Lecture 18 – Classical simulations of quantum computation

Outline

- general simulation
- Schrödinger vs Feynman approach
- Clifford circuits
- Stabilizer formalism
- Gottesman-Knill theorem

General simulation

The problem of simulating general quantum circuits is as follows

In: description of quantum circuit C

Out: x with prob $Pr(x) = |\langle x|C|0^n\rangle|^2$

As mentioned, if there exist poly-time classical alg. for this problem then PH collapses at 3rd level!

There's also the relaxed version of the problem where we merely want to sample x with prob

$$\Pr(x) = |\langle x|C|0^n\rangle|^2 \pm \varepsilon, \text{ for some } \varepsilon \geq 0$$

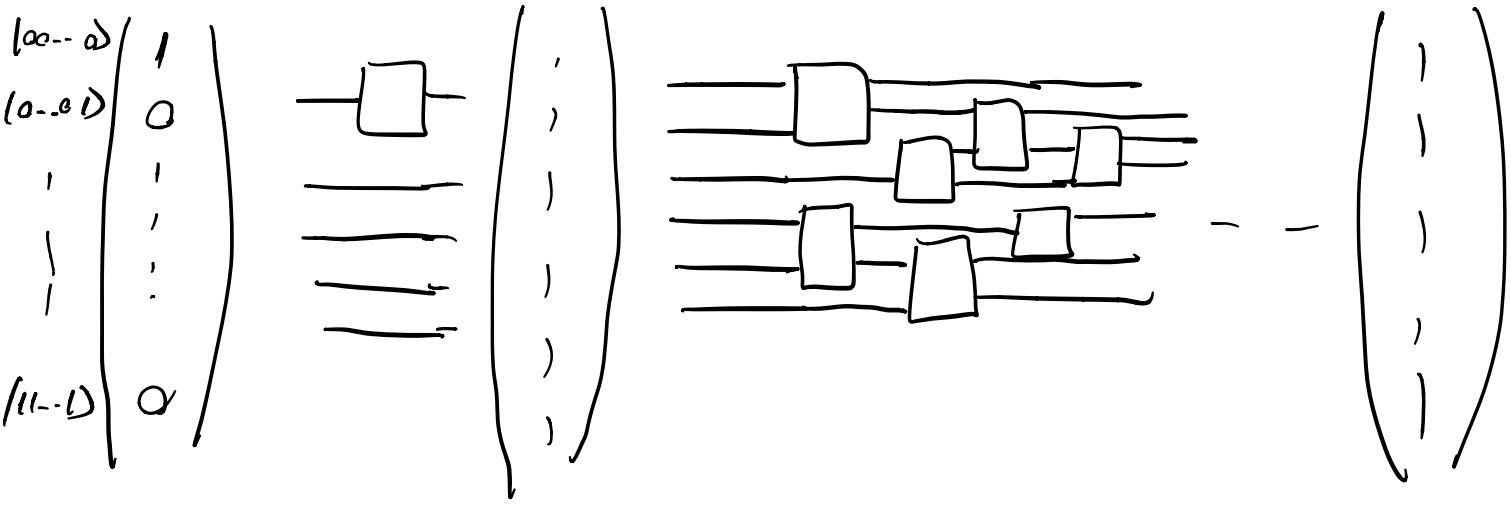
We believe this problem is also classically intractable though the evidence is not as strong.

If poly-time alg are ruled out what can we do instead?

From now on n denotes the number of qubits on which C acts, g denotes the number of gates in C and d denotes the depth of C.

Naive approach

Store in memory the state vector for the circuit input, say $|0^n\rangle$, as a list of the 2^n amplitudes. Update the vector by applying each gate in the circuit



What is the runtime of this algorithm?

$$O(g \cdot 2^n)$$

How much space does it use?

$$O(2^n)$$

↳ storing the state vector

This is the "Schrödinger" approach to simulation since we're taking over input state vector and evolving it by applying the circuit C.

What if we only want to compute the probability of a particular output?

Say estimate just $|\langle x|C|0^n\rangle|^2$.

Can we do better than the Schrödinger approach?

Yes... from a certain point of view.

$C = U_g \cdot U_{g-1} \cdots U_1$ where U_i is the i'th gate in the circuit.

So $\langle x|C|0^n\rangle = \langle x|U_g U_{g-1} \cdots U_1|0^n\rangle$

Terms of the form $\langle x|U_i|y\rangle$ can be estimated efficiently. Why? $|x\rangle$ and $|y\rangle$ are basis states and U_i is a 1 or 2-qubit gate. If x and y differ in any bits except for the ones where U_i acts then $\langle x|U_i|y\rangle = 0$. Otherwise we simply need to compute the action on those bits.

