Quantum algorithm for tree size estimation, with applications to backtracking and 2-player games

Andris Ambainis[†] Martins Kokainis[†]

Abstract

We study quantum algorithms on search trees of unknown structure, in a model where the tree can be discovered by local exploration. That is, we are given the root of the tree and access to a black box which, given a vertex v, outputs the children of v.

We construct a quantum algorithm which, given such access to a search tree of depth at most n, estimates the size of the tree T within a factor of $1 \pm \delta$ in $\tilde{O}(\sqrt{nT})$ steps. More generally, the same algorithm can be used to estimate size of directed acyclic graphs (DAGs) in a similar model.

We then show two applications of this result:

- We show how to transform a classical backtracking search algorithm which examines T nodes of a search tree into an $\tilde{O}(\sqrt{T}n^{3/2})$ time quantum algorithm, improving over an earlier quantum backtracking algorithm of Montanaro [12].
- We give a quantum algorithm for evaluating AND-OR formulas in a model where the formula can be discovered by local exploration (modeling position trees in 2-player games). We show that, in this setting, formulas of size T and depth $T^{o(1)}$ can be evaluated in quantum time $O(T^{1/2+o(1)})$. Thus, the quantum speedup is essentially the same as in the case when the formula is known in advance.

[†]Faculty of Computing, University of Latvia

1 Introduction

Many search algorithms involve exploring search trees of an unknown structure. For example, backtracking algorithms perform a depth-first search on a tree consisting of partial solutions to the computational task (for example, partial assignments for SAT in the well known DPLL algorithm [3, 4]), until a full solution is found. Typically, different branches of the tree stop at different depths (e.g., when the corresponding partial assignment can no longer be extended) and the structure of the tree can be only determined by exploring it.

Quantum algorithms provide a quadratic speedup for many search problems, from simple exhaustive search (Grover's algorithm [6]) to computing AND-OR formulas [1, 5, 14] (which corresponds to determining the winner in a 2-player game, given a position tree). These algorithms, however, assume that the structure of the search space is known. Grover's algorithm assumes that the possible solutions in the search space can be indexed by numbers $1, 2, \ldots, T$ so that, given i, one can efficiently (in constant or polylog time) find the i^{th} possible solution. In the case of backtracking trees, the unknown structure of the tree prevents us from setting up such an addressing scheme.

In the case of AND-OR formulas, the quantum algorithms of [1, 14] work for formula of any structure but the coefficients in algorithm's transformations depend on the sizes of different subtrees of the formula tree. Therefore, the algorithm can be only used if the whole AND-OR formula is known in advance (and only the values of the variables are unknown) which is not the case if the formula corresponds to a position tree in a game.

Despite the importance of such algorithms classically, there has been little work on quantum search on structures which can be only explored locally. The main algorithmic result of this type is a recent algorithm by Montanaro for quantum backtracking. Given a search tree of size T and depth n, Montanaro's algorithm detects if the tree contains a marked vertex in $O(\sqrt{Tn})$ steps (and finds a marked vertex in $O(\sqrt{T}n^{3/2})$ steps).

In this paper, we show three new quantum algorithms for trees of an unknown structure, including an improvement to Montanaro's algorithm. We start with

Quantum tree size estimation. We show that, given a tree $\mathcal{T}$ with a depth at most n, the size T of the tree $\mathcal{T}$ can be estimated to a multiplicative factor of $1 + \delta$, for an arbitrary constant $\delta > 0$, by a quantum algorithm that uses $\tilde{O}(\sqrt{Tn})$ steps[1]. More generally, our algorithm is also applicable to estimating size of directed acyclic graphs (DAGs) in a similar model.

We then apply the quantum tree size estimation algorithm to obtain two more results.

Improved quantum algorithm for backtracking. Montanaro's algorithm has the following drawback. Since classical search algorithms are optimized to search the most promising branches first, a classical search algorithm may find a marked vertex after examining $T' \ll T$ nodes of the tree. Since the running time of Montanaro's algorithm depends on T, the quantum speedup that it achieves can be much less than quadratic (or there might be no speedup at all).

We fix this problem by using our tree size estimation algorithm. Namely, we construct a quantum algorithm that searches a backtracking tree in $\tilde{O}(\sqrt{T'}n^{3/2})$ steps where T' is the number of nodes actually visited by the classical algorithm.

AND-OR formulas of unknown structure. We construct a quantum algorithm for comput-

[1]In the statements of results in the abstract and the introduction, $\tilde{O}$ hides $\log T$ and $\log n$ factors and the dependence of the running time on the maximum degree d of vertices in a tree $\mathcal{T}$ (or a DAG $\mathcal{G}$). More precise bounds are given in Section 3.

ing AND-OR formulas in a model where the formula is accessible by local exploration, starting from the root (which is given). More specifically, we assume query access to the following subroutines:

- given a node v, we can obtain the type of the node (AND, OR or a leaf),
- given a leaf v, we can obtain the value of the variable (true or false) at this leaf,
- given an AND/OR node, we can obtain pointers to the inputs of the AND/OR gate.

This models a position tree in a 2-player game (often mentioned as a motivating example for studying AND-OR trees) with OR gates corresponding to positions at which the 1^{st} player makes a move and AND gates corresponding to positions at which the 2^{nd} player makes a move.

We give an algorithm that evaluates AND-OR formulas of size T and depth $T^{o(1)}$ in this model with $O(T^{1/2+o(1)})$ queries. Thus, the quantum speedup is almost the same as in the case when the formula is known in advance (and only values at the leaves need to be queried) [1, 14], as long as the depth of the tree is not too large.

