Quantum Proofs of Proximity

Marcel Dall'Agnol
University of Warwick

Tom Gur
University of Warwick

Subhayan Roy Moulik
University of Oxford
& UC Berkeley

Justin Thaler
Georgetown University

BCTCS 2021

Introduction

Part I: Algorithms

Part II: Complexity separations

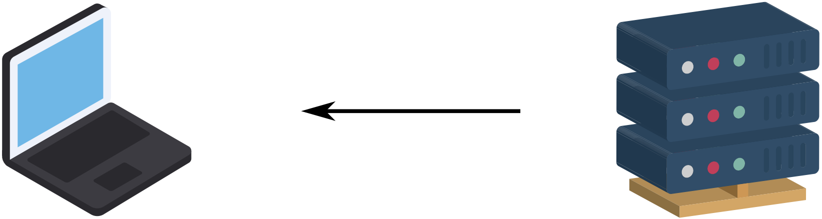
Introduction

Part I: Algorithms

Part II: Complexity separations

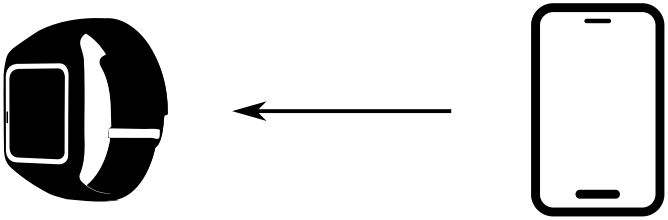
Introduction

Massive datasets, IoT: devices with *dramatically* different power.



Introduction

Massive datasets, IoT: devices with *dramatically* different power.



Massive datasets, IoT: devices with *dramatically* different power.

$$V \longleftarrow P$$

Introduction

Massive datasets, IoT: devices with *dramatically* different power.

$$V \longleftarrow P$$

Delegation of computation: *prover* computes, *verifier* checks.

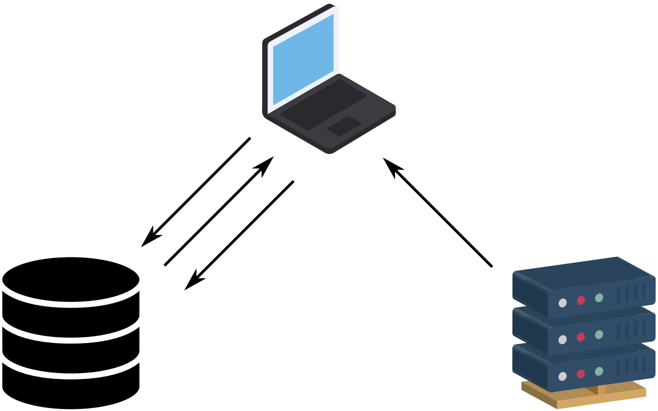
Introduction

Massive datasets, IoT: devices with *dramatically* different power.

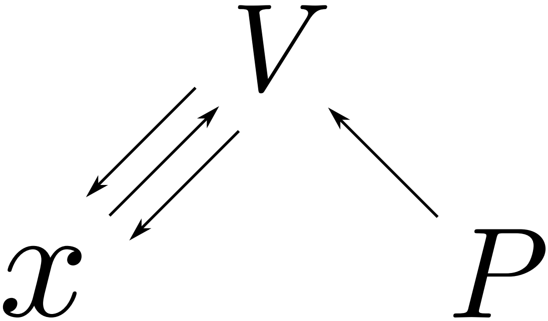
$$V \longleftarrow P$$

Delegation of computation: *prover* computes, *verifier* checks.
Efficient: $\tilde{O}(n)$ verifier runtime.

Proofs. . .

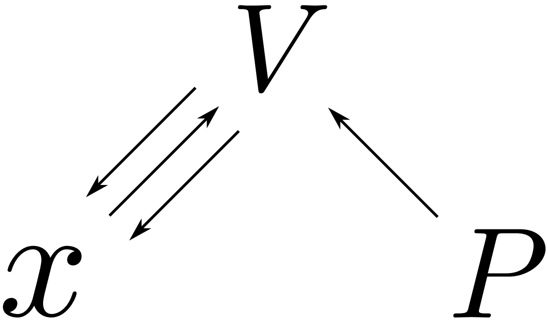


Proofs...



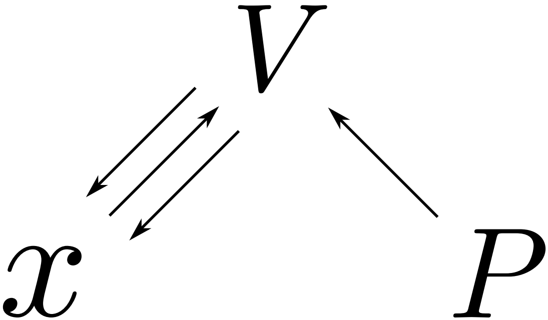
V checks if $x \in L$ by *querying* x with a *proof* from P.

Proofs. . .



V checks if $x \in L$ by *querying* x with a *proof* from P.
Query complexity q, proof complexity p.

