Is BQP Squeezed Out or In?

UB CSE Theory Group

Kenneth W. Regan[1]
University at Buffalo (SUNY)

6 May, 2024

[1]Includes joint work with Amlan Chakrabarti, U. Calcutta, and Chaowen Guan, U. Cincinnati

Dichotomy

The phenomenon of *natural* computational problems and mathematical entities clumping into an "easy" level A and a "hard" level B with little or nothing *salient* in between.

Dichotomy

The phenomenon of *natural* computational problems and mathematical entities clumping into an "easy" level A and a "hard" level B with little or nothing *salient* in between. *Examples:*

Dichotomy

The phenomenon of *natural* computational problems and mathematical entities clumping into an “easy” level A and a “hard” level B with little or nothing *salient* in between. *Examples:*

1. “Almost all” problems in NP are either in P or NP-complete.

Dichotomy

The phenomenon of *natural* computational problems and mathematical entities clumping into an "easy" level A and a "hard" level B with little or nothing *salient* in between. *Examples:*

1. "Almost all" problems in NP are either in P or NP-complete.
2. For "most" problems the best known algorithm either runs in polynomial time or in exponential time (meaning time $2^{\Omega(n)}$ or time $2^{n^{\Omega(1)}}$ depending on the problem and encoding).

Dichotomy

The phenomenon of *natural* computational problems and mathematical entities clumping into an "easy" level A and a "hard" level B with little or nothing *salient* in between. *Examples:*

1. "Almost all" problems in NP are either in P or NP-complete.
2. For "most" problems the best known algorithm either runs in polynomial time or in exponential time (meaning time $2^{\Omega(n)}$ or time $2^{n^{\Omega(1)}}$ depending on the problem and encoding).
3. Item 2 semi-follows from 1 insofar as all NP-complete languages are poly-time equivalent and exponential time is best known for SAT.

Dichotomy

The phenomenon of *natural* computational problems and mathematical entities clumping into an "easy" level A and a "hard" level B with little or nothing *salient* in between. *Examples:*

1. "Almost all" problems in NP are either in P or NP-complete.
2. For "most" problems the best known algorithm either runs in polynomial time or in exponential time (meaning time $2^{\Omega(n)}$ or time $2^{n^{\Omega(1)}}$ depending on the problem and encoding).
3. Item 2 semi-follows from 1 insofar as all NP-complete languages are poly-time equivalent and exponential time is best known for SAT.
4. Note: If $\mathsf{NP} \neq \mathsf{P}$ then there are languages in $\mathsf{NP} \setminus (\mathsf{NPC} \cup \mathsf{P})$.

Dichotomy

The phenomenon of *natural* computational problems and mathematical entities clumping into an "easy" level A and a "hard" level B with little or nothing *salient* in between. *Examples:*

1. "Almost all" problems in NP are either in P or NP-complete.
2. For "most" problems the best known algorithm either runs in polynomial time or in exponential time (meaning time $2^{\Omega(n)}$ or time $2^{n^{\Omega(1)}}$ depending on the problem and encoding).
3. Item 2 semi-follows from 1 insofar as all NP-complete languages are poly-time equivalent and exponential time is best known for SAT.
4. Note: If $\mathsf{NP} \neq \mathsf{P}$ then there are languages in $\mathsf{NP} \setminus (\mathsf{NPC} \cup \mathsf{P})$. But they are expressly diagonal and thus "artificial."

Dichotomy

The phenomenon of *natural* computational problems and mathematical entities clumping into an "easy" level A and a "hard" level B with little or nothing *salient* in between. *Examples:*

1. "Almost all" problems in NP are either in P or NP-complete.
2. For "most" problems the best known algorithm either runs in polynomial time or in exponential time (meaning time $2^{\Omega(n)}$ or time $2^{n^{\Omega(1)}}$ depending on the problem and encoding).
3. Item 2 semi-follows from 1 insofar as all NP-complete languages are poly-time equivalent and exponential time is best known for SAT.
4. Note: If $\mathsf{NP} \neq \mathsf{P}$ then there are languages in $\mathsf{NP} \setminus (\mathsf{NPC} \cup \mathsf{P})$. But they are expressly diagonal and thus "artificial."
5. There is a tight deterministic time hierarchy...

Dichotomy

The phenomenon of *natural* computational problems and mathematical entities clumping into an "easy" level A and a "hard" level B with little or nothing *salient* in between. *Examples:*

1. "Almost all" problems in NP are either in P or NP-complete.
2. For "most" problems the best known algorithm either runs in polynomial time or in exponential time (meaning time $2^{\Omega(n)}$ or time $2^{n^{\Omega(1)}}$ depending on the problem and encoding).
3. Item 2 semi-follows from 1 insofar as all NP-complete languages are poly-time equivalent and exponential time is best known for SAT.
4. Note: If $\mathsf{NP} \neq \mathsf{P}$ then there are languages in $\mathsf{NP} \setminus (\mathsf{NPC} \cup \mathsf{P})$. But they are expressly diagonal and thus "artificial."
5. There is a tight deterministic time hierarchy...but the languages involved are diagonal or are "artificially complete."

Cases Where Dichotomy Holds Completely

Cases Where Dichotomy Holds Completely

- Schaefer's Dichotomy Theorem for SAT.

Cases Where Dichotomy Holds Completely

- Schaefer's Dichotomy Theorem for SAT.
- Nonuniform CSP Dichotomy Theorem (see also this and these slides).

Cases Where Dichotomy Holds Completely

- Schaefer's Dichotomy Theorem for SAT.
- Nonuniform CSP Dichotomy Theorem (see also this and these slides).
- *Growth Rate in Groups:* Given an infinite group G with *finite* generating set S, put $f(n)$ = the number of elements in G expressible as length-n words of g or g^{-1} over $g \in S$.

Cases Where Dichotomy Holds Completely

- Schaefer's Dichotomy Theorem for SAT.
- Nonuniform CSP Dichotomy Theorem (see also this and these slides).
- *Growth Rate in Groups:* Given an infinite group G with *finite* generating set S, put $f(n) =$ the number of elements in G expressible as length-n words of g or g^{-1} over $g \in S$. Gromov's Theorem: $f(n) = n^{O(1)}$ iff G has a nilpotent subgroup of finite index.

Cases Where Dichotomy Holds Completely

- Schaefer's Dichotomy Theorem for SAT.
- Nonuniform CSP Dichotomy Theorem (see also this and these slides).
- *Growth Rate in Groups:* Given an infinite group G with *finite* generating set S, put $f(n) =$ the number of elements in G expressible as length-n words of g or g^{-1} over $g \in S$. Gromov's Theorem: $f(n) = n^{O(1)}$ iff G has a nilpotent subgroup of finite index.
- *Gap Conjecture:* Either $f(n) = n^{O(1)}$ or $f(n) = 2^{\Omega(\sqrt{n})}$.

Cases Where Dichotomy Holds Completely

- Schaefer's Dichotomy Theorem for SAT.
- Nonuniform CSP Dichotomy Theorem (see also this and these slides).
- *Growth Rate in Groups:* Given an infinite group G with *finite* generating set S, put $f(n)$ = the number of elements in G expressible as length-n words of g or g^{-1} over $g \in S$. Gromov's Theorem: $f(n) = n^{O(1)}$ iff G has a nilpotent subgroup of finite index.
- *Gap Conjecture:* Either $f(n) = n^{O(1)}$ or $f(n) = 2^{\Omega(\sqrt{n})}$.
- *Proved* when G is a finitely-generated subgroup of a connected Lie group. (J. Tits).

NP-Intermediate Status

NP-Intermediate Status

- **Graph Isomorphism** (GI) belongs to a natural cluster of algebraic problems.

NP-Intermediate Status

- **Graph Isomorphism** (GI) belongs to a natural cluster of algebraic problems.
- Laszló Babai recently put GI and hence all these problems into *quasipolynomial* (QP) time, indeed $n^{O(\log n)}$ time.

