

CS290A/Phys250, Spring 2007:

# Quantum Information and Quantum Computation

Wim van Dam

Harold Frank Hall, Room 2151

vandam@cs

[http://www.cs.ucsb.edu/~vandam/teaching/s07\\_CS290/](http://www.cs.ucsb.edu/~vandam/teaching/s07_CS290/)

# About the Final Paper

**There will be no final examination coming Monday.**

**The final paper is due via email on Friday, June 15.**

The crucial characteristics are:

- It should not be too long, aim for 5 pages of text.
- The tone should be that of a scientific article: it should contain the standard components like Definitions, Theorems, Proofs and a Bibliography.
- Show that you understand and can work with the central ideas of the course, both qualitatively and quantitatively: when writing about experiments, do not forget to include some theory; when writing about applications of the HSP, do not forget to include a quantum algorithm for the HSP, et cetera.

# Limits on Quantum Searching

[Grover] showed that searching a database of size  $N$  can be done with  $O(\sqrt{N})$  quantum queries.

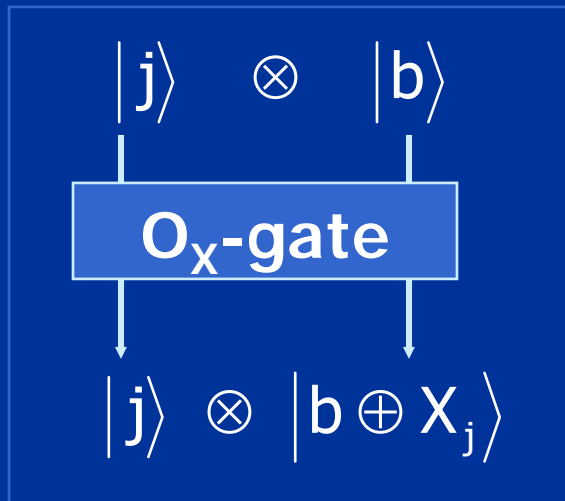
[Bennett, Bernstein, Brassard & Vazirani] showed (earlier) that  $\Omega(\sqrt{N})$  queries are required.

Note that a result of  $O(\log N)$  quantum queries would show that we can solve all **NP** problems in quantum-polynomial time. BBBV shows that life is not that easy.

This is typical: Quantum computing is no snake oil.

# Quantum Querying Functions

We assume that we have the network component:



Let  $X \in \{0,1\}^N$  be unknown and  $O_x$  is the 'oracle' that we have to query to know individual bits of  $X$ .

We want to evaluate a function  $F$  on  $X \in \{0,1\}^N$

**For which  $F$  do we need how many queries to  $O_x$  to get to know  $F(X)$  (approximately, probabilistically)?**

# Quantum Query Results for Functions $F$ on $\{0,1\}^N$

**Searching:** " $F(X) = X_1 \text{ OR } X_2 \text{ OR } \dots \text{ OR } X_N$ " ?

Grover's probabilistic quantum search:  $\Theta(\sqrt{N})$   
versus classical, probabilistic  $\Theta(N)$

**Parity:** " $F(X) = X_1 + \dots + X_N \text{ mod } 2$ " ?

Quantum, deterministic:  $\frac{1}{2} \cdot N$

Classical, probabilistic:  $N$

**Interrogation:** " $F(X) = [X_1, \dots, X_N]$ " ?

Probabilistic quantum:  $\frac{1}{2}N + O(\sqrt{N})$

instead of classical, probabilistic:  $N$

# Proving Lower Bounds

The lower bounds on searching and so on, prove that if we ignore the internal structure of the function  $F$ , we cannot get an exponential speedup the easy way.

The period finding algorithm shows that sometimes we can exploit the structure of  $F$ , to get a super-polynomial speed up.

We do not think that the search problem of **NP**-complete functions allows such a speed up.

But how do we prove those kinds of lower bounds to begin with?

# The Effect of One Query

When performing one oracle query to  $O_X$ ,  
we have the evolution from the input superposition

$$\sum_j a_{j,0} |j,0\rangle + a_{j,1} |j,1\rangle$$

to the output superposition

$$\sum_j ((1 - X_j) a_{j,0} + X_j a_{j,1}) |j,0\rangle + ((1 - X_j) a_{j,1} + X_j a_{j,0}) |j,1\rangle$$

Note that if  $a_{j,0}$  was a degree  $D$  polynomial in  $X_1, \dots, X_N$ ,  
then the 'new' amplitude of  $|j,0\rangle$  has at most degree  $D+1$ .

# Amplitudes as Polynomials

Unitary transformations that do not involve  $O_X$  give linear combinations of the existing amplitudes, and do not change the degree (in  $X$ ) of the polynomials of the amplitudes.

