NAND-Trees, Average Choice Complexity, and Effective Resistance

Stacey Jeffery[*] and Shelby Kimmel[†]

Abstract

We show that the quantum query complexity of evaluating NAND-tree instances with *average choice complexity* at most W is $O(W)$, where average choice complexity is a measure of the difficulty of winning the associated two-player game. This generalizes a superpolynomial speedup over classical query complexity due to Zhan *et al.* We further show that the player with a winning strategy for the two-player game associated with the NAND-tree can win the game with an expected $\widetilde{O}(N^{1/4}\sqrt{\mathcal{C}(x)})$ quantum queries against a random opponent, where $\mathcal{C}(x)$ is the average choice complexity of the instance. This gives an improvement over the query complexity of the naive strategy, which costs $\widetilde{O}(\sqrt{N})$ queries.

The results rely on a connection between NAND-tree evaluation and *st*-connectivity problems on certain graphs, and span programs for *st*-connectivity problems. Our results follow from relating average choice complexity to the effective resistance of these graphs, which itself corresponds to the span program witness size.

1 Introduction

NAND-tree evaluation is a Boolean formula evaluation problem that has proven to be a rich ground for developing quantum algorithms. The first quantum algorithm for evaluating NAND-trees was created by Farhi, Goldstone, and Gutmann in the continuous time model [FGG07], and showed that a tree with input size N, corresponding to a formula consisting of nested NAND-gates to depth $\log N$, could be evaluated in time $O(\sqrt{N})$. They used an algorithm that involved scattering a wavepacket off of a graph. The NAND-tree problem was quickly adapted to the query model, [CCJYM09, ACR+10], and seems to be a major motivation for the development of span program quantum algorithms in [RŠ12]. Through the refinement of span program algorithms, Reichardt was able to show that any formula with $O(1)$ fan-in on N variables (of which NAND-trees are a special case) can be evaluated in $O(\sqrt{N})$ queries [Rei11b]. Classically, the query complexity of evaluating NAND-trees is $\Theta(N^{.753})$ [SW86]. Variants of the NAND-tree problem have also been studied. For example, when the input is in a certain class, called *k-fault trees*, the quantum query complexity can be improved to $O(2^k)$ [ZKH12, Kim11]. A lower bound of $\Omega((\log \log N - \log k)^k)$ on the classical query complexity of evaluating *k*-fault trees makes this a *superpolynomial* quantum speedup for a range of values of k [ZKH12].

Every NAND-tree instance also defines a two-player game as follows. Players A and B move on a full binary tree of depth $\log N$, with leaves labeled by the input bits of $x \in \{0,1\}^N$. Starting from the root, A and B alternate choosing one child of the current node to move to until a leaf is reached. If the value of the leaf is 1, A wins, and if the value of the leaf is 0, B wins. It turns out

---

[*]Institute for Quantum Information and Matter (IQIM), Caltech, `sjeffery@caltech.edu`

[†]Joint Center for Quantum Information and Computer Science (QuICS), University of Maryland, `shelbyk@umd.edu`

that for 1-instances of NAND-tree evaluation, A can always win if she plays optimally, no matter what strategy B employs, and for 0-instances, B can always win if he plays optimally.

The results of [ZKH12] show a connection between the difficulty of winning the game associated with a NAND-tree instance, as measured by the *fault complexity*, and the quantum query complexity of evaluating a NAND-tree instance. A NAND-tree instance has fault complexity 2^k if the player with the winning strategy will encounter at most k *faults* on any winning path, where a fault is a node at which one choice leads to a sub-game in which the player still has a winning strategy, while the other does not. In this paper, we consider a more nuanced measure of the difficulty of winning the two-player game associated with a NAND-tree, which we call the *average choice complexity*. The average choice complexity is upper bounded by the fault complexity. Like the fault complexity, the average choice complexity is related to the number of critical decisions a player must make to win the two-player game, but rather than describing the worst case, as with the fault measure, the average choice complexity depends on decisions made in a good strategy, averaged over the opponent's strategy. The average choice complexity also provides a more subtle characterization of the criticality of the choice made at a particular node, and captures the fact that nodes that are not faults can still be important decision points.

By exploiting an elegant connection between NAND-trees (or $(\wedge, \vee)$-formulas in general) and st-connectivity problems on certain graphs, we find that the average choice complexity is exactly the span program witness size in a span program for evaluating NAND-trees. We are thus able to generalize the superpolynomial speedup of [ZKH12] to show that the bounded error quantum query complexity of evaluating NAND-trees, when we are promised that the average choice complexity of the input is at most W, is $O(W)$ (Theorem 3).

A related problem to *evaluating* a NAND-tree instance is *winning* a NAND-tree instance. The connection between NAND-tree evaluation and winning strategies suggests a strategy for winning the game: at every node, solve the instances of NAND-tree rooted at each child, and if possible, always choose one that evaluates to 1 if you are Player A, or 0 if you are Player B. The total number of quantum queries employed by this strategy is $\widetilde{O}(\sqrt{N})$. As a second application of the connection between average choice complexity and span program witness size, we are able to give a strategy that wins a NAND-tree against a random opponent using an expected $\widetilde{O}(N^{1/4}\sqrt{\mathcal{C}})$ quantum queries, where $\mathcal{C}$ is the average choice complexity (Theorem 6). This result gives an operational interpretation of average choice complexity as the difficulty, in quantum query complexity, of winning against a random opponent. The algorithm uses a recent result of [IJ15] that constructs an algorithm for estimating the span program witness size of any span program. The player uses this estimation algorithm to estimate the average choice complexity of both children and then chooses the path that is easier on average.

We achieve the connection between average choice complexity and witness size by exploiting a link between evaluating NAND-trees and solving st-connectivity in certain graphs, which may be of further interest. In particular, we find that 1-valued NAND-tree instances are related to st-connected subgraphs of a certain graph G, while 0-valued instances are related to st-connected subgraphs of the dual of G.

The relationship we uncover between NAND-trees, st-connectivity problems on graphs and their duals, and two-player games hints at many connections that might be explored. When is the quantum query complexity of evaluating Boolean formulas characterized by graph connectivity problems? What role do graphs and their duals play in span program algorithms? Our current span program algorithm characterizes properties of a two-player game assuming the opponent plays randomly; by adjusting the span program algorithm, could we characterize the same two-player game, but with different strategies for the players? On the other hand, it may be that these strategies are somehow particularly natural for quantum algorithms, in which case, we would like

to understand why.

Outline In Section 2, we give some requisite background information. In Section 3, we define average choice complexity. In Section 4, we prove our first main result: that NAND-trees with average choice complexity at most W can be evaluated in $O(W)$ quantum queries. We also show in this section that the average choice complexity can be estimated using $\widetilde{O}\left(\sqrt{\mathcal{C}(x)}N^{1/4}\right)$ queries, where $\mathcal{C}(x)$ is the average choice complexity of the instance. We prove these results by exploiting a connection between NAND-trees and *st*-connectivity on a family of graphs and their duals. We first relate the average choice complexity to the effective resistance of these graphs (Section 4.1). Then, we construct and analyze a span program for *st*-connectivity problems on these graphs, such that the witness size of the span program depends on the effective resistance of the graph (Section 4.2). In Section 5, we give a quantum algorithm for playing the two-player NAND-tree game that makes use of our algorithm for estimating $\mathcal{C}(x)$. Finally, in Section 6, we discuss some open problems.

2 Preliminaries

In this section, we will provide the necessary information to understand our paper: in Section 2.1, we describe NAND-trees, in Section 2.2 we provide some basic results about span programs and quantum algorithms, and in Section 2.3 we introduce some key concepts from graph theory.

2.1 NAND-Trees

A NAND-tree is a full binary tree of depth ℓ in which each leaf is labeled by a variable x_i, and each internal node is labeled by either $\vee$ (OR), if it is at even distance from the leaves, or $\wedge$ (AND), if it is at odd distance from the leaves. While a NAND-tree of depth ℓ is sometimes defined as a Boolean formula of NANDs composed to depth ℓ, we will instead think of the formula as an alternation of ANDs and ORs — when ℓ is even, these two characterizations are identical. An instance of NAND-tree is a binary string $x \in \{0,1\}^N$, where $N = 2^\ell$. We use NAND$_\ell$ to denote a complete NAND-tree of depth ℓ. For instance, a NAND-tree of depth 2 represents the formula:

$$\text{NAND}_2(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4). \tag{1}$$

If NAND$_\ell(x) = 1$, we say that x is a 1-instance, and otherwise we say it is a 0-instance.

A NAND-tree instance can be associated with a two-player game. Let $x \in \{0,1\}^N$ be an input to a depth-ℓ NAND-tree, so $N = 2^\ell$. Then we associate this input to a game played on a full binary tree as in Figure 1, where the leaves take the values x_i. The game starts at the root node v, which we call the live node. If the live node is at even distance from the leaves (an OR node) Player A chooses to move to one of the live node's children. The chosen child now becomes the live node. At each further round, as long as the live node is not a leaf, if the live node is at even (respectively odd) distance from the leaves, Player A (resp. Player B) chooses one of the live node's children to become the live node. When the live node becomes a leaf, if the leaf has value 1, then Player A wins, and if the leaf has value 0, then Player B wins. The sequence of moves by each player determines a path from the root to a leaf.

A simple inductive argument shows that if x is a 1-instance of NAND-tree, then there exists a strategy by which Player A can always win, no matter what strategy B employs; and if x is a 0-instance, there exists a strategy by which Player B can always win. We say an input x is A-winnable if it has value 1 and B-winnable if it has value 0.

In this paper, we will consider trees of even depth for simplicity, but the arguments can be easily extended to odd depth trees.

2.2 Span Programs

In this section, we review the concept of span programs, and their use in quantum algorithms. Span programs [KW93] were first introduced to the study of quantum algorithms by Reichardt and Špalek [RŠ12]. They have since proven to be a tool of immense importance for designing quantum algorithms in the query model.

Definition 1 (Span Program). *A span program $P = (H, V, \tau, A)$ on $\{0,1\}^n$ is made up of*

1. *finite-dimensional inner product spaces $H = H_1 \oplus \cdots \oplus H_n$, and $\{H_{j,b} \subseteq H_j\}_{j \in [n], b \in \{0,1\}}$ such that $H_{j,0} + H_{j,1} = H_j$,*

2. *a vector space V,*

3. *a target vector $\tau \in V$, and*

4. *a linear operator $A : H \to V$.*

For every string $x \in \{0,1\}^n$, we associate the subspace $H(x) := H_{1,x_1} \oplus \cdots \oplus H_{n,x_n}$, and an operator $A(x) := A\Pi_{H(x)}$, where $\Pi_{H(x)}$ is the orthogonal projector onto $H(x)$.

