Quantum Algorithms for Connectivity and Related Problems

Michael Jarret
Perimeter Institute, Waterloo, ON, Canada
mjarret@perimeterinstitute.ca

Stacey Jeffery
Qusoft CWI, Amsterdam, The Netherlands
jeffery@cwi.nl

Shelby Kimmel
Middlebury College, Middlebury, VT, USA
skimmel@middlebury.edu

Alvaro Piedrafita
Qusoft CWI, Amsterdam, The Netherlands
piedrafita@cwi.nl

Abstract

An important family of span programs, *st*-connectivity span programs, have been used to design quantum algorithms in various contexts, including a number of graph problems and formula evaluation problems. The complexity of the resulting algorithms depends on the largest *positive witness size* of any 1-input, and the largest *negative witness size* of any 0-input. Belovs and Reichardt first showed that the positive witness size is exactly characterized by the effective resistance of the input graph, but only rough upper bounds were known previously on the negative witness size. We show that the negative witness size in an *st*-connectivity span program is exactly characterized by the *capacitance* of the input graph. This gives a tight analysis for algorithms based on *st*-connectivity span programs on any set of inputs.

We use this analysis to give a new quantum algorithm for estimating the capacitance of a graph. We also describe a new quantum algorithm for deciding if a graph is connected, which improves the previous best quantum algorithm for this problem if we're promised that either the graph has at least $\kappa > 1$ components, or the graph is connected and has small *average resistance*, which is upper bounded by the diameter. We also give an alternative algorithm for deciding if a graph is connected that can be better than our first algorithm when the maximum degree is small. Finally, using ideas from our second connectivity algorithm, we give an algorithm for estimating the *algebraic connectivity* of a graph, the second largest eigenvalue of the Laplacian.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Electrical networks, Quantum algorithms, Span programs, Connectivity, Graph theory

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.49

Acknowledgements SJ is supported by an NWO WISE Grant and NWO Veni Innovational Research Grant under project number 639.021.752. SK completed some of this work while at the Joint Center for Quantum Information and Computer Science (QuICS) at the University of Maryland. This research was supported in part by Perimeter Institute for Theoretical Physics. Research at Perimeter Institute is supported by the Government of Canada through Industry Canada and by the Province of Ontario through the Ministry of Economic Development and Innovation.

26th Annual European Symposium on Algorithms (ESA 2018).
Editors: Yossi Azar, Hannah Bast, and Grzegorz Herman; Article No. 49; pp. 49:1–49:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Span programs are an algebraic model of computation first developed by Karchmer and Wigderson [10] to study classical logspace complexity, and introduced to the study of quantum algorithms by Reichardt and Špalek [15]. In [14, 12], Reichardt used the concept of span programs to prove that the general adversary bound gives a tight lower bound on the quantum query complexity of any given decision problem, thus showing the deep connection between span programs and quantum query algorithms.

Given a span program, a generic transformation compiles it into a quantum algorithm, whose query complexity is analyzed by taking the geometric mean of two quantities: the largest *positive witness size* of any 1-input; and the largest *negative witness size* of any 0-input. Thus, in order to analyze the query complexity of an algorithm obtained in this way, it is necessary to characterize, or at least upper bound, these quantities.

The relationship between quantum query algorithms and span programs is potentially a powerful tool, but this correspondence alone is not a recipe for finding such an algorithm, and producing an optimal span program for a given problem is generally difficult. Despite this difficulty, a number have been found for important problems such as k-distinctness [2], formula evaluation [15, 13], and st-connectivity [4]. The latter span program is of particular importance, as it has been applied to a number of graph problems [5], to generic formula evaluation problems [9], and underlies the learning graph framework [3]. The st-connectivity based algorithms are also of interest because, unlike with generic span program algorithms, it is often possible to analyze not only query complexity, but also the time complexity.

While span program algorithms are universal for quantum query algorithms, it can also be fruitful to analyze the unitaries used in these algorithms in ways that are different from how they appear in the standard span program algorithm. For example, Ref. [7] derives an algorithm to estimate span program witness sizes based on unitaries that appear in the span program algorithm. We will take a similar approach in this paper, deriving new algorithms based on unitaries that appear in the span program algorithm for st-connectivity.

The problems of st-connectivity and connectivity will be considered in this paper. For a family of undirected graphs G on N edges, for $N \in \mathbb{N}$, and vertex set containing s and t, the problem st-CONN_G is the following: Given $x \in \{0,1\}^{E(G)}$, decide if there is a path from s to t in $G(x)$, where $G(x)$ is the subgraph of G obtained by including an edge e if $x_e = 1$[1]. Similarly, the problem of CONN_G is the following: Given $x \in \{0,1\}^{E(G)}$, determine if every vertex in $G(x)$ is connected to every other vertex in $G(x)$.

1.1 Contributions

1. We provide a complete characterization of the query complexity of the st-connectivity span program algorithm. We do this by showing that the negative witness size of the st-connectivity span program is exactly the *effective capacitance* of the input graph. (The positive witness size for this span program was previously known to be exactly the *effective resistance* [4, 9].) The *effective capacitance* is a measure that depends on the size and number of cuts between s and t (in the case they are disconnected), and is commonly used to analyze electrical networks of capacitors. This characterization tells us that quantum algorithms can quickly decide st-connectivity on graphs that are promised to have either small effective resistance or small effective capacitance.

[1] We can consider more complicated ways of associating edges with input variables in 2.2, but the basic idea is captured by this simpler picture.

