Quantum Algorithms for Graph Connectivity and Formula Evaluation

Stacey Jeffery[1] and Shelby Kimmel[2]

[1]QuSoft and CWI, Amsterdam, the Netherlands

[2]Middlebury College, Middlebury, VT, USA

August 17, 2017

We give a new upper bound on the quantum query complexity of deciding st-connectivity on certain classes of planar graphs, and show the bound is sometimes exponentially better than previous results. We then show Boolean formula evaluation reduces to deciding connectivity on just such a class of graphs. Applying the algorithm for st-connectivity to Boolean formula evaluation problems, we match the $O(\sqrt{N})$ bound on the quantum query complexity of evaluating formulas on N variables, give a quadratic speed-up over the classical query complexity of a certain class of promise Boolean formulas, and show this approach can yield superpolynomial quantum/classical separations. These results indicate that this st-connectivity-based approach may be the "right" way of looking at quantum algorithms for formula evaluation.

1 Introduction

Deciding whether two points are connected in a network is a problem of significant practical importance. In this work, we argue that this problem, st-connectivity, is also important as a quantum algorithmic primitive.

Dürr, Heiligman, Høyer, and Mhalla designed a quantum algorithm for deciding st-connectivity that requires $O(|V|^{3/2})$ queries to the adjacency matrix of a graph on vertex set V [13]. Belovs and Reichardt later discovered an especially elegant span-program-based quantum algorithm for this problem, which is time-efficient and requires only logarithmic space [4]. Belovs and Reichardt's algorithm improves on the query complexity of Dürr et al.'s algorithm when the connecting path is promised to be short (if it exists).

Belovs and Reichardt's st-connectivity algorithm has already been adapted or been used as a subroutine for deciding other graph problems, such as detecting certain subgraphs [4], deciding whether a graph is a forest [8], and deciding whether a graph is bipartite [8].

In this work, we modify the span program algorithm used in [4], inheriting its space and time efficiency, and we restrict to deciding st-connectivity on a class of planar graphs. If the effective resistances of the set of graphs in question (and their planar duals) are small, then we find the quantum algorithm requires far fewer queries than suggested by the analysis in [4]. In fact, we obtain a polynomial to constant improvement in query complexity for some classes of graphs.

Stacey Jeffery: jeffery@cwi.nl, SJ completed parts of this work while at the Institute for Quantum Information and Matter (IQIM), Caltech

Shelby Kimmel: shelby.kimmel@gmail.com, SK completed parts of this work while at the Joint Center for Quantum Information and Computer Science (QuICS), University of Maryland

In addition to improving our understanding of the quantum query complexity of st-connectivity problems, we show that Boolean formula evaluation reduces (extremely naturally) to st-connectivity problems of the kind for which our improved analysis holds. Therefore, finding good algorithms for st-connectivity can lead to good algorithms for Boolean formula evaluation. While one might not expect that such a reduction would produce good algorithms, we find the reduction gives optimal performance for certain classes of Boolean formulas.

Boolean formula evaluation is a fundamental class of problems with wide-reaching implications in algorithms and complexity theory. Quantum speed-ups for evaluating formulas like OR [15] and the NAND-tree [14] spurred interest in better understanding the performance of quantum algorithms for Boolean formulas. This research culminated in the development of span program algorithms [23, 24], which can have optimal quantum query complexity for any problem [20]. Using span program algorithms, it was shown that $O(\sqrt{N})$ queries are sufficient for any read-once formula with N inputs [20, 22]. Classically, the query complexity of evaluating NAND-trees is $\Theta(N^{.753})$ [25] and the query complexity of evaluating arbitrary read-once formulas is $\Omega(N^{.51})$ [16].

While there are simple bounds on the quantum query complexity of total formula evaluation problems, promise versions are still not fully understood. Kimmel [19] showed that for a certain promise version of NAND-trees, called *k-fault trees*, the quantum query complexity is $O(2^k)$, while Zhan, Kimmel, and Hassidim [27] showed the classical query complexity is $\Omega((\log \frac{\log N}{k})^k)$, giving a *superpolynomial* quantum speed-up for a range of values of k. More general treatment of when promises on the inputs give superpolynomial query speed-ups can be found in [1].

Since our analysis of st-connectivity shows that graphs with small effective resistance can be decided efficiently, this in turn means that Boolean formula evaluation problems with the promise that their inputs correspond to low resistance graphs can also be evaluated efficiently. This result gives us new insight into the structure of quantum speed-ups for promise Boolean formulas.

Contributions. We summarize the main results in this paper as follows:

- Improved quantum query algorithm for deciding st-connectivity when the input is a subgraph of some graph G such that $G \cup \{\{s,t\}\}$ — G with an additional st-edge — is planar.
  - The analysis involves the effective resistance of the original graph and its planar dual.
  - We find families of graphs for which this analysis gives exponential and polynomial improvements, respectively, over the previous quantum analysis in [4].
- Algorithm for Boolean formula evaluation via reduction to st-connectivity.
  - Using this reduction, we provide a simple proof of the fact that read-once Boolean formulas with N input variables can be evaluated using $O(\sqrt{N})$ queries.
  - We show both a quadratic and a superpolynomial quantum-to-classical speed-up using this reduction, for certain classes of promise Boolean formula evaluation problems.

Open Problems. We would like to have better bounds on the classical query complexity of evaluating st-connectivity problems, as this would provide a new approach to finding separations between classical and quantum query complexity. Additionally, our reduction from Boolean formula evaluation to st-connectivity could be helpful in the design of new classical algorithms for formulas.

Another open problem concerns span programs in general: when can we view span programs as solving st-connectivity problems? This could be useful for understanding when span programs are time-efficient, since the time-complexity analysis of st-connectivity span programs is straightforward (see Appendix A.1, [4, Section 5.3], [17, Appendix B]).

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 [3]. 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. In our analysis of st-connectivity, 1-inputs and 0-inputs are on equal footing. This 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 its intuition and relative simplicity.

Organization: Section 2 provides background information. In Section 3, we describe our improved analysis of the span program algorithm for st-connectivity for subgraphs of graphs G such that $G \cup \{\{s,t\}\}$ is planar. In Section 4, we show that every formula evaluation problem is equivalent to an st-connectivity problem. In Section 5, we apply these results to promise NAND-trees, for which we are able to prove the most significant classical/quantum separation using our approach. Also in Section 5, we use these ideas to create an improved algorithm for playing the two-player game associated with a NAND-tree.

2 Preliminaries

2.1 Graph Theory

For an undirected weighted multigraph G, let $V(G)$ and $E(G)$ denote the vertices and edges of G respectively. In this work, we will only consider undirected multigraphs, which we will henceforth often refer to as *graphs*. To refer to an edge in a multigraph, we will specify the endpoints, as well as a label λ, so that an edge is written $(\{u,v\},\lambda)$. Although the label λ will be assumed to uniquely specify the edge, we include the endpoints for convenience. Let $\overrightarrow{E}(G) = \{(u,v,\lambda) : (\{u,v\},\lambda) \in E(G)\}$ denote the set of directed edges of G. For a planar graph G (with an implicit planar embedding) let $F(G)$ denote the faces of G. We call the infinite face of a planar graph the *external face*.

For any graph G with connected vertices s and t, we can imagine a fluid flowing into G at s, and traveling through the graph along its edges, until it all finally exits at t. The fluid will spread out along some number of the possible st-paths in G. Such a linear combination of st-paths is called an *st-flow*. More precisely:

Definition 1 (Unit st-flow)**.** *Let G be an undirected weighted graph with $s,t \in V(G)$, and s and t connected. Then a* unit st-flow *on G is a function $\theta : \overrightarrow{E}(G) \to \mathbb{R}$ such that:*

1. *For all $(u,v,\lambda) \in \overrightarrow{E}(G)$, $\theta(u,v,\lambda) = -\theta(v,u,\lambda)$;*
2. *$\sum_{v,\lambda:(s,v,\lambda)\in\overrightarrow{E}} \theta(s,v,\lambda) = \sum_{v,\lambda:(v,t,\lambda)\in\overrightarrow{E}} \theta(v,t,\lambda) = 1$; and*
3. *for all $u \in V(G) \setminus \{s,t\}$, $\sum_{v,\lambda:(u,v,\lambda)\in\overrightarrow{E}} \theta(u,v,\lambda) = 0$.*

Definition 2 (Unit Flow Energy)**.** *Given a unit st-flow θ on a graph G, the* unit flow energy *is*

$$J(\theta) = \sum_{(\{u,v\},\lambda)\in E(G)} \theta(u,v,\lambda)^2. \tag{1}$$

Definition 3 (Effective resistance)**.** *Let G be a graph with $s,t \in V(G)$. If s and t are connected in G, the* effective resistance *is $R_{s,t}(G) = \min_\theta J(\theta)$, where θ runs over all unit st-flows. If s and t are not connected, $R_{s,t}(G) = \infty$.*

Intuitively, $R_{s,t}(G)$ characterizes "how connected" the vertices s and t are. The more, shorter paths connecting s and t, the smaller the effective resistance.

The effective resistance has many applications. In a random walk on G, $R_{s,t}(G)|E(G)|$ is equal to the *commute time* between s and t, or the expected time a random walker starting from s takes to reach t and then return to s [2, 9]. If G models an electrical network in which each edge e of G is a unit resistor and a potential difference is applied between s and t, then $R_{s,t}(G)$ corresponds to the resistance of the network, which determines the ratio of current to voltage in the circuit (see [11]). We can extend these connections further by considering weighted edges. A *network* consists of a graph G combined with a positive real-valued *weight* function $c : E(G) \to \mathbb{R}^+$.

Definition 4 (Effective Resistance with weights)**.** *Let $\mathcal{N} = (G, c)$ be a network with $s,t \in V(G)$. The* effective resistance of $\mathcal{N}$ *is $R_{s,t}(\mathcal{N}) = \min_\theta \sum_{(\{u,v\},\lambda)\in E(G)} \frac{\theta(u,v,\lambda)^2}{c(\{u,v\},\lambda)}$, where θ runs over all unit st-flows.*

In a random walk on a network, which models any reversible Markov chain, a walker at vertex u traverses edge $(\{u,v\},\lambda)$ with probability proportional to $c(\{u,v\},\lambda)$. Then the commute time between s and t is $R_{s,t}(\mathcal{N})\sum_{e\in E(G)} c(e)$. When $\mathcal{N}$ models an electrical network in which each edge e represents a resistor with resistance $1/c(e)$, then $R_{s,t}(\mathcal{N})$ corresponds to the resistance of the network.

When G is a single edge $e = (\{s,t\},\lambda)$ with weight $c(e)$, then the resistance $R_{s,t}(G) = 1/c(e)$. When calculating effective resistance, $R_{s,t}$, we use the rule that for edges in series (i.e., a path), or more generally, graphs connected in series, resistances add. Edges in parallel, or more generally, graphs connected in parallel, follow the rule that conductances in parallel add, where the *conductance* of a graph is given by one over the resistance. (The conductance of an edge e is equal to $c(e)$, the weight of the edge.) More precisely, it is easy to verify the following:

Claim 5. *Let two networks $\mathcal{N}_1 = (G_1, c_1)$ and $\mathcal{N}_2 = (G_2, c_2)$ each have nodes s and t. If we create a new graph G by identifying the s nodes and the t nodes (i.e. connecting the graphs in parallel) and define $c : E(G) \to \mathbb{R}^+$ by $c(e) = c_1(e)$ if $e \in E(G_1)$ and $c(e) = c_1(e)$ if $e \in E(G_2)$, then*

$$R_{s,t}(G, c) = \left(\frac{1}{R_{s,t}(G_1, c_1)} + \frac{1}{R_{s,t}(G_2, c_2)}\right)^{-1}. \tag{2}$$

However, if we create a new graph G by identifying the t node of G_1 with the s node of G_2, relabeling this node $v \notin \{s,t\}$ (i.e. connecting the graphs in series) and define c as before, then

$$R_{s,t}(G, c) = R_{s,t}(G_1, c_1) + R_{s,t}(G_2, c_2). \tag{3}$$

As a bit of foreshadowing, if we let $R_{s,t}(G_1, c_1)$ and $R_{s,t}(G_2, c_2)$ take values 0, representing FALSE, or ∞, representing TRUE, then clearly (3) computes the function OR, since $0+0 = 0$, and $0 + \infty = \infty + 0 = \infty + \infty = \infty$. We also have that (2) computes the AND function, if we use $\frac{1}{0} = \infty$ and $\frac{1}{\infty} = 0$.

Definition 6 (st-cut)**.** *Given a graph G with $s, t \in V(G)$, if s and t are not connected, an* st-cut *is a function $\kappa : V(G) \to \{0, 1\}$ such that $\kappa(s) = 1$, $\kappa(t) = 0$, and $\kappa(v) - \kappa(u) = 0$ whenever $\{u, v\} \in E(G)$.*

In other words, κ defines a subset $S \subset V(G)$ such that $s \in S$, $t \notin S$, and there is no edge of G with one endpoint in S, and one endpoint in $\overline{S}$. An st-cut is a witness that s and t are in different components of G, so no path exists between s and t.

Finally, we consider dual graphs:

Definition 7 (Dual Graph)**.** *Let G be a planar graph (with an implicit embedding). The* dual *graph, $G^\dagger$, is defined as follows. For every face $f \in F(G)$, $G^\dagger$ 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, $e^\dagger$. By convention, e and $e^\dagger$ will always have the same label, so that if $e = (\{u, v\}, \lambda)$, then $e^\dagger = (\{v_f, v_{f'}\}, \lambda)$ for f and f' the faces of G on either side of the edge e.*

2.2 Span Programs and Quantum Query Algorithms

Span programs [18] were first introduced to the study of quantum algorithms by Reichardt and Špalek [24]. They have since proven to be immensely important for designing quantum algorithms in the query model.

Definition 8 (Span Program)**.** *A span program $P = (H, U, \tau, A)$ on $\{0, 1\}^N$ is made up of* **(I)** *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 $H_{j,0} + H_{j,1} = H_j$,* **(II)** *a vector space U,* **(III)** *a non-zero* target *vector $\tau \in U$, and* **(IV)** *a linear operator $A : H \to U$. 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 $H(x)$.*

Definition 9 (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:*

$$w_+(x, P) = w_+(x) = \min\{\||w\rangle\|^2 : |w\rangle \in H(x), A|w\rangle = \tau\}, \tag{4}$$

if there exists a positive witness for x, and $w_+(x) = \infty$ otherwise. Let $\mathcal{L}(U, \mathbb{R})$ denote the set of linear maps from U to $\mathbb{R}$. We call a linear map $\omega \in \mathcal{L}(U, \mathbb{R})$ 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:*

$$w_-(x, P) = w_-(x) = \min\{\|\omega A\|^2 : \omega \in \mathcal{L}(U, \mathbb{R}), \omega A\Pi_{H(x)} = 0, \omega\tau = 1\}, \tag{5}$$

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. In this way, the span program defines a partition (P_0, P_1) of $[N]$.*

For a function $f : X \to \{0, 1\}$, with $X \subseteq \{0, 1\}^N$, we say P *decides* f if $f^{-1}(0) \subseteq P_0$ and $f^{-1}(1) \subseteq P_1$. We can use P to design a quantum query algorithm that decides f, given access to the input $x \in X$ via queries of the form $\mathcal{O}_x : |i, b\rangle \mapsto |i, b \oplus x_i\rangle$.

Theorem 10 ([21])**.** *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 $O(\sqrt{W_+(f,P)W_-(f,P)})$.*

2.3 Boolean Formulas

A read-once Boolean formula can be expressed as a rooted tree in which the leaves are uniquely labeled by variables, $x_1, \dots, x_N$, and the internal nodes are labeled by gates from the set $\{\wedge, \vee, \neg\}$. Specifically, a node of degree 2 must be labeled by $\neg$ (NOT), whereas higher degree nodes are labeled by $\wedge$ (AND) or $\vee$ (OR), with the *fan-in* of the gate being defined as the number of children. The *depth* of a Boolean formula is the largest distance from the root to a leaf. We define an AND-OR *formula* (also called a *monotone formula*) as a read-once Boolean formula for which every internal node is labeled by $\wedge$ or $\vee$. Restricting to AND-OR formulas does not lose much generality, since for any formula, there is an equivalent formula in which all NOT-gates are at distance one from a leaf, and such NOT gates do not affect the query complexity of the formula. Moreover, although we only consider read-once formulas here, our techniques can be applied to more general formulas in which a single variable may label multiple leaves, since this is equivalent to a larger read-once formula with a promise on the input. Hereafter, when we refer to a *formula*, we will mean an AND-OR *read-once formula*.

In a slight abuse of notation, at times x_i will denote a Boolean variable, and at times, it will denote a bit instantiating that variable. If $x \in \{0,1\}^N$ is an instantiation of all variables labeling the leaves of a formula ϕ, then $\phi(x)$ is the value of ϕ on that input, defined as follows. If $\phi = x_i$ has depth 0, then $\phi(x) = x_i$. If ϕ has depth greater than 0, we can express ϕ recursively in terms of subformulas $\phi_1, \dots, \phi_l$, as $\phi = \phi_1 \wedge \dots \wedge \phi_l$, if the root is labeled by $\wedge$, or $\phi = \phi_1 \vee \dots \vee \phi_l$, if the root is labeled by $\vee$. In the former case, we define $\phi(x) = \phi_1(x) \wedge \dots \wedge \phi_l(x)$, and in the latter case, we define $\phi(x) = \phi_1(x) \vee \dots \vee \phi_l(x)$. A family of formulas $\phi = \phi_N$ on N variables gives rise to an evaluation problem, EVAL_ϕ, in which the input is a string $x \in \{0,1\}^N$, and the output is $\phi_N(x)$. If $\phi(x) = 0$, we say x is a 0-instance, and if $\phi(x) = 1$, x is a 1-instance. By $\phi_1 \circ \phi_2$, we mean ϕ_1 composed with ϕ_2. That is, if $\phi_1 : \{0,1\}^{N_1} \to \{0,1\}$ and $\phi_2 : \{0,1\}^{N_2} \to \{0,1\}$, then $\phi_1 \circ \phi_2 : \{0,1\}^{N_1 N_2} \to \{0,1\}$ evaluates as $\phi_1 \circ \phi_2(x) = \phi_1(\phi_2(x^1), \dots, \phi_2(x^{N_1}))$, where $x = (x^1, \dots, x^{N_1})$ for $x^i \in \{0,1\}^{N_2}$.

An important formula evaluation problem is NAND-tree evaluation. A NAND-tree is a full binary tree of arbitrary depth d — that is, every internal node has two children, and every leaf node is at distance d from the root — in which an internal node is labeled by $\vee$ if it is at even distance from the leaves, or $\wedge$ if it is at odd distance from the leaves. We use NAND_d to denote a NAND-tree of depth d. While NAND_d is sometimes defined as a Boolean formula of NAND-gates composed to depth d, we will instead think of the formula as alternating AND-gates and OR-gates — when d is even, these two characterizations are identical. An instance of NAND_d is a binary string $x \in \{0,1\}^N$, where $N = 2^d$. For example, the formula $\text{NAND}_2(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$ is a NAND-tree of depth 2. NAND_0 denotes the single-bit identity function.

A NAND_d instance $x \in \{0,1\}^{2^d}$ can be associated with a two-player game on the rooted binary tree that represents NAND_d, where the leaves take the values x_i, as in Figure 5. The game starts at the root node, which we call the current node. In each round of the game, as long as the current node is not a leaf, if the current node is at even (respectively odd) distance from the leaves, Player A (resp. Player B) chooses one of the current node's children to become the current node. When the current node is 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 the two players 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.

3 Improved Analysis of st-connectivity Algorithm

In this section, we give an improved bound on the runtime of a quantum algorithm for st-connectivity on subgraphs of G, where $G \cup \{\{s,t\}\}$ is planar.

Let $st\text{-CONN}_{G,D}$ be a problem parameterized by a family of multigraphs G, which takes as input a string $x \in D$ where $D \subseteq \{0,1\}^{E(G)}$. An input x defines a subgraph $G(x)$ of G by including the edge e if and only if $x_e = 1$. For all $x \in D$, $st\text{-CONN}_{G,D}(x) = 1$ if and only if there exists a path connecting s and t in $G(x)$. We write $st\text{-CONN}_G$ when $D = \{0,1\}^{E(G)}$. A quantum algorithm for $st\text{-CONN}_{G,D}$ accesses the input via queries to a standard quantum oracle O_x, defined $O_x|e\rangle|b\rangle = |e\rangle|b \oplus x_e\rangle$.