Given a span program, one can obtain a quantum query algorithm based on that span program. The parameters of the span program that determine the query complexity of the algorithm are the positive and negative witness sizes:

Definition 2 (Positive and Negative Witness). *Let P be a span program on $\{0,1\}^n$ and let x be a string $x \in \{0,1\}^n$. Then we call $|w\rangle$ a* positive witness *for x in P if $|w\rangle \in H(x)$, and $A|w\rangle = \tau$. We define the* positive witness size of x *as:*

$$w_+(x, P) = w_+(x) = \min\{\||w\rangle\|^2 : |w\rangle \in H(x) : A|w\rangle = \tau\},$$

if there exists a positive witness for x, and $w_+(x) = \infty$ otherwise. We call a linear map $\omega : V \to \mathbb{R}$ a negative witness *for x in P if $\omega A\Pi_{H(x)} = 0$ and $\omega\tau = 1$. We define the* negative witness size of x *as:*

$$w_-(x, P) = w_-(x) = \min\{\|\omega A\|^2 : \omega \in \mathcal{L}(V, \mathbb{R}) : \omega A\Pi_{H(x)} = 0, \omega\tau = 1\},$$

if there exists a negative witness, and $w_-(x) = \infty$ otherwise. If $w_+(x)$ is finite, we say that x is positive *(wrt. P), and if $w_-(x)$ is finite, we say that x is* negative. *We let P_1 denote the set of positive inputs, and P_0 the set of negative inputs for P.*

For a decision problem $f : X \to \{0,1\}$, with $X \subseteq \{0,1\}^n$, we say that P *decides* f if $f^{-1}(0) \subseteq P_0$ and $f^{-1}(1) \subseteq P_1$. In that case, we can use P to construct a quantum algorithm that decides f, where the number of queries used by the algorithm depends on the witness sizes:

Theorem 1 ([Rei09]). *Fix $X \subseteq \{0,1\}^n$ and $f : X \to \{0,1\}$, and let P be a span program on $\{0,1\}^n$ that decides f. Let $W_+(f, P) = \max_{x \in f^{-1}(1)} w_+(x, P)$ and $W_-(f, P) = \max_{x \in f^{-1}(0)} w_-(x, P)$. Then there is a bounded error quantum algorithm that decides f with quantum query complexity $O(\sqrt{W_+(f, P)W_-(f, P)})$.*

In [IJ15], Ito and Jeffery show that additional quantum algorithms can be derived from a span program P. These algorithms depend on the approximate positive and negative witness sizes.

Definition 3 (Approximate Positive Witness). *For any span program P on $\{0,1\}^n$ and $x \in \{0,1\}^n$, we define the* positive error *of x in P as:*

$$e_+(x) = e_+(x, P) := \min \left\{ \left\| \Pi_{H(x)^\perp} |w\rangle \right\|^2 : A|w\rangle = \tau \right\}.$$

We say $|w\rangle$ is an approximate positive witness *for x in P if $\left\| \Pi_{H(x)^\perp} |w\rangle \right\|^2 = e_+(x)$ and $A|w\rangle = \tau$. We define the* approximate positive witness size *as*

$$\tilde{w}_+(x) = \tilde{w}_+(x, P) := \min \left\{ \| |w\rangle \|^2 : A|w\rangle = \tau, \left\| \Pi_{H(x)^\perp} |w\rangle \right\|^2 = e_+(x) \right\}.$$

Note that if $x \in P_1$, then $e_+(x) = 0$. In that case, an approximate positive witness for x is a positive witness, and $\tilde{w}_+(x) = w_+(x)$. For negative inputs, the positive error is larger than 0.

We can define a similar notion of approximate negative witnesses:

Definition 4 (Approximate Negative Witness). *For any span program P on $\{0,1\}^n$ and $x \in \{0,1\}^n$, we define the* negative error *of x in P as:*

$$e_-(x) = e_-(x, P) := \min \left\{ \left\| \omega A \Pi_{H(x)} \right\|^2 : \omega(\tau) = 1 \right\}.$$

Any ω such that $\left\| \omega A \Pi_{H(x)} \right\|^2 = e_-(x, P)$ is called an approximate negative witness *for x in P. We define the* approximate negative witness size *as*

$$\tilde{w}_-(x) = \tilde{w}_-(x, P) := \min \left\{ \| \omega A \|^2 : \omega(\tau) = 1, \left\| \omega A \Pi_{H(x)} \right\|^2 = e_-(x, P) \right\}.$$

Note that if $x \in P_0$, then $e_-(x) = 0$. In that case, an approximate negative witness for x is a negative witness, and $\tilde{w}_-(x) = w_-(x)$. For positive inputs, the negative error is larger than 0.

Ito and Jeffery give a quantum algorithm to estimate the positive witness size (resp. negative witness size) of a span program; the query complexity of the algorithm depends on the approximate negative (resp. positive) witness sizes:

Theorem 2 (Witness Size Estimation Algorithm, [IJ15]). *Fix $X \subseteq \{0,1\}^n$ and $f : X \to \mathbb{R}_{\geq 0}$. Let P be a span program such that for all $x \in X$, $f(x) = w_+(x, P)$ and define $\widetilde{W}_- = \widetilde{W}_-(f, P) = \max_{x \in X} \tilde{w}_-(x, P)$. There exists a quantum algorithm that estimates f to relative error ε and that uses $\widetilde{O}\left(\frac{1}{\varepsilon^{3/2}} \sqrt{w_+(x)\widetilde{W}_-} \right)$ queries. Similarly, let P be a span program such that for all $x \in X$, $f(x) = w_-(x, P)$ and define $\widetilde{W}_+ = \widetilde{W}_+(f, P) = \max_{x \in X} \tilde{w}_+(x, P)$. Then there exists a quantum algorithm that estimates f to accuracy ε and that uses $\widetilde{O}\left(\frac{1}{\varepsilon^{3/2}} \sqrt{w_-(x)\widetilde{W}_+} \right)$ queries.*

2.3 Graph Theory

For an undirected graph G, we let $V(G)$ denote the vertices of G, and $E(G)$ denote the edges of G. We also define $\overrightarrow{E}(G) = \{(u,v) : \{u,v\} \in E(G)\}$ We will consider the effective resistance of graphs, for which we need the concept of a unit flow:

Definition 5 (Unit Flow). *Let G be an undirected graph with $s, t \in V(G)$. Then a* unit st-flow *on G is a function $\theta : \overrightarrow{E}(G) \to \mathbb{R}$ such that:*

1. *For all $(u, v) \in \overrightarrow{E}(G)$, $\theta(u, v) = -\theta(v, u)$;*

2. *$\sum_{v \in \Gamma(s)} \theta(s, v) = \sum_{v \in \Gamma(t)} \theta(v, t) = 1$, where $\Gamma(u)$ is the neighbourhood of u in G; and*

3. *for all $u \in V(G) \setminus \{s, t\}$, $\sum_{v \in \Gamma(u)} \theta(u, v) = 0$.*

A *circulation* is a function θ that satisfies (1), and in addition, satisifes (3) for *all* vertices in $V(G)$.

Definition 6 (Unit Flow Energy). *Given a unit st-flow θ on a graph G, the* unit flow energy *is $J(\theta) = \sum_{\{u,v\} \in E} \theta(u, v)^2$.*

The effective resistance between s and t is the smallest energy of any unit st-flow:

Definition 7 (Effective Resistance). *Fix an undirected graph G, and $s, t \in V(G)$. The effective resistance is defined $R_{s,t}(G) = \infty$ if s and t are not connected in G, and otherwise, $R_{s,t}(G) = \min_\theta \sum_{\{u,v\} \in E} \theta(u, v)^2$, where θ runs over all unit st-flows in G.*

We will also look at dual graphs:

Definition 8 (Dual Graph). *Let G be a planar graph (with an implicit planar embedding). The dual graph, $G^\dagger$, is defined as follows. For every face f of G, $G^\dagger$ has a vertex v_f, and any two vertices are adjacent if their corresponding faces share an edge, e. We call the edge between two such vertices the* dual edge *to e, $e^\dagger$.*

When G is a directed graph, we assign edge directions to the dual by orienting each dual edge $\pi/2$ radians clockwise from the primal edge.

3 Average Choice Complexity

In [ZKH12], Zhan, Kimmel and Hassidim find a relationship between the difficulty of playing the two-player game associated with a NAND-tree, and the witness size. Specifically, they find that trees with a smaller number of *faults*, or critical decisions for a player playing the associated two-player game, are easier to evaluate on a quantum computer. In this section, we give a more nuanced measure of the difficulty of a two-player game, the *average choice complexity*, which we will later see is also connected with the quantum query complexity of evaluating a NAND-tree.

Zhan *et al.* measure the difficulty of a tree in terms of faults. When playing a game, a fault is a node where if the player chooses the correct direction, she can win the game if she plays optimally, but if the player chooses the wrong direction, she is no longer guaranteed to win the game, and in fact will lose if the opposing player plays optimally. Equivalently, it is a node in which one child corresponds to a 0-instance of NAND-tree, and the other child corresponds to a 1-instance. Let x be a Z-winnable input for $Z \in \{A, B\}$. Consider all paths such that Player Z wins, and also Player Z never makes a move that would allow her opponent to win. We call such paths *Z-winning paths*. Zhan *et al.* call a tree k-fault[1] if each of these paths encounters at most k faults. Then the witness size of such a tree is less than 2^k [Kim11], and we call 2^k the *fault complexity* of the tree. Fault complexity encapsulates the worst case number of critical decisions that must be made in order to win the game, on any winning path.

In our case, the average choice complexity, defined shortly, does not characterize the number of decisions in the worst case over any winning path, but is rather a characterization of the expected number of decisions required when playing against a random opponent.

---

[1]We have actually used the more refined definition of k-fault from [Kim11]
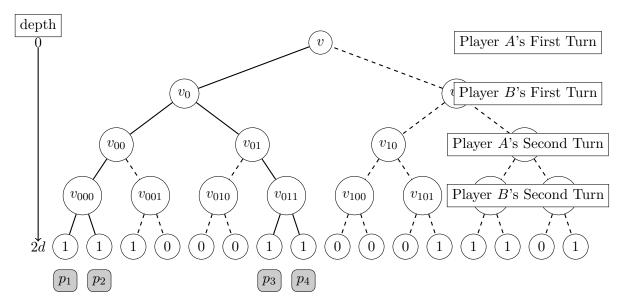
Figure 1: A winning strategy for Player A is defined by the solid lines. There are four possible paths that the winning strategy might take, labeled by p_i.

