Quantum Computation

(Lecture QC-3: Quantum Algorithms B)

Luís Soares Barbosa

Quantum Computing

Universidade do Minho, 2019

Quantum algorithms

Principles (review)

- Keep separate input-output registers (standard practice from reversible classic computation)
- Fix initial setting (preparation): typically the qubits in the initial classical state are put into a superposition of many states;
- Transform, through unitary operators applied to the superposed state;
- Measure, i.e. projecti onto a basis vector associated with a measurement tool.

Quantum algorithms

U_f is specified as a reversible (classical) transformation taking typically computational-basis states into computational-basis states. Its extension

to arbitrary complex superpositions of computational-basis states is necessarily unitary.

$$U_f(|x\rangle_n|y\rangle_m) \;=\; (|x\rangle_n|y \oplus f(x)\rangle_m$$

yielding, for $y = 0$

$$U_f(|x\rangle_n|0\rangle_m) \;=\; (|x\rangle_n|f(x)\rangle_m$$

Quantum algorithms

Build a superposition

$$
\begin{aligned}
(H \otimes H)(|0\rangle \otimes |0\rangle) &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\
&= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\
&= \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle)
\end{aligned}
$$

which generalises to

$$
H^{\otimes n}|0\rangle_n = \frac{1}{\sqrt{2^{n/2}}} \sum_{0 \leq x \leq 2^n} |x\rangle_n
$$

Quantum algorithms

Quantum 'parallelism'

$$
\begin{aligned}
U_f(H^{\otimes n} \otimes I_m)(|x\rangle_n|0\rangle_m) &= \frac{1}{2^{n/2}} \sum_{0 \leq x \leq 2^n} U_f(|x\rangle_n|0\rangle_m) \\
&= \frac{1}{2^{n/2}} \sum_{0 \leq x \leq 2^n} |x\rangle_n|f(x)\rangle_m
\end{aligned}
$$

The 'quantum parallelism' miracle is, to a great extent, only apparent. Actually, the result of the calculation is not 2^n evaluations of f: those evaluations characterize the form of the state that describes the output of the computation.

Quantum algorithms

What works indeed?

- What remains is the fact that the random selection of the x, for which $f(x)$ can be learned, being made only after the computation has been carried out.
- Note that asserting that the selection was made before the computation corresponds to look at a superposition as merely a probabilistic phenomenon (i.e. the qubit described by a superposition is actually in one or the other of the basis states).
- Further computation makes possible to extract useful information about relations between the values of x for several different values of x, which a classical computer could get only by making several independent evaluations.

Quantum algorithms

What works indeed?

- The price to be paid is the loss of the possibility of learning the actual value $f(x)$ for any individual x — cf Heisenberg uncertainty principle.
- cf the mistaken view that the quantum state encodes a property inherent in the qubits: it rather encodes only the possibilities available for the extraction of information from them.

The Bernstein-Vazirani algorithm

The problem

Let w be an unknown non-negative integer less than 2^n and consider a function $f(x) = w \cdot x$, where

$$w \cdot x \;=\; w_0 x_0 \oplus w_1 x_1 \oplus w_2 x_2 \oplus \cdots$$

How many times one has to call f to determine the value of the integer w?

- Classically, n times: the n values $w \cdot 2^m$, for $0 \leq m < n$.
- In a quantum computer a single invocation is enough, regardless of the number n of bits.

The Bernstein-Vazirani algorithm

- Prepare the single qubit output register as $H|1\rangle$ since oracle U_f applied to $|x\rangle_n|y\rangle_1$ flips the value y of the output register iff $f(x) = 1$. Thus,

$$U_f\, |x\rangle_n \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \;=\; (-1)^{f(x)}\, |x\rangle_n \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

converting a bit flip to an overall change of sign.

The Bernstein-Vazirani algorithm

- Superposition

$$
\begin{aligned}
H^{\otimes n}|x\rangle_n &= \frac{1}{2^{n/2}} \sum_{y_{n-1}=0}^{1} \cdots \sum_{y_0=0}^{1} (-1)^{\sum_{j=0}^{n-1} x_j y_j} |y_{n-1}\rangle \cdots |y_0\rangle \\
&= \frac{1}{2^{n/2}} \sum_{y=0}^{2_n-1} (-1)^{x \cdot y} |y\rangle_n
\end{aligned}
$$

cf

$$
H|x\rangle_1 = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^x|1\rangle) = \frac{1}{\sqrt{2}} \sum_{y=0}^{1} (-1)^{xy}|y\rangle
$$

