Upper bounds for query complexity inspired by the Elitzur-Vaidman bomb tester

Cedric Yen-Yu Lin, Han-Hsuan Lin

Center for Theoretical Physics
MIT

QIP 2015
January 12, 2015

arXiv:1410.0932

Overview

1. Bomb Query Complexity
   - Elitzur-Vaidman bomb tester
   - Bomb query complexity $B(f)$
   - Main result: $B(f) = \Theta(Q(f)^2)$
2. Algorithms
   - Introduction: $O(N)$ bomb query algorithm for OR
   - Main theorem 2: constructing q. algorithms from c. ones
   - Applications: graph problems
3. Summary and open problems

Section 1

Bomb Query Complexity

Elitzur-Vaidman Bomb Tester [EV93]

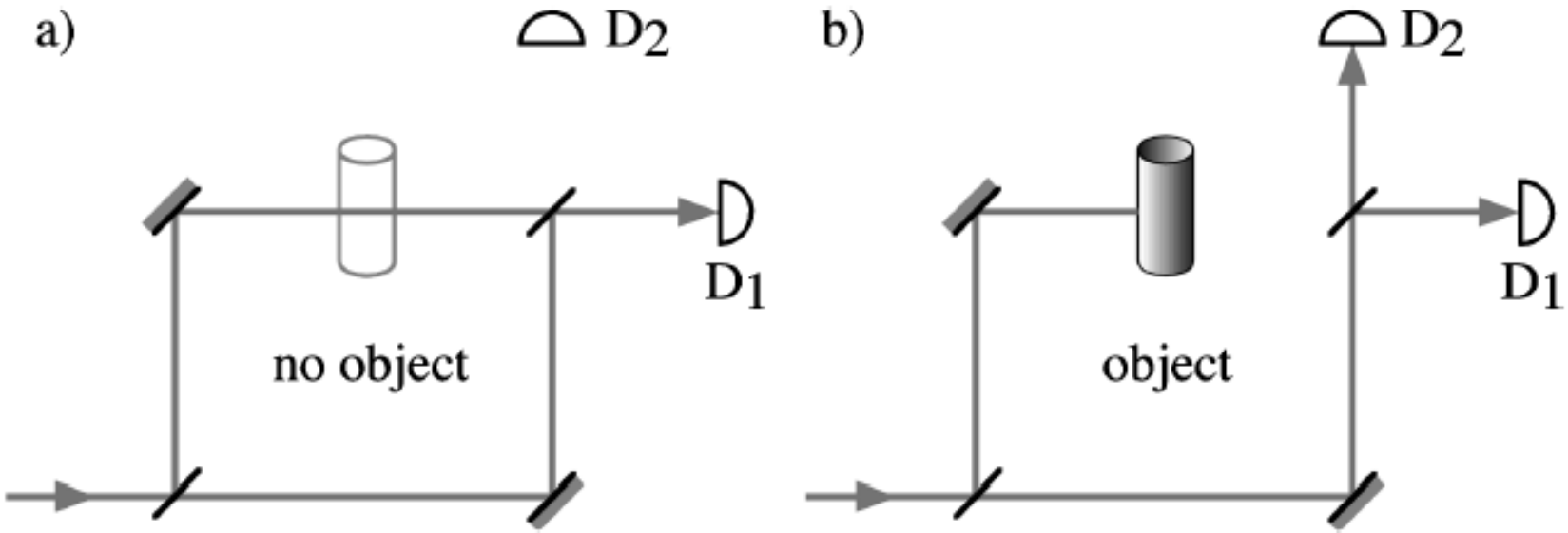
A collection of bombs, some of which are duds

Live: Explodes on contact with photon
Dud: No interaction with photon

Can we tell them apart without blowing ourselves up?

Elitzur-Vaidman Bomb Tester [EV93]

We can put a bomb in an Mach-Zehnder interferometer:

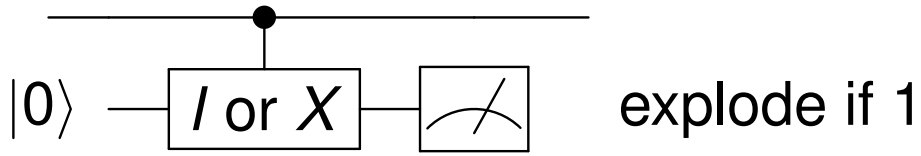


If D2 detects a photon, then we know the bomb is live, even though it has not exploded.

Image source: A. G. White et al., PRA 58, 605 (1998).