2. We describe a new quantum algorithm for estimating the effective capacitance of an input graph $G(x)$ to multiplicative error ε, with complexity $\widetilde{O}(\varepsilon^{-3/2}\sqrt{C_{s,t}(G(x))p})$, where $C_{s,t}(G(x))$ is the effective-capacitance between s and t in $G(x)$, and p is the length of the longest self-avoiding st-path in G.
3. We create and analyze a new algorithm for $\textsc{conn}_G$. Previously, for a graph with n vertices, an optimal $\widetilde{O}(n^{3/2})$ upper bound on the time complexity of this problem was known [6], and an optimal span-program-based quantum algorithm was presented by Āriņš [1], which also uses only $O(\log n)$ space. If R upper bounds the average resistance of any connected input, all disconnected inputs have at least $\kappa > 1$ components, and U is the cost of taking a step of a quantum walk on G then our algorithm has the following properties:
   - For graphs without multi-edges, our algorithm has query complexity $O(n\sqrt{R/\kappa})$ and time complexity $\widetilde{O}(n\sqrt{R/\kappa}\mathsf{U})$.
   - For graphs with multi-edges, our algorithm has query complexity $O(\frac{n^{3/4}\sqrt{Rd_{\max}(G)}}{\kappa^{1/4}})$, where $d_{\max}(G)$ is the maximum degree of any vertex in the graph, and time complexity $\widetilde{O}(n^{3/4}\sqrt{Rd_{\max}(G)}/\kappa^{1/4}\mathsf{U})$.
   - Our algorithm uses $O(\log n)$ space.
   - Our algorithm is the first connectivity algorithm to explicity apply to cases where G is not necessarily the complete graph.
   - In the worst case, our algorithm achieves the optimal query complexity of $O(n^{3/2})$.
4. We present an alternative approach to deciding graph connectivity using phase estimation on a unitary derived from the st-connectivity span program. This phase estimation uses a different initial state from that used in the span program algorithm. We first show that the quantum query complexity of deciding $\textsc{conn}_G$ is $O(\sqrt{nd_{\max}(G)/(\kappa\lambda)})$, when either $G(x)$ is connected and the second smallest eigenvalue of the Laplacian of $G(x)$, $\lambda_2(G(x))$, is at least λ, or $G(x)$ has at least $\kappa > 1$ connected components. We are able to give time-efficient versions of our second algorithm in two contexts:
   a. Under the promise that if $G(x)$ is connected, then $\lambda_2(G(x)) \geq \lambda$, and otherwise $G(x)$ has at least κ connected components, we can solve $\textsc{conn}_G$ in time complexity $\widetilde{O}\left(\sqrt{\frac{nd_{\text{avg}}(G)}{\kappa\lambda_2(G)}}\left(\mathsf{S}+\sqrt{\frac{d_{\max}(G)}{\lambda}}\mathsf{U}\right)\right)$, where U is the complexity of implementing a step of a quantum walk on G, S is the cost generating a quantum state corresponding to the stationary distribution of a random walk on G, and $d_{\text{avg}}(G)$ is the average degree of the vertices of G.
   b. When G is a Cayley graph of degree d, the time complexity is upper bounded by $\widetilde{O}\left(\sqrt{\frac{nd}{\kappa\lambda}}\mathsf{U}+\sqrt{\frac{nd}{\kappa\lambda_2(G)}}\Lambda\right)$, where Λ is the cost of computing the eigenvalues of G. This gives an upper bound of $\widetilde{O}(n/\sqrt{\lambda\kappa})$ when G is a complete graph, and $\widetilde{O}(\sqrt{n/(\lambda\kappa)})$ when G is a Boolean hypercube.
5. We give an algorithm to estimate the *algebraic connectivity* of $G(x)$, $\lambda_2(G(x))$, when G is a complete graph. The algebraic connectivity is closely related to the inverse of the mixing time, which is known to be small for many interesting families of graphs such as expander graphs. We give a protocol that with probability at least $2/3$ outputs an estimate of $\lambda_2(G(x))$ up to multiplicative error ε in time complexity $\widetilde{O}\left(\frac{1}{\varepsilon}\frac{n}{\sqrt{\lambda_2(G(x))}}\right)$.

1.2 Open Problems

Our work suggests several directions for new research. Since st-connectivity is fairly ubiquitous, it seems that our approach may, in turn, help analyze applications of *st*-connectivity. Additionally, we provide two algorithms for deciding connectivity, items 3 and 4 above. At least naively, it seems like our two algorithms are incomparable, even though they are based on similar unitaries. It would be worthwhile to understand whether the two approaches are fundamentally different. Finally, it would be interesting to see whether one can extend our algorithm for estimating algebraic connectivity to accept more general parent graphs than the complete graph.

2 Preliminaries

2.1 Linear Algebra Notation

For a subspace V of some inner product space, let Π_V denote the orthogonal projector onto V. For a linear operator A, let $\sigma_{\min}(A)$ (respectively $\sigma_{\max}(A)$) denote its smallest (resp. largest) non-zero singular value. Let $\ker A$ denote the kernel of A, $\mathrm{row}(A)$ denote the rowspace of A, and $\mathrm{col}(A)$ the columnspace of A. For a unitary U with eigenvalues $e^{i\theta_1}, \ldots, e^{i\theta_N}$, let $\Delta(U) = \min\{|\theta_i| : \theta_i \neq 0\}$ denote the *phase gap* of U.

2.2 Graph Theory

We will consider multigraphs, so we refer to each edge in the graph using its endpoints and a unique label ℓ, as, for example: $(\{u, v\}, \ell)$. The label ℓ uniquely specifies the edge, but we include the endpoints for convenience. Let $\overrightarrow{E}(G) = \{(u, v, \ell) : (\{u, v\}, \ell) \in E(G)\}$ be the directed edges of G. Furthermore, for any set of edges E, we let $\overrightarrow{E} = \{(u, v, \ell) : (\{u, v\}, \ell) \in E\}$ represent the corresponding set of directed edges. We will sometimes write (u, v, ℓ) for an undirected edge, but when talking about undirected edges, we have $(u, v, \ell) = (v, u, \ell)$.

For $x \in \{0, 1\}^{E(G)}$, we define $G(x)$ as the subgraph of G in which $e \in E(G)$ is included if and only if $x_e = 1$. In general there can be a more complicated association between the edges of G and literals x_i and $\bar{x}_i$, but for simplicity, we don't make this explicit.

A *network* $\mathcal{N} = (G, c)$ consists of a graph G combined with a positive real-valued *weight* function $c : E(G) \longrightarrow \mathbb{R}^+$. Since c is a map on undirected edges, we can easily extend it to map on directed edges such that $c(u, v, \ell) = c(v, u, \ell)$, and we overload our notation accordingly. We will often assume that some c is implicit for a graph G and let $\mathcal{A}_G = \sum_{(u,v,\ell)\in E(G)} c(u, v, \ell)(|u\rangle\langle v| + |v\rangle\langle u|)$ denote its weighted adjacency matrix. Note that $\mathcal{A}_G$ only depends on the total weight of edges from u to v, and is independent of the number of edges across which this weight is distributed. Let $d_G(u) = \sum_{v,\ell:(u,v,\ell)\in E(G)} c(u, v, \ell)$ denote the weighted degree of u in G, under the implicit weight function c, and let $d_{\max}(G) = \max_{u\in V(G)} d_G(u)$. Let $\mathcal{D}_G = \sum_{u\in V(G)} d_G(u)|u\rangle\langle u|$ denote the weighted degree matrix, and let $L_G = \mathcal{D}_G - \mathcal{A}_G$ denote the Laplacian of G. The Laplacian is always positive semidefinite, so its eigenvalues are real and non-negative. For $|\mu\rangle = \sum_{u\in V(G)} |u\rangle$, it is always the case that $L_G|\mu\rangle = 0$, so the smallest eigenvalue of L_G is 0. Let $\lambda_2(G)$ denote the second smallest eigenvalue of L_G, including multiplicity. This value is called the *algebraic connectivity* or the *Fiedler value* of G, and it is non-zero if and only if G is connected.