Consider an even-depth NAND-tree instance $x \in \{0, 1\}^{2^{2d}}$. A Z-strategy $\mathcal{S}(x)$ for input x is a complete deterministic proscription for how Player Z should act given any action of the opposing player. Any strategy consists of 2^d paths from the root to a leaf, which describe all possible responses to the opposing player. A Z-*winning* strategy is a Z-strategy $\mathcal{S}$ such that all paths are Z-winning paths. An example of a winning strategy for Player A and the paths of the strategy are given in Figure 1. In a valid strategy, only one choice at each of Player Z's turns must be on a path. For example, in Figure 1, at node v, to follow any of the paths, Player A must choose to go to node v_0 rather than v_1. Along any given path p, let $\nu_Z(p)$ be the set of nodes along the path at which it is Player Z's turn. Thus, $\nu_A(p)$ contains those nodes in p at even distance > 0 from the leaf, and $\nu_B(p)$ contains those nodes at odd distance from the leaf.

To each node v, we assign a criticality parameter $c(v)$ that ranges between 1 and 2, and is a representation of how important that decision is. A value of 1 signals that the decision made by the player at this node is inconsequential: loosely speaking, there are just as many ways to win in either sub-game the player could choose. However a criticality parameter of 2 signals that the decision is critical: one of the sub-games is winnable, and the other is not. In other words, any node with criticality value 2 is a fault node.

Definition 9 (Node Criticality and Average Choice Complexity). *For $x \in \{0, 1\}^{2^\ell}$ for $\ell \geq 0$ (note ℓ can be even or odd), let $\mathcal{C}_A(x)$ be the average choice complexity for Player A on input x, and $\mathcal{C}_B(x)$ be the average choice complexity for Player B. If x is A-winnable, then $\mathcal{C}_B(x) = \infty$, and if x is B-winnable, then $\mathcal{C}_A(x) = \infty$. For a depth-0 tree, we define $\mathcal{C}_A(1) = 1$ and $\mathcal{C}_B(0) = 1$. For depth ℓ trees with $\ell > 0$, we define average choice complexity, $\mathcal{C}_Z$, for $Z \in \{A, B\}$ recursively in terms of the* criticality, $c(v)$, *defined shortly:*

$$\mathcal{C}_Z(x) = \begin{cases} \min_{\mathcal{S} \in \mathscr{S}_Z(x)} \mathbb{E}_{p \in \mathcal{S}} \left[\prod_{v \in \nu_Z(p)} c(v) \right] & \text{if } x \text{ is } Z\text{-winnable} \\ \infty & \text{else,} \end{cases} \tag{2}$$

where $\mathscr{S}_Z(x)$ is the set of all Z-winning strategies.

7

We now define $c(v)$. As a base case, let v be a node whose children are leaves — that is, it is the root of a depth-1 subtree — with children labeled $x^0, x^1 \in \{0, 1\}$. Then define:

$$c(v) = \begin{cases} 1 & \text{if } x^0 = x^1 \\ 2 & \text{if } x^0 \neq x^1. \end{cases} \tag{3}$$

Let v be a node that is the root of a depth-ℓ subtree for $\ell > 1$. If ℓ is even, fix $Z' = A$, and if ℓ is odd, fix $Z' = B$. If v is the root of a tree that is not Z'-winnable, then we define $c(v) = 1$. If v is a fault, then we define $c(v) = 2$. Otherwise, let x^{00}, x^{01}, x^{10}, and x^{11} be the respective inputs to the subtrees rooted at v_{00}, v_{01}, v_{10}, and v_{11}, the four grandchildren of v. Then $\mathcal{C}_{Z'}(x^{00}), \mathcal{C}_{Z'}(x^{01}), \mathcal{C}_{Z'}(x^{10}), \mathcal{C}_{Z'}(x^{11})$ are all finite, and we define:

$$c(v) = \frac{\max\left\{\mathcal{C}_{Z'}(x^{00}) + \mathcal{C}_{Z'}(x^{01}), \mathcal{C}_{Z'}(x^{10}) + \mathcal{C}_{Z'}(x^{11})\right\}}{\text{avg}\left\{\mathcal{C}_{Z'}(x^{00}) + \mathcal{C}_{Z'}(x^{01}), \mathcal{C}_{Z'}(x^{10}) + \mathcal{C}_{Z'}(x^{11})\right\}}. \tag{4}$$

Note that exactly one of $\mathcal{C}_A(x)$ and $\mathcal{C}_B(x)$ is finite. Finally, we define:

$$\mathcal{C}(x) = \min\{\mathcal{C}_A(x), \mathcal{C}_B(x)\}. \tag{5}$$

The criticality should be thought of as measuring the difference between making a random choice at node v, and making a good choice at node v, with $c(v) = 1$ indicating that the two choices at v are equal. The expressions $\mathcal{C}_{Z'}(x^{00}) + \mathcal{C}_{Z'}(x^{01})$ and $\mathcal{C}_{Z'}(x^{00}) + \mathcal{C}_{Z'}(x^{01})$ are proportional to the average complexity (averaged over the opposing player's next decision) faced by Player Z' in each of the respective paths she might take. If these two values are nearly the same, it does not matter so much which path Player Z' takes at this turn, and $c(v)$ is close to 1. If these two values are very different, then $c(v)$ approaches 2. The criticality parameter is always in the range $[1, 2]$.

In Section 5, we further motivate the average choice complexity by showing that it is related to the quantum query complexity of winning a two-player NAND-tree game. We now compare the average choice complexity to fault complexity. If we let $\bar{c}(v)$ assign a value of 2 to every fault node, and 1 to every non-fault, and take the max over all Z-winning paths p, we recover the fault complexity of Zhan *et al.*:

$$\mathcal{F}_Z(x) = \max_{p \in P} \prod_{v \in \nu_Z(p)} \bar{c}(v) = \max_{p \in P} 2^{k_p}, \tag{6}$$

where x is Z-winnable, P runs over all Z-winning paths, and k_p is the number of faults in $\nu_Z(p)$. When x is not Z-winnable, $\mathcal{F}_Z(x) = \infty$. Then the fault complexity is $\mathcal{F}(x) = \min\{\mathcal{F}_A(x), \mathcal{F}_B(x)\}$.

We prove in Lemma 3 that $\mathcal{C}(x) \leq \mathcal{F}(x)$. In contrast to the setting of fault complexity, Definition 9 allows the criticality parameter to range between 1 and 2, only considers paths in a certain strategy, and takes an average over those paths rather than looking at the maximum. In the worst case, when every winning path has exactly k faults, and every non-fault node on every winning path has criticality 1, we have $\mathcal{C}(x) = \mathcal{F}(x)$, but in general, $\mathcal{C}(x)$ may be significantly smaller.

4 Evaluating NAND-Trees with low Average Choice Complexity

In this section, we generalize the speedup of Zhan *et al.* by proving the following theorem.

Theorem 3. *Fix $W = W(d)$. For all d, let $X_d \subseteq \{0, 1\}^{2^{2d}}$ be the set of instances such that $\mathcal{C}(x) \leq W$. Then the bounded error quantum query complexity of evaluating* NAND*-trees in $X = \bigcup_d X_d$ is $O(W)$.*

Let k be the largest integer such that $2^k \leq W$. Then by Lemma 3, X contains the set of k-fault trees. Thus, the lower bound of $\Omega((\log \log N - \log k)^k)$ on the classical query complexity of k-fault trees from [ZKH12] implies a lower bound of $\Omega((\log \log N - \log \log W)^{\log W})$ on evaluating NAND-trees in X, making Theorem 3 a superpolynomial speedup as well.

To prove Theorem 3, we describe in Section 4.1 how NAND-tree evaluation is equivalent to solving an st-connectivity problem on certain graphs, $G_d(x)$. In Lemma 3, we show that $\mathcal{C}_A(x) = R_{s,t}(G_d(x))$ and $\mathcal{C}_B(x)$ is the effective resistance on the dual-complement of $G_d(x)$. In Section 4.2, we present a span program whose positive witness sizes correspond to $R_{s,t}(G_d(x))$ (Lemma 4), and negative witness sizes correspond to the effective resistance on the dual complement (Lemma 5), completing the proof.

In this section, we also prove the following theorem, which will be used in Section 5:

Theorem 4. *Let $Z \in \{A, B\}$. The bounded error quantum query complexity of estimating the average choice complexity for Player Z of a NAND-tree instance $x \in \{0, 1\}^N$ to relative accuracy ε is $\widetilde{O}\left(\frac{1}{\varepsilon^{3/2}}\sqrt{\mathcal{C}_Z(x)}N^{1/4}\right)$.*

4.1   NAND-Trees and st-Connectivity

In this section, we present a useful relationship between the even-depth NAND-tree evaluation problem and the st-connectivity problem on certain graphs. Let G_0 be a graph on 2 vertices, labelled s and t, with a single edge. For $d > 0$, let G_d be the graph obtained from G_{d-1} by replacing each edge with a 4-cycle — or more specifically, replacing an edge $\{u, v\}$ with two paths of length 2 from u to v. The first three such graphs are illustrated in Figure 2. It is not difficult to see by recursive argument that the graph G_d has 2^{2d} edges.
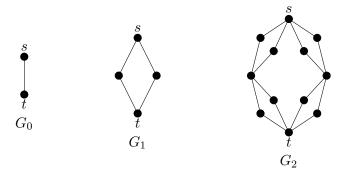

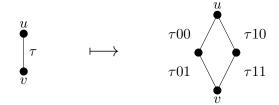Figure 2: The graphs G_0, G_1 and G_2.

Figure 3: This graph shows how to label edges of G_{d+1} given a label τ on an edge $\{u, v\} \in E(G_d)$. For example, if $\tau = 01$, the new labels will be $0100, 0101, 0110$, and 0111.
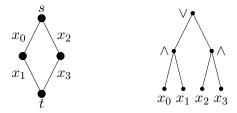
9

Figure 4: Here we show the relationship between the labeling of edges in the graph G_2, and the inputs x_i to a depth-2 NAND-tree.

Let $x \in \{0,1\}^{2^{2d}}$ be a depth-$2d$ NAND-tree instance. We can associate the 2^{2d} binary variables x_i with the edges of G_d inductively as follows. Begin with $d = 0$. G_0 has one edge, which we label with the empty string $\sigma = \emptyset$. We then associate the variable x_σ with the edge labeled by the string σ. In this case, there is only one variable $x_\emptyset$, and it is associated with the edge labeled by $\emptyset$.

For the inductive step, we have a graph G_d, for $d \geq 0$, with edges labeled by strings. We now want to label the strings of the graph G_{d+1}. To create G_{d+1} from G_d each edge of G_d is replaced by the graph G_1. Consider an edge between vertices u and v in the graph G_d that is labeled by the string τ. When the edge between u and v is replaced by a four-cycle to create G_{d+1}, we label the four edges as in Figure 3.