EV bomb in circuit model

We can rewrite the Elitzur-Vaidman bomb in the circuit model:

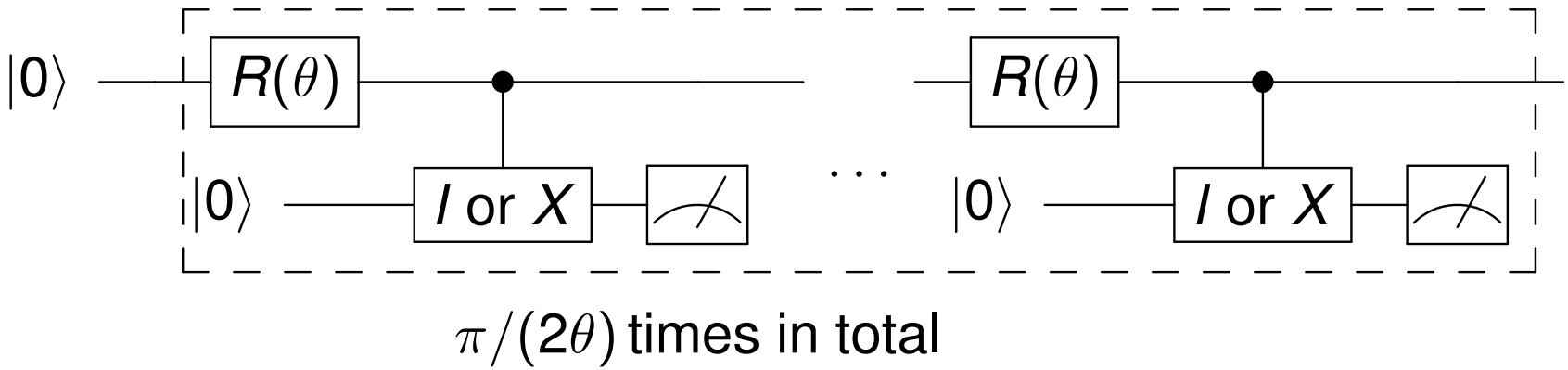


Live bomb: X in the above diagram

Dud: I in the above diagram

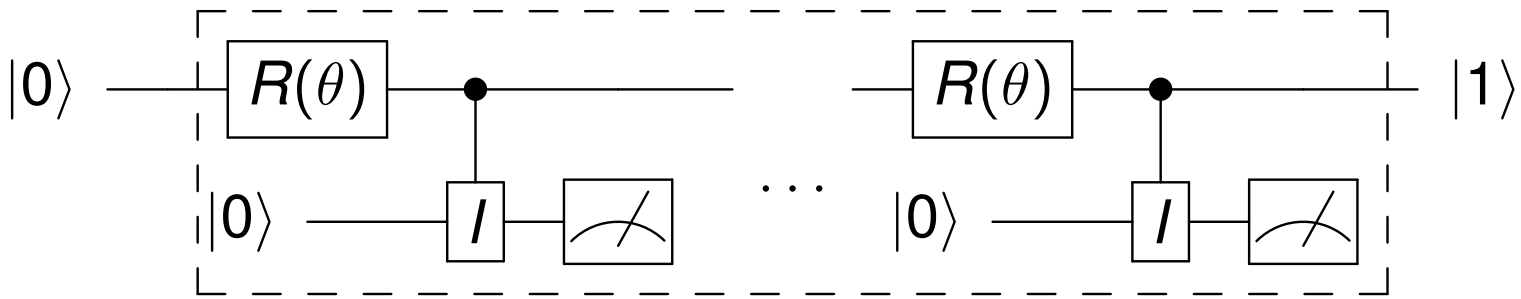
Quantum Zeno Effect [KWH+95]

Let $R(\theta) = \exp(i\theta X) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$.



Quantum Zeno Effect [KWH+95]

