Classical-Quantum Separations in Minimal Query Complexity of Boolean Functions

Chandra Sekhar Mukherjee and Subhamoy Maitra
Indian Statistical Institute, 203 B T Road, Kolkata 700 108, India

Query complexity is a model of computation in which functions are evaluated by making queries to the variables. In a very recent paper [Physical Review A 101, 022325 (2020)], Chen, Ye and Li provided a characterization of exact one-query quantum algorithms. We first note that this result is an immediate corollary of what was proved five years back by Montanaro, Jozsa and Mitchison [Algorithmica 71, pp. 775796 (2015)]. Further, in this model, we concentrate on understanding the differences between classical and quantum computation in terms of number of influencing variables of a function given lowest query complexity in the classical domain. We explore the parity decision tree model to portray possible separations between classical and quantum computations in specific cases. Our results classify different classes of Boolean functions on \( n \)-variables (\( n \) being a function of \( k \)), that can be evaluated (i) with exactly \( k \) queries in both classical and exact quantum model or (ii) with \( k \) queries in classical model and \((k - 1)\) queries in exact quantum model.

I. INTRODUCTION

Query Complexity is a model of computation in which a function \( f(x_1, x_2, \ldots, x_n) : \{0, 1\}^n \to \{0, 1\} \) is evaluated using queries to the variables \( x_i \), \( 1 \leq i \leq n \). The query complexity model has been widely studied under different computational scenarios, such as classical deterministic model and exact quantum model \[1\]. While the study can be conducted for functions with any finite range, Boolean functions are most widely studied in this area, for their simplicity as well as the richness in terms of generalization. Substantial work has been completed on asymptotic separation of query complexity under different models \[2–4\]. However, the work on finding query complexity of a generic Boolean function under different models and resources, such as specific number of variables, have not been equally exhaustively studied. In this direction, we use Boolean functions to characterize certain separations between the classical deterministic query model and the exact quantum query model. This categorization helps us to understand the capability and the difference between the power of the above two query complexity models. Let us now introduce the concept of classical and quantum oracles and describe the deterministic classical and exact quantum query model in details.

In the query complexity model, the value of any variable can only be queried using an oracle. An oracle is a black-box which can perform a particular computation. In the classical model, an oracle accepts an input \( i \) (\( 1 \leq i \leq n \)) and output the value of the variable \( x_i \). In the quantum model, the oracle needs to be reversible. It is represented as an unitary \( O_x \) which functions as follows:

\[
\begin{align*}
1 \leq i \leq n : \quad O_x |i\rangle |\phi\rangle &= |i\rangle |\phi \oplus x_i\rangle \\
i = 0 : \quad O_x |i\rangle |\phi\rangle &= |i\rangle |\phi\rangle 
\end{align*}
\]

Figure 1 represents the working of an oracle in the quantum complexity model, which is similar to what is presented in \[5\] Fig. 3.

![Figure 1: Working of a quantum oracle](image)

FIG. 1: Working of a quantum oracle

The query complexity of a function is the maximum number of times this oracle needs to be used to evaluate the value of the function \( f \) for any value of the variables \( x_1, x_2, \ldots, x_n \). We will be focusing on total Boolean functions from here on, i.e., \( f : \{0, 1\}^n \to \{0, 1\} \). Let us now specify the models.

Deterministic (Classical) Query Complexity: The minimum number of queries that a function \( f \) needs to be evaluated using a deterministic algorithm is called its Deterministic Query Complexity (\( D(f) \)). We generally omit the word ‘classical’. A query based classical deterministic algorithm for evaluating a Boolean function \( f : \{0, 1\}^n \to \{0, 1\} \) can be expressed as a rooted decision tree as follows.

In this model, every internal node corresponds to a query to a variable \( x_i \), \( 1 \leq i \leq n \). Each leaf is labeled as either 0 or 1. The tree is traversed from the root of the tree till it reaches a leaf in the following manner. Every internal node has exactly two children and depending on the outcome of the query (0 or 1 respectively), one of the two children are visited (left or right, respectively). That is this is a binary tree. The leaf nodes correspond to the output of \( f \) for different inputs. Every decision tree uniquely defines a Boolean function which we can obtain by deriving the Algebraic Normal Form (ANF) from a given tree. For example, the ANF of the Boolean function corresponding to the tree shown in Figure 2 is

\[
\text{ANF: } f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3
\]
\[(x_1 + 1)(x_2) + x_1(x_3 + 1) = x_1x_2 + x_1x_3 + x_1 + x_2 + x_3.\]

**FIG. 2: Example of a decision tree**

Corresponding to a function, there can be many Deterministic Query Algorithms that can evaluate it. The depth of a decision tree is defined as the number of edges encountered in the longest root to leaf path. Given \(f\), the shortest depth decision tree representing the function, is called the optimal decision tree of \(f\) and the corresponding depth is termed as the Deterministic classical complexity of \(f\), denoted as \(D(f)\).