NP-Intermediate Status

- **Graph Isomorphism** (GI) belongs to a natural cluster of algebraic problems.
- Laszló Babai recently put GI and hence all these problems into *quasipolynomial* (QP) time, indeed $n^{O(\log n)}$ time.
- Raises belief they are in P.

NP-Intermediate Status

- **Graph Isomorphism** (GI) belongs to a natural cluster of algebraic problems.
- Laszló Babai recently put GI and hence all these problems into *quasipolynomial* (QP) time, indeed $n^{O(\log n)}$ time.
- Raises belief they are in P. (Or just redefine QP as "easy.")

NP-Intermediate Status

- **Graph Isomorphism** (GI) belongs to a natural cluster of algebraic problems.
- Laszló Babai recently put GI and hence all these problems into *quasipolynomial* (QP) time, indeed $n^{O(\log n)}$ time.
- Raises belief they are in P. (Or just redefine QP as "easy.")
- **Factoring** and **Discrete Log** are related.

NP-Intermediate Status

- **Graph Isomorphism** (GI) belongs to a natural cluster of algebraic problems.
- Laszló Babai recently put GI and hence all these problems into *quasipolynomial* (QP) time, indeed $n^{O(\log n)}$ time.
- Raises belief they are in P. (Or just redefine QP as "easy.")
- **Factoring** and **Discrete Log** are related. $2^{\tilde{\Omega}(n^{1/3})}$ time lower bound for both?

NP-Intermediate Status

- **Graph Isomorphism** (GI) belongs to a natural cluster of algebraic problems.
- Laszló Babai recently put GI and hence all these problems into *quasipolynomial* (QP) time, indeed $n^{O(\log n)}$ time.
- Raises belief they are in P. (Or just redefine QP as "easy.")
- **Factoring** and **Discrete Log** are related. $2^{\tilde{\Omega}(n^{1/3})}$ time lower bound for both? Neither has much of a cluster.

NP-Intermediate Status

- **Graph Isomorphism** (GI) belongs to a natural cluster of algebraic problems.
- Laszló Babai recently put GI and hence all these problems into *quasipolynomial* (QP) time, indeed $n^{O(\log n)}$ time.
- Raises belief they are in P. (Or just redefine QP as "easy.")
- **Factoring** and **Discrete Log** are related. $2^{\tilde{\Omega}(n^{1/3})}$ time lower bound for both? Neither has much of a cluster.
- The **Minimum Circuit Size Problem** (MCSP) has structural evidence for both "not in P" and "not NP-complete."

NP-Intermediate Status

- **Graph Isomorphism** (GI) belongs to a natural cluster of algebraic problems.
- Laszló Babai recently put GI and hence all these problems into *quasipolynomial* (QP) time, indeed $n^{O(\log n)}$ time.
- Raises belief they are in P. (Or just redefine QP as "easy.")
- **Factoring** and **Discrete Log** are related. $2^{\tilde{\Omega}(n^{1/3})}$ time lower bound for both? Neither has much of a cluster.
- The **Minimum Circuit Size Problem** (MCSP) has structural evidence for both "not in P" and "not NP-complete." Featured prominently in a recent big article in *Quanta*.

NP-Intermediate Status

- **Graph Isomorphism** (GI) belongs to a natural cluster of algebraic problems.
- Laszló Babai recently put GI and hence all these problems into *quasipolynomial* (QP) time, indeed $n^{O(\log n)}$ time.
- Raises belief they are in P. (Or just redefine QP as "easy.")
- **Factoring** and **Discrete Log** are related. $2^{\tilde{\Omega}(n^{1/3})}$ time lower bound for both? Neither has much of a cluster.
- The **Minimum Circuit Size Problem** (MCSP) has structural evidence for both "not in P" and "not NP-complete." Featured prominently in a recent big article in *Quanta*.
- The **Kolmogorov Complexity Bounding Problem** (given a string x and number k, does x have a polynomial-time verifiable seed s of length at most k?) may be related to MCSP—but both are still fairly isolated.

NP-Intermediate Status

- **Graph Isomorphism** (GI) belongs to a natural cluster of algebraic problems.
- Laszló Babai recently put GI and hence all these problems into *quasipolynomial* (QP) time, indeed $n^{O(\log n)}$ time.
- Raises belief they are in P. (Or just redefine QP as "easy.")
- **Factoring** and **Discrete Log** are related. $2^{\tilde{\Omega}(n^{1/3})}$ time lower bound for both? Neither has much of a cluster.
- The **Minimum Circuit Size Problem** (MCSP) has structural evidence for both "not in P" and "not NP-complete." Featured prominently in a recent big article in *Quanta*.
- The **Kolmogorov Complexity Bounding Problem** (given a string x and number k, does x have a polynomial-time verifiable seed s of length at most k?) may be related to MCSP—but both are still fairly isolated.
- More-natural characterizations, or indelibly "Meta"?

Counting Problems

#P is the counting-problem analogue of NP. If $R(x, y)$ is a relation decidable in time $|x|^{O(1)}$, then

Counting Problems

#P is the counting-problem analogue of NP. If $R(x, y)$ is a relation decidable in time $|x|^{O(1)}$, then

- $L_R = (\exists y)R(x, y)$ defines a problem in NP;

Counting Problems

#P is the counting-problem analogue of NP. If $R(x,y)$ is a relation decidable in time $|x|^{O(1)}$, then

- $L_R = (\exists y)R(x,y)$ defines a problem in NP;
- $h_R(x) = |\{\, y : R(x,y) \,\}|$ defines a function in #P;

and all languages/functions in the respective classes arise that way.

Counting Problems

#P is the counting-problem analogue of NP. If $R(x, y)$ is a relation decidable in time $|x|^{O(1)}$, then

- $L_R = (\exists y)R(x, y)$ defines a problem in NP;
- $h_R(x) = |\{ y : R(x, y) \}|$ defines a function in #P;

and all languages/functions in the respective classes arise that way.

- E.g., the function h_{SAT} counting satisfying assignments of a 3CNF formula is complete for #P under polynomial-time mapping reductions f of functions: $g \leq_m^p h$ via f means $g(x) = h(f(x))$.

Counting Problems

#P is the counting-problem analogue of NP. If $R(x, y)$ is a relation decidable in time $|x|^{O(1)}$, then

- $L_R = (\exists y)R(x, y)$ defines a problem in NP;
- $h_R(x) = |\{ y : R(x, y) \}|$ defines a function in #P;

and all languages/functions in the respective classes arise that way.

- E.g., the function h_{SAT} counting satisfying assignments of a 3CNF formula is complete for #P under polynomial-time mapping reductions f of functions: $g \leq_m^p h$ via f means $g(x) = h(f(x))$.
- #P is polynomial-time *Turing*-equivalent to the language class **PP**, which is characterized by languages of the form $L_h = \{ (x, k) : h(x) \geq k \}$ over $h \in$ #P.

Counting Problems

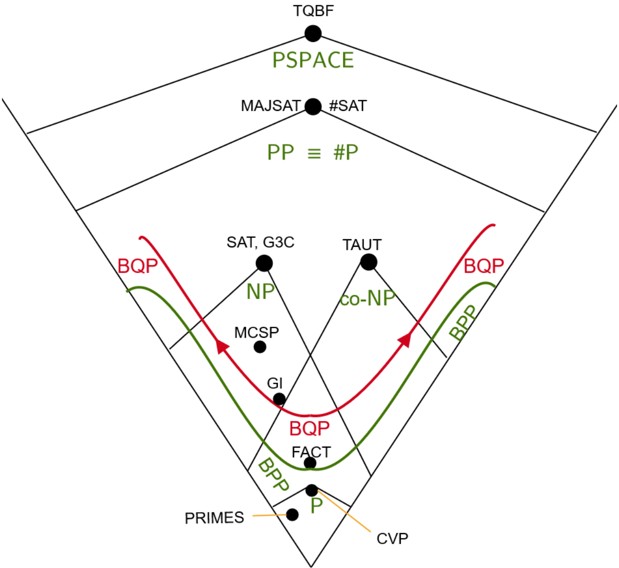
#P is the counting-problem analogue of NP. If $R(x, y)$ is a relation decidable in time $|x|^{O(1)}$, then

- $L_R = (\exists y)R(x, y)$ defines a problem in NP;
- $h_R(x) = |\{ y : R(x, y) \}|$ defines a function in #P;

and all languages/functions in the respective classes arise that way.