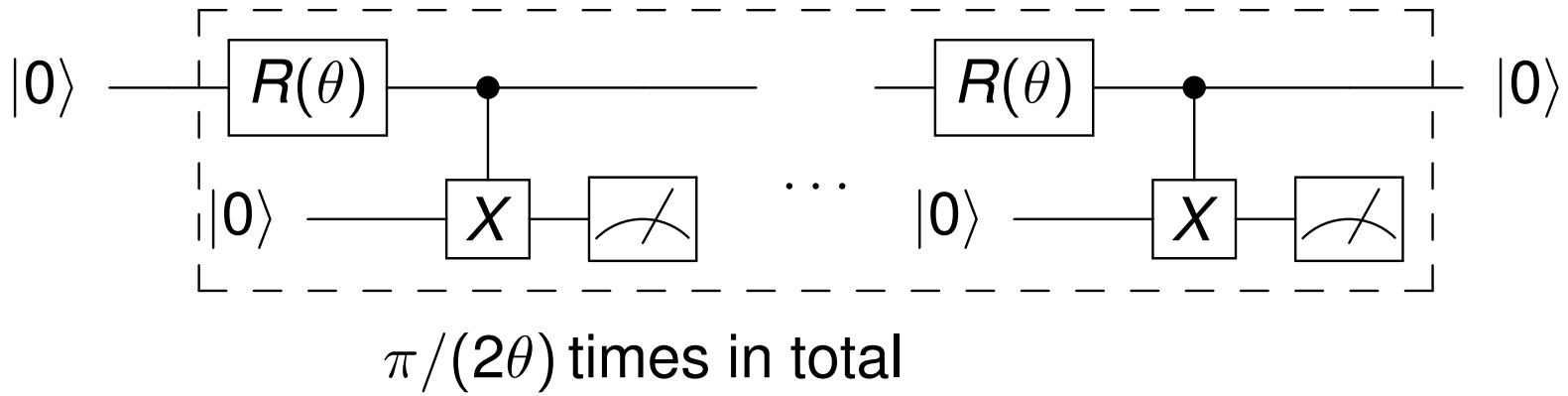
Let $R(\theta) = \exp(i\theta X) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$.



If dud: Ctrl-I does nothing, so $|0\rangle$ gets rotated to $|1\rangle$.

Quantum Zeno Effect [KWH+95]

Let $R(\theta) = \exp(i\theta X) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$.

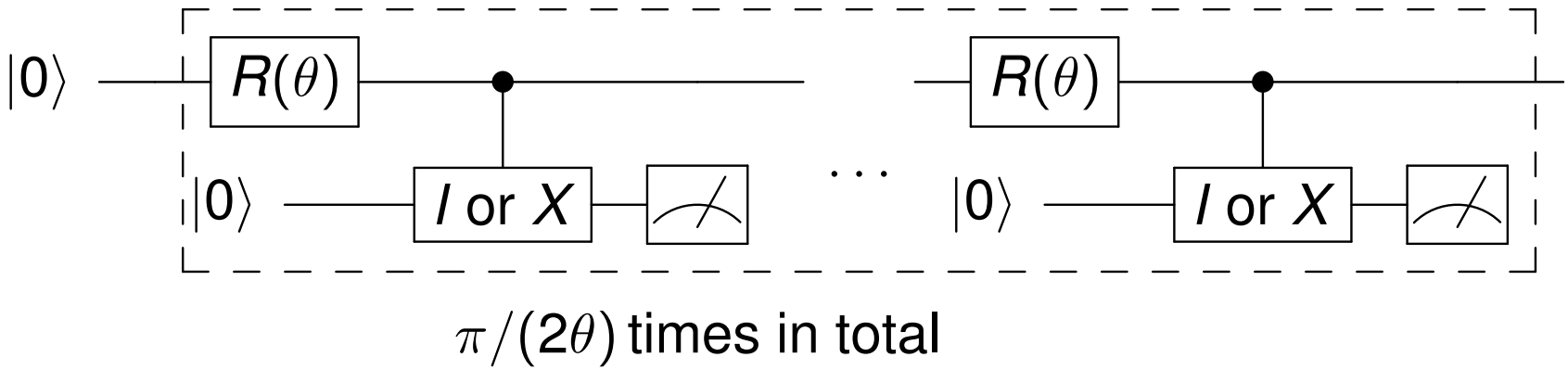


If live: First register is projected back to $|0\rangle$ on each measurement.

Probability of explosion: $\Theta(\theta^2) \times \Theta(1/\theta) = \Theta(\theta)$.

Quantum Zeno Effect [KWH+95]

Let $R(\theta) = \exp(i\theta X) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$.