The Bernstein-Vazirani algorithm

Putting everything together,

$$
\begin{aligned}
&(H^{\otimes n} \otimes I) U_f (H^{\otimes n} \otimes H)\, |0\rangle_n |1\rangle_1 \\
&= (H^{\otimes n} \otimes I) U_f \left(\frac{1}{2^{n/2}} \sum_{0 \le x \le 2_n - 1} |x\rangle \right) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \\
&= \frac{1}{2^{n/2}} \left(H^{\otimes n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \right) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \\
&= \frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{y=0}^{2^n-1} (-1)^{f(x) + x \cdot y} |y\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \\
&= |w\rangle_n |1\rangle_1
\end{aligned}
$$

because

$$
\sum_{x=0}^{2^n-1} (-1)^{w \cdot x} (-1)^{y \cdot x} = \prod_{j=1}^{n} \sum_{x_j=0}^{1} (-1)^{(w_j + y_j) x_j}
$$

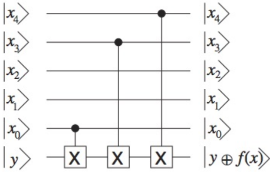
The Bernstein-Vazirani algorithm: another explanation

Some oracles can be implemented by simple circuits.

- In this case the action of U_f on the computational basis is to flip the 1 qubit output register once, whenever a bit of x and the corresponding bit of w are both 1.
- Put one cNOT for each nonzero bit of w, controlled by the qubit representing the corresponding bit of x.
- Their combined effect on every computational basis state is precisely that of U_f.

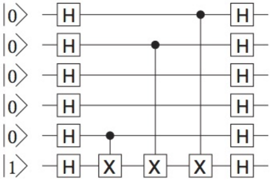
The Bernstein-Vazirani algorithm: another explanation

Example of the encoding for $w = 11001$



The Bernstein-Vazirani algorithm: another explanation

Envelop U_f into the algorithm



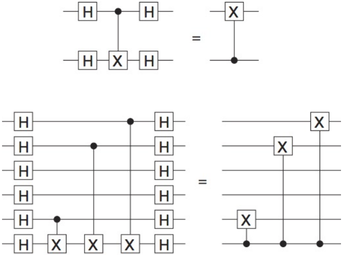
The effect is to convert every cNOTgate in the equivalent representation of U_f from C_{ij} to

$$C_{ji} = (H_i H_j) C_{ij} (H_i H_j)$$

reversing the target and control qubits.

The Bernstein-Vazirani algorithm: another explanation

Because: Law and its application



The Bernstein-Vazirani algorithm: another explanation

Thus

- After the reversal, the output register controls every one of the cNOT gates, and since the state of the output register is $|1\rangle$, every one of the NOT operators acts.
- That action flips just those qubits of the input register for which the corresponding bit of w is 1.
- Since the input register starts in the state $|0\rangle_n$, this changes the state of each qubit of the input to $|1\rangle$, iff it corresponds to a nonzero bit of w.
- Thus, in the end, the state of the input register changes from $|0\rangle_n$ to $|w\rangle_n$.

Simon's algorithm

The problem

Determine the period z of a function f from n to $n-1$ bits periodic under $\oplus$:

$$f(x \oplus z) = f(x)$$

Simon's algorithm, classically

- Compute f for sequence of values until finding a value x_j such that $f(x_j) = f(x_i)$ for a previous x_i. Then

$$z \;=\; x_j \oplus x_i$$

- At any previous stage, if this procedure has picked m different values of x, then one concludes that $z \neq x_j \oplus x_i$ for all such values.
- Thus, at most

$$\frac{1}{2}m(m-1)$$

  possible values for z have been discarded (vs $2^n - 1$ possible values for z).
- The procedure is unlike to succeed until m becomes of the order of $2^{n/2}$ — the execution time grows exponentially with the number of bits n.

Simon's algorithm, going quantum

- Transform the state of the input register into the uniformly weighted superposition of all possible inputs by the application of the usual recipe ($H^{\otimes n}$);
- Apply U_f obtaining

$$\frac{1}{2^{n/2}} \sum_{0 \leq x \leq 2^{n-1}} |x\rangle |f(x)\rangle$$

- Measure the output register. Since f appears in two terms in the expression above that have the same amplitudes, by the generalized Born rule, the input register will be left in the state

$$\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus z\rangle)$$

for that value of x_0 for which $f(x_0)$ agrees with the random value of f given by the measurement.

Recall: the generalized Born rule

... applies on measuring a single one of $n+1$ qubits.

