Structured Quantum Search: Quantum Walks and Quantum Backtracking

Simon Martiel

November 5, 2021

Everything is based on:

- Ronald de Wolf's lecture note on quantum algorithms
- Ashley's Montanaro papers on quantum backtracking

Grover (unstructured search)
Quick reminder

Input: a predicate $f : \{0,1\}^n \to \{0,1\}$ that marks elements in $M \subseteq X$

Goal: find x s.t. $f(x) = 1$

Grover (unstructured search)

Quick reminder

Input: a predicate $f : X \to \{0, 1\}$ that marks elements in $M \subseteq X$

Goal: find x s.t. $f(x) = 1$

Grover (unstructured search)

Quick reminder

Input: a predicate $f : X \to \{0,1\}$ that marks elements in $M \subseteq X$

Goal: find x s.t. $f(x) = 1$

Classical algorithm:

1. pick x uniformly at random
2. if $f(x)$, output x, else go to (1)

$\implies O(\frac{1}{\varepsilon})$ calls to f where $\varepsilon = \frac{|M|}{|X|}$

Grover (unstructured search)
Quick reminder

Input: a predicate $f : X \to \{0,1\}$ that marks elements in $M \subseteq X$

Goal: find x s.t. $f(x) = 1$

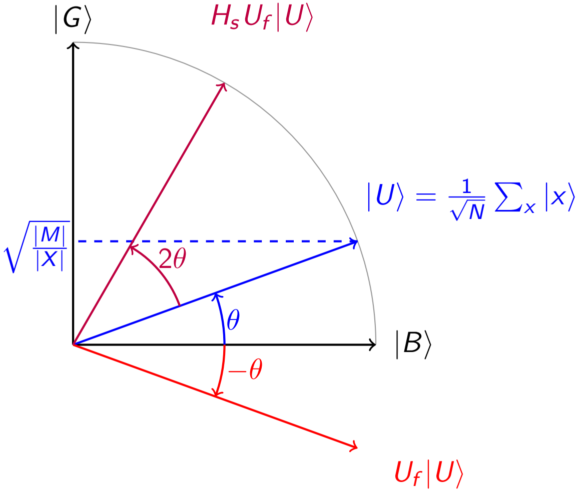
Quantum algorithm:

1. start from $|U\rangle = \frac{1}{\sqrt{|X|}} \sum_{x \in X} |x\rangle$
2. $O\left(\frac{1}{\sqrt{\varepsilon}}\right)$ times:
   - 2.1 reflect through $|B\rangle = \frac{1}{\sqrt{|X| - |M|}} \sum_{x \in X, P(x)=0} |x\rangle$ (call to U_f)
   - 2.2 reflect through $|U\rangle$ ($-H_s$ from Emmanuel's talk)
3. measure the system (go to (1) if the result is invalid)

$\implies O(\sqrt{1/\varepsilon})$ calls to U_f

Grover (unstructured search)

Quick reminder



Moving to structured search

Grover assumes NO structure on f:

- X is simple to sample uniformly
- X is huge (we never "prune" the search space)

Real life problems have structure :

- the search space can usually be "pruned"
- but this "pruned" space is hard to sample uniformly

Example: SAT

- Grover will "explore" $\{0,1\}^n$
- a smart classical backtracking (e.g. DPLL) will explore a tiny portion of $\{0,1\}^n$
- thus Grover fails to bring a quadratic speed up

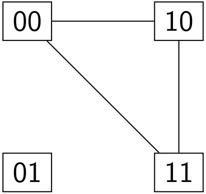
How can we "restrict" a quantum search to this "pruned" space ?

How to give structure to X ?

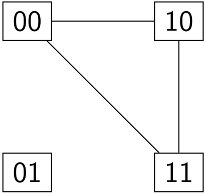
$\boxed{00}$ $\boxed{10}$

$\boxed{01}$ $\boxed{11}$

How to give structure to X ?



How to give structure to X ?



Natural structure: a graph $G(X, E)$

Random walk as a search algorithm

Question: how search for maked elements in a graph?

Random search in a graph:

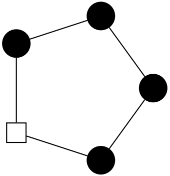
1. Pick any vertex x of G (randomly, or not)
2. While not $f(x)$:
   2.1 pick a neighbor y in $\mathcal{N}_x$ at random
   2.2 set $x \leftarrow y$

Random walk as a search algorithm

Question: how search for maked elements in a graph?

Random search in a graph:

1. Pick any vertex x of G (randomly, or not)
2. While not $f(x)$:
    2.1 pick a neighbor y in $\mathcal{N}_x$ at random
    2.2 set $x \leftarrow y$

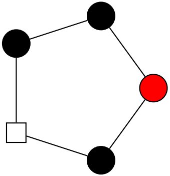


Random walk as a search algorithm

Question: how search for maked elements in a graph?

Random search in a graph:

1. Pick any vertex x of G (randomly, or not)
2. While not $f(x)$:
   - 2.1 pick a neighbor y in $\mathcal{N}_x$ at random
   - 2.2 set $x \leftarrow y$

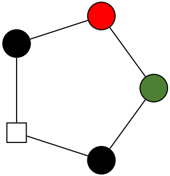


Random walk as a search algorithm

Question: how search for maked elements in a graph?

Random search in a graph:

1. Pick any vertex x of G (randomly, or not)
2. While not $f(x)$:
   2.1 pick a neighbor y in $\mathcal{N}_x$ at random
   2.2 set $x \leftarrow y$

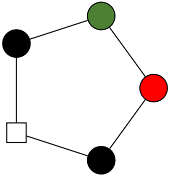


Random walk as a search algorithm

Question: how search for maked elements in a graph?

Random search in a graph:

1. Pick any vertex x of G (randomly, or not)
2. While not $f(x)$:
   - 2.1 pick a neighbor y in $\mathcal{N}_x$ at random
   - 2.2 set $x \leftarrow y$

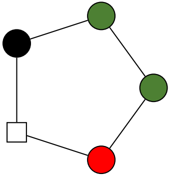


Random walk as a search algorithm

Question: how search for maked elements in a graph?