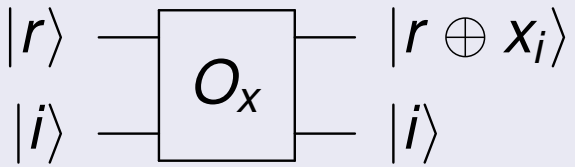


Probability of explosion: $\Theta(\theta)$

Number of queries: $\Theta(1/\theta)$

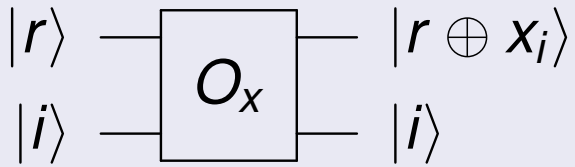
Quantum Query

Quantum query

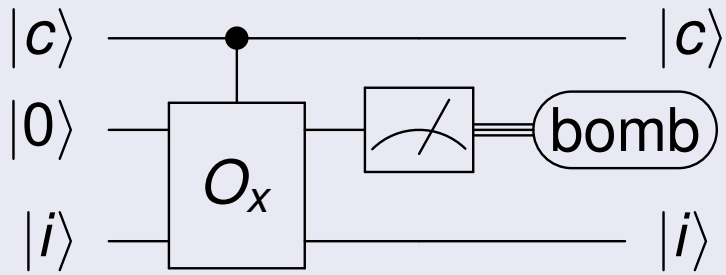


Quantum Query vs Bomb Query

Quantum query

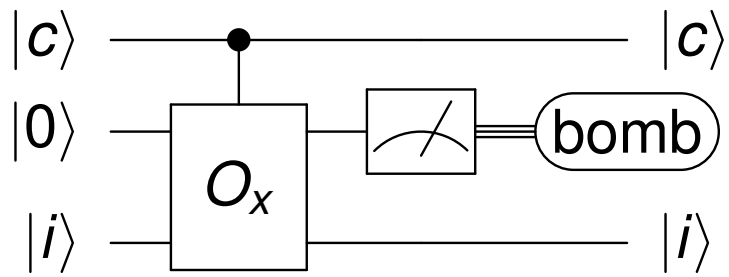


Bomb query



explodes if $c \cdot x_i = 1$

Bomb Query

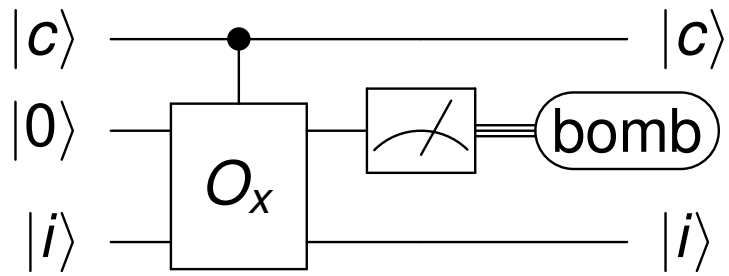


explodes if $c \cdot x_i = 1$

Differences from quantum query:

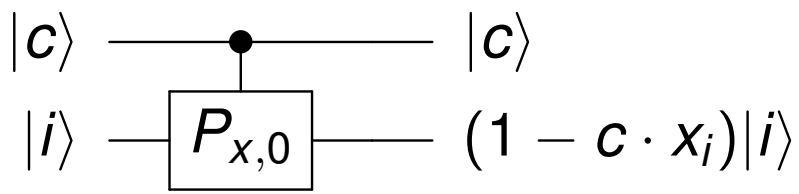
- Extra control register c.
- The record register, where we store the query result, *must* contain 0 as input.
- We *must* measure the query result after each query; if the result is 1, the bomb explodes and the algorithm fails.