- E.g., the function h_{SAT} counting satisfying assignments of a 3CNF formula is complete for #P under polynomial-time mapping reductions f of functions: $g \leq_m^p h$ via f means $g(x) = h(f(x))$.
- #P is polynomial-time *Turing*-equivalent to the language class **PP**, which is characterized by languages of the form $L_h = \{ (x, k) : h(x) \geq k \}$ over $h \in$ #P.
- PP is the lowest known "simple" upper bound for **BQP**, bounded-error quantum polynomial time. (A technical subclass called **AWPP** contains BQP.)

Diagram of These Classes and Problems



Dichotomy Within #P

Dichotomy Within #P

- Counting version of Schaefer's theorem proved by Creignou and Hermann.

Dichotomy Within #P

- Counting version of Schaefer's theorem proved by Creignou and Hermann.
- More cases are #P-complete, including **monotone #2SAT**.

Dichotomy Within #P

- Counting version of Schaefer's theorem proved by Creignou and Hermann.
- More cases are #P-complete, including **monotone #2SAT**.
- Same for $\#CSP$ for domain size 3 (A. Bulatov).

Dichotomy Within #P

- Counting version of Schaefer's theorem proved by Creignou and Hermann.
- More cases are #P-complete, including **monotone #2SAT**.
- Same for $\#CSP$ for domain size 3 (A. Bulatov). **Feder-Vardi conjecture**: ditto for all sizes.

Dichotomy Within #P

- Counting version of Schaefer's theorem proved by Creignou and Hermann.
- More cases are #P-complete, including **monotone #2SAT**.
- Same for $\#CSP$ for domain size 3 (A. Bulatov). **Feder-Vardi conjecture**: ditto for all sizes.
- Jin-Yi Cai and co-workers extended this to other CSP cases and also proved dichotomy for *graph homomorphisms* and *holant* problems.

Dichotomy Within #P

- Counting version of Schaefer's theorem proved by Creignou and Hermann.
- More cases are #P-complete, including **monotone #2SAT**.
- Same for $\#CSP$ for domain size 3 (A. Bulatov). **Feder-Vardi conjecture**: ditto for all sizes.
- Jin-Yi Cai and co-workers extended this to other CSP cases and also proved dichotomy for *graph homomorphisms* and *holant* problems. The former involve computing the **partition function**

$$Z_A(G) = \sum_{h:V \to [m]} \prod_{(u,v) \in E} A[h(u), h(v)]$$

where $G = (V, E)$ on n nodes and A is a symmetric $m \times m$ matrix.

Dichotomy Within #P

- Counting version of Schaefer's theorem proved by Creignou and Hermann.
- More cases are #P-complete, including **monotone #2SAT**.
- Same for $\#CSP$ for domain size 3 (A. Bulatov). **Feder-Vardi conjecture**: ditto for all sizes.
- Jin-Yi Cai and co-workers extended this to other CSP cases and also proved dichotomy for *graph homomorphisms* and *holant* problems. The former involve computing the **partition function**

$$Z_A(G) = \sum_{h:V\to[m]} \prod_{(u,v)\in E} A[h(u),h(v)]$$

  where $G = (V,E)$ on n nodes and A is a symmetric $m \times m$ matrix.
- For many other counting problems, seemingly small changes in settings flip the problem between P and #P-hard, with no sign of anything in between.

A Simple Example Over $\mathbb{Z}_4$

Consider *quadratic* polynomials $f(x_1, x_2, \ldots, x_n)$ modulo 4.

A Simple Example Over $\mathbb{Z}_4$

Consider *quadratic* polynomials $f(x_1, x_2, \ldots, x_n)$ modulo 4.

- Counting the number of zeroes is in P.

A Simple Example Over $\mathbb{Z}_4$

Consider *quadratic* polynomials $f(x_1, x_2, \ldots, x_n)$ modulo 4.

- Counting the number of zeroes is in P. (Follows by [Cai-Chen-Lipton-Luo, 2010].)

A Simple Example Over $\mathbb{Z}_4$

Consider *quadratic* polynomials $f(x_1, x_2, \ldots, x_n)$ modulo 4.

- Counting the number of zeroes is in P. (Follows by [Cai-Chen-Lipton-Luo, 2010].)
- Counting the number of zeroes in $\{0, 1\}^n$ is #P-complete.

A Simple Example Over $\mathbb{Z}_4$

Consider *quadratic* polynomials $f(x_1, x_2, \ldots, x_n)$ modulo 4.

- Counting the number of zeroes is in P. (Follows by [Cai-Chen-Lipton-Luo, 2010].)
- Counting the number of zeroes in $\{0, 1\}^n$ is #P-complete.
- But if all cross-terms are $2x_ix_j$ it is in P again.

A Simple Example Over $\mathbb{Z}_4$

Consider *quadratic* polynomials $f(x_1, x_2, \ldots, x_n)$ modulo 4.

- Counting the number of zeroes is in P. (Follows by [Cai-Chen-Lipton-Luo, 2010].)
- Counting the number of zeroes in $\{0, 1\}^n$ is #P-complete.
- But if all cross-terms are $2x_ix_j$ it is in P again.

We will see how this matters to *universal quantum circuits*.

A Simple Example Over $\mathbb{Z}_4$

Consider *quadratic* polynomials $f(x_1, x_2, \ldots, x_n)$ modulo 4.

- Counting the number of zeroes is in P. (Follows by [Cai-Chen-Lipton-Luo, 2010].)
- Counting the number of zeroes in $\{0, 1\}^n$ is #P-complete.
- But if all cross-terms are $2x_i x_j$ it is in P again.

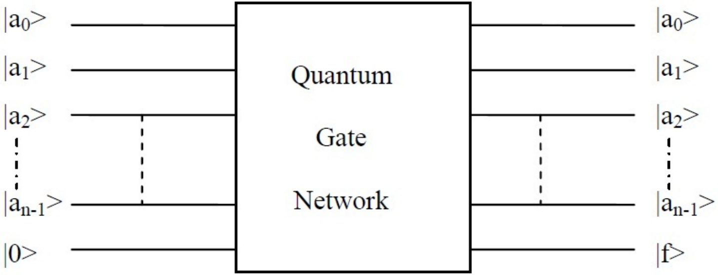
We will see how this matters to *universal quantum circuits*.
This brings up our main philosophical question:

> If there is "nothing natural" between P and #P-complete, where does that leave BQP?

(For this purpose, NP is tantamount to #P.)

Quantum Circuits

Quantum circuits look more constrained than Boolean circuits:

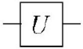


But Boolean circuits look similar if we do Savage's TM-to-circuit simulation and call each *column* for each tape cell a "cue-bit."