**Exact Quantum Query Complexity:** A Quantum Query Algorithm is defined using a start state |\(\psi_{\text{start}}\rangle\) and a series of unitary Transformations

\[U_0, O_x, U_1, O_x, \ldots, U_{t-1}, O_x, U_t,\]

where the unitary operations \(U_j\) are indifferent of the values of the variables \(x_i\) and \(O_x\) is the oracle as defined above. Therefore, the final state of the algorithm is

\[|\psi_{\text{final}}\rangle = U_t O_x U_{t-1} \ldots U_1 O_x U_0 |\psi_{\text{start}}\rangle\]

and the output is decided by some measurement of the state |\(\psi_{\text{final}}\rangle\). A quantum algorithm is said to exactly compute \(f\) if for all \((x_1, x_2, \ldots, x_n)\), it outputs the value of the function correctly with probability 1.

The minimum number of queries needed by a Quantum Algorithm to achieve this is called the Exact Quantum Query Complexity \(Q_E(f)\) of the function. In this paper we call a Boolean function \(f\) separable if \(Q_E(f) < D(f)\) and non-separable otherwise.

There are other computational models such as the classical randomized model and the bounded error quantum model [6] and there exists rich literature on work on these models as well. However, those are not in the scope of this work. Let us now describe certain basic notions that we will be referring to for the rest of this work.

**Algebraic Normal Form (ANF):** It is known that given any total Boolean function, there exists a unique multivariate polynomial defined over GF(2) which exactly defines the function. Formally, one can write,

\[f(x_1, x_2, \ldots, x_n) = \bigoplus_{a=(a_1, \ldots, a_n) \in \{0,1\}^n} \lambda_a \prod_{i=1}^{n} x_i^{a_i},\]

where \(\lambda_a \in \{0,1\}\) and \(x_1, \ldots, x_n \in \{0,1\}\). The Hamming weight of \(x \in \{0,1\}^n\), \(wt(x)\), is defined as \(wt(x) = \sum_{i=1}^{n} x_i\) where the sum is over ring of integers. The algebraic degree of \(f\), \(\text{deg}(f)\), is defined as \(\text{deg}(f) = \max_{a \in \{0,1\}^n} \{wt(a) : \lambda_a \neq 0\}\).

We also define the term influencing variables in this context. We call a variable \(x_i\) of a function \(f(x_1, x_2, \ldots, x_n)\) influencing if there exists a set of values \(\{x_1, x_2, \ldots, x_{i-1}, x_i+1, \ldots, x_n\}\) such that \(f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \neq f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)\). The number of influencing variables is also represented as the number of variables present in the ANF of the corresponding function.

**Isomorphism:** Two functions \(f\) and \(g\) over \(\{0,1\}^n\) are called isomorphic if the ANF of \(f\) can be derived from ANF of \(g\) by negation and permutation of the input variables of \(g\) and by adding the constant term 1 in the ANF, that is negation of the output. If \(f\) and \(g\) are isomorphic then \(D(f) = D(g)\) and \(Q_E(f) = Q_E(g)\) [7] Section 2.2.2.

**A. Organization & Contribution**

With that backdrop, we describe the setup of the rest of the paper. In Section [I] we revisit the work done by Chen et al [5] and show that the result was indeed implicitly found in an earlier work by Montanaro et al [7].

In Section [II], we first construct a function with maximum number of influencing variables \(2^k - 1\) that can be evaluated using \(k\) queries in the deterministic decision tree model. This class of functions (upto isomorphism) can be evaluated with minimum number of queries in deterministic classical paradigm and we prove that these functions will also require exactly \(k\) quantum queries to be evaluated. We further show that any function with \(2^k - 1\) influencing variables and \(k\) deterministic query complexity must have the same exact quantum query complexity.

Next, we define a special class of Boolean functions, called the “Query Friendly” functions. A function \(f\) with \(n\) influencing variables is called query friendly if there does not exist any other function with \(n\) influencing variables with lesser deterministic query complexity than \(f\). We first show that all query friendly functions with \(n = 2^k - 1\) influencing variables are non-separable. Then in Section [III] we identify a set of function belonging to this class that exhibit zero separation between deterministic and exact quantum query complexity for certain values of \(n \neq 2^k - 1\).

Finally, in Section [IV] we revisit the parity decision tree model and define another set of query friendly functions to characterize the minimum separation (i.e., one) between deterministic and exact quantum query complexity for certain generalized values of \(n\). We conclude the section by showing that for other values of \(n\) there
does not exist separable query friendly functions that can be completely described by the parity decision tree method.

That is, the main motivation of this paper is to explore functions on \( n \) (a function of \( k \)) input variables that have minimum deterministic classical query complexity \( k \) and then to explore whether the quantum model can provide less number of queries or not in such a situation. We provide both the examples, where there is no separation and where there exists a separation of one query.

We conclude the paper in Section V outlining the future direction of our work.

II. REVISITING THE CASE OF \( Q_E(f) = 1 \)

In their work, Chen et al [5] specified the Boolean functions that can be evaluated using only one query in the exact quantum query model. The result is as follows.