Bomb Query



explodes if $c \cdot x_i = 1$

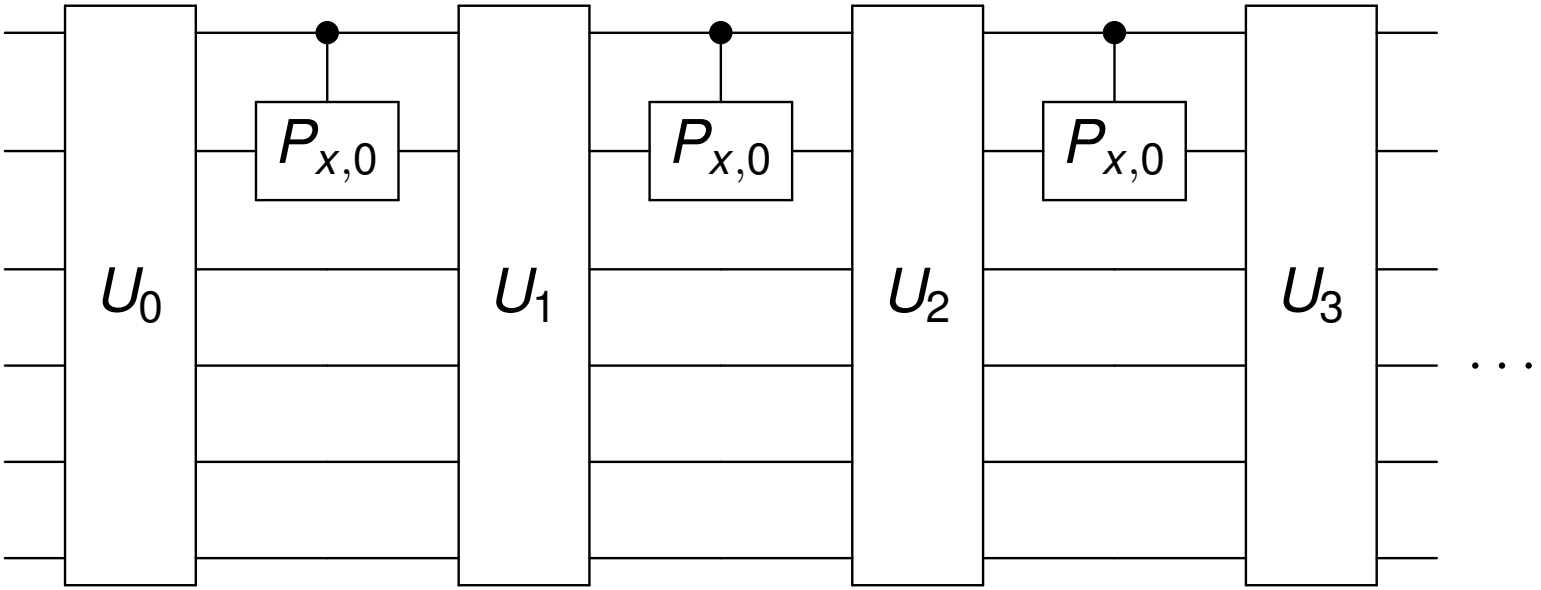
equivalent to



where

$$P_{x,0} = \sum_{x_i=0} |i\rangle\langle i|, \quad \text{Ctrl} - P_{x,0} = I - \sum_{x_i=1} |1,i\rangle\langle 1,i|$$

Bomb Query Complexity



Call the minimum number of bomb queries needed to determine f with bounded error, with probability of explosion $\leq \epsilon$, the bomb query complexity $B_\epsilon(f)$.

Main Theorem

Theorem

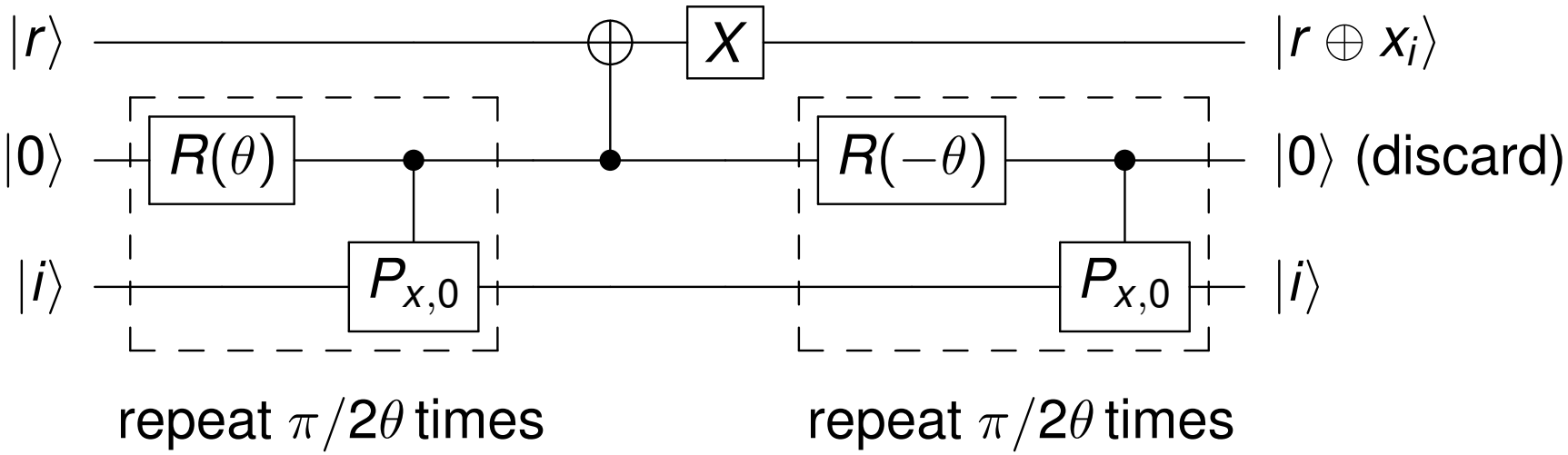
$$B_\epsilon(f) = \Theta(Q(f)^2/\epsilon).$$

Upper bound: Quantum Zeno effect.