As we start out with a  $X$ -independent input state (degree = 0), after  $k$  queries to  $O_X$  the amplitudes of the computational basis states can all be expressed as multivariate polynomials in  $X_1, \dots, X_N$  with complex coefficients of total degree not more than  $k$ .

When observing the final output state the probabilities are given by  $|a(X)|^2$ , hence these probabilities are polynomials in  $X$  with real coefficients and of degree no more than  $2k$ .

# Polynomial Method

**Key Idea:** When calculating a property  $F:\{0,1\}^N \rightarrow \{0,1\}$  we want a probability  $p:\{0,1\}^N \rightarrow [0,1]$  with  $F(X)=p(X)$  for all  $X$ .

When performing  $k$  quantum queries to  $O_x$  we get a probability function  $p$  that is a polynomial of degree no more than  $2k$ .

If  $F$  is a function that has a necessary high degree  $\deg(F)$ , then we need at least  $k = \frac{1}{2}\deg(F)$  queries to the oracle.

In other words, this  $\frac{1}{2}\deg(F)$  is a quantum lower bound on the query complexity of the function  $F$  for quantum algorithms that calculate  $F(X)$  deterministically (without errors).

# Some Degrees

The parity function XOR can be represented by the polynomial  $\text{XOR}(X) = \frac{1}{2} - \frac{1}{2}(1-2X_1)(1-2X_2)\dots(1-2X_N)$  of degree  $N$ .

One can show that this is optimal, hence we do indeed need  $N/2$  quantum queries to calculate the parity of  $X$ .

For the OR function we have  $F(X) = 1 - (1-X_1)(1-X_2)\dots(1-X_N)$ , and this  $\text{deg}(\text{OR}) = N$  is optimal as well: we need at least  $N/2$  quantum queries to calculate  $\text{OR}(X)$ .

But Grover's search did it with  $\sqrt{N}$  queries; what went wrong?

Answer: Grover only *approximates* the OR function.

# Approximate Polynomials

If we want to approximate a function  $F:\{0,1\}^N \rightarrow \{0,1\}$ , then we are looking for a probability function  $p:\{0,1\}^N \rightarrow [0,1]$  such that for all  $X \in \{0,1\}^N$  we have  $|F(X) - p(X)| \leq 1/3$ .

Analyzing the minimal degree of polynomials  $p$  approximating the function  $F$  is significantly more challenging, but can be done.

For the OR function one can show  $\deg(\sim\text{OR}) \in \Theta(\sqrt{N})$ , hence the square root speed-up that we know is optimal.

Also, one can show  $\deg(\sim\text{XOR}) = N$ , hence  $k=N/2$  is optimal.

# Beyond Polynomials

The analysis of minimal degrees of Boolean functions has been done both in the context of classical and quantum queries.

A central notion is that of 'block sensitivity'  $bs(F)$ , which quantifies how often  $p$  has to switch value on  $\{0,1\}^N$ .

Using this  $bs(F)$  one can show that for every possible  $F$  the difference between the quantum and classical query complexity is at most sextic, hence  $\#C\text{-queries} \in O((\#Q\text{-queries})^6)$ .

The suspicion is that this is not optimal and that the quadratic speed-up is the best one can get for any  $F$ .

# Back to State- of-the-Art Real Quantum Algorithms

# Recovering from the Polynomial Method

The polynomial method shows that for any Boolean function  $f:\{0,1\}^N\rightarrow\{0,1\}$  and hidden string  $X\in\{0,1\}^N$ , the difference between the quantum and the classical query complexity for evaluating  $f(X)$  is at most a degree 6 polynomial. Hence, in this setting there can not be an exponential quantum speed-up.

But what about factoring and discrete logarithms?

That these are not exactly Boolean functions or black-box problems is not crucial: they rely on the period finding problem.

The hidden information  $X$  for period finding is the periodic function and  $f(X) = r$ . The crucial difference is that we only require the algorithm to work on proper inputs. If  $X$  is not periodic, then we do not care about  $f(X)$ .

# Designing Quantum Algorithms with Exponential Speed-Ups

## Black-box problems with exponential speed-up:

- Design a partial function  $f$  such that  $f(X)$  can be calculated with a number of quantum queries that is exponentially smaller than the required number of classical queries.
- Take care that the computation around the queries requires only polynomial time as well.

Example: Given an 'injective' string  $X \in S^N$  and a  $s$ -shifted version  $Y$  with  $Y_j = X_{j+s \bmod N}$ , determine the hidden shift  $0 \leq s < N$ .

The classical query complexity of this problem is  $\text{poly}(N)$ .

The quantum query complexity of this is only  $\text{poly}(\log N)$ .

But...we do not know how to do the post-processing after the quantum queries in a time-efficient  $\text{poly}(\log N)$  manner.