E.g. $x = 010$, $y = 111$

$$\langle x|\, I\otimes H\otimes I\,|y\rangle = \langle 010|\, I\otimes H\otimes I\,|111\rangle =$$

$$= \alpha$$

$x = 101, \quad y = 111$

$$\langle x|\, I\otimes H\otimes I\,|y\rangle = \langle 101|\, I\otimes H\otimes I\,|111\rangle =$$

$$= \langle 0|H|1\rangle = \frac{1}{\sqrt{2}}$$

Recall that $I = \sum_{i\in\{0,1\}^n} |i\rangle\langle i|$

↑ n qubit identity operation

$$\langle x|C|0^n\rangle = \langle x|\, U_g U_{g-1} \cdots U_1 |0^n\rangle =$$

$$\sum_{i_1, i_2 \cdots i_g \in \{0,1\}^2} \langle x|\, U_g |i_g\rangle\langle i_g|\, U_{g-1} |i_{g-1}\rangle\langle i_{g-1}| \cdots \langle i_1|\, U_1 |0^n\rangle$$

↓ each such term can be computed in $O(1)$ time

We have a product of g terms $\Rightarrow$ $O(g)$ time

The sum is then over 4^g elements $\Rightarrow$

$$O(g \cdot 4^g) \text{ time}$$

It doesn't seem like we gained much in time complexity (since $g = \text{poly}(n)$). What about space?

We can just keep a current sum variable and add each term in the sum to it. $\Rightarrow$ space usage is

$$\text{poly}(n, g)$$

This is the Feynman sum over paths method!

Can we combine these two approaches to get algorithm that runs in $O(2^n)$ time and uses $\text{poly}(n, g)$ space? Yes... sort of! Can do $n \cdot d^{o(n)}$ time and $\text{poly}(n, g)$ space

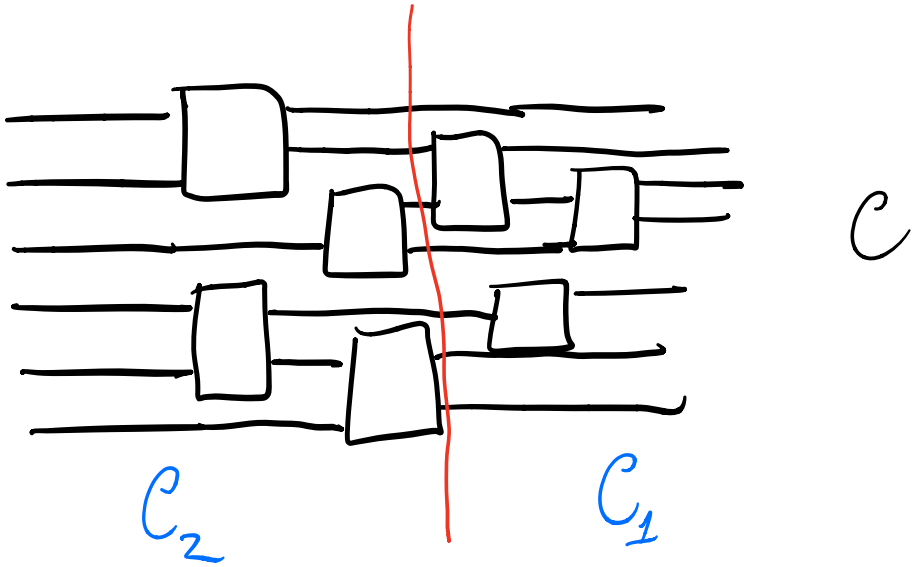
The Schrödinger-Feynman algorithm

(Aaronson-Chen '16)

$$\langle x|C|0^n\rangle = \sum_{y\in\{0,1\}^n} \langle x|C_1|y\rangle\langle y|C_2|0^n\rangle$$

left slice of C

right slice of C



Can compute $\langle x|C_1|y\rangle$, $\langle y|C_2|0^n\rangle$ by doing the same thing recursively.

Let $T(d)$ be runtime of this recursive approach for circuit of depth d. $T(1) = O(n)$

$$T(d) = 2^{n+1}\, T(d/2) = O(n\cdot 2^{(n+1)\log d}) =$$

$$= O(n\cdot d^{n+1})$$

In each level of the recursion need $O(n)$ space to store indices of x and y. There are $\log d$ levels of recursion $\Rightarrow$ $O(n \log d)$ space

There are other algorithms as well (regular approaches use tensor networks), however for the general case they all require exponential time.

What about less general cases?

Simulating Clifford circuits

We've seen that interference is the key to speedups in quantum algorithms. However, it can be misleading to assume that quantum algorithms perform general interference as we've seen in relation to #PP. General interference seems too powerful. However it can also be misleading to think that any interference setup can't be efficiently

classically simulated.

