

# On Finding Sensitivity of Quantum and Classical Gates

Debajyoti Bera, Steve Homer

## Abstract

We consider a fault model of Boolean gates, both classical and quantum, where some of the inputs may not be connected to the actual gate hardware. This model is somewhat similar to the stuck-at model which is a very popular model in testing Boolean circuits. We consider the problem of detecting such faults; the detection algorithm can query the faulty gate and its complexity is the number of such queries. This problem is related to determining the sensitivity of Boolean functions.

We show how quantum parallelism can be used to detect such faults. Specifically, we show that a quantum algorithm can detect such faults more efficiently than a classical algorithm for a Parity gate and an AND gate. We give explicit constructions of quantum detector algorithms and show lower bounds for classical algorithms. We show that the model for detecting such faults is similar to algebraic decision trees and extend some known results from quantum query complexity to prove some of our results.

## 1 Introduction

Like all hardware, quantum gates and circuits are prone to faults. In fact, implementations of quantum gates are extremely sensitive to various physical factors and often have errors and faults of various kinds. There has been extensive research on fault-tolerant quantum computing and detecting/correcting errors in a quantum computation. Our motivation is in a similar spirit but in a tangential direction. We are interested in exploiting *quantum parallelism* to identify certain types of errors in specific gates and circuits more efficiently than is possible using classical techniques. Our focus is not on arbitrary errors or arbitrary gates, but rather on a certain type of error for several specific, simple quantum operations. However, the problem we tackle has a close resemblance to the fault models used for the fault testing of classical Boolean circuits. We elaborate on this in section 3.

We will consider  $n$ -bit Boolean gates. Such a gate  $G$  computes a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ . Now usually  $f$  depends on all  $n$  of its inputs. However for this paper consider functions  $f$  which may be independent of some of these inputs, for example, if the connection from some input to the gates output is broken. We would like to determine which inputs it depends on or at least if there is some *faulty input* which  $f$  does not depend on at all. Here  $G$  could be any single-valued Boolean gate, even a full Boolean circuit with one output; we are merely interested in the input-output behavior of the gate.

Assume we are given a few samples of a gate which we want to test.<sup>1</sup> How easy or hard it is to find out whether  $G$  is faulty or not (and if possible, which inputs are not connected)? We want to investigate this question by considering classical and quantum algorithms which are given copies of the faulty gates which they can query. We measure the number of *queries* to

---

<sup>1</sup>Yes, it is usually unreasonable to assume multiple exact copies of the same faulty gate. Section 3 tries to explain some cases where this might be possible.

the faulty gates needed to determine the faulty input(s). Notice that this problem is similar to algebraic query complexity, where, instead of using a bit of the input, the nodes of a decision tree are labeled by the output of an algebraic function of the input. Using similar ideas we extend existing quantum query results to obtain upper and lower bounds for detecting faults in AND and other similar gates.

We claim that quantum algorithms can determine these faults more efficiently than classical ones. We will illustrate this for several common and central types of gates, the Parity gate and the AND gate, and other gates with similar properties. For the quantum case, we will construct quantum circuits which will be given the usual quantum analogue of these gates. The output of the checking circuits will indicate one or all the faults of the gate in question. Since a classical circuit usually has a single output whereas a quantum circuit has multiple outputs, for the classical case we show the lower bound for a simpler version of the problem: Detecting whether there is *any* faulty input.

## 2 Problem statement

We first define what we mean by a faulty classical gate. The faulty quantum gates used in this paper will be their reversible extensions <sup>2</sup>.

**Definition 2.1 (Faulty gate)** Consider a gate  $G$  computing a Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ . We say that the gate has at least  $s$  faults if there exists a set of faulty inputs  $\mathcal{S} \subseteq \{1, \dots, n\}$  of size  $s$  such that  $\forall (x_1, \dots, x_n) \in \{0, 1\}^n \forall (x'_1, \dots, x'_n) \in \{0, 1\}^n, f(x_1, \dots, x_n) = f(x'_1, \dots, x'_n)$  whenever  $\forall i \notin \mathcal{S}, x_i = x'_i$  (the value of  $f$  depends only on the non-faulty inputs irrespective of the inputs from  $\mathcal{S}$ ).

We need to be careful in dealing with such faulty gates since the functions computed by the faulty gate and the correct gate might be of a completely different nature. Certain functions do not have a natural restriction when defined only on a subset of its inputs; e.g., the function  $f(x_1, x_2, x_3) = x_1 \oplus x_2 x_3$  has no obvious unique restriction to only the first and the second inputs. On the other hand, for the function  $f(x_1, x_2, x_3) = x_1 \vee x_2 \vee x_3$ , a natural way to define its faulty version could be  $f(x_1, x_2, x_3) = x_1 \vee x_2$  (assuming the third input is faulty). In reality the faulty function in both the cases could be defined in a non-trivial way (e.g. depending on the actual hardware implementation) <sup>3</sup>. For the two gates we study in this paper, it is natural to assume that if some input wires are not connected, then the gate computes the natural restriction of the function on the rest of its inputs.

For a given gate, consider the largest possible set of faulty inputs; in the rest of this paper, we will denote the maximal set by  $\mathcal{S}$  and its size by  $s$ . Note that, by our definition, any subset of  $\mathcal{S}$  is a valid set of faulty inputs to the gate. We will use  $e$  to denote an elementary vector. Let  $e_i$  denote the  $i$ -th elementary  $n$ -bit string (only the  $i$ -th bit is set to 1, other  $n - 1$  bits are set to 0) and  $\bar{f}_i = \bar{e}_i$  (bit-wise complement). To allow our quantum circuits to directly output the faulty inputs in an easy manner, we will interchangeably use another representation of  $\mathcal{S}$  using elementary vectors:  $\{e_i \mid i \in \mathcal{S}\}$ . When represented using elementary vectors, if  $e_i \in \mathcal{S}$ , it will mean the  $i$ -th input is faulty. We will use  $\sigma \in \{0, 1\}^n$  to denote the characteristic vector of  $\mathcal{S}$ :  $\sigma_i = 1$  if and only if  $e_i \in \mathcal{S}$ . We will use the following notation in this paper:  $\exists^l x$  to indicate the existence of  $l$  unique values of  $x$ .

Our goal is to construct oracle circuits using these faulty gates to detect faults. The cost of the detection is measured by the number of faulty gates queried. The detecting circuits can

<sup>2</sup>A reversible way to compute a function  $f$  by a quantum gate is  $|x_1, \dots, x_n, y\rangle \rightarrow |x_1, \dots, x_n, y \oplus f(x_1, \dots, x_n)\rangle$

<sup>3</sup>Usually faults are not so well-behaved to be consistent across gates. But in this rather theoretical study, we will assume that our faulty gates are identically faulty.

output different kind of information:

1. A single bit Boolean output denoting if there is any fault or not (whether  $\mathcal{S} \stackrel{?}{=} \emptyset$ ).
2. An  $n$ -bit output representing any of the faulty inputs (output any elementary vector  $e \in \mathcal{S}$ ).
3. The output which indicates the maximal set of faulty inputs (that is, the  $n$ -bit binary string  $\sigma$ ).