We can now define a subgraph $G_d(x)$ of G_d by including only those edges of G_d in which the associated input variable is true. This allows us to directly connect instances of depth-$2d$ NAND-trees to subgraphs of G_d. In Figure 4, we show explicitly how the labels of a graph correspond to the inputs to a NAND-tree for $d = 2$. Then we have the following:

Lemma 1. *For all $x \in \{0,1\}^{2^{2d}}$, x is a 1-instance of depth-$2d$ NAND-tree evaluation if and only if s and t are connected in $G_d(x)$.*

Proof. We proceed by induction. We start with the depth-0 NAND-tree. The depth-0 NAND-tree is simply the identity function. The graph $G_0(x)$ associated with this tree consists of only the vertices s and t: if $x = 1$, there is an edge between s and t, and if $x = 0$, there is no edge between s and t. Thus s and t are connected in $G_0(x)$ if and only if the depth-0 NAND-tree evaluates to 1 on x.
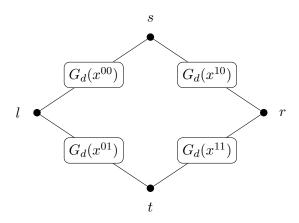

Figure 5: In the above graph we identify l with the original node t of $G_d(x^{00})$, l with the original node s of $G_d(x^{01})$, r with the original node t of $G_d(x^{10})$, and r with the original node s of $G_d(x^{11})$.

For the induction step, we suppose that the hypothesis is true for depth-$2d$ NAND-trees, and we

would like to prove it is true for depth-$2(d+1)$ NAND-trees. Now the function $\text{NAND}_{2(d+1)}$ evaluates to true if and only if

$$\left(\text{NAND}_{2d}\left(x^{00}\right) = 1 \wedge \text{NAND}_{2d}\left(x^{01}\right) = 1\right) \bigvee \left(\text{NAND}_{2d}\left(x^{10}\right) = 1 \wedge \text{NAND}_{2d}\left(x^{11}\right) = 1\right) = 1, \quad (7)$$

where x^{τ} indicates all bits of x whose labeling string begins with τ. But if any of these $\text{NAND}_{2d}\left(x^{\tau}\right) = 1$, by the induction assumption, there must be a connective path between the nodes at either end of the subgraph $G_d(x^{\tau})$ in Figure 5. This implies that s and t are connected in $G_{d+1}(x)$ if and only if $\text{NAND}_{2(d+1)}(x) = 1$. □

Dual Graphs We now define a class of graphs that relate 0-*instances* of NAND-trees to *st*-connectivity problems. Define G_0' as an edge with the endpoints labelled s' and t'. For $d \geq 1$, define G_d' recursively by replacing every edge in G_{d-1}' with the multigraph consisting of two 2-paths with the same three vertices. The first three such graphs are shown in Figure 6.
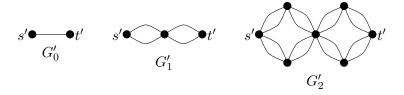

Figure 6: The graphs G_0', G_1' and G_2'.

Note that for any d, both G_d and G_d' are planar. Define $\bar{G}_d$ as a planar embedding of G_d with an additional edge $\{s, t\}$. Define $\bar{G}_d'$ as a planar embedding of G_d' with an additional edge $\{s', t'\}$. Then we have the following.

Theorem 5. $\bar{G}_d' = \bar{G}_d^{\dagger}$, the dual graph of $\bar{G}_d$.

Proof. We will prove the theorem by induction on d. For $d = 0$, both $\bar{G}_0$ and $\bar{G}_0'$ are just two edges with the same endpoints, and this graph is self-dual.

Note that for all $d \geq 1$, $\bar{G}_d$ has an edge $\{s, t\}$, and $\bar{G}_d'$ has an edge $\{s', t'\}$. These edges will always be dual. Suppose $\bar{G}_{d-1}' = \bar{G}_{d-1}^{\dagger}$. Fix some edge $e = \{u, v\} \in E(\bar{G}_{d-1})$ other than $\{s, t\}$. To get to $\bar{G}_d$, this edge is replaced by 4 edges, e_0, e_1, e_2, e_3 as in Figure 7 (a).

By the induction hypothesis, e has a dual edge $e^{\dagger}$ in $E(\bar{G}_{d-1}')$. To obtain $\bar{G}_d'$ from $\bar{G}_{d-1}'$, we replace $e^{\dagger} = \{u', v'\}$ with 4 edges, e_0', e_1', e_2', e_3' as in Figure 7 (b). Replacing e has introduced two new vertices, and one new face, whereas replacing $e^{\dagger}$ has introduced two new faces and one new vertex. We identify them as in Figure 7 (c). We can thus see that $e_b' = e_b^{\dagger}$ for all $b \in \{0, 1, 2, 3\}$. □
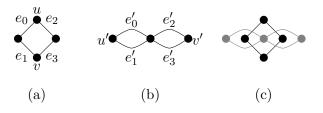

(a) (b) (c)

Figure 7: (a) shows a subgraph of G_d corresponding to an edge $e = \{u, v\}$ of G_{d-1}. (b) shows the subgraph of G_d' corresponding to the dual edge of e, $e^{\dagger} = \{u', v'\}$, in G_{d-1}', (u', v'). (c) shows how the two subgraphs are dual.
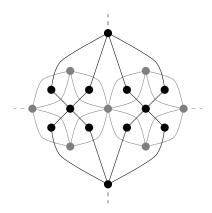
Figure 8: $\bar{G}_2$ and its dual $\bar{G}'_2$.

In Figure 8, we show how $\bar{G}_d$ and $\bar{G}'_d$ are dual for for $d = 2$.

Recall that each edge $e \in E(G_d)$ corresponds to some input variable x_e. Thus every edge $e \in E(G'_d)$ corresponds to an input variable $x_{e\dagger}$. For any instance $x \in \{0,1\}^{2^{2d}}$, we define a subgraph of G'_d, $G'_d(x)$, by including only those edges of G'_d for which the corresponding variable $x_{e\dagger}$ is 0. Just as $\text{NAND}_{2d}(x) = 1$ if and only if s and t are connected in $G_d(x)$, we now show $\text{NAND}_{2d}(x) = 0$ if and only if s' and t' are connected in $G'_d(x)$.

Lemma 2. *For any $x \in \{0,1\}^{2^{2d}}$, $\text{NAND}_{2d}(x) = 0$ if and only if s' and t' are connected in $G'_d(x)$.*

Proof. The proof is very similar to the proof of Lemma 1, so we merely sketch the argument. Define $\overline{\text{NAND}}_{2d}$ as the formula on a full binary tree of depth $2d$ in which we label vertices at even distance from the leaves by $\wedge$, and those at odd distance from the leaves by $\vee$ (the opposite of a NAND-tree). First, we have by De Morgan's Law, $\text{NAND}_{2d}(x) = 0$ if and only if $\overline{\text{NAND}}_{2d}(\bar{x}) = 1$, where $\bar{x}$ is the entrywise complement of x. A simple inductive proof shows that $\bar{x}$ is a 1-instance of $\overline{\text{NAND}}_{2d}$ if and only if s' and t' are connected in $G'_d(x)$. □

Connection to Average Choice Complexity We have described a connection between evaluating NAND-trees and solving st-connectivity problems. We will now connect the effective resistance of these graphs to average choice complexity, defined in Section 3.

Lemma 3. *For any instance of NAND-tree, $x \in \{0,1\}^{2^{2d}}$, $\mathcal{C}_A(x) = R_{s,t}(G_d(x))$ and $\mathcal{C}_B(x) = R_{s',t'}(G'_d(x))$. Furthermore, for all x, $\mathcal{C}(x) \leq \mathcal{F}(x)$, where $\mathcal{F}$ is the fault complexity.*

Proof. We will give a proof for $\mathcal{C}_A$, as the case of $\mathcal{C}_B$ is similar.

First, $R_{s,t}(G_d(x)) = \infty$ if and only if s and t are not connected in $G_d(x)$, if and only if x is a 0-instance, if and only if $\mathcal{C}_A(x) = \infty$. Thus, suppose this is not the case, so $\mathcal{C}(x) = \mathcal{C}_A(x) < \infty$.

The rest of the proof is by induction. For the case of $d = 0$, we have to consider the only A-winnable input in $\{0,1\}^{2^0}$, $x = 1$. In that case, we have $\mathcal{C}_A(x) = 1$, and since $G_0(x)$ is just a single edge from s to t, $R_{s,t}(G_0(x)) = 1$. We also have $\mathcal{F}(x) = 1$, since there are no choices, so there are no faults.

Let $x \in \{0,1\}^{2^{2(d+1)}}$ be any A-winnable input. Let $\mathscr{S}_A(x)$ be the set of all A-winning strategies on input x, and let $\mathcal{S}$ be a strategy in $\mathscr{S}_A(x)$. Let $b \in \{0,1\}$ be the choice of $\mathcal{S}$ at the root. Then $\mathcal{S}$ can be described recursively by b and a pair of winning sub-strategies, $\mathcal{S}^0 \in \mathscr{S}_A(x^{b0})$ and

$\mathcal{S}^1 \in \mathscr{S}_A(x^{b1})$, one for each of the possible first choices of Player B. If r is the root, we have:

$$
\begin{aligned}
\mathcal{C}_A(x) &= \min_{\mathcal{S} \in \mathscr{S}_A(x)} \mathbb{E}_{p \in \mathcal{S}} \left[\prod_{v \in \nu_A(p)} c(v) \right] \\
&= \min_{b \in \{0,1\}} \min_{\mathcal{S}^0 \in \mathscr{S}_A(x^{b0}), \mathcal{S}^1 \in \mathscr{S}_A(x^{b1})} \frac{1}{2} \left(\mathbb{E}_{p \in \mathcal{S}^0} \left[c(r) \prod_{v \in \nu_A(p)} c(v) \right] + \mathbb{E}_{p \in \mathcal{S}^1} \left[c(r) \prod_{v \in \nu_A(p)} c(v) \right] \right) \\
&= \frac{1}{2} c(r) \min_{b \in \{0,1\}} \{ \mathcal{C}_A(x^{b0}) + \mathcal{C}_A(x^{b1}) \}.
\end{aligned} \tag{8}
$$

We first consider the case that r is a fault, so $c(r) = 2$. We first show $\mathcal{C}_A(x) \leq \mathcal{F}(x)$. Using the induction hypothesis,

$$
\mathcal{C}_A(x) \leq \min_{b \in \{0,1\}} \{ \mathcal{F}_A(x^{b0}) + \mathcal{F}_A(x^{b1}) \} \leq \min_b \max_{b'} 2\mathcal{F}_A(x^{bb'}) = \mathcal{F}(x). \tag{9}
$$

Also by the induction hypothesis:

$$
\mathcal{C}_A(x) = \min_{b \in \{0,1\}} \{ R_{s,t}(G_d(x^{b0})) + R_{s,t}(G_d(x^{b1})) \}. \tag{10}
$$

Since r is a fault, exactly one of $\mathcal{C}_A(x^{b0}) + \mathcal{C}_A(x^{b1})$ for $b \in \{0,1\}$ is finite. Without loss of generality, suppose it is $b = 0$. Then $R_{s,t}(G_d(x^{10})) + R_{s,t}(G_d(x^{11})) = \infty$, meaning that s and t are not connected in one of $G_d(x^{10})$ and $G_d(x^{11})$. Referring to Figure 5, which shows how $G_{d+1}(x)$ is be constructed from $G_d(x^{00})$, $G_d(x^{01})$, $G_d(x^{10})$ and $G_d(x^{11})$, since resistances in series add, we have $R_{s,t}(G_{d+1}(x)) = R_{s,t}(G_d(x^{00})) + R_{s,t}(G_d(x^{01})) = \mathcal{C}_A(x)$, as desired.