Quantum Gates—three slides by M. Rötteler

Quantum gates

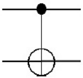
single qubit operation:



controlled-NOT:

control

target

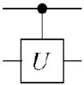


unitary matrix $= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

controlled-U:

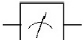
control

target



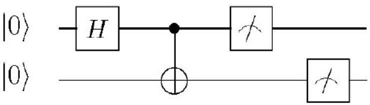
unitary matrix $= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{00} & U_{01} \\ 0 & 0 & U_{10} & U_{11} \end{pmatrix}$

measurement in the $|0\rangle, |1\rangle$ basis:



Quantum circuit example

$$H \otimes \mathbf{1}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \mathbf{1}_2$$



$\text{Prob}(\text{meas. } |00\rangle) = 0.5$

$\text{Prob}(\text{meas. } |11\rangle) = 0.5$

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle$$

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

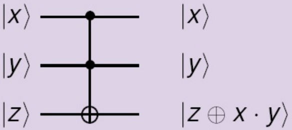
$$\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$$

$$|00\rangle \to |00\rangle$$
$$|01\rangle \to |01\rangle$$
$$|10\rangle \to |11\rangle$$
$$|11\rangle \to |10\rangle$$

Toffoli Gate

The Toffoli gate "TOF"

x	y	z	x'	y'	z'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0



Theorem (Toffoli, 1981)

Any reversible computation can be realized by using TOF gates and ancilla (auxiliary) bits which are initialized to 0.

Slides by Martin Rötteler

Some More Gates

$$\mathsf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathsf{Y} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \mathsf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

$$\mathsf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad \mathsf{T} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \quad \mathsf{R}_8 = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/8} \end{bmatrix},$$

$$\mathsf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathsf{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad \mathsf{CS} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}.$$

Some More Gates

$$\mathsf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathsf{Y} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \mathsf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

$$\mathsf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad \mathsf{T} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \quad \mathsf{R}_8 = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/8} \end{bmatrix},$$

$$\mathsf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathsf{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad \mathsf{CS} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}.$$

- The gates $\mathsf{H}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}, \mathsf{S}, \mathsf{CNOT}, \mathsf{CZ}$ generate *Clifford circuits*, which are simulable in polynomial time.

Some More Gates

$$\mathsf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathsf{Y} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \mathsf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

$$\mathsf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad \mathsf{T} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \quad \mathsf{R}_8 = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/8} \end{bmatrix},$$

$$\mathsf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathsf{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad \mathsf{CS} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}.$$

- The gates $\mathsf{H}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}, \mathsf{S}, \mathsf{CNOT}, \mathsf{CZ}$ generate *Clifford circuits*, which are simulable in polynomial time.
- Adding any of T, R_8, or CS gives the full power of BQP.

Bounded-error Quantum Poly-Time

A language A belongs to BQP if there are uniform poly-size quantum circuits C_n with n data qubits, plus some number $\alpha \geq 1$ of "ancilla qubits," such that for all n and $x \in \{0,1\}^n$,

$$\begin{aligned} x \in A &\implies \Pr[C_n \text{ given } \langle x0^\alpha| \text{ measures 1 on line } n+1] > 2/3; \\ x \notin A &\implies \Pr[\ldots] < 1/3. \end{aligned}$$

One can pretend $\alpha = 0$ and/or measure line 1 instead. One can also represent the output as the "triple product" $\langle b \mid C \mid a \rangle$, with $a = x0^\alpha$, $b = 0^{n+\alpha}$.

Two major theorems about BQP are:

(a) C_n can be composed of just Hadamard and Toffoli gates [Y. Shi].

(b) Factoring is in BQP [P. Shor].

More-general forms of a known relation

- Assume all nonzero entries $re^{i\theta}$ of gate matrices in quantum circuits C have equal magnitude $|r|$ and θ an integer multiple of $2\pi/K$.
- Suppose C has h Hadamard gates as nondeterministic games.
- Let G be a field or ring such that G^* embeds the K-th roots of unity ω^j by a multiplicative homomorphism $\iota(\omega^j)$.

More-general forms of a known relation

- Assume all nonzero entries $re^{i\theta}$ of gate matrices in quantum circuits C have equal magnitude $|r|$ and θ an integer multiple of $2\pi/K$.
- Suppose C has h Hadamard gates as nondeterministic games.
- Let G be a field or ring such that G^* embeds the K-th roots of unity ω^j by a multiplicative homomorphism $\iota(\omega^j)$.

Theorem (multiplicative form, case $G = \mathbb{F}_2$ is Dawson et al. (2004) + ...)

Any QC C of n qubits can be quickly transformed into a polynomial P_C of the form $\prod_g P_g$ and a constant $R > 0$ such that for all $x, z \in \{0,1\}^n$:

$$\langle z \mid C \mid x \rangle = \frac{1}{R} \sum_{j=0}^{K-1} \omega^j (\#y : P_C(x,y,z) = \iota(\omega^j))$$

More-general forms of a known relation

- Assume all nonzero entries $re^{i\theta}$ of gate matrices in quantum circuits C have equal magnitude $|r|$ and θ an integer multiple of $2\pi/K$.
- Suppose C has h Hadamard gates as nondeterministic games.
- Let G be a field or ring such that G^* embeds the K-th roots of unity ω^j by a multiplicative homomorphism $\iota(\omega^j)$.

Theorem (multiplicative form, case $G = \mathbb{F}_2$ is Dawson et al. (2004) + ...)

Any QC C of n qubits can be quickly transformed into a polynomial P_C of the form $\prod_g P_g$ and a constant $R > 0$ such that for all $x, z \in \{0,1\}^n$:

$$\langle z \mid C \mid x\rangle = \frac{1}{R}\sum_{j=0}^{K-1} \omega^j (\#y : P_C(x,y,z) = \iota(\omega^j)) = \frac{1}{R}\sum_y P_C(x,y,z).$$

Here g ranges over all gates and outputs of C and y ranges over $\{0,1\}^h$.

Degree is $\Theta(s)$ where s is the number of gates in C.

Additive Case

Theorem (RCG (2017), RC (2007-9), cf. Bacon-van Dam-Russell (2008))

Given C and K, we can efficiently compute a polynomial $Q_C(x_1, \ldots, x_n, y_1, \ldots, y_h, z_1, \ldots, z_n, w_1, \ldots, w_t)$ of degree $O(1)$ over $\mathbb{Z}_K$ and a constant R' such that for all $x, z \in \{0,1\}^n$:

$$\langle z \mid C \mid x \rangle = \frac{1}{R'} \sum_{j=0}^{K-1} \omega^j (\#y, w : Q_C(x, y, z, w) = j)$$

Additive Case

Theorem (RCG (2017), RC (2007-9), cf. Bacon-van Dam-Russell (2008))

Given C and K, we can efficiently compute a polynomial $Q_C(x_1, \ldots, x_n, y_1, \ldots, y_h, z_1, \ldots, z_n, w_1, \ldots, w_t)$ of degree $O(1)$ over $\mathbb{Z}_K$ and a constant R' such that for all $x, z \in \{0,1\}^n$:

$$\langle z \mid C \mid x \rangle = \frac{1}{R'} \sum_{j=0}^{K-1} \omega^j (\#y, w : Q_C(x,y,z,w) = j) = \frac{1}{R'} \sum_{y,w} \omega^{Q_C(x,y,z,w)},$$

where Q_C has the form $\sum_{gates\ g} q_g + \sum_{constraints\ c} q_c$.

Additive Case

Theorem (RCG (2017), RC (2007-9), cf. Bacon-van Dam-Russell (2008))

Given C and K, we can efficiently compute a polynomial $Q_C(x_1, \ldots, x_n, y_1, \ldots, y_h, z_1, \ldots, z_n, w_1, \ldots, w_t)$ of degree $O(1)$ over $\mathbb{Z}_K$ and a constant R' such that for all $x, z \in \{0,1\}^n$:

$$\langle z \mid C \mid x\rangle = \frac{1}{R'} \sum_{j=0}^{K-1} \omega^j (\#y, w : Q_C(x,y,z,w) = j) = \frac{1}{R'} \sum_{y,w} \omega^{Q_C(x,y,z,w)},$$

where Q_C has the form $\sum_{gates\ g} q_g + \sum_{constraints\ c} q_c$.