Note that, 1 can be reduced to 2 and 2 can be reduced to 3. Since classical gates have only one output, a classical circuit with  $n$ -bit output will inevitably require  $n$  gates. Thus, we show lower bounds for classical circuits of type 1. The quantum circuits we construct are of type 2 or 3. For background on quantum circuits, see [BGH07].

The output can be deterministic or probabilistic; we consider both cases in the paper. The two functions we will extensively deal with in this paper are the parity function  $\oplus(x_1, \dots, x_n) = x_1 \oplus \dots \oplus x_n$ , for which we consider deterministic detecting circuit and the AND function  $AND(x_1, \dots, x_n) = x_1 \wedge \dots \wedge x_n$ , for which we consider probabilistic detecting circuit.

## 2.1 Relation to sensitivity

Our definition of faulty inputs has a close similarity to the sensitivity of Boolean functions [BW02]. Let  $x$  denote an  $n$ -bit Boolean string and  $x^i$  is obtained from  $x$  by flipping  $x_i$ .

**Definition 2.2 (Sensitivity)** *An  $n$ -bit Boolean function  $f$  is sensitive to the  $i$ -th input bit on input  $x$  if  $f(x) \neq f(x^i)$ . The sensitivity of  $f$  on  $x$ ,  $s_x(f)$ , is the number of bits to which  $f$  is sensitive on  $x$ .*

*$f$  is sensitive to  $i$  if there is some  $x$  such that  $f$  is sensitive to  $i$  on  $x$ . The sensitivity of  $f$  is  $s(f) = \max_x s_x(f)$ .*

*The insensitivity of  $f$ , denoted by  $\bar{s}(f)$ , is the maximum number of  $i$  such that  $f$  is not sensitive to  $i$ .*

So,  $f$  is *insensitive* to the  $i$ -th input if  $f$  is insensitive to  $i$  on  $x$  for all  $x$ . According to our definition in this paper, if there is a gate  $G_f$  computing  $f$ , then  $f$  being insensitive to the  $i$ -th input is equivalent to the  $i$ -th input to  $G$  being faulty. From this perspective, in this paper we are trying to determine the input insensitivity of  $f$  using one or more copies of  $G_f$ . Note that for any  $n$ -bit function  $f$ ,  $s(f) \leq n - \bar{s}(f)$ , so the insensitivity gives us an upper bound on the sensitivity of  $f$ .

Suppose we are allowed to query  $n$  bit Boolean gate  $G_f$  to determine the input sensitivity of the Boolean function  $f$ . Obviously we need  $\Omega(n)$  queries. The hardest case is when we have no a priori knowledge about  $f$ ; it can be one of the  $2^{2^n}$  possible functions. We can determine if  $f$  depends on any particular bit using  $O(2^n)$  queries. We believe this is a tight upper bound.

However, we can do better if we know the possible ways  $f$  might behave when it is not reading all its inputs. Suppose  $f$  is insensitive to the input bits in the subset  $s \subseteq \{1, \dots, n\}$ . For certain functions, we might have information how  $f$  might behave given such an  $s$  e.g. if  $f$  computes the AND function, then we might be told that  $f$  computes the bit-wise AND of the inputs in  $\bar{s}$ . In such cases,  $f$  belongs to a set of  $2^n$  possible functions, one for each possible  $s$ . We consider two such functions here, the AND and the Parity function and show that it is possible to determine the sensitivity using  $O(n)$  queries. Furthermore, we show that for some functions we can use quantum techniques to determine the sensitivity more efficiently than is possible using classical algorithms.

## 2.2 Organization

The rest of the paper is organized in the following manner. In the following section, we present some related work. We discuss our results on Parity gates in section 4 and our results on AND

gates in section 5. For both the gates we present a quantum algorithm to detect faults and prove that it is optimal. We also prove that the quantum algorithm is provably better than any classical algorithm for detecting faults.

The construction and lower bounds for the AND gate will use techniques from query complexity. The proofs for the quantum case require extending a few quantum query complexity results to work with generalised oracles, which we discuss first in section 5.1 and then describe the actual quantum upper bound and lower bound in subsection 5.2 and subsection 5.3 respectively. For proving the classical lower bound, it will be convenient to introduce a related problem of finding a marked vertex in a directed acyclic graph using path queries. We will describe the graph problem in subsection 5.4 and then proceed to give the lower bound in subsection 5.5.

### 3 Related work

Boolean circuit testing is a well-established discipline within semiconductor circuit design. Over time circuits have increased in complexity and decreased in size, and the testing regimes have gotten more difficult and expensive. As they become *really* small quantum effects start to appear and need to be addressed. Recently testing methods have been suggested which use quantum circuits designed to find and analyze faults in circuits which compute classical Boolean functions. Exploiting the advantages of quantum techniques, these quantum circuits can be tested more easily and efficiently than their classical counterparts [CTK08].

Real world hardware circuits can fail in a multitude of ways. To deal with the variety of faults, the circuit testing research community has created different logical fault models. One of the most widely used logical fault models to represent faulty interconnections is the *stuck-at* model. In this model, one or more wires is assumed to be stuck at a fixed logic value, either 0 or 1 (respectively known as stuck-at-0 and stuck-at-1 model)<sup>4</sup>. The function computed by the corresponding gate then does not depend on the input carried by the faulty wire. This model captures several physical defects, e.g. a short-circuit between the wire and the ground, a disconnected wire or some internal fault which always sets the wire to a constant value.

Our problem is similar to the stuck-at fault model; we are dealing with circuits in which some wire may not be connected to its gate. This is similar to the situation where the wire maybe assumed to be stuck at some fixed value. However, our approach differs from the standard technique of detecting such faults (see e.g. [PBL05]); instead of generating a test set of input vectors whose output is then matched with the expected outputs, we design a circuit/algorithm using one or more faulty gates and expect to obtain information about the faulty inputs.

There is yet another difference with the classical case. An open wire in a classical reversible circuit can be modeled using multiple stuck-at faults. It has been shown that multiple stuck-at faults are functionally identical to single stuck-at faults for classical reversible circuits [PHM04]. Though the technique described in [CTK08] can detect stuck-at faults in quantum Boolean circuits, it cannot detect open wires. This is one piece of evidence that classical fault models might not directly work for quantum circuits.

Arvind et al. looked at a similar problem in [VA98] where they studied program checking (as defined by Blum) using  $AC^0$  circuits as checkers. Like us, they allowed their checkers to make queries to the program and defined the cost of checking as the number of queries. Note that they considered programs and not fixed input gates and they studied deterministic and randomized checkers for  $P$ -complete and  $NC^1$ -complete problems.

