Lecture 21 – Review of Quantum Algs for Classical Probs

[Summary of the quantum algorithms for classical problems we've seen.]

Grover's alg. for SAT:

Given [classical AND/OR/NOT circuit] $C: \{0,1\}^n \to \{0,1\}$:

outputs quantum code Q with $\approx 2^{n/2} \cdot \text{size}(C)$ instructions such that:

- C satisfiable $\Rightarrow$ with high probability Q prints "x" s.t. $C(x)=1$.
- C unsatisfiable $\Rightarrow$ with high probability Q prints "unsatisfiable"

[We didn't quite prove all details of this, but you could piece it together from the #SAT lecture.]

[Let's ask ourselves a series of questions ...]

- Interesting task? ✓✓✓ [SAT is fundamental, NP-complete]
- Could a classical computer match this?

- Could a classical computer match this?

  Probably not!

  "P≠NP" $\Leftarrow$ $\nexists$ poly-time classical alg. for SAT.

  "SETH" $\Leftarrow$ $\nexists$ classical SAT alg. in $\ll 2^n$ time.

  [Strong Exponential Time Hypothesis" — fairly strongly believed"]

  ↘ Quantum alg. beats this by $2^{n/2}$ factor!

  [exponential!]

  But still exponential time. ∴

- Could there be a faster quantum alg.?

  [Well, maybe. Never say never. But no new idea in ≈ 25 years.]

  "QSETH": $\nexists$ quantum SAT alg in $\ll 2^{n/2}$ time.

  [proposed in last few years]

  ↳ quantum computers can't efficiently solve NP-complete probs.

[Remark: imagine the "black-box" model where you can't "look into code of C", can only run it on inputs. Then it's easy to prove you need $\gtrsim 2^n$ time to solve SAT classically. You can also invent a "quantum black box" model where you can only apply "Add C(X) to Amps". In this model you can prove $\gtrsim 2^{n/2}$ time needed to solve SAT. Indeed, was done in 1994, before Grover's Alg!]

- Say a Q.C. company claimed to build a Q.C. that could run Grover with $n = 80$. [$2^{80} \approx$ unphysical; $2^{40} \approx$ physical]

Could we trust/test it? [Paradox? How to test a quantum computer with only "classical" powers?]

One-sided trust:

- if Q spits out x, can easily check if $C(x) = 1$
- " " " " "unsatisfiable", no reason to trust....

The nature of NP problems like SAT.

[can verify positive solutions, not necessarily for neg. sol'ns]

Rollercoaster ("Bernstein-Vazirani") Algorithm:

Given $C: \{0,1\}^n \to \{0,1\}$, promised to compute XOR_b for some mystery $b \in \{0,1\}^n$,

efficiently computes quantum code Q, $\approx$ size(C) instructions,

s.t. $\Pr[Q \text{ prints } b] = 100\%$.

- Interesting problem? × [Hard to imagine a scenario of use, or where you could be assured promise holds.]

- [Could there be a faster quantum alg.? Well, no, you're running in linear time, what more could you want?]

- Could a classical computer match this?
  → Seems to need $\approx n \cdot \text{size}(C)$ instructions.
  ‖ Can prove this is necessary in "black box" model. ‖
  ↳ quantum speedup is "just" factor-n. ∴
  ‖ Could still be great if n is like a million, but... kind of a moot point because the Rollercoaster Problem doesn't seem to ever naturally arise. ‖

- Could we verify <u>output</u> of untrusted Q.C. in time faster than just classically solving?
  Yes, allowing randomized tests! Nice exercise....

  <u>But</u>: A new difficulty: can we verify the <u>input</u> satisfies the promise (of computing XOR_b for some b)? Not really... So this Rollercoaster problem ain't so great.