The authors of [4] present a quantum query algorithm for $st\text{-CONN}_G$ when G is a complete graph, which is easily extended to any multigraph G. We further generalize their algorithm to depend on some weight function $c : E(G) \to \mathbb{R}^+$ (a similar construction is also implicit in [3]). We call the following span program $P_{G,c}$:

$$\begin{aligned} &\forall e \in \overrightarrow{E}(G): \ H_{e,0} = \{0\}, \quad H_{e,1} = \mathrm{span}\{|e\rangle\}, \qquad H = \mathrm{span}\{|e\rangle : e \in \overrightarrow{E}(G)\} \\ &U = \mathrm{span}\{|u\rangle : u \in V(G)\},\ \tau = |s\rangle - |t\rangle,\ A = \sum_{(u,v,\lambda)\in \overrightarrow{E}(G)} \sqrt{c(\{u,v\},\lambda)}(|u\rangle - |v\rangle)\langle u,v,\lambda|. \end{aligned} \tag{6}$$

For any choice of weight function c, this span program decides $st\text{-CONN}_G$, but as we will soon see, the choice of c may impact the complexity of the resulting algorithm.

Using $P_{G,c}$ with $c(\{u,v\},\lambda) = 1$ for all $(\{u,v\},\lambda) \in E(G)$, the authors of Ref. [4] show that the query complexity of evaluating $st\text{-CONN}_{G,D}$ is

$$O\left(\sqrt{\max_{x\in D: s,t \text{ are connected}} R_{s,t}(G(x)) \times |E(G)|}\right). \tag{7}$$

Their analysis was for the case where G is a complete graph, but it is easily seen to apply to more general multigraphs G. In fact, it is straightforward to show that this bound can be improved to

$$O\left(\sqrt{\max_{x\in D: s,t \text{ are connected}} R_{s,t}(G(x)) \times \max_{x\in D: s,t \text{ are not connected}} (C_{s,t}(G(x)))}\right). \tag{8}$$

where

$$C_{s,t}(G(x)) = \begin{cases} \min_{\kappa:\kappa \text{ is an } st\text{-cut of } G(x)} \sum_{(\{u,v\},\lambda)\in E(G)} |\kappa(u) - \kappa(v)| & \text{if } s \text{ and } t \text{ not connected} \\ \infty & \text{otherwise.} \end{cases} \tag{9}$$

In particular, when G is a complete graph on vertex set V, with the promise that if an st-path exists, it is of length at most k, Eq. (7) gives a bound of $O\left(\sqrt{k}|V|\right)$. In the worst case, when $k = |V|$, the analysis of [4] does not improve on the previous quantum algorithm of [13], which gives a bound of $O(|V|^{3/2})$.

In this paper, we consider in particular multigraphs that are planar even when an additional st-edge is added (equivalently, there exists a planar embedding in which s and t are on the same face), as in graph G in Figure 1. (In the case of Figure 1, s and t are both on the external face.) Given such a graph G, we define three other related graphs, which we denote by $\overline{G}$, $\overline{G}^\dagger$, and G'.

We first define the graph $\overline{G}$, which is the same as G, but with an extra edge labeled by $\emptyset$ connecting s and t. We then denote by $\overline{G}^\dagger$ the planar dual of $\overline{G}$. Because every planar dual has one edge crossing each edge of the original graph, there exists an edge that is dual to $(\{s,t\},\emptyset)$, also labeled by $\emptyset$. We denote by s' and t' the two vertices at the endpoints of $(\{s,t\},\emptyset)^\dagger = (\{s',t'\},\emptyset)$. Finally, we denote by G' the graph $\overline{G}^\dagger$ except with the edge $(\{s',t'\},\emptyset)$ removed.

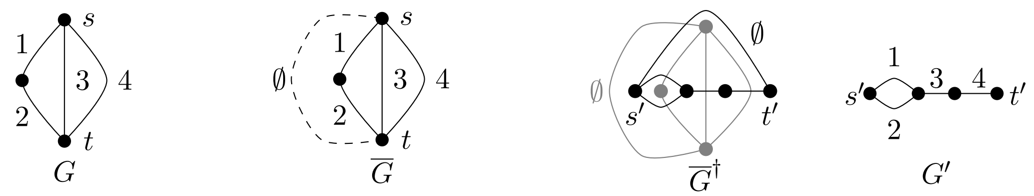

Figure 1: Example of how to derive $\overline{G}$, $\overline{G}^\dagger$, and G' from a planar graph G where s and t are on the same face. $\overline{G}$ is obtained from G by adding an edge $(\{s,t\},\emptyset)$. $\overline{G}^\dagger$ is the planar dual of $\overline{G}$. (In the diagram labeled by $\overline{G}^\dagger$, $\overline{G}$ is the gray graph, while $\overline{G}^\dagger$ is black). G' is obtained from $\overline{G}^\dagger$ by removing the edge $(\{s,t\},\emptyset)^\dagger$. Note that dual edges inherit their labels (in this case $1,2,3,4,\emptyset$) from the primal edge.

By construction, G' will always have the same number of edges as G. Then as x defines a subgraph $G(x)$ of G by including the edge e if and only if $x_e = 1$, we let $G'(x)$ be the subgraph of G' where we include the edge $e^\dagger$ if and only if $x_e = 0$.

If there is no path from s to t in $G(x)$, there must be a cut between s and t. Note that for any $e \in E(G)$, $e \in E(G(x))$ if and only if $e^\dagger \notin E(G'(x))$. Looking at Figure 1, one can convince oneself that any $s't'$-path in $G'(x)$ defines an st-cut in $G(x)$: simply define $\kappa(v) = 1$ for vertices *above* the path, and $\kappa(v) = 0$ for vertices *below* the path.

Let c be a weight function on $E(G)$. Then we define a weight function c' on $E(G')$ as $c'(e^\dagger) = 1/c(e)$. Then for every x there will be a path either from s to t in $G(x)$ (and hence $R_{s,t}(G(x),c)$) will be finite), or a path from s' to t' in $G'(x)$ (in which case $R_{s',t'}(G'(x),c')$ will be finite).

We can now state our main lemma:

Lemma 11. *Let G be a planar multigraph with $s,t \in V(G)$ such that $G \cup \{\{s,t\}\}$ is also planar, and let c be a weight function on $E(G)$. Let $x \in \{0,1\}^{E(G)}$. Then $w_+(x,P_{G,c}) = \frac{1}{2}R_{s,t}(G(x),c)$ and $w_-(x,P_{G,c}) = 2R_{s',t'}(G'(x),c')$.*

Using Lemma 11 and Theorem 10, we immediately have the following:

Theorem 12. *Let G be a planar multigraph with $s,t \in V(G)$ such that $G \cup \{\{s,t\}\}$ is also planar. Then the bounded error quantum query complexity of evaluating st-$\textsc{conn}_{G,D}$ is*

$$O\left(\min_c \sqrt{\max_{x\in D: st\text{-}\textsc{conn}_G(x)=1} R_{s,t}(G(x),c) \times \max_{x\in D: st\text{-}\textsc{conn}_G(x)=0} R_{s',t'}(G'(x),c')}\right) \tag{10}$$

where the minimization is over all positive real-valued functions c on $E(G)$.

While it might be difficult in general to find the optimal edge weighting c, any choice of c will at least give an upper bound on the query complexity. However, as we will see, sometimes the structure of the graph will allow us to efficiently find good weight functions.

The proof of Lemma 11 is in Appendix A. The positive witness result follows from generalizing the proof in [4] to weighted multigraphs. The idea is that an st-path witnesses that s and t are connected, as does any linear combination of such paths — i.e. an st-flow. The effective resistance $R_{s,t}(G(x), c)$ characterizes the size of the smallest possible st-flow.

Just as a positive witness is some linear combination of st-paths, similarly, a negative witness turns out to be a linear combination of st-cuts in $G(x)$. But as we've argued, every st-cut corresponds to an $s't'$-path in $G'(x)$. Using the correspondence between cuts and paths, we have that a negative witness is a linear combination of $s't'$-paths in $G'(x)$. This allows us to show a correspondence between complexity-optimal negative witnesses and minimal $s't'$-flows, connecting $w_-(x, P_{G,c})$ to $R_{s',t'}(G'(x), c')$.

In Appendix A.1, we show that if a quantum walk step on the network (G, c) can be implemented time efficiently, then this algorithm is not only query efficient, but also time efficient. Let

$$U_{G,c} : |u\rangle|0\rangle \mapsto \frac{1}{\sqrt{\sum_{v,\lambda:(u,v,\lambda)\in\overrightarrow{E}(G)} c(\{u,v\},\lambda)}} \sum_{v,\lambda:(u,v,\lambda)\in\overrightarrow{E}(G)} \sqrt{c(\{u,v\},\lambda)}|u\rangle|u,v,\lambda\rangle. \tag{11}$$

Then we show the following.

Theorem 13. *Let $P_{G,c} = (H, U, A, \tau)$ be defined as in (6). Let $S_{G,c}$ be an upper bound on the time complexity of implementing $U_{G,c}$. If G has the property that $G \cup \{\{s,t\}\}$ is planar, then the time complexity of deciding st-$\textsc{conn}_{G,D}$ is at most*

$$O\left(\min_c S_{G,c}\sqrt{\max_{x\in D: s,t \text{ are connected}} R_{s,t}(G(x), c) \times \max_{x\in D: s,t \text{ are not connected}} R_{s',t'}(G'(x), c')}\right). \tag{12}$$

In Appendix A.1, we also show that if the space complexity of implementing $U_{G,c}$ in time $S_{G,c}$ is $S'_{G,c}$, the algorithm referred to in Theorem 13 has space complexity at most $O(\max\{\log|E(G)|, \log|V(G)|\} + S'_{G,c})$.

3.1 Comparison to Previous Quantum Algorithm

When $G \cup \{\{s,t\}\}$ is planar, our algorithm always matches or improves on the algorithm in [4]. To see this, we compare Eqs. (10) and (8), and choose c to have value 1 on all edges of G. Then the first terms are the same in both bounds, so we only need to analyze the second term. However, using the duality between paths and cuts, we have

$$C_{s,t}(G(x)) = (\text{shortest path length from } s' \text{ to } t' \text{ in } G'(x)) \geq R_{s't'}(G'(x)). \tag{13}$$

To obtain the inequality in Eq. (13), we create an $s't'$-flow on $G'(x)$ that has value one on edges on the shortest path from s' to t' and zero on all other edges. Such a flow has unit flow energy equal to the shortest path. However, the true effective resistance can only be smaller than this, because it is the minimum energy over all possible $s't'$-flows.

We now present two simple examples where our algorithm and analysis do better than that of [4]. In the first example, we highlight how the change from $C_{s,t}(G(x))$ to $R_{s',t'}(G(x))$ in the complexity gives us an advantage for some graphs. In the second example, we show that being able to choose a non-trivial weight function c can give us an advantage for some graphs.

Let G be an st-path of length N: i.e., $N+1$ vertices arranged in a line so that each vertex is connected to its neighbors to the left and right by a single edge, and s and t are the vertices on either end of the line, as in Figure 2. For some $h \in \{1,\dots,N\}$, let $D = \{1^N\} \cup \{x \in \{0,1\}^N : |x| \leq N-h\}$, where 1^N is the all-one string of length N, and $|x|$ is the hamming weight of the string x.

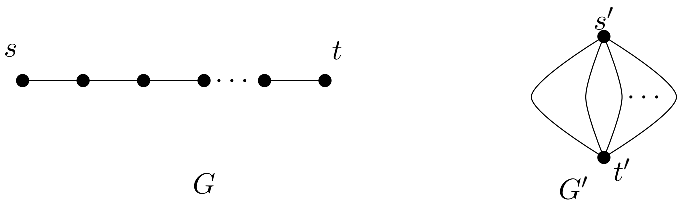

Figure 2: Example of graph for which our analysis does better than the analysis of [4], even with $c=1$ for all edges, under the promise that $G'(x)$ always contains at least h edges, if s' and t' are connected.

Then, choosing c to have value 1 on all edges of G, we have

$$\max_{x\in D: st\text{-}\textsc{conn}_G(x)=1} R_{s,t}(G(x)) = N \tag{14}$$

because the only $x \in D$ such that s and t are connected in $G(x)$ is $x = 1^N$, in which case the only unit flow has value 1 on each edge. This flow has energy N. However

$$\max_{x\in D: st\text{-}\textsc{conn}_G(x)=0} R_{s',t'}(G'(x)) \leq 1/h, \tag{15}$$

because when s and t are not connected in $G(x)$, $G(x)$ has at most $N-h$ edges, so $G'(x)$ has at least h edges. Thus we can define a unit flow with value $1/h$ on each of h parallel edges in $G'(x)$, giving an energy of $1/h$. On the other hand

$$\max_{x\in D: st\text{-}\textsc{conn}_G(x)=0} C_{s,t}(G(x)) = 1. \tag{16}$$

In fact, since $C_{s,t}(G(x))$ counts the minimum number of edges $(\{u,v\},\lambda)$ across any cut (i.e. such that $\kappa(u)=1$ and $\kappa(v)=0$), it is always at least 1, for *any* $G(x)$ in which an st-cut exists, whereas $R_{s',t'}(G'(x))$ can be as small as $1/N$ for some G.

Choosing $h = \sqrt{N}$ in our example, and applying Eqs. (8) and (10), the analysis in [4] gives a query complexity of $O(N^{1/2})$ while our analysis gives a query complexity of $O(N^{1/4})$. In Section 4 we will show that this bound is tight.

Now consider the graph G in Figure 3. It consists of N edges in a line, connecting vertices $s, u_1, \dots, u_N$, and then N multi-edges between u_N and t. We assign weights $c(e)=1$ for edges e on the path from s to u_N, and $c(e) = N^{-1}$ for all other edges.

Then,

$$\max_{x\in D: st\text{-}\textsc{conn}_G(x)=1} R_{s,t}(G(x),c) = 2N, \tag{17}$$

which occurs when only one of the multi-edges between u_N and t is present. In that case, the N edges $\{s,u_1\},\{u_1,u_2\},\dots,\{u_{N-1},u_N\}$ each contribute 1 to the effective resistance, and the final edge between u_N and t contributes $\frac{1}{c(e)} = N$. Also

$$\max_{x\in D: st\text{-}\textsc{conn}_G(x)=0} R_{s',t'}(G'(x),c') \leq 1, \tag{18}$$

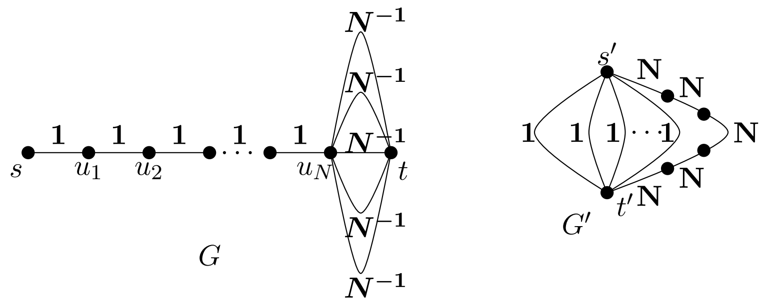

Figure 3: Example of graph for which our analysis does quadratically better than the analysis of [4] by taking advantage of a non-trivial weight function c. The values of c for each edge of G, and of c' for each edge of G', are shown in boldface.

where the maximum occurs when there is only one path from s' to t'. (If it is the path with N edges, each edge has weight N, and so contributes $1/N$ to the flow energy.) However

$$\max_{x\in D: st\text{-}\textsc{conn}_G(x)=0} C_{s,t}(G(x)) = N \tag{19}$$

for a cut across the multi-edges between u_N and t, and

$$\max_{x\in D: st\text{-}\textsc{conn}_G(x)=1} R_{s,t}(G(x)) = N+1, \tag{20}$$

which occurs when only one of the multi-edges between u_N and t is present.

Thus, the analysis in [4] gives a query complexity of $O(N)$ while our analysis gives a query complexity of $O(N^{1/2})$.

In Section 5 we will give an example where our analysis provides an exponential improvement over the analysis in [4].

4 AND-OR Formulas and st-Connectivity

In this section, we present a useful relationship between AND-OR formula evaluation problems and *st*-connectivity problems on certain graphs. As mentioned in Section 2, for simplicity we will restrict our analysis to read-once formulas, but the algorithm extends simply to "read-many" formulas. In this case, we will primarily be concerned with the query complexity: the input $x = (x_1, \dots, x_N)$ to a formula will be given via a standard quantum oracle O_x, defined $O_x|i\rangle|b\rangle = |i\rangle|b \oplus x_i\rangle$.

Given an AND-OR formula ϕ with N variables, we will recursively construct a planar multigraph G_ϕ, such that G_ϕ has two distinguished vertices labeled by s and t respectively, and every edge of G_ϕ is uniquely labeled by a variable $\{x_i\}_{i\in[N]}$. If $\phi = x_i$ is just a single variable, then G_ϕ is just a single edge with vertices labeled by s and t, and edge label x_i. That is $E(G_\phi) = \{(\{s,t\}, x_i)\}$ and $V(G_\phi) = \{s,t\}$.

Otherwise, suppose $\phi = \phi_1 \wedge \dots \wedge \phi_l$. Then G_ϕ is the graph obtained from the graphs $G_{\phi_1}, \dots, G_{\phi_l}$ by identifying the vertex labeled t in G_{ϕ_i} with the vertex labeled s in $G_{\phi_{i+1}}$, for all $i = 1, \dots, l-1$, and labeling the vertex labeled s in G_{ϕ_1} by s, and the vertex labeled t in G_{ϕ_l} by t. That is, we connect the graphs $G_{\phi_1}, \dots, G_{\phi_l}$ *in series*, as in Figure 4. (For a formal definition of G_ϕ, see Appendix B).

The only other possibility is that $\phi = \phi_1 \vee \dots \vee \phi_l$. In that case, we construct G_ϕ by starting with $G_{\phi_1}, \dots, G_{\phi_l}$ and identifying all vertices labeled by s, and labeling the resulting vertex with s, and identifying all vertices labeled by t, and labeling the resulting vertex by t. That is, we connect $G_{\phi_1}, \dots, G_{\phi_l}$ *in parallel* (see Figure 4). We note that graphs constructed in this way are exactly the set of *series-parallel* graphs with two terminals (see e.g. [26, Def. 3]), and are equivalent to graphs without a K_4 minor [10, 12].

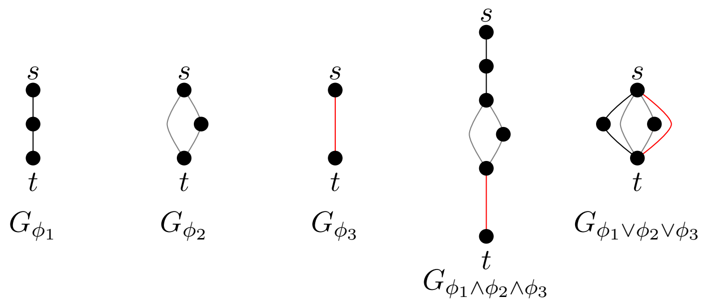

Figure 4: Let $\phi_1 = x_1 \wedge x_2$, $\phi_2 = x_3 \vee (x_4 \wedge x_5)$, and $\phi_3 = x_6$. Then we obtain $G_{\phi_1 \wedge \phi_2 \wedge \phi_3}$ by connecting G_{ϕ_1}, G_{ϕ_2}, and G_{ϕ_3} in series, and $G_{\phi_1 \vee \phi_2 \vee \phi_3}$ by connecting them in parallel.

Note that for any ϕ, G_ϕ is planar, and furthermore, both s and t are always on the same face. Thus, we can define G'_ϕ, $G_\phi(x)$ and $G'_\phi(x)$ as in Section 3. Then we can show the following:

Lemma 14. *Let ϕ be any* AND-OR *formula on N variables. For every $x \in \{0,1\}^N$, there exists a path from s to t in $G_\phi(x)$ if and only if $\phi(x) = 1$. Furthermore, for every $x \in \{0,1\}^N$, there exists a path from s' to t' in $G'_\phi(x)$ if and only if $\phi(x) = 0$.*

We give a formal proof of Lemma 14 in Appendix B, but the intuition is that an OR of subformulas, $\phi_1 \vee \dots \vee \phi_l$ evaluates to true if any of the subformulas evaluates to true, and likewise, if two vertices are connected by multiple subgraphs in parallel, the vertices are connected if there is a path in any of the subgraphs. An AND of subformulas $\phi_1 \wedge \dots \wedge \phi_l$ evaluates to true only if every subformula evaluates to true, and likewise, if two vertices are connected by multiple subgraphs in series, the vertices are only connected if there is a path through every subgraph. Thus, we can show by induction that s and t are connected in $G_\phi(x)$ if and only if $\phi(x) = 1$. To see that s' and t' are connected in $G'_\phi(x)$ if and only if $\phi(x) = 0$, we can use a similar argument, and make use of the fact that an $s't'$-path in $G'_\phi(x)$ is an st-cut in $G_\phi(x)$.