Let $\mathcal{P}_n = \{ \alpha \; P_1 \otimes P_2 \otimes \cdots \otimes P_n \mid \alpha \in \{-1, -i, 1, i\}, P_j \in \{I, X, Y, Z\} \}$

denote the n-qubit Pauli group.

Define the n-qubit Clifford group as

$$\mathcal{C}_n = \{ C \mid \forall P \in \mathcal{P}_n, \; CPC^\dagger \in \mathcal{P}_n \}$$

In other words the Clifford group is the normalizer of the Pauli group. It consists of those operations that, when conjugating a Pauli, result in another Pauli.

Use shorthand $C[P]$ for $CPC^\dagger$

E.g:

- all Paulis are Clifford operations
- CNOT is Clifford

$CNOT[II] = II$ $\quad CNOT[IX] = IX$

$CNOT[XI] = XX$ $\quad CNOT[IZ] = ZZ$

$CNOT[ZI] = ZI$ $\quad$ etc

- H is Clifford

$H[X] = Z,\ H[Z] = X,\ H[I] = I,\ H[Y] = Y$

- $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ is Clifford

Fact: Any $C \in \mathcal{C}_n$ can be generated from a poly(n) product of $\{H, S, CNOT\}$

Fact: $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$ is not Clifford

$$TXT^\dagger = \begin{pmatrix} 0 & e^{-i\pi/4} \\ e^{i\pi/4} & 0 \end{pmatrix}$$ - not a Pauli

As we'll see, Clifford circuits can be efficiently simulated classically.

To understand how we need to introduce one more concept:

Stabilizer states

We say that $|\psi\rangle$ is a stabilizer state if $\exists\, C \in \mathcal{C}_n$ such that

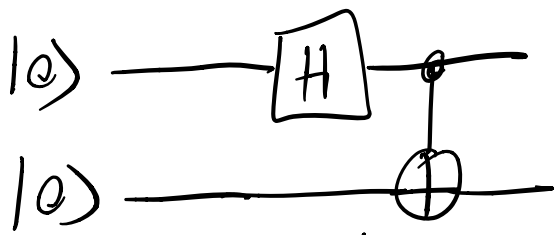
$$|\psi\rangle = C \cdot |0^n\rangle$$

E.g.

– any computational basis state

$$|x\rangle = X^{x^1} \otimes X^{x^2} \otimes \dots \otimes X^{x^n} |0^n\rangle$$

– Bell states



$$|\phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

↑ Clifford circuit

– GHZ states and poor man's cat state

An equivalent definition of stabilizer states

$|\psi\rangle$ is a stabilizer state if there exists a subgroup $G \leq \mathcal{P}_n$ of the Paulis such that

- $\forall\, P \in G, \quad P|\psi\rangle = |\psi\rangle$
- $\forall\, |\phi\rangle \neq |\psi\rangle, \ \exists\, P \in G$ s.t. $P|\phi\rangle \neq |\phi\rangle$

stabilizer of $|\psi\rangle$

G is reffered to as the stabilizer group of $|\psi\rangle$

E.g.

$|0\rangle, \quad G_{|0\rangle} = \{ Z, I \}$ $\qquad Z|0\rangle = |0\rangle$, $I|0\rangle = |0\rangle$

$|01\rangle, \quad G_{|01\rangle} = \{ \overset{ZI}{Z_1}, -Z_1, Z_2, -Z_2, I \}$

$|\phi_+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \quad G_{|\phi_+\rangle} = \{ X_1 X_2, Z_1 Z_2, \ldots \}$

Can represent a group by just its set of generators. Generators are a minimal set of group elements from which all group elements can be derived by taking products.

Fact: Let G be a finite group with minimal set of generators $\{g_1 \dots g_m\}$. Then $m = O(\log |G|)$

We now have all the elements for classically simulating Clifford circuits in polynomial time.

Gottesman–Knill theorem

The theorem simply states that any Clifford circuit acting on a stabilizer state can be simulated in polynomial time classically.

How?

Start with an n-qubit stabilizer state $|\psi\rangle$ for which we know the stabilizer group. By "know" we mean that we have a list of generators

$$\{g_1 \cdots g_m\}$$

Since $|\psi\rangle$ is on n-qubits $|G| = 2^{O(n)}$

But $m = O(\log|G|) = O(n)$

Each generator requires $O(n)$ bits to represent $\Rightarrow$ $O(n^2)$ bits total to represent $|\psi\rangle$.

Now suppose we act on $|\psi\rangle$ with a Clifford circuit C. How do we efficiently compute a generating set for $C|\psi\rangle$.