‖ <u>Simon's Algorithm</u>: We didn't cover it in this class, but the brief story is: It has similar properties to "Rollercoaster", except the quantum speedup is <u>exponential</u>. Cool, except like Rollercoaster, not a natural problem, and <u>input</u> again involves an unverifiable promise ∴. Simon's alg. inspired Shor, though ... ‖

Quantum Factoring Alg. [[Shor / Kitaev]]

Given F, of n digits,

a classical poly(n)-time random alg (random assuming "M" picked randomly) outputs quantum code Q s.t. with high probability Q prints factorization of F.

[[Well, in our proof, Q prints "K/L" to $O(n)$ digits of accuracy, and from (a few runs of) this we can classically compute factorization. But can build this classical post-processing into Q if you like...]]

- Interesting problem? ✓✓✓ [[All '70s-'00s cryptography assumes it's classically hard.]]
- Could a classical alg. match this?

  Very likely not... [[Not insanely certain, but of course no one has done it after centuries of trying....]]

  Fastest known: $\exp(n^{1/3})$ time.
- Could we verify solutions produced by "untrusted" quantum computer?

  Yes! ✓✓✓ [[This is an awesome property of factoring. Not even "one-sided trust" — every input has easy-to-verify output.]]

Complexity Theory digression:

Q: Is "Factoring" in NP?

A: Doesn't type-check. NP is about "decision problems" (yes/no answer)

[[Hmm. That's annoying. Well, you can "artificially" make Factoring into a decision problem of ≈ equal difficulty.]]

"Decision-Factoring": Input F, H (n bits).

Answer yes or no: F has a prime factor between 2 & H.

Efficient Factoring (poly(n)-time) ⇒ Efficient Decision-Factoring? ✓

[[obvious]]

Efficient Decision-Factoring ⇒ Efficient Factoring? ✓

[[exercise: binary search for prime factors by varying H.]]

Dec-Factoring ∈ NP? ⇔ efficiently verifiable "witnesses" when answer is "yes"?

✓ : Witness is prime factor between 2 & H

[[recall: testing primality in poly(n) time.]]

But unlike "SAT", Dec-Factoring also has efficiently verifiable witnesses when answer is "No"!!!

Witness for no prime factor between 2 & H?
→ complete prime factorization of F!

Fact: "NP-complete" problems (like SAT) do not have such "no-witnesses" (assuming "NP ≠ coNP", a strongly believed "P≠NP" variant)

∴ (Dec-) Factoring is (almost surely) NOT "NP-complete".

[[Although we don't know how to solve it in poly(n) time, it is somehow fundamentally easier than the NP-complete problems... Almost no other problem has this property...]]

[This is the craziest, most fascinating cosmic joke. 30 years ago we realize the universe gives us some computing power possibly beyond classical "P". You'd think, "OK, either it's no real extra power :(or it solves NP-complete probems :)." Nope! By some insanity a sci-fi writer wouldn't dare to suggest, it solves SAT faster by $2^{n/2}$ factor — but still exponentially; and, it solves in poly-time that one super-rare problem that is not known to be in P but is easier than NP-complete!!]

[aka "Deutsch-Jozsa"]

Bias-Busting: Given $C: \{0,1\}^n \to \{0,1\}$, efficiently outputs quantum code Q ($\times$ size(C) ops)

s.t.:

- C "biased" ($\Pr_x[C(x)=1] \neq 1/2$) $\Rightarrow \Pr[Q \text{ prints } \underbrace{000\ldots0}_{\text{meaning "yes"}}] > 0$
- C "unbiased" ($\Pr_x[C(x)=1] = 1/2$) $\Rightarrow \Pr[Q \text{ prints "yes"}] = 0$.

- Interesting Task? ✓✓✓ [You might not realize how cool this Bias-Busting task is, but....]

SAT $\leq_p$ Bias-Busting (i.e., Bias-Busting is NP-hard)

[$\leq_p$: efficiently reduces to]

Given C, want to know if satisfiable $\Leftrightarrow \Pr_x[C(x)=1] > 0$.

