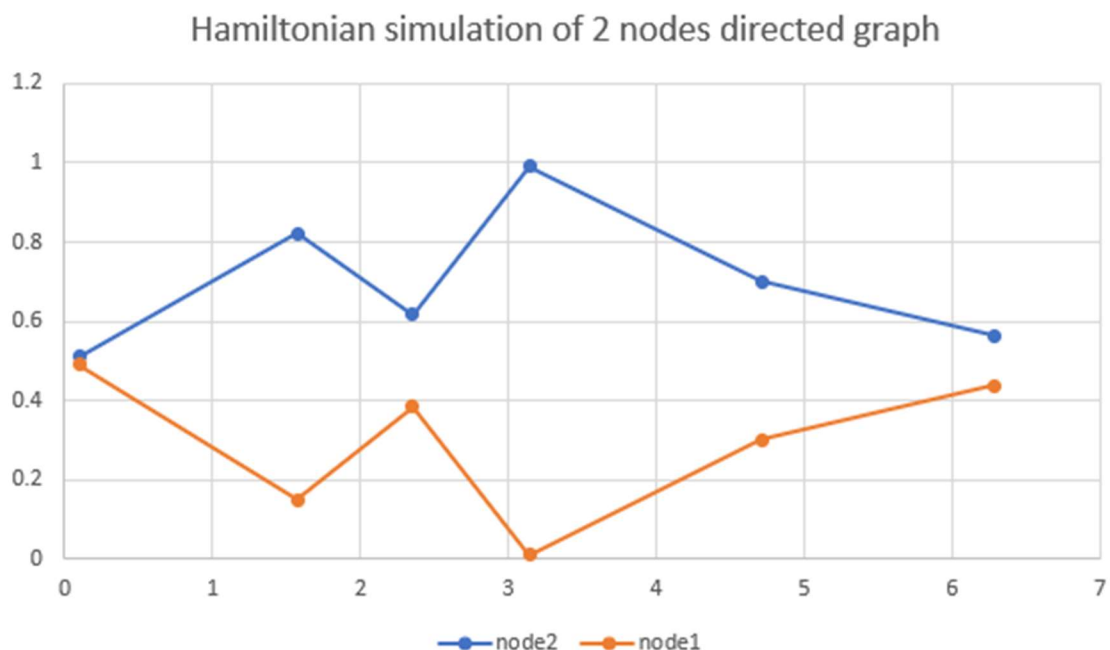
Building a qDRIFT [2] quantum circuit for this linear combination is very straightforward with *Qiskit* because it comes with the handy *PauliEvolutionGate*.

The circuit is available from my script `hamiltonianSimulation2nodes.py` [4]:



(Note that, as per *Qiskit*'s documentation, the printed circuit doesn't show the coefficients applied to X , Z and I)

Here is one of the Hamiltonian simulations I got with a small number of repetitions (`reps=3`). You will see that the probability distribution of the state vector roughly follows the predicted behavior from Schrödinger's equation:



There are several reasons why this sampling is rough and looks sketchy:

- qDRIFT performs a random, statistical sampling which "drifts" toward the Hamiltonian up to a certain precision which depends on the quantum computer capabilities in terms of circuits depth and decoherence;
- time discretisation errors grow as t increases;
- I have only sampled at times $0, \pi/2, \pi, 3\pi/2$ and 2π .

But still, running the control seems to let us correctly identify the null probability at time $t=\pi$

Conclusion

Quantum Computing is no more theoretical. The number of useful applications is growing every day.

This proof of concept shows the practical interest of running Hamiltonian simulations on directed graphs for IT security compliance. I hope it will raise the awareness and the interest of the cybersecurity community in Quantum Computing.

References

[1] Introducing quantum control for IT security, <https://www.quantumgrad.com/article/297>

[2] Earl Campbell, A random compiler for fast Hamiltonian simulation, <https://arxiv.org/abs/1811.08017>

[3] Joshua Izaac et al, Quantum centrality testing on directed graphs via PT-symmetric quantum walks, <https://arxiv.org/abs/1607.02673>

[4] <https://github.com/labyrinthinesecurity/quantumComputing/blob/main/hamiltonianSimulation2nodes.py>

[5] <https://github.com/labyrinthinesecurity/quantumComputing/blob/main/quantumWalk2nodes.py>