A Quantum Triangle Finding Algorithm and Quipper Programming Language

Geoffrey Schumacher

Dvision of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

schum476@morris.umn.edu

April 29, 2014

Overview

Outline

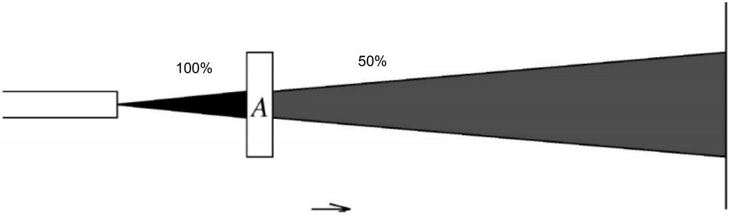
Superposition

- Quantum state x represented by $|x\rangle$
- Superposition of $a|x\rangle + b|y\rangle$
- Coefficients usually form $\frac{1}{\sqrt{A}}$
- Probability of $|x\rangle$ is $|a|^2 = \frac{1}{A}$
- $|a|^2 + |b|^2 = 1$

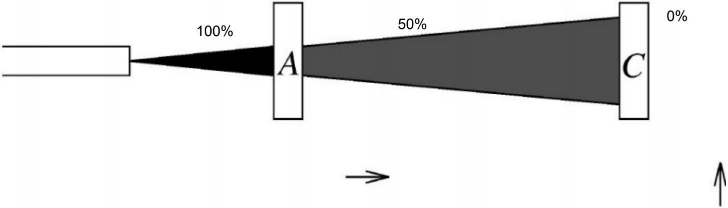
Photon Polarization Experiment

- Light Source (Laser Pointer)
- Lens A is Polarized $\rightarrow$
- Lens B is Polarized $\nearrow = \frac{1}{\sqrt{2}}(|\rightarrow\rangle + |\uparrow\rangle)$
- Lens C is Polarized $\uparrow$

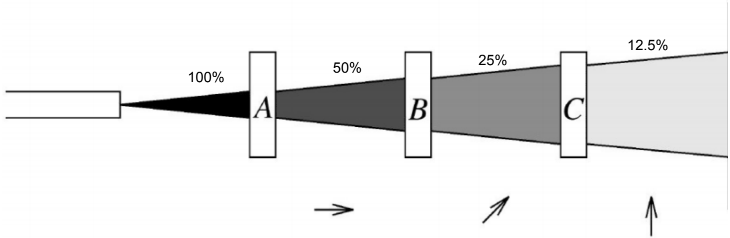
Photon Polarization Experiment



Photon Polarization Experiment



Photon Polarization Experiment



Outline

Qubits

- Classical Bits are represented by 0 and 1
- Quantum Bits are repsresented by $|0\rangle$ and $|1\rangle$
- Gates are used to put into superposition

Walsh-Hadamard Transformation

- Gates are transformation on bits
- *H* applied to *n* qubits is the *Walsh-Hadamard*, transformation *W*

$$H: \quad |0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Outline

General Purpose

- Black box function
- Given a state of n elements (represented as qubits)
- Runs in $O(\sqrt{n})$ queries to "black box"
- Probabilistic: finds answer with at least $\frac{1}{2}$ chance

Grover Example

- Set qubits to uniform distribution: $\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}} \cdots \frac{1}{\sqrt{N}}$
- $N = 2^n$
- Disturb state to eliminate non answers

Simple Example

▸ Grover Example

Outline

The Problem

- An undirected simple graph G of n vertices contains, at most, one triangle, $\triangle$
- To solve is to find the set of vertices $\{e_1, e_2, e_3\}$ that form $\triangle$
- Graph is "stored" within "black box" function
- Can go through n^2 edges in n time