Lower bound: Adversary method.

$B_\epsilon(f) = O(Q(f)^2/\epsilon)$: Proof

We can simulate each quantum query using $\Theta(1/\theta)$ bomb queries:



Total probability of explosion: $\Theta(\theta) \cdot Q(f) = \Theta(\epsilon)$, if $\theta = \Theta(\epsilon/Q(f))$.

Total number of bomb queries: $\Theta(1/\theta) \cdot Q(f) = O(Q(f)^2/\epsilon)$.

$B_\epsilon(f) = \Omega(Q(f)^2/\epsilon)$: Proof

The proof uses the general-weight adversary method [HLS07].
We know [Rei09,Rei11,LMR+11] that the general-weight adversary bound tightly characterizes quantum query complexity:
$\mathrm{Adv}^{\pm}(f) = \Theta(Q(f))$.

By modifying the proof of the general-weight adversary bound, we can show that $B_\epsilon(f) = \Omega(\mathrm{Adv}^{\pm}(f)^2/\epsilon)$.

This implies that $B_\epsilon(f) = \Omega(Q(f)^2/\epsilon)$.

Section 2

Algorithms

$O(N)$ Bomb Query Algorithm for OR

There are N bombs, want to check if any are live.

Check each bomb using $\Theta(\epsilon^{-1})$ queries, or $O(N/\epsilon)$ queries in total.

Each live bomb has $\Theta(\epsilon)$ chance of exploding.
Each dud has no chance of exploding.

Since we can stop at the first live bomb, the total chance of failure is only $\Theta(\epsilon)$. Therefore $B_\epsilon(OR) = O(N/\epsilon)$.

Since $B(OR) = O(N)$, $Q(OR) = O(\sqrt{N})$.

This is a nonconstructive proof of the existence of Grover's algorithm!

Can we generalize this further?

Main Theorem 2

Theorem

Suppose there is a classical randomized algorithm $\mathcal{A}$ that computes $f(x)$ using at most T queries. Moreover, suppose there is an algorithm $\mathcal{G}$ that predicts the results of each query $\mathcal{A}$ makes (0 or 1), making at most an expected G mistakes.

Then $B(f) = O(TG)$, and $Q(f) = O(\sqrt{TG})$.

Main Theorem 2

Theorem

Suppose there is a classical randomized algorithm $\mathcal{A}$ that computes $f(x)$ using at most T queries. Moreover, suppose there is an algorithm $\mathcal{G}$ that predicts the results of each query $\mathcal{A}$ makes (0 or 1), making at most an expected G mistakes.

Then $B(f) = O(TG)$, and $Q(f) = O(\sqrt{TG})$.

For example, for OR we have $T = N$ and $G = 1$, so $Q(f) = O(\sqrt{N})$.

Bomb algorithm with $B(f) = O(TG)$

For each classical query, check whether $\mathcal{G}$ correctly predicts the query result of $\mathcal{A}$ using $\Theta(G/\epsilon)$ bomb queries.

If $\mathcal{G}$ guesses incorrectly then the probability of explosion is $O(\epsilon/G)$; otherwise it is zero. (This actually requires defining an equivalent symmetric variant of the bomb query complexity.)

The total probability of explosion is $O(\epsilon/G) \cdot G = O(\epsilon)$, and the number of bomb queries used is $O(G/\epsilon) \cdot T = O(TG/\epsilon)$.

Explicit q. algorithm with $Q(f) = O(\sqrt{TG})$

- Repeat until all queries of $\mathcal{A}$ are determined:
  1. Use $\mathcal{G}$ to predict all remaining queries of $\mathcal{A}$, under assumption it makes no mistakes.
  2. Search for the location d_j of first mistake, using $O(\sqrt{d_j - d_{j-1}})$ quantum queries.
  3. This determines the actual query results up to the d_j-th query that $\mathcal{A}$ would have made.

Kothari's algorithm for oracle identification [Kot14] actually already uses these steps above.