---

<sup>4</sup><http://www.pld.ttu.ee/diagnostika/theory/fault.html>

## 4 Parity gate

For this section, we will consider *faulty parity gates* defined by the function  $\oplus_{\mathcal{S}}(x_1, \dots, x_n) = \bigoplus_{i \notin \mathcal{S}} x_i$  where  $\mathcal{S}$  is the set of faulty inputs. It will be useful to adopt a convention that  $\oplus_{\{\}}(x_1, \dots, x_n) = 0$ . The corresponding quantum gate  $U$  is the canonical way to reversibly compute the parity function:

$$U|x_1, \dots, x_n, y\rangle = |x_1, \dots, x_n, y \oplus (\oplus_{\mathcal{S}}(x_1, \dots, x_n))\rangle$$

There is an obvious  $O(n)$  algorithm to find all the faults of a parity gate: Query the gate on  $e_1, \dots, e_n$ . The  $i$ th input is faulty if the output for  $e_i$  is 0. In contrast, we will present a quantum circuit which will output all the faults of such a quantum parity gate using only one query. We will also show that this is significantly better than what classical circuits can do; a classical circuit requires  $\Omega(n)$  queries even to detect if there is a faulty input or not. We will also show that even a bounded error algorithm cannot do any better than the trivial algorithm presented above.

### 4.1 Quantum detection circuit

Quantum parity gates could be checked by a depth 3, linear size quantum circuit using only one parity gate.

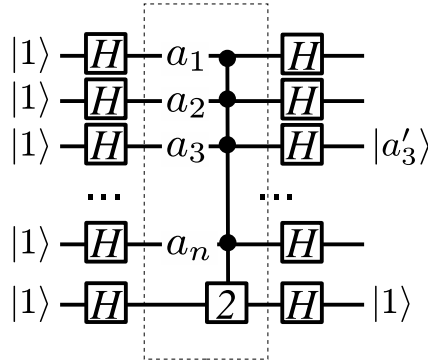


Figure 1: Circuit to check a faulty parity gate.  $a'_i$  is 1 if  $i \in \mathcal{S}$ , 0 otherwise.

The circuit to check a parity gate is given in Figure 1. A parity gate with Hadamard gates on all inputs on both the sides was shown equivalent to a quantum fanout gate<sup>5</sup> where the target qubit of the parity gate becomes the control qubit of the fanout gate [Moo99]. Hence, if input  $a_i$  is not faulty,  $|a'_i\rangle = |0\rangle$ . The qubits in the faulty input will be unaffected; for these qubits,  $|a'_i\rangle = |1\rangle$ . Thus, the output of the circuit will be exactly  $\sigma$ , the characteristic vector of the fault set.

**Theorem 4.1** *For a (possibly) faulty quantum parity gate  $\oplus$  on  $n$  inputs, there is a quantum circuit on  $n$  inputs, with linear size, has depth 3 and makes one call to the faulty gate, whose output qubit  $i$  measures 1 if the  $i$ -th input is faulty and 0 otherwise.*

<sup>5</sup>The fanout gate is a quantum analog of the classical fanout operation. It copies a standard basis state to multiple qubits:  $F_n|c, t_1, \dots, t_n\rangle = |c, c \oplus t_1, \dots, c \oplus t_n\rangle$

## 4.2 Classical deterministic lower bound

Here we show that even detecting if a given parity gate is faulty or not, requires  $n$  queries. Consider the decision tree of a detecting circuit,  $C$ , which makes  $k$  queries. We will construct two parity gates  $G_S$  and  $G_T$  with fault sets  $S \neq \emptyset$  and  $T = \emptyset$  respectively and show that for  $C$  to distinguish between  $G_S$  and  $G_T$ ,  $k \geq n$ .

We will construct  $S$  inductively based on the decision tree queries. Let  $S_j$  be the fault set we construct after the  $j$ -th query by the decision tree (so  $S = S_k$ ). We start with  $S_0 = \{1, \dots, n\}$  (all inputs are faulty). Now consider the  $j + 1$ -th query. Say  $C$  queries  $G_*(x_1, \dots, x_n)$ . Let  $b = \bigoplus_i x_i$ ; note that,  $G_T(x_1, \dots, x_n) = b$  and  $G_S(x_1, \dots, x_n) = \bigoplus_{i \notin S} x_i$ .

**Case:**  $G_{S_j}(x_1, \dots, x_n) = b$ . Set  $S_{j+1} = S_j$ . So, both  $G_T$  and  $G_{S_{j+1}}$  are consistent after  $1, \dots, j + 1$ -th queries.

**Case:**  $G_{S_j}(x_1, \dots, x_n) = \bar{b}$ . Then,  $\exists c \in S_j, x_c = 1$ . Set  $S_{j+1} = S_j \setminus \{c\}$ . Again, both  $G_T$  and  $G_{S_{j+1}}$  are consistent after  $1, \dots, j + 1$ -th queries.

So, after  $k$  queries,  $|S_k| \geq n - k$ . If  $k < n$ , then  $C$  cannot distinguish between  $G_S$  and  $G_T$ .

**Theorem 4.2** *Consider any classical Boolean circuit  $C$  which can also make queries to a (possibly) faulty  $\bigoplus$  gate on  $n$ -inputs. Then there are two such gates  $\bigoplus_1$  and  $\bigoplus_2$  which  $C$  cannot distinguish without making  $n$  queries.*

## 4.3 Classical probabilistic lower bound

We showed how to construct a quantum checker that deterministically finds all faults of a parity gate using only 1 query. We claimed that this is order of magnitude better than what a classical checker can do and as evidence, argued that any deterministic classical checker using less than  $n$  queries is incapable of finding all the faults of a parity gate. However it is conceivable that a probabilistic classical checker might be able to perform better than a classical checker. In this section, we give a negative evidence to this possibility; we show that any checker requires  $\Omega(n)$  queries even with two-sided error.

For the lower bound, we relate the fault detection of a parity gate to a property testing problem discussed by Buhrman et. al. in [BFNR08].

Let  $y \cdot i$  denote the inner product of two vectors  $y, i \in \mathcal{F}_2^n$  and for any  $A \subseteq \{0, 1\}^n$ , let  $P_A = \{x : \exists y \in A \forall i \in \{0, 1\}^n, x_i = y \cdot i\}$  denote the set of strings which represent all possible oracle replies for any string in  $A$ . Then  $P_A$  has an  $\epsilon$ -tester with  $q$  queries if there exists a query algorithm  $M$  which on input a  $2^n$ -bit  $x$ , makes at most  $q$  queries to obtain individual bits of  $x$ , such that

1. If  $x \in P_A$ , then  $\Pr(M \text{ accepts}) \geq 2/3$
2. Otherwise, if  $|\{z \in P_A : x_i \neq z_i\}| \geq \epsilon 2^n$ , then  $\Pr(M \text{ rejects}) \leq 1/3$

**Theorem 4.3 (Lemma 2, [BFNR08])** *For most  $A \subseteq \{0, 1\}^n$ , any  $\epsilon$ -tester (for  $\epsilon \leq 1/2$ ) for  $P_A$  requires  $\Omega(n)$  queries.*

Consider a set of faulty gates and let  $A$  denote the set of their characteristic vectors. Note that, for any faulty parity gate with characteristic vector  $\sigma \in \mathcal{F}_2^n$ , the output of the gate on input  $i \in \mathcal{F}_2^n$  is given by  $\sigma \cdot i$ ; so, querying the gate  $q$  times is equivalent to asking  $q$  bits of  $2^n$ -bit  $x$  where  $x_i = \sigma \cdot i$ . Also note that, for any  $\sigma, \sigma'$ , there are exactly  $2^n/2$  different  $i$  such that  $\sigma \cdot i \neq \sigma'_i$ . Since any classical algorithm to detect all faults should be able to classify if a given faulty gate belongs to  $A$  or not, we get the following lower bound as a corollary to the theorem above.