Consider a graph G with specially labeled vertices s and t that are connected in G. An *st-flow* is any linear combination of *st*-paths. More precisely:

▶ **Definition 1** (Unit *st*-flow and energy)**.** Let G be an undirected 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 $(u, v, \ell) \in \overrightarrow{E}(G)$, $\theta(u, v, \ell) = -\theta(v, u, \ell)$;
2. $\sum_{v,\ell:(s,v,\ell)\in\overrightarrow{E}(G)} \theta(s, v, \ell) = \sum_{v,\ell:(v,t,\ell)\in\overrightarrow{E}(G)} \theta(v, t, \ell) = 1$; and
3. for all $u \in V(G) \setminus \{s, t\}$, $\sum_{v,\ell:(u,v,\ell)\in\overrightarrow{E}(G)} \theta(u, v, \ell) = 0$.

Given an implicit weighting c, the *unit flow energy* of θ on $E' \subseteq E(G(x))$, is $J_{E'}(\theta) = \frac{1}{2}\sum_{e\in\overrightarrow{E'}} \frac{\theta(e)^2}{c(e)}$.

▶ **Definition 2** (Effective resistance and average resistance)**.** Let G be a graph with implicit weighting c and $s, t \in V(G)$. If s and t are connected in $G(x)$, the *effective resistance* of $G(x)$ between s and t is $R_{s,t}(G(x)) = \min_\theta J_{E(G(x))}(\theta)$, where θ runs over all unit st-unit flows of $G(x)$. If s and t are not connected in $G(x)$, $R_{s,t}(G(x)) = \infty$. For a connected graph G, we can define the *average resistance* by $R_{\text{avg}}(G) := \frac{1}{n(n-1)} \sum_{s,t\in V:s\neq t} R_{s,t}(G)$.

Intuitively, $R_{s,t}$ characterizes "how connected" the vertices s and t are in a network. The more, shorter paths connecting s and t, and the more weight on those paths, the smaller the effective resistance. We next introduce a measure of how disconnected s and t are, in the case that we are considering a subgraph $G(x)$ of G where s and t are not connected.

▶ **Definition 3** (Unit *st*-potential)**.** Let G be an undirected weighted graph with $s, t \in V(G)$, and s and t connected. For $G(x)$ such that s and t are not connected, a *unit st-potential* on $G(x)$ is a function $\mathcal{V} : V(G) \to \mathbb{R}^+$ such that $\mathcal{V}(s) = 1$ and $\mathcal{V}(t) = 0$ and $\mathcal{V}(u) = \mathcal{V}(v)$ if $(u, v, \ell) \in E(G(x))$.

A unit st-potential is a witness of the disconnectedness of s and t in $G(x)$, which generalizes the notion of an st-cut. (An st-cut is a unit potential that only takes values 0 and 1.)

▶ **Definition 4** (Unit Potential Energy)**.** Given a graph G with implicit weighting c and a unit st-potential $\mathcal{V}$ on $G(x)$, the *unit potential energy* of $\mathcal{V}$ on $E' \subseteq E(G)$ is defined $\mathcal{J}_{E'}(\mathcal{V}) = \frac{1}{2}\sum_{(u,v,\ell)\in\overrightarrow{E'}} (\mathcal{V}(u) - \mathcal{V}(v))^2 c(u, v, \ell)$.

▶ **Definition 5** (Effective capacitance)**.** Let G be a graph with implicit weighting c and $s, t \in V(G)$. If s and t are not connected in $G(x)$, the *effective capacitance* between s and t of $G(x)$ is $C_{s,t}(G(x)) = \min_{\mathcal{V}} \mathcal{J}_{E(G)}(\mathcal{V})$, where $\mathcal{V}$ runs over all unit st-potentials on $G(x)$. If s and t are connected, $C_{s,t}(G(x)) = \infty$.

In physics, capacitance measures how well a system of two separated conductors stores electric charge. The ratio of the amount of stored charge to the voltage difference between the conductors is a constant that depends only on the geometry of the set-up. This ratio is called the effective capacitance. We discuss this intuition further in Section 2.2 and Appendix A of [8].

Connectivity and *st*-connectivity

We will consider problems parametrized by a parent graph G, by which we more precisely mean a family of graphs $\{G_n\}_{n\in\mathbb{N}}$ where G_n is a graph on n vertices. We will generally drop the subscript n.

A graph is connected if there is a path between every pair of vertices. For a family of graphs G, and $X \subseteq \{0, 1\}^{E(G)}$, $\text{CONN}_{G,X}$ is the *connectivity problem*, defined for all $x \in X$ by $\text{CONN}_{G,X}(x) = 1$ if $G(x)$ is connected, and $\text{CONN}_{G,X}(x) = 0$ if $G(x)$ is not connected.

Similarly, for $s, t \in V(G)$, defined st-$\text{CONN}_{G,X}$ by st-$\text{CONN}_{G,X}(x) = 1$ if there is a path from s to t in $G(x)$, and st-$\text{CONN}_{G,X}(x) = 0$ otherwise, for all $x \in X$.

We will consider CONN and *st*-CONN in the edge-query input model, meaning that we have access to a standard quantum oracle O_x, defined $O_x|i\rangle|b\rangle = |i\rangle|b \oplus x_i\rangle$, where x_i is the i^{th} bit of i. Since every edge of G is associated with an input variable, as described in 2.2, for any edge in G, we can check if it is also present in $G(x)$ using one query to O_x.

2.3 Span Programs and Witness Sizes

Span programs [10] were introduced to quantum algorithms by Reichardt and Špalek [15], and have since proven to be important for designing quantum algorithms in the query model.

▶ **Definition 6** (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 $A(x) := A\Pi_{H(x)}$.

▶ **Definition 7** (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:

$$w_+(x,P) = w_+(x) = \min\{\||w\rangle\|^2 : |w\rangle \in H(x), A|w\rangle = \tau\}, \tag{1}$$

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:

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

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

For a 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 $f^{-1}(1) \subseteq P_1$. Given a span program P that decides f, one can use it to design a quantum algorithm whose output is $f(x)$ (with high probability), given access to the input $x \in X$ via queries of the form $\mathcal{O}_x : |i,b\rangle \mapsto |i, b \oplus x_i\rangle$.

The following theorem is due to [12] (see [7] for a version with similar notation).