Lemma 14 implies that we can solve a formula evaluation problem $\textsc{Eval}_\phi$ by solving the associated st-connectivity problem, in which the input is a subgraph of G_ϕ. By our construction, G_ϕ will always be a planar graph with s and t on the external face, so moreover, we can apply Theorem 12 to obtain the following.

Theorem 15. *For any family ϕ of* AND-OR *formulas, the bounded error quantum query complexity of $\textsc{Eval}_\phi$ when the input is promised to come from a set D is*

$$O\left(\min_c \sqrt{\max_{x \in D : \phi(x)=1} R_{s,t}(G_\phi(x), c) \times \max_{x \in D : \phi(x)=0} R_{s',t'}(G'_\phi(x), c')}\right), \tag{21}$$

where the minimization is over all positive real-valued functions c on $E(G_\phi)$.

Proof. By Lemma 14, the query complexity of $\textsc{Eval}_\phi$ on D is at most the query complexity of $st\text{-}\textsc{conn}_{G_\phi, D}$. Since G_ϕ is planar, and has s and t on the same face, we can apply Theorem 12, which immediately implies the result. ☐

4.1 Comparison to Existing Boolean Formula Algorithms

Reichardt proved that the quantum query complexity of evaluating any formula on N variables is $O(\sqrt{N})$ [22, Corollary 1.6]. Our algorithm recovers this result:

Theorem 16. *Let ϕ be a read-once formula on N variables. Then there exists a choice of c on $E(G_\phi)$ such that the quantum algorithm obtained from the span program $P_{G_\phi,c}$ computes* $\textsc{Eval}_\phi$ *with bounded error in $O(\sqrt{N})$ queries.*

We need the following claim, which we prove in Appendix B:

Claim 17. *If $\phi = \phi_1 \vee \phi_2 \vee \cdots \vee \phi_l$, then $G'_\phi(x)$ is formed by composing $\{G'_{\phi_i}(x)\}_i$ in series, and if $\phi = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_l$, then $G'_\phi(x)$ is formed by composing $\{G'_{\phi_i}(x)\}_i$ in parallel.*

The intuition behind Claim 17 is the following. Although G'_ϕ is defined via the dual of G_ϕ, which is constructed through a sequence of series and parallel compositions, G'_ϕ itself can also be built up through a sequence of series and parallel compositions. For any and-or formula ϕ on N variables, we can define a formula ϕ' on N variables by replacing all $\vee$-nodes in ϕ with $\wedge$-nodes, and all $\wedge$-nodes in ϕ with $\vee$-nodes. By de Morgan's law, for all $x \in \{0,1\}^N$, $\phi(x) = \neg\phi'(\bar{x})$, where $\bar{x}$ is the entrywise negation of x. A simple inductive proof shows that $G_{\phi'} = G'_\phi$, and for all x, $G_{\phi'}(\bar{x}) = G'_\phi(x)$ (see Lemma 35 in Appendix B).

Proof of Theorem 16. We will make use of the following fact: for any network (G, c), and any positive real number W:

$$R_{s,t}(G, c/W) = \min_\theta \sum_{e\in E(G)} \frac{\theta(e)^2}{c(e)/W} = W \min_\theta \sum_{e\in E(G)} \frac{\theta(e)^2}{c(e)} = W R_{s,t}(G, c). \tag{22}$$

We now proceed with the proof. For any formula ϕ in $\{\wedge, \vee, \neg\}$, by repeated applications of de Morgan's law, we can push all NOT-gates to distance-1 from a leaf. Since x_i and $\neg x_i$ can both be learned in one query, we can restrict our attention to AND-OR formulas.

If ϕ has only $N = 1$ variable, it's easy to see that $W_+(P_{G_\phi,c})W_-(P_{G_\phi,c}) \leq N$ for c taking value 1 on the single edge in G_ϕ. We will prove by induction that this is true for any ϕ, for some choice of c, completing the proof, since the complexity of our algorithm obtained from $P_{G_\phi,c}$ is $O\left(\sqrt{W_+(P_{G_\phi,c})W_-(P_{G_\phi,c})}\right)$.

Suppose $\phi = \phi_1 \wedge \cdots \wedge \phi_l$ for formulas ϕ_i on N_i variables, so ϕ has $N = \sum_i N_i$ variables. For $x \in \{0,1\}^N$, we will let $x^i \in \{0,1\}^{N_i}$ denote the $(N_1 + \cdots + N_{i-1} + 1)$-th to $(N_1 + \cdots + N_i)$-th bits of x. For each G_{ϕ_i}, by the induction hypothesis, there is some weight function c_i on $E(G_{\phi_i})$ such that $W_+(P_{G_{\phi_i},c_i})W_-(P_{G_{\phi_i},c_i}) \leq N_i$.

Using our construction, G_ϕ is formed by composing $\{G_{\phi_i}\}_i$ in series. Thus every edge $(\{u,v\},\lambda) \in E(G_\phi)$ corresponds to an edge $(\{u,v\},\lambda) \in E(G_{\phi_i})$ for some i. We create a weight function $c : E(G_\phi) \to \mathbb{R}^+$ such that $c(\{u,v\},\lambda) = \frac{c_i(\{u,v\},\lambda)}{W_-(P_{G_{\phi_i},c_i})}$ if $(\{u,v\},\lambda)$ is an edge originating from the graph G_{ϕ_i}. That is, our new weight function is the same as combining all of the old weight functions, up to scaling factors $\{W_-(P_{G_{\phi_i},c_i})\}_i$.

Using Lemma 11, Claim 5, and Eq. (22), for any 1-instance x,

$$\begin{aligned} w_+(x, P_{G_phi,c}) &= \frac{1}{2}R_{s,t}(G_\phi(x), c) = \frac{1}{2}\sum_{i=1}^{l} R_{s,t}\left(G_{\phi_i}(x), \frac{c_i}{W_-(P_{G_{\phi_i},c_i})}\right) \\ &= \frac{1}{2}\sum_{i=1}^{l} W_-(P_{G_{\phi_i},c_i})R_{s,t}\left(G_{\phi_i}(x), c_i\right) \leq \sum_{i=1}^{l} W_-(\phi_i, P_{G_{\phi_i},c_i})W_+(P_{G_{\phi_i},c_i}). \end{aligned} \tag{23}$$

Thus

$$W_+(P_{G_\phi,c}) \leq \sum_{i=1}^{l} W_-(P_{G_{\phi_i},c_i})W_+(P_{G_{\phi_i},c_i}) \leq \sum_{i=1}^{l} N_i = N. \tag{24}$$

Recall that for a weight function c on G_ϕ, we define a weight function c' on G'_ϕ by $c'(e^\dagger) = 1/c(e)$. Then for an edge $e \in E(G_{\phi_i}(x^i))$, we have $c'(e^\dagger) = W_-(P_{G_{\phi_i},c_i})/c_i(e) = W_-(P_{G_{\phi_i},c_i})c'_i(e^\dagger)$. By Claim 17, G'_ϕ is formed by composing $\{G'_{\phi_i}\}_i$ in parallel, so by Lemma 11, Claim 5, and Eq. (22):

$$\begin{aligned} w_-(x, P_{G_\phi,c}) &= 2R_{s',t'}(G'_\phi(x), c') = 2\left(\sum_{i=1}^{l} \frac{1}{R_{s',t'}(G'_{\phi_i}(x^i), c')}\right)^{-1} \\ &= 2\left(\sum_{i=1}^{l} \frac{1}{R_{s',t'}(G'_{\phi_i}(x^i), W_-(P_{G_{\phi_i},c_i})c'_i)}\right)^{-1} = 2\left(\sum_{i=1}^{l} \frac{W_-(P_{G_{\phi_i},c_i})}{R_{s',t'}(G'_{\phi_i}(x^i), c'_i)}\right)^{-1} \\ &= 2\left(\sum_{i=1}^{l} \frac{W_-(P_{G_{\phi_i},c_i})}{w_-(x^i, P_{G_{\phi_i},c_i})}\right)^{-1}. \end{aligned} \tag{25}$$

Whenever x is a 0-instance of ϕ, the set $S \subseteq [l]$ of i such that x^i is a 0-instance of ϕ_i is non-empty. This is exactly the set of i such that $w_-(x^i, P_{G_{\phi_i},c_i}) < \infty$. Continuing from above, we have:

$$w_-(x, P_{G_\phi,c}) = \left(\sum_{i\in S} \frac{W_-(P_{G_{\phi_i},c_i})}{w_-(x^i, P_{G_{\phi_i},c_i})}\right)^{-1} \leq \left(\sum_{i\in S} \frac{W_-(P_{G_{\phi_i},c_i})}{W_-(P_{G_{\phi_i},c_i})}\right)^{-1} = \frac{1}{|S|} \leq 1. \tag{26}$$

Thus $W_-(P_{G_{\phi_i},c_i}) \leq 1$. Combining this with Eq. (24) we have $W_+(P_{G_\phi,c})W_-(P_{G_\phi,c}) \leq N$, as desired.
The proof for the case $\phi = \phi_1 \vee \cdots \vee \phi_l$ is similar. □

An immediate corollary of Theorem 16 is the following.

Corollary 18. *Deciding st-connectivity on subgraphs of two-terminal series-parallel graphs of N edges can be accomplished using $O(\sqrt{N})$ queries, if s and t are chosen to be the two terminal nodes.*

As with many results in this field, characterizing classical complexity seems to be more difficult than quantum complexity. However, we show we can lower bound the classical query complexity of a class of Boolean formulas in terms of the effective resistance of their corresponding graphs, achieving a quadratic quantum/classical speed-up in query complexity.

We consider AND-OR formulas on restricted domains. For $N, h \in \mathbb{Z}^+$, let $D_{N,h} = \{x \in \{0,1\}^N : |x| = N \text{ or } |x| \leq N - h\}$ and let $D'_{N,h} = \{x \in \{0,1\}^N : |x| = 0 \text{ or } |x| \geq h\}$. We will analyze AND-OR formulas such that the input to every gate in the formula comes from $D_{N,h}$ (in the case of AND), which we denote $\text{AND}|_{D_{N,h}}$ and $D'_{N,h}$ (in the case of OR), which we denote $\text{OR}|_{D'_{N,h}}$. These promises on the domains make it easier to evaluate both functions. For example, if OR evaluates to 1, we are promised that there will not be just one input with value 1, but at least h.

Then using sabotage complexity [5] to bound the classical query complexity, we have the following theorem, whose (somewhat long, but not technical) proof can be found in Appendix C:

Theorem 19. *Let $\phi = \phi_1 \circ \phi_2 \circ \cdots \circ \phi_l$, where for each $i \in [l]$, $\phi_i = \text{OR}|_{D'_{N_i,h_i}}$ or $\phi_i = \text{AND}|_{D_{N_i,h_i}}$. Then the randomized bounded-error query complexity of evaluating ϕ is $\Omega\left(\prod_{i=1}^{l} N_i/h_i\right)$, and the bounded-error quantum query complexity of evaluating ϕ is $O\left(\prod_{i=1}^{l} \sqrt{N_i/h_i}\right)$.*

Note that in the above theorem, when we compose formulas with promises on the input, we implicitly assume a promise on the input to the composed formula. More precisely, for ϕ_1 on $D_1 \subseteq \{0,1\}^{N_1}$ and ϕ_2 on $D_2 \subseteq \{0,1\}^{N_2}$, $\phi = \phi_1 \circ \phi_2$ is defined on all $x = (x^1, \ldots, x^{N_1}) \in \{0,1\}^{N_1 N_2}$ such that $x^i \in D_2$ for all $i \in [N_1]$, and $(\phi_2(x^1), \ldots, \phi_2(x^{N_1})) \in D_1$.

Theorem 19 is proven by showing that

$$\frac{\prod_{i=1}^{l} N_i}{\prod_{i=1}^{l} h_i} = \left(\max_{x \in D:\phi(x)=1} R_{s,t}(G_\phi(x))\right)\left(\max_{x \in D:\phi(x)=0} R_{s,t}(G'_\phi(x))\right), \tag{27}$$

and using sabotage complexity to show that this is a lower bound on the randomized query complexity of ϕ. This gives us a quadratic separation between the randomized and quantum query complexities of this class of formulas. For details, see Appendix C.

Using the composition lower bound for promise Boolean functions of [19], and the lower bound for Grover's search with multiple marked items [6], we have that the quantum query complexity of Theorem 19 is tight. Additionally, in light of our reduction from Boolean formula evaluation to st-connectivity, we see that our example from Figure 2 in Section 3 is equivalent to the problem of $\text{AND}|_{D_{N,h}}$, so our query bound in that example is also tight.

Based on Theorem 19, one might guess that when evaluating formulas using the st-connectivity reduction, one can obtain at most a quadratic speed-up over classical randomized query complexity. However, it is in fact possible to obtain a superpolynomial speed-up for certain promise problems using this approach, as we will discuss in Section 5.1.

5 NAND-tree Results

5.1 Query Separations

In this section, we prove two query separations that are stronger than our previous results. These query separations rely on the NAND-tree formula with a promise on the inputs. This restriction, the k-fault promise, will be defined shortly. Let F_k^d be the set of inputs to NAND_d that satisfy the k-fault condition. Then the two results are the following:

Theorem 20. *Using the st-connectivity approach to formula evaluation (Theorem 15), one can solve $\text{EVAL}_{\text{NAND}_d}$ when the input is promised to be from $F_{\log d}^d$ with $O(d)$ queries, while any classical algorithm requires $\Omega(d^{\log\log(d)})$ queries.*

For a different choice of k, this example demonstrates the dramatic improvement our st-connectivity algorithm can give over the analysis of [4] — in this case, an exponential (or more precisely, a polynomial to constant) improvement:

Theorem 21. *Consider the problem $st\text{-CONN}_{G_{\text{NAND}_d}, F_1^d}$. The analysis of [4] gives a bound of $O(N^{1/4})$ quantum queries to decide this problem (where $N = 2^d$ is the number of edges in G_{NAND_d}), while our analysis shows this problem can be decided with $O(1)$ quantum queries.*

We now define what we mean by k-fault NAND-trees. In [27], Zhan et al. find a relationship between the difficulty of playing the two-player game associated with a NAND-tree, and the witness size of a particular span program for $\textsc{nand}_d$. They find that trees with fewer *faults*, or critical decisions for a player playing the associated two-player game, are easier to evaluate on a quantum computer. We show that our algorithm does at least as well as the algorithm of Zhan et al. for evaluating k-fault trees. To see this, we relate fault complexity to effective resistances of $G_{\textsc{nand}_d}(x)$ or $G'_{\textsc{nand}_d}(x)$.

Consider a $\textsc{nand}_d$ instance $x \in \{0,1\}^{2^d}$, and recall the relationship between a NAND-tree instance and the two-player NAND-tree game described in Section 2.3. We call the sequence of nodes that Player A and Player B choose during the course of a game a path p — this is just a path from the root of the NAND-tree to a leaf node. If x is Z-winnable, we call $\mathcal{P}_Z(x)$ the set of paths where Player Z wins, and Player Z never makes a move that would allow her opponent to win. That is, a path in $\mathcal{P}_A(x)$ (resp. $\mathcal{P}_B(x)$) only encounters nodes that are themselves the roots of A-winnable (resp. B-winnable) subtrees and never passes through a node where Player B (resp. Player A) could make a decision to move to a B-winnable (resp. A-winnable) subtree. Whether a node in the tree is the root of an A-winnable or B-winnable subtree can be determined by evaluating the subformula corresponding to that subtree. See Figure 5 for an example of $\mathcal{P}_A$. Let $\nu_Z(p)$ be the set of nodes along a path p at which it is Player Z's turn. Thus, $\nu_A(p)$ (resp. $\nu_B(p)$) contains those nodes in p at even (resp. odd) distance > 0 from the leaves.

Zhan et al. call a node v a *fault* if one child is the root of an A-winnable tree, while the other child is the root of a B-winnable tree. Such a node constitutes a critical decision point. If we let $f_Z(p)$ denote the number of faults in $\nu_Z(p)$, we can define the fault complexity $\mathcal{F}(x)$ of input x as[1] $\mathcal{F}(x) = \min\{\mathcal{F}_A(x), \mathcal{F}_B(x)\}$, where:

$$\mathcal{F}_Z(x) = \begin{cases} 2^{\max_{p\in\mathcal{P}_Z(x)} f_Z(p)} & \text{if } x \text{ is } Z\text{-winnable} \\ \infty & \text{otherwise.} \end{cases} \tag{28}$$

For $k = 0, \ldots, d/2$, the set of k-fault trees, F^d_k, are those instances $x \in \{0,1\}^{2^d}$ with $\log \mathcal{F}(x) \leq k$. In these trees, the winning player will encounter at most k fault nodes on their path to a leaf. Kimmel [19] shows there exists a span program for evaluating NAND-trees whose witness size for an instance x is at most the fault complexity $\mathcal{F}(x)$.

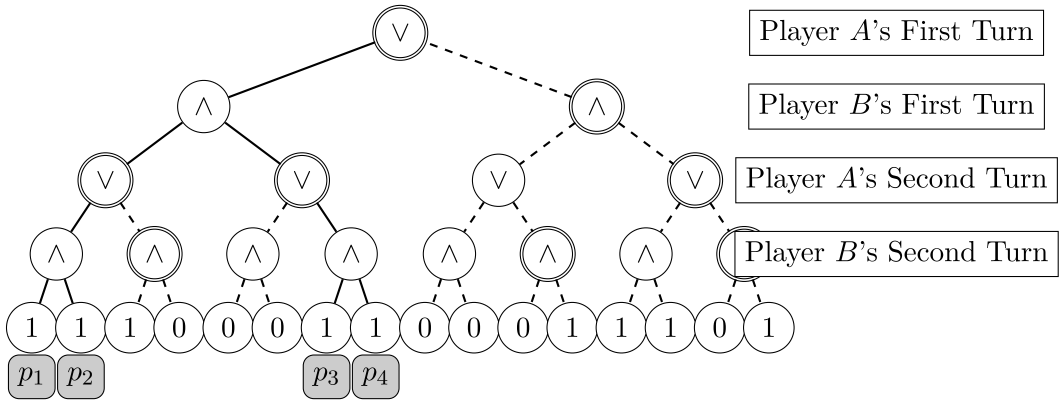

Figure 5: Depiction of a depth-4 NAND-tree as a two-player game. Let x be the input to $\textsc{nand}_4$ shown in the figure. This instance is A-winnable, and $\mathcal{P}_A(x)$ consists of the paths $\{p_1, p_2, p_3, p_4\}$, shown using solid lines. Fault nodes are those with double circles. Each path in $\mathcal{P}_A(x)$ encounters two faults at nodes where Player A makes decisions. Therefore, $\mathcal{F}_A(x) = 4$.

We first show a relationship between effective resistance of $G_{\textsc{nand}_d}(x)$ or $G'_{\textsc{nand}_d}(x)$ and $\mathcal{F}(x)$:

[1] We have actually used the more refined definition of k-fault from [19].

Lemma 22. *For any $x \in \{0,1\}^{2^d}$, if d is even, then we have $R_{s,t}(G_{\text{NAND}_d}(x)) \leq \mathcal{F}_A(x)$ and $R_{s',t'}(G'_{\text{NAND}_d}(x)) \leq \mathcal{F}_B(x)$, while if d is odd, we have $R_{s,t}(G_{\text{NAND}_d}(x)) \leq 2\mathcal{F}_A(x)$ and $R_{s',t'}(G'_{\text{NAND}_d}(x)) \leq 2\mathcal{F}_B(x)$.*

The proof of Lemma 22 can be found in Appendix D. An immediate corollary of Lemma 22 and Theorem 15 is the following.