**Corollary 4.4** *Any bounded error classical algorithm for detecting all faults of a faulty parity gate requires  $\Omega(n)$  queries.*

## 5 AND gate

Next, we will consider faulty gates computing the AND function  $AND_{\mathcal{S}}(x_1, \dots, x_n) = \bigwedge_{i \notin \mathcal{S}} x_i$ . As before, we adopt the convention that  $AND_{\emptyset} = 0$ . The corresponding quantum gate is basically the unbounded Toffoli gate

$$U|x_1, \dots, x_n, y\rangle = |x_1, \dots, x_n, y \oplus (AND_{\mathcal{S}}(x_1, \dots, x_n))\rangle$$

Unlike Parity gate, for this gate, we will show our results for probabilistic algorithms. We will present a  $O(\sqrt{n})$ -query quantum algorithm to output one of the faults. We will also prove that our quantum detection algorithm makes optimum (up to a constant) number of calls to the faulty gate. We will then show a lower bound of  $\Omega(n)$  for classically detecting whether any input of an AND gate is faulty with high probability. Similar to the parity gate, this bound is tight for a classical algorithm: The  $i$ th input is faulty if the output of the gate on input  $f_i$  is 1.

### 5.1 Algebraic query complexity

The problem of detecting faulty inputs of gates by querying the gates can be approached from the algebraic query complexity perspective. Usually the nodes of a decision tree is labeled by  $x_i$ . Generalizing it, the detection algorithms can be modeled as decision trees whose nodes are labeled by  $g(x, i)$  where  $g$  is a Boolean function. For instance, the nodes in the decision tree for a parity gate  $\oplus_{\mathcal{S}}$  is labeled by  $g(\sigma, x) = \bigoplus_i (\sigma_i \cdot x_i)$  where  $\sigma$  is the characteristic vector for  $\mathcal{S}$ . Similarly, the nodes in the decision tree for an AND gate  $AND_{\mathcal{S}}$  is labeled by  $\bigwedge_i (\bar{\sigma}_i \vee x_i)$ .

Some of the quantum query complexity results can be shown to also work for quantum algebraic decision trees. For example, Grover's unordered search [Gro96] algorithm can be extended to include algebraic functions at the nodes.

**Theorem 5.1 (Unordered search with generalized oracle)** *Let  $g : \{0, 1\}^n \times Y \rightarrow \{0, 1\}$  be a Boolean function, where the second input to  $g$  is from a subset  $Y \subseteq \{0, 1\}^k$  for some  $k$ . Let  $x$  be an  $n$ -bit binary input,  $x = x_1 \dots x_n$ , and  $O_x$  be an oracle gate to compute  $g$ , so  $g : O_x|i, b\rangle = |i, b \oplus g(x, i)\rangle$ .*

*Then there is a quantum oracle circuit  $C$ , with  $O(\sqrt{|Y|})$  oracle gates, such that the measurement of the output qubit of  $C|0^n, w\rangle$  ( $w$  denotes the oracle workspace qubits) is any  $\hat{i} \in Y$  such that  $g(x, \hat{i}) = 1$  with probability  $O(1)$ .*

The proof is a simple generalization of Grover's original proof [Gro96]. For the sake of completeness, we include a proof in appendix A. There are several subtleties in using the unbounded search technique e.g. for a deterministic running time, the number of solutions should to be known beforehand. There are many variations of the original technique to deal with such issues. All these techniques also work with the generalized oracle.

Similarly, we can extend the quantum adversary technique to prove lower bounds against such decision trees. We will use a special form of the theorem 1 in [Amb00] in this paper which we state here. The proof is a simple modification of the original proof, which we include in appendix B.

**Theorem 5.2** *Let  $g : \{0, 1\}^n \times Z \rightarrow \{0, 1\}$  be a Boolean function, where  $Z \subseteq \{0, 1\}^k$  for some  $k$ . Let  $O_x$  be an oracle gate to compute  $g : O_x|i, b\rangle = |i, b \oplus g(x, i)\rangle$ . Let  $f(x_1, \dots, x_n)$  be a function and  $X, Y \subseteq \{0, 1\}^n$  be two sets of inputs such that  $f(x) \neq f(y)$  if  $x \in X, y \in Y$ . Let  $R \subseteq X \times Y$  such that*

1.  $\forall x \in X, \exists^m y \in Y, (x, y) \in R$
2.  $\forall y \in Y, \exists^{m'} x \in X, (x, y) \in R$

3.  $\forall x \in X, z \in Z$ , there exists at most  $l$  different  $y \in Y$  such that  $(x, y) \in R$  and  $g(x, z) \neq g(y, z)$
4.  $\forall y \in Y, z \in Z$ , there exists at most  $l'$  different  $x \in X$  such that  $(x, y) \in R$  and  $g(x, z) \neq g(y, z)$

Then, any oracle quantum circuit computing  $f$  uses  $\Omega(\sqrt{\frac{mm'}{l'}})$  queries to  $O_x$ .

## 5.2 Quantum detection algorithm

We can use the generalized unordered search from theorem 5.1 to detect a faulty input for a Toffoli gate with high probability.

Given  $AND_S$ , think of  $x$  as  $\sigma$ , the characteristic vector of  $S$  and take  $Y = \{f_1, \dots, f_n\}$ . The action of querying the Toffoli gate can be written as

$$T_S |y_1, \dots, y_n, b\rangle = |y_1, \dots, y_n, b \oplus \bigwedge_{i \notin S} y_i\rangle = |y_1, \dots, y_n, b \oplus \bigwedge_{i: \sigma_i=0} y_i\rangle$$

which we will use as the oracle gate to compute  $g(\sigma, y) = \bigwedge_{i: \sigma_i=0} y_i$  for  $y \in Y$ .

By theorem 5.1, we can construct an oracle quantum circuit that uses  $O(\sqrt{n})$  queries to  $T_S$  and outputs any  $f_j$  such that  $\bigwedge_{i \notin S} (f_j)_i = 1$  i.e.  $j \in S$ . Thus the position of the 0 in the output will indicate the position of a faulty input.

**Theorem 5.3** *For a (possibly) faulty Toffoli gate on  $n$  inputs, there is a quantum circuit on  $n$  inputs, making  $O(\sqrt{n})$  calls to the faulty gate, whose output qubit indicates one of the faulty inputs with high probability.*

### 5.2.1 Faults in other quantum Boolean gates

Observe that our technique for determining faults in a Toffoli gate can be also used to determine faults for other quantum Boolean gates.