The joint state can be taken as

$$|\Psi\rangle \;=\; \alpha_0|0\rangle|\phi_0\rangle_n + \alpha_1|1\rangle|\phi_1\rangle_n$$

with $|\alpha_0|^2 + |\alpha_1|^2 = 1$, following from the general fom

$$|\Psi\rangle_{n+1} \;=\; \sum_{x=0}^{2^{n+1}-1} \gamma(x)|x\rangle_{n+1}$$

Recall: the generalized Born rule

Thus,

$$|\Psi_0\rangle_n = \frac{1}{\alpha_0}\sum_{x=0}^{2^n-1}\gamma(x)|x\rangle_n \quad |\Psi_1\rangle_n = \frac{1}{\alpha_1}\sum_{x=0}^{2^n-1}\gamma(2^n+x)|x\rangle_n$$

$$\alpha_0^2 = \sum_{x=0}^{2^n-1}|\gamma(x)|^2 \quad \alpha_1^2 = \sum_{x=0}^{2^n-1}|\gamma(2^n+x)|^2$$

i.e., if one measures only the single qubit whose state symbol is explicitly separated out from the others in $n+1$ qubits state, then this single measurement gate will indicate x (0 or 1) with probability $|\alpha_x|^2$, after which the $n+1$ qubits state can be taken to be the product state

$$|x\rangle|\Psi_x\rangle_n$$

Simon's algorithm, going quantum

- A superposition of two computational-basis states, associated with two n-bit integers, that differ by z was computed. But we are not able to known those two integers ...
- ... it also does not help to run the algorithm many times, which is likely to get states of a similar form for different random values of x_0...
- A direct measurement will just produce a random number (cf x_0 or $x_0 \oplus z$): however the number z one is interested in appears only in the relation between those two random numbers, only one of which one can learn.

Simon's algorithm, going quantum

How to extract this relation (i.e. number z)?

Recall $H^{\otimes n}|x\rangle_n$, from the Bernstein-Vazirani algorithm. Thus,

$$\begin{aligned} H^{\otimes n}\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus z\rangle) &= \frac{1}{2^{(n+1)/2}}\sum_{y=0}^{2^n-1}\left((-1)^{x_0\cdot y)} + (-1)^{(x_0\oplus z)\cdot y}\right)|y\rangle \\ &= \frac{1}{2^{(n-1)/2}}\sum_{z\cdot y=0}(-1)^{(x_0\oplus y)}|y\rangle \end{aligned}$$

because, since $(-1)^{(x_0\oplus z)\cdot y} = (-1)^{x_0\cdot y} = (-1)^{z\cdot y}$, the coefficient of $|y\rangle$ above becomes:

$$\begin{cases} 0 & \Leftarrow z\cdot y = 1 \\ 2(-1)^{x_0\cdot y} & \Leftarrow z\cdot y = 0 \end{cases}$$

Simon's algorithm, going quantum

How to extract this relation (i.e. number z)?

The sum in the red expression

$$H^{\otimes n}\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus z\rangle) = \frac{1}{2^{(n+1)/2}}\sum_{y=0}^{2^n-1}\left((-1)^{x_0\cdot y} + (-1)^{(x_0\oplus z)\cdot y}\right)|y\rangle$$
$$= \frac{1}{2^{(n-1)/2}}\sum_{z\cdot y=0}(-1)^{(x_0\oplus y)}|y\rangle$$

is restricted to those y for which $z \cdot y = 0$.
Thus, measuring the input register, yields, with equal probability, any of the values of y for which $z \cdot y = 0$, i.e. for which

$$\sum_{i=0}^{n-1} y_i z_i = 0 \pmod 2$$

Simon's algorithm, going quantum

Are we done?

Each invocation of U_f yields a random y satisfying $z \cdot y = 0$, which allows one to determine z with high probability with not many more than n iterations.

Simon's algorithm, going quantum

Are we done?

- A single invocation of U_f gives one such y and thus a nontrivial subset of the n bits of z whose modulo-2 sum vanishes.
- One of those bits is entirely determined by the others in the subset, which cuts the number of possible choices for z in half: from $2^n - 1$ to $2^{n-1} - 1$ (the -1 comes from assumption $z \neq 0$).
- This assertion above is probabilistic: there is a very small probability, $\frac{1}{2^{n-1}}$, that $y = 0$ ...
- Repeating the process there is a very high probability the getting a new value y, different from 0 and the ones already found, therefore yielding a new nontrivial relation among the bits of z which reduces again the number of candidates by ar factor of 2.

Simon's algorithm, going quantum

It can be shown that with $n + d$ iterations, the probability of acquiring enough information to determine z is given by

$$\left(1 - \frac{1}{2^{n+d}}\right)\left(1 - \frac{1}{2^{n+d-1}}\right)\cdots\left(1 - \frac{1}{2^{d+2}}\right) < 1 - \frac{1}{2^{d+1}}$$

Conclusion

With a relatively small d, period z is found with very high probability, no matter how large n may be.