Corollary 23. *The span program P_{G_ϕ} for $\phi = \text{NAND}_d$ decides $\text{EVAL}_{\text{NAND}_d}$ restricted to the domain X in $O(\max_{x\in X} \mathcal{F}(x))$ queries. In particular, it decides k-fault trees (on domain F_k^d) in $O(2^k)$ queries.*

Proof of Theorem 20. Theorem 20 is now an immediate consequence of Corollary 23, with k set to $\log(d)$, along with the fact from [27] that the classical query complexity of evaluating such formulas is $\Omega(d^{\log\log(d)})$. □

We will use Corollary 23, along with the following claim, to prove Theorem 21:

Claim 24. *Let two graphs G_1 and G_2 each have nodes s and t and let $x^1 \in \{0,1\}^{E(G_1)}$ and $x^2 \in \{0,1\}^{E(G_2)}$. Suppose we create a new graph G by identifying the s nodes and the t nodes (i.e. connecting the graphs in parallel), then*

$$C_{s,t}(G(x^1,x^2)) = C_{s,t}(G_1(x^1)) + C_{s,t}(G_2(x^2)) \tag{29}$$

If we create a new graph G by identifying the t node of G_1 with the s node of G_2 and relabeling this node $v \notin \{s,t\}$ (i.e. connecting the graphs in series), then

$$C_{s,t}(G(x^1,x^2)) = \min\{C_{s,t}(G_1(x^1)), C_{s,t}(G_2(x^2))\}. \tag{30}$$

Proof of Theorem 21. Using Corollary 23, our analysis shows that $st\text{-CONN}_{G_{\text{NAND}_d},F_1^d}$ can be decided in $O(1)$ queries.

We apply Eq. (8) to compare to the analysis of [4]. We must characterize the quantity $\max\{C_{s,t}(G_{\text{NAND}_d}(x)) : x \in F_1^d, \text{and } s,t \text{ are not connected}\}$, since we already have

$$\max_{x\in F_1^d: s,t \text{ are connected}} R_{s,t}(G(x)) = O(1). \tag{31}$$

We now prove that for *every* $x \in \{0,1\}^{2^d}$ such that $\text{NAND}_d(x) = 0$, $C_{s,t}(G_{\text{NAND}_d}(x)) = 2^{\lfloor d/2\rfloor}$. Thus, for any promise D on the input, as long as there exists some $x \in D$ such that $\text{NAND}_d(x) = 0$, we have $\max_{x\in D:\text{NAND}_d(x)=0} C_{s,t}(G_{\text{NAND}_d}(x)) = 2^{\lfloor d/2\rfloor}$. Intuitively, this is because every st-cut on any subgraph of G_{NAND_d} cuts across $2^{\lfloor d/2\rfloor}$ edges of G_{NAND_d}.

The proof is by induction on d. For the base even case, $d = 0$ and G_{NAND_0} is a single edge connecting s and t. The only input $x \in \{0,1\}^{2^0}$ such that $\text{NAND}_d(x) = 0$ is $x = 0$, in which case, the st-cut is $\kappa(s) = 1$ and $\kappa(t) = 0$, so the cut is across the unique edge in G_{NAND_0}, so $C_{s,t}(G_{\text{NAND}_0}(x)) = 1$.

For the induction step, we treat even and odd separately. Suppose $d > 0$ is odd. Then $\text{NAND}_d(x) = \text{NAND}_{d-1}(x^0) \wedge \text{NAND}_{d-1}(x^1)$, where $x^0 = (x_1,\dots,x_{2^{d-1}})$ and $x^1 = (x_{2^{d-1}+1},\dots,x_{2^d})$. Thus $G_{\text{NAND}_d}(x)$ involves composing two graphs $G_{\text{NAND}_{d-1}}(x^0)$ and $G_{\text{NAND}_{d-1}}(x^1)$ in series. Since we are assuming s and t are not connected in $G_{\text{NAND}_d}(x)$, at least one of $G_{\text{NAND}_{d-1}}(x^0)$ and $G_{\text{NAND}_{d-1}}(x^1)$ must not be connected. Without loss of generality, suppose $G_{\text{NAND}_{d-1}}(x^0)$ is not connected and $C_{s,t}(G_{\text{NAND}_{d-1}}(x^0)) \leq C_{s,t}(G_{\text{NAND}_{d-1}}(x^1))$. By induction, $C_{s,t}(G_{\text{NAND}_{d-1}}(x^0)) = 2^{(d-1)/2}$. Thus using Eq. (30) in Claim 24,

$$C_{s,t}(G_{\text{NAND}_d}(x)) = 2^{(d-1)/2} = 2^{\lfloor d/2\rfloor}. \tag{32}$$

Now suppose $d > 0$ is even. Then $\textsc{nand}_d(x) = \textsc{nand}_{d-1}(x^0) \vee \textsc{nand}_{d-1}(x^1)$. Thus $G_{\textsc{nand}_d}(x)$ involves composing two graphs $G_{\textsc{nand}_{d-1}}(x^0)$ and $G_{\textsc{nand}_{d-1}}(x^1)$ in parallel. Since we are assuming s and t are not connected in $G_{\textsc{nand}_d}(x)$, both of $G_{\textsc{nand}_{d-1}}(x^0)$ and $G_{\textsc{nand}_{d-1}}(x^1)$ must not be connected, and so by induction, we have $C_{s,t}(G_{\textsc{nand}_{d-1}}(x^0)) = C_{s,t}(G_{\textsc{nand}_{d-1}}(x^1)) = 2^{\lfloor (d-1)/2 \rfloor} = 2^{d/2-1}$, since d is even. Thus using Eq. (29) in Claim 24,

$$C_{s,t}(G_{\textsc{nand}_d}(x)) = 2^{d/2-1} + 2^{d/2-1} = 2^{d/2} = 2^{\lfloor d/2 \rfloor}. \tag{33}$$

Therefore, using Eq. (8), we have that the analysis of [4] for d-depth $\textsc{nand}$-trees with inputs in F_1^d gives a query complexity of $O(\sqrt{2^{\lfloor d/2 \rfloor}}) = O(N^{1/4})$, where $N = 2^d$ is the number of input variables. Comparing with our analysis, which gives a query complexity of $O(1)$, we see there is a polynomial to constant improvement. □

5.2 Winning the NAND-tree Game

In this section, we describe a quantum algorithm that can be used to help a player make decisions while playing the $\textsc{nand}$-tree game. In particular, we consider the number of queries to x needed by Player A to make decisions throughout the course of the game in order to win with probability $\geq 2/3$. (In this section, we focus on A-winnable trees, but the case of B-winnable trees is similar.)

We first describe a naive strategy, which uses a quantum algorithm [21, 23] that decides if a depth-d tree is winnable with bounded error in $O(2^{d/2} \log d)$ queries. If Player A must decide to move to node v_0 or v_1, she evaluates each subtree rooted at v_0 and v_1, amplifying the success probability to $\Omega(1/d)$ by using $O(\log d)$ repetitions, and moves to one that evaluates to 1. Since Player A has $O(d)$ decisions to make, this strategy succeeds with bounded error, and since evaluating a $\textsc{nand}$-tree of depth r costs $O(2^{r/2})$ quantum queries, the total query complexity is:

$$O\left(2\sum_{i=0}^{\frac{d}{2}} 2^{\frac{d-2i}{2}} \log d\right) = O\left(2\sum_{i=0}^{\frac{d}{2}} 2^i \log d\right) = O\left(2^{\frac{d}{2}} \log d\right) = O\left(\sqrt{N} \log\log N\right). \tag{34}$$

This strategy does not use the fact that some subtrees may be easier to win than others. For example, if one choice leads to a subtree with all leaves labeled by 1, whereas the other subtree has all leaves labeled by 0, the player just needs to distinguish these two disparate cases. More generally, one of the subtrees might have a small positive witness size — i.e., it is very winnable — whereas the other has a large positive witness size — i.e., is not very winnable.

Our strategy will be to move to the subtree whose formula corresponds to a graph with smaller effective resistance, unless the two subtrees are very close in effective resistance, in which case it doesn't matter which one we choose. For a depth d game on instance x, we show if $R_{s,t}(G_{\textsc{nand}_d}(x))$ is small and Player B plays randomly, this strategy does better than the naive strategy, on average.

We estimate the effective resistance of both subtrees of the current node using the witness size estimation algorithm of [17]. In particular, in Appendix D.2 we prove:

Lemma 25 (`Est` Algorithm)**.** *Let ϕ be an $\textsc{and}$-$\textsc{or}$ formula with constant fan-in l, $\vee$-depth $d_\vee$ and $\wedge$-depth $d_\wedge$. Then the quantum query complexity of estimating $R_{s,t}(G_\phi(x))$ (resp. $R_{s,t}(G'_\phi(x))$) to relative accuracy ϵ is $\widetilde{O}\left(\frac{1}{\varepsilon^{3/2}}\sqrt{R_{s,t}(G_\phi(x))l^{d_\vee}}\right)$ (resp. $\widetilde{O}\left(\frac{1}{\varepsilon^{3/2}}\sqrt{R_{s,t}(G'_\phi(x))l^{d_\wedge}}\right)$).*

Let $\texttt{Est}(x)$ be the algorithm from Lemma 25 with $\varepsilon = \frac{1}{3}$, and $\phi = \text{NAND}_d$, so $l = 2$, and both $d_\vee$ and $d_\wedge$ are at most $\lceil d/2 \rceil$. While estimating the effective resistance of two subtrees, we only care about which of the subtrees has the smaller effective resistance, so we do not want to wait for both iterations of $\texttt{Est}$ to terminate. Let $p(d)$ be some polynomial function in d such that $\texttt{Est}(x)$ always terminates after at most $p(d)\sqrt{R_{s,t}(G_\phi(x))2^{d/4}}$ queries, for all $x \in \{0,1\}^{2^d}$. We define a subroutine, $\texttt{Select}(x^0, x^1)$, that takes two instances, $x^0, x^1 \in \{0,1\}^{2^{d-1}}$, and outputs a bit b such that $R_{s,t}(G_{\text{NAND}_{d-1}}(x^b)) \leq 2R_{s,t}(G_{\text{NAND}_{d-1}}(x^{\bar{b}}))$, where $\bar{b} = b \oplus 1$. $\texttt{Select}$ works as follows. It runs $\texttt{Est}(x^0)$ and $\texttt{Est}(x^1)$ in parallel. If one of these programs, say $\texttt{Est}(x^b)$, outputs some estimate w_b, then it terminates the other program after $p(d)\sqrt{w_b}2^{d/4}$ 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 $w_b \leq w_{\bar{b}}$. In Appendix D.3, we prove the following lemma.

Lemma 26. *Let $x^0, x^1 \in \{0,1\}^{2^d}$ be instances of NAND_d with at least one of them a 1-instance. Let $N = 2^d$, and $w_{\min} = \min\{R_{s,t}(G_{\text{NAND}_d}(x^0)), R_{s,t}(G_{\text{NAND}_d}(x^1))\}$. Then $\texttt{Select}(x^0, x^1)$ terminates after $\widetilde{O}\left(N^{1/4}\sqrt{w_{\min}}\right)$ queries to (x^0, x^1) and outputs b such that $R_{s,t}(G_{\text{NAND}_d}(x^b)) \leq 2R_{s,t}(G_{\text{NAND}_d}(x^{\bar{b}}))$ with bounded error.*

Using Lemma 26, we can prove the following (the inductive proof is in Appendix D.3):

Theorem 27. *Let $x \in \{0,1\}^N$ for $N = 2^d$ be an A-winnable input to NAND_d. 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 $\widetilde{O}\left(N^{1/4}\sqrt{R_{s,t}(G_{\text{NAND}_d}(x))}\right)$ queries on average, where the average is taken over the randomness of Player B's choices.*

6 Acknowledgments

SK is funded by the US Department of Defense. SJ acknowledges funding provided by the Institute for Quantum Information and Matter, an NSF Physics Frontiers Center (NFS Grant PHY-1125565) with support of the Gordon and Betty Moore Foundation (GBMF-12500028); and an NWO WISE Fellowship.

References

[1] S. Aaronson and S. Ben-David. Sculpting quantum speedups. In *Proceedings of the 31st Conference on Computational Complexity*, CCC '16, pages 26:1–26:28, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-008-8. DOI: 10.4230/LIPIcs.CCC.2016.26. URL http://dl.acm.org/citation.cfm?id=2982445.2982471.

[2] D. Aldous and J. Fill. Reversible Markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014, available at http://www.stat.berkeley.edu/~aldous/RWG/book.html.

[3] 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. DOI: 10.1145/2213977.2213985.

[4] 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. DOI: 10.1007/978-3-642-33090-2_18.

[5] S. Ben-David and R. Kothari. Randomized query complexity of sabotaged and composed functions. In *Proceedings of the 43th International Colloquium on Automata, Languages and Programming (ICALP 2016)*, volume 55, pages 60:1–60:14, 2016. DOI: 10.4230/LIPIcs.ICALP.2016.60.

[6] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, 1998. ISSN 1521-3978. DOI: 10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P. URL http://dx.doi.org/10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P.

[7] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[8] C. Cade, A. Montanaro, and A. Belovs. Time and space efficient quantum algorithms for detecting cycles and testing bipartiteness, 2016. `arXiv:1610.00581`.

[9] A. K. Chandra, P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari. The electrical resistance of a graph captures its commute and cover times. *Computational Complexity*, 6(4):312–340, 1996. DOI: 10.1007/BF01270385.

[10] G. A. Dirac. A property of 4-chromatic graphs and some remarks on critical graphs. *Journal of the London Mathematical Society*, 1(1):85–92, 1952. DOI: 10.1112/jlms/s1-27.1.85.

[11] 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.

[12] R. J. Duffin. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, 10(2):303–318, 1965. ISSN 0022-247X. DOI: http://dx.doi.org/10.1016/0022-247X(65)90125-3. URL http://www.sciencedirect.com/science/article/pii/0022247X65901253.

[13] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006. DOI: 10.1137/050644719.

[14] E. Farhi, J. Goldstone, and S. Gutmann. A quantum algorithm for the Hamiltonian NAND tree. *Theory of Computing*, 4(8):169–190, 2008. DOI: 10.4086/toc.2008.v004a008. `arXiv:quant-ph/0702144`.

[15] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th annual ACM Symposium on Theory of computing (STOC 1996)*, pages 212–219. ACM, 1996. DOI: 10.1145/237814.237866.

[16] R. Heiman and A. Wigderson. Randomized vs. deterministic decision tree complexity for read-once Boolean functions. *Computational Complexity*, 1(4):311–329, 1991. DOI: 10.1007/BF01212962.

[17] T. Ito and S. Jeffery. Approximate span programs. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, pages 12:1–12:14, 2016. DOI: 10.4230/LIPIcs.ICALP.2016.12. `arXiv:1507.00432`.

[18] 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. DOI: 10.1109/SCT.1993.336536.

[19] S. Kimmel. Quantum adversary (upper) bound. *Chicago Journal of Theoretical Computer Science*, 2013(4), 2011. DOI: 10.4086/cjtcs.2013.004.

[20] T. Lee, R. Mittal, B. W. Reichardt, R. Špalek, and M. Szegedy. Quantum query complexity of state conversion. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011)*, pages 344–353, 2011. DOI: 10.1109/FOCS.2011.75.

[21] 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. DOI: 10.1109/FOCS.2009.55. `arXiv:quant-ph/0904.2759`.

[22] B. W. Reichardt. Span programs and quantum query algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:110, 2010.

[23] 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.

[24] B. W. Reichardt and R. Špalek. Span-program-based quantum algorithm for evaluating formulas. *Theory of Computing*, 8(13):291–319, 2012. DOI: 10.4086/toc.2012.v008a013.

[25] 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. DOI: 10.1109/SFCS.1986.44.

[26] J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series parallel digraphs. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC 1979)*, pages 1–12. ACM, 1979. DOI: 10.1145/800135.804393. URL http://doi.acm.org/10.1145/800135.804393.

[27] 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. ISBN 978-1-4503-1115-1. DOI: 10.1145/2090236.2090258. URL http://doi.acm.org/10.1145/2090236.2090258.

A Analysis of the Span Program for st-Connectivity

In this section, we analyze the complexity of our span-program-based algorithms, proving Lemma 11, first stated in Section 3, which relates witness sizes of the span program $P_{G,c}$ to the effective resistance of graphs related to G.

We need the concept of a circulation, which is like a flow but with no source and no sink.

Definition 28 (Circulation)**.** *A* circulation *on a graph G is a function $\theta : \overrightarrow{E}(G) \to \mathbb{R}$ such that:*

1. *For all $(u, v, \lambda) \in \overrightarrow{E}(G)$, $\theta(u, v, \lambda) = -\theta(v, u, \lambda)$;*
2. *for all $u \in V(G)$, $\sum_{v,\lambda:(u,v\lambda)\in\overrightarrow{E}(G)} \theta(u, v, \lambda) = 0$.*

The following easily verified observations will be useful in several of the remaining proofs in this section.

Claim 29. *Let θ be a unit st-flow in some multigraph G. We can consider the corresponding vector $|\theta\rangle = \sum_{(u,v,\lambda)\in\overrightarrow{E}(G)} \theta(u, v, \lambda)|u, v, \lambda\rangle$. Then $|\theta\rangle$ can be written as a linear combination* *of vectors corresponding to self-avoiding st-paths and cycles that are edge-disjoint from these paths.*

Let σ be a circulation on G. Then $|\sigma\rangle$ can be written as a linear combination of cycles in G. Furthermore, $|\sigma\rangle$ can be written as a linear combination of cycles such that each cycle goes around a face of G.

The next claim shows a direct correspondence between positive witnesses, and st-flows.

Claim 30. *Fix a span program $P_{G,c}$ as in (6). Call $|w\rangle \in H$ a positive witness in $P_{G,c}$ if $A|w\rangle = \tau$ (note that such a $|w\rangle$ is not necessarily a positive witness for any particular input x). Then if θ is a unit st-flow in G, $\frac{1}{2}\sum_{(u,v,\lambda)\in\overrightarrow{E}(G)}\frac{\theta(u,v,\lambda)}{\sqrt{c(\{u,v\},\lambda)}}|u,v,\lambda\rangle$ is a positive witness in $P_{G,c}$, and furthermore, if $|w\rangle$ is a positive witness in $P_{G,c}$, then $\theta(u,v,\lambda) = \sqrt{c(\{u,v\},\lambda)}(\langle w|u,v,\lambda\rangle - \langle w|v,u,\lambda\rangle)$ is a unit st-flow in G.*

Proof. The proof is a straightforward calculation. Let θ be a unit st-flow on G. Then

$$\begin{aligned}
&A\left(\frac{1}{2}\sum_{(u,v,\lambda)\in\overrightarrow{E}(G)}\frac{\theta(u,v,\lambda)}{\sqrt{c(\{u,v\},\lambda)}}|u,v,\lambda\rangle\right)\\
&=\frac{1}{2}\sum_{(u,v,\lambda)\in\overrightarrow{E}(G)}\theta(u,v,\lambda)(|u\rangle-|v\rangle)\\
&=\frac{1}{2}\sum_{u\in V(G)}\left(\sum_{v,\lambda:(u,v,\lambda)\in\overrightarrow{E}(G)}\theta(u,v,\lambda)\right)|u\rangle+\frac{1}{2}\sum_{v\in V(G)}\left(\sum_{u,\lambda:(v,u,\lambda)\in\overrightarrow{E}(G)}\theta(v,u,\lambda)\right)|v\rangle\\
&=\frac{1}{2}(|s\rangle-|t\rangle)+\frac{1}{2}(|s\rangle-|t\rangle)=\tau. \qquad (35)
\end{aligned}$$

Above we have used that $\theta(u,v,\lambda) = -\theta(v,u,\lambda)$, and $\sum_{v,\lambda:(u,v,\lambda)\in\overrightarrow{E}(G)}\theta(u,v,\lambda) = 0$ when $u \notin \{s,t\}$, 1 when $u = s$, and -1 when $u = t$.

To prove the second half of the claim, let $|w\rangle$ be such that $A|w\rangle = \tau$, and define $\theta(u,v,\lambda) = \sqrt{c(\{u,v\},\lambda)}(\langle w|u,v,\lambda\rangle - \langle w|v,u,\lambda\rangle)$. We immediately see that $\theta(u,v,\lambda) = -\theta(v,u,\lambda)$ for all (u,v,λ). Furthermore, we have:

$$\begin{aligned}
|s\rangle-|t\rangle = A|w\rangle &= \sum_{u\in V(G)}\left(\sum_{(u,v,\lambda)\in\overrightarrow{E}(G)}\sqrt{c(\{u,v\},\lambda)}\langle u,v,\lambda|w\rangle\right)|u\rangle\\
&\quad-\sum_{v\in V(G)}\left(\sum_{(u,v,\lambda)\in\overrightarrow{E}(G)}\sqrt{c(\{u,v\},\lambda)}\langle u,v,\lambda|w\rangle\right)|v\rangle\\
&=\sum_{u\in V(G)}\left(\sum_{(u,v,\lambda)\in\overrightarrow{E}(G)}\sqrt{c(\{u,v\},\lambda)}(\langle u,v,\lambda|w\rangle-\langle v,u,\lambda|w\rangle)\right)|u\rangle\\
&=\sum_{u\in V(G)}\left(\sum_{(u,v,\lambda)\in\overrightarrow{E}(G)}\theta(u,v,\lambda)\right)|u\rangle. \qquad (36)
\end{aligned}$$

Thus, for all $u \in V(G(x))\setminus\{s,t\}$, $\sum_{(u,v,\lambda)\in\overrightarrow{E}(G)}\theta(u,v,\lambda) = 0$, and $\sum_{(s,v,\lambda)\in\overrightarrow{E}(G)}\theta(s,v,\lambda) = \sum_{(v,t,\lambda)\in\overrightarrow{E}(G)}\theta(v,t,\lambda) = 1$. Thus, θ is a unit st-flow on G. □

The next claim shows a direct correspondence between negative witnesses, and $s't'$-flows.