Note the following

$$\text{if } P \in G \Rightarrow P|\psi\rangle = |\psi\rangle \quad , \quad Q \in \mathcal{P}_n$$

$$C|\psi\rangle = C \cdot P|\psi\rangle = \underbrace{CPC^\dagger}_{I} C|\psi\rangle$$

$\Rightarrow$ $Q \cdot (C|\psi\rangle) = C|\psi\rangle$

where $Q = CPC^\dagger$

$\Rightarrow$ We can get a generating set for $C|\psi\rangle$ by conjugating the generating set of $|\psi\rangle$!

$\Rightarrow$ compute $\{ Cg_1C^\dagger, Cg_2C^\dagger, \cdots, Cg_mC^\dagger \}$

How do we compute each conjugation efficiently?

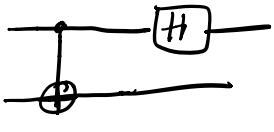
We can assume C consists of just $\{H, S, CNOT\}$.

We know how each one of these acts on all the Paulis.

$\Rightarrow$ Just go through C gate by gate and conjugate g_j.

E.g. $g = XI$

$C = (H \cdot I) \cdot CNOT_{12}$



$\Rightarrow (H \cdot I)\, CNOT_{12} \cdot (XI)\, CNOT_{12} \cdot (H \cdot I)$

$CNOT\,[XI] = XX$

$\Rightarrow (H \cdot I) \cdot (XX)\,(H \cdot I)$

$H\,[X] = Z$

$\Rightarrow ZX = C g C^{\dagger}$

This is our updated stabilizer element.

Time required is $O(|C| \cdot n)$

($|C|$: size of circuit)

At this point we have the generating set for $C|\psi\rangle$.

What about measurement?

We'll suppose measurements are in the computational basis.

Fact: If $|\phi\rangle$ is a stabilizer state, the probabilities for

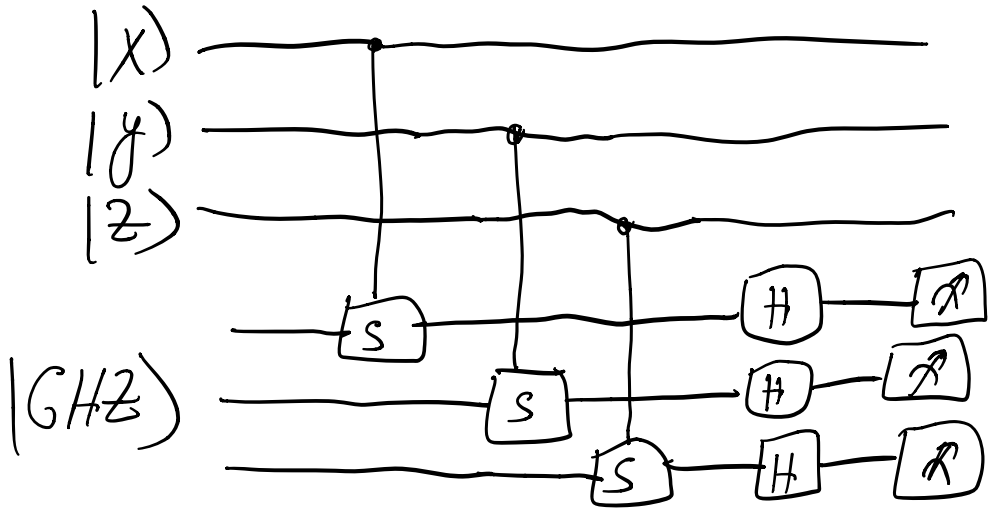
outcomes 0 or 1 when measuring qubit can be only 0, 1 or 1/2.

When measuring qubit i, check if any of the generators has an X or a Y on qubit i.
If so outcome will have prob 1/2.
Otherwise can be shown that either Z_i or $-Z_i$ stabilizes the state. Can determine which (from generators) using Gaussian elimination. If it's Z_i outcome is 0, otherwise it's 1
(this can be extended to multi-qubit measurements)

$\Rightarrow$ can classically simulate Clifford circuits in poly time.

This is surprising because Clifford circuits perform interference!

Recall the parity halving problem. The quantum solution for it was a Clifford circuit



⇒ PHP can be solved efficiently classically (just not in constant depth).

Observations

- the techniques used to show q. advantage for shallow circuits cannot be applied as is to general circuits, since Cliffords are efficiently simulatable
- Cliffords are not universal, not even classically universal!
- Interference as used in Shor, Simon etc somehow leverages non-Clifford gates

– there's more to the representation of q. states and operations than just vectors and matrices. The stabilizer formalism highlights the difficulty in ruling out efficient classical descriptions of complex states and operations.

Interference is subtle but not malicious! :)