Going further: Shor and beyond

The essence of Simon's algorithm, common to many other quantum algorithms resides in combining

- Playing with interference and superposition to acquire fundamental information to solve the problem,
- with specific mathematical arguments to confirm that the output of the quantum procedure does indeed provide the needed information and to fix the (probabilistically relevant) number of iterations.

Going further: Shor and beyond

Shor algorithm (1994)

Factoring

$$N = p \cdot q$$

for p, q very big primes, is closely tied to the ability to find the period of

$$n^x \bmod N$$

for integers n that do not share factors with N.

Combines efficient period-finding quantum procedures with non trivial results in number theory.
(cf details on Assis Azevedo talk on Q-Days)

Going further: Shor and beyond

- Very hard problem: dealing with functions on the integers whose values within a period are virtually random from one integer to the next, and therefore give no hint of the value of the period itself.
- Major speed-up (scales slightly above n^3, if n is the number of bits in the binary representation of the period).
- Huge impact
  (cf JMValena talk on Q-Days on post-quantum crypto).

Which problems a Quantum Computer can solve?

No magic ...

- One can store and manipulate a huge amount of information in the states of a relatively small number of qubits,
- ... but measurement will pick up just one of the computed solutions and colapse the whole (quantum) state

... but engineering:

As amplitudes interfere, a suitably engineered algorithm will ensure that computational paths leading to a wrong answer would cancel out, and the ones leading to a correct answer would reinforce, thus boosting the probability of finding them when the state is measured at the end.

Complexity classes

P

Problems that can be solved efficiently, in polynomial time.
Example: *Given a road map showing n towns, can you get from any town to every other town?*

NP

Problems whose solutions, once found, can be recognized as correct in polynomial time — even though the solution itself might be hard to find.
Example: *Given a map with thousands of islands and bridges, find a tour that visits each island once*

NP complete

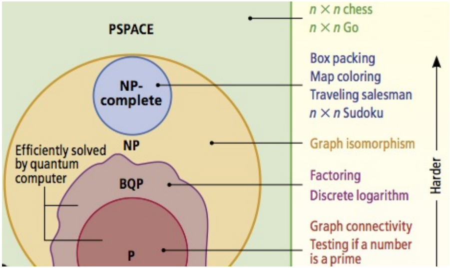
Problems that if an efficient solution to one of them existed, it would provide an efficient solution to all NP problems.
Example: *Given a map, can you color it using only three colours so that no neighboring countries are the same colour?*

The BQP class

Bounded-error Quantum Polynomial time: contains all the decision problems that quantum computers can solve efficiently.

P $\subseteq$ BQP

Quantum computers can solve all the problems that classical computers can solve (Bernstein and Vazirani, 1993).

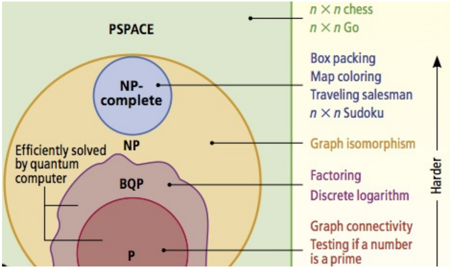


The BQP class

BQP cannot extend outside PSPACE, which also contains all the NP problems.

PSPACE

PSPACE problems are those that a conventional computer can solve using only a polynomial amount of memory but possibly requiring an exponential number of steps.



Which problems a Quantum Computer can solve?

- 1994: Peter Shor's factorization algorithm (exponential speed-up),
- 1996: Grover's unstructured search (modest, quadratic speed-up, most relevant in practice),
- 2018: Advances in hash collision search, i.e finding two items identical in a long list — serious threat to the basic building blocks of secure electronic commerce.
- 2019: Announced first BQP algorithm with no classical solution (based on oracle estimation)
- ...
- ... but no quantum algorithm is known to solve a NP-complete problem.

An efficient algorithm for an NP-complete problem would mean NP = P.

What we (think we) know?

- A quantum algorithm capable of solving NP-complete problems efficiently would, similarly to what happens in Simon's or Shor's algorithms, have to resort and exploit the problems' structure,
- Achieve an exponential speedup by treating the problems as structureless black boxes, consisting of an exponential number of solutions to be tested in parallel, is an illusion.
- Recently research has shown that modest, but often relevant speedups are the limit for many problems such as searching a list, counting ballots in an election, finding the shortest route on a map, and playing games of strategy such as chess or Go.
- If quantum computers ever become a reality, the killer app for them will be on simulating quantum physics — key for chemistry, nanotechnology, etc.