Explicit q. algorithm with $Q(f) = O(\sqrt{TG})$

- Repeat until all queries of $\mathcal{A}$ are determined:
  1. Use $\mathcal{G}$ to predict all remaining queries of $\mathcal{A}$, under assumption it makes no mistakes.
  2. Find the location d_j of first mistake, using $O(\sqrt{d_j - d_{j-1}})$ queries to the black box.
  3. This determines the actual query results up to the d_j-th query that $\mathcal{A}$ would have made.

i	2	5	4	3	12	7	6	9	10
x_i	0	1	1	0	1	1	0	0	1

Explicit q. algorithm with $Q(f) = O(\sqrt{TG})$

- Repeat until all queries of $\mathcal{A}$ are determined:
  1. Use $\mathcal{G}$ to predict all remaining queries of $\mathcal{A}$, under assumption it makes no mistakes.
  2. Find the location d_j of first mistake, using $O(\sqrt{d_j - d_{j-1}})$ queries to the black box.
  3. This determines the actual query results up to the d_j-th query that $\mathcal{A}$ would have made.

i	2	5	4	3	12	7	6	9	10
x_i	0	1	1	1	1	1	0	0	1

Explicit q. algorithm with $Q(f) = O(\sqrt{TG})$

- Repeat until all queries of $\mathcal{A}$ are determined:
  1. Use $\mathcal{G}$ to predict all remaining queries of $\mathcal{A}$, under assumption it makes no mistakes.
  2. Find the location d_j of first mistake, using $O(\sqrt{d_j - d_{j-1}})$ queries to the black box.
  3. This determines the actual query results up to the d_j-th query that $\mathcal{A}$ would have made.

i	2	5	4	3					
x_i	0	1	1	1					

Explicit q. algorithm with $Q(f) = O(\sqrt{TG})$

- Repeat until all queries of $\mathcal{A}$ are determined:
  1. Use $\mathcal{G}$ to predict all remaining queries of $\mathcal{A}$, under assumption it makes no mistakes.
  2. Find the location d_j of first mistake, using $O(\sqrt{d_j - d_{j-1}})$ queries to the black box.
  3. This determines the actual query results up to the d_j-th query that $\mathcal{A}$ would have made.

i	2	5	4	3	10	1	15	7	13
x_i	0	1	1	1	0	0	1	0	1

Explicit q. algorithm with $Q(f) = O(\sqrt{TG})$

- Repeat until all queries of $\mathcal{A}$ are determined:
  1. Use $\mathcal{G}$ to predict all remaining queries of $\mathcal{A}$, under assumption it makes no mistakes.
  2. Find the location d_j of first mistake, using $O(\sqrt{d_j - d_{j-1}})$ queries to the black box.
  3. This determines the actual query results up to the d_j-th query that $\mathcal{A}$ would have made.

i	2	5	4	3	10	1	15	7	13
x_i	0	1	1	1	0	1	1	0	1

Explicit q. algorithm with $Q(f) = O(\sqrt{TG})$

- Repeat until all queries of $\mathcal{A}$ are determined:
  1. Use $\mathcal{G}$ to predict all remaining queries of $\mathcal{A}$, under assumption it makes no mistakes.
  2. Find the location d_j of first mistake, using $O(\sqrt{d_j - d_{j-1}})$ queries to the black box.
  3. This determines the actual query results up to the d_j-th query that $\mathcal{A}$ would have made.

i	2	5	4	3	10	1			
x_i	0	1	1	1	0	1			

Explicit q. algorithm with $Q(f) = O(\sqrt{TG})$

- Repeat until all queries of $\mathcal{A}$ are determined:
  1. Use $\mathcal{G}$ to predict all remaining queries of $\mathcal{A}$, under assumption it makes no mistakes.
  2. Find the location d_j of first mistake, using $O(\sqrt{d_j - d_{j-1}})$ queries to the black box.
  3. This determines the actual query results up to the d_j-th query that $\mathcal{A}$ would have made.

Query complexity: $O(G) \cdot O(\sqrt{T/G}) = O(\sqrt{TG})$.
It looks like error reduction may give extra log factors, but [Kot14] showed that the log factors can be removed using span programs.

Applications: Breadth First Search

Problem: Unweighted Single-Source Shortest Paths

Given the adjacency matrix of an unweighted graph as a black box, find the distances from a vertex s to all other vertices.

Classical algorithm: *Breadth First Search*.

Breadth First Search

1. Initialize an array *dist* that will hold the distances of the vertices from s. Set $dist[s] := 0$, and $dist[v] := \infty$ for $v \neq s$.
2. For $d = 1, \cdots, n-1$:
   1. For all vertices v with $dist[v] = d - 1$, query its outgoing edges (v, w) to all vertices w whose distance we don't know ($dist[w] = \infty$). If (v, w) is an edge, set $dist[w] := d$.