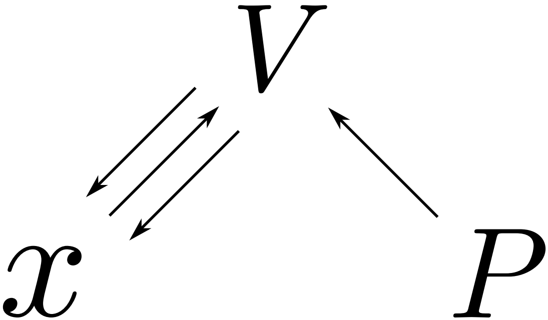
Proofs...



V checks if $x \in L$ by *querying* x with a *proof* from P.
Efficient: $o(n)$ queries and proof length.

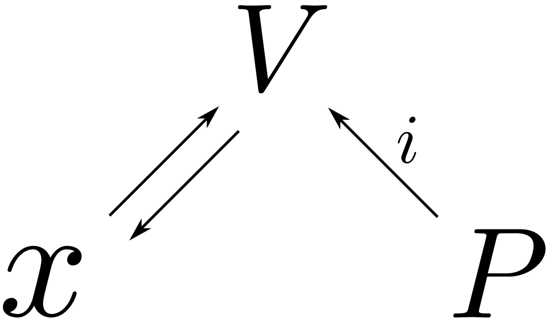
Proofs. . .

Example: Does $x \in \{0,1\}^n$ contain a 1?

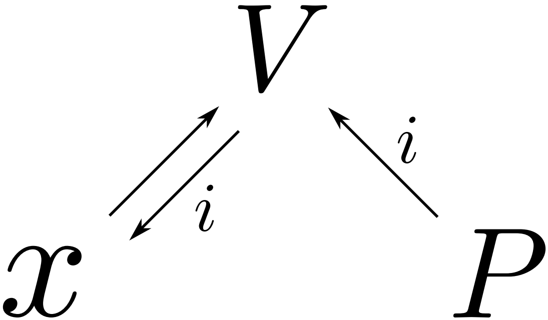


Proofs...

Example: Does $x \in \{0,1\}^n$ contain a 1?

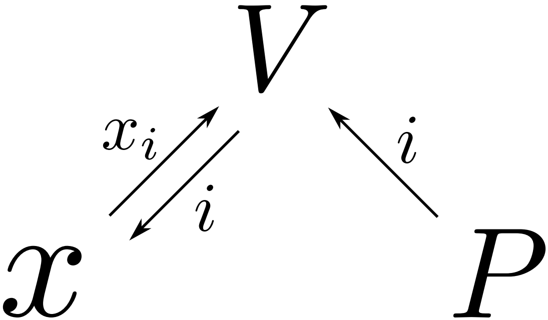


Example: Does $x \in \{0,1\}^n$ contain a 1?



Proofs. . .

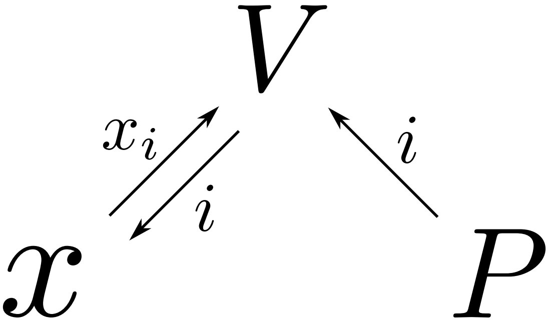
Example: Does $x \in \{0,1\}^n$ contain a 1?



Query complexity 1, proof complexity $\log n$.

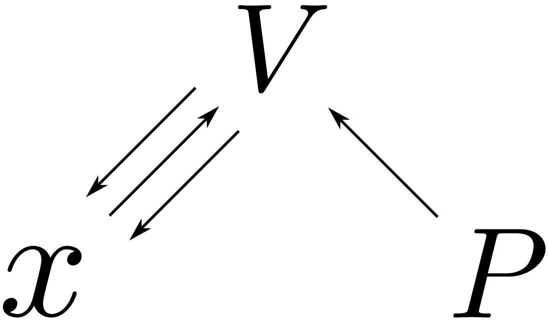
Proofs...

Example: Does $x \in \{0,1\}^n$ contain a 1?

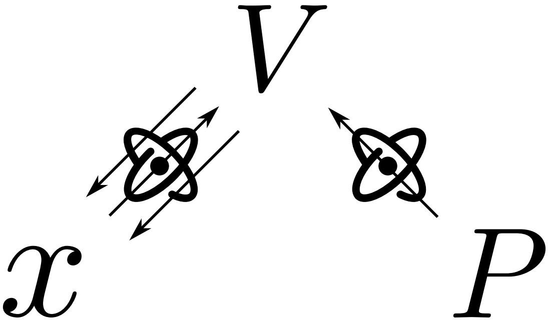


Query complexity 1, proof complexity $\log n$.
($\Omega(n)$ with no proof!)