- Gives a particularly efficient reduction from BQP to #P.

Additive Case

Theorem (RCG (2017), RC (2007-9), cf. Bacon-van Dam-Russell (2008))

Given C and K, we can efficiently compute a polynomial $Q_C(x_1,\dots,x_n,y_1,\dots,y_h,z_1,\dots,z_n,w_1,\dots,w_t)$ of degree $O(1)$ over $\mathbb{Z}_K$ and a constant R' such that for all $x,z \in \{0,1\}^n$:

$$\langle z \mid C \mid x \rangle = \frac{1}{R'}\sum_{j=0}^{K-1} \omega^j(\#y,w : Q_C(x,y,z,w) = j) = \frac{1}{R'}\sum_{y,w} \omega^{Q_C(x,y,z,w)},$$

where Q_C has the form $\sum_{gates\ g} q_g + \sum_{constraints\ c} q_c$.

- Gives a particularly efficient reduction from BQP to #P.
- In P_C, illegal paths that violate some constraint incur the value 0.

Additive Case

Theorem (RCG (2017), RC (2007-9), cf. Bacon-van Dam-Russell (2008))

Given C and K, we can efficiently compute a polynomial $Q_C(x_1, \ldots, x_n, y_1, \ldots, y_h, z_1, \ldots, z_n, w_1, \ldots, w_t)$ of degree $O(1)$ over $\mathbb{Z}_K$ and a constant R' such that for all $x, z \in \{0,1\}^n$:

$$\langle z \mid C \mid x \rangle = \frac{1}{R'} \sum_{j=0}^{K-1} \omega^j (\#y, w : Q_C(x,y,z,w) = j) = \frac{1}{R'} \sum_{y,w} \omega^{Q_C(x,y,z,w)},$$

where Q_C has the form $\sum_{gates\ g} q_g + \sum_{constraints\ c} q_c$.

- Gives a particularly efficient reduction from BQP to #P.
- In P_C, illegal paths that violate some constraint incur the value 0.
- In Q_C, any violation creates an additive term $T = w_1 \cdots w_{\log_2 K}$ using fresh variables whose assignments give all values in $0 \,..\, K-1$, which *cancel*.

Additive Case

Theorem (RCG (2017), RC (2007-9), cf. Bacon-van Dam-Russell (2008))

Given C and K, we can efficiently compute a polynomial $Q_C(x_1, \dots, x_n, y_1, \dots, y_h, z_1, \dots, z_n, w_1, \dots, w_t)$ of degree $O(1)$ over $\mathbb{Z}_K$ and a constant R' such that for all $x, z \in \{0,1\}^n$:

$$\langle z \mid C \mid x \rangle = \frac{1}{R'} \sum_{j=0}^{K-1} \omega^j (\#y, w : Q_C(x, y, z, w) = j) = \frac{1}{R'} \sum_{y,w} \omega^{Q_C(x,y,z,w)},$$

where Q_C has the form $\sum_{gates\ g} q_g + \sum_{constraints\ c} q_c$.

- Gives a particularly efficient reduction from BQP to #P.
- In P_C, illegal paths that violate some constraint incur the value 0.
- In Q_C, any violation creates an additive term $T = w_1 \cdots w_{\log_2 K}$ using fresh variables whose assignments give all values in $0 \,..\, K-1$, which *cancel*. (This trick is my main original contribution.)

Constructing the Polynomials

Constructing the Polynomials

- Initially $P_C = 1$, $Q_C = 0$.

Constructing the Polynomials

- Initially $P_C = 1$, $Q_C = 0$.
- For Hadamard on line i (u_i—H–), allocate new variable y_j and do:

$$\begin{aligned} P_C &\mathrel{*}= (1 - u_i y_j) \\ Q_C &\mathrel{+}= 2^{k-1} u_i y_j. \end{aligned}$$

Constructing the Polynomials

- Initially $P_C = 1$, $Q_C = 0$.
- For Hadamard on line i (u_i—H–), allocate new variable y_j and do:

$$\begin{aligned} P_C &\ast= (1 - u_i y_j) \\ Q_C &+= 2^{k-1} u_i y_j. \end{aligned}$$

- CNOT with incoming terms u_i on control, u_j on target: u_i stays, $u_j := 2u_i u_j - u_i - u_j$.

Constructing the Polynomials

- Initially $P_C = 1$, $Q_C = 0$.
- For Hadamard on line i (u_i—H–), allocate new variable y_j and do:

$$\begin{aligned} P_C &\; *= \; (1 - u_i y_j) \\ Q_C &\; += \; 2^{k-1} u_i y_j. \end{aligned}$$

- CNOT with incoming terms u_i on control, u_j on target: u_i stays, $u_j := 2u_i u_j - u_i - u_j$. No change to P_C or Q_C.

Constructing the Polynomials

- Initially $P_C = 1$, $Q_C = 0$.
- For Hadamard on line i (u_i—H–), allocate new variable y_j and do:

$$\begin{aligned} P_C &\ *= & (1 - u_i y_j) \\ Q_C &\ += & 2^{k-1} u_i y_j. \end{aligned}$$

- CNOT with incoming terms u_i on control, u_j on target: u_i stays, $u_j := 2u_i u_j - u_i - u_j$. No change to P_C or Q_C.
- S-gate: Q_C adds u_i^2.

Constructing the Polynomials

- Initially $P_C = 1$, $Q_C = 0$.
- For Hadamard on line i (u_i—H–), allocate new variable y_j and do:

$$\begin{aligned} P_C &\mathrel{*}= (1 - u_i y_j) \\ Q_C &\mathrel{+}= 2^{k-1} u_i y_j. \end{aligned}$$

- CNOT with incoming terms u_i on control, u_j on target: u_i stays, $u_j := 2u_i u_j - u_i - u_j$. No change to P_C or Q_C.
- S-gate: Q_C adds u_i^2.
- CS-gate: Q_C adds $u_i u_j$.

Constructing the Polynomials

- Initially $P_C = 1$, $Q_C = 0$.
- For Hadamard on line i (u_i—H–), allocate new variable y_j and do:

$$\begin{aligned} P_C &\; *= \; (1 - u_i y_j) \\ Q_C &\; += \; 2^{k-1} u_i y_j. \end{aligned}$$

- CNOT with incoming terms u_i on control, u_j on target: u_i stays, $u_j := 2u_i u_j - u_i - u_j$. No change to P_C or Q_C.
- S-gate: Q_C adds u_i^2.
- CS-gate: Q_C adds $u_i u_j$.
- Thereby CS escapes the easy case over $\mathbb{Z}_4$ (with $k = 2$).

Constructing the Polynomials

- Initially $P_C = 1$, $Q_C = 0$.
- For Hadamard on line i (u_i—H–), allocate new variable y_j and do:

$$\begin{aligned} P_C &\;*= & (1 - u_i y_j) \\ Q_C &\;+= & 2^{k-1} u_i y_j. \end{aligned}$$

- CNOT with incoming terms u_i on control, u_j on target: u_i stays, $u_j := 2u_iu_j - u_i - u_j$. No change to P_C or Q_C.
- S-gate: Q_C adds u_i^2.
- CS-gate: Q_C adds u_iu_j.
- Thereby CS escapes the easy case over $\mathbb{Z}_4$ (with $k = 2$).
- TOF: controls u_i, u_j stay, target u_k changes to $2u_iu_ju_k -$ u_iu_j $-u_k$.

Constructing the Polynomials

- Initially $P_C = 1$, $Q_C = 0$.
- For Hadamard on line i (u_i—H–), allocate new variable y_j and do:

$$\begin{aligned} P_C &\ast= (1 - u_i y_j) \\ Q_C &+= 2^{k-1} u_i y_j. \end{aligned}$$

- CNOT with incoming terms u_i on control, u_j on target: u_i stays, $u_j := 2u_iu_j - u_i - u_j$. No change to P_C or Q_C.
- S-gate: Q_C adds u_i^2.
- CS-gate: Q_C adds u_iu_j.
- Thereby CS escapes the easy case over $\mathbb{Z}_4$ (with $k = 2$).
- TOF: controls u_i, u_j stay, target u_k changes to $2u_iu_ju_k -$ u_iu_j $-u_k$.
- T-gate also goes cubic.

