USC University of Southern California

USC CQIST Center for Quantum Information Science and Technology

Quantum approach to information retrieval: Adiabatic quantum PageRank algorithm

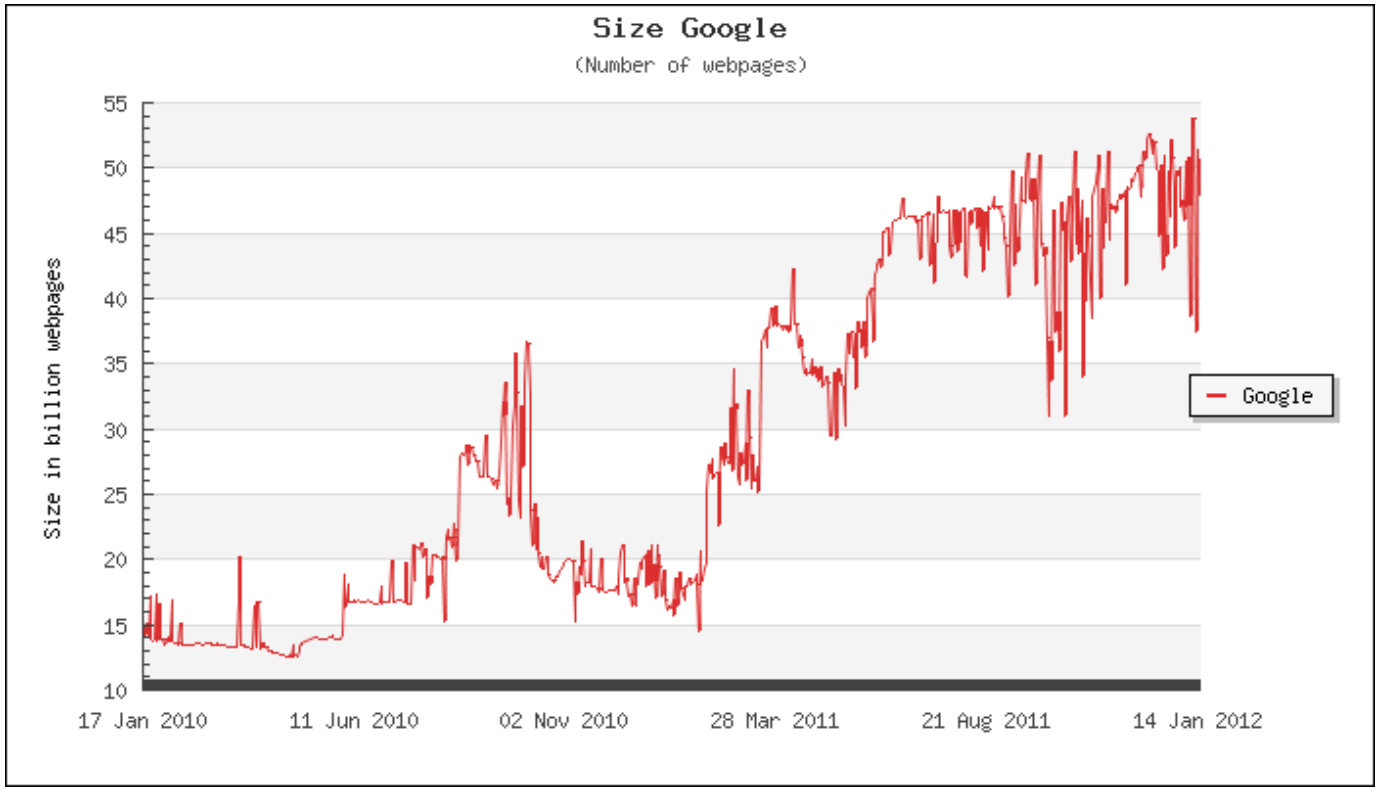
arXiv:1109.6546

with Silvano Garnerone and Paolo Zanardi

First NASA Quantum Future Technologies Conference

$: LOCKHEED MARTIN Google research NASA

The WWW is a big place

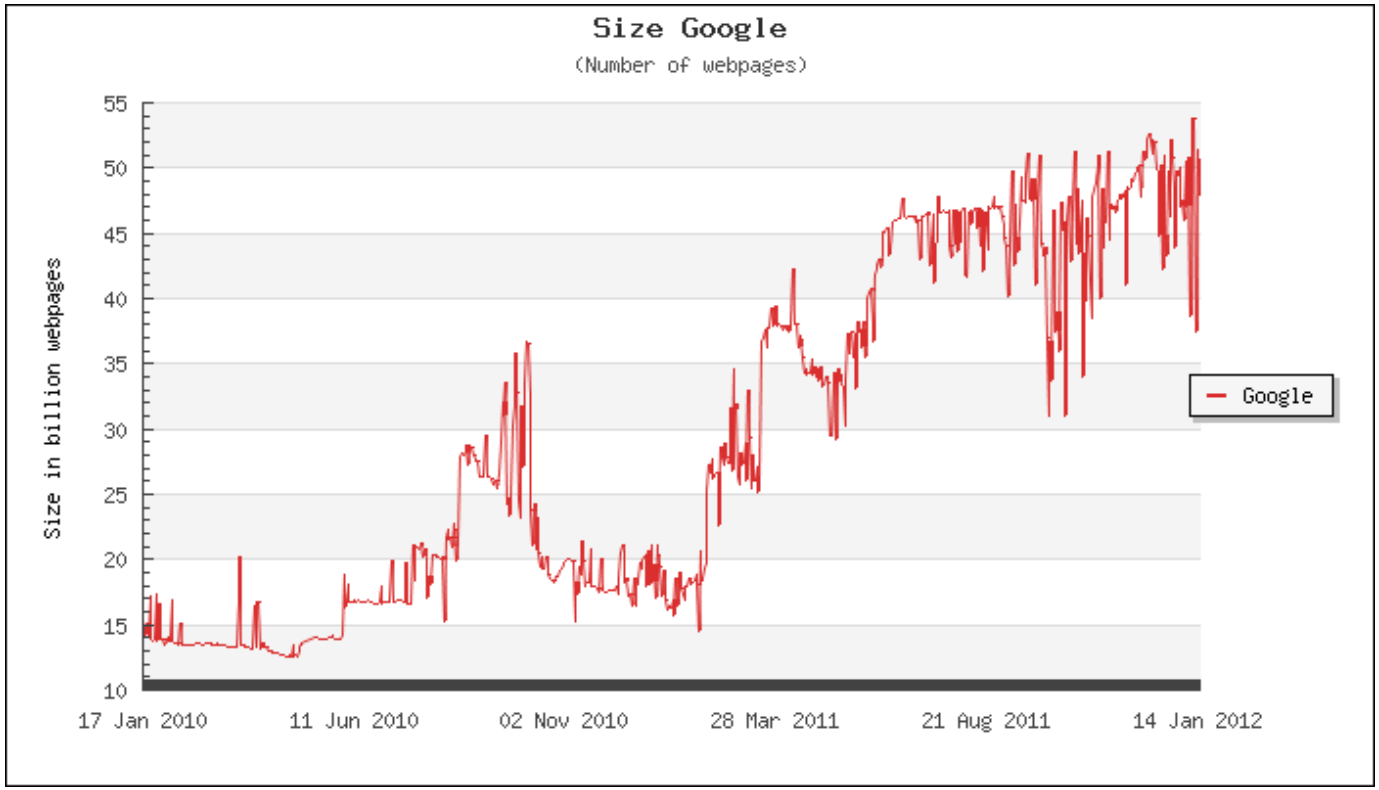


www.worldwidewebsize.com

The WWW is a big place

...

and is hard to search



www.worldwidewebsize.com

"The certitude that some shelf in some hexagon held precious books and that these precious books were inaccessible seemed almost intolerable"
J.L. Borges in The library of Babel

Google to the rescue: Brin & Page, 1998

Computer Networks and ISDN Systems 30 (1998) 107–117

COMPUTER NETWORKS and ISDN SYSTEMS

The anatomy of a large-scale hypertextual Web search engine [1]

Sergey Brin [2], Lawrence Page *,[2]

Computer Science Department, Stanford University, Stanford, CA 94305, USA

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/

Google scholar: >8500 citations

What does Google do?

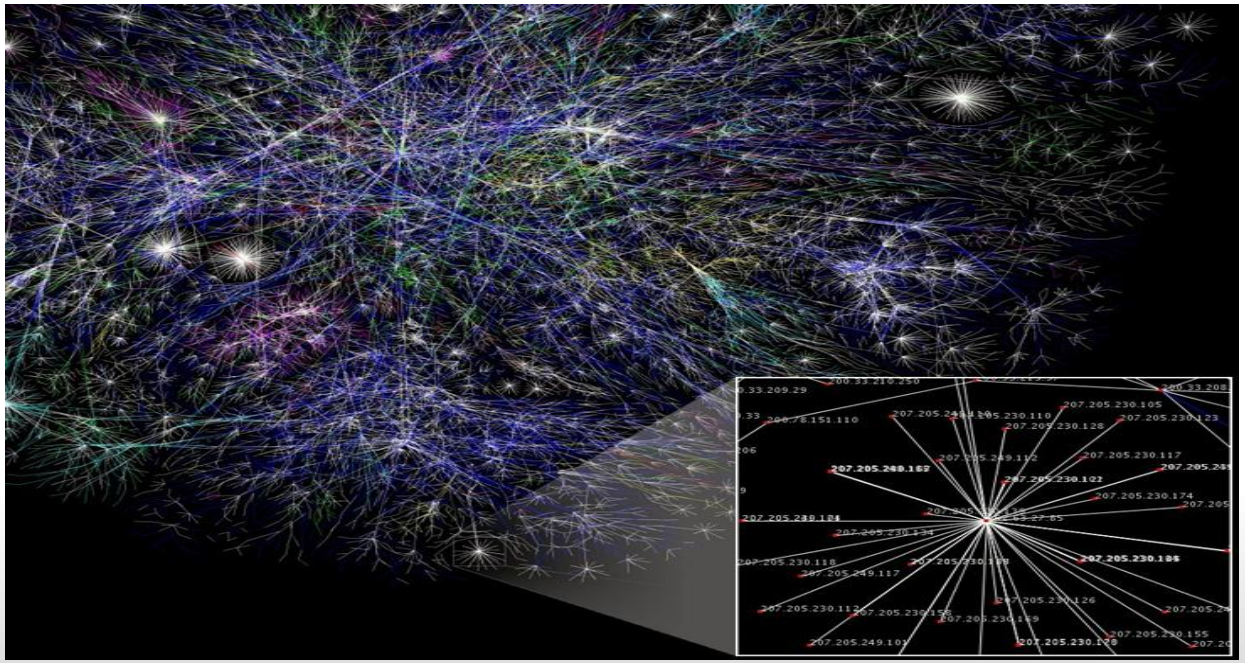
Google calculates an eigenvector

$$G\vec{\pi} = \vec{\pi}$$

Google calculates an eigenvector

$$G\vec{\pi} = \vec{\pi}$$

$\vec{\pi}$ is the
stationary state
of a surfer hopping
randomly on the
web graph



the **PageRank** vector

$\vec{\pi}_i$ = rank of *i*'th page:
the relative time spent there by the random surfer

Google calculates an eigenvector

$$G\vec{\pi} = \vec{\pi}$$

- *G* is a big matrix: dimension = number of webpages *n*. Updated about once a month