Random search in a graph:

1. Pick any vertex x of G (randomly, or not)
2. While not $f(x)$:
   2.1 pick a neighbor y in $\mathcal{N}_x$ at random
   2.2 set $x \leftarrow y$

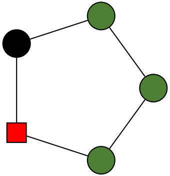


Random walk as a search algorithm

Question: how search for maked elements in a graph?

Random search in a graph:

1. Pick any vertex x of G (randomly, or not)
2. While not $f(x)$:
   2.1 pick a neighbor y in $\mathcal{N}_x$ at random
   2.2 set $x \leftarrow y$

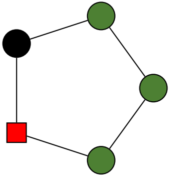


Random walk as a search algorithm

Question: how search for maked elements in a graph?

Random search in a graph:

1. Pick any vertex x of G (randomly, or not)
2. While not $f(x)$:
   2.1 pick a neighbor y in $\mathcal{N}_x$ at random
   2.2 set $x \leftarrow y$



How can we analyze this algorithm ?

Random walk as a search algorithm

Theorem

If G is connected, not bipartite, and of constant degree:

- *this process converges to the uniform distribution over X*
- *the speed of convergence depends on the spectral gap δ of G*

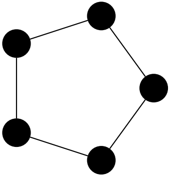
$$\delta = \lambda_0 - \max |\lambda_i|$$

where λ_i are eigenvalues of the (normalized) adjacency matrix of G, $\lambda_0 = 1$ being the largest one.

Convergence "speed" in $O(\frac{1}{\delta})$

Random walk as a search algorithm

What is this spectral gap thing ?



$$P = \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Eigenvalues of P: 1 0.309 0.309 - 0.809 - 0.809

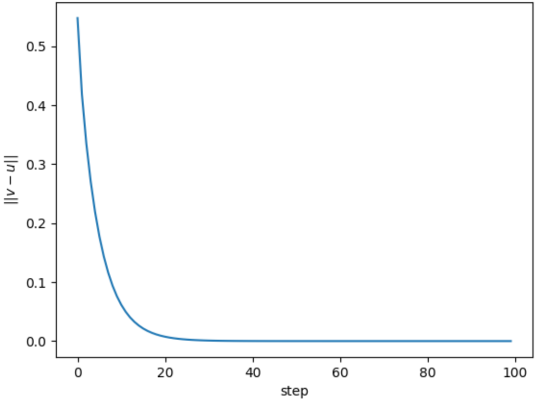
Hence $\delta = \lambda_0 - \max|\lambda_i| = 1 - 0.809 \approx 0.19$

Random walk as a search algorithm

In other words:

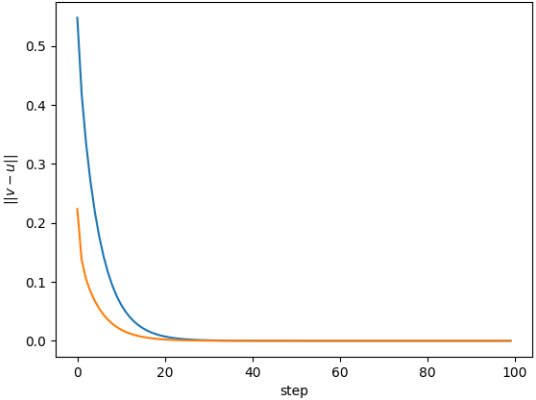
- we can associate to G some quantity δ
- if we "walk" randomly around G for $O(\frac{1}{\delta})$ steps we end up picking a vertex uniformly at random

Random walk as a search algorithm



Initial distribution: $[1, 0, 0, 0, 0]$

Random walk as a search algorithm



Initial distribution: $[1, 0, 0, 0, 0]$ $[0.5, 0.5, 0, 0, 0]$

Random walk as a search algorithm

Random search in a graph:

1. Pick any vertex x of G (randomly, or not)
2. repeat $O(\frac{1}{\delta})$ times:
   2.1 pick a neighbor y in $\mathcal{N}_x$ at random
   2.2 set $x \leftarrow y$
3. if $f(x)$ return x else go to (2)

Random walk as a search algorithm

Random search in a graph:

1. Pick any vertex x of G (randomly, or not)
2. repeat $O(\frac{1}{\delta})$ times:
   2.1 pick a neighbor y in $\mathcal{N}_x$ at random
   2.2 set $x \leftarrow y$
3. if $f(x)$ return x else go to (2) $\leftarrow$ x is picked almost uniformly

Expected cost of this procedure:

$$S + \frac{1}{\varepsilon}(C + \frac{1}{\delta}N)$$

S: cost of picking the first vertex
C: cost of a call to the predicate f
N: cost of picking a neighbor of a vertex

Random walk as a search algorithm

Some simple remarks:

- if G is the complete graph, this is the classical algorithm of the first slide
$$\delta_{K_N} = 1 - \frac{1}{N}$$
$\implies$ very large gap
- this is a cheap algorithm ($O(log(|X|))$ space)
- this is a tool to uniformly sample structured spaces : e.g. sampling valid partial assignments in a CSP

There is a "quantization" of this approach: Quantum Walks !