2 Preliminaries

2.1 Setting

We consider a tree $\mathcal{T}$ of an unknown structure given to us in the following way:

- We are given the root r of $\mathcal{T}$.
- We are given a black box which, given a vertex v, returns the number of children $d(v)$ for this vertex.
- We are given a black box which, given a vertex v and $i \in [d(v)]$, returns the i^{th} child of v.

Trees of unknown structure come up in several different settings.

Backtracking. Let $\mathcal{A}$ be a backtracking algorithm that searches a solution space $\mathcal{D}$ in a depth-first fashion. The space $\mathcal{D}$ consists of partial solutions where some of the relevant variables have been set. (For example, $\mathcal{D}$ can be the space of all partial assignments for a SAT formula.) Then, the corresponding tree $\mathcal{T}$ is defined as follows:

- vertices v_x correspond to partial solutions $x \in \mathcal{D}$;
- the root r corresponds to the empty solution where no variables have been set;
- children of a vertex v_x are the vertices v_y corresponding to possible extensions y of the partial solution x that $\mathcal{A}$ might try (for example, a backtracking algorithm for SAT might choose one variable and try all possible values for this variable), in the order in which $\mathcal{A}$ would try them.

Two-player games. $\mathcal{T}$ may also be a position tree in a 2-player game, with r corresponding to the current position. Then, children of a node v are the positions to which one could go by making a move at the position v. A vertex v is a leaf if we stop the evaluation at v and do not evaluate children of v.

DAGs of unknown structure. We also consider a generalization of this scenario to directed acyclic graphs (DAGs). Let $\mathcal{G}$ be a directed acyclic graph. We assume that

- Every vertex v is reachable from the root r via a directed path.
- The vertices of $\mathcal{G}$ can be divided into layers so that all edges from layer i go to layer $i+1$.
- Given a vertex v, we can obtain the number $d(v)$ of edges (u,v) and the number $d'(v)$ of edges (v,u).
- Given a vertex v and a number $i \in [d(v)]$, we can obtain the i^{th} vertex u with an edge (u,v).
- Given a vertex v and a number $i \in [d'(v)]$, we can obtain the i^{th} vertex u with an edge (v,u).

2.2 Notation

By $[a\,..\,b]$, with a, b being integers, $a \leq b$, we denote the set $\{a, a+1, a+2, \ldots, b\}$. When $a = 1$, notation $[a\,..\,b]$ is simplified to $[b]$.

We shall use the following notation for particular matrices:

- $\mathbf{I}_k$: the $k \times k$ identity matrix;
- $\mathbf{0}_{k_1,k_2}$: the $k_1 \times k_2$ all-zeros matrix.

We use the following notation for parameters describing a tree $\mathcal{T}$ or a DAG $\mathcal{G}$:

- T denotes the number of edges in $\mathcal{T}$ (or $\mathcal{G}$) or an upper bound on the number of edges which is given to an algorithm.
- n denotes the depth of $\mathcal{T}$ (or $\mathcal{G}$) or an upper bound on the depth which is given to an algorithm.
- d denotes the maximum possible total degree of a vertex $v \in \mathcal{T}$ ($\mathcal{G}$).
- For any vertex $x \in \mathcal{T}$ (where $\mathcal{T}$ is a tree), the subtree rooted at x will be denoted by $\mathcal{T}(x)$.

2.3 Eigenvalue estimation

Quantum eigenvalue estimation is an algorithm which, given a quantum circuit implementing a unitary U and an eigenstate $|\psi\rangle$ s.t. $U|\psi\rangle = e^{i\theta}|\psi\rangle$, produces an estimate for θ. It is known that one can produce an estimate $\hat{\theta}$ such that $Pr[|\theta - \hat{\theta}| \leq \delta_{est}] \geq 1 - \epsilon_{est}$ with $O(\frac{1}{\delta_{est}} \log \frac{1}{\epsilon_{est}})$ repetitions of a circuit for controlled-U.

If eigenvalue estimation is applied to a quantum state $|\psi\rangle$ that is a superposition of several eigenstates

$$|\psi\rangle = \sum_j \alpha_j |\psi_j\rangle, \quad U|\psi_j\rangle = e^{i\theta_j}|\psi_j\rangle,$$

the result is as if we are randomly choosing j with probability $|\alpha_j|^2$ and estimating θ_j.

In this paper, we use eigenvalue estimation to estimate the eigenvalue $e^{i\theta_{min}}$ that is closest to 1 (by that, here and later, we mean the eigenvalue which is closest to 1 among all eigenvalues that are distinct from 1, i.e., the eigenvalue $e^{i\theta_{min}}$ with the smallest nonzero absolute value $|\theta_{min}|$). We assume that we can produce a state $|\psi_{start}\rangle$ such that $|\psi_{start}\rangle$ is orthogonal to all 1-eigenvectors of U and

$$|\langle\Psi_+|\psi_{start}\rangle|^2 + |\langle\Psi_-|\psi_{start}\rangle|^2 \geq C$$

where $|\Psi_+\rangle$ and $|\Psi_-\rangle$ are eigenstates with eigenvalues $e^{i\theta_{min}}$ and $e^{-i\theta_{min}}$ and C is a known constant. (If U does not have an eigenvector with an eigenvalue $e^{-i\theta_{min}}$, the condition should be replaced by $|\langle\Psi_+|\psi_{start}\rangle|^2 \geq C$.) We claim

Lemma 1. *Under the conditions above, there is an algorithm which produces an estimate $\hat{\theta}$ such that $Pr[|\theta_{min} - \hat{\theta}| \leq \delta_{min}] \geq 1 - \epsilon_{min}$ with*

$$O\left(\frac{1}{C}\frac{1}{\delta_{min}}\log\frac{1}{C}\log^2\frac{1}{\epsilon_{min}}\right)$$

repetitions of a circuit for controlled-U.