**Parity:** Though we described a quantum detection circuit for a Parity gate, the above technique can be also be used to detect faults. Instead of the Parity function, we will use its complement, an equivalent function  $\bigoplus'(x_1, \dots, x_n) = 1 \oplus (\bigoplus(x_1, \dots, x_n))$ . The quantum analogue of this gate is the parity gate followed by an  $X$  gate on the target qubit, so the faulty lines are preserved in the modified gate.

Take  $Y = \{e_1, \dots, e_n\}$ . The action of the faulty gate can be described as, for  $y \in Y$ ,

$$\bigoplus_S |y_1, \dots, y_n, b\rangle = |y_1, \dots, y_n, b \oplus \left( \bigoplus_{i \notin S} y_i \right)\rangle$$

which is an oracle gate for  $g(\sigma, e_i) = 1$  iff  $i \in S$ .

After  $O(\sqrt{n})$  queries, with high probability the output of the circuit is some  $e_i$  such that  $g(\sigma, e_i) = 1$  i.e.  $i \in S$ .

## 5.3 Quantum lower bound for Toffoli

We can use a slight modification of the quantum adversary method (Theorem 5.2) to obtain a tight lower bound for the query complexity of quantum detection circuit for the Toffoli gate.

Let  $X = \{0\}$  denote a set of gate with no faults and  $Y = \{e_1, \dots, e_n\}$  denote a set of gates with exactly one faulty input. Take  $R = X \times Y$ . Let  $Z = \{f_1, \dots, f_n\}$ . Observe

that, we can assume all queries are from  $\{f_i\}$ . For other queries, the output is always 0 or 1 irrespective of the input. Then the Toffoli gate acts as an oracle gate for the following function:  $g(\sigma, f_i) = \bigwedge_{\sigma_j=0} (f_i)_j = 1$  iff  $i$  is the only faulty input.

Then in theorem 5.2,  $m = n$ ,  $m' = 1$  and  $l = l' = 1$ . We get the following lower bound for any circuit that can distinguish between  $X$  and  $Y$ :

**Theorem 5.4** *Any quantum oracle circuit that can detect if an  $n$ -input Toffoli gate has one or no faulty input lines with high probability, makes  $\Omega(\sqrt{n})$  queries to the gate.*

## 5.4 Path query model for directed graphs

We would like to point out an interesting correspondence of the fault detection problem with the following graph problem which is of independent interest **FIND<sub>PATH</sub>**: Finding a marked vertex using path queries.

**FIND<sub>PATH</sub>**: Assume we are given a *directed acyclic* graph  $G = (V, E)$  on  $n$  vertices where there is a self-loop on every vertex. One of the vertices  $v^* \in V$  is secretly marked and our goal is to find it. We are allowed to ask if there is a path from *any* vertex to  $v^*$ .

It turns out that this problem is related to finding faults of an AND gate. The similarity will be evident if we consider another related problem **FIND<sub>SUBSET</sub>**: Given a set  $S$  of  $n$  elements, to find a hidden subset  $s \subseteq S$ . We are allowed to query, for any subset  $x$ , is  $x \subseteq s$ ? Observe that **FIND<sub>SUBSET</sub>** is a special case of **FIND<sub>PATH</sub>** with  $G$  as the  $n$ -dimensional hypercube. The vertices of  $G$  are labeled by subsets of  $S$  and there is a directed edge between from  $x$  to  $y$  if  $y$  is  $x$  and exactly one new element. Then, a path exists from  $x$  to  $s$  in  $G$  if and only if  $x \subset s$ .

Next we show that **FIND<sub>SUBSET</sub>** is equivalent to finding faults in an AND gate. A **FIND<sub>SUBSET</sub>** for a  $S = \{1, \dots, n\}$  and hidden subset  $s$  is equivalent to a faulty  $n$ -bit gate  $AND_s$  (the  $i$ th input is faulty iff  $i \in s$ ). A subset query  $x \subseteq s$  is same as  $AND_s(\bar{x})$  where  $\bar{x}$  is the complement of the characteristic vector for  $x$  ( $\bar{x}_i = 0$  iff  $i \in x$ ). This equivalence will be used in the lower bound proof below.

For the general problem of **FIND<sub>PATH</sub>**, the marked vertex  $v^*$  can be obviously found using  $O(n)$  path-queries. The algorithm first finds any vertex  $v$  that has a path to  $v^*$ . It then creates a traversal tree for  $C$  rooted at  $v$ . Finding  $v^*$  then amounts to travelling down the tree along any branch that reaches  $v^*$ ;  $v^*$  is the last vertex on that branch to have a path to  $v^*$ . The worst case query complexity is also  $\Omega(n)$  - consider a star graph and mark one of the  $n - 1$  edge vertices; any algorithm who is given the graph and asks fewer than  $n - 2$  queries cannot reliably determine which vertex is marked.

## 5.5 Randomized classical lower bound for classical AND

We want to claim that no bounded error randomized algorithm can detect faults in a classical AND gate using  $o(n)$  queries. As usual, we will show a lower bound on a related decision version of the problem: To detect if there are an odd or even number of faults.

We will use a modified version of the lower bound technique for randomized algorithms by Aaronson [Aar06]. Consider any randomized algorithm which correctly answers true or false with high probability on any input by queries for bits of the input. They show the following theorem, where  $A(i)$  denotes the  $i^{th}$  bit of  $n$ -bit  $A$  (and  $i \in \{1, \dots, n\}$ ).

**Theorem 5.5** ([Aar06], Th. 5) *For an  $n$ -bit boolean function  $F$ , let  $\mathcal{A}$  denote the set of inputs  $\{x \mid F(x) = 0\}$  and  $\mathcal{B}$  denote the set of inputs which evaluate to 1. Let  $R(A, B) \geq 0$  be a real-valued function, and for  $A \in \mathcal{A}, B \in \mathcal{B}$  and  $i$  (representing the queries that  $F$  can make to*

the input) let,

$$\theta(A, i) = \frac{\sum_{B^* \in \mathcal{B}: A(i) \neq B^*(i)} R(A, B^*)}{\sum_{B^* \in \mathcal{B}} R(A, B^*)}$$

$$\theta(B, i) = \frac{\sum_{A^* \in \mathcal{A}: A^*(i) \neq B(i)} R(A^*, B)}{\sum_{A^* \in \mathcal{A}} R(A^*, B)}$$

where the denominators are all non-zero. Then the number of randomized queries needed to evaluate  $F$  with at least  $9/10$  probability is  $\Omega(1/v_{min})$ , where

$$v_{min} = \max_{\substack{A \in \mathcal{A}, B \in \mathcal{B}, i \\ R(A, B) > 0, A_i \neq B_i}} \min\{\theta(A, i), \theta(B, i)\}$$

For  $n$  bit strings  $i$  and  $A$ , we will say  $i$  is a 1-substring of  $A$  if  $A$  is 1 at all bit positions where  $i$  is 1 and denote it as  $i \subseteq^1 A$ . We notice that this theorem can be modified to work with queries that, instead of asking for input bits, ask if an  $n$ -bit string  $i$  is a 1-substring of the input  $A$ . So the above theorem holds also for this notation:  $A(i) = 1$  if  $i \subseteq^1 A$ . The following corollary then gives us the required lower bound.