Claim 31. *For a planar graph G, fix a span program $P_{G,c}$ as in (6). Call a linear function $\omega : V(G) \to \mathbb{R}$ a negative witness if $\omega\tau = 1$. Then $\theta((u,v,\lambda)^\dagger) = \omega(u) - \omega(v)$ is a unit $s't'$-flow on G', and furthermore, for every $s't'$-flow θ on G' there is a negative witness ω such that $\theta((u,v,\lambda)^\dagger) = \omega(u) - \omega(v)$ for all $(u,v,\lambda) \in \overrightarrow{E}(G)$.*

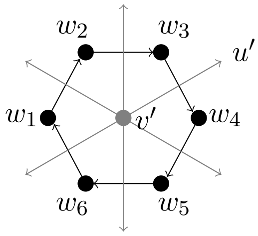

Figure 6: The duality between a cycle and a star.

Proof. When we consider the edges of G as directed edges, we assign edge directions to the dual by orienting each dual edge $\pi/2$ radians counter-clockwise from the primal edge.

Note that without loss of generality, if ω is a negative witness, we can assume $\omega(s) = 1$ and $\omega(t) = 0$. This is because $\|\omega A\|$ and $\left\|\omega A \Pi_{H(x)}\right\|$ are invariant under affine transformations of ω.

We first show that if ω is a negative witness in $P_{G,c}$, then $\theta : \overrightarrow{E}(G') \to \mathbb{R}$ defined $\theta((u,v,\lambda)^\dagger) = \omega(u) - \omega(v)$ is a unit $s't'$-flow on G'. To begin with, we will define $\theta'((u,v,\lambda)^\dagger) = \omega(u) - \omega(v)$ on $\overrightarrow{E}(\overline{G}^\dagger)$, so θ' agrees with θ everywhere θ is defined, and in addition, $\theta'(s',t',\emptyset) = \theta'((s,t,\emptyset)^\dagger) = \omega(s) - \omega(t) = 1$, and $\theta'(t',s',\emptyset) = -1$. Then clearly we have $\theta'(u',v',\lambda) = -\theta'(v',u',\lambda)$ for all $(\{u',v'\},\lambda) \in E(\overline{G}^\dagger)$.

Next, every $v' \in V(\overline{G}^\dagger)$ corresponds to a face $f_{v'}$ of $\overline{G}$, and the edges coming out of v' are dual to edges going clockwise around the face $f_{v'}$ (see Figure 6). If $(w_1, w_2, \lambda_1), \dots, (w_k, w_{k+1}, \lambda_k)$, for $w_{k+1} = w_1$, are the directed edges going clockwise around $f_{v'}$, then we have:

$$0 = \sum_{i=1}^{k} (\omega(w_i) - \omega(w_{i+1})) = \sum_{i=1}^{k} \theta'((w_i, w_{i+1}, \lambda_i)^\dagger) = \sum_{\substack{u',\lambda: \\ (\{v',u'\},\lambda) \in E(\overline{G}')}} \theta'(v',u',\lambda). \tag{37}$$

Thus, θ' is a circulation. Then, since $\theta'(s',t',\emptyset) = 1$, if we remove the flow on this edge, which recovers θ, we get a unit $s't'$-flow on G'.

Next we will show that if θ is a unit $s't'$-flow on G', then there exists a negative witness ω in $P_{G,c}$ such that for all $(u,v,\lambda) \in \overrightarrow{E}(G)$, $\theta((u,v,\lambda)^\dagger) = \omega(u) - \omega(v)$.

Define θ' to be the circulation on $\overline{G}^\dagger$ obtained from defining $\theta'(u',v',\lambda) = \theta(u',v',\lambda)$ for all $(u',v',\lambda) \in \overrightarrow{E}(G')$, and $\theta'(s',t',\emptyset) = -\theta'(t',s',\emptyset) = 1$. Then if we define $|\theta'\rangle = \sum_{(u,v,\lambda) \in \overrightarrow{E}(\overline{G}^\dagger)} \theta'(u,v,\lambda)|u,v,\lambda\rangle$, we can express $|\theta'\rangle$ as a linear combination of cycles around the faces of $\overline{G}^\dagger$, $|\theta'\rangle = \sum_{f \in F(\overline{G}^\dagger)} \alpha_f |\overrightarrow{C}_f\rangle + \sum_{f \in F(\overline{G}^\dagger)} \alpha'_f |\overleftarrow{C}_f\rangle$, where if $w_{k+1} = w_1$ and $(w_1, w_2, \lambda_1), \dots, (w_k, w_{k+1}, \lambda_k)$ is a clockwise cycle around f, $|\overrightarrow{C}_f\rangle = \sum_{i=1}^{k} |w_i, w_{i+1}, \lambda_i\rangle$ is the clockwise cycle around the face f, and $|\overleftarrow{C}_f\rangle = \sum_{i=1}^{k} |w_{i+1}, w_i, \lambda_i\rangle$ is the counter-clockwise cycle around f. There is a one-to-one correspondance between vertices in $V(\overline{G}) = V(G)$ and faces in $F(\overline{G}^\dagger)$, so we can define $\omega : V(G) \to \mathbb{R}$ by $\omega(v_f) = \frac{1}{2}(\alpha_f - \alpha'_f)$.

We claim that for all $(u,v,\lambda) \in \overrightarrow{E}(\overline{G})$, $\omega(u) - \omega(v) = \theta'((u,v,\lambda)^\dagger)$. Let (u',v',λ) be any edge in $\overrightarrow{E}(\overline{G}^\dagger)$. This edge is part of a clockwise cycle around one face in $\overline{G}^\dagger$, call it f, and a counter clockwise cycle around one face in $\overrightarrow{G}^\dagger$, call it g. Since these are the only two faces containing the edge (u',v',λ), we must have $\theta'(u',v',\lambda) = \langle u',v',\lambda|\theta'\rangle = \alpha_f + \alpha_g'$. Since $\theta'(u',v',\lambda) = -\theta'(v',u',\lambda)$, we have $\alpha_f + \alpha_g' = -\alpha_f' - \alpha_g$. Thus:

$$\omega(v_f) - \omega(v_g) = \frac{1}{2}\left(\alpha_f - \alpha_f' - \alpha_g + \alpha_g'\right) = \frac{1}{2}\left(\theta'(u',v',\lambda) - \theta'(v',u',\lambda)\right) = \theta'((v_f,v_g,d)^\dagger). \tag{38}$$

In particular, this means that $\omega(s)-\omega(t) = \theta'((s,t,\emptyset)^\dagger) = \theta'(s',t',\emptyset) = 1$, so ω is a negative witness, and for all $(u,v,\lambda) \in \overrightarrow{E}(G)$, $\omega(u) - \omega(v) = \theta((u,v,\lambda)^\dagger)$. □

Now we can prove the main result of this section, Lemma 11:

Lemma 11. *Let G be a planar multigraph with $s,t \in V(G)$ such that $G \cup \{\{s,t\}\}$ is also planar, and let c be a weight function on $E(G)$. Let $x \in \{0,1\}^{E(G)}$. Then $w_+(x, P_{G,c}) = \frac{1}{2}R_{s,t}(G(x),c)$ and $w_-(x,P_{G,c}) = 2R_{s',t'}(G'(x),c')$.*

Proof. If x is a 1-instance, s and t are connected in $G(x)$, so there exists a unit st-flow on $G(x)$, which is a unit st-flow on G that is supported only on $\overrightarrow{E}(G(x))$. Let θ be the flow on $G(x)$ such that $R_{s,t}(G(x),c) = \sum_{(\{u,v\},\lambda)\in E(G(x))} \frac{\theta(u,v,\lambda)^2}{c(\{u,v\},\lambda)}$. By Claim 30, $|w\rangle = \frac{1}{2}\sum_{(u,v,\lambda)} \frac{\theta(u,v,\lambda)}{\sqrt{c(\{u,v\},\lambda)}}|u,v,\lambda\rangle$ is a positive witness in $P_{G,c}$, and since θ is supported on $\overrightarrow{E}(G(x))$, $|w\rangle \in H(x)$, and so $|w\rangle$ is a positive witness for x in $P_{G,c}$. Thus

$$w_+(x,P_{G,c}) \leq \||w\rangle\|^2 = \frac{1}{4}\sum_{(u,v,\lambda)\in\overrightarrow{E}(G(x))} c(\{u,v\},\lambda)\theta(u,v,\lambda)^2 = \frac{1}{2}R_{s,t}(G(x),c). \tag{39}$$

On the other hand, let $|w\rangle$ be an optimal positive witness for x. By Claim 30, $\theta(u,v,\lambda) = \sqrt{c(\{u,v\},\lambda)}(\langle u,v,\lambda|w\rangle - \langle v,u,\lambda|w\rangle)$ is a unit st-flow on G, and since $|w\rangle \in H(x)$, $\theta(u,v,\lambda)$ is only non-zero on $\overrightarrow{E}(G(x))$, so θ is a unit st-flow on $G(x)$. Thus,

$$\begin{aligned} R_{s,t}(G(x),c) &\leq \sum_{(\{u,v\},\lambda)\in E(G(x))} \frac{\theta(u,v,\lambda)^2}{c(\{u,v\},\lambda)} = \frac{1}{2}\sum_{(u,v,\lambda)\in\overrightarrow{E}(G(x))} (\langle u,v,\lambda|w\rangle - \langle v,u,\lambda|w\rangle)^2 \\ &= \sum_{(u,v,\lambda)\in\overrightarrow{E}(G(x))} \langle u,v,\lambda|w\rangle^2 - \sum_{(u,v,\lambda)\in\overrightarrow{E}(G(x))} \langle u,v,\lambda|w\rangle\langle v,u,\lambda|w\rangle \leq 2\||w\rangle\|^2 \end{aligned} \tag{40}$$

where the last inequality is by Cauchy-Schwarz. Thus, $w_+(x,P_{G,c}) = \frac{1}{2}R_{s,t}(G(x))$.

Now we prove that $w_-(x,P_{G,c}) = 2R_{s',t'}(G'(x),c')$. Let $x \in \{0,1\}^{E(G)}$ be such that s and t are not connected in $G(x)$. Fix an optimal negative witness ω for x. By Claim 31 the linear function $\theta : \overrightarrow{E}(G') \to \mathbb{R}$ defined by $\theta((u,v,\lambda)^\dagger) = \omega(u) - \omega(v)$ is a unit $s't'$ -flow on G'. Since ω is a negative witness for x, we also have:

$$\begin{aligned} 0 = \left\|\omega A \Pi_{H(x)}\right\|^2 &= \sum_{(u,v,\lambda)\in\overrightarrow{E}(G(x))} c(\{u,v\},\lambda)(\omega(u)-\omega(v))^2 \\ &= \sum_{(u,v,\lambda)\in\overrightarrow{E}(G(x))} c(\{u,v\},\lambda)\theta((u,v,\lambda)^\dagger)^2 \\ &= \sum_{(u',v',\lambda)\in\overrightarrow{E}(G')\setminus\overrightarrow{E}(G'(x))} \frac{\theta(u',v',\lambda)^2}{c'(\{u',v'\},\lambda)}, \end{aligned} \tag{41}$$

since $(u,v,\lambda) \in \overrightarrow{E}(G(x))$ exactly when $(u,v,\lambda)^\dagger \notin \overrightarrow{E}(G'(x))$. So θ is only supported on $\overrightarrow{E}(G'(x))$, and so it is a unit $s't'$-flow on $G'(x)$. Thus

$$w_-(x,P_{G,c}) = \|\omega A\|^2 = \sum_{(u,v,\lambda)\in\overrightarrow{E}(G)} c(\{u,v\},\lambda)(\omega(u)-\omega(v))^2$$
$$= \sum_{(u',v',\lambda)\in\overrightarrow{E}(G'(x))} \frac{\theta(u',v',\lambda)^2}{c'(\{u',v'\},\lambda)} \geq 2R_{s',t'}(G'(x),c'). \tag{42}$$

For the other direction, let θ be an $s't'$-flow in $G'(x)$ with minimal energy. By Claim 31, there is a negative witness ω such that $\theta((u,v,\lambda)^\dagger) = \omega(u)-\omega(v)$. Since θ is supported on edges $(u',v',\lambda) \in \overrightarrow{E}(G'(x))$, which are exactly those edges such that $(u',v',\lambda)^\dagger \notin \overrightarrow{E}(G(x))$, we have

$$0 = \sum_{\substack{(u,v,\lambda)\\ \in\overrightarrow{E}(G(x))}} c(\{u,v\},\lambda)\theta((u,v,\lambda)^\dagger)^2 = \sum_{\substack{(u,v,\lambda)\\ \in\overrightarrow{E}(G(x))}} c(\{u,v\},\lambda)(\omega(u)-\omega(v))^2 = \left\|\omega A\Pi_{H(x)}\right\|^2, \tag{43}$$

so ω is a negative witness for x in $P_{G,c}$. Thus:

$$w_-(x,P_{G,c}) \leq \|\omega A\|^2 = \sum_{(u,v,\lambda)\in\overrightarrow{E}(G)} c(\{u,v\},\lambda)(\omega(u)-\omega(v))^2$$
$$= \sum_{(u',v',\lambda)\in\overrightarrow{E}(G'(x))} \frac{\theta(u',v',\lambda)^2}{c'(\{u',v'\},\lambda)} = 2R_{s',t'}(G'(x),c'), \tag{44}$$

completing the proof. □

A.1 Time and Space Analysis of the Span Program Algorithm for st-Connectivity

In this section, we will give an upper bound on the time complexity of st-CONN$_G$ in terms of the time complexity of implementing a step of a discrete-time quantum walk on G. The analysis follows relatively straightforwardly from [4, Section 5.3], but we include it here for completeness. At the end of this section, we show the space complexity of the algorithm is $O(\max\{\log|E(G)|, \log|V(G)|\})$.

We first describe the algorithm that can be derived from a span program, following the conventions of [17]. Throughout this section, we will let Π_S denote the orthogonal projector onto an inner product space S. For a span program $P=(H,U,A,\tau)$, the corresponding algorithm performs phase estimation on the unitary $(2\Pi_{H(x)}-I)(2\Pi_{\ker A}-I)$ applied to initial state $|w_0\rangle = A^+\tau$, where $\Pi_{H(x)}$ denotes the orthogonal projector onto $H(x)$, and $\Pi_{\ker A}$ denotes the orthogonal projector onto the kernel of A, and A^+ denotes the pseudo-inverse of A. To decide a function f on domain D, it is sufficient to perform phase estimation to precision $O\left(\sqrt{\max_{x\in D:f(x)=1} w_+(x) \times \max_{x\in D:f(x)=0} w_-(x)}\right)$.

In case of the st-connectivity span program $P_{G,c}$ in (6), it is a simple exercise to see that $2\Pi_{H(x)}-I$ can be implemented in $O(1)$ quantum operations, including 2 queries to x. The reflection $2\Pi_{\ker A}-I$ is independent of x, and so requires 0 queries to implement, however, it could still require a number of gates that grows quickly with the size of G. We will show that implementing $2\Pi_{\ker A}-I$ can be reduced to implementing a discrete-time quantum walk on G, a task which could be quite easy, depending on the structure of G (for example, in the case that G is a complete graph on n vertices, this can be done in $O(\log n)$ gates [4]).

For a multigraph G and weight function c, we define a *quantum walk step on G* to be a unitary $U_{G,c}$ that acts as follows for any $u \in V(G)$:

$$U_{G,c} : |u\rangle|0\rangle \mapsto \frac{1}{\sqrt{\sum_{v,\lambda:(u,v,\lambda)\in\overrightarrow{E}(G)} c(\{u,v\},\lambda)}} \sum_{v,\lambda:(u,v,\lambda)\in\overrightarrow{E}(G)} \sqrt{c(\{u,v\},\lambda)}|u\rangle|u,v,\lambda\rangle. \tag{45}$$

Theorem 13. *Let $P_{G,c} = (H, U, A, \tau)$ be defined as in (6). Let $S_{G,c}$ be an upper bound on the time complexity of implementing $U_{G,c}$. If G has the property that $G \cup \{\{s,t\}\}$ is planar, then the time complexity of deciding st-$\textsc{conn}_{G,D}$ is at most*

$$O\left(\min_c S_{G,c}\sqrt{\max_{x\in D: s,t \text{ are connected}} R_{s,t}(G(x),c) \times \max_{x\in D: s,t \text{ are not connected}} R_{s',t'}(G'(x),c')}\right). \tag{12}$$

This theorem follows from Lemma 32, stated below, and Lemma 33, which deals with the construction of the algorithm's initial state.

Lemma 32. *Let A be defined as in (6). Let $S_{G,c}$ be an upper bound on the time complexity of implementing $U_{G,c}$. Then $2\Pi_{\ker A} - I$ can be implemented in time complexity $O(S_{G,c})$.*

Proof. This analysis follows [4] (see also [17]). Let

$$d(u) = \sum_{v,\lambda:(u,v,\lambda)\in\overrightarrow{E}(G)} c(\{u,v\},\lambda). \tag{46}$$

Define spaces Z and Y as follows.

$$Z = \operatorname{span}\left\{|z_u\rangle := \sum_{v,\lambda:(u,v,\lambda)\in\overrightarrow{E}(G)} \frac{\sqrt{c(\{u,v\},\lambda)}}{\sqrt{2d(u)}}\left(|0,u,u,v,\lambda\rangle + |1,u,v,u,\lambda\rangle\right) : u \in V(G)\right\} \tag{47}$$

$$Y = \operatorname{span}\left\{|y_{u,v,\lambda}\rangle := \frac{|0,u,u,v,\lambda\rangle - |1,v,u,v,\lambda\rangle}{\sqrt{2}} : (u,v,\lambda) \in \overrightarrow{E}(G)\right\} \tag{48}$$

Define isometries whose column-spaces are Z and Y respectively:

$$M_Z = \sum_{u\in V(G)} |z_u\rangle\langle u| \quad \text{and} \quad M_Y = \sum_{(u,v,\lambda)\in\overrightarrow{E}(G)} |y_{u,v,\lambda}\rangle\langle u,v,\lambda|. \tag{49}$$

Now we note that for any $(\{u,v\},\lambda) \in E(G)$, we have the following:

$$\langle z_u|y_{u,v,\lambda}\rangle = \frac{\sqrt{c(\{u,v\},\lambda)}}{2\sqrt{d(u)}}, \quad \text{and} \quad \langle z_v|y_{u,v,\lambda}\rangle = -\frac{\sqrt{c(\{u,v\},\lambda)}}{2\sqrt{d(v)}}. \tag{50}$$

Thus, we can calculate:

$$\begin{aligned}
M_Z^\dagger M_Y &= \sum_{(u,v,\lambda)\in\overrightarrow{E}(G)} \left(\frac{|u\rangle}{2\sqrt{d(u)}} - \frac{|v\rangle}{2\sqrt{d(v)}}\right)\sqrt{c(\{u,v\},\lambda)}\langle u,v,\lambda| \\
&= \sum_{u'\in V(G)} \frac{|u'\rangle\langle u'|}{2\sqrt{d(u')}} \sum_{(u,v,\lambda)\in\overrightarrow{E}(G)} \sqrt{c(\{u,v\},\lambda)}(|u\rangle - |v\rangle)\langle u,v,\lambda| \\
&= \sum_{u'\in V(G)} \frac{|u'\rangle\langle u'|}{2\sqrt{d(u')}} A.
\end{aligned} \tag{51}$$

Note that the rows of $M_Z^\dagger M_Y$ are non-zero multiples of the rows of A, so $\mathrm{row}(M_Z^\dagger M_Y) = \mathrm{row}(A)$, and thus $\ker(M_Z^\dagger M_Y) = \ker A$.

Define $W = (2\Pi_Z - I)(2\Pi_Y - I)$. We now claim that M_Y maps $\ker A$ to the (-1)-eigenspace of W, and $(\ker A)^\perp$ to the 1-eigenspace of W, so that $2\Pi_{\ker A} - I = M_Y^\dagger W M_Y$. To see this, note that if $|\psi\rangle \in \ker A$, then $|\psi\rangle \in \ker(M_Z^\dagger M_Y)$ so $M_Y|\psi\rangle \in \ker M_Z^\dagger = Z^\perp$. Thus $M_Y|\psi\rangle \in Y \cap Z^\perp$, which is in the (-1)-eigenspace of W.