▶ **Theorem 8.** *Let $U(P,x) = (2\Pi_{\ker A} - I)(2\Pi_{H(x)} - I)$. 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 by making $O(\sqrt{W_+(f,P)W_-(f,P)})$ calls to $U(P,x)$, and elementary gates. In particular, this algorithm has quantum query complexity $O(\sqrt{W_+(f,P)W_-(f,P)})$.*

Ref. [7] defines the *approximate positive witness size*, $\tilde{w}_+(x,P)$ as the smallest $\||w\rangle\|^2$ such that $A|w\rangle = \tau$ and $\left\|\Pi_{H(x)^\perp}|w\rangle\right\|$, rather than being required to be 0, should be as small as possible. In particular, every x has a finite approximate positive witness size, not only those in P_1.

▶ **Theorem 9** ([7]). *Let $U(P,x) = (2\Pi_{\ker A} - I)(2\Pi_{H(x)} - I)$. Fix $X \subseteq \{0,1\}^N$ and $f : X \to \mathbb{R}_{\geq 0}$. Let P be a span program on $\{0,1\}^N$ such that for all $x \in X$, $f(x) = w_-(x,P)$ and define $\widetilde{W}_+ = \widetilde{W}_+(P) = \max_{x\in X} \tilde{w}_+(x,P)$. Then there is a quantum algorithm that estimates f to accuracy ϵ and that uses $\widetilde{\mathrm{O}}\left(\epsilon^{-3/2}\sqrt{w_-(x)\widetilde{W}_+}\right)$ calls to $U(P,x)$ and elementary gates.*

A span program for *st*-connectivity

An important example of a span program is one for *st*-connectivity, first introduced in [10], and used in [4] to give a new quantum algorithm for *st*-connectivity. Given some implicit weighting function c on G, the span program is as follows, which we denote P_G:

$$\begin{aligned} &\forall e \in E(G), H_{e,1} = \mathrm{span}\{|u,v,\ell\rangle, |v,u,\ell\rangle : e = (\{u,v\},\ell)\} \qquad U = \mathrm{span}\{|v\rangle : v \in V(G)\} \\ &\forall e = (u,v,\ell) \in \overrightarrow{E}(G) : \; A|u,v,\ell\rangle = \sqrt{c(u,v,\ell)}(|u\rangle - |v\rangle) \qquad \tau = |s\rangle - |t\rangle \end{aligned} \tag{3}$$

If s and t are connected in $G(x)$, then a linear combination of weighted *st*-paths in $G(x)$ is a positive witness for x. Furthermore, this is the only possible positive witness form, so x is a positive input for P_G if and only if $G(x)$ is *st*-connected, and in particular, $w_+(x, P_G) = \frac{1}{2}R_{s,t}(G(x))$ [4]. Since the weights $c(e)$ are positive, the set of positive inputs of P_G are independent of the choice of c, however, the witness sizes will depend on c.

3 Effective Capacitance and *st*-connectivity

In [8, Theorem 17], we prove the following theorem, exactly characterizing the negative witness size of the *st*-connectivity span program:

▶ **Theorem 10.** *Let P_G be the span program in Eq. (3). Then for any $x \in \{0,1\}^N$, $w_-(x, P_G) = 2C_{s,t}(G(x))$.*

Previously, the negative witness size of P_G was characterized by the size of a cut [15] or, in planar graphs, the effective resistance of a graph related to the planar dual of $G(x)$ [9].

As a corollary of Theorem 10, we have the following:

▶ **Theorem 11.** *Let G be a multigraph with $s,t \in V(G)$. Then for* any *choice of (non-negative, real-valued) implicit weight function, the bounded error quantum query complexity of evaluating $st\text{-CONN}_{G,X}$ is*

$$O\left(\sqrt{\max_{x\in X : st\text{-CONN}_{G,X}(x)=1} R_{s,t}(G(x)) \times \max_{x\in X : st\text{-CONN}_{G,D}(x)=0} C_{s,t}(G(x))}\right). \tag{4}$$

Proof. This follows from Theorem 10 and the fact that $w_+(x, P_G) = \frac{1}{2}R_{s,t}(G(x))$, proven in [4] and generalized to the weighted case in [9]. Then Theorem 8 gives the result. ◀

We emphasize that Theorem 11 holds for $R_{s,t}$ and $C_{s,t}$ defined with respect to any weight function, some of which may give a significantly better complexity for solving this problem.

3.1 Estimating the Capacitance of a Circuit

By Theorem 10, $w_-(x, P_G) = 2C_{s,t}(G(x))$, so we can apply Theorem 9 to estimate $C_{s,t}(G(x))$. By Theorem 9, the complexity of doing this depends on $C_{s,t}(G(x))$ and $\widetilde{W}_+(P_G) = \max_x \tilde{w}_+(x, P_G)$. We prove the following theorem in [8]:

▶ **Theorem 12.** *For the span program P_G, we have that $\widetilde{W}_+(P_G) = O(\max_p J_{E(G)}(p))$, where the maximum runs over all st-unit flows p that are paths from s to t.*

To prove 12, we first relate unit *st*-flows on G to approximate positive witnesses. Intuitively, an approximate positive witness is an *st*-flow on G that has energy as small as possible on edges in $E(G) \setminus E(G(x))$. Thus, we can upper bound the approximate positive witness size

by the highest possible energy of any st-flow on G, which is always achieved by a flow that is an st-path. Note that when the weights are all 1, $\max_p J_E(G)(p)$ is just the length of the longest self-avoiding st-path in G. Combining Theorems 10, 12 and 9, we have:

▶ **Corollary 13.** *Given a network (G, c), with $s, t \in V(G)$ and access to an oracle O_x, the bounded error quantum query complexity of estimating $C_{s,t}(G(x))$ to accuracy ϵ is $\widetilde{O}(\epsilon^{-3/2}\sqrt{C_{s,t}(G(x))\max_p J_{E(G)}(p)})$ where the maximum runs over all st-unit flows p that are paths from s to t.*

▶ **Corollary 14.** *Let U be the cost of implementing $|u\rangle|0\rangle \mapsto \sum_{(u,v,\ell)\in\overrightarrow{E}(G)} \sqrt{\frac{c(u,v,\ell)}{d_G(u)}}|u,v,\ell\rangle$. Then the quantum time complexity of estimating $C_{s,t}(G(x))$ to accuracy ϵ is*

$$\widetilde{O}\left(\epsilon^{-3/2}\sqrt{C_{s,t}(G(x))\max_p J_{E(G)}(p)}\mathsf{U}\right).$$

Proof. By [9] (generalizing [4]), $U(P_G, x)$, from Theorem 9, can be implemented in cost $O(\mathsf{U})$. ◀

3.2 Deciding Connectivity

Note that $\textsc{conn}_{G,X} = \bigwedge_{\{u,v\}:u,v\in V(G)} uv\text{-}\textsc{conn}_{G,X}$. Thus connectivity is equivalent [11, 9] to $n(n-1)/2$ st-connectivity problems in series, one for each pair of distinct vertices in $V(G)$. (Ref. [1] uses a similar approach, but only looks at $n-1$ instances — the pairs s and v for each $v \in V(G)$. Our approach is symmetrized over the vertices, so the analysis is simpler.)