Google calculates an eigenvector

$$G\vec{\pi} = \vec{\pi}$$

- *G* is a big matrix: dimension = number of webpages *n*.
  Updated about once a month

- *G* is computed from the directed adjacency matrix of the webgraph

$$G = \alpha S$$

hyperlink matrix of webgraph,
normalized columns; reflects the
directed connectivity structure of the webgraph

Google calculates an eigenvector

$$G\vec{\pi} = \vec{\pi}$$

- *G* is a big matrix: dimension = number of webpages *n*.
  Updated about once a month

- *G* is computed from the directed adjacency matrix of the webgraph
  + random hopping to avoid traps from nodes with no outgoing links:

$$G = \alpha S + \frac{1}{n}(1-\alpha)E$$

hyperlink matrix of webgraph, normalized columns; reflects the *directed* connectivity structure of the webgraph

matrix of all 1's

"teleport" parameter: 0.85

Google calculates an eigenvector

$$G\vec{\pi} = \vec{\pi}$$

- *G* is a big matrix: dimension = number of webpages *n*. Updated about once a month

- *G* is computed from the directed adjacency matrix of the webgraph + random hopping to avoid traps from nodes with no outgoing links:

$$G = \alpha S + \frac{1}{n}(1-\alpha)E$$

hyperlink matrix of webgraph, normalized columns; reflects the *directed* connectivity structure of the webgraph

matrix of all 1's

"teleport" parameter: 0.85

- *G* is a "primitive" matrix ($G_{ij} \geq 0, \exists k > 0$ s.t. $(G_{ij})^k > 0$, $\forall i, j$):

Perron-Frobenius theorem ➔ $\vec{\pi}$ is unique, and a probability vector:
$\vec{\pi}$ encodes the relative ranking of the nodes of the webgraph

This talk

Can (adiabatic) quantum computation help to compute $\vec{\pi}$?