# Finding New Natural Problems

**There are no black boxes in reality.**

To mold a black box problem into a natural one you have to find a function that acts as a black box:

- It should have a short description
- It should be possible to implement it in an efficient and coherent way so that we can use it in a quantum circuit.

Typical example: The  $f(x) = a^x \bmod N$  function of Shor.

Another example: The function  $f(\pi) = \pi(M_G)$ , where  $M_G$  is the adjacency matrix of a graph  $G$  and  $\pi$  is a permutation of the rows and columns of  $M_G$ ; this function hides symmetries of  $G$ .

# More Quantum Algorithms

Pell's equation: "What are the integer solutions  $(x,y)$  to the equation:  $x^2 - dy^2 = 1$ , with  $d > 0$  a non-square?" (also "Principal Ideal Problem" in Number theory). This breaks the Buchmann-Williams crypto system.

Simulating quantum mechanical systems.  
(Could be relevant for many sciences.)

Several other problems in number theory:

- Counting solution  $F(x,y)=0$  over finite fields  $x,y \in \mathbf{F}_q$
- Estimating Gauß sums, Tutte polynomials,...

# Where to Look for New Problems?

The consensus is that we have to look for problems that are not in **BPP**, but that are not **NP**-complete either.

Two very interesting candidates:

- Graph automorphism or isomorphism problem
- Shortest vector problem

Both have not been solved yet.

# Shortest Vector

Consider a  $d$ -dimensional basis  $B=[B_1, \dots, B_d]$ , with  $B_j \in \mathbf{Z}^d$  for all  $1 \leq j \leq d$ . The *lattice*  $L(B)$  is defined by  $\{ \beta_1 B_1 + \dots + \beta_d B_d : \beta_j \in \mathbf{Z} \}$  (this is not a vector space).

The shortest/closest vector problem asks for a vector  $W \in \mathbf{Z}^d$  if there is a point in  $L(B)$  that is 'close' to  $W$ .

Variants of this problem are between **P** and **NP-complete** and are used in cryptographic protocols.

The logo for Ntru, featuring the word "Ntru" in a stylized font. The "N" is large and green, while "tru" is smaller and blue. The letters are bold and sans-serif.

# Adiabatic Computation

Adiabatic quantum computation is a heuristic quantum approach to combinatorial optimizations problems.

Just like the classical simulated annealing approach, it tries to find an optimum by 'quantum walking' through the large space of possible answers in a smart way.

The problem of getting stuck in a local minimum occurs also for adiabatic algorithms, but the quantum algorithm is (sometimes) better at getting out of it.

Ultimate complexity is unknown and hard to determine.

# A Few Words About Actually Building Quantum Computers

# Experimental Work

Implementations of quantum communication protocols using photon polarization.

Implementations of small quantum algorithms (“proof of principle”) using NMR, trapped ions,...

Towards a scalable quantum computer using ion traps, solid state NMR, superconducting qubits,...

Will it work for imperfect devices?

# Problems with Quantum Error Correcting Codes

We cannot copy a state to protect against errors.

We cannot just measure the state to inspect the error that might have occurred.

Many ways one can disturb one qubit  $\alpha|0\rangle + \beta|1\rangle$ .

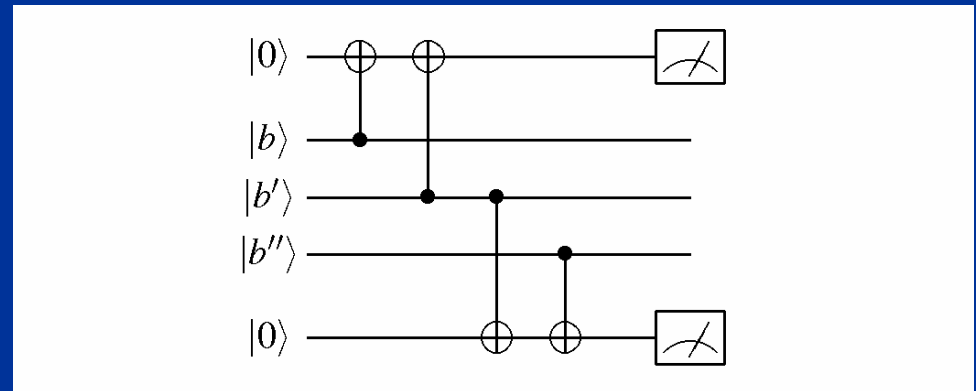
What error model do we assume?

How to use the encoded states in a computation?