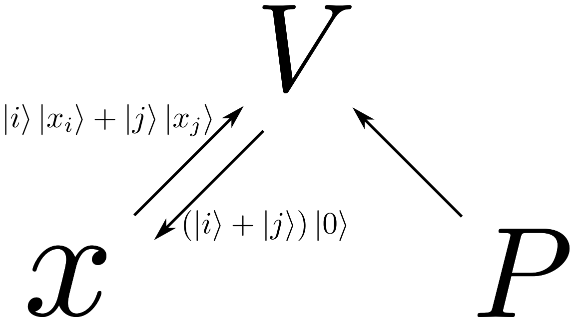
. . . of proximity. . .



V checks if $x \in \Pi$ **or x is ε-far from** Π (property testing).
Query complexity q, proof complexity p.



V checks if $x \in \Pi$ **or x is ε-far from** Π (property testing).
Query complexity q, proof complexity p.


V
$|i\rangle |x_i\rangle + |j\rangle |x_j\rangle$
$(|i\rangle + |j\rangle) |0\rangle$
x
P

Introduction

Part I: Algorithms

Part II: Complexity separations

Theorem (Amplitude amplification [Brassard et al., 2002])

If a one-sided randomised algorithm makes q queries and detects an error with probability ρ, there is a quantum algorithm making $O(q/\sqrt{\rho})$ queries that succeeds w. p. $2/3$.

Problem: Verify if $x \in \{0,1\}^n$ has even parity.

Problem: Verify if $x \in \{0,1\}^n$ has even parity.

Classically, we're out of luck: $\Omega(n)$ with any proof.

Problem: Verify if $x \in \{0,1\}^n$ has even parity.

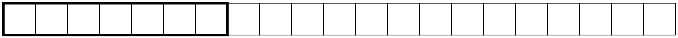
Classically, we're out of luck: $\Omega(n)$ with any proof.
Quantumly, an $n^{2/3}$-bit proof and $O(n^{2/3})$ queries suffice!

Problem: Verify if $x \in \{0,1\}^n$ has even parity.

$x =$ 

If the parity of the proof π is odd, reject.
Sample $i \in [p]$ uniformly and query the i^{th} block of n/p bits. Accept if their parity matches π_i, reject otherwise.

Problem: Verify if $x \in \{0,1\}^n$ has even parity.

$x =$ 

If the parity of the proof π is odd, reject.
Sample $i \in [p]$ uniformly and query the i^{th} block of n/p bits. Accept if their parity matches π_i, reject otherwise.

Problem: Verify if $x \in \{0,1\}^n$ has even parity.

$x =$ 

If the parity of the proof π is odd, reject.
Sample $i \in [p]$ uniformly and query the i^{th} block of n/p bits. Accept if their parity matches π_i, reject otherwise.

Problem: Verify if $x \in \{0,1\}^n$ has even parity.

$x =$ 

If the parity of the proof π is odd, reject.
Sample $i \in [p]$ uniformly and query the i^{th} block of n/p bits. Accept if their parity matches π_i, reject otherwise.

Problem: Verify if $x \in \{0,1\}^n$ has even parity.

$x =$ | | | | | | | | | | | | | | | | | | | | | |

$\pi = \ 1 \ 1 \ 0$

If the parity of the proof π is odd, reject.
Sample $i \in [p]$ uniformly and query the i^{th} block of n/p bits. Accept if their parity matches π_i, reject otherwise.

Problem: Verify if $x \in \{0,1\}^n$ has even parity.

$x =$ 

$\pi = 1\ 1\ 0$

If the parity of the proof π is odd, reject.
Sample $i \in [p]$ uniformly and query the i^{th} block of n/p bits. Accept if their parity matches π_i, reject otherwise.

Problem: Verify if $x \in \{0,1\}^n$ has even parity.

$$x = \square\,\square\,\square\,\square\,\square\,\square\,\square\,\boxed{1\,|\,1\,|\,0\,|\,0\,|\,1\,|\,0\,|\,0}\,\square\,\square\,\square\,\square\,\square\,\square\,\square$$

$$\pi = 1\ \ 1\ \ 0$$

If the parity of the proof π is odd, reject.
Sample $i \in [p]$ uniformly and query the i^{th} block of n/p bits. Accept if their parity matches π_i, reject otherwise.

Problem: Verify if $x \in \{0,1\}^n$ has even parity.

$$x = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline & & & & & & & & & & & & & \\ \hline \end{array}\boxed{\begin{array}{c|c|c|c|c|c|c} 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}}$$

$$\pi = \;1\;\;1\;\;0$$

If the parity of the proof π is odd, reject.
Sample $i \in [p]$ uniformly and query the i^{th} block of n/p bits. Accept if their parity matches π_i, reject otherwise.

Theorem (Amplitude amplification)

q queries	$\Longrightarrow$	*$q/\sqrt{\rho}$ queries*
ρ detection probability		*2/3 detection probability*

Theorem (Amplitude amplification)

q queries *ρ detection probability*	$\Longrightarrow$	*$q/\sqrt{\rho}$ queries* *$2/3$ detection probability*

$$\Downarrow$$

Seting $p = n^{2/3}$ in the previous algorithm, it makes $n^{1/3}$ queries to detect an error w. p. $1/n^{2/3}$. Therefore,

$$q = O\left(\frac{n^{1/3}}{\sqrt{1/n^{2/3}}}\right) = O(n^{2/3}).$$

Main result

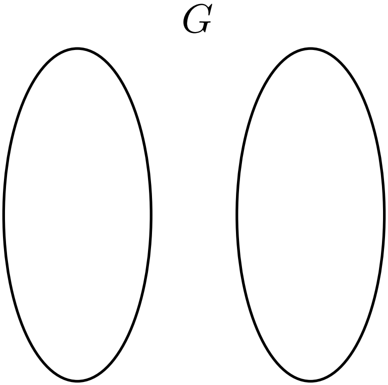
Theorem

A similar strategy works for every decomposable property.