**Theorem 1.** The Boolean functions that can be evaluated exactly with one quantum query complexity are of the form \( f = x_1 \) or \( f = x_1 + x_2 \) up to isomorphism.

However, in an earlier work Montanaro et al [7] calculated the exact query complexity of all Boolean functions of up to 4 influencing variables using semi definite programming, an idea proposed by Barnum et al [5] to characterize the power of query complexity models. The work of [7] Section 6.1] has shown that

- The single variable function \( x_1 \), and
- the only two variable Boolean function \( x_1 + x_2 \) up to isomorphism have \( Q_E(f) = 1 \). Then it was shown in [7] Section 6.2] that the minimum quantum exact query complexity of any 3 variable Boolean function is 2. This implies that there does not exist any Boolean function with 4 or more number of influencing variables with \( Q_E(f) = 1 \) which implicitly proves the statement of theorem [7]

The argument is as follows. Any Boolean function \( f \) with \( n \) influencing variables \( (x_1, x_2, \ldots, x_n) \) can be represented as \( x_n f_1(x_1, x_2, \ldots, x_{n-1}) + (1 + 1) f_2(x_1, x_2, \ldots, x_n) \), where \( Q_E(f) \geq \max(Q_E(f_1), Q_E(f_2)) \). One may note that \( f_1(x_1, x_2, \ldots, x_{n-1}) = f(x_1, x_2, \ldots, x_{n-1}, x_n = 1) \) and \( f_2(x_1, x_2, \ldots, x_{n-1}) = f(x_1, x_2, \ldots, x_{n-1}, x_n = 0) \). This completes the proof.

Thus, it is shown that the main result presented in [5] Section III] can be seen as a corollary of the work of [7]. It should be pointed out that the proof technique of [5] is different from [7] and it was commented in the abstract of [5] as

“Note that, unlike most work in the literature based on the polynomial method, our proof does not resort to any knowledge about the polynomial degree of \( f \).”

The results for one, two and three variables are obtained with experimental results using the convex optimization package CVX [9] for Matlab in [7] and then the proof follows easily.

III. DECISION TREES AND NO-SEPARATION RESULTS

As we have discussed, query algorithms can be expressed as decision trees in the classical deterministic model. Using this model we try to explore \( n_k \), where \( n_k \) is the maximum number of influencing variable of a Boolean function, that can be evaluated using \( k \) classical queries.

In this regard, let us present the following technical result.

**Lemma 1.** There exists a Boolean function \( f_k \) with \( 2^k - 1 \) influencing variables such that \( D(f) \leq k \).

**Proof.** We construct this function for any \( k \) as follows. We know that if a Boolean function \( f \) can be expressed as a decision tree of depth \( d \), then \( D(f) \leq d \). Now let us construct the candidate Boolean function. When we construct a decision tree such that no variable appears twice in the tree, we can identify an internal node with the variable that it queries. In such a case we use the notation \( val(x_i, c) \) to denote the left or right children of the internal node which queries the variable \( x_i \), where \( c \) is 0 or 1, respectively.

We now build a decision tree, which is a complete binary tree of depth \( k \). Each of the internal nodes in this tree is a unique variable, that is, no variable appears in the decision tree more than once. Since there are \( 2^k - 1 \) internal nodes in such a tree, this decision tree represents a Boolean function \( f_k \) on \( 2^k - 1 \) variables with \( D(f_k) \leq k \). Without loss of generality we can name the root variable as \( x_1 \) and label the variables from left to right at each level in ascending order. The resultant structure of the tree is shown in Figure 3

![FIG. 3: Decision Tree corresponding to function \( f \) with maximum influencing variables for \( D(f) = k \)](image)

---

**FIG. 3:** Decision Tree corresponding to function \( f \) with maximum influencing variables for \( D(f) = k \).
Having constructed such a Boolean function $f_k$, we now prove that is indeed the function with the maximum number of influencing variables that can be evaluated using the deterministic computational model using $k$ queries.

**Lemma 2.** Given any integer $k$, the maximum number of influencing variables that a Boolean function $f$ has such that $D(f) = k$ is $2^k - 1$.

**Proof.** Suppose there exists a Boolean function with $n_1 (> 2^k - 1)$ influencing variables that can be evaluated using $k$ queries. This implies that there exists a corresponding decision tree of depth $k$ that expresses this function. However, in a decision tree corresponding to a Boolean function $f$, all the influencing variables should be present as an internal node at least once in the decision tree. Otherwise,

$$f(x_1, x_2, \ldots, x_{i-1}, 0, \ldots, x_n) = f(x_1, x_2, \ldots, x_{i-1}, 1, \ldots, x_n) \forall x_j \in \{0, 1\} : j \neq i,$$

which implies that $x_i$ is not an influencing variable of the function. Since there cannot exist a decision tree of depth $k$ that has more than $2^k - 1$ internal nodes, such a function cannot exist.