Quantum walks

We need two registers storing vertices:

$$|x\rangle|y\rangle, x, y \in X$$

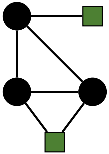
Morally:

- $|x\rangle$ stores the previous vertex
- $|y\rangle$ stores the current vertex

Notation:

$$|p_x\rangle = \frac{1}{\sqrt{d(x)}} \sum_{y \in \mathcal{N}_x} |y\rangle$$

$$|\phi_x\rangle = |x\rangle|p_x\rangle$$



Quantum walks

We need two registers storing vertices:

$$|x\rangle|y\rangle, x, y \in X$$

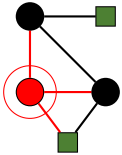
Morally:

- $|x\rangle$ stores the previous vertex
- $|y\rangle$ stores the current vertex

Notation:

$$|p_x\rangle = \frac{1}{\sqrt{d(x)}} \sum_{y \in \mathcal{N}_x} |y\rangle$$

$$|\phi_x\rangle = |x\rangle|p_x\rangle$$



Quantum walks

Good state:

$$|G\rangle = \frac{1}{\sqrt{|M|}} \sum_{x \in M} |x\rangle |p_x\rangle$$

Bad state:

$$|B\rangle = \frac{1}{\sqrt{|X| - |M|}} \sum_{x \notin M} |x\rangle |p_x\rangle$$

Uniform state:

$$|U\rangle = \frac{1}{\sqrt{|X|}} \sum_{x \in X} |x\rangle |p_x\rangle = \sin\theta |G\rangle + \cos\theta |B\rangle$$

with $\theta = \arcsin\left(\sqrt{\varepsilon}\right)$

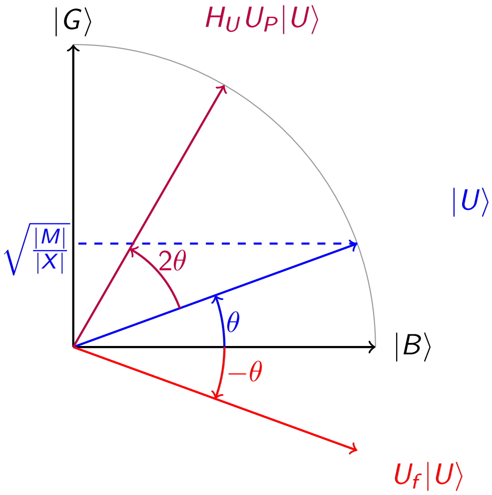
Quantum search with a Quantum Walk:

1. prepare the system in state $|U\rangle$
2. Repeat $O(\frac{1}{\sqrt{\varepsilon}})$ times
   2.1 reflect through $|B\rangle$
   2.2 reflect through $|U\rangle$
3. measure and check if the result is valid

Proof that it works: same as Grover

Quantum walk

Quantum Search



After p steps: $\sin\left((2p+1)\theta\right)|G\rangle + \cos\left((2p+1)\theta\right)|B\rangle$

If we manage to implement:

- a reflection through $|B\rangle$
- a reflection through $|U\rangle$

We have a similar argument as Grover

Reflection through $|B\rangle$:

$$|x\rangle \mapsto (-1)^{f(x)}|x\rangle$$

Reflection through $|U\rangle$: tricky

Theorem

Reflection through $|U\rangle$ can be implemented using $O(\frac{1}{\sqrt{\delta}})$ calls to:

$$N : |x\rangle|0\rangle \mapsto |x\rangle|p_x\rangle$$

Final complexity:

$$S + \frac{1}{\sqrt{\varepsilon}}(C + \frac{1}{\sqrt{\delta}}N)$$

Quantum Walk

Magniez Nayak Roland Santha (MRNS)

Reflection through $|B\rangle$:

$$|x\rangle \mapsto (-1)^{f(x)}|x\rangle$$

Reflection through $|U\rangle$: tricky

Theorem

Reflection through $|U\rangle$ can be implemented using $O(\frac{1}{\sqrt{\delta}})$ calls to:

$$N : |x\rangle|0\rangle \mapsto |x\rangle|p_x\rangle$$

Final complexity:

$$S + \frac{1}{\sqrt{\varepsilon}}(C + \frac{1}{\sqrt{\delta}}N)$$

Proof: using a Quantum Phase Estimation to distinguish $|U\rangle$ from other states

Aparté on Quantum Phase Estimation

Input: a unitary operator W and an eigenvector $|s\rangle$:

$$W|s\rangle = e^{2i\pi\theta_s}|s\rangle$$

Goal: produce an estimate of θ_s

Quantum Phase Estimation:

$$QPE|s\rangle|0\rangle \mapsto |s\rangle|\tilde{\theta}_s\rangle$$

where $\tilde{\theta}_s = \lfloor \theta_s \times 2^k \rfloor$

Aparté on Quantum Phase Estimation

$$W|s\rangle = e^{0.8125*2i\pi}|s\rangle$$

with $k = 0$?

$$0.8125 \times 2^0 = 0.8125 \approx 0$$

$$\tilde{\theta}_s = "0"$$

Aparté on Quantum Phase Estimation

$$W|s\rangle = e^{0.8125 * 2i\pi}|s\rangle$$

with $k = 1$?

$$0.8125 \times 2^1 = 1.625 \approx 1$$

$$\tilde{\theta}_s = "01"$$

Aparté on Quantum Phase Estimation

$$W|s\rangle = e^{0.8125*2i\pi}|s\rangle$$

with $k = 2$?

$$0.8125 \times 2^2 = 3.25 \approx 3$$

$$\tilde{\theta}_s = "011"$$

Aparté on Quantum Phase Estimation

$$W|s\rangle = e^{0.8125*2i\pi}|s\rangle$$

with $k = 3$?

$$0.8125 \times 2^3 = 6.5 \approx 6$$

$$\tilde{\theta}_s = "0110"$$

Aparté on Quantum Phase Estimation

$$W|s\rangle = e^{0.8125*2i\pi}|s\rangle$$

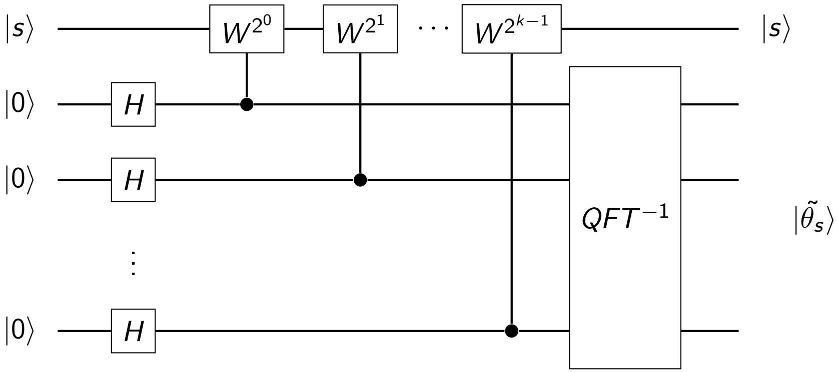
with $k = 4$?

$$0.8125 \times 2^4 = 13$$

$$\tilde{\theta}_s = "01101"$$

Aparté on Quantum Phase Estimation

How to?