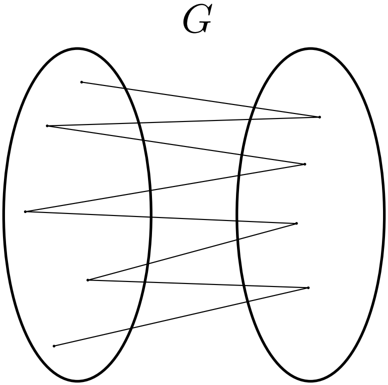
(Includes k-monotonicity, acceptance by branching programs, membership in context-free languages, and Eulerian graph orientations.)

Problem: Verify if the graph G is bipartite or ε-*far* from it.

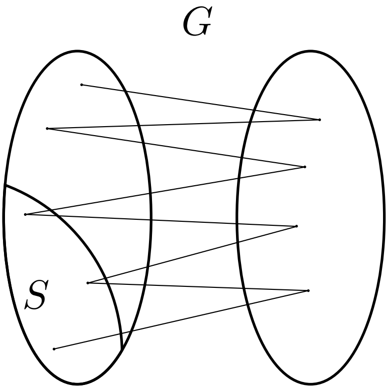
Problem: Verify if the graph G is bipartite or ε-*far* from it.



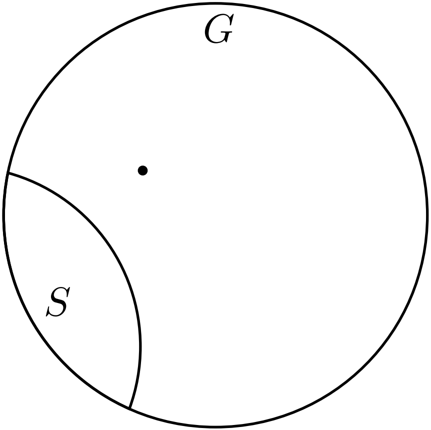
Problem: Verify if the graph G is bipartite or ε-*far* from it.



Problem: Verify if the graph G is bipartite or ε-*far* from it.

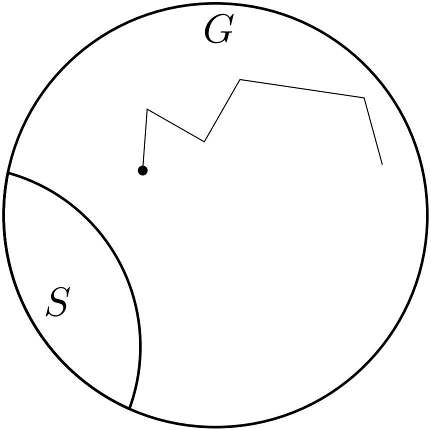


Problem: Verify if the graph G is bipartite or ε-*far* from it.



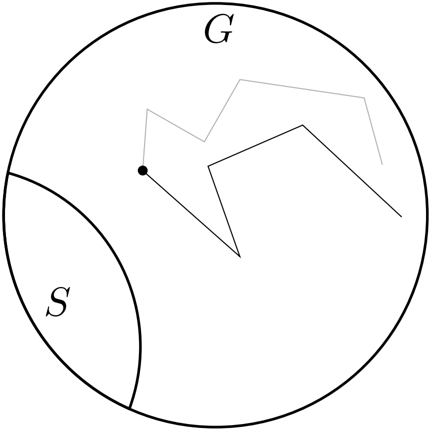
Sample a vertex, take many random walks from it and check whether any pair of walks ends at S with different parities. If so, reject, and accept otherwise.

Problem: Verify if the graph G is bipartite or ε-*far* from it.



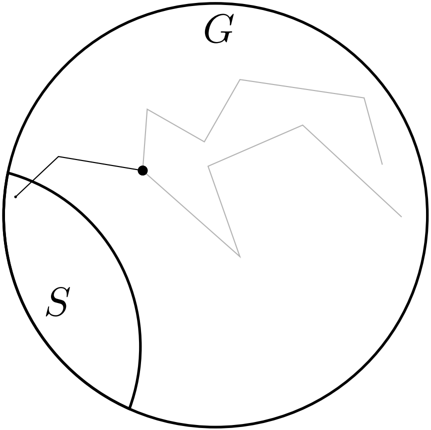
Sample a vertex, take many random walks from it and check whether any pair of walks ends at S with different parities. If so, reject, and accept otherwise.

Problem: Verify if the graph G is bipartite or ε-*far* from it.