Now suppose that r is not a fault. Continuing from (8), we have:

$$
\begin{aligned}
\mathcal{C}_A(x) &= \frac{1}{2} \frac{\max\{\mathcal{C}(x^{00}) + \mathcal{C}(x^{01}), \mathcal{C}(x^{10}) + \mathcal{C}(x^{11})\}}{\operatorname{avg}\{\mathcal{C}(x^{00}) + \mathcal{C}(x^{01}), \mathcal{C}(x^{10}) + \mathcal{C}(x^{11})\}} \min_{b \in \{0,1\}} \{ \mathcal{C}(x^{b0}) + \mathcal{C}(x^{b1}) \} \\
&= \frac{(\mathcal{C}(x^{00}) + \mathcal{C}(x^{01}))(\mathcal{C}(x^{10}) + \mathcal{C}(x^{11}))}{\mathcal{C}(x^{00}) + \mathcal{C}(x^{01}) + \mathcal{C}(x^{10}) + \mathcal{C}(x^{11})}.
\end{aligned} \tag{11}
$$

Since v is not a fault, we have $\mathcal{F}(x) = \max_{b,b' \in \{0,1\}} \mathcal{F}(x^{bb'})$. Using (11) and the induction hypothesis, we get:

$$
\mathcal{C}_A(x) \leq \frac{1}{2} \max_{b \in \{0,1\}} \{ \mathcal{C}_A(x^{b0}) + \mathcal{C}_A(x^{b1}) \} \leq \frac{1}{2} \max_{b \in \{0,1\}} \{ \mathcal{F}(x^{b0}) + \mathcal{F}(x^{b1}) \} \leq \mathcal{F}(x). \tag{12}
$$

By the induction hypothesis and (11), we get:

$$
\mathcal{C}_A(x) = \frac{(R_{s,t}(G_d(x^{00})) + R_{s,t}(G_d(x^{01})))(R_{s,t}(G_d(x^{10})) + R_{s,t}(G_d(x^{11})))}{R_{s,t}(G_d(x^{00})) + R_{s,t}(G_d(x^{01})) + R_{s,t}(G_d(x^{10})) + R_{s,t}(G_d(x^{11}))}. \tag{13}
$$

To complete the proof, we refer to Figure 5, which shows how $G_{d+1}(x)$ can be constructed from $G_d(x^{00})$, $G_d(x^{01})$, $G_d(x^{10})$ and $G_d(x^{11})$. Since resistances in series add, and inverse resistances in parallel add, we have:

$$
\begin{aligned}
\frac{1}{R_{s,t}(G_{d+1}(x))} &= \frac{1}{R_{s,t}(G_d(x^{00})) + R_{s,t}(G_d(x^{01}))} + \frac{1}{R_{s,t}(G_d(x^{01})) + R_{s,t}(G_d(x^{11}))} \\
&= \frac{R_{s,t}(G_d(x^{00})) + R_{s,t}(G_d(x^{01})) + R_{s,t}(G_d(x^{00})) + R_{s,t}(G_d(x^{01}))}{R_{s,t}(G_d(x^{00})) + R_{s,t}(G_d(x^{01}))(R_{s,t}(G_d(x^{00})) + R_{s,t}(G_d(x^{01})))},
\end{aligned} \tag{14}
$$

so $\mathcal{C}_A(x) = R_{s,t}(G_{d+1}(x))$.

A similar analysis for $\mathcal{C}_B$ completes the proof. $\qquad\square$

4.2 Span Program for NAND-Trees

We use the relationship between st-connectivity on G_d and depth-$2d$ NAND-tree evaluation to define a span program for NAND-tree evaluation. It is nearly identical to a span program for st-connectivity given in [BR12] (see also [IJ15, Section 4]), except that we exploit the fact that we are only considering subgraphs of G_d, so not all edges are possible. We call the following span program P_{G_d}:

$$H_{e,0} = \{0\}, \quad H_{e,1} = \text{span}\{|e\rangle\}, \quad H = \text{span}\{|e\rangle : e \in \overrightarrow{E}(G_d)\}, \quad V = \text{span}\{|u\rangle : u \in V(G_d)\}$$

$$\tau = |s\rangle - |t\rangle, \quad A = \sum_{(u,v) \in \overrightarrow{E}(G_d)} (|u\rangle - |v\rangle)\langle u, v|.$$

We have used e in $H_{e,1}$ as a short-hand for the input variable associated with e.

Lemma 4. *Let $x \in \{0,1\}^{E(G_d)}$ be a 1-instance of* NAND*-tree evaluation, and let $G_d(x)$ be the associated subgraph of G_d. Then $w_+(x, P_{G_d}) = \frac{1}{2}R_{s,t}(G_d(x))$.*

Proof. Let $P_K = (\tilde{H}, \tilde{V}, \tilde{\tau}, \tilde{A})$ be the span program for st-connectivity on any $|V(G_d)|$-vertex graph, defined in [IJ15, Section 4]. Then it is easy to see that $\tau = \tilde{\tau}$, and $\tilde{A}\Pi_{\tilde{H}(G_d(x))} = A\Pi_{H(x)}$, so $w_+(x, P_{G_d}) = w_+(G_d(x), P_K)$. By [IJ15, Lemma 4.1], $w_+(G_d(x), P_K) = \frac{1}{2}R_{s,t}(G_d(x))$. $\square$

We now show that the negative witness size of the span program for st-connectivity on $G_d(x)$ is given by the effective resistance of the dual graph $G'_d(x)$:

Lemma 5. *Let $x \in \{0,1\}^{E(G_d)}$ be a 0-instance of* NAND*-tree, and let $G'_d(x)$ be the associated subgraph of G'_d. Then $w_-(x, P_{G_d}) = 2R_{s',t'}(G'_d(x))$.*

Proof. The proof exploits the duality between an st-path and an st-cut.

For $x \in \{0,1\}^{2^{2d}}$, define $\bar{G}_d(x)$ (respectively $\bar{G}'_d(x)$) as a planar embedding of $G_d(x)$ (resp. $G'_d(x)$) with an additional edge $\{s,t\}$ (resp. $\{s',t'\}$). We first show that a negative witness for x in P_{G_d} corresponds to a unit $s't'$-flow in $G'_d(x)$ whose energy equals twice the negative witness size. Fix a negative witness ω for x, and define a function $\theta : \overrightarrow{E}(\bar{G}'_d) \to \mathbb{R}$ by $\theta((u,v)^\dagger) = \omega(u) - \omega(v)$. Then clearly we have $\theta(u',v') = -\theta(v',u')$ for all $\{u',v'\} \in E(\bar{G}'_d)$. Since ω is a negative witness for x, $\sum_{\{u,v\} \in E(G_d(x))} (\omega(u) - \omega(v))^2 = 0$, so whenever $\{u,v\} \in E(G_d(x))$, $\omega(u) = \omega(v)$, and thus $\theta((u,v)^\dagger) = 0$. This happens precisely when $\{u,v\}^\dagger \notin E(G'_d(x))$. Thus θ is only nonzero on the edges of $\bar{G}'_d(x)$.
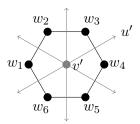

Figure 9: The dualtiy between a cycle and a star.

Next, we notice that for any $v' \in V(\bar{G}'_d)$, by the definition of the dual graph, the nodes u' such that $\{u',v'\} \in \bar{G}'_d$ correspond to edges around a face, $f_{v'}$, of $\bar{G}_d$ (see Figure 9). If $(w_1, \ldots, w_k = w_1)$ are the nodes in $\bar{G}_d$ forming the cycle around the face $f_{v'}$, then

$$0 = \sum_{i=1}^{k}(\omega(w_i) - \omega(w_{i+1})) = \sum_{i=1}^{k}\theta((w_i, w_{i+1})^\dagger) = \sum_{u':\{v',u'\} \in E(\bar{G}'_d)} \theta(v',u') = \sum_{u':\{v',u'\} \in E(\bar{G}'_d(x))} \theta(v',u').$$

$$(15)$$

Thus, θ is a circulant on $\bar{G}'_d(x)$, so if we define θ' so that $\theta'(s', t') = 0$, and $\theta'(v', u') = \theta(v', u')$ when $\{v', u'\} \neq \{s', t'\}$, then θ' is an $s't'$-flow on $G'_d(x)$. Furthermore, since $\theta(s', t') = \omega(s) - \omega(t) = 1$ (since $\omega\tau = 1$), θ' is a *unit* flow. The energy of θ' is:

$$\sum_{\{v',u'\}\in E(G'_d(x))} \theta(v',u')^2 = \frac{1}{2} \sum_{(v',u')\in \overrightarrow{E}(G'_d)} \theta(v',u')^2 = \frac{1}{2} \sum_{(v',u')\in \overrightarrow{E}(G_d)} (\omega(v') - \omega(u'))^2 = \frac{1}{2} \|\omega A\|^2,$$
(16)

where the factor of $\frac{1}{2}$ comes from having to sum each edge twice (once for each direction). Thus $w_-(x, P_{G_d}) \geq 2R_{s',t'}(G'_d(x))$.

Next we show that every unit $s't'$-flow in G'_d corresponds to a negative witness for x in P whose witness size is equal to half the energy of the flow. Let θ' be a unit flow on $G'_d(x)$ from s' to t'. Let $\theta(u, v) = \theta'(u, v)$ when $\{u, v\} \neq \{s', t'\}$ and $\theta(s', t') = 1$. Since θ' is a unit $s't'$-flow, θ is a circulant on $\bar{G}'_d(x)$. It is thus also a circulant on $\bar{G}'_d$, and so it can be decomposed into a combination of cyclic-flows around the faces of $\bar{G}'_d$.

To see this decomposition, let F be the set of faces of $\bar{G}'_d$ with *clockwise* orientation, and F' is the same faces but with *counter- clockwise* orientation. Then we can write $|\theta\rangle = \sum_e \theta(e)|e\rangle$ as $|\theta\rangle = \sum_{f\in F\cup F'} \alpha_f |C_f\rangle$ where $f \in F \cup F'$ and $\alpha_f \in \mathbb{R}$, and where $|C_f\rangle = \sum_{i=1}^{k-1} |w_i, w_{i+1}\rangle$ with $f = (w_1, \ldots, w_k = w_1)$ for a set of vertices $w_i \in \bar{G}'_d$.