**Corollary 5.6** *Any randomized algorithm which detects if an AND gate on  $n$  inputs has even or odd number of faults with high probability requires  $\Omega(n)$  calls to the faulty gate.*

**Proof.** Let  $F$  be a randomized algorithm which on an input a (possible faulty)  $AND_{\mathcal{S}}$  gate output 0 if it has even number of faults and output 1 if it has odd number of faults. Following the notation in Theorem 5.5,  $\mathcal{A}$  is the set of gates with even number of faults and  $\mathcal{B}$  is the set of gates with odd number of faults.  $R(A, B) = 1$  if  $A$  and  $B$  differ at only one faulty input and 0 otherwise. Then for any  $A \in \mathcal{A}$  (and similarly for any  $B \in \mathcal{B}$ ),  $\sum_{B^* \in \mathcal{B}} R(A, B^*) = n$ .

Consider any  $AND_{\mathcal{S}_1} \in \mathcal{A}$  and  $AND_{\mathcal{S}_2} \in \mathcal{B}$  ( $\sigma_1$  and  $\sigma_2$  will denote the characteristic vectors for the respective faults) with  $R(AND_{\mathcal{S}_1}, AND_{\mathcal{S}_2}) = 1$ . Notice that querying the faulty gate is same as a 1-substring query: for any  $n$ -bit binary input to  $AND_{\mathcal{S}}$ ,  $AND_{\mathcal{S}}(i) = 1$  iff  $\bar{i} \subseteq^1 \sigma$ , so for the rest of the proof we will focus only on 1-substring queries to  $\sigma_1$  and  $\sigma_2$ .

For any  $i$  such that  $AND_{\mathcal{S}_1}(i) \neq AND_{\mathcal{S}_2}(i)$ , either  $i \subseteq^1 \sigma_1$  or  $i \subseteq^1 \sigma_2$ . Consider the case  $i \subseteq^1 \sigma_1$  and  $i \not\subseteq^1 \sigma_2$  (the other case is similar). To compute  $\theta(AND_{\mathcal{S}_2}, i)$  we need to count the number of possible  $\sigma^*$  which differ at one bit position from  $\sigma_2$ , has even number of bits set to 1 and  $i \subseteq^1 \sigma^*$ . We claim that there is at most one possibility for  $\sigma^*$ . Consider the bit positions  $I'$  where  $i$  is set but  $\sigma_2$  is not set. If  $I'$  contains more than one position, then no  $\sigma^*$  is possible (since  $i \subseteq^1 \sigma^*$ ,  $\sigma^*$  would differ at more than one position from  $\sigma_2$ ). If  $I'$  contains only one position,  $\sigma^*$  is  $\sigma_1$  along with the bit at that position set to 1. So,  $\theta(AND_{\mathcal{S}_2}, i) \leq 1$ .

So  $\min\{\theta(AND_{\mathcal{S}_1}, i), \theta(AND_{\mathcal{S}_2}, i)\} \leq 1$ . Taking the maximum of all such  $\mathcal{S}_1, \mathcal{S}_2$  and  $i$ , we get  $v_{min} = 1/n$ .  $\square$

## 6 Conclusion

For both the Boolean functions considered here, Parity and AND, a classical circuit can detect faults in the corresponding classical gates using  $n$  queries. For AND, query the gate on  $f_1, \dots, f_n$  successively; if the output on  $f_i$  is 1, then the  $i$ -th input is faulty. For Parity gate, query the gate on  $e_1, \dots, e_n$ ; the output on  $e_i$  is 0 if and only if the  $i$ -th input is faulty. But it is not clear if there is any non-trivial upper bound to the number of queries for any general Boolean function.

Similarly, we do not know if there is a general lower bound for classical circuits, for either deterministic or probabilistic detection. For example, detecting the faulty input in a Majority gate<sup>6</sup> with only one fault can be done in  $\lceil \log_2 n \rceil$  queries. Consider the case that  $n$  is odd (the other case is similar). Observe the output of the gate on  $\lfloor n/2 \rfloor$  0's and  $\lceil n/2 \rceil$  1's. If the output is 1, then the fault is within the inputs which were set to 0 and vice versa. This strategy can be used to binary search for the faulty input. We believe the bound is tight.

We showed how our fault detection problem is related to determining input sensitivity of Boolean functions. There are several versions of sensitivity e.g. average sensitivity, block sensitivity, which are important complexity measures of functions. The complexity of estimating these values given query access to the function is an interesting research direction which might shed light on the complexity of the sensitivity function itself.

Quantum techniques might turn out to be more efficient in determining the sensitivity of a function. Consider the query complexity of distinguishing between  $s(f) = 0$  and  $s(f) = 1$ , given only black box access to  $f$ . Whereas  $s(f) = 0$  implies  $f$  is a constant function,  $s(f) = 1$  can be shown to imply that  $f$  is balanced (exactly half of the inputs evaluate to 0). While any classical deterministic algorithm must make  $\Omega(2^n)$  queries to  $f$ , using the well known Deutsch-Jozsa algorithm [DJ92], there is quantum algorithm to distinguish between these two cases using one query to  $f$ . However, the technique cannot be generalized to other values of  $s(f)$ . The query complexity of  $s(f)$  is somewhere between  $\Omega(n)$  and  $O(2^n)$ ; it would be interesting to close the gap. In addition to it, it would be worthwhile to investigate what additional properties of  $f$  would make it easier to determine  $s(f)$  <sup>7</sup>.

In this paper we showed how to relate quantum query complexity to quantum circuits. We extended two widely used query complexity results by allowing general algebraic functions for the query steps instead of querying the bits of the input. These results are of independent interest. Algebraic query complexity for quantum circuits would be an interesting area of research. In an algebraic decision tree, we are essentially trying to compute one function using the output of other functions on the input. This can essentially help us in understanding the hardness of a function relative to another function e.g. how hard is it to simulate a Toffoli gate using Parity gates.

We also considered a graph problem which bears an interesting relation with finding faults of an AND gate. As we show earlier, the deterministic query complexity for the finding a marked vertex using path queries is  $\Theta(n)$ ; however we also show that for a hypercube with  $m$  vertices, the query complexity is much lower  $\Theta(\log m)$ . We would like to ask what, if any, is the relation of the path-query complexity to any structural property of the graph.

There are a several other natural extensions of the topics discussed here which we feel worth pursuing. One is to use the techniques examined here to find defects in more complicated circuits, specifically circuits involving several gates and different types of gates. Some of the techniques discussed here are fairly general and might be applicable for other gates as well, e.g. the quantum circuit construction for the faulty Toffoli gate. A second extension is to try to find automatic fault correction strategies for faulty gates and circuits. Such strategies often go hand in hand with fault detection in the classical case. We leave the exploration of these ideas for future research.

We would like to thank Frederic Green for the idea of fault detection of parity gate and other discussions.

---

<sup>6</sup> $MAJ_n(x_1, \dots, x_n) = 1$  iff  $\sum_i x_i \geq n/2$

<sup>7</sup>E.g., can we relate the query complexity of  $s(f)$  to to  $s(f)$  itself ?