More precisely, we define a graph $\mathcal{G}$ such that:

$$V(\mathcal{G}) = V(G) \times \{\{u,v\} : u \neq v \in V(G)\}, \quad E(\mathcal{G}) = E(G) \times \{\{u,v\} : u \neq v \in V(G)\} \quad (5)$$

where $\times$ denotes the Cartesian product. We think of the $\{u, v\}$ terms in (5) as an extra label denoting that that edge or vertex is in the $\{u,v\}^{\text{th}}$ copy of the graph G present as a subgraph in $\mathcal{G}$. Choose any labeling of the vertices from 1 to n (with slight abuse of notation, we use u both for the original vertex name and the label). We then label the vertex $(1, \{1, 2\})$ as s and the vertex $(n, \{n-1, n\})$ as t. Next identify vertices $(v, \{u, v\})$ and $(u, \{u, v+1\})$ if $u < v$ and $v < n$, and identify vertices $(v, \{u, v\})$ and $(u+1, \{u+1, u+2\})$ if $v = n$ and $u < n-1$. See [8] for a graphical example of this construction.

Finally, we define $\mathcal{G}(x)$ to be the subgraph of $\mathcal{G}$ with edges $E(\mathcal{G}(x)) = E(G(x)) \times \{\{u,v\} : u \neq v \in V(G)\}$. Clearly, any st-path in $\mathcal{G}(x)$ must go through each of the copies of $G(x)$, meaning it must include, for each $\{u, v\}$, a uv-path through the copy of $G(x)$ labeled $\{u, v\}$. Thus, there is an st-path in $\mathcal{G}(x)$ if and only if $G(x)$ is connected.

We consider the span program $P_{\mathcal{G}}$, where $c(e) = 1$ for all $e \in E(\mathcal{G})$. We will use $P_{\mathcal{G}}$ to solve st-connectivity on $\mathcal{G}(x)$. To analyze the resulting algorithm, we need to upper bound the negative and positive witness sizes $w_-(x, P_{\mathcal{G}}) = 2C_{s,t}(\mathcal{G}(x))$ and $w_+(x, P_{\mathcal{G}}) = \frac{1}{2}R_{s,t}(\mathcal{G}(x))$. Using the rule that resistances in series add, we get:

▶ **Lemma 15.** *For any x such that $G(x)$ is connected, $w_+(x, P_{\mathcal{G}}) = \frac{n(n-1)}{2}R_{\text{avg}}(G(x))$.*

In [8, Lemma 24], we bound $C_{s,t}(\mathcal{G}(x))$, to prove the following:

▶ **Lemma 16.** *Fix $\kappa > 1$, and suppose $G(x)$ has κ connected components. Then if G is a subgraph of a complete graph (that is, G has at most one edge between any pair of vertices), we have $w_-(x, P_{\mathcal{G}}) = O(1/\kappa)$. Otherwise, we have $w_-(x, P_{\mathcal{G}}) = O(d_{\max}(G)/\sqrt{n\kappa})$.*

Combining Lemmas 16 and 15 with Theorem 8, we have the following:

▶ **Theorem 17.** *For any family of graphs G such that G is a subgraph of a complete graph, and $X \subseteq \{0,1\}^{E(G)}$ such that for all $x \in X$, if $G(x)$ is connected, $R_{\text{avg}}(G(x)) \leq R$, and if $G(x)$ is not connected, it has at least κ components, the bounded error quantum query complexity of $\text{CONN}_{G,X}$ is $O\left(n\sqrt{R/\kappa}\right)$.*

For any family of connected graphs G and $X \subseteq \{0,1\}^{E(G)}$ such that for all $x \in X$, if $G(x)$ is connected, $R_{\text{avg}}(G(x)) \leq R$, and if $G(x)$ is not connected, it has at least κ components, the bounded error quantum query complexity of $\text{CONN}_{G,X}$ is $O\left(n^{3/4}\sqrt{Rd_{\max}(G)}/\kappa^{1/4}\right)$.

▶ **Corollary 18.** *Let U be the cost of implementing $|u\rangle|0\rangle \mapsto \sum_{(u,v,\ell)\in\vec{E}(G)} \sqrt{d_G^{-1}(u)}|u,v,\ell\rangle$. If G is subset of a complete graph, the quantum time complexity of $\text{CONN}_{G,X}$ is $O(n\sqrt{R/\kappa}\mathsf{U})$. For any family of connected graphs G and $X \subseteq \{0,1\}^{E(G)}$ such that for all $x \in X$, if $G(x)$ is connected, $R_{\text{avg}}(G(x)) \leq R$, and if $G(x)$ is not connected, it has at least κ components, the quantum time complexity of $\text{CONN}_{G,X}$ is $O\left(n^{3/4}\sqrt{Rd_{\max}(G)}/\kappa^{1/4}\mathsf{U}\right)$.*

Proof. By [9] (generalizing [4]), $U(P_{\mathcal{G}}, x)$ can be implemented in cost $O(\mathsf{U})$. ◀

4 Spectral Algorithm for Deciding Connectivity

In this section, we give alternative quantum algorithms for connectivity. We first present an algorithmic template, outlined in Algorithm 25, that requires the instantiation of a certain initial state. Since this initial state is independent of the input, we already get an upper bound on the quantum query complexity, as follows:

▶ **Corollary 19.** *Fix any $\lambda > 0$ and $\kappa > 1$. For any family of connected graphs G and $X \subseteq \{0,1\}^{E(G)}$ such that for all $x \in X$, either $\lambda_2(G(x)) \geq \lambda$ or $G(x)$ has at least κ connected components, the bounded error quantum query complexity of $\text{CONN}_{G,X}$ is $O\left(\sqrt{\frac{nd_{\max}(G)}{\kappa\lambda}}\right)$.*

In [8, Section 5.1], we describe one such initial state, and how to prepare it, leading to the following upper bound, in which U is the cost of performing one step of a quantum walk on G, and S is the cost of preparing a quantum state corresponding to the stationary distribution of a quantum walk on G:

▶ **Theorem 20.** *Fix any $\kappa > 1$ and $\lambda > 0$. For any family of connected graphs G and $X \subseteq \{0,1\}^{E(G)}$ such that $\forall x \in X$, either $\lambda_2(G(x)) \geq \lambda$, or $G(x)$ has at least κ components, $\text{CONN}_{G,X}$ can be solved in bounded error in time $\widetilde{O}\left(\sqrt{\frac{nd_{\text{avg}}(G)}{\kappa\lambda_2(G)}}\left(\mathsf{S} + \sqrt{\frac{d_{\max}(G)}{\lambda}}\mathsf{U}\right)\right)$.*

In [8, Section 5.2], we restrict our attention to the case where G is a Cayley graph, and give an alternative instantiation of Algorithm 25, proving the following, where Λ is the cost of computing the eigenvalues of G:

▶ **Theorem 21.** *Fix any $\lambda > 0$ and integer $\kappa > 1$. For any family of connected graphs G such that each G is a Cayley graph over an Abelian group and $X \subseteq \{0,1\}^{E(G)}$ such that for all $x \in X$, either $\lambda_2(G(x)) \geq \lambda$ or $G(x)$ has at least κ components, $\text{CONN}_{G,X}$ can be solved in bounded error in time $\widetilde{O}\left(\sqrt{\frac{nd}{\kappa\lambda}}\mathsf{U} + \sqrt{\frac{nd}{\kappa\lambda_2(G)}}\Lambda\right)$.*

The results in this section, in contrast to the previous connectivity algorithm, apply with respect to any weighting of the edges of G. Applying non-zero weights to the edges of G does not change which subgraphs $G(x)$ are connected, but it does impact the complexity of our algorithm. Thus, we get algorithms with the complexities given in Corollary 19 and Theorems 20 and 21, where $d_{\max}(G)$ and $d_{\text{avg}}(G)$ are in terms of weighted degrees, and $\lambda_2(G)$ and $\lambda_2(G(x))$ are in terms of weighted Laplacians.

Finally, in 4.1, we describe how when G is a complete graph, these ideas can be used to design algorithms, not only for deciding connectivity, but also for estimating the *algebraic connectivity* of a graph, a measure of how connected a graph is. In particular, we show:

▶ **Theorem 22.** *Let G be the complete graph on n vertices. There exists a quantum algorithm that, on input x, with probability at least 2/3, outputs an estimate $\tilde{\lambda}$ such that $\left|\tilde{\lambda} - \lambda_2(G(x))\right| \leq \varepsilon\lambda_2(G(x))$, where $\lambda_2(G(x))$ is the algebraic connectivity of $G(x)$, in complexity $\widetilde{O}\left(n/\varepsilon\sqrt{\lambda_2(G(x))}\right)$.*

Let $P_G = (H, U, A, \tau)$ be the span program for st-connectivity defined in 3. Note that only τ depends on s and t, and we will not be interested in τ here. We let $A(x) = A\Pi_{H(x)}$. A simple calculation gives $A(x)A(x)^T = 2L_{G(x)}$ and $AA^T = 2L_G$, where $L_{G(x)}$ and L_G are the Laplacians of $G(x)$ and G respectively. Recall that for any G, the eigenvalues of L_G lie in $[0, d_{\max}]$, with $|\mu\rangle = \frac{1}{\sqrt{n}}\sum_v |v\rangle$ as a 0-eigenvalue. In our case, since G is assumed to be connected, $|\mu\rangle$ is the only 0-eigenvector of L_G, so $\text{row}(L_G)$ is the orthogonal complement of $|\mu\rangle$. For any x, $G(x)$ also has $|\mu\rangle$ as a 0-eigenvalue, but if $G(x)$ is connected, this is the only 0-eigenvalue. In general, the dimension of the 0-eigenspace of $L_{G(x)}$ is the number of components of $G(x)$. Thus, we have the following.

- The multiset of nonzero eigenvalues of L_G are exactly half of the squared singular values of A, and in particular, since no eigenvalue of L_G can be larger than the maximum degree of G, $\sigma_{\max}(A) \leq \sqrt{2d_{\max}(G)}$.
- The multiset of nonzero eigenvalues of $L_{G(x)}$ are exactly half the squared singular values of $A(x)$, and if $G(x)$ is connected, then $\sigma_{\min}(A(x)) = \sqrt{2\lambda_2(G(x))}$, where $\lambda_2(G(x))$ is the second smallest eigenvalue of $L_{G(x)}$, which is non-zero if and only if $G(x)$ is connected.
- The support of L_G is $\text{col}(A)$, which is the orthogonal subspace of $|\mu\rangle = \frac{1}{\sqrt{n}}\sum_v |v\rangle$.

For a particular span program P, and input x, an associated unitary $U(P, x) = (2\Pi_{\ker A} - I)(2\Pi_{H(x)} - I)$ can be used to construct quantum algorithms, for example, for deciding the span program. Then by [7, Theorem 3.10], which states that $\Delta(U(P, x)) \geq 2\sigma_{\min}(A(x))/\sigma_{\max}(A)$, we have the following.

▶ **Lemma 23.** *Let P_G be the st-connectivity span program from 3. Then $\Delta(U(P, x)) \geq 2\sqrt{\lambda_2(G(x))/d_{\max}(G)}$.*

Our algorithm will be based on the following connection between the connectivity of $G(x)$ and the presence of a 0-phase eigenvector of $U(P, x)$ in $\text{row}(A)$, proven in [8, Lemma 32].

▶ **Lemma 24.** *$G(x)$ is not connected if and only if there exists $|\psi\rangle \in \text{row}(A)$ that is fixed by $U(P, x)$. Moreover, if $G(x)$ has $\kappa > 1$ components, there exists a $(\kappa - 1)$-dimensional subspace of $\text{row}(A)$ that is fixed by $U(P, x)$.*

Thus, to determine if $G(x)$ is connected, it is sufficient to detect the presence of *any* 0-phase eigenvector of $U(P, x)$ on $\text{row}(A)$. Let $\{|\psi_i\rangle\}_{i=1}^{n-1}$ be any basis for $\text{row}(A)$, not necessarily orthogonal, and suppose we have access to an operation that generates $|\psi_{\text{init}}\rangle = \sum_{i=1}^{n-1} |i\rangle|\psi_i\rangle$. Such a basis is independent of the input, so we can certainly perform such a map with 0 queries. We will later discuss cases in which we can implement such a map time efficiently.