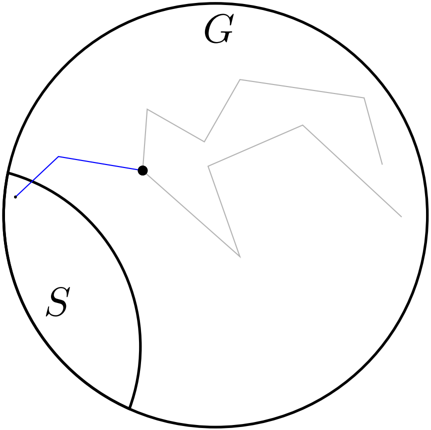
Sample a vertex, take many random walks from it and check whether any pair of walks ends at S with different parities. If so, reject, and accept otherwise.

Problem: Verify if the graph G is bipartite or ε-*far* from it.



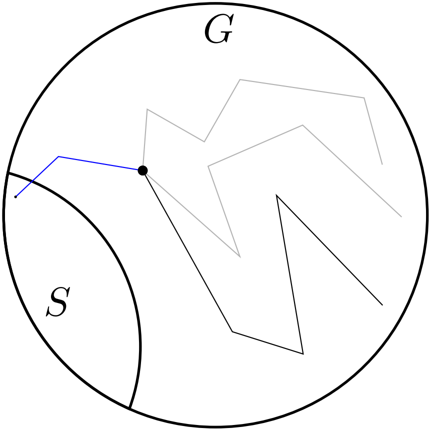
Sample a vertex, take many random walks from it and check whether any pair of walks ends at S with different parities. If so, reject, and accept otherwise.

Problem: Verify if the graph G is bipartite or ε-*far* from it.



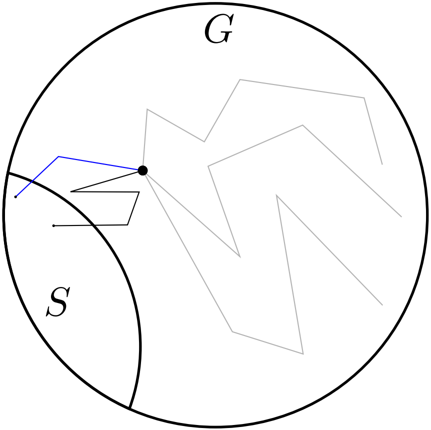
Sample a vertex, take many random walks from it and check whether any pair of walks ends at S with different parities. If so, reject, and accept otherwise.

Problem: Verify if the graph G is bipartite or ε-*far* from it.



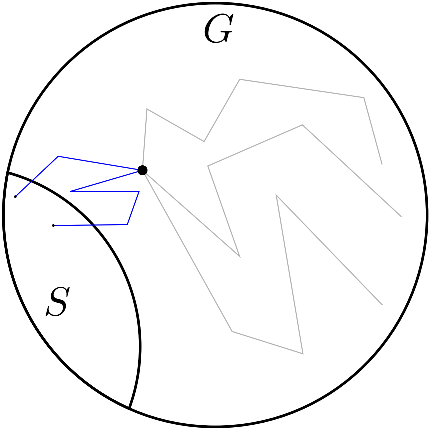
Sample a vertex, take many random walks from it and check whether any pair of walks ends at S with different parities. If so, reject, and accept otherwise.

Problem: Verify if the graph G is bipartite or ε-*far* from it.



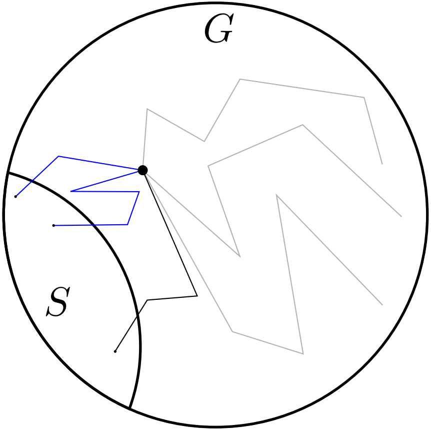
Sample a vertex, take many random walks from it and check whether any pair of walks ends at S with different parities. If so, reject, and accept otherwise.

Problem: Verify if the graph G is bipartite or ε-*far* from it.



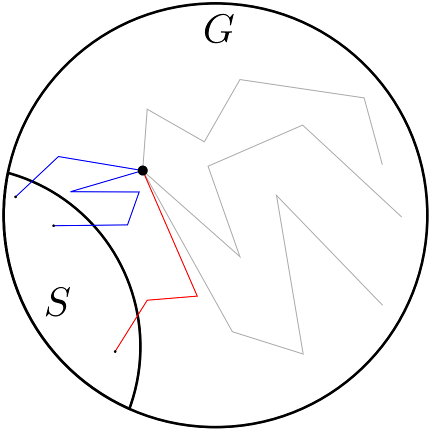
Sample a vertex, take many random walks from it and check whether any pair of walks ends at S with different parities. If so, reject, and accept otherwise.

Problem: Verify if the graph G is bipartite or ε-*far* from it.



Sample a vertex, take many random walks from it and check whether any pair of walks ends at S with different parities. If so, reject, and accept otherwise.

Problem: Verify if the graph G is bipartite or ε-*far* from it.



Sample a vertex, take many random walks from it and check whether any pair of walks ends at S with different parities. If so, reject, and accept otherwise.