For a clockwise oriented face $\overrightarrow{f} \in F$, let $\overleftarrow{f}$ be the counter-clockwise orientation of the same face. Every face $\overrightarrow{f} \in F$ corresponds to a vertex $v_f \in V(\bar{G}_d)$, so define $\omega(v_f) := \frac{1}{2}(\alpha_{\overrightarrow{f}} - \alpha_{\overleftarrow{f}})$. We claim that ω is a negative witness. Let (u', v') be any directed edge in $\overrightarrow{E}(G'_d)$. Then since $\{u', v'\}$ is adjacent to two faces, (u', v') is part of one face in F, and one face in F'. Let $\overrightarrow{f}$ be the clockwise oriented face containing (u', v'), and $\overleftarrow{g}$ be the counter-clockwise oriented face containing (u', v'). Since these are the only faces containing (u', v'), we must have $\theta(u', v') = \alpha_{\overrightarrow{f}} + \alpha_{\overleftarrow{g}}$. Since $\theta(u', v') = -\theta(v', u')$, we have $\alpha_{\overrightarrow{f}} + \alpha_{\overleftarrow{g}} = -\alpha_{\overleftarrow{f}} - \alpha_{\overrightarrow{g}}$. Thus:

$$\omega(v_f) - \omega(v_g) = \frac{1}{2}(\alpha_{\overrightarrow{f}} - \alpha_{\overleftarrow{f}} - \alpha_{\overrightarrow{g}} + \alpha_{\overleftarrow{g}}) = \frac{1}{2}(\theta(u', v') - \theta(v', u')) = \theta(u', v'). \quad (17)$$

So for every edge $(u, v) \in \overrightarrow{E}(\bar{G}_d)$, $\omega(u) - \omega(v)$ is exactly the flow across $(u, v)^\dagger$. Thus for all $\{u, v\} \in E(G_d(x))$, $\{u, v\}^\dagger \notin E(G'_d(x))$, so $\omega(u) - \omega(v) = 0$. Furthermore, $\omega(s) - \omega(t) = 1$. Thus ω is a negative witness, and we have:

$$\|\omega A\|^2 = \sum_{(u,v)\in \overrightarrow{E}(G_d)} (\omega(u) - \omega(v))^2 = 2 \sum_{e^\dagger \in E(G'_d)} \theta(e^\dagger)^2 = 2J(\theta). \quad (18)$$

Because the smallest possible energy of any unit $s't'$-flow is $R_{s',t'}(G'_d(x))$, we have that $w_-(x) \leq 2R_{s',t'}(G'_d(x))$, completing the proof. $\square$

Theorem 3 now follows immediately.

Proof of Theorem 3. Let f be the problem of evaluating NAND-trees promised to have $\mathcal{C}(x) \leq W$. By Lemmas 3, 4 and 5, we have, for all 1-instances of f, $w_+(x, P_{G_d}) = \frac{1}{2}R_{s,t}(G_d(x)) = \frac{1}{2}\mathcal{C}(x) \leq \frac{1}{2}W$, and for all 0-instances of f $w_-(x, P_{G_d}) = 2R_{s',t'}(G'_d(x)) = 2\mathcal{C}(x) \leq 2W$. Thus, by Theorem 1, the bounded error quantum query complexity of f is at most $O(\sqrt{W_+(f, P_{G_D})W_-(f, P_{G_d})}) = O(\sqrt{W^2}) = O(W)$. $\square$

15

In Section 5, we will consider a different problem: winning the two-player game associated with a NAND-tree instance x against an adversary making random choices. To solve this problem, it will be useful not only to evaluate a NAND-tree, but to estimate the span program witness size, $w_+(x)$ (or $w_-(x)$). By Theorem 2, we can construct such an algorithm fom P_{G_d}. In order to analyze this algorithm's query complexity, we upper bound the approximate negative witness size of inputs to P_{G_d}:

Lemma 6. $\widetilde{W}_-(P_{G_d}) \leq 2^{d+1}$ and $\widetilde{W}_+(P_{G_d}) \leq 2^{d+1}$.

Proof. For negative inputs, $\tilde{w}_-(x) = w_-(x) = 2R_{s',t'}(G'_d(x)) \leq 2^{d+1}$, since, it is easy to see by induction, every self-avoiding $s't'$-path in G'_d has length 2^d. Thus, we limit ourselves to positive inputs.

Recall that an approximate negative witness is a function $\omega : V(G_d) \to \mathbb{R}$ that minimizes $\|\omega A \Pi_{H(x)}\|^2 = \sum_{(u,v)\in \overrightarrow{E}(G_d(x))}(\omega(u) - \omega(v))^2$ and satisfies $\omega\tau = \omega(s) - \omega(t) = 1$. Then a valid ω is a voltage induced by a unit potential difference between s and t in the resistor network $G_d(x)$ (Dirichlet's principle, or see [DS84]). Without loss of generality, we can assign ω such that $\omega(s) = 1$, $\omega(t) = 0$, and then on the component of $G_d(x)$ containing s and t, ω is the unique harmonic function with these boundary conditions, and on any other component of $G_d(x)$, ω is constant. Then the negative approximate witness size is

$$\tilde{w}_-(x) = \min_{\text{valid } \omega}\left\{\|\omega A\|^2 = \sum_{(u,v)\in\overrightarrow{E}(G_d)}(\omega(u) - \omega(v))^2\right\}. \tag{19}$$

We prove the result inductively. We start with $d = 0$. In this case, the graph consists only of nodes s and t, which must take values 1 and 0, so $\|\omega A\|^2 = (\omega(s) - \omega(t))^2 + (\omega(t) - \omega(s))^2 = 2$.

Now we look at the case of $G_{d+1}(x)$, which is formed by replacing the edges of a four cycle with graphs $G_d(x^\tau)$ as in Figure 5. For any approximate negative witness $\tilde{\omega}$ for x, let $\gamma(\tilde{\omega}) = \tilde{\omega}(l)$ and $\gamma'(\tilde{\omega}) = \tilde{\omega}(r)$. Let ω be an optimal approximate negative witness for x, and we denote $\gamma \equiv \gamma(\omega)$ $\gamma' \equiv \gamma(\omega)$. For $b, b' \in \{0, 1\}$, let $\omega_{bb'}$ be the restriction of ω to $V(G_d(x^{bb'}))$ that takes value 0 on vertices not in $V(G_d(x^{bb'}))$. Then we have:

$$\|\omega A\|^2 = \sum_{(u,v)\in\overrightarrow{E}(G_d)}(\omega_{00}(u) - \omega_{00}(v))^2 + \sum_{(u,v)\in\overrightarrow{E}(G_d)}(\omega_{01}(u) - \omega_{01}(v))^2$$
$$+ \sum_{(u,v)\in\overrightarrow{E}(G_d)}(\omega_{10}(u) - \omega_{10}(v))^2 + \sum_{(u,v)\in\overrightarrow{E}(G_d)}(\omega_{11}(u) - \omega_{11}(v))^2. \tag{20}$$

Consider ω_{00}. Since ω is a harmonic function on $G_{d+1}(x)$ with boundary conditions $\omega(s) = 1$ and $\omega(t) = 0$ that minimizes $\|\omega A\|^2$, ω_{00} is a harmonic function on $G_d(x^{00})$ with boundary conditions $\omega_{00}(s) = 1$ and $\omega_{00}(l) = \gamma$ that minimizes $\sum_{(u,v)\in\overrightarrow{E}(G_d)}(\omega_{00}(u) - \omega_{00}(v))^2$. If $\gamma = 1$, ω_{00} is necessarily constant, and so $\sum_{(u,v)\in\overrightarrow{E}(G_d)}(\omega_{00}(u) - \omega_{00}(v))^2 = 0$. Otherwise, $\frac{1}{1-\gamma}\omega_{00}$ is a harmonic function on $G_d(x^{00})$ with boundary conditions satisfying $\frac{1}{1-\gamma}\omega_{00}(s) - \frac{1}{1-\gamma}\omega_{00}(l) = 1$, minimizing $\sum_{(u,v)\in\overrightarrow{E}(G_d)}(\omega(u) - \omega(v))^2$, so $\frac{1}{1-\gamma}\omega_{00}$ is an optimal approximate negative witness for x^{00}, so by the induction hypothesis, we have:

$$\sum_{(u,v)\in\overrightarrow{E}(G_d)}\left(\frac{1}{1-\gamma}\omega_{00}(u) - \frac{1}{1-\gamma}\omega_{00}(v)\right)^2 \leq 2^{d+1}. \tag{21}$$

So for any γ,

$$\sum_{(u,v)\in \vec{E}(G_d)} (\omega_{00}(u) - \omega_{00}(v))^2 \leq (1-\gamma)^2 2^{d+1}. \tag{22}$$

By similar arguments, we have

$$\sum_{(u,v)\in \vec{E}(G_d)} (\omega_{01}(u) - \omega_{01}(v))^2 \leq \gamma^2 2^{d+1}, \tag{23}$$

$$\sum_{(u,v)\in \vec{E}(G_d)} (\omega_{10}(u) - \omega_{10}(v))^2 \leq (1-\gamma')^2 2^{d+1}, \tag{24}$$

$$\sum_{(u,v)\in \vec{E}(G_d)} (\omega_{11}(u) - \omega_{11}(v))^2 \leq (\gamma')^2 2^{d+1}. \tag{25}$$

$$\tag{26}$$

Thus, combining (22), (23), (24) and (25) with (20):

$$\|\omega A\|^2 \leq (1-\gamma)^2 2^{d+1} + \gamma^2 2^{d+1} + (1-\gamma')^2 2^{d+1} + (\gamma')^2 2^{d+1}. \tag{27}$$

We note that if r and l are in the same component as s and t, then by the harmonic property, we must have $\gamma, \gamma' \in [0,1]$, and so $(1-\gamma)^2 + \gamma^2 + (1-\gamma')^2 + (\gamma')^2 \leq 2$, and so $\|\omega A\|^2 \leq 2^{d+2}$.

Otherwise, suppose r is in a different component than s and t. In that case, ω can take any constant value on that component and still be an approximate negative witness, so we choose $\tilde{\omega} = 0$ on the component containing r and $\tilde{\omega} = \omega$ everywhere else. In that case, $\gamma'(\tilde{\omega}) = 0$, so using arguments similar to those above, we have $\|\tilde{\omega}A\|^2 \leq 2^{d+2}$. Finally, since ω is an optimal approximate negative witness, $\|\omega A\|^2 \leq \|\tilde{\omega}A\|^2$. A similar argument works for the case when l is in a different component.