Next, suppose $|\psi\rangle \in (\ker A)^\perp = (\ker(M_Z^\dagger M_Y))^\perp$, so since M_Y is an isometry, $M_Y|\psi\rangle \in (\ker M_Z)^\perp = \mathrm{row} M_Z = Z$. Thus $M_Y|\psi\rangle \in Y \cap Z$, which is in the 1-eigenspace of W.

Thus, we can implement $2\Pi_{\ker A} - I$ by $M_Y^\dagger W M_Y$. It only remains to argue that each of M_Y, $2\Pi_Z - I$ and $2\Pi_Y - I$ can be implemented in time complexity at most $O(S_{G,c})$.

We first show that we can implement the isometry M_Y, or rather a unitary U_Y that acts as $|0\rangle|0\rangle|u,v,\lambda\rangle \mapsto M_Y|u,v,\lambda\rangle = |y_{u,v,\lambda}\rangle$. First, use HX on the first qubit to perform the map:

$$|0\rangle|0\rangle|u,v,\lambda\rangle \mapsto |-\rangle|0\rangle|u,v,\lambda\rangle. \tag{52}$$

Conditioned on the value of the first register, copy either u or v into the second register to get:

$$\frac{1}{\sqrt{2}}(|0,u,u,v,\lambda\rangle - |1,v,u,v,\lambda\rangle) = |y_{u,v,\lambda}\rangle. \tag{53}$$

Thus, we can implement U_Y in the time it takes to write down a vertex of G, $O(\log|V(G)|)$, which is at most $O(S_{G,c})$. Using the ability to implement U_Y, we can implement $2\Pi_Y - I$ as $U_Y R_Y U_Y^\dagger$, where R_Y is the reflection that acts as the identity on computational basis states of the form $|0\rangle|0\rangle|u,v,\lambda\rangle$, and reflects computational basis states without this form.

Next, we implement a unitary U_Z that acts as $|0\rangle|u\rangle|0\rangle \mapsto M_Z|u\rangle = |z_u\rangle$. First, use the quantum walk step $U_{G,c}$, which can be implemented in time $S_{G,c}$, to perform:

$$|+\rangle|u\rangle|0\rangle \mapsto \frac{1}{2\sqrt{d(u)}} \sum_{v,\lambda:(u,v,\lambda)\in\overrightarrow{E}(G)} \sqrt{c(\{u,v\},\lambda)}(|0\rangle+|1\rangle)|u\rangle|u,v,\lambda\rangle. \tag{54}$$

Conditioned on the bit in the first register, swap the third and fourth registers, to get:

$$\frac{1}{2\sqrt{d(u)}} \sum_{v,\lambda:(u,v,\lambda)\in\overrightarrow{E}(G)} \sqrt{c(\{u,v\},\lambda)}(|0\rangle|u\rangle|u,v,\lambda\rangle + |1\rangle|u\rangle|v,u,\lambda\rangle) = |z_u\rangle. \tag{55}$$

The total cost of implementing U_Z is $O(S_{G,c} + \log|V(G)|) = O(S_{G,c})$. Thus, we can implement $2\Pi_Z - I$ in $O(S_{G,c})$ quantum gates. □

Lemma 33. *Let A and τ be defined as in (6). Let $S_{G,c}$ be an upper bound on the complexity of implementing $U_{G,c}$. Then the initial state of the algorithm, $\frac{|w_0\rangle}{\||w_0\rangle\|}$ where $|w_0\rangle = A^+\tau$, can be approximated in time $O(S_{G,c})$.*

Proof. Without loss of generality, we can assume that G includes the edge $(\{s,t\},\emptyset)$ (we can simply not include it in any subgraph). Furthermore, we set $c(\{s,t\},\emptyset) = 1/r$, for some positive r to be specified later, so that $A|s,t,\emptyset\rangle = r^{-1/2}\tau$. This has no effect on other edges in G. Note that

$$\Pi_{(\ker A)^\perp}|s,t,\emptyset\rangle = A^+A|s,t,\emptyset\rangle = r^{-1/2}A^+\tau = r^{-1/2}|w_0\rangle, \tag{56}$$

so

$$|s,t,\emptyset\rangle = r^{-1/2}|w_0\rangle + |w_0^\perp\rangle \tag{57}$$

for some $|w_0^\perp\rangle \in \ker A$. Thus, constant precision phase estimation on $2\Pi_{\ker A} - I$ maps $|s,t,\emptyset\rangle$ to

$$r^{-1/2}|0\rangle|w_0\rangle + |1\rangle|w_0^\perp\rangle. \tag{58}$$

Using quantum amplitude amplification [7], we can amplify the amplitude on the $|0\rangle|w_0\rangle$ part of this arbitrarily close to 1 using a number of calls to $2\Pi_{\ker A} - I$ proportional to $\left\| r^{-1/2}|w_0\rangle \right\|^{-1}$.

In fact, it is straightforward to show that for any $|\mu\rangle \in \mathrm{row} A$, the vector $|\nu\rangle$ with smallest norm that satisfies $A|\nu\rangle = |\mu\rangle$, is $A^+|\mu\rangle$ [17]. Using this fact along with Claim 30 and Definition 9, we have $\| |w_0\rangle \|^2 = R_{s,t}(G,c)$.

Let $R = R_{s,t}(G \setminus \{(\{s,t\},\emptyset)\}, c)$ be the effective resistance of G without the edge $(s,t,\emptyset)$. Now we can think of $(\{s,t\},\emptyset)$ and $G \setminus \{(\{s,t\},\emptyset)\}$ as two graphs in parallel, so using Claim 5, we have

$$\| |w_0\rangle \|^2 = \frac{1}{1/R + 1/r}. \tag{59}$$

Setting $r = R$, we have $\| |w_0\rangle \|^2 = R/2$ and $\| r^{-1/2}|w_0\rangle \|^{-1} = O(1)$. Thus, using $O(1)$ calls to $2\Pi_{\ker A} - I$, we can approximate the initial state $|w_0\rangle$. □

Finally, we note that the space required by the algorithm, in addition to whatever auxiliary space we need to implement $U_{G,c}$, is $O(\max\{\log|E(G)|, \log|V(G)|\})$. U_Y and $U_{G,c}$ each act on a Hilbert space of dimension less than $4|V(G)|^2|E(G)|$, so can in principle be implemented on $O(\max\{\log|E(G)|, \log|V(G)|\})$ qubits, however, a time-efficient implementation of $U_{G,c}$ may also make use of some number $S'_{G,c}$ of auxiliary qubits. We use these unitaries to perform phase estimation on $(2\Pi_{H(x)} - I)(2\Pi_{\ker A} - I)$ to precision

$$O\left(\min_c \sqrt{ \max_{x \in D:\phi(x)=1} R_{s,t}(G_\phi(x), c) \times \max_{x \in D:\phi(x)=0} R_{s',t'}(G'_\phi(x), c') } \right) = O(|E(G)|). \tag{60}$$

Thus we need $O(\log(|E(G)|)$ qubits to store the output of the phase estimation. Putting everything together gives the claimed space complexity.

B Formula Evaluation and st-Connectivity

In this section, we prove the correspondence between evaluating the formula ϕ, and solving st-connectivity on the graph G_ϕ. We first give a formal definition of G_ϕ.

Definition 34 (G_ϕ). *If $\phi = x_i$ is a single-variable formula, then $V(G_\phi) = \{s,t\}$ and $E(G_\phi) = \{(\{s,t\}, x_i)\}$.*

If $\phi = \phi_1 \wedge \cdots \wedge \phi_l$, then define $V(G_\phi) = \{(i,v) : i \in [l], v \in V(G_{\phi_i}) \setminus \{s,t\}\} \cup \{s, s_2, \ldots, s_l, t\}$ and, letting $s_1 = s$ and $s_{l+1} = t$, define:

$$\begin{aligned} E(G_\phi) = &\{(\{(i,u),(i,v)\}, x_j) : i \in [l], u,v \in V(G_{\phi_i}) \setminus \{s,t\}, (\{u,v\},x_j) \in E(G_{\phi_i})\} \\ &\cup \{(\{(i,u), s_i\}, x_j) : i \in [l], u \in V(G_{\phi_i}), (\{s,u\},x_j) \in E(G_{\phi_i})\} \\ &\cup \{(\{(i,u), s_{i+1}\}, x_j) : i \in [l], u \in V(G_{\phi_i}), (\{t,u\},x_j) \in E(G_{\phi_i})\}. \end{aligned} \tag{61}$$

If $\phi = \phi_1 \vee \cdots \vee \phi_l$ *define* $V(G_\phi) = \{(i,v) : i \in [l], v \in V(G_{\phi_i}) \setminus \{s,t\}\} \cup \{s,t\}$ *and*

$$\begin{aligned} E(G_\phi) = &\{(\{(i,u),(i,v)\}, x_j) : i \in [l], u,v \in V(G_{\phi_i}) \setminus \{s,t\}, (\{u,v\},x_j) \in E(G_{\phi_i})\} \\ &\cup \{(\{(i,u),s\},x_j) : i \in [l], u \in V(G_{\phi_i}), (\{u,s\},x_j) \in E(G_{\phi_i})\} \\ &\cup \{(\{(i,u),t\},x_j) : i \in [l], u \in V(G_{\phi_i}), (\{u,t\},x_j) \in E(G_{\phi_i})\}. \end{aligned} \tag{62}$$

In order to prove Lemma 14, we will first prove Lemma 35:

Lemma 35. *For an* AND-OR *formula* ϕ *on* $\{0,1\}^N$*, define* ϕ' *to be the formula obtained by replacing* $\vee$*-nodes with* $\wedge$*-nodes and* $\wedge$*-nodes with* $\vee$*-nodes in* ϕ*. Then for all* $x \in \{0,1\}^N$*, if* $\bar{x}$ *denotes the bitwise complement of* x*, then* $\phi'(x) = \neg\phi(\bar{x})$*. Furthermore up to an isomorphism that maps* s *to* s'*,* t *to* t'*, and an edge labeled by any label* λ *to an edge labeled by* λ*, we have* $G'_\phi = G_{\phi'}$ *and* $G'_\phi(x) = G_{\phi'}(\bar{x})$*.*

Proof. The first part of the proof is by induction. Suppose ϕ has depth 0, so $\phi = x_i$ for some variable x_i. Then $\phi'(x) = \phi(x) = \neg(\phi(\bar{x}))$. So suppose $\phi = \phi_1 \wedge \cdots \wedge \phi_l$. Then $\phi' = \phi'_1 \vee \cdots \vee \phi'_l$. Then by the induction hypothesis,

$$\phi'(x) = \phi'_1(x) \vee \cdots \vee \phi'_l(x) = (\neg\phi_1(\bar{x})) \vee \cdots \vee (\neg\phi_l(\bar{x})) = \neg(\phi_1(\bar{x}) \wedge \cdots \wedge \phi_l(\bar{x})) = \neg\phi(\bar{x}) \tag{63}$$

where the second to last equality is de Morgan's law. The case $\phi = \phi_1 \vee \cdots \vee \phi_l$ is similar.

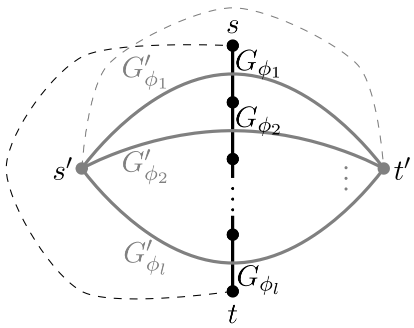

Figure 7: $\overline{G}_\phi$ shown in black, and its dual, $\overline{G}_{\phi'}$, shown in grey. The thick lines represent graphs. Edges in G_{ϕ_i} are dual to edges in G'_{ϕ_i}, and the dotted edge $(\{s,t\},\emptyset)$ is dual to $(\{s',t'\},\emptyset)$.

We will now prove that $\overline{G}_\phi^\dagger = \overline{G}_{\phi'}$, and furthermore, dual edges have the same label, by induction on the depth of ϕ, from which the result follows immediately.

If $\phi = x_i$ is a depth-0 formula, then $\phi' = x_i$. In that case, G_ϕ is just an edge from s to t, labeled by x_i, and G'_ϕ is just an edge from s' to t' labeled x_i, so $G'_\phi = G_{\phi'}$.

For the inductive step, to show that $\overline{G}_\phi$ and $\overline{G}_{\phi'}$ are dual, and therefore $G'_\phi = G_{\phi'}$. It suffices to exhibit a bijection $\zeta : V(\overline{G}_{\phi'}) \to F(\overline{G}_\phi)$ such that $(\{u,v\},x_j) \in E(\overline{G}_{\phi'})$ if and only if the faces $\zeta(u)$ and $\zeta(v)$ are separated by an edge in $E(\overline{G}_\phi)$ with the label x_j. We first consider the case that $\phi = \phi_1 \wedge \cdots \wedge \phi_l$, so $\phi' = \phi'_1 \vee \cdots \vee \phi'_l$. Then, $\overline{G}_\phi$ consists of the graphs $G_{\phi_1}, \dots, G_{\phi_l}$, chained together in series as in Figure 7, with an additional edge from s to t, so the faces of $\overline{G}_\phi$ are exactly all the interior faces of each G_{ϕ_i}, as well as the two faces on either side of the st-edge $(\{s,t\},\emptyset)$, which we will denote by $f^{s'}$ and $f^{t'}$. That is, adding an i to the label of each internal face of G_{ϕ_i}:

$$F(\overline{G}_\phi) = \{(i,f) : i \in [l], f \in F(\overline{G}_{\phi_i}) \setminus \{f^{s'}, f^{t'}\}\} \cup \{f^{s'}, f^{t'}\}, \tag{64}$$

since $F(\overline{G}_{\phi_i}) \setminus \{f^{s'}, f^{t'}\} = F(G_{\phi_i}) \setminus \{f^E\}$, where f^E is the external face. Since $\phi' = \phi'_1 \vee \cdots \vee \phi'_l$ we also have

$$V(\overline{G}_{\phi'}) = V(G_{\phi'}) = \{(i,v) : i \in [l], v \in V(G_{\phi'_i}) \setminus \{s,t\}\} \cup \{s',t'\}, \tag{65}$$

where we will use the labels s' and t' in anticipation of the isometry between G'_ϕ and $G_{\phi'}$.

By the induction hypothesis, for each $i \in [l]$, there exists a bijection $\zeta_i : V(\overline{G}_{\phi'_i}) \to F(\overline{G}_{\phi_i})$ such that for all $u,v \in V(\overline{G}_{\phi'_i}) = V(G_{\phi'_i})$, $(\{u,v\}, x_j) \in E(\overline{G}_{\phi'_i})$ if and only if $\zeta_i(u)$ and $\zeta_i(v)$ are faces separated by an edge with the label x_j. We define ζ by $\zeta(i,v) = (i, \zeta_i(v))$ for all $i \in [l]$ and $v \in V(G_{\phi'_i}) \setminus \{s,t\}$, $\zeta(s') = f^{s'}$, and $\zeta(t') = f^{t'}$. By the induction hypothesis, we can see that for any edge $(\{u,v\}, x_j) \in E(\overline{G}_{\phi'}) \setminus (\{s',t'\}, \emptyset)$, $\zeta(u)$ and $\zeta(v)$ are separated by an edge labeled x_j. This is because this edge is in one of the $G_{\phi'_i}$, and so it has a dual edge in G_{ϕ_i}, by the induction hypothesis (see Figure 7). The only other edge in $\overline{G}_{\phi'}$ is the edge $(\{s',t'\}, \emptyset)$, and $\zeta(s')$ and $\zeta(t')$ are exactly those faces on either side of $(\{s,t\}, \emptyset)$ in $\overline{G}_\phi$, completing the proof that $\overline{G}_\phi^\dagger = \overline{G}_{\phi'}$.

If $\phi = \phi_1 \vee \cdots \vee \phi_l$, then $\phi' = \phi'_1 \wedge \cdots \wedge \phi'_l$, and a nearly identical proof shows that $\overline{G}_\phi^\dagger = \overline{G}_{\phi'}$.

Now that we have shown an isomorphism between G'_ϕ and $G_{\phi'}$, note that $G'_\phi(x)$ is the subgraph of G'_ϕ that includes all those edges where $x_e = 0$. On the other hand $G_{\phi'}(x)$ is the graph that includes all those edges where $x_e = 1$. Taking the bitwise negation of x, we find that $G'_\phi(x) = G_{\phi'}(\bar{x})$. □

Lemma 35 allows us to prove Claim 17:

Claim 17. *If $\phi = \phi_1 \vee \phi_2 \vee \cdots \vee \phi_l$, then $G'_\phi(x)$ is formed by composing $\{G'_{\phi_i}(x)\}_i$ in series, and if $\phi = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_l$, then $G'_\phi(x)$ is formed by composing $\{G'_{\phi_i}(x)\}_i$ in parallel.*

Proof. If $\phi = \phi_1 \vee \cdots \vee \phi_l$, then $\phi' = \phi'_1 \wedge \cdots \wedge \phi'_l$. From Lemma 35, $G'_\phi(x) = G_{\phi'}(\bar{x})$, which using Definition 34 is composed of $\{G_{\phi'_i}(\bar{x})\}_{i=1}^l$ in series. But using the isomorphism of Lemma 35 again, this is just $\{G'_{\phi_i}(x)\}_{i=1}^l$ composed in series. The proof for $\phi = \phi_1 \wedge \cdots \wedge \phi_l$ is similar. □

Now we can prove Lemma 14, which relates the existence of a path in $G_\phi(x)$ or $G'_\phi(x)$ to the value of the function $\phi(x)$:

Lemma 14. *Let ϕ be any* AND-OR *formula on N variables. For every $x \in \{0,1\}^N$, there exists a path from s to t in $G_\phi(x)$ if and only if $\phi(x) = 1$. Furthermore, for every $x \in \{0,1\}^N$, there exists a path from s' to t' in $G'_\phi(x)$ if and only if $\phi(x) = 0$.*

Proof. We will prove the statement by induction on the depth of ϕ. If $\phi = x_j$ has depth 0, then G_ϕ is just an edge $(\{s,t\}, x_j)$, and G'_ϕ is just an edge $(\{s',t'\}, x_j)$. Thus s and t are connected in $G_\phi(x)$ if and only if $x_j = 1$, in which case ϕ evaluates to 1, and s' and t' are connected in $G_{\phi'}$ if and only if $x_j = 0$, in which case ϕ evaluates to 0.

If $\phi = \phi_1 \wedge \cdots \wedge \phi_l$, then G_ϕ consists of $G_{\phi_1}, \ldots, G_{\phi_l}$ connected in series from s to t, and moreover, $G_\phi(x)$ consists of $G_{\phi_1}(x), \ldots, G_{\phi_l}(x)$ connected in series from s to t. Thus an st-path in $G_\phi(x)$ consists of an st-path in $G_{\phi_1}(x)$, followed by an st-path in $G_{\phi_2}(x)$, etc., up to an st-path in $G_{\phi_l}(x)$. Thus, s and t are connected in $G_\phi(x)$ if and only if s and t are connected in each $G_{\phi_1}(x), \ldots, G_{\phi_l}(x)$, which happens if and only if $\phi_1(x) \wedge \cdots \wedge \phi_l(x) = 1$.

On the other hand, by Claim 17, G'_ϕ consists of $G'_{\phi_1}, \ldots, G'_{\phi_l}$ connected in parallel between s' and t'. So any $s't'$-path in $G'_\phi(x)$ is an $s't'$-path in one of the $G'_{\phi_i}(x)$, which is equivalent to an st-path in one of $G_{\phi'_i}(\bar{x})$. Thus, by Lemma 35 s' and t' are connected in

$G'_\phi(x)$ if and only if $\phi'_1(\bar{x}) \vee \cdots \vee \phi'_l(\bar{x}) = \neg\phi_1(x) \vee \cdots \vee \neg\phi_l(x) = 1$. By de Morgan's law is true if and only if $\phi(x) = \phi_1(x) \wedge \cdots \wedge \phi_l(x) = 0$.
The case when $\phi = \phi_1 \vee \cdots \vee \phi_l$ is similar. $\square$

C Classical Lower Bound on Class of Promise Boolean Formulas

In this section, we consider the query complexity of classical algorithms for AND-OR formulas, proving Theorem 19. To do this, we use a recent tool from Ben-David and Kothari [5]. They show that the bounded-error classical randomized query complexity of a function f, denoted $R(f)$, satisfies $R(f) = \Omega(RS(f))$, where $RS(f)$ is the randomized sabotage complexity, defined presently. Furthermore, they prove that for a composed function $f \circ g$, $RS(f \circ g) \geq (RS(f)RS(g))$.