This talk

Can (adiabatic) quantum computation help to compute $\vec{\pi}$?

PageRank can be:

prepared with exp speedup

read out with poly speedup
for top-ranked $\log(n)$ pages

Why?
gap of certain Hamiltonian
having PageRank as ground
state scales as

$$1/\mathrm{poly}(\log(n))$$

numerical evidence:
arXiv:1109.6546

Classical PageRank computation

The PageRank is the **principal eigenvector** of *G*; unique eigenvector with maximal eigenvalue 1

$$G\vec{\pi} = \vec{\pi}$$

How do you get the PageRank, classically?

Classical PageRank computation

$$G = \alpha S + \frac{1}{n}(1-\alpha)E$$

Power method: *G* is a Markov (stochastic) matrix, so

$$\pi = \lim_{k \to \infty} G^k \pi_0$$

$$\pi^{(k+1)} = G\pi^{(k)}$$

Guaranteed to converge for any initial probability vector.

Scaling?

Classical PageRank computation

$$G = \alpha S + \frac{1}{n}(1-\alpha)E$$

Power method: *G* is a Markov (stochastic) matrix, so

$$\pi = \lim_{k\to\infty} G^k \pi_0$$

$$\pi^{(k+1)} = G\pi^{(k)}$$

Guaranteed to converge for any initial probability vector.

Scaling?

$$\text{time} \sim s n \frac{\log(\epsilon)}{|\log(\alpha)|}$$

ϵ = desired accuracy
s = *sparsity* of the adjacency (or hyperlink) matrix. Typically *s*~10

Classical PageRank computation

Markov Chain Monte Carlo:

Uses direct simulation of rapidly mixing random walks to estimate the PageRank at each node.

$$\text{time} \sim O[n \log(n)]$$

[Modulus of the second eigenvalue of *G* is upper-bounded by α
➔ *G* is a gapped stochastic matrix
➔ walk converges in time $O[\log(n)]$ per node]

Classical computation is already efficient; why do we need quantum?

power method:

time $\sim s n \frac{\log(\epsilon)}{\log(\alpha)}$

Markov chain Monte Carlo:

$O[n \log(n)]$

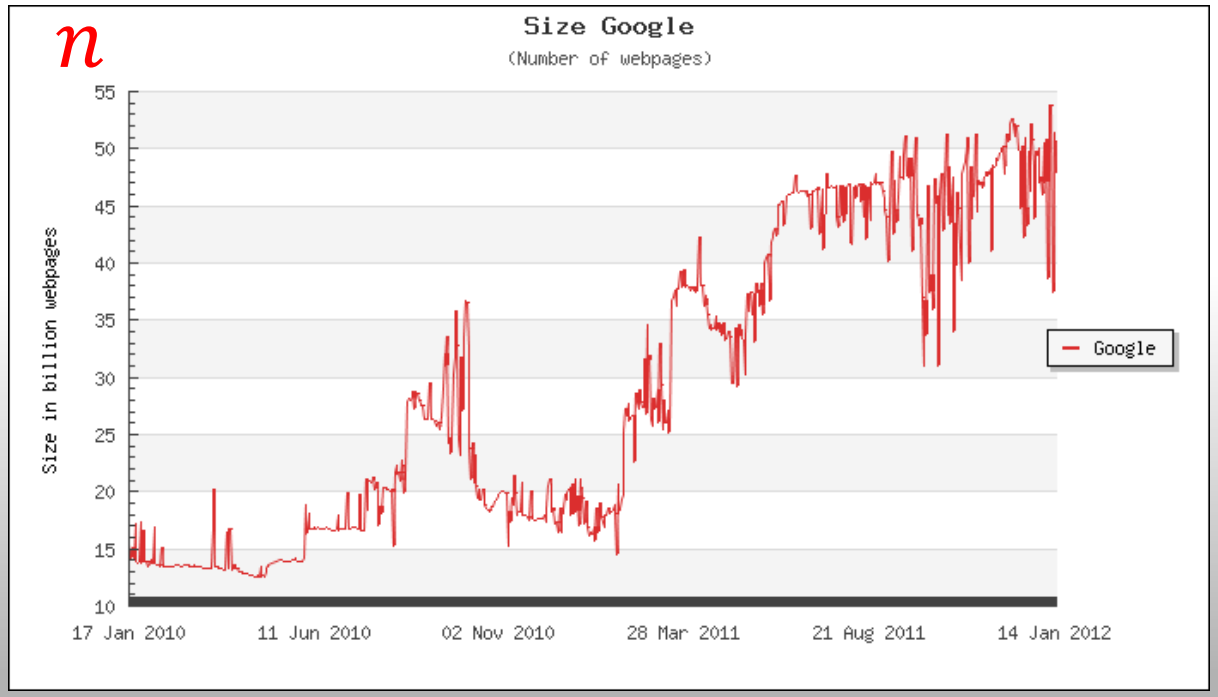
Classical computation is already efficient; why do we need quantum?

power method:

time $\sim sn\,\frac{\log(\epsilon)}{\log(\alpha)}$

Markov chain Monte Carlo:

$O[n\log(n)]$



updating PageRank already takes weeks; will only get worse.

