Quantum Algorithms

James Yakura

Quantum physics is the physics of the very small and the very cold. Small amounts of matter, at low energies, can behave differently from the ways that are familiar to us. A quantum computer is a computer designed to exploit these behaviors, and a quantum algorithm is an algorithm designed to take advantage of a quantum computer's abilities.

To give an example of the differences between quantum and classical physics, classical objects are generally in only one place at a time. A quantum object, however, is simultaneously in a large number of places, and can even interact with itself. The object behaves as a wave rather than a particle, producing "ripples" that interfere with one another constructively and destructively. When something interacts with it, the wave collapses into a particle, with the wave's amplitude at a given point corresponding to the particle's likelihood of being at that point. (The reality is somewhat more complicated than this; treating amplitude as though it were purely probability can give strange results like negative or imaginary probabilities.)

Some of the best-known and most reliable uses for a quantum computer revolve around the classes of problems known as NP and NP-Complete. NP problems (which includes all NP-Complete problems) are, on a classical computer, easy to check but difficult to solve. One property of NP-Complete, though, is that any kind of NP problem can be translated into a NP-Complete problem. That is, on some level, breaking a hashed password is the same thing as packing the most valuable set of objects possible into a bin of limited size (the Knapsack Problem), which is the same thing as finding the shortest path through a graph that visits all nodes (the Traveling Salesman).

One of the properties of quantum systems is that the rules governing them are NP-Complete. Because of this, it is possible to convert any NP problem into a quantum system. Any possible state of the quantum system will be equivalent to some potential solution to the problem, and any correct solution will correspond to a valid state of the quantum system.

In other words, in order to find a path through a graph that visits each node exactly once, you can do this:

1. Translate the problem into a quantum system.
2. Set up the quantum computer to contain that system.
3. Allow the system to develop.
4. Measure the system's state.
5. Check whether the problem is solved. If not, repeat from step 2.

This will at least drastically reduce the number of possibilities that the computer must check, and possibly arrive at the correct solution on the first attempt.

An example of this approach is Shor's algorithm, which is used for finding the factors of numbers. This is done by translating the problem into another problem called phase estimation, constructing a quantum system that solves the phase estimation problem (and in the process, taking advantage of a more minor increase in speed from another algorithm called the quantum Fourier transform), and then translating that result back to a factoring problem.

(A related application is using a quantum computer to simulate other quantum systems. This works in roughly the same way as using a classical computer to simulate classical systems.)

Another use for quantum computers revolves around the randomness inherent to them. Most computing problems assume that the answer must be correct the first time. But what if you could be wrong, but only a certain amount of the time?

Quantum Algorithms

James Yakura

Since a quantum system exists in multiple states simultaneously, it is possible for a quantum computer to exist in multiple states simultaneously, and then collapse into one at random. A trivial application of this is to generate truly random numbers rather than the pseudo-random numbers that a classical computer generates deterministically; this can be used in cryptography to generate one-time pads, by an operating system to resolve resource deadlocks, in games to create truly fair and random dice or shufflings, or for anything else that requires random numbers. (There are also classical ways of producing truly random numbers; for example by amplifying line noise, or by measuring something chaotic and with unknown inputs such as network traffic. Even obtaining quantum random numbers does not require a dedicated computer; single photons and radioactive atoms behave randomly.)

A more advanced use is in *probabilistic algorithms*. A probabilistic algorithm is one that includes some random element, which makes it sometimes wrong, but right *often enough*. A probabilistic algorithm trades certainty for speed.

The way that a probabilistic algorithm works is that it branches nondeterministically at certain points. That is, it runs in parallel, testing multiple possible paths that it could run. Once it has finished all paths, it chooses a path at random from those it has found. There are methods for making certain outcomes of a probabilistic algorithm more likely than others.

A classical computer running a probabilistic algorithm is limited to a single path, at least without trying to simulate a nondeterministic machine (which sacrifices the speed advantage of probabilistic algorithms). For classical computers, the only way to improve the chances of a correct answer is to weight the branching decisions so that the computer will follow the paths most likely to be correct. (However, if you know which paths are “better” in advance, you would probably be better off using an ordinary deterministic program, and checking other paths only if the most likely path fails.)

A quantum computer, however, is able to effectively change its branching probabilities retroactively to some extent. Once a path has determined itself to be correct, it is possible to program it to make itself more likely to be chosen. Likewise, once a path has determined itself to be incorrect, it is possible to program it to make itself less likely. (Another way of looking at this is that incorrect answers are made to “cancel out” one another.) This cannot completely eliminate the possibility of an incorrect path being chosen, but it can make the program right more often than not.

Note that this approach is very much not the same as simply trying all possible options and picking the correct one! While the computer does try all possible options, the “picking the correct one” part can take a long time and is not certain.

One example of this approach is Grover’s algorithm, which can search a space for items matching some criteria. It does this by constructing a *quantum oracle operator*: a function that it can run repeatedly to make correct answers more likely and incorrect answers less likely. In less than linear time, Grover’s algorithm can be nearly certain, even though the oracle does not have the answer.

As mentioned earlier, other quantum algorithms exist that are faster than their classical equivalents. The Fourier transform is one example; its increase in speed is logarithmic. Spanning trees can also be calculated more quickly on a quantum computer than a classical one, though this uses Grover’s algorithm.

There are other uses for quantum algorithms as well. Quantum systems exhibit a number of other strange properties, and quantum computers could theoretically be designed to use them.