Proof. In Section 4. □

3 Results and algorithms

3.1 Results on estimating sizes of trees and DAGs

In this subsection, we consider the following task:

Tree size estimation. The input data consist of a tree $\mathcal{T}$ and a value T_0 which is supposed to be an upper bound on the number of vertices in the tree. The algorithm must output an estimate for the size of the tree. The estimate can be either a number $\hat{T} \in [T_0]$ or a claim "$\mathcal{T}$ contains more than T_0 vertices". We say that the estimate is δ-correct if:

1. the estimate is $\hat{T} \in [T_0]$ and it satisfies $|T - \hat{T}| \leq \delta T$ where T is the actual number of vertices;
2. the estimate is "$\mathcal{T}$ contains more than T_0 vertices" and the actual number of vertices T satisfies $(1+\delta)T > T_0$.

We say that an algorithm solves the tree size estimation problem up to precision $1 \pm \delta$ with correctness probability at least $1 - \epsilon$ if, for any $\mathcal{T}$ and any T_0, the probability that it outputs a δ-correct estimate is at least $1 - \epsilon$.

More generally, we can consider a similar task for DAGs.

DAG size estimation. The input data consist of a directed acyclic graph $\mathcal{G}$ and a value T_0 which is supposed to be an upper bound on the number of edges in $\mathcal{G}$. The algorithm must output an estimate for the number of edges. The estimate can be either a number $\hat{T} \in [T_0]$ or a claim "$\mathcal{G}$ contains more than T_0 edges". We say that the estimate is δ-correct if:

1. the estimate is $\hat{T} \in [T_0]$ and it satisfies $|T - \hat{T}| \leq \delta T$ where T is the actual number of edges;
2. the estimate is "$\mathcal{G}$ contains more than T_0 edges" and the actual number of edges T satisfies $(1+\delta)T > T_0$.

We say that an algorithm solves the DAG size estimation problem up to precision $1 \pm \delta$ with correctness probability at least $1 - \epsilon$ if, for any $\mathcal{G}$ and any T_0, the probability that it outputs a δ-correct estimate is at least $1 - \epsilon$.

Tree size estimation is a particular case of this problem: since a tree with T edges has $T + 1$ vertices, estimating the number of vertices and the number of edges are essentially equivalent for trees. We show

Theorem 2. *DAG size estimation up to precision $1\pm\delta$ can be solved with the correctness probability at least $1-\epsilon$ by a quantum algorithm which makes*

$$O\left(\frac{\sqrt{nT_0}}{\delta^{1.5}} d \log^2 \frac{1}{\epsilon}\right)$$

queries to black boxes specifying $\mathcal{G}$ and $O(\log T_0)$ non-query transformations per query.

Note. If we use $\tilde{O}$-notation, the $O(\log T_0)$ factor can be subsumed into the $\tilde{O}$ and the time complexity is similar to query complexity.

3.2 Algorithm for DAG size estimation

In this subsection, we describe the algorithm of Theorem 2. The basic framework of the algorithm (the state space and the transformations that we use) is adapted from Montanaro [12].

Let $\mathcal{G} = (\mathsf{V}, \mathsf{E})$ be a directed acyclic graph, with $|\mathsf{V}| = V$ vertices and $|\mathsf{E}| = T$ edges. We assume that the root is labeled as v_1.

For each vertex $v_i \in \mathsf{V}$

- $\ell(i) \leq n$ stands for the distance from v_i to the root,
- $d_i \leq d$ stands for the total degree of the vertex v_i. In notation in Section 2.1, we have $d_i = d(v_i) + d'(v_i)$.

For technical purposes we also introduce an additional vertex v_{V+1} and an additional edge $e_{T+1} = (v_{V+1}, v_1)$ which connects v_{V+1} to the root. Let $\mathsf{V}' = \mathsf{V} \cup \{v_{V+1}\}$, $\mathsf{E}' = \mathsf{E} \cup \{e_{T+1}\}$ and $\mathcal{G}' = (\mathsf{V}', \mathsf{E}')$.

For each vertex v by $\mathsf{E}(v)$ we denote the set of all edges in E incident to v (in particular, when $v = v_1$ is the root, the additional edge $e_{T+1} \notin \mathsf{E}$ is not included in $\mathsf{E}(v_1)$).

Let $\mathsf{V_A}$ be the set of vertices at an even distance from the root (including the root itself) and $\mathsf{V_B}$ be the set of vertices at an odd distance from the root. Let $A = |\mathsf{V_A}|$ and $B = |\mathsf{V_B}|$, then $V = A+B$. Label the vertices in V so that $\mathsf{V_A} = \{v_1, v_2, \ldots, v_A\}$ and $\mathsf{V_B} = \{v_{A+1}, v_{A+2}, \ldots, v_{A+B}\}$.