Form $C'(x_1,\ldots,x_n,x_{n+1}) =$ "x_{n+1} OR $C(x_1,\ldots,x_n)$".

Then $\Pr_{x_1,\ldots,x_{n+1}}[C'(x_1,\ldots,x_n)=1] = \frac{1}{2} + \frac{1}{2}\cdot\Pr_x[C(x)=1]$.

($\Pr[x_{n+1}=1]$, $\Pr[x_{n+1}=0]$)

$\therefore$ · C satisfiable $\Rightarrow$ C' (positively) biased.

· C unsatisfiable $\Rightarrow$ C' unbiased.

[So wait. The ability to solve Bias-Busting efficiently lets you solve SAT efficiently. Why don't quantum computers solve SAT efficiently then?]

[The catch is that the quantum Bias-Busting alg. is not really satisfactorily "solving" the problem. ∴]

Quantum alg. has one-sided error "yes means yes" style. But probability of Q "certifying" that C is biased could be as small as $\approx 4^{-n}$. ∴

[If we had C biased $\Rightarrow$ Pr["yes"] $\geq$ 90% or $\geq$ 10% or even $\geq \frac{1}{\text{poly}}$ we'd be happy, and we could solve SAT efficiently (w/ high prob.), but "$\geq 4^{-n}$" is practically useless.]

- Could quantum alg. solve Bias-Busting better?
  Unlikely, as we don't believe Q.C.'s can solve SAT efficiently (QSETH).

[- Suppose a Q.C. company claimed to implement Bias-Busting alg. Also suppose you ran it on C and, surprisingly, it output "yes" (= 000···0 = "biased"). Could you definitely trust this? Doesn't feel like a real "certificate" of "yes"...]

Q: [We saw Bias-Busting is NP-hard, but....]
Is Bias-Busting $\in$ NP? (hence NP-complete)
Efficiently checkable (classical) witnesses that C is biased?

A: Doesn't seem likely. [No one has any idea, and indeed we have evidence it's not true.]

But suppose, hypothetically, Bias-Busting $\in$ NP.

So $\exists$ poly(n)-time (classical) verifier V s.t.:

- C biased $\Rightarrow$ $\exists$ witness w s.t. $V(C,w) = 1$
- C unbiased $\Rightarrow$ $\forall$ w, $V(C,w) = 0$.

Now define this poly(n)-time classical probabilistic alg:

R: · pick w at random
· output $V(C,w)$.

Then C biased $\Rightarrow$ $\Pr[R \text{ outputs } 1] > 0$ [because some w exists]

C unbiased $\Rightarrow$ $\Pr[R \text{ outputs } 1] = 0$.

Just like quantum alg Q!

[Well, but as I said, it doesn't seem likely that Bias-Busting $\in$ NP. In fact, in 1990, prior to quantum computing, some computational complexity theorists had studied this exact question....!]

Toda-Ogiwara Theorem: Bias-Busting $\notin$ NP, assuming "$P^{\Sigma_2} \neq NP^{\Sigma_2}$" [some strongly-believed variant of "$P \neq NP$"]

[also was comparing NES game soundtracks around this time]

Conclusion: Assuming "$P^{\Sigma_2} \neq NP^{\Sigma_2}$", can't convert efficiently convert quantum code to classical probabilistic code with same* output probabilities

(Else: could convert Bias-Busting alg Q to classical R, violating Toda-Ogiwara Thm.)

* In fact: can't even get output probs correct up to a factor of 999, because ">0" ÷ 999 is still ">0" and 0.999 is still 0.

[Next lecture: can we classically approximate output probabilities of quantum code at all??]

That's it. That's [honestly, more or less] all the quantum algs for classical problems known. Nothing really new since 1996. ∴

[No obvious reason to expect more, either. Well, don't be sad. The interesting thing to study is quantum algs. for quantum problems.....]