The proof that $\tilde{w}_+(x) \leq 2^{d+1}$ for all $x \in \{0,1\}^{2^{2d}}$ is similar, so we omit the details. $\qquad \square$

Proof of Theorem 4. Let $x \in \{0,1\}^{2^{2d}}$. By Theorem 2, since $\mathcal{C}_B(x) = 2w_-(x)$ for all 0-instances, and $\mathcal{C}_A(x) = \frac{1}{2}w_+(x)$ for all 1-instances, we can estimate $\mathcal{C}_A(x)$ in query complexity

$$\widetilde{O}\left(\frac{1}{\varepsilon^{2/3}}\sqrt{\mathcal{C}_A(x)\widetilde{W}_-^{1/2}}\right) = \widetilde{O}\left(\frac{1}{\varepsilon^{2/3}}\sqrt{\mathcal{C}_A(x)2^d}\right) = \widetilde{O}\left(\frac{1}{\varepsilon^{2/3}}\sqrt{\mathcal{C}_A(x)}N^{1/4}\right), \tag{28}$$

since $N = 2^{2d}$, and similarly, we can estimate $\mathcal{C}_B(x)$ in query complexity $\widetilde{O}\left(\frac{1}{\varepsilon^{2/3}}\sqrt{\mathcal{C}_B(x)}N^{1/4}\right)$. Extending to odd-depth trees is a straightforward exercise. $\qquad \square$

5   The Query Complexity of Winning a NAND-Tree

Given an A-winnable input, suppose Player A can access x via queries of the form: $\mathcal{O}_x : |i,b\rangle \mapsto |i, b \oplus x_i\rangle$. We consider the number of queries needed by Player A to make decisions throughout the course of the game such that the final live node has value 1 — *i.e.* Player A wins the game — with probability $\geq 2/3$. (In this section, we focus on A-winnable trees, but the case of B-winnable trees is similar.)

Naive Strategy There is a quantum query algorithm [Rei09, Rei11a] that decides if a tree of depth $2d$ is winnable with bounded error ϵ in $O(2^d \log \frac{1}{\epsilon})$ queries. Thus, if Player A must make a decision at a node v with children v_0 and v_1, a naive strategy is to evaluate the subtrees with roots v_0 and v_1 and move to one that evaluates to 1 (if they both evaluate to 0, then since Player A's turns correspond to nodes labelled by $\vee$, v also evaluates to 0, and the tree is not A-winnable). By setting the error to $O(1/d)$ at each decision, this strategy succeeds with bounded error and costs:

$$\sum_{\ell=0}^{d-1} 2^{d-\ell} \log d = \sum_{r=1}^{d} 2^r \log d \le 2^{d+1} \log d = O(2^d \log d) = O(\sqrt{N} \log \log N). \tag{29}$$

The naive strategy does not account for the fact that some subtrees may be easier to win than others. For example, suppose one of the subtrees rooted at v_0 or v_1 has all leaves labeled by 1, whereas the other subtree has all leaves labeled by 0. In that case, the player just needs to distinguish these two very disparate cases. More generally, one of the subtrees might have a very small positive witness — i.e., it is very winnable — whereas the other has a very large witness — i.e., is not very winnable. In that case, it may save work later on if Player A chooses the more winnable subtree. Using the witness size estimation algorithm of [IJ15], we can create a strategy to quickly distinguish an easily winnable tree from a much less easily winnable (possibly unwinnable) tree.

Strategy with Witness Size Estimation We now discuss a winning algorithm that uses witness size estimation. When the tree has small average choice complexity and the opponent plays randomly, this strategy does better than the naive strategy, on average. The new strategy will be to always choose the subtree with smaller average choice complexity, unless the two subtrees are very close in average choice complexity, in which case it doesn't matter which one we choose.

We estimate the average choice complexity of both subtrees using Theorem 4. Let $\mathtt{Est}(x)$ be the algorithm from Thoerem 4 with $\varepsilon = \frac{1}{3}$. We are only concerned about which of two average choice complexities is smaller, so we do not want to wait for both iterations of $\mathtt{Est}$ to terminate. Let c be some polylogarithmic function in N such that $\mathtt{Est}(x)$ always terminates after at most $c\sqrt{\mathcal{C}_A(x)}N^{1/4}$, for all $x \in \{0,1\}^N$. We define a subroutine, $\mathtt{Select}(x^0, x^1)$, that takes two instances, x^0 and x^1, and outputs a bit b such that $\mathcal{C}_A(x^b) \le 2\mathcal{C}_A(x^{\bar{b}})$, where $\bar{b} = b \oplus 1$. $\mathtt{Select}$ works as follows. It runs $\mathtt{Est}(x^0)$ and $\mathtt{Est}(x^1)$ in parallel. If one of these programs, say $\mathtt{Est}(x^b)$, outputs some estimate w_b, then it terminates the other program after $c\sqrt{w_b}N^{1/4}$ steps. If only the algorithm running on x^b has terminated after this time, it outputs b. If both programs have terminated, it outputs a bit b such that $w_b \le w_{\bar{b}}$.

Lemma 7. *Let $x^0, x^1 \in \{0,1\}^N$ be* NAND-*tree instances such that at least one of them is a 1-instance. Let $w_{\min} = \min\{\mathcal{C}_A(x^0), \mathcal{C}_A(x^1)\}$. Then $\mathtt{Select}(x^0, x^1)$ terminates after*

$$\widetilde{O}\left(N^{1/4}\sqrt{w_{\min}}\right) \tag{30}$$

queries to x^0 and x^1 and outputs b such that $\mathcal{C}_A(x^b) \le 2\mathcal{C}_A(x^{\bar{b}})$ with bounded error.

Proof. Since at least one of x^0 and x^1 is a 1-instance, at least one of the programs will terminate. Suppose without loss of generality that $\mathtt{Est}(x^0)$ is the first to terminate, outputting w_0. Then there are two possibilities: $\mathtt{Est}(x^1)$ does not terminate after $c\sqrt{w_0}N^{1/4}$ steps, and $\mathtt{Select}$ outputs 0; or $\mathtt{Est}(x^1)$ outputs w_1 before $c\sqrt{w}N^{1/4}$ steps have passed and $\mathtt{Select}$ outputs b such that $w_b \le w_{\bar{b}}$.

Consider the first case. Suppose $\mathcal{C}_A(x^0) > 2\mathcal{C}_A(x^1)$. Then $\mathtt{Est}(x^1)$ must terminate after $c\sqrt{\mathcal{C}_A(x^1)}N^{1/4} \le \frac{1}{\sqrt{2}}c\sqrt{\mathcal{C}_A(x^0)}N^{1/4}$ steps. By assumption, we have $|w_0 - \mathcal{C}_A(x^0)| \le \varepsilon\mathcal{C}_A(x)$, so

18

$w_0 \geq (1-\varepsilon)\mathcal{C}_A(x^0) = \frac{2}{3}\mathcal{C}_A(x^0)$. Thus $\texttt{Est}(x^1)$ must terminate after $\frac{1}{\sqrt{2}}c\sqrt{\frac{3}{2}w_0}N^{1/4} < c\sqrt{w_0}N^{1/4}$ steps, which is a contradiction. Thus, $\mathcal{C}_A(x^0) \leq 2\mathcal{C}_A(x^1)$, so outputting 0 is correct. Furthermore, since we terminate after $c\sqrt{w_0}N^{1/4} = \widetilde{O}(\sqrt{\mathcal{C}_A(x^0)}N^{1/4})$ steps, and $\mathcal{C}_A(x^0) = O(\mathcal{C}_A(x^1))$, the running time is at most $\widetilde{O}\left(N^{1/4}\sqrt{w_{\min}}\right)$.

We now consider the second case, in which both programs output estimates w_0 and w_1, such that $|w_b - \mathcal{C}_A(x^b)| \leq \varepsilon\mathcal{C}_A(x^b)$ for $b = 0, 1$. Suppose $w_b \leq w_{\bar{b}}$. We then have

$$\frac{\mathcal{C}_A(x^b)}{\mathcal{C}_A(x^{\bar{b}})} \leq \frac{\mathcal{C}_A(x^b)}{w_b}\frac{w_{\bar{b}}}{\mathcal{C}_A(x^{\bar{b}})} \leq \frac{1+\varepsilon}{1-\varepsilon} = \frac{4/3}{2/3} = 2. \tag{31}$$

Thus $\mathcal{C}_A(x^b) \leq 2\mathcal{C}_A(x^{\bar{b}})$, as required. Furthermore, the running time of the algorithm is bounded by the running time of $\texttt{Est}(x^1)$, the second to terminate. We know that $\texttt{Est}(x^1)$ has running time at most $\widetilde{O}(\sqrt{\mathcal{C}_A(x^1)}N^{1/4})$ steps, and by assumption, $\texttt{Est}(x^1)$ terminated after less than $c\sqrt{w_0}N^{1/4} = \widetilde{O}(\sqrt{\mathcal{C}_A(x^0)}N^{1/4})$ steps, so the total running time is at most $\widetilde{O}\left(N^{1/4}\sqrt{w_{\min}}\right)$. $\qquad\square$

We can thus prove the main theorem of this section:

Theorem 6. *Let $x \in \{0,1\}^N$ for $N = 2^{2d}$ be an A-winnable input. At every node v where Player A makes a decision, let Player A use the $\texttt{Select}$ algorithm in the following way. Let v_0 and v_1 be the two children of v, with inputs to the respective subtrees of v_0 and v_1 given by x^0 and x^1 respectively. Then Player A moves to v_b where b is the outcome that occurs a majority of times when $\texttt{Select}(x^0, x^1)$ is run $O(\log d)$ times. Then if Player B, at his decision nodes, chooses left and right with equal probability, Player A will win the game with probability at least $2/3$, and will use $\tilde{O}(N^{1/4}\sqrt{\mathcal{C}(x)})$ queries on average, where the average is taken over the randomness of Player B's choices.*

Proof. We first note that Player A must make d choices over the course of the game. Thus we amplify Player A's probability of success by repeating $\texttt{Select}$ at each decision node $O(\log d)$ times and taking the majority. Then the probability that Player A chooses the wrong direction at any node is $O(1/d)$. By doing this we ensure that her probability of choosing the wrong direction over the course of the algorithm is $< 1/3$. From here on, we analyze the error free case.