This implies that for any function $f$ with $n = 2^k - 1$ influencing variables and $D(f) = k$, the corresponding decision tree is a $k$-depth complete tree where every variable is queried only once.

It immediately follows that a function $f$ with $n = 2^k - 1$ influencing variables has deterministic query complexity $D(f) \geq k$.

**Theorem 2.** Given any Boolean function $f$ with $2^k - 1$ influencing variables and $D(f) = k$ we have $Q_{E}(f) = k$.

**Proof.** This is proven by showing that any function $f$ characterized as above is at least as hard to evaluate as the function $AND_k$, which is AND of $k$ variables.

Given such a function $f$, there exists a corresponding $k$-depth complete tree $T_f$. As we have shown in Lemma 2, in such a tree all internal nodes will query a variable and all the variables will appear in the tree exactly once.

Given the decision tree $T_f$ corresponding to $f$ let $x_{i_1}, x_{i_2}, x_{i_3}, \ldots, x_{i_k}$ be a root to internal node path in the tree such that children of $x_{i_k}$ are the leaf nodes. Here

$$\text{val}(x_{i_t}, 1) = x_{i_{t+1}}, 1 \leq t \leq k - 1.$$

We call this set of variables $s_{\text{max}}$. We fix the values of the variables $\{x_1, x_2, x_3, \ldots, x_{2^k - 1}\} \setminus s_{\text{max}}$ as follows. Each of the variables at a level less than or equal to $k - 1$ is assigned either 0 or 1. Now either $\text{val}(x_{i_k}, 0) = 0$ and $\text{val}(x_{i_k}, 0) = 1$ or $\text{val}(x_{i_k}, 0) = 1$ and $\text{val}(x_{i_k}, 0) = 0$.

- In the first case, the values of variables $y_i$ in the $k$-th level is fixed at the value $c_i$ so that $\text{val}(y_i, c_i) = 1$. Then the function is reduced to $\prod_{t=1}^{k} x_{i_k}$.

The reduced function is $\text{AND}_k$ in the first case and $\text{OR}_k$ in the second case. In both the cases we have $Q_{E}(f) \geq k$, as $Q_{E}(\text{AND}_k) = Q_{E}(\text{OR}_k) = k \ [4, Table 1]$. We also know that $Q_{E}(f) \leq D(f)$ for any Boolean function $f$ and therefore $Q_{E}(f) \leq k$. Combining the two we get $Q_{E}(f) = k$.

We reiterate the idea behind the proof to simplify the argument. Reducing a function to $\text{AND}_k$ essentially implies that there exists a set of variables $x_1, x_2, \ldots, x_k$, such that if they are not all equal to 1, then the function outputs 0. In terms of the tree the implication is as follows. Let the path in the proof of Theorem 2 be $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ such that the function is reduced to $\text{AND}_k$ by fixing values of the other variables. Then while the decision tree is traversed from the root, if any of these $k$ variable’s value is 0, we move to a node that is out of the path, and then the value of the other internal nodes should be so fixed that we always reach a 0-valued leaf node.

### A. Query Friendly Functions

Having established these results, we characterize a special class of Boolean functions. Given any $n$, We call the Boolean functions with $n$ influencing variables that have minimum deterministic query complexity as the query friendly functions on $n$ variables. We denote the corresponding query complexity of this class of functions as $DQ_n$, and its value is calculated as follows.

**Lemma 3.** The value of $DQ_n$ is equal to $\lceil \log(n + 1) \rceil$.

**Proof.** We consider any $n$ such that $2^k - 1 < n \leq 2^k - 1$. We have shown in lemma 2 that there cannot exist a Boolean function with $n$ variables that can be evaluated with $k - 1$ classical queries.

Since the maximum number of influencing variables that a Boolean function with $k$ query complexity has is $2^k - 1$ as proven above, there exists a Boolean function with $n$ variables with $D(f) = k$. Now $\lceil \log(n + 1) \rceil = k$, which concludes the proof.

**Corollary 1.** For $n = 2^k - 1$, there does not exist any separable query friendly functions.

**Proof.** For $n = 2^k - 1$, we have $DQ_n = k$. We have shown in Theorem 2 that any function $f$ with $2^k - 1$ influencing variables and $D(f) = k$ has $Q_{E}(f) = k$.

Now let us provide some examples of such functions where the deterministic classical and exact quantum query complexities are equal.
• $k = 2, n = 2^k - 1 = 3, Q_E(f) = D(f) = 2$: the function is $f = (x_1+1)x_2 + x_1x_3 = x_1x_2 + x_1x_3 + x_2$.
• $k = 3, n = 2^k - 1 = 7, Q_E(f) = D(f) = 3$: the function is $f = (x_1+1)(x_2+1)x_4 + x_2x_5(x_3+1)x_6 + x_3x_7 = x_1x_2x_4 + x_1x_2x_5 + x_1x_4 + x_2x_4 + x_2x_5 + x_4 + x_1x_3x_6 + x_1x_3x_7 + x_1x_6$.