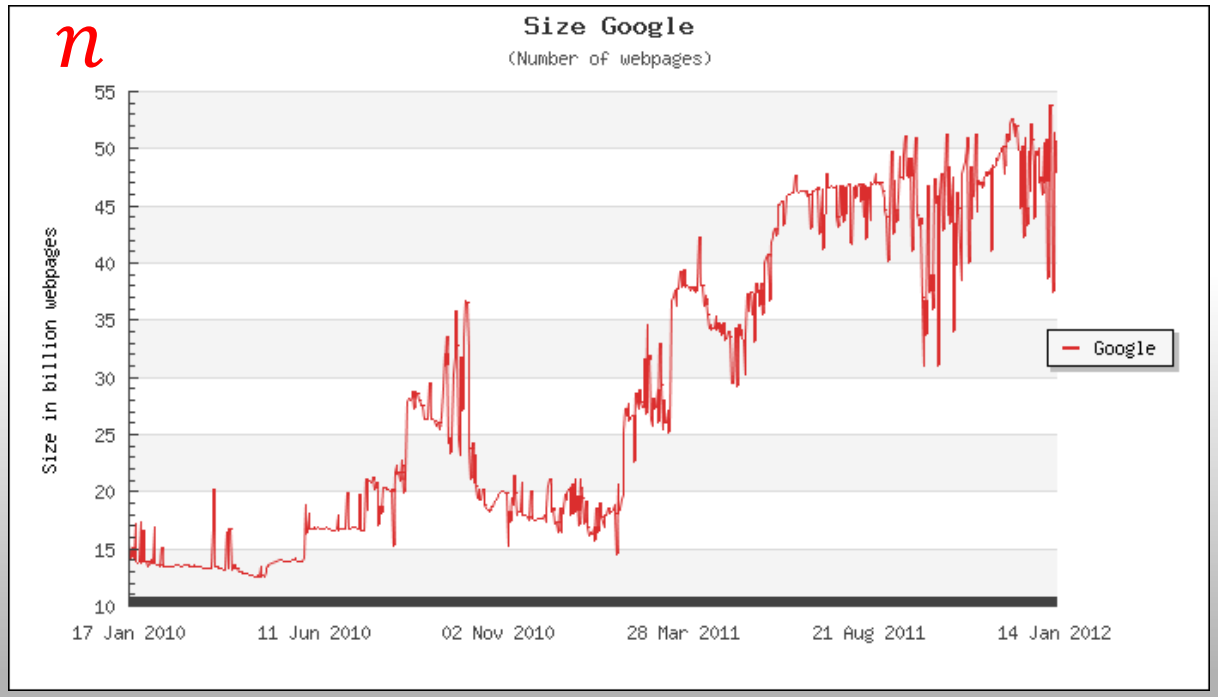
Classical computation is already efficient; why do we need quantum?

power method:

time $\sim sn\,\frac{\log(\epsilon)}{\log(\alpha)}$

Markov chain Monte Carlo:

$O[n\log(n)]$



updating PageRank already takes weeks; will only get worse.

With q-adiabatic algo can prepare PageRank in time

$O[\mathrm{poly}(\log(n))]$

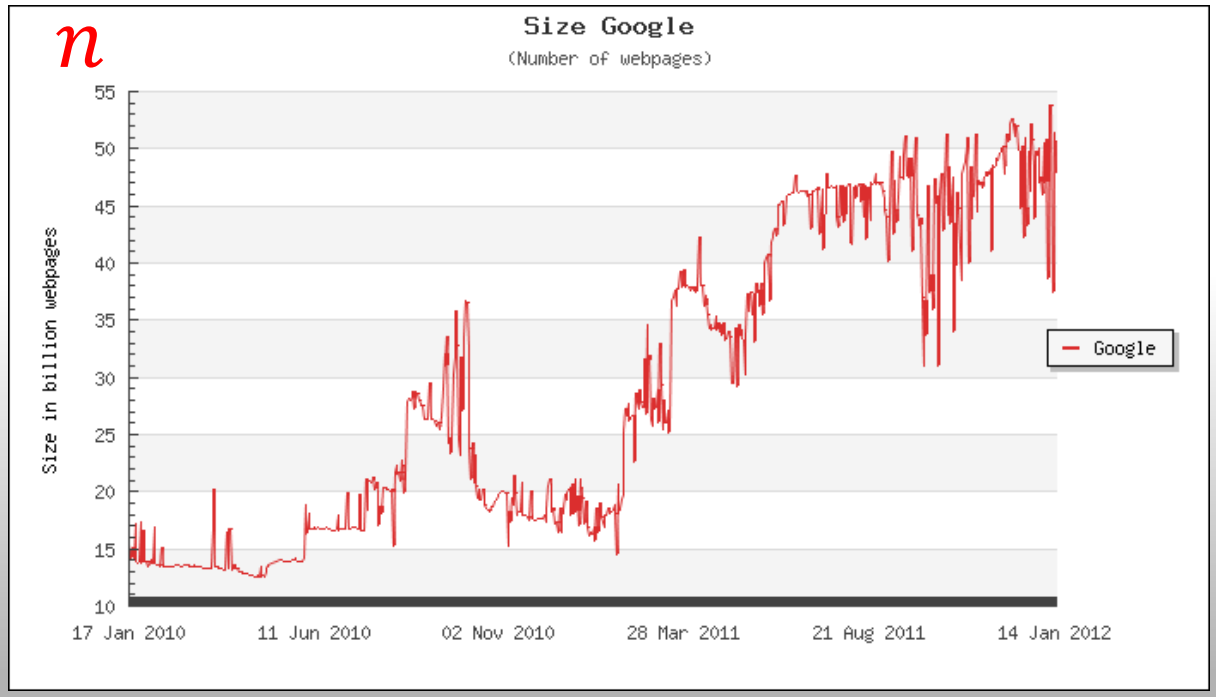
Classical computation is already efficient; why do we need quantum?

power method:

time $\sim s n \frac{\log(\epsilon)}{\log(\alpha)}$

Markov chain Monte Carlo:

$O[n \log(n)]$



updating PageRank already takes weeks; will only get worse.