Aparté on Quantum Phase Estimation

Step by step

$$
\begin{aligned}
|s\rangle|0\rangle &\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} |s\rangle|x\rangle && H^{\otimes k} \\
&\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} W^{x_0}|s\rangle|x\rangle && C-W^{2^0}
\end{aligned}
$$

Aparté on Quantum Phase Estimation

Step by step

$$|s\rangle|0\rangle \mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} |s\rangle|x\rangle \qquad H^{\otimes k}$$

$$\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} e^{2i\pi x_0\theta_s}|s\rangle|x\rangle \qquad C-W^{2^0}$$

Aparté on Quantum Phase Estimation

Step by step

$$
\begin{aligned}
|s\rangle|0\rangle &\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} |s\rangle|x\rangle && H^{\otimes k} \\
&\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} e^{2i\pi x_0\theta_s}|s\rangle|x\rangle && C-W^{2^0} \\
&\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} e^{2i\pi x_0\theta_s} W^{2^{x_1}}|s\rangle|x\rangle && C-W^{2^1}
\end{aligned}
$$

Aparté on Quantum Phase Estimation

Step by step

$$
\begin{aligned}
|s\rangle|0\rangle &\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} |s\rangle|x\rangle && H^{\otimes k} \\
&\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} e^{2i\pi x_0\theta_s}|s\rangle|x\rangle && C-W^{2^0} \\
&\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} e^{2i\pi (x_0+2x_1)\theta_s}|s\rangle|x\rangle && C-W^{2^1}
\end{aligned}
$$

Aparté on Quantum Phase Estimation

Step by step

$$
\begin{aligned}
|s\rangle|0\rangle &\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} |s\rangle|x\rangle && H^{\otimes k} \\
&\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} e^{2i\pi x_0\theta_s}|s\rangle|x\rangle && C-W^{2^0} \\
&\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} e^{2i\pi(x_0+2x_1)\theta_s}|s\rangle|x\rangle && C-W^{2^1} \\
&\vdots \\
&\mapsto \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} e^{2i\pi x\theta_s}|s\rangle|x\rangle && C-W^{2^{k-1}}
\end{aligned}
$$

$$|s\rangle \left[\frac{1}{\sqrt{2^k}} \sum_{x \in \{0,1\}^k} e^{2i\pi x \theta_s} |x\rangle \right]$$

The thing on the right can be written as :

$$\frac{1}{\sqrt{2^k}} \sum_{x \in \{0,1\}^k} e^{2i\pi x \tilde{\theta}_s / 2^k} |x\rangle$$

$$|s\rangle \left[\frac{1}{\sqrt{2^k}} \sum_{x \in \{0,1\}^k} e^{2i\pi x \theta_s} |x\rangle \right]$$

The thing on the right can be written as :

$$\frac{1}{\sqrt{2^k}} \sum_{x \in \{0,1\}^k} e^{\frac{2i\pi}{2^k} x \tilde{\theta}_s} |x\rangle$$

$$|s\rangle \left[\frac{1}{\sqrt{2^k}} \sum_{x \in \{0,1\}^k} e^{2i\pi x \theta_s} |x\rangle \right]$$

The thing on the right can be written as :

$$\frac{1}{\sqrt{2^k}} \sum_{x \in \{0,1\}^k} \omega_{2^k}^{x\tilde{\theta}_s} |x\rangle$$

where ω_{2^k} is the 2^kth root of the unity

$$|s\rangle \left[\frac{1}{\sqrt{2^k}} \sum_{x \in \{0,1\}^k} e^{2i\pi x \theta_s} |x\rangle \right]$$

The thing on the right can be written as :

$$\frac{1}{\sqrt{2^k}} \sum_{x \in \{0,1\}^k} y_x |x\rangle$$

where y_x is the xth coefficient of the Fourier transform of $\tilde{\theta}_s$

Quantum Fourier Transform

In one slide :/

Theorem

One can build a circuit with $O(n^2)$ gates that performs a Fourier Transform:

$$QFT|x\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_y \omega_{2^n}^{xy} |y\rangle$$

Tiny remark: classically this requires $O(n2^n)$ classical operations (FFT)

Consequently (for our application), one can inverse the QFT in $O(k^2)$:

$$|s\rangle \underbrace{\left[\frac{1}{\sqrt{2^k}} \sum_{x \in \{0,1\}^k} e^{ix\theta_s} |x\rangle\right]}_{\approx QFT|\tilde{\theta}_s\rangle} \mapsto |s\rangle|\tilde{\theta}_s\rangle \qquad QFT^{-1}$$

Summing up

$$QPE|s\rangle|0\rangle \mapsto |s\rangle|\tilde{\theta}_s\rangle$$

Aparté on Quantum Phase Estimation

Step by step

Summing up

$$QPE|s\rangle|0\rangle \mapsto |s\rangle|\tilde{\theta}_s\rangle$$

Complexity:

$$O(2^k) \text{ calls to } W \text{ for a resolution of } 2^{-k}$$

Summing up

$$QPE|s\rangle|0\rangle \mapsto |s\rangle|\tilde{\theta}_s\rangle$$

Complexity:

$$O(\tfrac{1}{\nu}) \text{ calls to } W \text{ for a resolution of } \nu$$

Back to quantum walks

We want to perform a reflection through:

$$|U\rangle = \sum_x |x\rangle|p_x\rangle$$

One can build a $W(G)$ such that:

$$W(G)|U\rangle = |U\rangle$$

using 4 calls to:

$$N : |x\rangle|0\rangle \mapsto |x\rangle|p_x\rangle$$

Back to quantum walks

What is this $W(G)$?