## References

- [Aar06] Scott Aaronson. Lower bounds for local search by quantum arguments. *Siam Journal on Computing*, 35(4):804–824, 2006.
- [Amb00] Andris Ambainis. Quantum lower bounds by quantum arguments. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 636–643. ACM, 2000.
- [BFNR08] Harry Buhman, Lance Fortnow, Ilan Newman, and Hein Rhrig. Quantum property testing. *Siam Journal on Computing*, 37(5):1387–1400, 2008.
- [BGH07] Debajyoti Bera, Frederic Green, and Steven Homer. Small depth quantum circuits. *SIGACT News*, 38(2):35–50, 2007.
- [BW02] Harry Buhman and Ronald De Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1):21–43, 2002.
- [CTK08] Yao-Hsin Chou, I-Ming Tsai, and Sy-Yen Kuo. Quantum boolean circuits are 1-testable. *IEEE Transactions on Nanotechnology*, 7(4):484–492, July 2008.
- [DJ92] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Royal Society of London Proceedings Series A*, 439:553–558, October 1992.
- [Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.
- [Moo99] Christopher Moore. Quantum circuits: Fanout, parity, and counting. arxiv:quant-ph/9903046, 1999.
- [PBL05] M. Perkowski, J. Biamonte, and M. Lukac. Test generation and fault localization for quantum circuits. In *Proceedings of the 35th International Symposium on Multiple-Valued Logic*, pages 62–68, May 2005.
- [PHM04] Ketan N. Patel, John P. Hayes, and Igor L. Markov. Fault testing for reversible circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23:410–416, 2004.
- [VA98] N.V. Vinodchandran V. Arvind, K.V. Subrahmanyam. *The Query Complexity of Program Checking by Constant-Depth Circuits*, volume 1741/1999 of *Lecture Notes in Computer Science*, pages 123–132. Springer Berlin / Heidelberg, 1998.

## A Generalized unordered search

We need a technical result, showing that it is possible to create a unitary transformation which transforms a particular pure state into another pure state. We include the proof for completeness.

**Lemma A.1** *Given any two pure states over  $n$  qubits,  $|x\rangle$  and  $|y\rangle$  where  $|x\rangle \neq |y\rangle$ , there is a unitary transformation  $U$  such that  $U|x\rangle = |y\rangle$  and  $U|y\rangle = |x\rangle$ .*

**Proof.** Let  $|b_1\rangle = \frac{1}{\sqrt{2}}|x\rangle - \frac{1}{\sqrt{2}}|y\rangle$  and  $|b_2\rangle = \frac{1}{\sqrt{2}}|x\rangle + \frac{1}{\sqrt{2}}|y\rangle$ . Note that  $|b_1\rangle$  and  $|b_2\rangle$  are orthonormal vectors. Extend them to form a complete orthonormal basis for the  $N = 2^n$  dimensional Hilbert space  $\{|b_1\rangle, |b_2\rangle, \dots, |b_N\rangle\}$ . Observe that,  $|x\rangle = \frac{1}{\sqrt{2}}|b_1\rangle + \frac{1}{\sqrt{2}}|b_2\rangle$  and  $|y\rangle = -\frac{1}{\sqrt{2}}|b_1\rangle + \frac{1}{\sqrt{2}}|b_2\rangle$ .

Then  $U$  can be defined as the reflection about the hyperplane orthogonal to  $|b_1\rangle$  :

$$U = -|b_1\rangle\langle b_1| + \sum_{i=2}^N |b_i\rangle\langle b_i|. \quad \square$$

Now we show how to construct a circuit to perform unordered search with generalized oracle gates.

**Theorem A.2** *Let  $g : \{0, 1\}^n \times Y \rightarrow \{0, 1\}$  be a Boolean function, where the second input to  $g$  is from a subset  $Y \subseteq \{0, 1\}^k$  for some  $k$ . Let  $x$  be an  $n$ -bit binary input,  $x = x_1, \dots, x_n$ , and  $O_x$  be an oracle gate to compute  $g$ , so  $g: O_x|i, b\rangle = |i, b \oplus g(x, i)\rangle$ .*

*Then there is a quantum oracle circuit  $C$ , with  $O(\sqrt{|Y|})$  oracle gates, such that the measurement of the output qubit of  $C|0^n, w\rangle$  ( $w$  denotes the oracle workspace qubits) is any  $\hat{i} \in Y$  such that  $g(x, \hat{i}) = 1$  with probability  $O(1)$ .*

**Given:**

- A subset of  $k$  bit binary strings  $Y \subseteq \{0, 1\}^k$
- $n$ -bit binary input  $x \in \{0, 1\}^n$
- A function  $g : \{0, 1\}^n \times Y \rightarrow \{0, 1\}$  and an oracle gate to compute  $g: O_x|i, b\rangle = |i, b \oplus g(x, i)\rangle$

**Output:** The result of the measurement is any  $\hat{i} \in Y$  with probability  $O(1)$  such that  $g(x, \hat{i}) = 1$ .

**Proof.** As usual, the oracle gate  $O$  can be transformed, using one extra ancilla, to change the phase of the register instead of changing the state:  $O_x|i\rangle = (-1)^{g(x, i)}|i\rangle$ . The ancilla can be reused for all the oracle calls, so we will not explicitly mention it while describing the circuit.

Let  $|\Psi\rangle = \sum_{y \in Y} |y\rangle$ . We need a unitary operation  $H'$  to create a uniform superposition of all elements in  $Y$ :  $H'|0^n\rangle = |\Psi\rangle$  and  $H'|\Psi\rangle = |0^n\rangle$ . We know there is gate which performs this operation from lemma A.1. Note that, this gate does not query the oracle gate and so is not counted towards the cost of the circuit.

The circuit performs the following actions in order:

1. Start with  $|0^n\rangle$ .
2. Apply  $H'$  to get  $|\Psi\rangle$ .
3. Repeat the following in order  $t$  number of times.
  - Apply the oracle gate.
  - Apply  $H'$ .
  - Perform  $\Pi = 2|0\rangle\langle 0| - I$  which negates the phase of every basis state except  $|0\rangle$ . This operation again does not incur any cost.
  - Apply  $H'$ .

Let  $G = (H'(2|0\rangle\langle 0| - I)H')O = (2|\Psi\rangle\langle \Psi| - I)O$ . Let  $Y' = \{i \in Y \mid g(x, i) = 1\}$  denote the solution space and  $Y'' = Y \setminus Y'$ . Let  $|\alpha\rangle = \frac{1}{\sqrt{|Y'|}} \sum_{i \in Y'} |i\rangle$  and  $|\beta\rangle = \frac{1}{\sqrt{|Y''|}} \sum_{i \in Y''} |i\rangle$ .