Next we move to a generalization when $n \neq 2^k - 1$.

B. Extending the result for $n \neq 2^k - 1$

We now identify a generic set of non-separable query friendly functions where $n \neq 2^k - 1$. We define this set using the decision tree model again. Let $2^k - 1 < n < 2^k - 1$. Consider a decision tree of depth $k$ such that the first $k - 1$ levels are completely filled and every variable occurs exactly once in the decision tree. That implies there are $n = 2^k - 1 + 1$ nodes in the $k$-th level. Let us denote the corresponding function as $f(n,1)$.

**Theorem 3.** The Boolean function $f(n,1)$ on $n$ influencing variables has $D(f(n,1)) = Q_E(f(n,1))$.

**Proof.** This $k$-depth decision tree constructed for any $n$ such that $2^k - 1 < n < 2^k - 1$ has the following properties.

- The corresponding function has deterministic query complexity equal to $k$. This is because the number of influencing variables in the function is more than the number of variables that a Boolean function with deterministic query complexity $k-1$ can have.
- There is at least one internal node at $k$-th level. Let that node be called $x_{ik}$. Let the root to $x_{ik}$ path be $x_1, x_2, x_3,..., x_{i-1}, x_{ik}$ such that $val(x_1, d_1) = x_{i2}, val(x_{i2}, d_2) = x_{i3}$ and so on. Applying the reduction used in Theorem 2 corresponding Boolean function can be reduced to the function $(x_1 + d_1)(x_2 + d_2)...(x_k + d_k)$ which is isomorphic to $AND_k$. (Note that $d_i = 1 + d_i$, i.e., the complement of $d_i$.) This implies that $Q_E(f_{n,1}) \geq k$. We also know $D(f_{n,1}) = k$, and therefore the exact quantum query complexity of the function is $k$.

Figure 4 gives an example of a function in $f(5,1)$. The result is thus a generalization when $2^k - 1 < n < 2^k - 1$, in identifying a class of functions where the separation between classical and quantum domain is not possible.

**IV. PARITY DECISION TREES AND SEPARATION RESULTS**

We now explore the parity decision tree model introduced in [7]. This model is constructed using the fact that in the exact quantum query model, the value of $x_i + x_{i2}$ can be evaluated using a single query. Let $f$ be a Boolean function that can be expressed as a $k$-depth decision tree in which every internal node either queries a variable $x_i$ or the parity of two variables, $x_{i1} + x_{i2}$. We can then say that $Q_E(f) \leq k$. Figure 5 gives an example of a parity decision tree.

FIG. 4: Decision Tree corresponding to $f(5,1)$

FIG. 5: Example of a parity decision tree

The corresponding Boolean function is $(x_1 + x_2)x_4 + (x_1 + x_2 + 1)x_3$, with deterministic query complexity 3 and exact quantum query complexity 2.

A $k$-depth parity decision tree can only evaluate a function of degree less than or equal to $k$, whereas there may exist a Boolean function of degree higher than $k$ that can be evaluated using $k$ queries. Thus, although this model does not completely capture the power of the quantum query model, we use the generalized structure of this model to find separable query friendly functions for certain values of $n$.

We say a parity decision tree $T$ completely describes a Boolean function $f$ if $T$ is a parity decision tree with the minimum depth (say $depth_f$) among all parity decision trees that represent $f$ and $Q_E(f)$ is equal to $depth_f$.

**Lemma 4.** Given any $k$ there exists a Boolean function $f$ with $2^{k+1} - 2$ variables such that $Q_E(f) = k$.

**Proof.** This proof follows directly from the definition of parity decision trees and the proof of existence of a Boolean function with $2^k - 1$ variables with $D(f) = k$. 

The result is such a generalization when $2^{k-1} - 1 < n < 2^k - 1$, in identifying a class of functions where the separation between classical and quantum domain is not possible.
Again we construct a \( k \) depth complete parity decision tree such that every internal node is a query of the form \( x_{i_1} + x_{i_2} \) such that no variable appears twice in the tree. This tree represents a Boolean function \( f \) of \( 2(2^k - 1) \) variables and inherently \( Q_E(f) \leq k \). This function can also be reduced to the \( \text{AND}_k \) function which implies \( Q_E(f) \geq f \). This implies \( Q_E(f) = k \). We skip the proof of reduction to avoid repetition.

This is also the maximum number of influencing variables that a function \( f \) can have so that \( Q_E(f) = k \) and \( f \) can be completely described using parity decision trees. This can be proven in the same way as in lemma 2 and we do not repeat it for brevity.

We now prove some observations related to separability for a broader class of functions and then explore separability in query friendly functions.

**Theorem 4.** If \( n \neq 2^k - 1 \) for any \( k \), then there exists a Boolean function for which \( Q_E(f) < DQ_n \).