# Quantum Error Correction

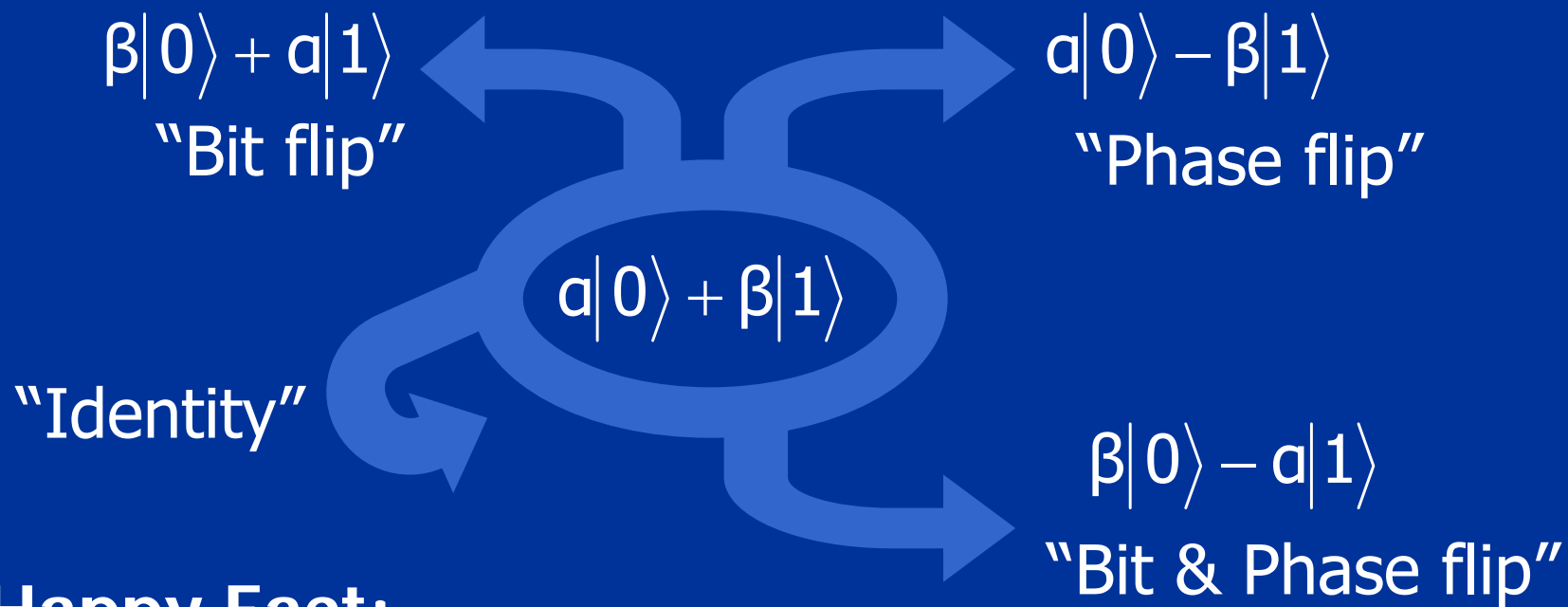
Consider quantum bit wires that sometimes flip bits.  
This changes qubit values:  $\alpha|0\rangle + \beta|1\rangle \mapsto \beta|0\rangle + \alpha|1\rangle$ .  
How to protect oneself against this?

If we use the encoding  
 $\alpha|0_L\rangle + \beta|1_L\rangle = \alpha|000\rangle + \beta|111\rangle$   
then we are protected  
against single bit flips.



What about general qubit errors?

# The 4 Qubit Errors:



## Happy Fact:

If we can correct for these 4 errors,  
then we can correct any qubit error.

(Classical error correction deals with bit-flips only.)

# A [9,1] Quantum Error Code

Logical qubit:  $\alpha|0_L\rangle + \beta|1_L\rangle$

Bit-flip code:  
 $\alpha|000\rangle + \beta|111\rangle$

Phase-flip code:  
 $\alpha \cdot \frac{1}{\sqrt{8}} (|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)$   
 $+ \beta \cdot \frac{1}{\sqrt{8}} (|0\rangle - |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)$

Combined, this gives  
the 9 qubit 'Shor code':

$$\alpha \cdot \frac{1}{\sqrt{8}} (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle)$$
$$+ \beta \cdot \frac{1}{\sqrt{8}} (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle)$$

This code protects against an arbitrary bit error.

# Quantum Error Correction

- Using the results of classical error correction, many quantum codes can be devised.
- Fault-tolerant *computation* is also possible.
- Combined, they give rise to several “thresholds for reliable quantum computation” results.

**An error rate of  $\sim 0.001$  is sufficient to have a working quantum computer.**

# Will It Work?

Nobody knows for sure.

Regardless, quantum computing forces us to rethink our assumptions about computation and information.

For physicists it has come as a big surprise that quantum mechanics can be so useful, and that you can think about quantum mechanics in a CS kind-of-way.

For theoretical computer scientists it has come as a big surprise that they should care more about physics if they want to study a proper computational model.