With q-adiabatic algo can prepare PageRank in time

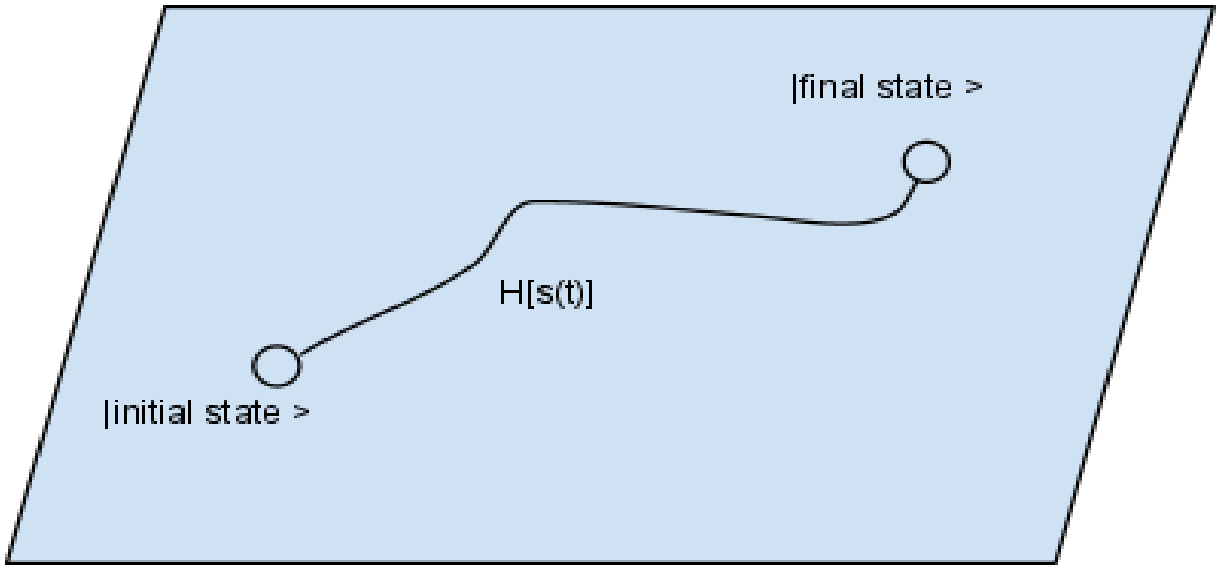
$O[\text{poly}(\log(n))]$

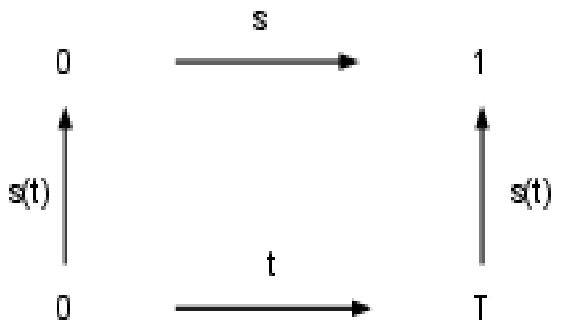
Application: run successive PageRanks and compare in time $O(1)$; use to decide whether to run classical update

Quantum approach

Adiabatic quantum computation of the PageRank vector

Adiabatic quantum computation





$$h[s(t)] = [1 - s(t)]h_0 + s(t)h_P$$

initial Hamiltonian problem Hamiltonian

The q-algorithm for PageRank

$t = 0$: prepare ground state of the initial Hamiltonian

$$|\psi_0\rangle = \sum_{j=1}^{n} \frac{1}{\sqrt{n}} |j\rangle$$

Uniform superposition over the complete graph of *n* nodes.

This requires $\log(n)$ qubits so assume n is power of 2.

The q-algorithm for PageRank

$t = T$: evolve to ground state of the final Hamiltonian

The **problem Hamiltonian** is $h_p = (I - G)^\dagger (I - G)$

The q-algorithm for PageRank

$t = T$: evolve to ground state of the final Hamiltonian

The **problem Hamiltonian** is $h_p = (I - G)^\dagger (I - G)$

- **Positive semidefinite**, with **0** the **unique min eigenvalue**
- If $G\vec{\pi} = \vec{\pi}$ then $|\pi\rangle = \vec{\pi}/\|\vec{\pi}\|_2$ is corresponding ground state of h_p

Note for experts: since *G* is not reversible (doesn't satisfy detailed balance) we cannot apply the standard "Szegedy trick" of quantum random walks (mapping to a discriminant matrix)

The q-algorithm for PageRank

$t = T$: evolve to ground state of the final Hamiltonian

The **problem Hamiltonian** is $h_p = (I - G)^\dagger (I - G)$

- **Positive semidefinite**, with **0** the **unique min eigenvalue**
- If $G\vec{\pi} = \vec{\pi}$ then $|\pi\rangle = \vec{\pi}/\|\vec{\pi}\|_2$, $|\pi\rangle_i \neq \sqrt{\pi_i}$ is corresponding ground state of h_p

Yet the amplitudes of the final ground state respect **the same ranking order** as the PageRank,
and **amplify** higher ranked pages

Efficiency of the q-algorithm

According to the adiabatic theorem, to get

$$\text{adiabatic error } \varepsilon := \sqrt{1 - f^2}, \text{ fidelity } f := |\langle \psi(T) | \pi \rangle|$$

actual final state

desired ground state

for $h[s(t)] = [1 - s(t)]h_0 + s(t)h_P$

Efficiency of the q-algorithm

According to the adiabatic theorem, to get

$$\text{adiabatic error } \varepsilon := \sqrt{1 - f^2}, \text{ fidelity } f := |\langle \psi(T) | \pi \rangle|$$

actual final state — desired ground state

for $h[s(t)] = [1 - s(t)]h_0 + s(t)h_P$

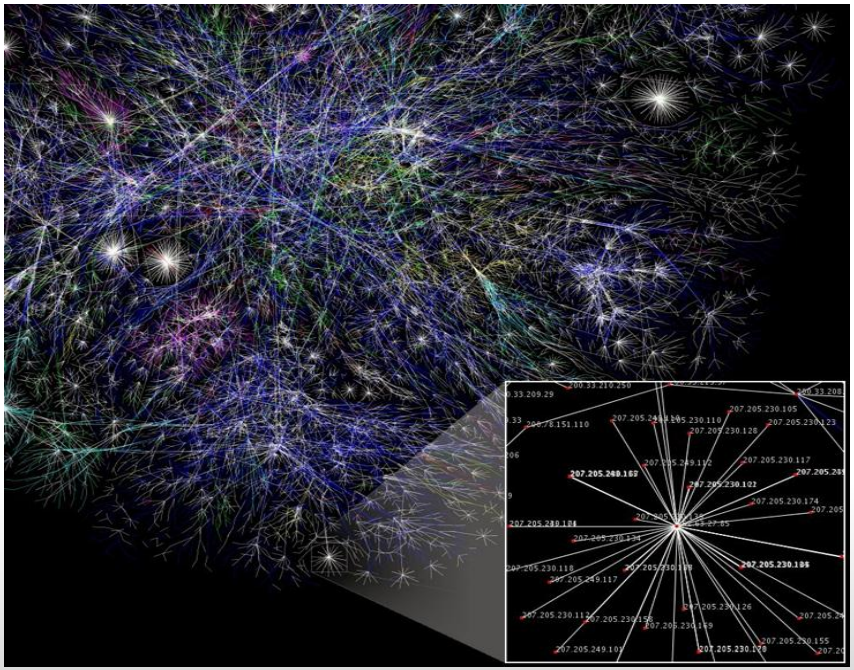
need $$T \sim \text{poly}\left[\frac{1}{\min_{s\in[0,1]}(\text{gap})}, \max_{s\in[0,1]}\left\|\frac{dh}{ds}\right\|, \frac{1}{\varepsilon}\right]$$