**Proof.** Let \( 2^k - 1 < n < 2^k - 1 \) for some natural number \( k \). In this case \( DQ_n = k \). However, there exist Boolean functions \( f_Q \) with \( n \) influencing variables such that \( Q_E(f_Q) = k - 1 \). We define a generic class of such functions using parity decision trees. Let \( n = 2^{k-1} - 1 \). Then we can always construct a complete parity decision tree of depth \( k - 1 \) with the following constraints:

- Every variable appears only once in the tree.
- \( y \) internal nodes have query of the form \( x_{i_1} + x_{i_2} \).
  
  The rest of the internal nodes query the value of a single variable.

Since \( y \leq 2^{k-1} - 1 \), which is the number of internal nodes in a complete parity decision tree of depth \( k - 1 \), such a function always exists. 

However, if \( n = 2^k - 1 \) for some \( k \), then there does not exist any Boolean function \( f \) that can be completely expressed using the parity decision trees, such that \( Q_E(f) = DQ_n \). If \( n = 2^k - 1 \) then \( DQ_n = k \) as well and there does not exist any Boolean function \( f_Q \) with \( n \) variables that can be expressed using parity trees and has \( Q_E(f) \leq k - 1 \). This is true as we have already obtained that the Boolean function with maximum number of influencing variables and depth \( k - 1 \), that can be expressed using parity decision tree is \( 2^k - 2 \) (putting \( k - 1 \) in place of \( k \) in Lemma 4 above).

Moreover, there does not exist any Boolean function with 3 influencing variables such that exact query complexity is less than \( DQ_3 \), which is equal to 2. It is interesting to note that if for some \( n = 2^k - 1 \) there exists a Boolean function with \( Q_E(f) = k - 1 \) then there exists separation for all \( n = 2^j - 1 \) if \( j > k \). This can be proven with induction.

**Lemma 5.** If there exists a function \( f_k \) with \( 2^k - 1 \) influencing variables such that \( Q_E(f) = k - 1 \), then there exists a function \( f_j \) with \( 2^j - 1 \) influencing variables such that \( Q_E(f) \leq j - 1 \) for all \( j > k \).

**Proof.** If there exists a function \( f_k \) with the specified property then \( f_{k+1} \) can be constructed as follows. \( f_{k+1} = x_{2^k+1} + (x_{2^k} + x_{2^k+1} + \ldots + x_{2^k+1}) \). It is easy to see \( Q_E(f_{k+1}) \leq k \). Using this construction recursively yields a desired function for any \( j > k \).

We complete the categorization by defining a generalized subclass of Query friendly Boolean functions. We define this subclass such that a function \( f \), belonging to this, has \( Q_E(f) = DQ_n - 1 \).

### A. Separable Query Friendly functions

We construct a generic function for this set of query friendly functions using parity decision trees for values of \( n \) such that there exists \( k \), \( 2^{k-1} - 1 < n \leq 2^{k-1} - 1 \). We first describe the construction using a parity decision tree and then prove the query complexity values of the function.

Let us construct a parity decision tree of depth \( k - 1 \) in the following manner. The first \( k - 2 \) levels are completely filled, with each internal node querying a single variable. All variable appears exactly once in this tree. Let these variables be termed \( x_1, x_2, \ldots, x_{2^{k-2}-1} \). In the \((k-1)\)-th level, there are \( \left\lceil \frac{n-(2^{k-2}-1)}{2} \right\rceil \) internal nodes, with each query being of the form \( x_{i_1} + x_{i_2} \). (In case \( n - 2^{k-2} + 1 \) is odd, there is one node querying a single variable). Then if \( n = 2^{k-1} \) there are \( 2^{k-3} + 1 \) internal nodes in \((k-1)\)-th level and if \( n = 2^{k-1} + 2^{k-2} - 1 \) there are \( 2^{k-2} \) nodes in the \((k-1)\)-th level, resulting in a complete binary tree of depth \( k - 1 \). We denote this generic function as \( f(n, 2) \).

**Theorem 5.** The Boolean function \( f(n, 2) \) on \( n \) influencing variables has \( D(f) = DQ_n \) and \( Q_E(f) = DQ_n - 1 \).

**Proof.** If \( 2^k - 1 < n \leq 2^{k-1} + 2^{k-2} - 1 \) then \( DQ_n = k \). We first prove that \( Q_E(f(n, 2)) = k - 1 \). Since there exists a parity decision tree of depth \( k - 1 \),

\[
Q_E(f(n, 2)) \leq k - 1. \quad (1)
\]

If we fix one of the variables of each query of type \( x_{i_1} \) to zero then the reduced tree corresponds to a non-separable function shown in Fig. 3 of depth \( k - 1 \), that is the function can be reduced to \( \text{AND}_{k-1} \). This implies

\[
Q_E(f(n, 2)) \geq k - 1. \quad (2)
\]

Combining \(1\) and \(2\) we get \( Q_E(f(n, 2)) = k - 1 \).

Now we show that \( D(f(n, 2)) = k \) by converting the parity decision tree to a deterministic decision tree of depth \( k \). All the internal nodes of the parity decision tree from level 1 to level \( k - 2 \) queries a single variable. The nodes in the \((k-1)\)-th level have queries of the form \( x_{i_1} + x_{i_2} \). Each such node can be replaced by a deterministic
tree of depth 2 in the following way. Suppose there is a internal node $x_i + x_j$ in the $(k-1)$-th level.