If $f : D \to \{0,1\}$, with $D \subseteq \{0,1\}^N$, let $f_{\text{sab}} : D_{\text{sab}} \to \{0,1\}$, where

$$D_{\text{sab}} = \{x \in \{0,1,*\}^N \cup \{0,1,\dagger\}^N : x \text{ is consistent with } y, y' \in D, \text{ s.t. } f(y) \neq f(y')\}. \tag{66}$$

We say $x \in \{0,1,*,\dagger\}^N$ is consistent with $y \in \{0,1\}^N$ if $x_i = y_i$ for all $i \in [N]$ such that $x_i \in \{0,1\}$. Then, $f_{\text{sab}}(x) = 1$ if $x \in \{0,1,*\}^N$, and $f_{\text{sab}}(x) = 0$ if $x \in \{0,1,\dagger\}^N$. Finally, the randomized sabotage complexity is given by $RS(f) = R_0(f_{\text{sab}})$, where $R_0(f)$ is the zero-error randomized query complexity of f. (For further classical query complexity definitions, see [5].)

We first bound the sabotage complexity of $\text{AND}|_{D_{N,h}}$ and $\text{OR}|_{D'_{N,h}}$:

Lemma 36. *$RS\left(\text{OR}|_{D'_{N,h}}\right) = RS\left(\text{AND}|_{D_{N,h}}\right) = \Omega(N/h)$.*

Proof. For $x \in [D'_{N,h}]_{\text{sab}}$ to be consistent with $y, y' \in D'_{N,h}$ such that $\text{OR}(y) \neq \text{OR}(y')$, we must have that $x \in \{0,*\}^N \cup \{0,\dagger\}^N$. Furthermore, the number of $*$'s or $\dagger$'s in x must be at least h. Thus the sabotaged problem reduces to finding at least one marked item out of n, promised there are at least h marked items. The randomized bounded-error query complexity of this task is $\Omega(N/h)$, and so by Theorem 3 in [5],

$$RS\left(\text{OR}|_{D'_{N,h}}\right) = R_0\left((\text{OR}|_{D'_{N,h}})_{\text{sab}}\right) = \Omega\left(R\left((\text{OR}|_{D'_{N,h}})_{\text{sab}}\right)\right) = \Omega(N/h). \tag{67}$$

The proof for AND is similar. $\square$

The next corollary follows immediately from Lemma 36 and the composition property of sabotage complexity:

Corollary 37. *Let $\phi = \phi_1 \circ \phi_2 \circ \cdots \circ \phi_l$, where for each $i \in [l]$, $\phi_i = \text{OR}|_{D'_{N_i,h_i}}$ or $\phi_i = \text{AND}|_{D_{N_i,h_i}}$. Then $R(\phi) = \Omega\left(\prod_{i=1}^l N_i/h_i\right)$.*

Now that we understand the query complexity of symmetric composed AND-OR formulas, we can look at how this compares to the quantum query complexity of evaluating such functions. We now prove the following lemma.

Lemma 38. *Let $\phi = \phi_1 \circ \phi_2 \circ \cdots \circ \phi_l$, where for each $i \in [l]$, $\phi_i = \text{OR}|_{D'_{N_i,h_i}}$ or $\phi_i = \text{AND}|_{D_{N_i,h_i}}$. Let D be the domain of ϕ. Then*

$$\frac{\prod_{i=1}^l N_i}{\prod_{i=1}^l h_i} = \left(\max_{x \in D:\phi(x)=1} R_{s,t}(G_\phi(x))\right)\left(\max_{x \in D:\phi(x)=0} R_{s,t}(G'_\phi(x))\right). \tag{68}$$

Proof. The proof follows by induction on the number of compositions. First suppose that $\phi = \text{OR}|_{D'_{N,h}}$. Then G_ϕ consists of N edges connected in parallel between s and t, and G'_ϕ consists of N edges connected in series. The only input x such that $\phi(x) = 0$ is the all zeros input. Therefore $\max_{x\in D:\phi(x)=0} R_{s,t}(G'_\phi(x)) = N$. Now notice (using Claim 5) that $R_{s,t}(G_\phi(x)) = 1/|x|$. However because of the domain of OR_{N_i,h_i}, inputs x have $|x| \geq h$, so $\max_{x\in D:\phi(x)=1} R_{s,t}(G_\phi(x)) = 1/h$. Thus

$$N/h = \left(\max_{x\in D:\phi(x)=1} R_{s,t}(G_\phi(x))\right)\left(\max_{x\in D:\phi(x)=0} R_{s,t}(G'_\phi(x))\right). \tag{69}$$

A similar analysis holds for the base case $\phi = \text{AND}|_{D_{N,h}}$.

Now for the inductive step, let $\phi = \phi_1 \circ \xi$ for $\xi = \phi_2 \circ \cdots \circ \phi_l$, where for each i, ϕ_i is either $\text{OR}|_{D'_{N_i,h_i}}$ or $\text{AND}|_{D_{N_i,h_i}}$. Let D_ξ be the domain of ξ and let $x^j \in D_\xi$ denote the bits of x that are input to the j^{th} copy of ξ. Suppose first that $\phi_1 = \text{OR}|_{D'_{N_1,h_1}}$. G'_ϕ is formed by taking the N_1 graphs G'_ξ and connecting them in series. The only way $\phi(x) = 0$ is if the input $x^j \in D_\xi$ to each of the ξ functions satisfies $\xi(x^j) = 0$, so by Claim 5

$$\max_{x\in D:\phi(x)=0} R_{s,t}(G'_\phi(x)) = N_1 \max_{y\in D_\xi:\xi(y)=0} R_{s,t}(G'_\xi(y)). \tag{70}$$

On the other hand, G_ϕ is formed by taking N_1 graphs G_ξ and connecting them in parallel. Using Claim 5, if $x^j \in D_\xi$ is the input to j^{th} function ξ, we have

$$R_{s,t}(G_\phi(x)) = \left(\sum_{j=1}^{N} \frac{1}{R_{s,t}(G_\xi(x^j))}\right)^{-1}. \tag{71}$$

Thus larger values for $R_{s,t}(G_\phi(x))$ come from cases where $R_{s,t}(G_\xi(x^j))$ are large. Now

$$\max_{x\in D_\xi} R_{s,t}(G_\xi(x)) = \infty, \tag{72}$$

which occurs when $\xi(y) = 0$. Because of the promise on the domain of ϕ_1, there must be at least h_1 of the N_1 subformulas ξ that evaluate to 1. On each of those subformulas, we want to have an input $x^j \in D_\xi$ that maximizes the effective resistance of that subformula. Therefore, we have

$$\max_{x\in D:\phi(x)=1} R_{s,t}(G_\phi(x)) = \left(\frac{h_1}{\max_{y\in D_\xi:\xi(y)=1} R_{s,t}(G_{\xi_j}(y)}\right)^{-1} = \frac{\max_{y\in D_\xi:\xi(y)=1} R_{s,t}(G_{\xi_j}(y))}{h_1}. \tag{73}$$

Therefore, using the inductive assumption,

$$\begin{aligned}&\left(\max_{x\in D:\phi(x)=1} R_{s,t}(G_\phi(x))\right)\left(\max_{x\in D:\phi(x)=0} R_{s,t}(G'_\phi(x))\right)\\ =&\frac{N_1}{h_1}\max_{y\in D_\xi:\xi(y)=1} R_{s,t}(G_{\xi_j}(y^j))\max_{y\in D_\xi:\xi(y)=0} R_{s,t}(G'_\xi(y)) = \frac{\prod_{i=1}^l N_i}{\prod_{i=1}^l h_i}.\end{aligned} \tag{74}$$

The inductive step for $\phi_1 = \text{AND}|_{D_{N,h}}$ is similar. ☐

Corollary 37 and Lemma 38 give Theorem 19.

D NAND-tree Proofs

D.1 Relationship Between Faults and Effective Resistance

In this section, we prove Lemma 22:

Lemma 22. *For any $x \in \{0,1\}^{2^d}$, if d is even, then we have $R_{s,t}(G_{\text{NAND}_d}(x)) \leq \mathcal{F}_A(x)$ and $R_{s',t'}(G'_{\text{NAND}_d}(x)) \leq \mathcal{F}_B(x)$, while if d is odd, we have $R_{s,t}(G_{\text{NAND}_d}(x)) \leq 2\mathcal{F}_A(x)$ and $R_{s',t'}(G'_{\text{NAND}_d}(x)) \leq 2\mathcal{F}_B(x)$.*

Proof. We will give a proof for $\mathcal{F}_A(x)$; the case of $\mathcal{F}_B(x)$ is similar.

First, $R_{s,t}(G_{\text{NAND}_d}(x)) = \infty$ if and only if s and t are not connected in $G_{\text{NAND}_d}(x)$, which, by Lemma 14, occurs if and only if x is a 0-instance. This means exactly that x is not A-winnable, which, by Eq. (28), holds if and only if $\mathcal{F}_A(x) = \infty$. Thus, suppose this is not the case, so $\mathcal{F}_A(x) < \infty$.

The rest of the proof is by induction. We need to look at both odd and even cases. For the case of $d = 0$, the only A-winnable input in $\{0,1\}^{2^0}$ is $x = 1$. In that case, using Eq. (28), $\mathcal{F}_A(x) = 1$, since there are no decision nodes for Player A, and since $G_{\text{NAND}_0}(x)$ is just a single edge from s to t, $R_{s,t}(G_{\text{NAND}_0}(x)) = 1$.

Let $x \in \{0,1\}^{2^d}$ be any A-winnable input with $d > 1$. We let x^0 be the first 2^{d-1} bits of x and x^1 be the last 2^{d-1} bits of x, so $x = (x^0, x^1)$.

We first consider odd $d > 1$. Using the definition of G_ϕ from Section 4, and the fact that for d odd, the root node is an $\wedge$-node, we see that $G_{\text{NAND}_d}(x)$ consists of $G_{\text{NAND}_{d-1}}(x^0)$ and $G_{\text{NAND}_{d-1}}(x^1)$ connected in series, so by Claim 5

$$R_{s,t}(G_{\text{NAND}_d}(x)) = R_{s,t}(G_{\text{NAND}_{d-1}}(x^0)) + R_{s,t}(G_{\text{NAND}_{d-1}}(x^1)). \tag{75}$$

Now the root can not be a fault, because it is a decision node for Player B, but we know the tree is A-winnable, so no choice Player B makes would allow her to win the game. Therefore, both subtrees connected to the root node must be A-winnable. Using Eq. (28) we have

$$\mathcal{F}_A(x^0) + \mathcal{F}_A(x^1) \leq \max_{b\in\{0,1\}} 2\mathcal{F}_A(x^b) = 2\mathcal{F}_A(x). \tag{76}$$

Combining Eqs. (75) and (76) and the inductive assumption for even depth trees, we have for odd d,

$$R_{s,t}(G_{\text{NAND}_d}(x)) \leq 2\mathcal{F}_A(x). \tag{77}$$

Now we consider the case that d is even, so the root is a decision node for Player A. Consequently, the root node is a $\vee$-node, so by Claim 5

$$R_{s,t}(G_{\text{NAND}_d}(x)) = \left(\frac{1}{R_{s,t}(G_{\text{NAND}_{d-1}}(x^0))} + \frac{1}{R_{s,t}(G_{\text{NAND}_{d-1}}(x^1))}\right)^{-1}. \tag{78}$$

Suppose the root is a fault. Without loss of generality, let's assume the subtree with input x^0 is not A-winnable. Then $R_{s,t}(G_{\text{NAND}_{d-1}}(x^0)) = \infty$ so Eq. (78) becomes

$$R_{s,t}(G_{\text{NAND}_d}(x)) = R_{s,t}(G_{\text{NAND}_{d-1}}(x^1)). \tag{79}$$

Using the inductive assumption for odd depth trees, Eq. (28), and the fact that the root is a fault, we have

$$R_{s,t}(G_{\text{NAND}_d}(x)) \leq 2\mathcal{F}(x^1) = \mathcal{F}(x). \tag{80}$$

If the root is not a fault, then both $R_{s,t}(G_{\text{NAND}_{d-1}}(x^0))$ and $R_{s,t}(G_{\text{NAND}_{d-1}}(x^1))$ are finite, so from (78), and using the inductive assumption, we have

$$\begin{aligned} R_{s,t}(G_{\text{NAND}_d}(x)) &\leq \frac{1}{2}\max\{R_{s,t}(G_{\text{NAND}_{d-1}}(x^0)), R_{s,t}(G_{\text{NAND}_{d-1}}(x^1))\} \\ &\leq \max\{\mathcal{F}(x^0), \mathcal{F}(x^1)\} = \mathcal{F}(x). \end{aligned} \tag{81}$$

A similar analysis for $\mathcal{F}_B(x)$ completes the proof. □

D.2 Estimating Effective Resistances

In this section, we will prove Lemma 25, which bounds the query complexity of estimating the effective resistance of a graph corresponding to a Boolean formula. In [17], Ito and Jeffery describe a quantum query algorithm to estimate the positive or negative witness size of a span program given access to $\mathcal{O}_x$. We will describe how to use this algorithm to estimate the effective resistance of graphs $G_\phi(x)$ or $G'_\phi(x)$.

Ref. [17] define the *approximate positive and negative witness sizes*, $\tilde{w}_+(x,P)$ and $\tilde{w}_-(x,P)$. These are similar to the positive and negative witness sizes, but with the conditions $|w\rangle \in H(x)$ and $\omega A \Pi_{H(x)} = 0$ relaxed.

Definition 39 (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:*

$$e_+(x) = e_+(x,P) := \min\left\{\left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 : A|w\rangle = \tau\right\}. \tag{82}$$

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*

$$\tilde{w}_+(x) = \tilde{w}_+(x,P) := \min\left\{\||w\rangle\|^2 : A|w\rangle = \tau, \left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 = e_+(x)\right\}. \tag{83}$$

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 40 (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:*

$$e_-(x) = e_-(x,P) := \min\left\{\left\|\omega A \Pi_{H(x)}\right\|^2 : \omega \in \mathcal{L}(U,\mathbb{R}), \omega\tau = 1\right\}. \tag{84}$$

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*

$$\tilde{w}_-(x) = \tilde{w}_-(x,P) := \min\left\{\|\omega A\|^2 : \omega \in \mathcal{L}(U,\mathbb{R}), \omega\tau = 1, \left\|\omega A \Pi_{H(x)}\right\|^2 = e_-(x,P)\right\}. \tag{85}$$

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.

Then Ito and Jeffery give the following result:

Theorem 41 ([17])**.** *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 $\widetilde{W}_- = \widetilde{W}_-(P, f) = \max_{x\in X} \tilde{w}_-(x, P)$. There exists a quantum algorithm that estimates f to relative error ε and that uses $\widetilde{\mathrm{O}}\left(\frac{1}{\varepsilon^{3/2}}\sqrt{w_+(x)\widetilde{W}_-}\right)$ queries. Similarly, let P be a span program such that for all $x \in X$, $f(x) = w_-(x, P)$ and define $\widetilde{W}_+ = \widetilde{W}_+(P, f) = \max_{x\in X} \tilde{w}_+(x, P)$. Then there exists a quantum algorithm that estimates f to accuracy ε and that uses $\widetilde{\mathrm{O}}\left(\frac{1}{\varepsilon^{3/2}}\sqrt{w_-(x)\widetilde{W}_+}\right)$ queries.*

We will apply Theorem 41 to the span program $P_{G,c}$ defined in Eq. (6), with $G = G_\phi$. Throughout this section, we will always set the weight function c to take value one on all edges of the graph G. In this, case, to simplify notation, we will denote the span program $P_{G,c}$ as P_G. To apply Theorem 41, we need bounds on $\widetilde{W}_+(P_{G_\phi})$ and $\widetilde{W}_-(P_{G_\phi})$. We will prove:

Lemma 42. *For any formula ϕ, its $\wedge$-depth is the largest number of $\wedge$-labeled nodes on any path from the root to a leaf. Let ϕ be any* AND-OR *formula with maximum fan-in l, $\wedge$-depth $d_\wedge$, and $\vee$-depth $d_\vee$. Then $\widetilde{W}_+(P_{G_\phi}) \leq \frac{1}{2}l^{d_\wedge}$ and $\widetilde{W}_-(P_{G_\phi}) \leq 2l^{d_\vee}$.*

Then, applying Lemma 42 and Theorem 41, we have the main result of this section, which was first stated in Section 5.2:

Lemma 25 (`Est` Algorithm)**.** *Let ϕ be an* AND-OR *formula with constant fan-in l, $\vee$-depth $d_\vee$ and $\wedge$-depth $d_\wedge$. Then the quantum query complexity of estimating $R_{s,t}(G_\phi(x))$ (resp. $R_{s,t}(G'_\phi(x))$) to relative accuracy ϵ is $\widetilde{\mathrm{O}}\left(\frac{1}{\varepsilon^{3/2}}\sqrt{R_{s,t}(G_\phi(x))l^{d_\vee}}\right)$ (resp. $\widetilde{\mathrm{O}}\left(\frac{1}{\varepsilon^{3/2}}\sqrt{R_{s,t}(G'_\phi(x))l^{d_\wedge}}\right)$).*

Proof of Lemma 25. By Theorem 41, since $R_{s,t}(G_\phi(x)) = \frac{1}{2}w_+(x, P_{G_\phi})$ (Lemma 11), we can estimate this quantity using a number of queries that depends on $\widetilde{W}_-(P_{G_\phi})$. By Lemma 42, we have that $\widetilde{W}_-(P_{G_\phi}) \leq 2l^{d_\vee}$, so we can estimate $w_+(x) = R_{s,t}(G_\phi(x))$ in $\widetilde{O}\left(\frac{1}{\varepsilon^{2/3}}\sqrt{w_+(x)}\widetilde{W}_-^{1/2}\right) = \widetilde{O}\left(\frac{1}{\varepsilon^{2/3}}\sqrt{R_{s,t}(G_\phi(x))l^{d_\vee}}\right)$ queries. Similarly, $R_{s,t}(G'_\phi(x)) = 2w_-(x, P_{G_\phi})$ for all 0-instances, and $\widetilde{W}_+ \leq \frac{1}{2}l^{d_\wedge}$, so we can estimate $R_{s,t}(G'_\phi(x))$ in $\widetilde{O}\left(\frac{1}{\varepsilon^{2/3}}\sqrt{R_{s,t}(G'_\phi(x))l^{d_\wedge}}\right)$ queries. □

To prove Lemma 42, we will use the following observation, which gives an upper bound on the length of the longest self-avoiding st-path in G_ϕ, in terms of the $\wedge$-depth of ϕ. This bound is not tight in general.

Claim 43. *Let ϕ be an* AND-OR *formula with constant fan-in l. If ϕ has $\wedge$-depth $d_\wedge$, then the longest self-avoiding path connecting s and t in G_ϕ has length at most $l^{d_\wedge}$.*

Proof. We will prove the statement by induction. If ϕ has $\wedge$-depth $d_\wedge = 0$, then it has no $\wedge$-nodes. Thus, it is easy to see that G_ϕ has only two vertices, s and t, with some number of edges connecting them, so every st-path has length 1.