▶ **Algorithm 25.** *Assume there is a known constant λ such that if $G(x)$ is connected, then $\lambda_2(G(x)) \geq \lambda$. Let $\{|\psi_i\rangle\}_i$ be some states that span the rowspace of A, whose choice determines the cost of the amplitude estimation step.*

1. *Prepare $|\psi_{\text{init}}\rangle = \sum_{i=1}^{n-1} \frac{1}{\sqrt{n-1}}|i\rangle|\psi_i\rangle$.*
2. *Perform the phase estimation procedure of [8, Theorem 9] of $U(P,x)$ on the second register, to precision $\sqrt{\lambda/d_{\max}(G)}$, and accuracy ϵ.*
3. *Use amplitude estimation to determine if the amplitude on $|0\rangle$ in the phase register is 0, in which case, output "connected", or > 0, in which case, output "not connected."*

The algorithm performs phase estimation (see [8, Theorem 8]) on the second register of $|\psi_{\text{init}}\rangle$, with precision $\sqrt{\lambda/d_{\max}(G)}$. Intuitively, this will distinguish any part of the second register that is in the 0-phase space of $U(P,x)$, labeling it with $|0\rangle$ in a new phase register, from any part of the state that is in the span of the θ-phase vectors of $U(P,x)$ for $|\theta| > \sqrt{\lambda/d_{\max}(G)}$. We can then estimate the part of the state in the 0-phase space of $U(P,x)$ by using amplitude estimation in Step 3. First, suppose that there are $\kappa - 1 > 0$ orthonormal 0-phase eigenvectors of $U(P,x)$ in $\text{row}(A)$, and let Π be the orthonormal projector onto their span. By [8, Theorem 8], for each i, the phase estimation step will map $|i\rangle\,(\Pi|\psi_i\rangle)$ to $|i\rangle|0\rangle\,(\Pi|\psi_i\rangle)$. Thus, the squared amplitude on $|0\rangle$ in the phase register will be at least:

$$\varepsilon := \frac{\|(I\otimes\Pi)|\psi_{\text{init}}\rangle\|^2}{\||\psi_{\text{init}}\rangle\|^2} = \frac{1}{\||\psi_{\text{init}}\rangle\|^2}\sum_{i=1}^{n-1}\|\Pi|\psi_i\rangle\|^2 > 0, \tag{6}$$

since the $|\psi_i\rangle$ span $\text{row}(A)$.

On the other hand, suppose $G(x)$ is connected, so there is no 0-phase eigenvector in $\text{row}(A)$. Then all phases will be at least $\Delta(U(P,x)) \geq \sqrt{\lambda/d_{\max}(G)}$, so the phase register will have squared overlap at most ϵ with $|0\rangle$. Setting $\epsilon = \varepsilon/2$, we just need to distinguish between an amplitude of $\geq \varepsilon$ and an amplitude of $\leq \varepsilon/2$ on $|0\rangle$, which we can do using amplitude estimation in $\frac{1}{\sqrt{\varepsilon}}$ calls to Steps 1 and 2 (See [8] for details). Step 2 can be implemented using $\sqrt{d_{\max}(G)/\lambda}\log\frac{1}{\varepsilon}$ calls to $U(P,x)$. By [9, Theorem 13], if U is the cost of implementing, for any $u \in V$, the map

$$|u,0\rangle \mapsto \sum_{(v,\ell)\in\Gamma(u)} \sqrt{c(u,v,\ell)/d_G(u)}|u,v,\ell\rangle, \tag{7}$$

which corresponds to one step of a quantum walk on G, then $U(P,x)$ can be implemented in time $O(\mathsf{U})$. We thus get the following (formally proven in [8, Theorem 34]):

▶ **Theorem 26.** *Fix $\lambda > 0$. Let Init denote the cost of generating the initial state $|\psi_{\text{init}}\rangle$, and U the cost of the quantum walk step in equation 7. Let ε be as in equation 6. Then for any family of connected graphs G and $X \subseteq \{0,1\}^{E(G)}$ such that for all $x \in X$, either $\lambda_2(G(x)) \geq \lambda$ or $G(x)$ is not connected, $\text{CONN}_{G,X}$ can be decided by a quantum algorithm with cost $O\left(\frac{1}{\sqrt{\varepsilon}}\left(\mathsf{Init} + \sqrt{\frac{d_{\max}(G)}{\lambda}}\mathsf{U}\log\frac{1}{\varepsilon}\right)\right)$.*

In [8, Sections 5.1 and 5.2], we discuss particular implementations of this algorithm, but if we only care about query complexity, we already have Corollary 19. We formally prove Corollary 19 in [8], but the idea is that Init costs 0 queries, $U(P,x)$ can be implemented in 2 queries, and if there are $\kappa - 1$ orthonormal 0-phase vectors of $U(P,x)$ in $\text{row}(A)$, they each contribute at least $\frac{1}{n-1}$ to $\epsilon = \|(I\otimes\Pi)|\psi_{\text{init}}\rangle\|^2$, so $\epsilon \geq \frac{\kappa-1}{n-1}$, giving a total query complexity of $O(\sqrt{nd_{\max}(G)(\kappa\lambda)^{-1}})$.

4.1 Estimating the connectivity when G is a complete graph

For the remainder of this section, let G be the complete graph on n vertices, K_n. In that case, we can not only decide if $G(x)$ is connected, but estimate $\lambda_2(G(x))$. The idea is to relate the smallest phase of $U(P,x)$ on $\mathrm{row}(A)$ to $\lambda_2(G(x))$, and estimate this value using quantum phase estimation. We use a correspondence between the phases of the product of two reflections $U = (2\Pi_A - I)(2\Pi_B - 1)$ and the singular values of its discriminant, defined $\Pi_A\Pi_B$ due to Szegedy [16] to prove the following in [8, Section 5.3].

▶ **Lemma 27.** *Let $U = (2\Pi_A - I)(2\Pi_B - I)$ and $D = \Pi_A\Pi_B$ be its discriminant. Then $\Delta(-U) = 2\sin^{-1}(\sigma_{\min}(D))$. Moreover, when G is a complete graph on n vertices, we have for any x, $\lambda_2(G(x)) = n\sin^2(\Delta(U(P,x))/2)$.*

This correspondence also implies the following, which allows us to restrict our attention to $\mathrm{row}(A)$ in searching for the smallest phase of $U(P,x)$ and is proven in [8, Section 5.3]:

▶ **Lemma 28.** *Let $U = (2\Pi_A - I)(2\Pi_B - I)$, and let $|\Delta_+\rangle$ be a $\Delta(U)$-eigenvector of U, and $|\Delta_-\rangle$ a $(-\Delta(U))$-eigenvector of U. Then there exists a vector $|u\rangle$ in the support of A such that $|u\rangle \in \mathrm{span}\{|\Delta_+\rangle, |\Delta_-\rangle\}$. In particular, if $|\Delta_\pm\rangle$ are $\pm\Delta(U(P,x))$-eigenvectors of $U(P,x)$, then there exists a vector $|u\rangle$ in $\mathrm{row}(A)$ such that $|u\rangle \in \mathrm{span}\{|\Delta_+\rangle, |\Delta_-\rangle\}$.*

We will estimate the value $\tau = \Delta(U(P,x))/\pi$ in the range $[0,1]$, which we will then transform into an estimate of $\lambda_2(G(x))$. At every iteration, c will denote a lower bound for τ and C will denote the current upper bound. At the beginning of the algorithm we have $c = 0$, $C = 1$, and every iteration will result in updating either C or c in such a manner that the new interval for τ is reduce by a fraction of $2/3$. The algorithm is described in Algorithm 29.