Gottesman-Knill: alternative methodology

Gottesman-Knill: alternative methodology

- To represent u_i—S— we need $K = 4$.

Gottesman-Knill: alternative methodology

- To represent u_i—S— we need $K = 4$.
- H gives $Q_C \mathrel{+}= 2u_i y_j$.

Gottesman-Knill: alternative methodology

- To represent u_i—S— we need $K = 4$.
- H gives $Q_C \mathrel{+}= 2u_i y_j$.
- CNOT: Nonlinear term has a 2 which will cancel the 2 from Hadamard.

Gottesman-Knill: alternative methodology

- To represent u_i—S— we need $K = 4$.
- H gives $Q_C \mathrel{+}= 2u_i y_j$.
- CNOT: Nonlinear term has a 2 which will cancel the 2 from Hadamard.
- Equality constraint $w_j(u_i + z_i - 2u_i z_i)$: OK with [G-K], [CCLL] because w_j appears only here.

Gottesman-Knill: alternative methodology

- To represent u_i—S— we need $K = 4$.
- H gives $Q_C \mathrel{+}= 2u_iy_j$.
- CNOT: Nonlinear term has a 2 which will cancel the 2 from Hadamard.
- Equality constraint $w_j(u_i + z_i - 2u_iz_i)$: OK with [G-K], [CCLL] because w_j appears only here.
- S: u_i left alone but $Q_C \mathrel{+}= u_i^2$.

Gottesman-Knill: alternative methodology

- To represent u_i—S— we need $K = 4$.
- H gives $Q_C \mathrel{+}= 2u_i y_j$.
- CNOT: Nonlinear term has a 2 which will cancel the 2 from Hadamard.
- Equality constraint $w_j(u_i + z_i - 2u_i z_i)$: OK with [G-K], [CCLL] because w_j appears only here.
- S: u_i left alone but $Q_C \mathrel{+}= u_i^2$.
- Inductively every term in Q_C has form y_j^2 or $2y_i y_j$.

Gottesman-Knill: alternative methodology

- To represent u_i—S— we need $K = 4$.
- H gives $Q_C \mathrel{+}= 2u_i y_j$.
- CNOT: Nonlinear term has a 2 which will cancel the 2 from Hadamard.
- Equality constraint $w_j(u_i + z_i - 2u_i z_i)$: OK with [G-K], [CCLL] because w_j appears only here.
- S: u_i left alone but $Q_C \mathrel{+}= u_i^2$.
- Inductively every term in Q_C has form y_j^2 or $2y_i y_j$.
- These terms are invariant under $0 \leftrightarrow 2$, $1 \leftrightarrow 3$.

Gottesman-Knill: alternative methodology

- To represent u_i—S— we need $K = 4$.
- H gives $Q_C \mathrel{+}= 2u_i y_j$.
- CNOT: Nonlinear term has a 2 which will cancel the 2 from Hadamard.
- Equality constraint $w_j(u_i + z_i - 2u_i z_i)$: OK with [G-K], [CCLL] because w_j appears only here.
- S: u_i left alone but $Q_C \mathrel{+}= u_i^2$.
- Inductively every term in Q_C has form y_j^2 or $2y_i y_j$.
- These terms are invariant under $0 \leftrightarrow 2$, $1 \leftrightarrow 3$.
- Hence poly-time simulation by solution counting in $\mathbb{Z}_4$.

Overpowered for Universal Quantum Circuits

Overpowered for Universal Quantum Circuits

- When we have a universal gate sets, these simulations zoom to #P-complete cases.

Overpowered for Universal Quantum Circuits

- When we have a universal gate sets, these simulations zoom to #P-complete cases.
- Chaowen Guan devised and programmed a simulation via Boolean formulas, but #SAT is #P-complete.

Overpowered for Universal Quantum Circuits

- When we have a universal gate sets, these simulations zoom to #P-complete cases.
- Chaowen Guan devised and programmed a simulation via Boolean formulas, but #SAT is #P-complete.
- Does not seem to reveal a "natural" subset B of Boolean formulas for which $\#B$ is equivalent to BQP.

Overpowered for Universal Quantum Circuits

- When we have a universal gate sets, these simulations zoom to #P-complete cases.
- Chaowen Guan devised and programmed a simulation via Boolean formulas, but #SAT is #P-complete.
- Does not seem to reveal a "natural" subset B of Boolean formulas for which $\#B$ is equivalent to BQP.
- The Bremner-Jozsa-Shepherd IQP circuits are a postulated intermediate class, but even their simulation collapses the polynomial hierarchy.

Overpowered for Universal Quantum Circuits

- When we have a universal gate sets, these simulations zoom to #P-complete cases.
- Chaowen Guan devised and programmed a simulation via Boolean formulas, but #SAT is #P-complete.
- Does not seem to reveal a "natural" subset B of Boolean formulas for which $\#B$ is equivalent to BQP.
- The Bremner-Jozsa-Shepherd IQP circuits are a postulated intermediate class, but even their simulation collapses the polynomial hierarchy.
- IQP circuits use Hadamard gates only at the beginning and end of the circuit, CS gates, and diagonal one-qubit gates.

Overpowered for Universal Quantum Circuits

- When we have a universal gate sets, these simulations zoom to #P-complete cases.
- Chaowen Guan devised and programmed a simulation via Boolean formulas, but #SAT is #P-complete.
- Does not seem to reveal a “natural” subset B of Boolean formulas for which $\#B$ is equivalent to BQP.
- The Bremner-Jozsa-Shepherd IQP circuits are a postulated intermediate class, but even their simulation collapses the polynomial hierarchy.
- IQP circuits use Hadamard gates only at the beginning and end of the circuit, CS gates, and diagonal one-qubit gates.
- The same idea with only CZ and (optionally, for self-loops) Z gates between the two banks of Hadamards are called **graph-state circuits**, and are equivalent to general Clifford circuits in power.

Rest of Talk

Rest of Talk

- Show https://rjlipton.com/2022/01/05/quantum-graph-theory/

Rest of Talk

- Show https://rjlipton.com/2022/01/05/quantum-graph-theory/
- Show https://rjlipton.com/2019/06/17/contraction-and-explosion/

Rest of Talk

- Show https://rjlipton.com/2022/01/05/quantum-graph-theory/
- Show https://rjlipton.com/2019/06/17/contraction-and-explosion/
- Show https://rjlipton.com/2019/08/26/a-matroid-quantum-connection/

Rest of Talk

- Show https://rjlipton.com/2022/01/05/quantum-graph-theory/
- Show https://rjlipton.com/2019/06/17/contraction-and-explosion/
- Show https://rjlipton.com/2019/08/26/a-matroid-quantum-connection/
- A graph can be viewed as a **polymatroid** in which the **rank** of an edge subset $A \subseteq E$ is the number of nodes involved in A.

Rest of Talk

- Show https://rjlipton.com/2022/01/05/quantum-graph-theory/
- Show https://rjlipton.com/2019/06/17/contraction-and-explosion/
- Show https://rjlipton.com/2019/08/26/a-matroid-quantum-connection/
- A graph can be viewed as a **polymatroid** in which the **rank** of an edge subset $A \subseteq E$ is the number of nodes involved in A.
- Augment the idea with “half loops” and “half edges” for S and CS, respectively.