Theorem

For any k, bipartiteness of rapidly-mixing n-vertex graphs (in the bounded-degree model) admits quantum proofs of proximity with $k\log(n)$-bit proofs and query complexity $\tilde{O}\left(\left(\frac{n}{k\varepsilon^2}\right)^{1/3}\right)$.

Theorem

For any k, bipartiteness of rapidly-mixing n-vertex graphs (in the bounded-degree model) admits quantum proofs of proximity with $k\log(n)$-bit proofs and query complexity $\tilde{O}\left(\left(\frac{n}{k\varepsilon^2}\right)^{1/3}\right)$.

Best quantum tester makes $O(n^{1/3})$ queries, so $k \approx \sqrt{n}$ beats it.
In best classical proof of proximity, $k \approx n^{2/3}$ for $O(n^{1/3})$ queries.
(Also improves dependence on ε.)

Theorem

For any k, bipartiteness of rapidly-mixing n-vertex graphs (in the bounded-degree model) admits quantum proofs of proximity with $k\log(n)$-bit proofs and query complexity $\tilde{O}\left(\left(\frac{n}{k\varepsilon^2}\right)^{1/3}\right)$.

Best quantum tester makes $O(n^{1/3})$ queries, so $k \approx \sqrt{n}$ beats it.
In best classical proof of proximity, $k \approx n^{2/3}$ for $O(n^{1/3})$ queries.
(Also improves dependence on ε.)

Techniques

Decomposable properties: known *classical* proofs of proximity
[Gur and Rothblum, 2018, Goldreich et al., 2018]

Bipartiteness: Quantum collision-finding algorithm
[Ambainis, 2007, Ambainis et al., 2011]

Techniques

Decomposable properties: known *classical* proofs of proximity
[Gur and Rothblum, 2018, Goldreich et al., 2018]

Bipartiteness: Quantum collision-finding algorithm
[Ambainis, 2007, Ambainis et al., 2011]

Techniques

Decomposable properties: known *classical* proofs of proximity [Gur and Rothblum, 2018, Goldreich et al., 2018]

Bipartiteness: Quantum collision-finding algorithm [Ambainis, 2007, Ambainis et al., 2011]

Introduction

Part I: Algorithms

Part II: Complexity separations

Complexity classes

	V	$V \leftarrow P$	$V \leftrightarrow P$
Classical	$\mathcal{PT}$	$\mathcal{MAP}$	$\mathcal{IPP}$
Quantum	$\mathcal{QPT}$	$\mathcal{QMAP}$	$\mathcal{QIPP}$

Also $\mathcal{QCMAP}$: classical proof, quantum input access.

Complexity classes

	V	$V \leftarrow P$	$V \leftrightarrow P$
Classical	$\mathcal{PT}$	$\mathcal{MAP}$	$\mathcal{IPP}$
Quantum	$\mathcal{QPT}$	$\mathcal{QMAP}$	$\mathcal{QIPP}$

Also $\mathcal{QCMAP}$: classical proof, quantum input access.

$\mathcal{C} := \mathcal{C}(\varepsilon, p, q)$ with $p, q = \mathrm{polylog}(n)$ and

ε a small enough constant.

Main result

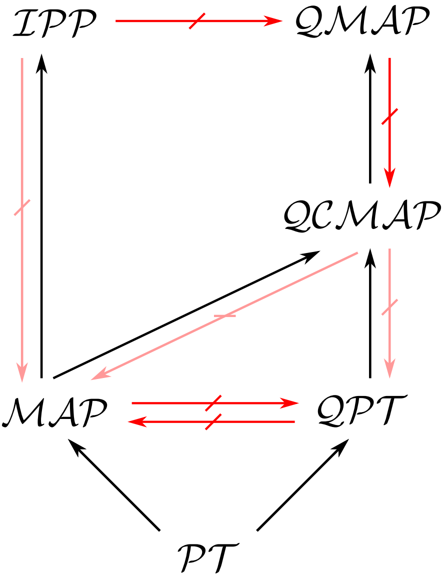
Theorem

The following separations hold:

- $\mathcal{QMAP} \not\subseteq \mathcal{MAP} \cup \mathcal{QPT}$, *i.e., quantum input access with a proof are more powerful in tandem than separately;*
- $\mathcal{QMAP} \not\subseteq \mathcal{QCMAP}$, *i.e., classical proofs are weaker than quantum even with a quantum verifier;*
- $\mathcal{IPP} \not\subseteq \mathcal{QMAP}$, *i.e., quantum proofs cannot substitute for interaction.*

Main result

$$
\begin{tikzcd}
\mathcal{IPP} \arrow[rr, red, "/" marking] & & \mathcal{QMAP} \arrow[d, red, shift left, "/" marking] \\
& & \mathcal{QCMAP} \arrow[u, shift left] \\
\mathcal{MAP} \arrow[uu] \arrow[urr] \arrow[rr, red, shift left, "/" marking] & & \mathcal{QPT} \arrow[u] \arrow[ll, red, shift left, "/" marking] \\
& \mathcal{PT} \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$


$\mathcal{IPP}$
$\mathcal{QMAP}$
$\mathcal{QCMAP}$
$\mathcal{MAP}$
$\mathcal{QPT}$
$\mathcal{PT}$