Let $\alpha > 0$ be fixed. Define a Hilbert space $\mathcal{H}$ spanned by $\{|e\rangle \mid e \in \mathsf{E}'\}$ (one basis state per edge, including the additional edge). For each vertex $v \in \mathsf{V}$ define a vector $|s_v\rangle \in \mathcal{H}$ as

$$|s_v\rangle = \begin{cases} |e_{T+1}\rangle + \alpha \sum_{e \in \mathsf{E}(v)} |e\rangle\,, & v = v_1 \\ \sum_{e \in \mathsf{E}(v)} |e\rangle\,, & v \neq v_1. \end{cases}$$

Define a subspace $\mathcal{H}_\mathsf{A} \subset \mathcal{H}$ spanned by $\{|s_v\rangle \mid v \in \mathsf{V_A}\}$ and a subspace $\mathcal{H}_\mathsf{B} \subset \mathcal{H}$ spanned by $\{|s_v\rangle \mid v \in \mathsf{V_B}\}$. Define a unitary operator R_A which negates all vectors in $\mathcal{H}_\mathsf{A}$ (i.e., maps $|\psi\rangle$ to $-|\psi\rangle$ for all $|\psi\rangle \in \mathcal{H}_\mathsf{A}$) and leaves $\mathcal{H}_\mathsf{A}^\perp$ invariant. Analogously, a unitary operator R_B negates all vectors in $\mathcal{H}_\mathsf{B}$ and leaves $\mathcal{H}_\mathsf{B}^\perp$ invariant.

Similarly as in [12], both R_A and R_B are implemented as the direct sum of diffusion operators D_v. Let a subspace $\mathcal{H}_v$, $v \in \mathsf{V}$, be spanned by $\{|e\rangle \mid e \in \mathsf{E}(v)\}$ (or $\{|e\rangle \mid e \in \mathsf{E}(v)\} \cup \{|e_{T+1}\rangle\}$ when $v = v_1$ is the root). Define the diffusion operator D_v, which acts on the subspace $\mathcal{H}_v$, as $I - \frac{2}{\||s_v\rangle\|^2} |s_v\rangle \langle s_v|$. This way, each D_v can be implemented with only knowledge of v and its neighborhood. (A minor difference from [12]: since we are concerned with tree size estimation problem now, we make no assumptions about any vertices being marked at this point and therefore D_v is not the identity for any $v \in \mathsf{V}$.)

Then,

$$R_{\mathsf{A}} = \bigoplus_{v \in \mathsf{V_A}} D_v \quad \text{and} \quad R_{\mathsf{B}} = |e_{T+1}\rangle \langle e_{T+1}| + \bigoplus_{v \in \mathsf{V_B}} D_v.$$

In Section 5, we show

Lemma 3. *Transformations R_{A} and R_{B} can be implemented using $O(d)$ queries and $O(d \log V)$ non-query gates.*

We note that R_{A} and R_{B} are defined with respect to a parameter α, to be specified in the algorithm that uses the transformations R_{A} and R_{B}.

The algorithm of Theorem 2 for estimating size of DAGs is as follows:

Algorithm 1 Algorithm for DAG size estimation

1. Apply the algorithm of Lemma 1 for the transformation $R_{\mathsf{B}} R_{\mathsf{A}}$ (with $\alpha = \sqrt{2n\delta^{-1}}$) with the state $|\psi_{start}\rangle = |e_{T+1}\rangle$ and parameters $C = \frac{4}{9}$, $\epsilon_{min} = \epsilon$, $\delta_{min} = \frac{\delta^{1.5}}{24\sqrt{3nT_0}}$
2. Output $\hat{T} = \frac{1}{\alpha^2 \sin^2 \frac{\hat{\theta}}{2}}$ as the estimate for the number of edges.

3.3 Analysis of Algorithm 1

We now sketch the main ideas of the analysis of Algorithm 1. From Lemma 13 in Section 6.2 it follows that $R_{\mathsf{B}} R_{\mathsf{A}}$ has no 1-eigenvector $|\psi\rangle$ with $\langle \psi | e_{T+1} \rangle \neq 0$. Let $|\Psi_+\rangle$ and $|\Psi_-\rangle$ be the two eigenvectors of $R_{\mathsf{B}} R_{\mathsf{A}}$ with eigenvalues $e^{\pm i\theta}$ closest to 1. Lemma 4 shows that the starting state $|e_{T+1}\rangle$ has sufficient overlap with the subspace spanned by these two vectors for applying the algorithm of Lemma 1.

Lemma 4. *If $\alpha \geq \sqrt{2n}$, we have*

$$\langle e_{T+1} | q_2 \rangle \geq \frac{2}{3}$$

for a state $|q_2\rangle \in \operatorname{span}\{|\Psi_+\rangle, |\Psi_-\rangle\}$.

Proof. In Section 6.4. □

Lemma 5 shows that the estimate $\hat{\theta}$ provides a good estimate $\hat{T}$ for the size of the DAG.

Lemma 5. *Suppose that $\delta \in (0,1)$. Let $\alpha = \sqrt{2n\delta^{-1}}$ and $\hat{\theta} \in (0; \pi/2)$ satisfy*

$$\left| \hat{\theta} - \theta \right| \leq \frac{\delta^{1.5}}{24\sqrt{3nT}}.$$

Then

$$(1-\delta)T \leq \frac{1}{\alpha^2 \sin^2 \frac{\hat{\theta}}{2}} \leq (1+\delta)T.$$

Proof. In Section 6.4. □

Lemmas 1, 4 and 5 together imply that Algorithm 1 outputs a good estimate with probability at least $1-\epsilon$. According to Lemma 1, we need to invoke controlled versions of R_{A} and R_{B}

$$O\left(\frac{1}{C}\frac{1}{\delta_{min}}\log\frac{1}{C}\log^2\frac{1}{\epsilon_{min}}\right)=O\left(\frac{\sqrt{nT_0}}{\delta^{1.5}}\log^2\frac{1}{\epsilon}\right)$$

times and because of Lemma 3, each of these transformations can be performed with $O(d)$ queries and $O(d\log V)=O(d\log T_0)$ non-query transformations.

