Quantum Computing PHYS-541, Project 2

Teacher : vincenzo.savona@epfl.ch
Assistant : sara.alvesdossantos@epfl.ch, david.linteau@epfl.ch, shao.chiew@epfl.ch

Simulation of Shor's factoring algorithm

We have studied in class *Shor's factoring algorithm*. The goal of the project is to code from scratch a highly optimized state-vector simulation of Shor's factoring algorithm, and study its computational complexity. **How big an integer N can you factor on your laptop?**

To simulate Shor's factoring you must keep in mind two important facts. First, there is a "semiclassical" version of the algorithm which uses the semiclassical QFT. It is sometimes called the one-qubit trick, because the phase register requires only one qubit [1, 2]. If you want to simulate big, you should use this variant of the algorithm. Second, the semiclassical version of Shor's algorithm is highly sparse: both the gates and the state vector are very sparse (many zeros and only few nonzero elements). If you want to simulate big, you should consider exploiting this feature[1]. To simplify the task, you may use a compiled version of the $c-U^{2^j}$ operators, which you may write as single unitary matrices acting on the whole state. This will be much simpler than actually writing the modular multiplication in terms of elementary gates, both in the compiled [3, 4] or bottom-up [1, 5, 6] approaches.

For the project, you are expected to:

1. Read and understand the chapter on Shor's factoring algorithm in the lecture notes, and possibly on other sources, and present to a sufficient level of detail how Shor's algorithms works.

2. Using the programming language of your choice (Python, Julia, Rust, Matlab, etc.) write a code that represents the state of the quantum computer as a vector, has functions to apply all quantum operations needed for the execution of the algorithm, and has functions to measure each qubit.

3. Test the code on your personal computer. As a reference, consider integers N given by the product of two subsequent safe primes. A safe prime p is a prime that can be expressed as $p = 2q + 1$, where q is also a prime. Integers $N = p_1 p_2$, where p_1 and p_2 are safe primes, have the property that if $mathrmgcd(x, N) = 1$, then x has a very large probability of having the largest possible period r with respect to the modular product. The first safe primes are

$$p = 5,\ 7,\ 11,\ 23,\ 47,\ 59,\ 83,\ 107,\ 167,\ 179,\ 227,\ 263,\ 347,\ 359,\ 383,\ 467,\ 479,\ 503,\ 563,\ 587,$$
$$719,\ 839,\ 863,\ 887,\ 983,\ 1019,\ 1187,\ 1283,\ 1307,\ 1319,\ 1367,\ 1439,\ 1487,\ 1523,\ 1619, \ldots$$

   How far can you go? How does the simulation time scale with N?

[1] Using a library for the simulation of sparse quantum circuits, such as qblaze, will not be accepted, as the goal is to write your own quantum circuit simulator.

References

[1] Stephane Beauregard. Circuit for Shor's algorithm using 2n+3 qubits, February 2003. arXiv:quant-ph/0205095.

[2] Igor L. Markov and Mehdi Saeedi. Constant-Optimized Quantum Circuits for Modular Multiplication and Exponentiation, April 2015. arXiv:1202.6614 [cs].

[3] Omar Gamel and Daniel F. V. James. Simplified Factoring Algorithms for Validating Small-Scale Quantum Information Processing Technologies, November 2013. arXiv:1310.6446 [quant-ph].

[4] Robert L. Singleton Jr. Shor's Factoring Algorithm and Modular Exponentiation Operators. Quanta, 12:41–130, September 2023. arXiv:2306.09122 [quant-ph].

[5] Thomas G. Draper. Addition on a Quantum Computer, August 2000. arXiv:quant-ph/0008033.

[6] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit, October 2004. arXiv:quant-ph/0410184.