▶ **Algorithm 29.** *To begin, let $c = 0$ and $C = 1$.*

1. *Set $\varphi = \frac{C-c}{3}$, $\epsilon = \frac{1}{\sqrt{2n}}$, $\delta = c + \varphi$.*
2. *For $j = 1, \ldots, 4\log(n/\varepsilon)$:*
   a. *Prepare $\sum_{i=1}^{n-1} \frac{1}{\sqrt{n-1}}|i\rangle|\psi_i\rangle|0\rangle_B|0\rangle_P$.*
   b. *Perform the gapped phase estimation algorithm $GPE(\varphi,\epsilon,\delta)$ of [8, Theorem 9] to $U(P,x)$ on the second register.*
   c. *Use amplitude estimation to distinguish between the case when the amplitude on $|0\rangle_B$ is $\geq \frac{1}{\sqrt{n}}$, in which case output "$a_j = 0$", and the case where the ampltiude is $\leq \frac{1}{\sqrt{2n}}$, in which case, output "$a_j = 1$".*
3. *Compute $\tilde{a} = Maj(a_1, \ldots, a_{4\log(n/\varepsilon)})$. If the result is 0, set $C = \delta + \varphi$. If the result is 1, set $c = \delta$. If $C - c \leq 2\varepsilon c$, then output $n\sin^2\left(\frac{\pi(C+c)}{4}\right)$. Otherwise, return to Step 1.*

We say an iteration of the algorithm *succeeds* if $\tilde{a} = Maj(a_1, \ldots, a_{4\log(n/\varepsilon)})$ correctly indicates whether the amplitude on $|0\rangle_B$ is $\geq n^{-1/2}$ or $\leq (2n)^{-1/2}$. This happens with probability $\Omega(1-(\varepsilon/n)^4)$. Since we will shortly see that the algorithm runs for at most $\widetilde{\mathrm{O}}(n/\varepsilon\sqrt{\lambda_2(G(x))}) \leq \widetilde{\mathrm{O}}\left(n^2\varepsilon^{-1}\right)$ steps, a Taylor series approximation guarantees that the probability that every iteration succeeds is at least $\Omega\left(1-(\varepsilon/n)^2\right)$. It is therefore reasonable to assume that every iteration succeeds, since this happens with high probability. We first note that if every iteration succeeds, throughout the algorithm we have $\tau = \Delta(U(P,x))/\pi \in [c,C]$.

▶ **Lemma 30.** *Let $\tau = \Delta(U(P,x))/\pi$. For any φ and δ, if $\tau \geq \delta + \varphi$, applying $GPE(\varphi,\epsilon,\delta)$ to $|\psi_{\mathrm{init}}\rangle$ results in a state with amplitude at most $\frac{1}{\sqrt{2n}}$ on $|0\rangle_B$ in register B; and if $\tau \leq \delta$, this results in a state with amplitude at least $\frac{1}{\sqrt{n}}$ on $|0\rangle_B$ in register B. Thus, if every iteration succeeds, at every iteration, we have $\tau \in [c,C]$.*

The proof of Lemma 30 is found in [8, Section 5.3]. Next, we analyze the running time of Algorithm 29 to get the following theorem, also proven in [8, Section 5.3].

▶ **Theorem 31.** *With probability* $\Omega(1-(\varepsilon/n)^2)$*, Algorithm 29 will terminate after time* $\widetilde{O}(n/\varepsilon\sqrt{\lambda_2(G(x))})$.

Finally, we prove in [8, Section 5.3] that the algorithm outputs an estimate that is within ε multiplicative error of $\lambda_2(G(x))$. Theorem 22 follows from Theorems 31 and 32.

▶ **Theorem 32.** *With probability at least* $\Omega(1-(\varepsilon/n)^2)$*, Algorithm 29 outputs an estimate* $\tilde{\lambda}$ *such that* $\left|\lambda_2(G(x))-\tilde{\lambda}\right| \leq \frac{\pi^2 3}{4}\varepsilon\lambda_2(G(x))$.

References

1. A. Āriņš. *Span-Program-Based Quantum Algorithms for Graph Bipartiteness and Connectivity*, pages 35–41. Springer International Publishing, Cham, 2016. doi:10.1007/978-3-319-29817-7_4.
2. A. Belovs. Learning-graph-based quantum algorithm for k-distinctness. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 207–216, 2012. doi:10.1109/FOCS.2012.18.
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.
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.
5. C. Cade, A. Montanaro, and A. Belovs. Time and space efficient quantum algorithms for detecting cycles and testing bipartiteness, 2016. arXiv:1610.00581.
6. 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.
7. 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. arXiv:1507.00432.
8. Michael Jarret, Stacey Jeffery, Shelby Kimmel, and Alvaro Piedrafita. Quantum algorithms for connectivity and related problems. *arXiv preprint arXiv:1804.10591*, 2018.
9. S. Jeffery and S. Kimmel. Quantum algorithms for graph connectivity and formula evaluation. *Quantum*, 1:26, 2017. doi:10.22331/q-2017-08-17-26.
10. M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the 8th Annual IEEE Conference on Structure in Complexity Theory*, pages 102–111, 1993.
11. N. Nisan and A. Ta-Shma. Symmetric logspace is closed under complement. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing (STOC 1995)*, pages 140–146, New York, NY, USA, 1995. ACM. doi:10.1145/225058.225101.
12. B. W. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pages 544–551, 2009. arXiv:quant-ph/0904.2759.
13. B. W. Reichardt. Span programs and quantum query algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:110, 2010.
14. 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.
15. B. W. Reichardt and R. Špalek. Span-program-based quantum algorithm for evaluating formulas. *Theory of Computing*, 8(13):291–319, 2012.
16. M. Szegedy. Quantum speed-up of Markov chain based algorithms. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004)*, pages 32–41, Washington, DC, USA, 2004. IEEE Computer Society. doi:10.1109/FOCS.2004.53.