We replace this node with a tree, whose root is $x_i$. Both the children of the node queries $x_i$ and the leaf node values are swaped in the two subtrees. Without loss of generality, suppose in the original tree $val(x_i + x_j, 0) = 0$ and $val(x_i + x_j, 1) = 1$. Then in the root node $val(val(x, 0), 0) = 0$ and $val(val(x, 1), 0) = 1$ and so on. Figure 6 gives a pictorial representation of the transformation. The resultand deterministic decision tree is of depth $k$ as there is at least $2^{k-3}$ node in the $k-1$-th level in the parity decision tree which goes through transformation. This implies $D(f_{n,2}) \leq k$. We also know that in this case $DQ_n = k$. Combining the two results we get $D(f_{n,2}) = k$. □

![FIG. 6: Conversion of a node in the parity decision tree to a deterministic decision tree](image)

Let us now consider a function of the form $f_{5,2}$ described by its ANF as below:

$$f = (x_1 + 1)(x_2 + x_3) + x_1(x_4 + x_5) = x_1x_2 + x_1x_3 + x_1x_4 + x_1x_5 + x_2 + x_3.$$ 

This provides an example for $n = 5, D(f) = 3$, and $Q(f) = 2$. In Figure 7 we present the decision tree for this function and the corresponding quantum circuit is provided in Figure 8.

We now explain for the sake of completeness the difference in working of the exact quantum and deterministic algorithm for this function.

Suppose we want to evaluate this function at the point $(1,0,1,0,1)$. The deterministic algorithm will first query $x_1$, and getting its value as 1 it will then query $x_4$. Since $x_4$ is 0 it will query the $x_3$ node which is it's left children and then output 1 as $x_5$ is 1.

The quantum algorithm will evaluate as follows.

1. Here $\psi_{start} = |0\rangle |0\rangle |0\rangle |0\rangle |0\rangle$.
2. The first $X$ gate transforms it into $|1\rangle_2 |0\rangle |0\rangle |0\rangle |0\rangle$

   Here $|i\rangle_2$ implies $|a\rangle |b\rangle |c\rangle$ where abc is the binary representation of integer i.
3. Then we get $O_x(|1\rangle_2 |0\rangle |0\rangle) = |1\rangle_2 |x_1\rangle |0\rangle = |1\rangle_2 |1\rangle |0\rangle$.

![FIG. 7: Parity Decision(a) and Deterministic(b) Tree corresponding to $f_{5,2}$](image)

4. The CNOT gates, the not gate and the Hadamard gates ($H^3$ and $H^4$) transform the state into $\frac{(|4\rangle_2 + |5\rangle_2)}{\sqrt{2}} |−⟩ |0\rangle$ where $|−⟩ = \frac{|0\rangle − |1\rangle}{\sqrt{2}}$.

5. Now:

$$O_x\left(\frac{|4\rangle_2 + |5\rangle_2}{\sqrt{2}} |−⟩ |0\rangle\right) = \frac{(-1)^{x_4} |4\rangle_2 + (-1)^{x_5} |5\rangle_2}{\sqrt{2}} |−⟩ |0\rangle.$$ 

Let this state be $|\phi\rangle$.

6. $H^3 |\phi\rangle = \frac{1}{2}((-1)^{x_4} + (-1)^{x_5}) |4\rangle_2 + ((-1)^{x_4} - (-1)^{x_5}) |5\rangle_2 |−⟩ |0\rangle$.

7. since $x_4 = 0$ and $x_5 = 1$ we get $|5\rangle_2 |−⟩ |0\rangle$ which is equal to $|1\rangle |0\rangle |1⟩ |−⟩ |0\rangle$. Measuring the third qubit in computational basis we get the desired output, 1.

This completes the example of separation.

Finally, we conclude this section by proving that our construction of separable query friendly function indeed finds such examples for all cases where a parity decision
Theorem 6. If $2^{k-1} + 2^{k-2} - 1 < n \leq 2^k - 1$, there does not exist any separable query friendly function that can be completely described using parity decision trees.

Proof. Let $f_n$ be a query friendly function on $2^{k-1} + 2^{k-2} + 1 < n \leq 2^k - 1$ influencing variables. In this case $DQ_n = k$, and hence $D(f_n) = k$. Therefore there exists a corresponding $k$-depth decision tree $T_f$. As we know there are at most $2^{k-1}$ internal nodes in such a tree and at least $2^{k-1} + 2^{k-2}$ variables that needs to be queried at least once. Therefore there can be at most $2^{k-2} - 1$ internal nodes which query variables that appear more than once in the tree.