Rest of Talk

- Show https://rjlipton.com/2022/01/05/quantum-graph-theory/
- Show https://rjlipton.com/2019/06/17/contraction-and-explosion/
- Show https://rjlipton.com/2019/08/26/a-matroid-quantum-connection/
- A graph can be viewed as a **polymatroid** in which the **rank** of an edge subset $A \subseteq E$ is the number of nodes involved in A.
- Augment the idea with "half loops" and "half edges" for S and CS, respectively.
- General observations—how wide are the possibilities and prospects?

-

-

More Ideas and the Logic Side

More Ideas and the Logic Side

- Idea is to postpone exponential blowup until the end...

More Ideas and the Logic Side

- Idea is to postpone exponential blowup until the end...
- ...when a full spec can be fed to equation solvers or SAT solvers.

More Ideas and the Logic Side

- Idea is to postpone exponential blowup until the end...
- ...when a full spec can be fed to equation solvers or SAT solvers.
- Algebraic side is joint work with Amlan Chakrabarti (U. Calcutta) since 2007.

More Ideas and the Logic Side

- Idea is to postpone exponential blowup until the end...
- ...when a full spec can be fed to equation solvers or SAT solvers.
- Algebraic side is joint work with Amlan Chakrabarti (U. Calcutta) since 2007.
- Logical side with Chaowen Guan, UB.

More Ideas and the Logic Side

- Idea is to postpone exponential blowup until the end...
- ...when a full spec can be fed to equation solvers or SAT solvers.
- Algebraic side is joint work with Amlan Chakrabarti (U. Calcutta) since 2007.
- Logical side with Chaowen Guan, UB. Jointly became paper [RCG18] in *Transactions on Computational Science*, 2018.

More Ideas and the Logic Side

- Idea is to postpone exponential blowup until the end…
- …when a full spec can be fed to equation solvers or SAT solvers.
- Algebraic side is joint work with Amlan Chakrabarti (U. Calcutta) since 2007.
- Logical side with Chaowen Guan, UB. Jointly became paper [RCG18] in *Transactions on Computational Science*, 2018.
- Logic-based full QC simulator, 8,000+ lines of C++. [show demo]

Theoretical Advance: Quadratic Equations over $\mathbb{F}_2$

Theoretical Advance: Quadratic Equations over $\mathbb{F}_2$

- *Stabilizer circuits* ($\equiv$ *Clifford circuits*) are the most salient classically solvable case.

Theoretical Advance: Quadratic Equations over $\mathbb{F}_2$

- *Stabilizer circuits* ($\equiv$ *Clifford circuits*) are the most salient classically solvable case.
- Vital in quantum error-correcting codes for fault-tolerant QC.

Theoretical Advance: Quadratic Equations over $\mathbb{F}_2$

- *Stabilizer circuits* ($\equiv$ *Clifford circuits*) are the most salient classically solvable case.
- Vital in quantum error-correcting codes for fault-tolerant QC.
- Classical simulation of n qubits takes $O(n^2)$ time per single-qubit measurement [Aaronson-Gottesman, 2004], $O(n^3)$ time to measure all n qubits.

Theoretical Advance: Quadratic Equations over $\mathbb{F}_2$

- *Stabilizer circuits* ($\equiv$ *Clifford circuits*) are the most salient classically solvable case.
- Vital in quantum error-correcting codes for fault-tolerant QC.
- Classical simulation of n qubits takes $O(n^2)$ time per single-qubit measurement [Aaronson-Gottesman, 2004], $O(n^3)$ time to measure all n qubits.
- We improve to time $O(n^\omega)$ where $\omega < 2.3729$ is the known exponent for $n \times n$ matrix multiplication.

Theoretical Advance: Quadratic Equations over $\mathbb{F}_2$

- *Stabilizer circuits* ($\equiv$ *Clifford circuits*) are the most salient classically solvable case.
- Vital in quantum error-correcting codes for fault-tolerant QC.
- Classical simulation of n qubits takes $O(n^2)$ time per single-qubit measurement [Aaronson-Gottesman, 2004], $O(n^3)$ time to measure all n qubits.
- We improve to time $O(n^\omega)$ where $\omega < 2.3729$ is the known exponent for $n \times n$ matrix multiplication.
- Also give $O(N)$-time reduction ($N = n^2$) from computing $n \times n$ matrix rank over $\mathbb{F}_2$ to the QC simulation.

Theoretical Advance: Quadratic Equations over $\mathbb{F}_2$

- *Stabilizer circuits* ($\equiv$ *Clifford circuits*) are the most salient classically solvable case.
- Vital in quantum error-correcting codes for fault-tolerant QC.
- Classical simulation of n qubits takes $O(n^2)$ time per single-qubit measurement [Aaronson-Gottesman, 2004], $O(n^3)$ time to measure all n qubits.
- We improve to time $O(n^\omega)$ where $\omega < 2.3729$ is the known exponent for $n \times n$ matrix multiplication.
- Also give $O(N)$-time reduction ($N = n^2$) from computing $n \times n$ matrix rank over $\mathbb{F}_2$ to the QC simulation.
- Means that the n^2-vs.-n^ω weak/strong simulation gap canot be closed unless matrix rank is in $O(n^2)$ time over $\mathbb{F}_2$.

How Achieved

How Achieved

- Stabilizer circuits C yield *classical* quadratic forms q_C over $\mathbb{Z}_4$.

How Achieved

- Stabilizer circuits C yield *classical* quadratic forms q_C over $\mathbb{Z}_4$.
- Exploit normal form q' for q_C by Schmidt [2009].

How Achieved

- Stabilizer circuits C yield *classical* quadratic forms q_C over $\mathbb{Z}_4$.
- Exploit normal form q' for q_C by Schmidt [2009].
- Apply new algorithm for LDU decompositions over $\mathbb{F}_2$ by Dumas-Pernet [2018].

How Achieved

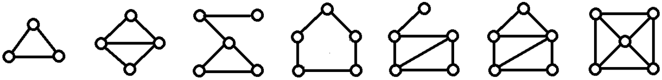
- Stabilizer circuits C yield *classical* quadratic forms q_C over $\mathbb{Z}_4$.
- Exploit normal form q' for q_C by Schmidt [2009].
- Apply new algorithm for LDU decompositions over $\mathbb{F}_2$ by Dumas-Pernet [2018].
- Invert the LDU process but calculating in $\mathbb{Z}_4$.

How Achieved

- Stabilizer circuits C yield *classical* quadratic forms q_C over $\mathbb{Z}_4$.
- Exploit normal form q' for q_C by Schmidt [2009].
- Apply new algorithm for LDU decompositions over $\mathbb{F}_2$ by Dumas-Pernet [2018].
- Invert the LDU process but calculating in $\mathbb{Z}_4$.
- Painstaking analysis of how distributions of values were mapped yields a simple recursion then gives the result from the final "spectrum."

How Achieved

- Stabilizer circuits C yield *classical* quadratic forms q_C over $\mathbb{Z}_4$.
- Exploit normal form q' for q_C by Schmidt [2009].
- Apply new algorithm for LDU decompositions over $\mathbb{F}_2$ by Dumas-Pernet [2018].
- Invert the LDU process but calculating in $\mathbb{Z}_4$.
- Painstaking analysis of how distributions of values were mapped yields a simple recursion then gives the result from the final "spectrum."
- Also yields an apparently new class of undirected graphs:



Boolean Logic Simulation

Boolean Logic Simulation

- Allocate free variables x_i for every input (qu)bit, z_i for corresponding outputs, and y_j for every nondeterministic gate (wlog. Hadamard gate).

Boolean Logic Simulation

- Allocate free variables x_i for every input (qu)bit, z_i for corresponding outputs, and y_j for every nondeterministic gate (wlog. Hadamard gate).
- Also maintain "forced" variables giving the current *phase* and *location* of every Feynman path.