Proof of Lemmas 4 and 5 consists of a number of steps:

1. We first relate the eigenvalues of $R_{\mathsf{A}}R_{\mathsf{B}}$ to singular values of an $A\times B$ matrix L. The matrix L is defined by $L[v,w]=\frac{\langle s_v|s_w\rangle}{\|s_v\|\cdot\|s_w\|}$. Because of a correspondence by Szegedy [15], a pair of eigenvalues $e^{\pm i\theta}$ of $R_{\mathsf{B}}R_{\mathsf{A}}$ corresponds to a singular value $\lambda=\cos\frac{\theta}{2}$ of L.

2. Instead of L, we consider $K=(I-LL^*)^{-1}$ (with both rows and columns indexed by elements of V_{A}). A singular value λ of L corresponds to an eigenvalue $(1-\lambda^2)^{-1}$ of K.

3. We relate K to the fundamental matrix N of a certain classical random walk on the graph $\mathcal{G}$. (The entries of the fundamental matrix $N[i,j]$ are the expected number of visits to j that the random walk makes if it is started in the vertex i.)

4. We relate N to the resistance between i and j if the graph $\mathcal{G}$ is viewed as an electric network.

5. We bound the electric resistance, using the fact that the resistance only increases if an edge is removed form $\mathcal{G}$. Thus, the maximum resistance is achieved if $\mathcal{G}$ is a tree.

This analysis yields that the entries of K can be characterized by the inequalities

$$\alpha^2 a[i]a[j]\le K[i,j]\le\left(\alpha^2+n\right)a[i]a[j]$$

where $a[1]=\sqrt{d_1+\alpha^{-2}}$ and $a[j]=\sqrt{d_j}$ for $j\in[2\,..\,A]$ (Lemma 14 in Section 6.3). From this we derive bounds on the largest eigenvalue of K which imply bounds on the eigenvalue of $R_{\mathsf{B}}R_{\mathsf{A}}$ that is closest to 1.

We describe the analysis in more detail in Section 6.

3.4 Better backtracking algorithm

Backtracking task. We are given a tree $\mathcal{T}$ and a black-box function

$$P:V(\mathcal{T})\to\{true,false,indeterminate\}$$

(with $P(x)$ telling us whether x is a solution to the computational problem we are trying to solve), where $V(\mathcal{T})$ stands for the set of vertices of $\mathcal{T}$ and $P(v)\in\{true,false\}$ iff v is a leaf. A vertex $v\in V(\mathcal{T})$ is called marked if $P(v)=true$. We have to determine whether $\mathcal{T}$ contains a marked vertex.

For this section, we assume that the tree is binary. (A vertex with d children can be replaced by a binary tree of depth $\lceil\log d\rceil$. This increases the size of the tree by a constant factor, the depth by a factor of at most $\lceil\log d\rceil$ and the complexity bounds by a polylogarithmic factor of d.)

Theorem 6. *[12] There is a quantum algorithm which, given*

- *a tree $\mathcal{T}$ (accessible through black boxes, as described in Section 2.1),*
- *an access to the black-box function P, and*
- *numbers T_1 and n which are upper bounds on the size and the depth of $\mathcal{T}$,*

determines if the tree contains a marked vertex with query and time complexity $O(\sqrt{T_1 n}\log\frac{1}{\epsilon})$, with the probability of a correct answer at least $1-\epsilon$.

The weakness of this theorem is that the complexity of the algorithm depends on T_1. On the other hand, a classical backtracking algorithm $\mathcal{A}$ might find a solution in substantially less than T_1 steps (either because the tree $\mathcal{T}$ contains multiple vertices $x : P(x) = true$ or because the heuristics that $\mathcal{A}$ uses to decide which branches to search first are likely to lead to $x : P(x) = true$).

We improve on Montanaro's algorithm by showing

Theorem 7. *Let $\mathcal{A}$ be a classical backtracking algorithm. There is a quantum algorithm that, with probability at least $1-\epsilon$, outputs 1 if $\mathcal{T}$ contains a marked vertex and 0 if $\mathcal{T}$ does not contain a marked vertex and uses*

$$O\left(n\sqrt{nT}\log^2\frac{n\log T_1}{\epsilon}\right)$$

queries and $O(\log T_1)$ non-query transformations per query where

- *T_1 is an upper bound on the size of $\mathcal{T}$ (which is given to the quantum algorithm),*
- *n is an upper bound on the depth of the $\mathcal{T}$ (also given to the quantum algorithm),*
- *T is the number of vertices of $\mathcal{T}$ actually explored by $\mathcal{A}$.*

Proof. The main idea of our search algorithm is to generate subtrees of $\mathcal{T}$ that consist of first approximately 2^i vertices visited by the classical backtracking strategy $\mathcal{A}$, increasing i until a marked vertex is found or until we have searched the whole tree $\mathcal{T}$.

Let $\mathcal{T}_m$ be the subtree of $\mathcal{T}$ consisting of the first m vertices visited by the classical backtracking algorithm $\mathcal{A}$. Then, we can describe T_m by giving a path

$$r = u_0 \to u_1 \to u_2 \to \ldots \to u_l = u \tag{1}$$

where u is the m^{th} vertex visited by $\mathcal{A}$. Then, $\mathcal{T}_m$ consists of all the subtrees $\mathcal{T}(u)$ rooted at u such that u is a child of u_i (for some $i \in [0\,..\,l-1]$) that is visited before u_{i+1} and the vertices $u_0, \ldots, u_l$ on the path.