Techniques

$\mathcal{MAP} \not\subseteq \mathcal{QPT}$: disjointness + relaxed locally decodable code

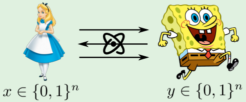
$\mathcal{QPT} \not\subseteq \mathcal{MAP}$: Forrelation [Aaronson and Ambainis, 2018]

$\mathcal{QMAP} \not\subseteq \mathcal{QCMAP}$: recasting $\mathcal{QMA} \not\subseteq \mathcal{QCMA}$ [Aaronson and Kuperberg, 2007]

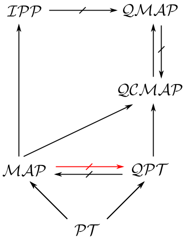
$\mathcal{IPP} \not\subseteq \mathcal{QMAP}$: permutation testing [Gur et al., 2018, Sherstov and Thaler, 2019]

Techniques

$\mathcal{MAP} \not\subseteq \mathcal{QPT}$: disjointness + relaxed locally decodable code



Given $C(x)$ and $C(y)$, $\exists i \in [n]$ such that $x_i = y_i = 1$?



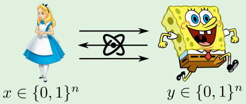
$\mathcal{QPT} \not\subseteq \mathcal{MAP}$: Forrelation [Aaronson and Ambainis, 2018]

$\mathcal{QMAP} \not\subseteq \mathcal{QCMAP}$: recasting $\mathcal{QMA} \not\subseteq \mathcal{QCMA}$ [Aaronson and Kuperberg, 2007]

$\mathcal{IPP} \not\subseteq \mathcal{QMAP}$: permutation testing [Gur et al., 2018, Sherstov and Thaler, 2019]

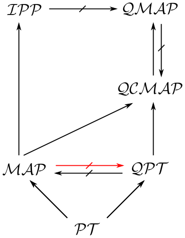
Techniques

$\mathcal{MAP} \not\subseteq \mathcal{QPT}$: disjointness + relaxed locally decodable code



Given $C(x)$ and $C(y)$, $\exists i \in [n]$ such that $x_i = y_i = 1$?

- $\Omega(\sqrt{n})$ without proof
- $O(1)$ with $\log n$ proof



$\mathcal{QPT} \not\subseteq \mathcal{MAP}$: Forrelation [Aaronson and Ambainis, 2018]

$\mathcal{QMAP} \not\subseteq \mathcal{QCMAP}$: recasting $\mathcal{QMA} \not\subseteq \mathcal{QCMA}$ [Aaronson and Kuperberg, 2007]

$\mathcal{IPP} \not\subseteq \mathcal{QMAP}$: permutation testing [Gur et al., 2018, Sherstov and Thaler, 2019]

Techniques

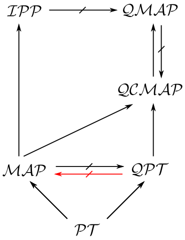
$\mathcal{MAP} \not\subseteq \mathcal{QPT}$: disjointness + relaxed locally decodable code

$\mathcal{QPT} \not\subseteq \mathcal{MAP}$: Forrelation [Aaronson and Ambainis, 2018]

Given $f, g : \{0,1\}^{\log n} \to \{0,1\}$, is $\langle f, \hat{g} \rangle$ small?

$\mathcal{QMAP} \not\subseteq \mathcal{QCMAP}$: recasting $\mathcal{QMA} \not\subseteq \mathcal{QCMA}$ [Aaronson and Kuperberg, 2007]

$\mathcal{IPP} \not\subseteq \mathcal{QMAP}$: permutation testing [Gur et al., 2018, Sherstov and Thaler, 2019]



Techniques

$\mathcal{MAP} \not\subseteq \mathcal{QPT}$: disjointness + relaxed locally decodable code

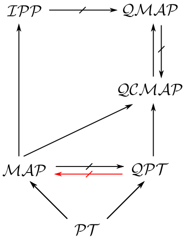
$\mathcal{QPT} \not\subseteq \mathcal{MAP}$: Forrelation [Aaronson and Ambainis, 2018]

Given $f, g : \{0,1\}^{\log n} \to \{0,1\}$, is $\langle f, \hat{g} \rangle$ small?

- $O(1)$ quantum, without proof
- $p \cdot q = \Omega(n^{1/4})$ classical, with proof

$\mathcal{QMAP} \not\subseteq \mathcal{QCMAP}$: recasting $\mathcal{QMA} \not\subseteq \mathcal{QCMA}$ [Aaronson and Kuperberg, 2007]

$\mathcal{IPP} \not\subseteq \mathcal{QMAP}$: permutation testing [Gur et al., 2018, Sherstov and Thaler, 2019]



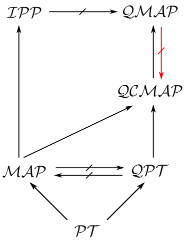
Techniques

$\mathcal{MAP} \not\subseteq \mathcal{QPT}$: disjointness + relaxed locally decodable code

$\mathcal{QPT} \not\subseteq \mathcal{MAP}$: Forrelation [Aaronson and Ambainis, 2018]