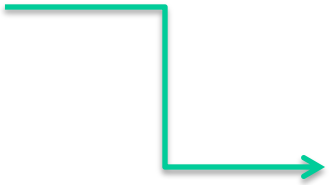
- not necessarily $\min(\text{gap})^{-2}$: can have -1 (best case) or -3 (worst case)
- scaling of numerator can be important; needs to be checked

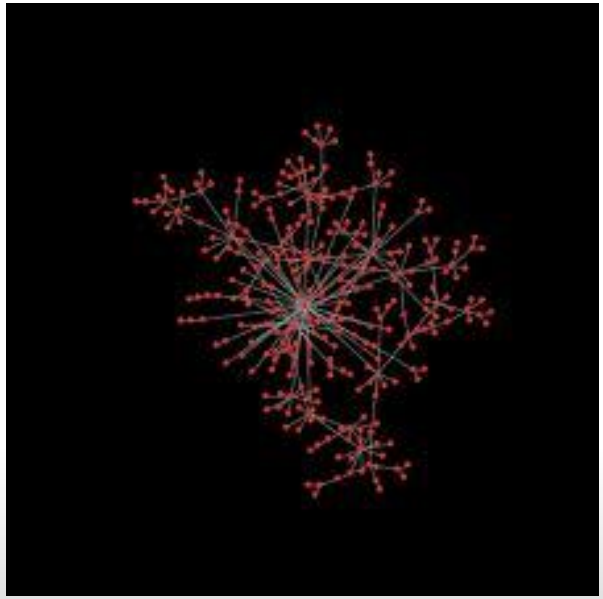
DAL, A. Rezakhani, and A. Hamma, J. Math. Phys. (2009)

Testing the q-algo on webgraph models

We tested the algorithm on **random** webgraph models,
sparse, small-world, scale-free (power law degree distribution):



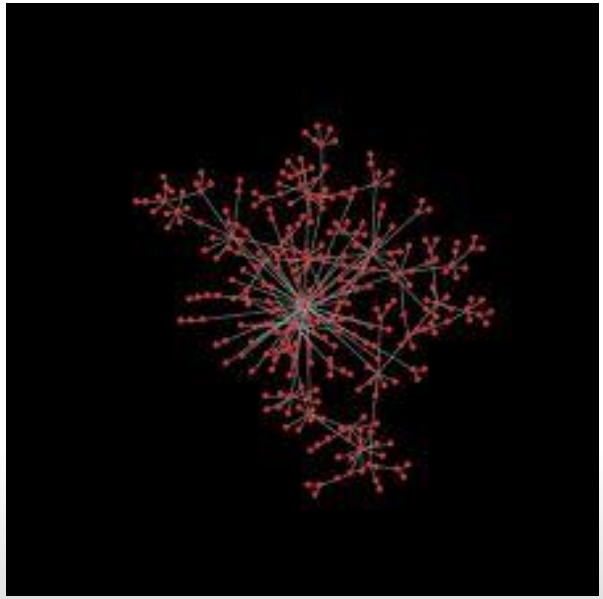




Testing the q-algo on webgraph models

We tested the algorithm on **random** webgraph models,
sparse, small-world, scale-free (power law degree distribution):

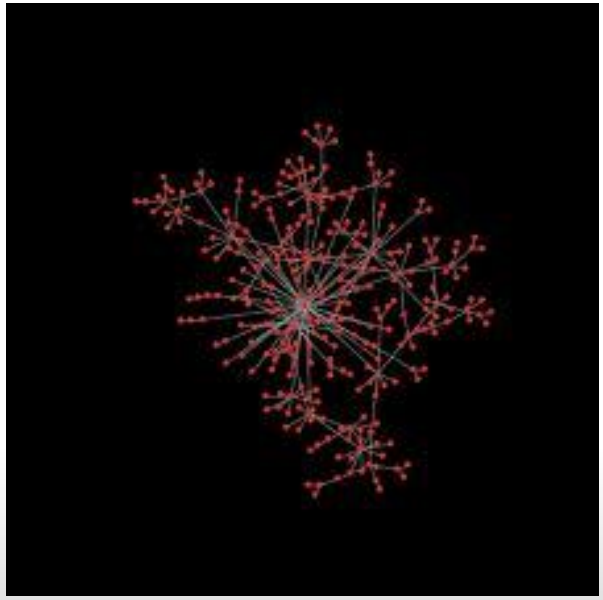
- **preferential attachment model**
  links are added at random with a bias for high-degree nodes
  drawback: requires global knowledge of graph
  Degree distribution: $N(d) \propto d^{-3}$



Testing the q-algo on webgraph models

We tested the algorithm on **random** webgraph models,
sparse, small-world, scale-free (power law degree distribution):

- **preferential attachment model**
  links are added at random with a bias for high-degree nodes
  drawback: requires global knowledge of graph
  Degree distribution: $N(d) \propto d^{-3}$