Given access to $\mathcal{T}$ and the path (1), one can simulate Montanaro's algorithm on $\mathcal{T}_m$. Montanaro's algorithm consists of performing the transformations similar to R_{A} and R_{B} described in Section 3.2, except that D_v is identity if v is marked. To run Montanaro's algorithm on $\mathcal{T}_m$, we use access to $\mathcal{T}$ but modify the transformations of the algorithm as follows:

- when performing D_v for some v, we check if v is one of vertices u_i on the path;
- if $v = u_i$ for some $i \in [0\,..\,l-1]$ and u_{i+1} is the first child of u_i, we change D_v as if u_{i+1} is the only child of u_i;
- otherwise, we perform D_v as usually.

Lemma 8. *There is a quantum algorithm that generates the path*

$$r = u_0 \to u_1 \to u_2 \to \ldots \to u_l = u$$

corresponding to a subtree $\mathcal{T}_{\hat{m}}$ for $\hat{m} : |m - \hat{m}| \leq \delta m$ with a probability at least $1 - \epsilon$ and uses

$$O\left(\frac{n^{1.5}\sqrt{m}}{\delta^{1.5}} \log^2 \frac{n}{\epsilon}\right)$$

queries and $O(\log T_0)$ non-query transformations per query.

Proof. The algorithm **Generate-path** is described as Algorithm 2.

Algorithm 2 Procedure **Generate-path**(v, m)

Generate-path(v, m) returns a path (1) defining a subtree $\mathcal{T}_{\hat{m}}$, with $\hat{m}$ satisfying $|m - \hat{m}| \leq \delta m$ with a probability at least $1 - \epsilon$.

1. If v is a leaf, return the empty path.
2. Otherwise, let v_1, v_2 be the children of v, in the order in which $\mathcal{A}$ visits them.
3. Let m_1 be an estimate for the size of $\mathcal{T}(v_1)$, using the algorithm for the tree size estimation with the precision $1 \pm \delta$, the probability of a correct answer at least $1 - \frac{\epsilon}{n}$ and $\frac{m-1}{1-\delta}$ as the upper bound on the tree size.
4. If $m_1 > m - 1$, return the path obtained by concatenating the edge $v \to v_1$ with the path returned by **Generate-path**$(v_1, m - 1)$.
5. If $m_1 = m - 1$, return the path obtained by concatenating the edge $v \to v_1$ with the path from v_i to the last vertex of $\mathcal{T}(v_i)$ (that is, the path in which we start at v_i and, at each vertex, choose the child that is the last in the order in which $\mathcal{A}$ visits the vertices).
6. If $m_1 < m - 1$, return the path obtained by concatenating the edge $v \to v_2$ with the path returned by **Generate-path**$(v_2, m - 1 - m_1)$.

Correctness. Generate-path(v, m) invokes the tree size estimation once and may call itself recursively once, with v_1 or v_2 instead of v. Since the depth of the tree is at most n, the depth of the recursion is also at most n. On all levels of recursion together, there are at most n calls to tree size estimation. If we make the probability of error for tree size estimation at most $\frac{\epsilon}{n}$, the probability that all tree size estimations return sufficiently precise estimates is at least $1 - \epsilon$. Under this assumption, the number of vertices in each subtree added to $\mathcal{T}'$ is within a factor of $1 \pm \delta$ of the estimate. This means that the total number of vertices in $\mathcal{T}'$ is within $1 \pm \delta$ of m (which is equal to the sum of estimates).

Query complexity. Tree size estimation is called at most n times, with the complexity of

$$O\left(\frac{\sqrt{nm}}{\delta^{1.5}} \log^2 \frac{n}{\epsilon}\right)$$

each time, according to Theorem 2. Multiplying this complexity by n gives Lemma 8. □

Algorithm 3 Main part of the quantum algorithm for speeding up backtracking

1. Let $i = 1$.
2. Repeat:
   (a) Run **Generate-path**$(r, 2^i)$ with $\delta = \frac{1}{2}$ and error probability at most $\frac{\epsilon}{2\lceil \log T_1 \rceil}$, obtain a path defining a tree $\mathcal{T}' = \mathcal{T}_{\hat{m}}$.
   (b) Run Montanaro's algorithm on $\mathcal{T}'$, with the upper bound on the number of vertices $\frac{3}{2}2^i$ and the error probability at most $\frac{\epsilon}{2\lceil \log T_1 \rceil}$, stop if a marked vertex is found.
   (c) Let $i = i + 1$.

   until a marked vertex is found or $\mathcal{T}'$ contains the whole tree.
3. If $\mathcal{T}'$ contains the whole tree and no marked vertex was found in the last run, stop.

We now continue with the main algorithm for Theorem 7 (Algorithm 3).

Correctness. Each of the two subroutines (**Generate-path** and Montanaro's algorithm) is invoked at most $\lceil \log T_1 \rceil$ times. Hence, the probability that all invocations are correct is at least $1 - \epsilon$.