$\mathcal{QMAP} \not\subseteq \mathcal{QCMAP}$: recasting $\mathcal{QMA} \not\subseteq \mathcal{QCMA}$ [Aaronson and Kuperberg, 2007]

$\mathcal{IPP} \not\subseteq \mathcal{QMAP}$: permutation testing [Gur et al., 2018, Sherstov and Thaler, 2019]



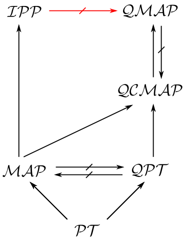
Techniques

$\mathcal{MAP} \not\subseteq \mathcal{QPT}$: disjointness + relaxed locally decodable code

$\mathcal{QPT} \not\subseteq \mathcal{MAP}$: Forrelation [Aaronson and Ambainis, 2018]

$\mathcal{QMAP} \not\subseteq \mathcal{QCMAP}$: recasting $\mathcal{QMA} \not\subseteq \mathcal{QCMA}$ [Aaronson and Kuperberg, 2007]

$\mathcal{IPP} \not\subseteq \mathcal{QMAP}$: permutation testing [Gur et al., 2018, Sherstov and Thaler, 2019]



Given $f : [n] \to [n]$, is f a permutation?

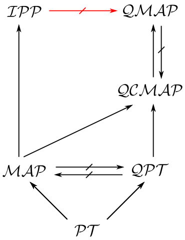
Techniques

$\mathcal{MAP} \not\subseteq \mathcal{QPT}$: disjointness + relaxed locally decodable code

$\mathcal{QPT} \not\subseteq \mathcal{MAP}$: Forrelation
[Aaronson and Ambainis, 2018]

$\mathcal{QMAP} \not\subseteq \mathcal{QCMAP}$: recasting $\mathcal{QMA} \not\subseteq \mathcal{QCMA}$
[Aaronson and Kuperberg, 2007]

$\mathcal{IPP} \not\subseteq \mathcal{QMAP}$: permutation testing
[Gur et al., 2018, Sherstov and Thaler, 2019]



Given $f : [n] \to [n]$, is f a permutation?

- $O(1)$ with (classical) interaction
- $p \cdot q = \Omega(n^{1/3})$ quantum, without interaction

Open problems

- What about $\mathcal{QIPP}$?

Open problems

- What about $\mathcal{QIPP}$?
- Can QIPPs test $\mathcal{NC}$ languages with $o(\sqrt{n})$ proof and query complexities? [Rothblum and Rothblum, 2020]

Open problems

- What about $\mathcal{QIPP}$?
- Can QIPPs test $\mathcal{NC}$ languages with $o(\sqrt{n})$ proof and query complexities? [Rothblum and Rothblum, 2020]
- Extend bipartiteness to non-rapidly-mixing graphs.

Open problems

- What about $\mathcal{QIPP}$?
- Can QIPPs test $\mathcal{NC}$ languages with $o(\sqrt{n})$ proof and query complexities? [Rothblum and Rothblum, 2020]
- Extend bipartiteness to non-rapidly-mixing graphs.

Thank you!

References I

Aaronson, S. and Ambainis, A. (2018).
Forrelation: A problem that optimally separates quantum from classical computing.
SIAM Journal on Computing, 47(3):982–1038.

Aaronson, S. and Kuperberg, G. (2007).
Quantum versus classical proofs and advice.
In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 115–128. IEEE Computer Society.

Ambainis, A. (2007).
Quantum walk algorithm for element distinctness.
SIAM Journal on Computing, 37(1):210–239.

Ambainis, A., Childs, A. M., and Liu, Y.-K. (2011).
Quantum property testing for bounded-degree graphs.
In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 365–376, Berlin, Heidelberg. Springer Berlin Heidelberg.

Brassard, G., Hoyer, P., Mosca, M., and Tapp, A. (2002).
Quantum amplitude amplification and estimation.
Contemporary Mathematics, 305:53–74.

Goldreich, O., Gur, T., and Rothblum, R. D. (2018).
Proofs of proximity for context-free languages and read-once branching programs.
Information and Computation, 261:175–201.

Gur, T., Liu, Y. P., and Rothblum, R. D. (2018).
An exponential separation between MA and AM proofs of proximity.
In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

References II

Gur, T. and Rothblum, R. D. (2018).
Non-interactive proofs of proximity.
computational complexity, 27(1):99–207.

Rothblum, G. N. and Rothblum, R. D. (2020).
Batch verification and proofs of proximity with polylog overhead.
In *Theory of Cryptography Conference*, pages 108–138. Springer.

Sherstov, A. A. and Thaler, J. (2019).
Vanishing-error approximate degree and QMA complexity.
arXiv:1909.07498.

Images:

Server Icon by Rank Sol on Iconscout
Mobile by Momento Design from the Noun Project
Laptop Icon by Jemis Mali from Iconscout
Smartwatch by juan manjarrez from the Noun Project
database by mardjoe from the Noun Project
atom by Fengquan Li from the Noun Project

https://disney.fandom.com/wiki/Alice/Gallery?file=Alice_Render.png

https://loathsomecharacters.miraheze.org/wiki/File:SpongeBob_SquarePants.png