This implies by pigeon-hole principle that there exists two sibling nodes in the $k$-th level querying $x_{i_1}^1$ and $x_{i_2}^1$ such that these variables appear only once in the decision tree. We consider the root($x_i$) to $x_{i_1}^1$ path. It is to be noted that the root variable needs to be queried only once in any optimal tree. Let us also assume for simplicity that $val(x_{i_1}^1, 0) = 0, val(x_{i_2}^1, 0) = 0$ and

$$val(x_{i_{t}}, d_t) = x_{i_{t+1}}, 1 \leq t \leq k - 2$$
$$val(x_{i_{k-1}}, d_{k-1}) = x_{i_{k}}$$

Let us now define the following sets of variables:

$$X_j \subseteq \{x_1, x_2, \ldots, x_n\}$$
$$Y_j \subseteq (\{x_1, x_2, \ldots, x_n\} \setminus \{x_{i_1}^j, x_{i_2}^j\})$$
where $1 \leq j \leq k$

Let $g_j$ and $h_j, 1 \leq j \leq k$ be functions with influencing variables belonging from the sets $X_j$, $Y_j$ respectively. Then the ANF of $f_n$ can be described as:

$$f_n = (x_{i_1} + \overline{x_{i_1}})g_1(X_1) + (x_{i_1} + d_1)h_1(Y_1)$$

This is because the variables $x_{i_1}^j$ and $x_{i_2}^j$ can influence the function if and only if $x_{i_1} = d_1$. This is due to the fact that they are each queried only once in the decision tree. Similarly,

$$g_1(X_1) = (x_{i_2} + \overline{x_{i_2}})g_2(X_2) + (x_{i_2} + d_2)h_2(Y_2),$$

and so on. Finally we have

$$g_{k-2}(X_{k-2}) = (x_{i_{k-1}} + \overline{x_{i_{k-1}}}x_{i_1}^k + (x_{i_{k-1}} + d_{k-1})x_{i_2}^k.$$ 

Therefore, the function $f_n$ can be written as

$$f_n = (x_{i_1} + \overline{x_{i_1}})(x_{i_1} + \overline{d_2})\ldots$$
$$\ldots (x_{i_{k-1}} + \overline{d_{k-1}})x_{i_1}^k + (x_{i_{k-1}} + d_{k-1})x_{i_2}^k) + h_k(Y_k).$$

This, in turn, implies that the resultant ANF contains a $k$-term monomial $x_{i_1}x_{i_2}\ldots x_{i_k}$, which implies $deg(f) \geq k$.

It has been shown in [3, 3.1] that the minimum depth of any parity decision tree completely describing $f$ is at equal to or greater than $deg(f)$, which implies there does not exist any query friendly function that can be completely described with a parity decision tree of depth $k-1$. This concludes our proof.

V. CONCLUSION

In this paper we have discussed the separation between the deterministic and exact quantum query model in terms of the number of influencing variable. We have further used the parity decision tree model to find separation between deterministic and exact quantum query in a special class of Boolean functions (Query Friendly functions) using the structured nature of the parity decision tree model. The characterization achieved by us in terms of query friendly functions is as follows.

1. If $n = 2^k - 1$, then all query friendly functions are non separable.
2. If $n \neq 2^k - 1$, then we construct a set of non-separable functions, namely $f_{(n,1)}$.
3. If $2^k - 1 < n \leq 2^k - 1 + 2^{k-2} - 1$, then we construct a set of separable functions, namely $f_{(n,2)}$.
4. If $2^k - 1 + 2^{k-2} - 1 < n < 2^k - 1$, we show that no separable function on $n$ variables can be completely described using parity decision trees.

We have also observed some crucial problems which shall exhaustively determine the limitation of this model in these cases. The problems are as follows:

1. Does there exist a function $f_1$ with $n = 2^{k-1}$ influencing variables such that $Q_E(f_1) < k$?
2. Does there exist a separable query friendly function $f_2$ with $n$ influencing variables, where $2^{k-1} + 2^{k-2} - 1 < n < 2^k - 1$?

If any of the above problems yield a negative result that would imply the parity decision tree model indeed completely characterizes the functions in such a scenario.

[1] H. Buhrman and R. de Wolf, Theoretical Computer Science, 288(1), pp. 21-43, (2002).
[2] A. Ambainis, In Proceedings of the 45th Annual ACM
Symposium on Theory of Computing (ACM Press, New York, 2013), pp. 891900.

[3] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, Journal of the ACM, 48(4), pp. 778-797 (2001).

[4] A. Ambainis, SIAM J. Comput. 45, pp. 617-631 (2016).

[5] W. Chen, L. Li and Z. Ye, Phys. Rev. A 101, 022325 (2020).

[6] A. Ambainis, Proceedings of the International Congress of Mathematicians, pp. 3265-3285 (2018).

[7] A. Montanaro, R. Jozsa, and G. Mitchison. Algorithmica 71, pp. 775796 (2015).

[8] H. Barnum, M. Saks and M. Szegedy, In proceedings of 18th IEEE Annual Conference on Computational Complexity, pp. 179-193, (2003).

[9] M. Grant and S. Boyd, http://cvxr.com/cvx April 2011