Query complexity. By Lemma 8, the number of queries performed by **Generate-path** in the i^{th} stage of the algorithm is

$$O\left(n^{1.5}\sqrt{2^i}\log^2\frac{n\log T_1}{\epsilon}\right)$$

and the complexity of Montanaro's algorithm in the same stage of the algorithm is of a smaller order. Summing over i from 1 to $\lceil \log \frac{T}{1-\delta} \rceil$ (which is the maximum possible value of i if all the subroutines are correct) gives the query complexity

$$O\left(n^{1.5}\sqrt{T}\log^2\frac{n\log T_1}{\epsilon}\right).$$

□

Note. If T is close to the size of the entire tree, the complexity of Algorithm 3 (given by Theorem 7) may be larger than the complexity of Montanaro's algorithm (given by Theorem 6). To deal with this case, one can stop Algorithm 3 when the number of queries exceeds the expression in Theorem 6 and then run Montanaro's algorithm on the whole tree. Then, the complexity of the resulting algorithm is the minimum of complexities in Theorems 6 and 7.

3.5 Evaluating AND-OR formulas of unknown structure

We now consider evaluating AND-OR formulas in a similar model where we are given the root of the formula and can discover the formula by exploring it locally. This corresponds to position trees in 2-player games where we know the starting position (the root of the tree) and, given a position, we can generate all possible positions after one move. More precisely, we assume access to

- a formula tree $\mathcal{T}$ (in the form described in Section 2.1);

- a black box which, given an internal node, answers whether AND or OR should be evaluated at this node;
- a black box which, given a leaf, answers whether the variable at this leaf is 0 or 1.

Theorem 9. *There is a quantum algorithm which evaluates an AND-OR tree of size at most T and depth $n = T^{o(1)}$ in this model running in time $O(T^{1/2+\delta})$, for an arbitrary $\delta > 0$.*

Proof. We assume that the tree is binary. (An AND/OR node with k inputs can be replaced by a binary tree of depth $\lceil \log k \rceil$ consisting of gates of the same type. This increases the size of the tree by a constant factor and the depth by a factor of at most $\lceil \log k \rceil$.)

We say that $\mathcal{T}'$ is an m-heavy element subtree of $\mathcal{T}$ if it satisfies the following properties:

1. $\mathcal{T}'$ contains all x with $|\mathcal{T}(x)| \geq m$ and all children of such x;
2. all vertices in $\mathcal{T}'$ are either x with $|\mathcal{T}(x)| \geq \frac{m}{2}$ or children of such x.

Lemma 10. *Let $\mathcal{T}$ be a tree and let T be an upper bound on the size of $\mathcal{T}$. There is a*

$$O\left(\frac{n^{1.5}T}{\sqrt{m}} \log^2 \frac{T}{\epsilon}\right)$$

time quantum algorithm that generates $\mathcal{T}'$ such that, with probability at least $1-\epsilon$, $\mathcal{T}'$ is an m-heavy element subtree of $\mathcal{T}$.

Proof. The algorithm is as follows.

Algorithm 4 Algorithm **Heavy-subtree**(r, m, ϵ)

1. Run the tree size estimation for $\mathcal{T}(r)$ with m as the upper bound on the number of vertices and parameters $\delta = \frac{1}{4}$ and $\epsilon' = \frac{m}{6nT}\epsilon$.
2. If the estimate is smaller than $\frac{2m}{3}$, return $\mathcal{T}'$ consisting of the root r only.
3. Otherwise, let $\mathcal{T}' = \{r\}$. Let v_1 and v_2 be the children of r. For each i, invoke **Heavy-subtree**(v_i, m, ϵ) recursively and add all vertices from the subtree returned by **Heavy-subtree**(v_i, m, ϵ) to $\mathcal{T}'$. Return $\mathcal{T}'$ as the result.
4. If, at some point, the number of vertices added to $\mathcal{T}'$ reaches $\frac{6T}{m}n$, stop and return the current $\mathcal{T}'$.

The proof of correctness and the complexity bounds are given in Section 7. □

To evaluate an AND-OR formula with an unknown structure, let T be an upper bound on the size of the formula F. Let c be an integer. For $i = 1, \ldots, c$, we define $T_i = T^{i/c}$. To evaluate F, we identify an T_{c-1}-heavy element subtree F' and then run the algorithm of [1] or [14] for evaluating formulas with a known structure on it. The leaves of this F' are roots of subtrees of size at most T_{c-1}. To perform queries on them, we call the same algorithm recursively. That is, given a leaf v, we identify a T_{c-2}-heavy element subtree $F'(v)$ and then run the algorithm of [1] or [14] for

Algorithm 5 Algorithm **Unknown-evaluate**(r, i, ϵ)

1. If $i = 1$, determine the structure of the tree by exploring it recursively, let $\mathcal{T}'$ be the resulting tree.
2. If $i > 1$, use **Heavy-subtree**$(r, T_{i-1}, \epsilon/5)$ to obtain $\mathcal{T}'$. Let s be the size of $\mathcal{T}'$.
3. Run the AND-OR formula evaluation algorithm for known formulas of [1] to evaluate the formula corresponding to $\mathcal{T}'$, with a probability of a correct answer at least $1 - \frac{\epsilon}{5}$. If $i > 1$, use calls to **Unknown-evaluate**$(v, i-1, \epsilon/s^3)$ instead of queries at leaves v.
4. If **Unknown-evaluate** is used as a query at a higher level (that is, if $i < c$), perform a phase flip to simulate the query and run the first three steps in reverse, erasing all the information that was obtained during this execution of **Unknown-evaluate**.

evaluating formulas with a known structure on it, with queries at the leaves replaced by another recursive call of the same algorithm, now on a subtree of size at most T_{c-2}.

The algorithm that is being called recursively is described as Algorithm 5. To evaluate the original formula F, we call **Unknown-evaluate**(r, c, ϵ).