Boolean Logic Simulation

- Allocate free variables x_i for every input (qu)bit, z_i for corresponding outputs, and y_j for every nondeterministic gate (wlog. Hadamard gate).
- Also maintain "forced" variables giving the current *phase* and *location* of every Feynman path.
- Translation from circuit C to Boolean formula ϕ_C is again real-time.

Boolean Logic Simulation

- Allocate free variables x_i for every input (qu)bit, z_i for corresponding outputs, and y_j for every nondeterministic gate (wlog. Hadamard gate).
- Also maintain "forced" variables giving the current *phase* and *location* of every Feynman path.
- Translation from circuit C to Boolean formula ϕ_C is again real-time.
- Solution counts over each phase for a target location yield its amplitude.

Boolean Logic Simulation

- Allocate free variables x_i for every input (qu)bit, z_i for corresponding outputs, and y_j for every nondeterministic gate (wlog. Hadamard gate).
- Also maintain “forced” variables giving the current *phase* and *location* of every Feynman path.
- Translation from circuit C to Boolean formula ϕ_C is again real-time.
- Solution counts over each phase for a target location yield its amplitude.
- *#SAT solvers* such as *sharpSAT* and *Cachet* give hope of heuristic simulations of harder classes of circults.

Boolean Logic Simulation

- Allocate free variables x_i for every input (qu)bit, z_i for corresponding outputs, and y_j for every nondeterministic gate (wlog. Hadamard gate).
- Also maintain "forced" variables giving the current *phase* and *location* of every Feynman path.
- Translation from circuit C to Boolean formula ϕ_C is again real-time.
- Solution counts over each phase for a target location yield its amplitude.
- *#SAT solvers* such as *sharpSAT* and *Cachet* give hope of heuristic simulations of harder classes of circults.
- Our C++ simulator outputs DIMACS-compliant files for these solvers. SAT solvers have seen great success in many areas

Boolean Logic Simulation

- Allocate free variables x_i for every input (qu)bit, z_i for corresponding outputs, and y_j for every nondeterministic gate (wlog. Hadamard gate).
- Also maintain "forced" variables giving the current *phase* and *location* of every Feynman path.
- Translation from circuit C to Boolean formula ϕ_C is again real-time.
- Solution counts over each phase for a target location yield its amplitude.
- *#SAT solvers* such as *sharpSAT* and *Cachet* give hope of heuristic simulations of harder classes of circults.
- Our C++ simulator outputs DIMACS-compliant files for these solvers. SAT solvers have seen great success in many areas, but maybe not QC...

Boolean Logic Simulation

- Allocate free variables x_i for every input (qu)bit, z_i for corresponding outputs, and y_j for every nondeterministic gate (wlog. Hadamard gate).
- Also maintain "forced" variables giving the current *phase* and *location* of every Feynman path.
- Translation from circuit C to Boolean formula ϕ_C is again real-time.
- Solution counts over each phase for a target location yield its amplitude.
- *#SAT solvers* such as *sharpSAT* and *Cachet* give hope of heuristic simulations of harder classes of circults.
- Our C++ simulator outputs DIMACS-compliant files for these solvers. SAT solvers have seen great success in many areas, but maybe not QC...
- Second main purpose of simulator [show] is to enable tinkering with approximative methods.

Higher Algebra and Applications

Higher Algebra and Applications

- Invariants based on Strassen's *geometric degree* $\gamma(f)$ concept may help quantify both entanglement and effort to keep coherence.

Higher Algebra and Applications

- Invariants based on Strassen's *geometric degree* $\gamma(f)$ concept may help quantify both entanglement and effort to keep coherence.
- Baur-Strassen showed that $\Omega(\log_2 \gamma(f))$ lower-bounds the arithmetical complexity of f, indeed the number of binary multiplication gates.

Higher Algebra and Applications

- Invariants based on Strassen's *geometric degree* $\gamma(f)$ concept may help quantify both entanglement and effort to keep coherence.
- Baur-Strassen showed that $\Omega(\log_2 \gamma(f))$ lower-bounds the arithmetical complexity of f, indeed the number of binary multiplication gates. Apply similar to quantum circuits?

Higher Algebra and Applications

- Invariants based on Strassen's *geometric degree* $\gamma(f)$ concept may help quantify both entanglement and effort to keep coherence.
- Baur-Strassen showed that $\Omega(\log_2 \gamma(f))$ lower-bounds the arithmetical complexity of f, indeed the number of binary multiplication gates. Apply similar to quantum circuits?
- Already hard to formulate n-partite entanglement of (pure or mixed) *states*.

Higher Algebra and Applications

- Invariants based on Strassen's *geometric degree* $\gamma(f)$ concept may help quantify both entanglement and effort to keep coherence.
- Baur-Strassen showed that $\Omega(\log_2 \gamma(f))$ lower-bounds the arithmetical complexity of f, indeed the number of binary multiplication gates. Apply similar to quantum circuits?
- Already hard to formulate n-partite entanglement of (pure or mixed) *states*. How to define for *circuits*?

Higher Algebra and Applications

- Invariants based on Strassen's *geometric degree* $\gamma(f)$ concept may help quantify both entanglement and effort to keep coherence.
- Baur-Strassen showed that $\Omega(\log_2 \gamma(f))$ lower-bounds the arithmetical complexity of f, indeed the number of binary multiplication gates. Apply similar to quantum circuits?
- Already hard to formulate n-partite entanglement of (pure or mixed) *states*. How to define for *circuits*? Plausible axioms:

$$\begin{aligned} e(C^*) &= e(C), \\ e(C_1 \otimes C_2) &= e(C_1) + e(C_2), \\ e(C; \mathit{measure}) &\leq e(C), \\ e(C + \text{LOCC}) &= e(C) \end{aligned}$$

Higher Algebra and Applications

- Invariants based on Strassen's *geometric degree* $\gamma(f)$ concept may help quantify both entanglement and effort to keep coherence.
- Baur-Strassen showed that $\Omega(\log_2 \gamma(f))$ lower-bounds the arithmetical complexity of f, indeed the number of binary multiplication gates. Apply similar to quantum circuits?
- Already hard to formulate n-partite entanglement of (pure or mixed) *states*. How to define for *circuits*? Plausible axioms:

$$\begin{aligned} e(C^*) &= e(C), \\ e(C_1 \otimes C_2) &= e(C_1) + e(C_2), \\ e(C; \textit{measure}) &\leq e(C), \\ e(C + \mathrm{LOCC}) &= e(C) \end{aligned}$$

- Singular points of varieties determine (most of?) amplitude under the Principle of Least Action, conjectured by Bacon, van Dam, and Russell [2008, unpublished]).

Summary

Summary

- Novel research ideas.

Summary

- Novel research ideas.
- Development of program infrastructure to experiment with them.

Summary

- Novel research ideas.
- Development of program infrastructure to experiment with them.
- Indo-US collaboration.

Summary

- Novel research ideas.
- Development of program infrastructure to experiment with them.
- Indo-US collaboration.
- References: *Gödel's Lost Letter* blog, textbook with MIT Press.

Summary

- Novel research ideas.
- Development of program infrastructure to experiment with them.
- Indo-US collaboration.
- References: *Gödel's Lost Letter* blog, textbook with MIT Press.
- Some other ideas there: chaotic walks on graphs, quantum graph networks.

Summary

- Novel research ideas.
- Development of program infrastructure to experiment with them.
- Indo-US collaboration.
- References: *Gödel's Lost Letter* blog, textbook with MIT Press.
- Some other ideas there: chaotic walks on graphs, quantum graph networks.
- Greater relation to tensor network simulations of quantum circuits?

Summary

- Novel research ideas.
- Development of program infrastructure to experiment with them.
- Indo-US collaboration.
- References: *Gödel's Lost Letter* blog, textbook with MIT Press.
- Some other ideas there: chaotic walks on graphs, quantum graph networks.
- Greater relation to tensor network simulations of quantum circuits?
- Involvement in the general debate over *Quantum Advantage*.