BFS: Quantum Query Complexity

Breadth First Search

1. Initialize an array *dist* that will hold the distances of the vertices from s. Set $dist[s] := 0$, and $dist[v] := \infty$ for $v \neq s$.
2. For $d = 1, \cdots, n-1$:
   1. For all vertices v with $dist[v] = d-1$, query its outgoing edges (v, w) to all vertices w whose distance we don't know ($dist[w] = \infty$). If (v, w) is an edge, set $dist[w] := d$.

Worst case query complexity is $T = O(n^2)$, where n is no. of vertices. If we guess that each queried pair (v, w) is not an edge, then we make at most $G = n - 1$ mistakes, since each vertex is only discovered once.

$Q(uSSSP) = O(\sqrt{TG}) = O(n^{3/2})$, matches lower bound of [DHH+04].

Applications: k-Source Shortest Paths

What if we instead want the distances from k different sources?

Problem: Unweighted k-Source Shortest Paths

Given the adjacency matrix of an unweighted graph as a black box, find the distances from vertices $s_1, \cdots, s_k$ to all other vertices.

Classical: Run BFS k times.

Quantum: $G = k(n-1)$, but $T = O(n^2)$ instead of $O(kn^2)$.
Therefore $Q(\text{k}SSP) = O(k^{1/2}n^{3/2})$.

Dhariwal and Mayar showed tight lower bound;
available on S. Aaronson's blog, Dec. 26, 2014:
`http://www.scottaaronson.com/blog/?p=2109`

Applications: Maximum Bipartite Matching

Problem: Maximum Bipartite Matching

A *matching* in an undirected graph is a set of edges that do not share vertices. Given a bipartite graph, find a matching with the maximum possible number of edges.

Classical algorithm: Hopcroft-Karp algorithm.
Essentially proceeds by using $O(\sqrt{n})$ rounds of BFS and modified DFS (depth-first search).

Quantum: $G = O(\sqrt{n} \times n) = O(n^{3/2})$, and $T = O(n^2)$ (not $O(n^{2.5})$).
Therefore $Q(MBM) = O(n^{7/4})$. First nontrivial upper bound!

Summary

- Inspired by the EV bomb tester, we defined the notion of *bomb query complexity*, and showed the relation $B(f) = \Theta(Q(f)^2)$.

- Bomb query complexity further lead us to a general construction of quantum query algorithms from classical algorithms, giving us an $O(n^{1.75})$ quantum query algorithm for maximum bipartite matching.

Open Questions

- Can we relate G, the number of wrong guesses, to classical measures of query complexity (e.g. certificate, sensitivity...)?
- Time complexity of algorithms?
- Algorithms for adjacency list model?
- Other problems e.g. matching for general graphs?
- Relationship between $R(f)$ and $B(f)$?

Relationship between $R(f)$ and $B(f)$?

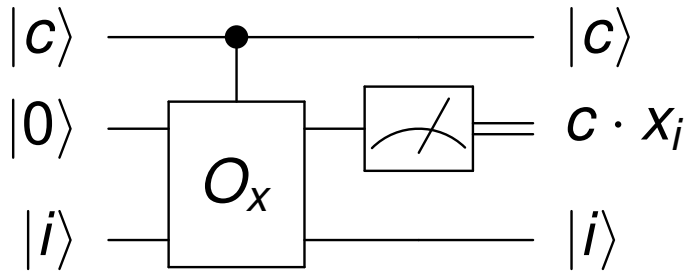
For total functions the largest known separation between $R(f)$ and $Q(f)$ is quadratic (for the OR function).
It is conjectured this is the extreme case, $R(f) = O(Q(f)^2)$.

We know that $B(f) = \Theta(Q(f)^2)$.
Therefore the conjecture is equivalent to $R(f) = O(B(f))$.

We give some motivation for why this conjecture might be true...

Projective Query Complexity, $P(f)$

Aaronson (unpublished, 2002) considered allowing access to the black box only with the following:



We call the number of queries required the *projective query complexity*, $P(f)$. Note the algorithm does *not* end on measuring a 1.

Straightforwardly $Q(f) \leq P(f) \leq R(f)$ and $P(f) \leq B(f)$.
Regev and Schiff [RS08]: $P(OR) = \Omega(N)$.

Open question: Does $P(f) = \Theta(R(f))$ for all total functions?
If this is true, implies $R(f) = O(B(f)) = O(Q(f)^2)$.

Thank You!