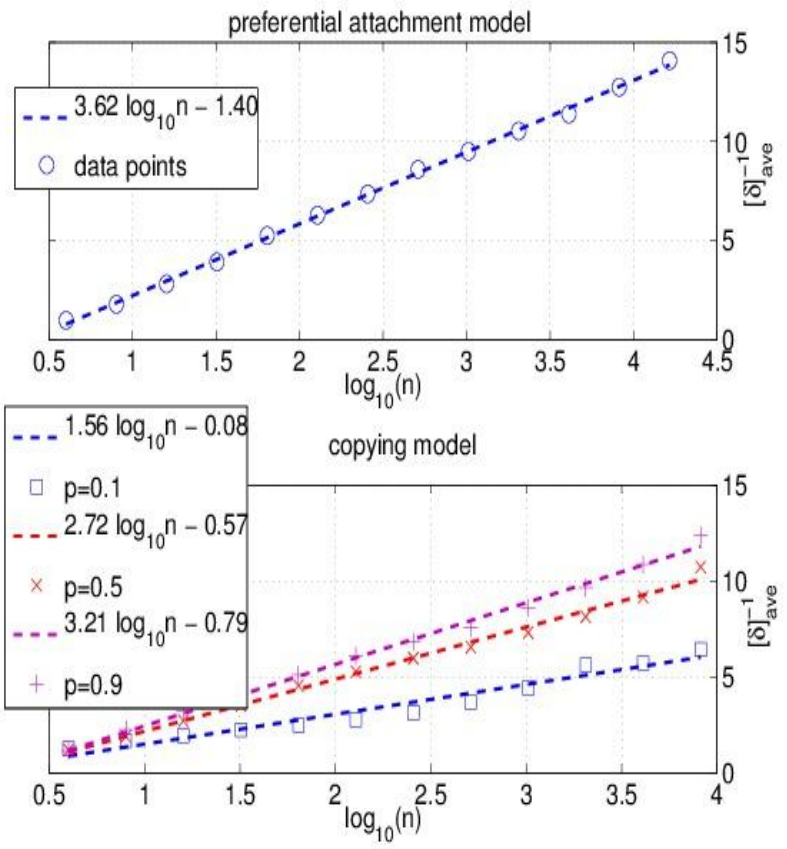


- **copy-model**
  - start from a small fixed initial graph of constant out-degree
  - each time step:
    - choose pre-existing "copying vertex" uniformly at random
    - Probability $1-p$: For each neighbor of the copying vertex, add a link from a new added vertex to that neighbor
    - Probability p: add link from newly added vertex to uniformly random chosen one

  requires only local knowledge of graph; has tuning parameter p

  Degree distribution: $N(d) \propto d^{(2-p)/(1-p)}$

Efficiency of the q-algorithm



numerical diagonalization

ave. min gap scaling:

$$\delta \sim 1/\mathrm{poly}(\log n)$$

[Note: we computed same for generic sparse random matrices and found gap $\sim 1/\mathrm{poly}(n)$ instead]

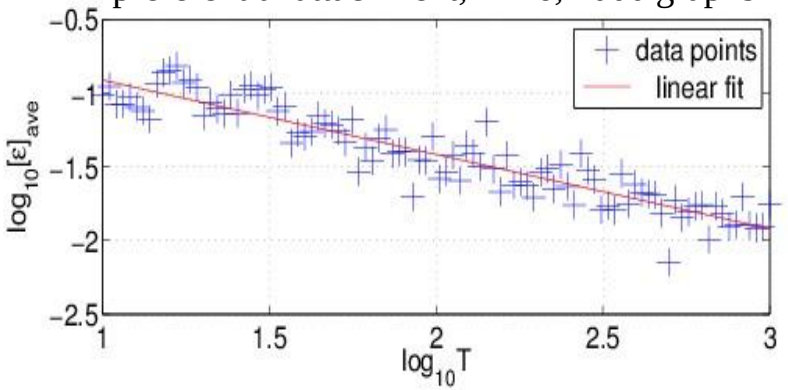
$$T \sim \mathrm{poly}\left[\frac{1}{\min(\mathrm{gap})}, \max_{s\in[0,1]}\left\|\frac{dh}{ds}\right\|, \frac{1}{\varepsilon}\right]$$

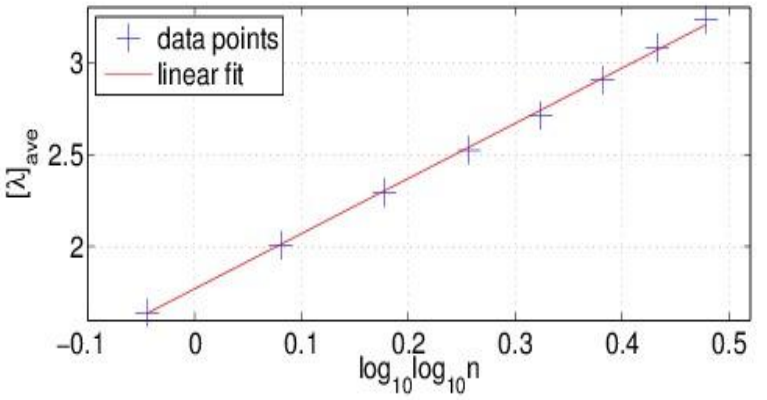
Efficiency of the q-algorithm

run Schrodinger equation with different T:

$$T \sim \varepsilon^{-2}$$

preferential attachment, n=16, 1000 graphs





$$\left\| \frac{dh}{ds} \right\| = \lambda \sim \text{poly}(\log \log n)$$

$$T \sim \text{poly}\left[\frac{1}{\min(\text{gap})}, \max_{s \in [0,1]} \left\| \frac{dh}{ds} \right\|, \frac{1}{\varepsilon} \right]$$

Efficiency of the q-algorithm

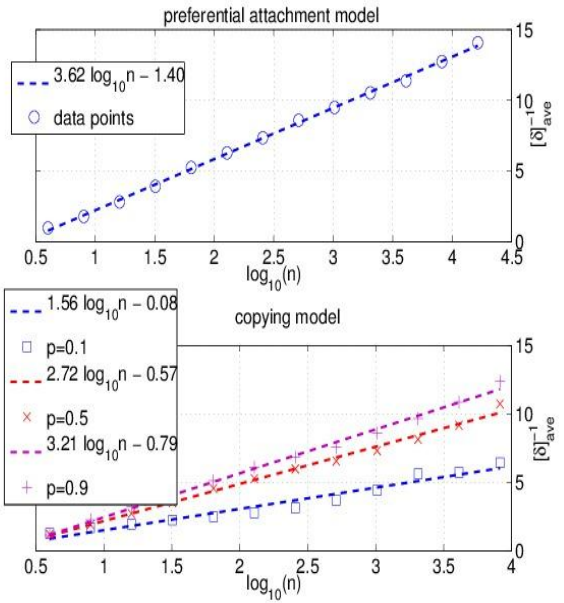
$$\delta \sim 1/\mathrm{poly}(\log n)$$

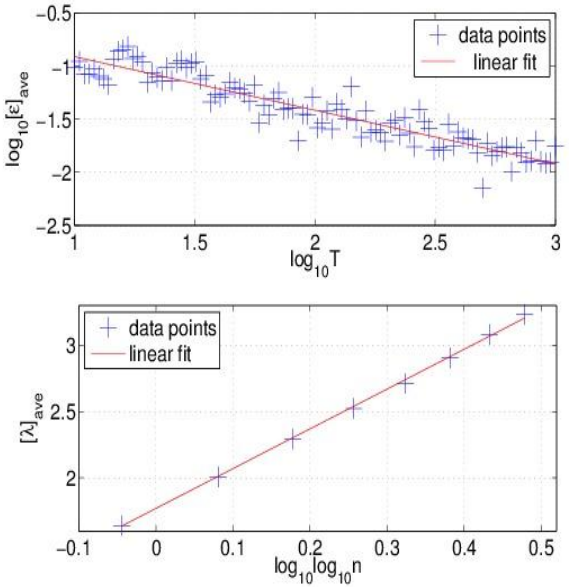
&

$$T \sim \varepsilon^{-2}$$

&

$$\left\| \frac{dh}{ds} \right\| = \mathrm{poly}(\log\log n)$$





$$T \sim \mathrm{poly}\left[\frac{1}{\min(\mathrm{gap})}, \max_{s\in[0,1]} \left\| \frac{dh}{ds} \right\|, \frac{1}{\varepsilon} \right]$$