- a reflection through $\mathcal{A} = span\{|x\rangle|p_x\rangle\}$
- a reflection through $\mathcal{B} = span\{|p_y\rangle|y\rangle\}$

Implementing reflections

What is a reflection $R(\mathcal{C})$ through some subspace $\mathcal{C}$?
if $w \in \mathcal{C}$:

$$R(\mathcal{C}) \cdot w = w$$

if w is orthogonal to $\mathcal{C}$:

$$R(\mathcal{C}) \cdot w = -w$$

Implementing reflections

How to implement $R(\mathcal{A}) = R(span\{|x\rangle|p_x\rangle\})$?
Using our "graph" oracle:

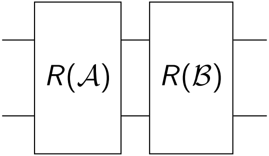
$$N : |x\rangle|0\rangle \mapsto |x\rangle|p_x\rangle$$

1. apply $N^{-1} : |x\rangle|p_x\rangle \mapsto |x\rangle|0\rangle$
   vectors in $\mathcal{A}$ now have a $|0\rangle$ in the second register
2. If the second register is NOT $|0\rangle$, inverse the phase
3. apply N

$R(\mathcal{B})$: same thing, just invert the two registers

Back to quantum walks

Now, we know how to implement $W(G)$:



Back to quantum walks

Some people can prove that:

$$W(G)|U\rangle = |U\rangle = e^{i0}|U\rangle$$

Back to quantum walks

Some people can prove that:

$$W(G)|U\rangle = |U\rangle = e^{i0}|U\rangle$$

and that the other eigenvalues of $W(G)$ are:

$$e^{\pm 2i \arccos(|\lambda_j|)} = e^{\pm i\theta_j}$$

where λ_j are the eigenvalues of G.

Back to quantum walks

Some people can prove that:

$$W(G)|U\rangle = |U\rangle = e^{i0}|U\rangle$$

and that the other eigenvalues of $W(G)$ are:

$$e^{\pm 2i \arccos(|\lambda_j|)} = e^{\pm i\theta_j}$$

where λ_j are the eigenvalues of G.
Thus we can bound:

$$\cos(\theta_j) = |\lambda_j| \leq 1 - \delta$$

and using:

$$\cos(\theta) \geq 1 - \frac{\theta^2}{2}$$

we get:

$$\theta_j \geq \sqrt{2\delta}$$

Implementing $R(|U\rangle)$

We have a $W(G)$ and we need to distinguish:

- $|U\rangle$, its $1 = e^{i0}$ eigenvector
- from other eigenvectors with eigenvalues $e^{\pm i\theta_j}$ with

$$\theta_j \geq \sqrt{2\delta}$$

Implementing $R(|U\rangle)$

We have a $W(G)$ and we need to distinguish:

- $|U\rangle$, its $1 = e^{i0}$ eigenvector
- from other eigenvectors with eigenvalues $e^{\pm i\theta_j}$ with

$$\theta_j \geq \sqrt{2\delta}$$

We can use a QPE:

$$\begin{aligned}
|w\rangle|0\rangle &\mapsto |w\rangle|\tilde{\theta_w}\rangle && QPE \\
&\mapsto (-1)^{\tilde{\theta_w}\neq 0}|w\rangle|\tilde{\theta_w}\rangle && \text{simple oracle} \\
&\mapsto (-1)^{\tilde{\theta_w}\neq 0}|w\rangle|0\rangle && QPE^{-1}
\end{aligned}$$

We will need a precision of about $\sqrt{\delta}/2$!

Implementing $R(|U\rangle)$

We have a $W(G)$ and we need to distinguish:

- $|U\rangle$, its $1 = e^{i0}$ eigenvector
- from other eigenvectors with eigenvalues $e^{\pm i\theta_j}$ with

$$\theta_j \geq \sqrt{2\delta}$$

We can use a QPE:

$$\begin{aligned}
|w\rangle|0\rangle &\mapsto |w\rangle|\tilde{\theta_w}\rangle && QPE \\
&\mapsto (-1)^{\tilde{\theta_w}\neq 0}|w\rangle|\tilde{\theta_w}\rangle && \text{simple oracle} \\
&\mapsto (-1)^{\tilde{\theta_w}\neq 0}|w\rangle|0\rangle && QPE^{-1}
\end{aligned}$$

We will need a precision of about $\sqrt{\delta}/2$!
Hence, this will require $O(\frac{1}{\sqrt{\delta}})$ calls to $W(G)$

Summing up the algorithm

Step 1: prepare the unform state

$$|U\rangle = \frac{1}{\sqrt{|X|}} \sum_x |x\rangle |p_x\rangle$$

Step 2: perform $O(\frac{1}{\sqrt{\varepsilon}})$ "amplification" steps:

- use U_f to reflect through $|B\rangle$
- use $R(|U\rangle)$ to reflect through $|U\rangle$:
  - use $O(\frac{1}{\sqrt{\delta}})$ calls to N to perform a QPE of $W(G)$
  - flip the phase if the estimate of θ_j is not 0
  - undo the QPE

Step 3: measure the first register

Example

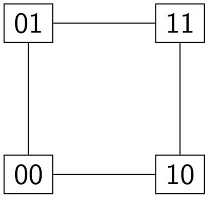
Here I should alt-tab and show some implementation

github.com $\rightarrow$ smartiel + quantum_walks_example

We will use: myqlm.github.io

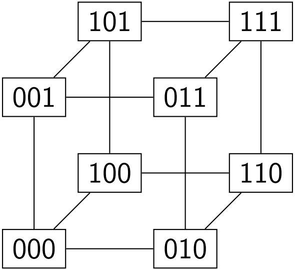
An hypercube of dimension n

Dimension 2:



An hypercube of dimension n

Dimension 3:



An hypercube of dimension n

Dimension n:
neighbors of $x = (x_1, ..., x_i, ..., x_n)$ are $(x_1, ..., \neg x_i, ..., x_n)$
For our QWalk we need to implement:

An hypercube of dimension n

Dimension n:
neighbors of $x = (x_1, ..., x_i, ..., x_n)$ are $(x_1, ..., \neg x_i, ..., x_n)$
For our QWalk we need to implement:

An hypercube of dimension n

$$N : |x\rangle|0\rangle \mapsto |x\rangle|p_x\rangle$$

An hypercube of dimension n

$$N : |x\rangle|0\rangle \mapsto \frac{1}{n}|x\rangle \sum_{y \in \mathcal{N}_x} |y\rangle$$

Step 1:

$$|x\rangle|0\rangle \mapsto \frac{1}{\sqrt{n}}|x\rangle \sum_i |e_i\rangle$$

Step 2:

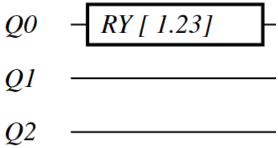
$$\frac{1}{\sqrt{n}}|x\rangle \sum_i |e_i\rangle \mapsto \frac{1}{\sqrt{n}}|x\rangle \sum_i |x \oplus e_i\rangle$$

An hypercube of dimension n

$1|000\rangle$

An hypercube of dimension n

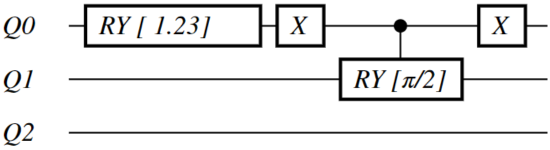
$$\begin{tikzcd} 1|000\rangle \arrow[r] \arrow[dr] & \frac{1}{\sqrt{3}}|100\rangle \\ & \sqrt{\frac{2}{3}}|000\rangle \end{tikzcd}$$



$$2\arcsin\left(\frac{1}{\sqrt{3}}\right) \approx 1.23$$

An hypercube of dimension n

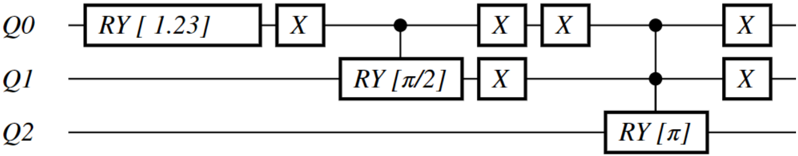
$$
\begin{tikzcd}
1|000\rangle \arrow[r] \arrow[dr] & \frac{1}{\sqrt{3}}|100\rangle \arrow[r] & \frac{1}{\sqrt{3}}|100\rangle \\
& \sqrt{\frac{2}{3}}|000\rangle \arrow[r] \arrow[dr] & \frac{1}{\sqrt{3}}|010\rangle \\
& & \frac{1}{\sqrt{3}}|000\rangle
\end{tikzcd}
$$



$$2\arcsin\left(\frac{1}{\sqrt{2}}\right) = \pi/2$$

An hypercube of dimension n

$$\begin{array}{ccccccc}
1|000\rangle & \longrightarrow & \frac{1}{\sqrt{3}}|100\rangle & \longrightarrow & \frac{1}{\sqrt{3}}|100\rangle & \longrightarrow & \frac{1}{\sqrt{3}}|100\rangle \\
& \searrow & & & & & \\
& & \sqrt{\frac{2}{3}}|000\rangle & \longrightarrow & \frac{1}{\sqrt{3}}|010\rangle & \longrightarrow & \frac{1}{\sqrt{3}}|010\rangle \\
& & & \searrow & & & \\
& & & & \frac{1}{\sqrt{3}}|000\rangle & \longrightarrow & \frac{1}{\sqrt{3}}|001\rangle
\end{array}$$



$$2\arcsin(1) = \pi$$

Le retour de l'arnaque

We presented a particular framework :

$$S + \frac{1}{\sqrt{\varepsilon}}(C + \frac{1}{\sqrt{\delta}}N)$$

We assumed that we can prepare $|U\rangle = \frac{1}{\sqrt{|X|}} \sum_x |x\rangle|p_x\rangle$

There are many quantum walk frameworks:

Framework	Complexity
Hitting time framework [Sze04, KMOR16, AGJK19]	$\mathsf{S} + \sqrt{\mathrm{HT}(P, M)}(\mathsf{U} + \mathsf{C})$
MNRS framework [MNRS11]	$\mathsf{S} + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C})$
Electric network framework [BCJ+13, Bel13]	$\mathsf{S}(\sigma) + \sqrt{C_{\sigma,M}}(\mathsf{U}(\sigma) + \mathsf{C})$
Controlled quantum amplification [DH17]	$\mathsf{S} + \sqrt{\mathrm{HT}(P, \{m\})}\mathsf{U} + \frac{1}{\sqrt{\varepsilon}}\mathsf{C}$

Table 1: Comparison of different quantum walk frameworks.

They were unified recently !