Let v_τ be any node in the tree at distance $2k$ from the root, for $k \in \{0, \ldots, d-1\}$, with children $v_{\tau 0}$ and $v_{\tau 1}$. The node v_τ is the root of an instance of NAND-tree of depth $2(d-k)$. Because it is the root of an even-depth subtree, it is a node where Player A must make a decision. Player A runs $\texttt{Select}(x^{\tau 0}, x^{\tau 1})$, which returns $b_1 \in \{0,1\}$ such that (by Lemma 7)

$$\mathcal{C}_A(x^{\tau b_1}) \leq 2\mathcal{C}_A(x^{\tau \bar{b}_1}). \tag{32}$$

If $x^{\tau b}$ represents the bits labeling the leaves of an odd-depth subtree, the root of the subtree, $v_{\tau b}$ is a node where Player B makes a choice. Because we assume Player B chooses uniformly at random,

$$
\begin{aligned}
\mathcal{C}_A(x^{\tau b}) &= \min_{\mathcal{S} \in \mathscr{S}_A(x^{\tau b})} \mathbb{E}_{p \in \mathcal{S}}\left[\prod_{v \in \nu_A(p)} c(v)\right] \\
&= \frac{1}{2}\left(\min_{\mathcal{S} \in \mathscr{S}_A(x^{\tau b0})} \mathbb{E}_{p \in \mathcal{S}}\left[\prod_{v \in \nu_A(p)} c(v)\right] + \min_{\mathcal{S} \in \mathscr{S}_A(x^{\tau b1})} \mathbb{E}_{p \in \mathcal{S}}\left[\prod_{v \in \nu_A(p)} c(v)\right]\right) \\
&= \frac{1}{2}\left(\mathcal{C}_A(x^{\tau b0}) + \mathcal{C}_A(x^{\tau b1})\right). \tag{33}
\end{aligned}
$$

19

Thus, (32) becomes

$$\frac{1}{2}\left(\mathcal{C}_A(x^{\tau b_1 0}) + \mathcal{C}_A(x^{\tau b_1 1})\right) \leq \mathcal{C}_A(x^{\tau \bar{b}_1 0}) + \mathcal{C}_A(x^{\tau \bar{b}_1 1}). \tag{34}$$

If v_τ is a fault, then $\max\{\mathcal{C}_A(x^{\tau b_1}), \mathcal{C}_A(x^{\tau \bar{b}_1})\} = \infty$, and so we must have $\mathcal{C}_A(x^{\tau b_1}) < \mathcal{C}_A(x^{\tau \bar{b}_1}) = \infty$. Then by (8) and (33), $\mathcal{C}_A(x^\tau) = \mathcal{C}_A(x^{\tau b_1 0}) + \mathcal{C}(x^{\tau b_1 1}) = 2\mathcal{C}_A(x^{\tau b})$.

If v_τ is not a fault, by (11), we have

$$
\begin{aligned}
\mathcal{C}_A(x^\tau) &= \frac{(\mathcal{C}_A(x^{\tau b_1 0}) + \mathcal{C}_A(x^{\tau b_1 1}))(\mathcal{C}_A(x^{\tau \bar{b}_1 0}) + \mathcal{C}_A(x^{\tau \bar{b}_1 1}))}{\mathcal{C}_A(x^{\tau b_1 0}) + \mathcal{C}_A(x^{\tau b_1 1}) + \mathcal{C}_A(x^{\tau \bar{b}_1 0}) + \mathcal{C}_A(x^{\tau \bar{b}_1 1})} \\
&\geq \frac{\frac{1}{2}\left(\mathcal{C}_A(x^{\tau b_1 0}) + \mathcal{C}_A(x^{\tau b_1 1})\right)^2}{\frac{3}{2}\left(\mathcal{C}_A(x^{\tau b_1 0}) + \mathcal{C}_A(x^{\tau b_1 1})\right)} \quad \text{by (34)}.
\end{aligned} \tag{35}
$$

Thus, whether v_τ is a fault or not, we have:

$$\mathcal{C}_A(x^{\tau b}) = \frac{1}{2}\left(\mathcal{C}_A(x^{\tau b_1 0}) + \mathcal{C}_A(x^{\tau b_1 1})\right) \leq \frac{3}{2}\mathcal{C}_A(x^\tau). \tag{36}$$

Since v_τ is the root of a NAND-tree of size $N = 2^{2(d-k)}$, by Lemma 7, there exists $c \in \text{poly}(\log N)$ such that running Select at node v_τ has query complexity at most

$$c\left(2^{2(d-k)}\right)^{1/4}\sqrt{\mathcal{C}_A(x^{\tau b})} \leq c 2^{(d-k)/2}\sqrt{\frac{3}{2}\mathcal{C}_A(x^\tau)}. \tag{37}$$

We now show, by induction on k, that the expected value of this expression, in τ, is bounded from above by $c 2^{d/2}\sqrt{\frac{3}{2}\mathcal{C}_A(x)}$. For the $k = 0$ step this is clear. We next consider the inductive step.

Let $\tau b_1 b_2$, for some string $\tau \in \{0,1\}^{2k-2}$ be the sequence of choices made thus far, so b_2 is the last choice made by Player B, and b_1 the last choice made by Player A. Then the query complexity of Select for $v^{\tau b_1 b_2}$, that is, Player A's next choice, will be at most $c 2^{(d-k)/2}\sqrt{\frac{3}{2}\mathcal{C}_A(x^{\tau b_1 b_2})}$. This has expected value, in Player B's uniform random choice b_2:

$$
\begin{aligned}
&\frac{1}{2}\left(c 2^{(d-k)/2}\sqrt{\frac{3}{2}\mathcal{C}_A(x^{\tau b_1 0})} + c 2^{(d-k)/2}\sqrt{\frac{3}{2}\mathcal{C}_A(x^{\tau b_1 1})}\right) \\
&\leq \frac{c}{2}\sqrt{3} 2^{(d-k)/2}\sqrt{\frac{1}{2}\left(\mathcal{C}_A(x^{\tau b_1 0}) + \mathcal{C}_A(x^{\tau b_1 1})\right)} \quad \text{by Jensen's inequality,} \\
&= \frac{c}{2}\sqrt{3} 2^{(d-k)/2}\sqrt{\mathcal{C}_A(x^{\tau b_1})} \quad \text{by (33)} \\
&\leq \frac{c\sqrt{3}}{2\sqrt{2}} 2^{(d-(k-1))/2}\sqrt{\frac{3}{2}\mathcal{C}_A(x^\tau)} \quad \text{by (36)}.
\end{aligned} \tag{38}
$$

By the induction hypothesis, this has expected value in τ at most $c\frac{\sqrt{3}}{2} 2^{d/2}\sqrt{\frac{3}{2}\mathcal{C}_A(x)}$. Using $\sqrt{3}/2 < 1$ completes the proof that every call to Select has expected query complexity at most $c 2^{d/2}\sqrt{\frac{3}{2}\mathcal{C}_A(x)}$.

Summing over the expected query complexity at all d levels, and including success probability amplification, we have that the query complexity is

$$\tilde{O}\left(2^{d/2}\sqrt{\mathcal{C}_A(x)}\right) = \tilde{O}\left(N^{1/4}\sqrt{\mathcal{C}_A(x)}\right). \tag{39}$$

$\square$

6 Open Problems

In this work, we have introduced average choice complexity, a measure of the difficulty of winning the two-player game associated with a NAND-tree, which allowed us to generalize a superpolynomial quantum speedup of [ZKH12], and give a new quantum algorithm for winning the two-player NAND-tree game. To accomplish this, we exploited a connection between NAND-trees and st-connectivity problems on a family of graphs, and their dual graphs, which we believe may be of further interest.

One interesting problem raised by this work concerns span programs in general: when can we view span programs as solving st-connectivity problems? This could be particularly interesting for understanding when span programs are time-efficient, since the time-complexity analysis of st-connectivity span programs is straightforward (see [BR12, Section 5.3] or [IJ15, Appendix B]).

Another motivation for considering the connection between span programs and st-connectivity problems is the nice characterization of the duality between 1-instances and 0-instances given by the duality of the corresponding graphs. An important class of st-connectivity-related span programs are those arising from the learning graph framework, which provides a means of designing quantum algorithms that is much simpler and more intuitive than designing a general span program [Bel12]. A limitation of this framework is its one-sidedness with respect to 1-certificates: whereas a learning graph algorithm is designed to detect 1-certificates, a framework capable of giving optimal quantum query algorithms for any decision problem would likely treat 0- and 1-inputs symmetrically. The duality between 1- and 0-inputs in st-connectivity problems could give insights into how to extend the learning graph framework to a more powerful framework, without losing the intuition and relative simplicity.

7 Acknowledgments

References

[ACR+10] A. Ambainis, A. M. Childs, B. W. Reichardt, R. Špalek, and S. Zhang. Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. *SIAM Journal on Computing*, 39(6):2513–2530, 2010.

[Bel12] A. Belovs. Span programs for functions with constant-sized 1-certificates. In *Proceedings of the 44th Symposium on Theory of Computing (STOC 2012)*, pages 77–84, 2012.

[BR12] A. Belovs and B. W. Reichardt. Span programs and quantum algorithms for st-connectivity and claw detection. In *Proceedings of the 20th European Symposium on Algorithms (ESA 2012)*, pages 193–204, 2012.

[CCJYM09] A. M. Childs, R. Cleve, S. P. Jordan, and D. Yonge-Mallo. Discrete-query quantum algorithm for NAND trees. *Theory of Computing*, 5:119–123, 2009.

[DS84]   P. G. Doyle and J. L. Snell. *Random Walks and Electrical Networks*, volume 22 of *The Carus Mathematical Monographs*. The Mathematical Association of America, 1984.

[FGG07]   E. Farhi, J. Goldstone, and S. Gutmann. A quantum algorithm for the Hamiltonian NAND tree, 2007. `arXiv:quant-ph/0702144`.

[IJ15]   T. Ito and S. Jeffery. Approximate span programs, 2015. `arXiv:1507.00432 [quant-ph]`.

[Kim11]   S. Kimmel. Quantum adversary (upper) bound. *Chicago Journal of Theoretical Computer Science*, 2013(4), 2011.

[KW93]   M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the IEEE 8th Annual Conference on Structure in Complexity Theory*, pages 102–111, 1993.

[Rei09]   B. W. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pages 544–551, 2009. `arXiv:quant-ph/0904.2759`.

[Rei11a]   B. W. Reichardt. Reflections for quantum query algorithms. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pages 560–569. SIAM, 2011.

[Rei11b]   B. W. Reichardt. Span-program-based quantum algorithm for evaluating unbalanced formulas. In *Theory of Quantum Computation, Communication, and Cryptography (TQC 2011)*, pages 73–103. Springer, 2011.

[RŠ12]   B. W. Reichardt and R. Špalek. Span-program-based quantum algorithm for evaluating formulas. *Theory of Computing*, 8(13):291–319, 2012.

[SW86]   M. Saks and A. Wigderson. Probabilistic boolean decision trees and the complexity of evaluating game trees. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (FOCS 1986)*, pages 29–38. IEEE, 1986.

[ZKH12]   B. Zhan, S. Kimmel, and A. Hassidim. Super-polynomial quantum speed-ups for boolean evaluation trees with hidden structure. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS 2012)*, pages 249–265, New York, NY, USA, 2012. ACM.