small integer >0

$$T \sim \varepsilon^{-2} (\log\log n)^{b-1} (\log n)^{b}$$

checked and confirmed using solution of the full Schrodinger equation, for $b = 3$:
actual error always less than ε

So is this really an efficient q-algorithm?

- **Problem 1**: The Google matrix *G* is a full matrix...
  ➔ $h[s(t)]$ requires many-body interactions...

So is this really an efficient q-algorithm?

- **Problem 1**: The Google matrix *G* is a full matrix...
  ➔ $h[s(t)]$ requires many-body interactions...

- Can be reduced to 1&2 qubit interactions by using one qubit per node:
  go from $\log(n)$ qubits to n qubits (unary representation), i.e., map to n-dim. "single particle excitation" subspace of 2^n-dim Hilbert space:

$$H(s) = \sum_{i=1}^{n} h(s)_{ii}\sigma_i^+\sigma_i^- + \sum_{i<j}^{n} h(s)_{ij}\left(\sigma_i^+\sigma_j^- + \sigma_j^+\sigma_i^-\right)$$

matrix elements of $h[s(t)] = [1 - s(t)]h_0 + s(t)h_P$

probability of finding excitation at site *i* gives PageRank of page *i*

$H(s)$ (in 1-excitation subspace) and $h(s)$ have same spectrum ➔ same T scaling

Measuring the PageRank

- **Problem 2**: Once the ground-state has been prepared one needs to measure the site occupation probabilities $(\pi_i)^2 / \|\vec{\pi}\|^2$

  To recover the complete length-*n PageRank* vector takes at least *n* measurements (Chernoff bound*)

  ➔ back to the classical performance 

- Same problem as in the quantum algorithm for solving linear equations [Harrow, Hassidim, Lloyd, PRL (2009)];

  actually our algorithm is an instance of solving linear equations, but assumes a lot more structure

*To estimate *i*th prob. amplitude with additive error e_i need number of measurements ~ 1/poly(e_i)

Measuring the PageRank

- **Problem 2**: Once the ground-state has been prepared one needs to measure the site occupation probabilities $(\pi_i)^2 / \|\vec{\pi}\|^2$

  To recover the complete length-*n PageRank* vector takes at least *n* measurements (Chernoff bound*)

  ➔ back to the classical performance 

- However: one is typically interested only in the *top ranked pages*
- For these pages we nevertheless obtain (again using the Chernoff bound) a **polynomial speed-up** for estimating the ranks of the **top $\log(n)$ pages**
- This is because of the **amplification** of top PageRank entries and **power-law distribution** of the PageRank entries

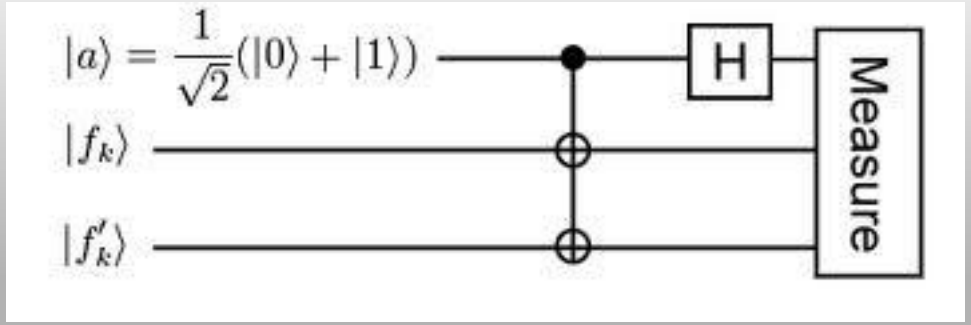
Summary of results and applications

- Can map adiabatic PageRank algo to Hamiltonians with 1&2 body interactions, with one qubit per node
- Polynomial speed-up for top-$\log(n)$ set of nodes
- Exponential speedup in preparation of PageRank

Summary of results and applications

- Can map adiabatic PageRank algo to Hamiltonians with 1&2 body interactions, with one qubit per node
- Polynomial speed-up for top-$\log(n)$ set of nodes
- Exponential speedup in preparation of PageRank allows for an **efficient decision procedure about updating** of the classical PageRank:
  - Prepare pre-perturbation PageRank state $|\pi\rangle$: $T \sim O[\mathrm{poly}(\log(n))]$
  - Prepare post-perturbation PageRank state $|\pi'\rangle$: $T' \sim O[\mathrm{poly}(\log(n'))]$
  - Compute $|\langle\pi|\pi'\rangle|$ using the SWAP test: $\sim O(1)$

  ➔ Decide whether update needed



$\rightarrow |\langle f_k | f'_k \rangle|$

Conclusions

- Information retrieval provides new set of problems for Quantum Computation
- Given the existence of efficient classical algorithms it is non-trivial that QC can provide some form of speedup
- The humongous size of the WWW is an important motivation to look for such a speedup
- Showed tasks for which adiabatic quantum PageRank provides a speedup with respect to classical algorithms

Conclusions

- Information retrieval provides new set of problems for Quantum Computation
- Given the existence of efficient classical algorithms it is non-trivial that QC can provide some form of speedup
- The humongous size of the WWW is an important motivation to look for such a speedup
- Showed tasks for which adiabatic quantum PageRank provides a speedup with respect to classical algorithms

- Why does it work? **Sparsity** alone seems insufficient.
- Other key features of the webgraph are
  - **small-world** (each node reachable from any other is $\log(n)$ steps)
  - **degree distribution** of nodes is power-law

  Which of these is necessary/sufficient?