(Not so) Quantum Backtracking

Consider a CSP:

- with n variables $x_1, ..., x_n$
- each variable takes value in $[d]$
- a predicate $P : [d]^n \to \{0, 1\}$

Goal: find $x \in [d]^n$ s.t. $P(x) = 1$

(Not so) Quantum Backtracking

Consider a CSP:

- with n variables $x_1, ..., x_n$
- each variable takes value in $[d]$
- a predicate $P : [d]^n \to \{0, 1\}$

Goal: find $x \in [d]^n$ s.t. $P(x) = 1$

Ingredients for a backtracking:

- a variant of P that can check partial assignments:

$$P' : ([d] \cup \{\star\})^n \to \{0, 1, \mathrm{indeterminate}\}$$

- a heuristic h that tells me what to do next:

$$h : ([d] \cup \{\star\})^n \to [n]$$

(Not so) Quantum Backtracking

Backtracking algorithm:

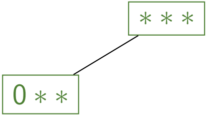
$\texttt{bt}(x)$:

1. If $P(x)$ is true, output x and return.
2. If $P(x)$ is false, or x is a complete assignment, return.
3. Set $j = h(x)$.
4. For each $w \in [d]$:
   (a) Set y to x with the j'th entry replaced with w.
   (b) Call $\texttt{bt}(y)$.

$\Longrightarrow$ call $bt(\star^n)$

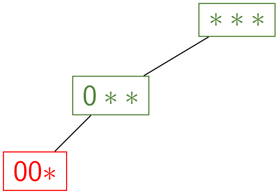
(Not so) Quantum Backtracking

Example: satisfy $a \oplus b \wedge b \oplus c$



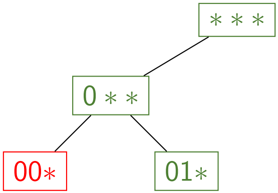
(Not so) Quantum Backtracking

Example: satisfy $a \oplus b \wedge b \oplus c$



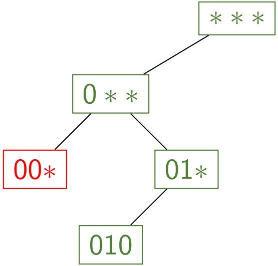
(Not so) Quantum Backtracking

Example: satisfy $a \oplus b \wedge b \oplus c$



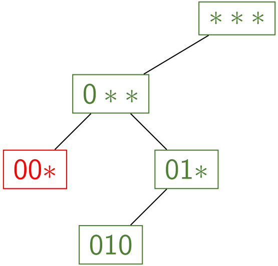
(Not so) Quantum Backtracking

Example: satisfy $a \oplus b \wedge b \oplus c$



(Not so) Quantum Backtracking

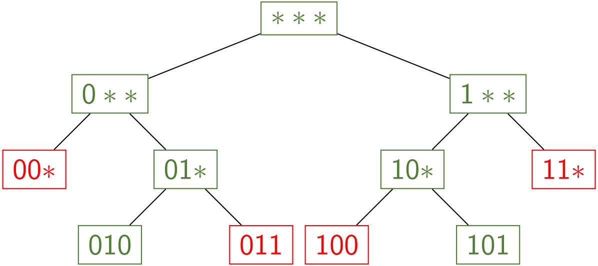
Example: satisfy $a \oplus b \wedge b \oplus c$



We explored 5 valuations !

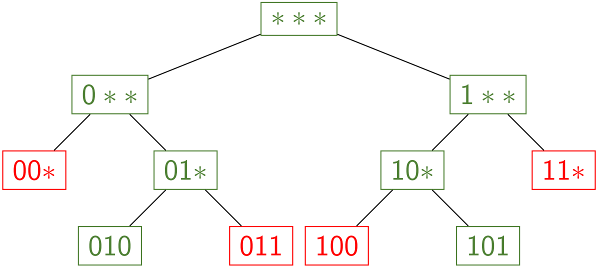
(Not so) Quantum Backtracking

We could run it further to enumerate all the solutions:



(Not so) Quantum Backtracking

We could run it further to enumerate all the solutions:



Total number of partial assignments:

$$3^3 = 27$$

Here we explored 11 partial assignments

Main idea:

- P/P' can be turned into quantum oracles
- h can be turned into a quantum oracle
- we have all we need to run a Quantum Walk on the backtracking tree !

Montanaro's Quantum Backtracking

We store assignments in the quantum memory as arrays:

$$|v_1\rangle \dots |v_n\rangle$$

$v_i \in [d] \cup \{\star\}$

Use U_P and U_h to implement:

- a reflection $R(\mathcal{A}) = R(span\{|x\rangle|p_x\rangle\})$
- a reflection $R(\mathcal{B}) = R(span\{|p_y\rangle|y\rangle\})$

This should seem familiar :)

Let's call it W !

Montanaro's Quantum Backtracking

First remark: we can't produce $|U\rangle$
(this would entail knowing the "pruned" tree)

Lemma

W admits as eigenvector of eigenvalue 1 superpositions of shape:

$$\sqrt{n}|\,\mathrm{root}\rangle + \sum_{x \neq r, x \to x_0} (-1)^{l(x)} |x\rangle$$

where x_0 is a valid leaf of the backtracking tree.

Montanaro's Quantum Backtracking

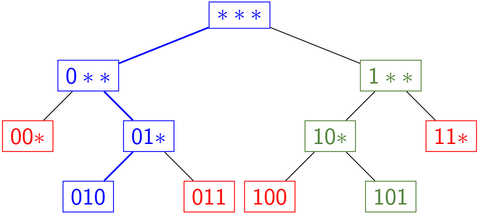
First remark: we can't produce $|U\rangle$
(this would entail knowing the "pruned" tree)

Lemma

W admits as eigenvector of eigenvalue 1 superpositions of shape:

$$\sqrt{n}|\text{ root}\rangle + \sum_{x \neq r, x \to x_0} (-1)^{l(x)} |x\rangle$$

where x_0 is a valid leaf of the backtracking tree.



$$W\left(\sqrt{3}| * **\rangle - |0 * *\rangle + |01*\rangle - |010\rangle\right) = \sqrt{3}|***\rangle - |0**\rangle + |01*\rangle - |010\rangle$$