A quantum system breaks down when something outside of it interacts with it, and its state cannot be observed without such interaction. This can create messages that have “tamper-evident seals”, so that they are changed if any third party reads them. This field of *quantum cryptography* would allow a cryptographic system to verify its confidentiality and integrity; this requires physically sending a quantum system to another computer, but allows messages to be kept secure. While current network infrastructure cannot send quantum systems, this has already been developed at a small scale and is in use by the Department of Defense.

Some computing problems involve something called an *oracle*: an automated hint-giver that cannot give an answer, but can give parts of the answer or answer certain questions about it. Certain problems have been shown to be solvable in a single oracle query by a quantum computer, that would be completely unsolvable by a classical computer even with an oracle. One such problem is deciding whether one black-box function is a transform of another: a classical computer would need to either know the function definitions or compare the functions at every possible point, but a quantum computer can solve the problem using its oracle only once.

Despite the promise of quantum computing, it still has certain limitations in common with classical computing, and other limitations all its own.

The largest limitation on using quantum computing in the ways previously described is that quantum computers are difficult to build and expensive to run. Since the quantum computer must be a single system, and there are limits to how massive a system can be before it starts losing its quantum properties, this limits the size of the computer; a mere 50-qubit computer is considered quite impressive. In addition, because quantum systems require low energies, the computer must be kept far colder than a classical computer’s fans could manage. (In fact, air cooling would actually work against this goal; most quantum computer designs only function at temperatures of a few millikelvin, and only one design has been proposed that might work at room temperature.)

There are further problems when using a quantum computer to carry out a probabilistic algorithm. Since probabilistic algorithms cannot guarantee a correct answer, they should not be used when the answer is hard to check, or when the answer must be correct on the first attempt.

A common misconception is that quantum computers process every possibility at once, and then select the correct one. This is a serious misunderstanding; a possibility when carrying out a probabilistic program is chosen *at random*. A quantum computer is not the same thing as a nondeterministic computer, and has no way to actually pick which possibility it will choose. Because of this, even with easy access to quantum computing, there are problems that cannot be easily solved, and algorithms that are fast and accurate on a nondeterministic computer can be either slow or unreliable on a quantum computer.

More than this, most quantum algorithms are to some extent probabilistic. Since a quantum system cannot be reduced to only a single state, there is nearly always some amount of error in the calculation. Even quantum algorithms that do not take advantage of the probabilistic nature of quantum computation must be checked on a classical computer if the result must be certain. This is not a problem when using a quantum computer to work NP problems, since NP problems can be checked easily (by definition), but there are other problems that cannot be checked quickly.

Likewise, certain problems are totally unsolvable by any kind of computer. No computer with finite speed can prove whether an arbitrary program will ever stop. No computer will ever be able to prove that an arbitrary program will always produce exactly the intended result. Even quantum computers cannot communicate faster than light, or receive results from their own future.

Quantum Algorithms
James Yakura

References

Aditya, J., & Rao, P. S. (n.d.). *Quantum cryptography - stanford computer science*. Stanford University. Retrieved December 7, 2021, from https://cs.stanford.edu/people/adityaj/QuantumCryptography.pdf.

Ball, P. (n.d.). *The ERA of quantum computing is here. outlook: Cloudy*. Quanta Magazine. Retrieved December 7, 2021, from https://d2r55xnwy6nx47.cloudfront.net/uploads/2018/01/the-era-of-quantum-computing-is-here-outlook-cloudy-20180124.pdf.

Blain, L. (2021, September 28). *Quantum computing hits the desktop, no cryo-cooling required*. New Atlas. Retrieved December 7, 2021, from https://newatlas.com/quantum-computing/quantum-computing-desktop-room-temperature/.

Hartnett, K. (2020, August 28). *Finally, a problem that only quantum computers will ever be able to solve*. Quanta Magazine. Retrieved December 7, 2021, from https://www.quantamagazine.org/finally-a-problem-that-only-quantum-computers-will-ever-be-able-to-solve-20180621/.

J., A., Adedoyin, A., Ambrosiano, J., Anisimov, P., Bärtschi, A., Casper, W., Chennupati, G., Coffrin, C., Djidjev, H., Gunter, D., Karra, S., Lemons, N., Lin, S., Malyzhenkov, A., Mascarenas, D., Mniszewski, S., Nadiga, B., O'Malley, D., Oyen, D., … Lokhov, A. Y. (2020, March 18). *Quantum algorithm implementations for Beginners*. arXiv.org. Retrieved December 7, 2021, from https://arxiv.org/abs/1804.03719.

Kaur @DashveenjitK , D. (2020, December 15). *IBM 'super-fridge' aims to solve quantum computer cooling problem*. TechHQ. Retrieved December 7, 2021, from https://techhq.com/2020/12/ibm-super-fridge-aims-to-solve-quantum-computer-cooling-problem/.

The Physics arXiv Blog. (2014, April 6). *The astounding link between the P≠NP problem and the quantum nature of universe*. Medium. Retrieved December 7, 2021, from https://medium.com/the-physics-arxiv-blog/the-astounding-link-between-the-p-np-problem-and-the-quantum-nature-of-universe-7ef5eea6fd7a.

Public Broadcasting Service. (2014, April 11). *Nova - Is there anything beyond quantum computing?* PBS. Retrieved December 7, 2021, from https://www.pbs.org/wgbh/nova/article/is-there-anything-beyond-quantum-computing/.