Classical Algorithm with Quantum Speedup

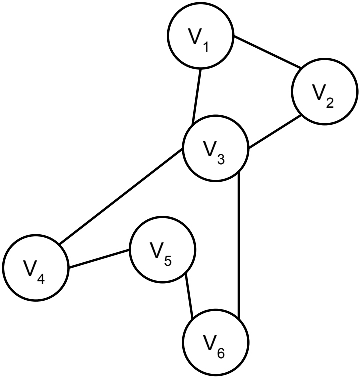
- Always rejects if there is no triangle in G
- Probabilistic: will return $\triangle$ with probability 1-$O(\frac{1}{n})$
- n is number of vertices
- Three inputs ϵ, δ, & ϵ'
- Efficiency $O(n^{\frac{10}{7}} log(n))$ with $\epsilon = \frac{3}{7}$, $\epsilon' = \delta = \frac{1}{7}$

Safe Grover Search

- Uses Safe Grover Search for subroutines
- Based on *n* iterations of Grover's Algorithm

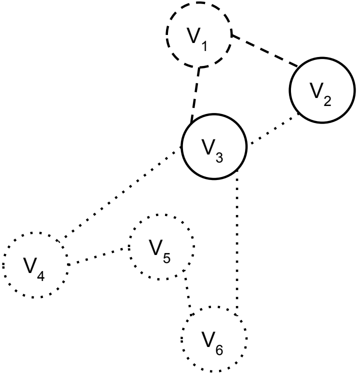
Steps 1 & 2: Set k & Choose Random Sample

- Set k $= \lceil 4n^{\epsilon} log(n) \rceil = 3$
- Choose vertices v_1, v_6, and v_5



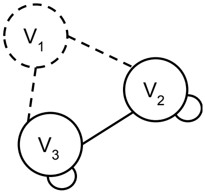
Step 3: Find Neighborhoods

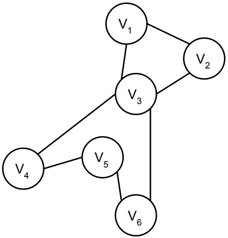
- Find nodes adjacent to v_1, v_6, and v_5
- Do not include node in question



Step 4: Check for Triangle

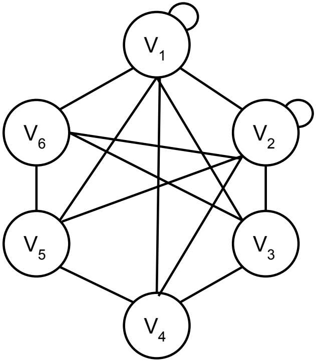
- Neighborhood $v_1 = \{v_2,\ v_3\}$
- Complete set of pairs: $\{(v_2,v_2),\ (v_2,v_3),\ (v_3,v_3)\}$
- Find interseciton of G and complete set of pairs from neighborhoods
- Safe Grover Search
- If any edge is in G, return $\triangle$





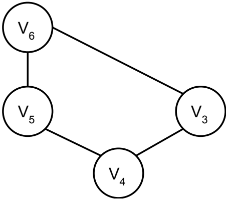
Step 5: Set G'

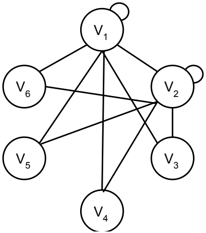
- New random sample: v_4, v_5, v_6
- $[n]^2 = \{(1,1),(1,2),...,(6,5),(6,6)\}$
- Set $G' = [n]^2 \setminus \cup_i \nu_G(v_i)$



Steps 6.1 & 6.2 (a): Add Edges to T

- Initialize graphs T & E as empty
- If $t(G', v, w) < n^{1-\epsilon'} = 4$, move to T





Step 6.2 (b): Sampling Strategy

- Try to find vertices with relatively high chance of being in $\triangle$
- If so, check neighborhood for $\triangle$
- If no reason to think so, put in E

Step 6.2 (b): Sampling Strategy

- Set a counter, C to 0
- Query 2 random edges from each pair of v from $[n]$
- If in G, increment C
- Safe Grover Search
- 2 evaluated from input δ