Montanaro's Quantum Backtracking

For any $|\phi\rangle$ that is an eigenvector of eigenvalue 1 of W, we have:

$$\langle \text{root} \,|\phi\rangle \geq \frac{1}{\sqrt{2}}$$

Montanaro's Quantum Backtracking

For any $|\phi\rangle$ that is an eigenvector of eigenvalue 1 of W, we have:

$$\langle \text{root} \,|\phi\rangle \geq \frac{1}{\sqrt{2}}$$

This entails that:

- the $|\,\text{root}\rangle = |\,**...*\rangle$ state overlaps these eigenvectors "well"
- in particular, we can run a Quantum Phase Estimation on W to see if we can detect an eigenvalue $1 = e^{i0}$

Montanaro's Quantum Backtracking

For any $|\phi\rangle$ that is an eigenvector of eigenvalue 1 of W, we have:

$$\langle \text{root} | \phi \rangle \geq \frac{1}{\sqrt{2}}$$

This entails that:

- the $|\text{root}\rangle = |**...*\rangle$ state overlaps these eigenvectors "well"
- in particular, we can run a Quantum Phase Estimation on W to see if we can detect an eigenvalue $1 = e^{i0}$

Hence we can build a routine that:

- prepare an initial state $|\text{root}\rangle|p_{root}\rangle$
- perform a QPE on this state
- measure the phase estimation

If we measure $\theta = 0$, it means that there is a valid leaf !

Montanaro's Quantum Backtracking

The cost is determined by the precision of the QPE.

Theorem

We need a precision of $O(\frac{1}{\sqrt{Tn}})$ where T is the size of the backtracking tree (or any upper bound on it)

Montanaro's Quantum Backtracking

The cost is determined by the precision of the QPE.

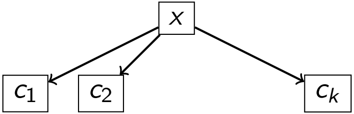
Theorem

We need $O(\sqrt{Tn})$ calls to W where T is the size of the backtracking tree (or any upper bound on it)

So we can detect the presence of a valid leaf in time $O(\sqrt{T})$.

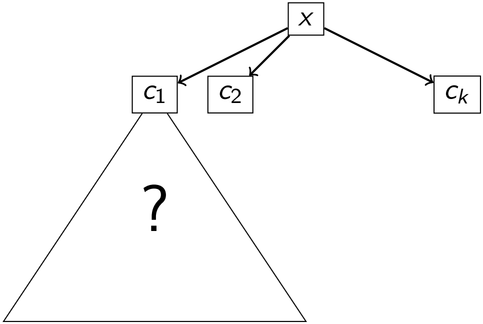
Montanaro's Quantum Backtracking

An hybrid algorithm



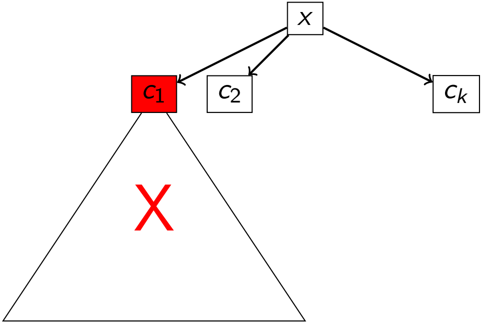
Montanaro's Quantum Backtracking

An hybrid algorithm



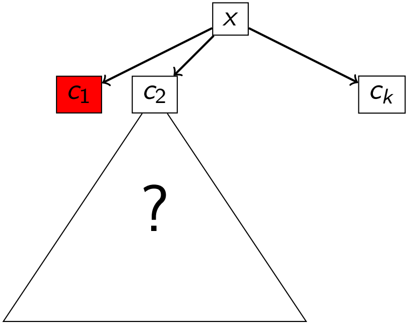
Montanaro's Quantum Backtracking

An hybrid algorithm



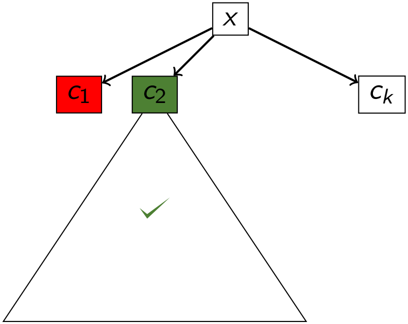
Montanaro's Quantum Backtracking

An hybrid algorithm



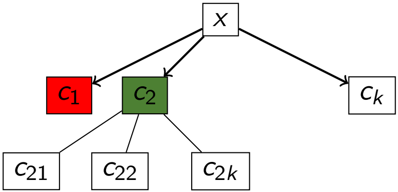
Montanaro's Quantum Backtracking

An hybrid algorithm



Montanaro's Quantum Backtracking

An hybrid algorithm



Montanaro's Quantum Backtracking

Complexity of finding a marked element (failure probability of δ):

$$O(\sqrt{T}n^{3/2}\log n \log 1/\delta)$$

Complexity of finding all marked elements:

$$O(k\sqrt{T}n^{3/2}\log n \log k/\delta)$$

Neat trick if there is a single marked element:

$$O(\sqrt{Tn}\log^3 n)$$

Applications

Campbell, Khurana, Montanaro

- estimate the size of the quantum circuit to implement W
- estimate the size of the backtracking tree for some problems (graph coloring and SAT solving)
- estimate the overhead due to quantum error correction
- deduce the effective advantage

Conclusion:

- it's (way) more competitive than Grover
- it needs lots of qubits (10^{13} for graphs that are not so big)
- for 1 day of computation: speed up of roughly 10^3 to 10^5 w.r.t to classical backtracking

A last word

Grover:

$$\frac{1}{\sqrt{\varepsilon}} C$$

Random walk:

$$S + \frac{1}{\varepsilon}(C + \frac{1}{\delta} N)$$

Quantum Walk (MRNS):

$$S + \frac{1}{\sqrt{\varepsilon}}(C + \frac{1}{\sqrt{\delta}} N)$$

Montanaro backtracking (more or less):

$$O(\sqrt{T})$$

Realistically:

- can provide speed up
- but the number of qubits is detrimental (10^{13})

Still gives hope for structured search !