The rest of the proof follows as in the original proof [Gro96].  $|\Psi\rangle$  is in the space spanned by  $|\alpha\rangle$  and  $|\beta\rangle$ .  $O$  performs a reflection about  $|\beta\rangle$  and  $(2|\Psi\rangle\langle \Psi| - I)$  performs a reflection about  $|\Psi\rangle$ , both in the same plane. Thus in effect,  $G$  rotates any state in this space towards  $|\alpha\rangle$ . The final measurement gives us a uniformly chosen element of  $Y'$  with probability close to 1 after  $O(\sqrt{\frac{|Y|}{|Y'|}})$  queries to the oracle gate.  $\square$

## B Lower bound for generalized query complexity

**Theorem B.1** *Let  $g : \{0, 1\}^n \times Z \rightarrow \{0, 1\}$  be a Boolean function, where  $Z \subseteq \{0, 1\}^k$  for some  $n, k$ . Let  $O_x$  be an oracle gate to compute  $g$ :  $O_x|i, b\rangle = |i, b \oplus g(x, i)\rangle$ . Let  $f(x_1, \dots, x_n)$  be a function and  $X, Y \subseteq \{0, 1\}^n$  be two sets of inputs such that  $f(x) \neq f(y)$  if  $x \in X, y \in Y$ . Let  $R \subseteq X \times Y$  such that*

1.  $\forall x \in X, \exists^m y \in Y, (x, y) \in R$
2.  $\forall y \in Y, \exists^{m'} x \in X, (x, y) \in R$
3.  $\forall x \in X, z \in Z$ , there exists at most  $l$  different  $y \in Y$  such that  $(x, y) \in R$  and  $g(x, z) \neq g(y, z)$
4.  $\forall y \in Y, z \in Z$ , there exists at most  $l'$  different  $x \in X$  such that  $(x, y) \in R$  and  $g(x, z) \neq g(y, z)$

Then, any oracle quantum circuit computing  $f$  uses  $\Omega(\sqrt{\frac{mm'}{ll'}})$  queries to  $O_x$ .

**Proof.** The proof closely follows the proof by Ambainis (Theorem 2 in [Amb00]). Let  $T$  denote the number of queries made by the circuit  $C$  and  $S$  denote  $X \cup Y$ .

$C$  works on two registers: a workspace register initialized to 0 (this subspace is denoted by  $\mathcal{H}_A$ ) and a query register initialized to the uniform superposition of all strings in  $X \cup Y$  (this subspace is denoted by  $\mathcal{H}_I$ ):

$$\frac{1}{\sqrt{2|X|}} \sum_{x \in X} |0\rangle \otimes |x\rangle + \frac{1}{\sqrt{2|Y|}} \sum_{y \in Y} |0\rangle \otimes |y\rangle$$

Let  $\rho_k$  denote the density matrix of  $\mathcal{H}_I$  after  $k$  queries. We track the change in the sum of the absolute value of its off-diagonal entries  $S_k = \sum_{(x,y) \in R} |(\rho_k)_{xy}|$ , where  $k \in \{0, \dots, T\}$ .

As in the original proof, it can be shown that  $S_0 - S_T \geq (\frac{1}{2} - \sqrt{\epsilon(1-\epsilon)})\sqrt{mm'}$ .

Next, we will try to estimate an upper bound of  $S_{k-1} - S_k$ .

Writing the state before the  $k$ -th query as a superposition over the states from  $\mathcal{H}_I \otimes \mathcal{H}_A$ :

$$|\Psi_{k-1}\rangle = \sum_{i,z} \sqrt{p_{i,z}} |i, z\rangle \otimes |\Phi_{i,z}\rangle, \text{ where}$$

$$|\Phi_{i,z}\rangle = \sum_{x \in S} \alpha_{i,z,x} |x\rangle$$

Tracing out  $\mathcal{H}_A$ ,  $\rho_{k-1} = \sum_{i,z} p_{i,z} |\Phi_{i,z}\rangle \langle \Phi_{i,z}|$ . The  $k$ -th query essentially changes the sign of the phase of  $|i, z\rangle \otimes |x\rangle$  if  $g(x, i) = 1$ . So,  $\rho_k = \sum_{i,z} p_{i,z} |\Phi'_{i,z}\rangle \langle \Phi'_{i,z}|$ , where

$$|\Phi'_{i,z}\rangle = \sum_{x \in S: g(x,i)=0} \alpha_{i,z,x} |x\rangle - \sum_{x \in S: g(x,i)=1} \alpha_{i,z,x} |x\rangle$$

The rest of the proof is exactly the same: Let  $\rho_{i,z} = |\Phi_{i,z}\rangle \langle \Phi_{i,z}|$  and  $\rho'_{i,z} = |\Phi'_{i,z}\rangle \langle \Phi'_{i,z}|$ .

If  $g(x, i) = g(y, i)$ , then  $(\rho_{i,z})_{xy} = (\rho'_{i,z})_{xy}$ ; otherwise,  $(\rho_{i,z})_{xy} = -(\rho'_{i,z})_{xy}$  and  $|(\rho_{i,z})_{xy} - (\rho'_{i,z})_{xy}| = 2|\alpha_{i,z,x}||\alpha_{i,z,y}|$ . This gives us

$$\begin{aligned}
\sum_{(x,y) \in R} |(\rho_{i,z})_{xy} - (\rho'_{i,z})_{xy}| &= \sum_{(x,y) \in R: g(x,i) \neq g(y,i)} |(\rho_{i,z})_{xy} - (\rho'_{i,z})_{xy}| \\
&= \sum_{(x,y) \in R: g(x,i) \neq g(y,i)} 2|\alpha_{i,z,x}||\alpha_{i,z,y}| \\
&\leq \sum_{(x,y) \in R: g(x,i) \neq g(y,i)} 2\sqrt{\frac{l'}{l}}|\alpha_{i,z,x}|^2 + \sqrt{\frac{l}{l'}}|\alpha_{i,z,y}|^2 \\
&\leq \sum_{x \in X} l\sqrt{\frac{l'}{l}}|\alpha_{i,z,x}|^2 + \sum_{y \in Y} l'\sqrt{\frac{l}{l'}}|\alpha_{i,z,y}|^2 \\
&= \sqrt{ll'} \sum_{x \in X \cup Y} |\alpha_{i,z,x}|^2 \\
&= \sqrt{ll'}
\end{aligned}$$

So, we get,

$$\begin{aligned}
S_{k-1} - S_k &= \sum_{(x,y) \in R} |(\rho_{k-1})_{xy}| - \sum_{(x,y) \in R} |(\rho_k)_{xy}| \\
&\leq \sum_{(x,y) \in R} |(\rho_{k-1})_{xy} - (\rho_k)_{xy}| \\
&= \sum_{(x,y) \in R} \left| \sum_{i,z} p_{i,z}(\rho_{i,z})_{xy} - \sum_{i,z} p_{i,z}(\rho'_{i,z})_{xy} \right| \\
&\leq \sum_{i,z} p_{i,z} \sum_{(x,y) \in R} |(\rho_{i,z})_{xy} - (\rho'_{i,z})_{xy}| \\
&\leq \sum_{i,z} p_{i,z} \sqrt{ll'} = \sqrt{ll'}
\end{aligned}$$

Combining with the lower bound on  $S_0 - S_T$ , we get  $T \in \Omega(\sqrt{\frac{mm'}{ll'}})$ . □