Step 6.2 (b): Sampling Strategy

- Repeat K (sufficient) times: 2.015
- If $C < K/2$ accept low-degree: 1.0007
- Else accept high-degree

Step 6.2 (b): Sampling Strategy

2 Sampling Rounds

$v_1 \rightarrow (v_1, v_6), (v_1, v_4)/(v_1, v_1), (v_1, v_5)$

$v_2 \rightarrow (v_2, v_5), (v_2, v_1)/(v_2, v_1), (v_2, v_6)$

$v_3 \rightarrow (v_3, v_1), (v_3, v_5)/(v_3, v_4), (v_3, v_6)$

$v_4 \rightarrow (v_4, v_2), (v_4, v_4)/(v_4, v_2), (v_4, v_3)$

$v_5 \rightarrow (v_5, v_3), (v_5, v_4)/(v_5, v_3), (v_5, v_5)$

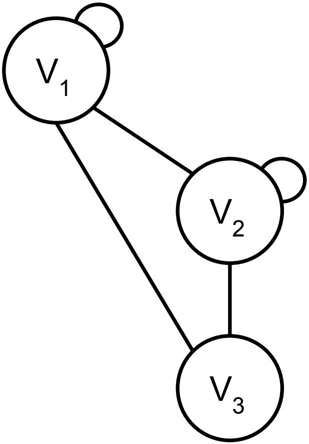
$v_6 \rightarrow (v_6, v_6), (v_6, v_2)/(v_6, v_6), (v_6, v_5)$

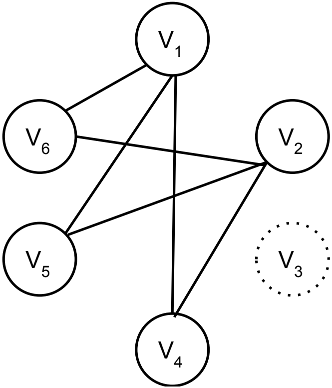
Counter

$3 \rightarrow$ High Hypothesis

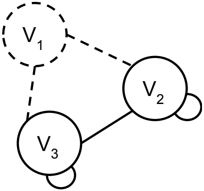
$1 \rightarrow$ Low Hypothesis

Step 6.2 c

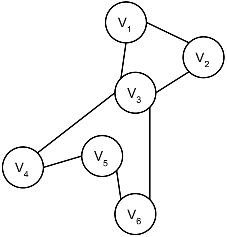




Step 6.2 (d) i & ii: Check for Triangle



- Just like step 4
- Search neighborhoods of v_1, v_2, & v_3
- Safe Grover Search



Remaining Steps: Search from G' to G & Search for Triangles in T & E

- Search for edges left in G' in G
- Search for $\triangle$ in G among $\triangle$ in T
- Search for $\triangle$ in $G \cap E$
- Output $\triangle$ if found, otherwise reject

Outline

Knil's QRAM Model for Quantum Computation

- Quantum computer is a quantum device that is controlled by a classical computer
- Quantum device contains n individually addressable qubit
- Instruction 1: Apply built-in gate U to qubit k, apply gate V to qubits j and k, etc
- Instruction 2: Measure qubit k

Aggregate Gate Counts

From implementing another quantum triangle finding algorithm

- Simple command for gate and qubit count
- Over 30 trillion total gates and 4,676 qubits

Quipper vs QCL

- QCL arguably the oldest "concrete" quantum programming language
- Comparison done implementeing Binary Welded Tree Algorithm
- QCL Version: 17,358 gates and used 58 qubits
- Quipper Version: 1,300 gates and 26 qubits

Outline

Conclusions

- Hardware implementation largest bottleneck
- Already many quantum algorithms
- Quantum programming languages allow more discussion
- Better equipt when quantum device arrives

Questions

Questions?