Suppose ϕ has $\wedge$-depth $d_\wedge > 0$. First, suppose $\phi = \phi_1 \wedge \dots \wedge \phi_l$. Then since G_ϕ consists of $G_{\phi_1}, \dots, G_{\phi_l}$ connected in series, any st-path in G_ϕ consists of an st-path in G_{ϕ_1}, followed by an st-path in G_{ϕ_2}, etc. up to an st-path in G_{ϕ_l}, so if $d_\wedge(\phi_i)$ is the $\wedge$-depth of ϕ_i, then the longest st-path in G_ϕ has length at most:

$$l^{d_\wedge(\phi_1)} + \dots + l^{d_\wedge(\phi_l)} \leq l l^{d_\wedge - 1} = l^{d_\wedge}. \tag{86}$$

If $\phi = \phi_1 \vee \cdots \vee \phi_l$, then $\max_i d_\wedge(\phi_i) = d_\wedge(\phi) = d_\wedge$, and G_ϕ consists of $G_{\phi_1}, \ldots, G_{\phi_l}$, connected in parallel. Any self-avoiding st-path must include exactly one edge adjacent to s and one edge adjacent to t. However, any path that includes an edge from G_{ϕ_i} and G_{ϕ_j} for $i \neq j$ must go through s or t, so it must have more than one edge adjacent to s, or more than one edge adjacent to t, so such a path can never be a self-avoiding st-path. Thus, any self-avoiding st-path must be contained completely in one of the G_{ϕ_i}. The longest such path is thus the longest self-avoiding st-path in any of the G_{ϕ_i}, which, by induction, is $\max_i l^{d_\wedge(\phi_i)} = l^{d_\wedge}$. $\square$

Now we can prove Lemma 42:

Proof of Lemma 42. To begin, we will prove the upper bound on $\widetilde{W}_+$. Suppose $|\tilde{w}\rangle$ is an optimal approximate positive witness for x. By Claim 30, if $|\tilde{w}\rangle$ is an approximate positive witness, then since $A|\tilde{w}\rangle = \tau$, and c has unit value on all edges of G, $\theta(u,v,\lambda) = \langle u,v,\lambda|\tilde{w}\rangle - \langle v,u,\lambda|\tilde{w}\rangle$ is a unit flow on G. Since $|\tilde{w}\rangle$ is an approximate positive witness for x, it has minimal error for x, so it minimizes $\left\|\Pi_{H(x)^\perp}|\tilde{w}\rangle\right\|^2$, and since it is optimal, it minimizes $\||\tilde{w}\rangle\|^2$ over all approximate positive witnesses. Define $|\theta\rangle = \sum_{(u,v,\lambda)\in\overrightarrow{E}(G)} \theta(u,v,\lambda)|u,v,\lambda\rangle$, so we know that $\frac{1}{2}|\theta\rangle$ also maps to τ under A, so is also a positive witness in P_{G_ϕ}.Then we have

$$\left\|\Pi_{H(x)^\perp}|\theta\rangle\right\|^2 = 2 \sum_{\substack{(u,v,\lambda)\in \\ \overrightarrow{E}(G)\setminus\overrightarrow{E}(G(x))}} \langle u,v,\lambda|\tilde{w}\rangle^2 - 2 \sum_{\substack{(u,v,\lambda)\in \\ \overrightarrow{E}(G)\setminus\overrightarrow{E}(G(x))}} \langle u,v,\lambda|\tilde{w}\rangle\langle v,u,\lambda|\tilde{w}\rangle \leq \left\|2\Pi_{H(x)^\perp}|\tilde{w}\rangle\right\|^2, \tag{87}$$

where the last inequality uses Cauchy-Schwarz, so $\frac{1}{2}|\theta\rangle$ is also an approximate positive witness for x. Similarly,

$$\||\theta\rangle\|^2 \leq \|2|\tilde{w}\rangle\|^2, \tag{88}$$

so $\frac{1}{2}|\theta\rangle$ is optimal.

By Claim 29, we can consider a decomposition of $|\theta\rangle$ into self-avoiding paths p_i and cycles c_i such that all cycles are disjoint from all paths, $|\theta\rangle = \sum_{i=1}^{r} \alpha_i|p_i\rangle + \sum_{i=1}^{r'} \beta_i|c_i\rangle$, where for each i,

$$|p_i\rangle = \sum_{j=1}^{L_i} |u_j^{(i)}, u_{j+1}^{(i)}, \lambda_{i,j}\rangle - \sum_{j=1}^{L_i} |u_{j+1}^{(i)}, u_j^{(i)}, \lambda_{i,j}\rangle, \tag{89}$$

$$|c_i\rangle = \sum_{j=1}^{L_i'} |v_j^{(i)}, v_{j+1}^{(i)}, \lambda_{i,j}'\rangle - \sum_{j=1}^{L_i'} |v_{j+1}^{(i)}, v_j^{(i)}, \lambda_{i,j}'\rangle \tag{90}$$

where $v_{L_j'+1}^{(i)} = v_1^{(i)}$ and $\{\lambda_{i,j}\}_{i,j} \cap \{\lambda_{i,j}'\}_{i,j} = \emptyset$. It's easy to see (in the case of unit edge weights) that $A|c_i\rangle = 0$ for all i, so

$$A\frac{1}{2}\sum_{i=1}^{r} \alpha_i|p_i\rangle = A\frac{1}{2}|\theta\rangle = \tau. \tag{91}$$

Let $|\theta'\rangle = \sum_{i=1}^{r} \alpha_i|p_i\rangle$. Then since c_i and p_j have no common edges, we have $\langle c_i|p_j\rangle = 0$, and also $\langle c_i|(I - \Pi_{H(x)})|p_j\rangle = 0$, so the error of $\frac{1}{2}|\theta'\rangle$ is $\frac{1}{4}\left\|\Pi_{H(x)^\perp}|\theta'\rangle\right\|^2 \leq \frac{1}{4}\left\|\Pi_{H(x)^\perp}|\theta\rangle\right\|^2$, so $\frac{1}{2}|\theta'\rangle$ also has minimal error. Furthermore, $\||\theta'\rangle\|^2 \leq \||\theta\rangle\|^2$, with equality if and only if there are no cycles in the decomposition. By the optimality of $\frac{1}{2}|\theta\rangle$ as an approximate positive witness for x, we can conclude that $|\theta\rangle = \sum_{i=1}^r \alpha_i |p_i\rangle$, and since $A|p_i\rangle = 2\tau$ for all i, and $A|\theta\rangle = 2\tau$, we have $\sum_{i=1}^r \alpha_i = 1$. Then

$$\||\theta\rangle\|^2 \leq \max_i \||p_i\rangle\|^2 = \max_i 2L_i. \tag{92}$$

Since the longest self-avoiding st-path in G_ϕ has length at most $l^{d_\wedge}$, and each L_i is the length of a self-avoiding path in G_ϕ, we have $\tilde{w}_+(x, P_{G_\phi}) \leq \frac{1}{4} 2 l^{d_\wedge} = \frac{1}{2} l^{d_\wedge}$. Thus $\widetilde{W}_+(P_{G_\phi}) \leq \frac{1}{2} l^{d_\wedge}$.

Next we prove the bound on $\widetilde{W}_-$. A min-error approximate negative witness for x in P_{G_ϕ} is a function $\omega : V(G_\phi) \to \mathbb{R}$ such that $\omega\tau = \omega(s) - \omega(t) = 1$, and $\left\|\omega A \Pi_{H(x)}\right\|^2 = \sum_{(u,v,\lambda)\in \overrightarrow{E}(G_\phi(x))} (\omega(u) - \omega(v))^2$ is minimized. By Claim 31, since $\omega\tau = 1$, the function $\theta : \overrightarrow{E}(G'_\phi) \to \mathbb{R}$ defined by $\theta((u,v,\lambda)^\dagger) = \omega(u) - \omega(v)$ is a unit $s't'$-flow on $G'_\phi = G_{\phi'}$, and the witness complexity is

$$\|\omega A\|^2 = \sum_{(u,v,\lambda)\in \overrightarrow{E}(G_\phi)} (\omega(u) - \omega(v))^2 = \sum_{(u',v',\lambda)\in \overrightarrow{E}(G'_\phi)} \theta(u',v',\lambda)^2 = \||\theta\rangle\|^2 \tag{93}$$

where we create $|\theta\rangle$ from θ in the usual way. By an argument similar to the previous argument, if ω is an optimal approximate negative witness for x, then $\||\theta\rangle\|^2$ is upper bounded by twice the length of the longest self-avoiding $s't'$-path in $G'_\phi = G_{\phi'}$. By Lemma 35 and Claim 43, this is upper bounded by $2l^{d_\wedge(\phi')} = 2l^{d_\vee(\phi)}$, where $d_\wedge(\phi')$ is the $\wedge$-depth of ϕ', and $d_\vee = d_\vee(\phi)$ is the $\vee$-depth of ϕ. Thus $\tilde{w}_-(x, P_{G_\phi}) \leq 2l^{d_\vee}$, and so $\widetilde{W}_- \leq 2l^{d_\vee}$. $\square$

D.3 Winning the NAND-tree

We now analyze the algorithm for winning the game associated with a NAND-tree, proving Lemma 26 and Theorem 27.

Lemma 26. *Let $x^0, x^1 \in \{0,1\}^{2^d}$ be instances of NAND_d with at least one of them a 1-instance. Let $N = 2^d$, and $w_{\min} = \min\{R_{s,t}(G_{\mathrm{NAND}_d}(x^0)), R_{s,t}(G_{\mathrm{NAND}_d}(x^1))\}$. Then* `Select`$(x^0, x^1)$ *terminates after $\widetilde{O}\left(N^{1/4}\sqrt{w_{\min}}\right)$ queries to (x^0, x^1) and outputs b such that $R_{s,t}(G_{\mathrm{NAND}_d}(x^b)) \leq 2R_{s,t}(G_{\mathrm{NAND}_d}(x^{\bar{b}}))$ with bounded error.*

Proof. Since at least one of x^0 and x^1 is a 1-instance, using the description of `Select` in Section 5.2, at least one of the programs will terminate. Suppose without loss of generality that `Est`(x^0) is the first to terminate, outputting w_0. Then there are two possibilities: `Est`(x^1) does not terminate after $p(d)\sqrt{w_0}N^{1/4}$ steps, in which case, $R_{s,t}(G_{\mathrm{NAND}_d}(x^0)) \leq 2R_{s,t}(G_{\mathrm{NAND}_d}(x^1))$, and `Select` outputs 0; or `Est`(x^1) outputs w_1 before $p(d)\sqrt{w_0}N^{1/4}$ steps have passed and `Select` outputs b such that $w_b \leq w_{\bar{b}}$.

We will prove the first case by contradiction. Suppose

$$2R_{s,t}(G_{\mathrm{NAND}_d}(x^1)) < R_{s,t}(G_{\mathrm{NAND}_d}(x^0)). \tag{94}$$

Then `Est`(x^1) must terminate after

$$p(d)\sqrt{R_{s,t}(G_{\mathrm{NAND}_d}(x^1))}N^{1/4} \leq \frac{1}{\sqrt{2}} p(d)\sqrt{R_{s,t}(G_{\mathrm{NAND}_d}(x^0))}N^{1/4} \tag{95}$$

steps. In `Select`, we run `Est` to relative accuracy $\varepsilon = 1/3$, so we have

$$|w_0 - R_{s,t}(G_{\mathrm{NAND}_d}(x^0))| \leq \frac{1}{3} R_{s,t}(G_{\mathrm{NAND}_d}(x^0)), \tag{96}$$

and so

$$w_0 \geq \frac{2}{3} R_{s,t}(G_{\text{NAND}_d}(x^0)). \tag{97}$$

Plugging Eq. (97) into Eq. (95), we have $\texttt{Est}(x^1)$ must terminate after $\frac{1}{\sqrt{2}} p(d)\sqrt{\frac{3}{2} w_0} N^{1/4} < p(d)\sqrt{w_0} N^{1/4}$ steps, which is a contradiction.

Thus, $R_{s,t}(G_{\text{NAND}_d}(x^0)) \leq 2R_{s,t}(G_{\text{NAND}_d}(x^1))$, so outputting 0 is correct. Furthermore, since we terminate after $p(d)\sqrt{w_0}N^{1/4} = \widetilde{O}(\sqrt{R_{s,t}(G_{\text{NAND}_d}(x^0))} N^{1/4})$ steps, and since $R_{s,t}(G_{\text{NAND}_d}(x^0)) = O(R_{s,t}(G_{\text{NAND}_d}(x^1)))$, 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 $|w_b - R_{s,t}(G_{\text{NAND}_d}(x^b))| \leq \varepsilon R_{s,t}(G_{\text{NAND}_d}(x^b))$ for $b = 0, 1$. Suppose $w_b \leq w_{\bar{b}}$. We then have

$$\frac{R_{s,t}(G_{\text{NAND}_d}(x^b))}{R_{s,t}(G_{\text{NAND}_d}(x^{\bar{b}}))} \leq \frac{R_{s,t}(G_{\text{NAND}_d}(x^b))}{w_b} \frac{w_{\bar{b}}}{R_{s,t}(G_{\text{NAND}_d}(x^{\bar{b}}))} \leq \frac{1+\varepsilon}{1-\varepsilon} = \frac{4/3}{2/3} = 2. \tag{98}$$

Thus $R_{s,t}(G_{\text{NAND}_d}(x^b)) \leq 2R_{s,t}(G_{\text{NAND}_d}(x^{\bar{b}}))$, 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 $\widetilde{O}\left(\sqrt{R_{s,t}(G_{\text{NAND}_d}(x^1))} N^{1/4}\right)$ steps, and by assumption, $\texttt{Est}(x^1)$ terminated after less than $p(d)\sqrt{w_0}N^{1/4} = \widetilde{O}\left(\sqrt{R_{s,t}(G_{\text{NAND}_d}(x^0))} N^{1/4}\right)$ steps, so the total running time is at most $\widetilde{O}\left(N^{1/4}\sqrt{w_{\min}}\right)$. □

Theorem 27. *Let $x \in \{0,1\}^N$ for $N = 2^d$ be an A-winnable input to NAND_d. 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 $\widetilde{O}\left(N^{1/4}\sqrt{R_{s,t}(G_{\text{NAND}_d}(x))}\right)$ queries on average, where the average is taken over the randomness of Player B's choices.*

Proof. First note that Player A must make $O(d)$ choices over the course of the game. 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 $O(1/d)$, and 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 $p(d)$ be a non-decreasing polynomial function in d such that $\texttt{Select}$, on inputs $x^0, x^1 \in \{0,1\}^{2^d}$, terminates in at most $p(d)2^{d/4}\sqrt{\min\{R_{s,t}(G_{\text{NAND}_d}(x^0)), R_{s,t}(G_{\text{NAND}_d}(x^1))\}}$ queries. Then we will prove that for trees of odd depth d, the expected number of queries by Player A over the course of the game is at most $p(d)2^{d/4+5}\sqrt{R_{s,t}(G_{\text{NAND}_d}(x))}$, while for even depth trees, it is at most $p(d)2^{d/4+11/2}\sqrt{R_{s,t}(G_{\text{NAND}_d}(x))}$, thus proving the main result.

We prove the result by induction on the depth of the tree. For depth zero trees, there are no decisions, $N = R_{s,t}G_{\text{NAND}_0}(x) = 1$, so the result holds.

For the inductive case, we treat odd and even depth cases separately. First consider an instance of NAND_d with $d > 0$, d odd. Thus $\text{NAND}_d(x) = \text{NAND}_{d-1}(x^0) \wedge \text{NAND}_{d-1}(x^1)$, where $x=(x^0,x^1)$. Because the root is at odd distance from the leaves, the root is a decision node for Player B. Because we are in an A-winnable tree, no matter which choice Player B makes, we will end up at an A-winnable subtree of depth $d-1$, so the inductive assumption holds for those trees. That is, the expected number of queries for Player A must make to win the subtree with input x^b (for $b\in\{0,1\}$) averaged over Player B's choices is at most

$$p(d-1)2^{(d-1)/4+11/2}\sqrt{R_{s,t}(G_{\text{NAND}_{d-1}}(x^b))}. \tag{99}$$

We are assuming that Player B chooses left and right with equal probability. Thus, the expected number of queries that Player A must make over Player B's choices throughout the game is at most

$$\begin{aligned}
&\frac{1}{2}\Big(p(d-1)2^{(d-1)/4+11/2}\sqrt{R_{s,t}(G_{\text{NAND}_{d-1}}(x^0))}\\
&\qquad\qquad +p(d-1)2^{(d-1)/4+11/2}\sqrt{R_{s,t}(G_{\text{NAND}_{d-1}}(x^1))}\Big)\\
&\leq p(d-1)2^{(d-1)/4+11/2}\sqrt{\frac{1}{2}\left(R_{s,t}(G_{\text{NAND}_{d-1}}(x^0))+R_{s,t}(G_{\text{NAND}_{d-1}}(x^1))\right)} \quad \text{by Jensen's ineq.,}\\
&= p(d-1)2^{(d-1)/4+11/2}\sqrt{\frac{1}{2}R_{s,t}(G_{\text{NAND}_d}(x))} \quad \text{by Claim 5,}\\
&\leq p\,(d)\,2^{d/4-1/4+11/2-1/2}\sqrt{R_{s,t}(G_{\text{NAND}_d}(x))}\\
&\leq p\,(d)\,2^{d/4+5}\sqrt{R_{s,t}(G_{\text{NAND}_d}(x))},
\end{aligned} \tag{100}$$

proving the case for odd d.

Now consider an instance of NAND_d with $d>0$, d even. Thus $\text{NAND}_d(x)=\text{NAND}_{d-1}(x^0)\vee \text{NAND}_{d-1}(x^1)$, where $x=(x^0,x^1)$. Because the root is at even distance from the leaves, the root is a decision node for Player A. Player A runs $\texttt{Select}(x^0,x^1)$, which returns $b\in\{0,1\}$ such that (by Lemma 26)

$$R_{s,t}(G_{\text{NAND}^{d-1}}(x^b))\leq 2R_{s,t}(G_{\text{NAND}^{d-1}}(x^{\bar{b}})), \tag{101}$$

which requires at most

$$\min_{b^*\in\{0,1\}} p\,(d-1)\,2^{(d-1)/4}\sqrt{R_{s,t}(G_{\text{NAND}_{d-1}}(x^{b^*}))} \tag{102}$$

queries.

After making the choice to move to the subtree with input x^b, by the inductive assumption, the expected number of queries that Player A need to make throughout the rest of the game (averaged over Player B's choices) is

$$p\,(d-1)\,2^{d/4+5}\sqrt{R_{s,t}(G_{\text{NAND}_{d-1}}(x^b))}. \tag{103}$$

There are two cases to consider. If $R_{s,t}(G_{\text{NAND}_{d-1}}(x^b))\leq R_{s,t}(G_{\text{NAND}_{d-1}}(x^{\bar{b}}))$, then combining Eq. (102) and Eq. (103), we have that the total number of queries averaged over Player B's choices is

$$\begin{aligned}
&p\,(d-1)\,2^{(d-1)/4}\sqrt{R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^b))} + p\,(d-1)\,2^{(d-1)/4+5}\sqrt{R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^b))}\\
\le &p\,(d-1)\,2^{(d-1)/4}\sqrt{R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^b))}(1+2^5)\\
\le &p\,(d-1)\,2^{(d-1)/4+5+1/16}\sqrt{R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^b))}\\
\le &p\,(d-1)\,2^{(d-1)/4+5+1/16+1/2}\sqrt{R_{s,t}(G_{\mathrm{NAND}_{d}}(x))}\\
\le &p\,(d)\,2^{d/4+11/2}\sqrt{R_{s,t}(G_{\mathrm{NAND}_{d}}(x))}
\end{aligned} \tag{104}$$

where we've used $R_{s,t}(G_{\mathrm{NAND}_{d}}(x)) = \left(R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^0))^{-1} + R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^1))^{-1}\right)^{-1}$ from Claim 5 and the fact that $R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^b)) \le R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^{\bar{b}}))$ to bound the value $R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^b))$ by $2R_{s,t}(G_{\mathrm{NAND}_{d}}(x))$. This proves the even induction step for this case.

The other case is if $R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^b)) > R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^{\bar{b}}))$. In that case, again using the fact that $R_{s,t}(G_{\mathrm{NAND}_{d}}(x)) = \left(R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^0))^{-1} + R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^1))^{-1}\right)^{-1}$, we have

$$R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^{\bar{b}})) = R_{s,t}(G_{\mathrm{NAND}_{d}}(x))\left(1+\frac{R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^{\bar{b}}))}{R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^{b}))}\right)^{-1} \le \frac{2}{3}R_{s,t}(G_{\mathrm{NAND}_{d}}(x)), \tag{105}$$

where the inequality follows from Eq. (101). Thus, the average total number of queries is

$$\begin{aligned}
&p\,(d-1)\,2^{(d-1)/4}\sqrt{R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^{\bar{b}}))} + p\,(d-1)\,2^{(d-1)/4+5}\sqrt{R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^b))}\\
\le &p(d-1)2^{(d-1)/4}\left(\sqrt{R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^{\bar{b}}))} + 2^5\sqrt{2R_{s,t}(G_{\mathrm{NAND}_{d-1}}(x^{\bar{b}}))}\right)\\
\le &p(d-1)2^{(d-1)/4}(1+2^{5+1/2})\sqrt{\frac{2}{3}R_{s,t}(G_{\mathrm{NAND}_{d}}(x))}\\
\le &p(d)2^{d/4-1/4+5}\sqrt{R_{s,t}(G_{\mathrm{NAND}_{d}}(x))}\\
\le &p(d)2^{d/4+5}\sqrt{R_{s,t}(G_{\mathrm{NAND}_{d}}(x))}.
\end{aligned} \tag{106}$$

This proves the induction step for the other case. □