The proof of correctness and the complexity bounds are given in Section 7. □

4 Proof of Lemma 1

We perform ordinary eigenvalue estimation $t = \lceil \frac{1}{C} \ln \frac{2}{\epsilon_{min}} \rceil$ times, with parameters $\delta_{est} = \delta_{min}$ and $\epsilon_{est} = \frac{\epsilon_{min}}{2t}$. We then take

$$\hat{\theta} = \min(|\hat{\theta_1}|, \ldots, |\hat{\theta_t}|)$$

where $\hat{\theta_1}, \ldots, \hat{\theta_t}$ are the estimates that have been obtained.

To see that $Pr[|\theta_{min} - \hat{\theta}| \leq \delta_{min}] \geq 1 - \epsilon_{min}$, we observe that:

1. The probability that none of $\hat{\theta_j}$ is an estimate for $\pm\theta_{min}$ is at most

$$(1-C)^t \leq e^{-\ln \frac{2}{\epsilon_{min}}} = \frac{\epsilon_{min}}{2}.$$

2. The probability that one or more of $\hat{\theta_j}$ differs from the corresponding θ_j by more than δ_{est} is at most $t\epsilon_{est} = \frac{\epsilon_{min}}{2}$.

If none of these two "bad events" happens, we know that, among $\hat{\theta_j}$, there is an estimate for $\pm\theta_{min}$ that differs from θ_{min} or $-\theta_{min}$ by at most δ_{est}. Moreover, any estimate $\hat{\theta_j}$ for $\theta_j \neq \pm\theta_{min}$ must be at least $|\theta_j| - \delta_{est} \geq \theta_{min} - \delta_{est}$ in absolute value. Therefore, even if $\hat{\theta_j}$ with the smallest $|\hat{\theta_j}|$ is not an estimate for $\pm\theta_{min}$, it must still be in the interval $[\theta_{min} - \delta_{est}, \theta_{min} + \delta_{est}]$.

The number of repetitions of controlled-U is

$$O\left(t\frac{1}{\delta_{est}} \log \frac{1}{\epsilon_{est}}\right) = O\left(\frac{1}{C}\left(\log \frac{1}{\epsilon_{min}}\right) \frac{1}{\delta_{min}} \log \frac{1}{\epsilon_{est}}\right)$$

and we have $\log \frac{1}{\epsilon_{est}} = \log t + \log \frac{1}{\epsilon_{min}} + O(1)$. Also, $\log t \leq \log \frac{1}{C} + \log\log \frac{1}{\epsilon_{min}} + O(1)$. Therefore,

$$\log \frac{1}{\epsilon_{est}} \leq \log \frac{1}{C} + O\left(\log \frac{1}{\epsilon_{min}}\right) = O\left(\log \frac{1}{C} \log \frac{1}{\epsilon_{min}}\right),$$

implying the lemma.

5 Implementing transformations R_{A} and R_{B}

We represent the basis states $|e\rangle$ as $|u, w\rangle$ where $u \in \mathsf{V_A}$ and $w \in \mathsf{V_B} \cup \{v_{V+1}\}$. Let $\mathcal{H}_{\mathbf{1}}$ and $\mathcal{H}_{\mathbf{2}}$ be the registers holding u and w, respectively.

Each $|s_u\rangle$, $u \in \mathsf{V_A}$, can be expressed as $|s_u\rangle = |u\rangle \otimes |s'_u\rangle$. We can view R_{A} as a transformation in which, for each u in $\mathcal{H}_{\mathbf{1}}$, we perform $D'_u = I - \frac{2}{\|s'_u\|^2} |s'_u\rangle \langle s'_u|$ on the subspace $|u\rangle \otimes \mathcal{H}_{\mathbf{2}}$.

The transformation D'_u can be performed as follows:

1. Use queries to obtain the numbers of incoming and outgoing edges $d(u)$ and $d'(u)$ (denoted by d_{in} and d_{out} from now on). Use queries to obtain vertices $w_1, \ldots, w_{d_{in}}$ with edges (w_j, u) and vertices $w'_1, \ldots, w'_{d_{out}}$ with edges (u, w_j).

2. Let $\mathcal{H}_{\mathbf{3}}$ be a register with basis states $|0\rangle, \ldots, |d+1\rangle$. Use the information from the first step to perform a map on $\mathcal{H}_{\mathbf{2}} \otimes \mathcal{H}_{\mathbf{3}}$ that maps $|w_1\rangle|0\rangle$, …, $|w_{d_{in}}\rangle|0\rangle$, $|w'_1\rangle|0\rangle$, …, $\left|w'_{d_{out}}\right\rangle|0\rangle$ to $|0\rangle|1\rangle$, …, $|0\rangle|d_{in}+d_{out}\rangle$ and, if $u = v_1$, also maps $|v_{V+1}\rangle|0\rangle$ to $|0\rangle|d_{in}+d_{out}+1\rangle$.

3. Perform the transformation $D = I - 2|\psi\rangle\langle\psi|$ on $\mathcal{H}_{\mathbf{3}}$ where $|\psi\rangle$ is the state obtained by normalizing $\sum_{i=1}^{d_{in}+d_{out}} |i\rangle$ if $u \neq v_1$ and by normalizing $\sum_{i=1}^{d_{in}+d_{out}} |i\rangle + \alpha|d_{in}+d_{out}+1\rangle$ if $u = v_1$.

4. Perform steps 2 and 1 in reverse.

The first step consists of at most $d+2$ queries (two queries to obtain $d(u)$ and $d'(u)$ and at most d queries to obtain the vertices w_i and w'_i) and some simple operations between queries, to keep count of vertices w_i or w'_i that are being queried.

For the second step, let $\mathcal{H}_j$ be the register holding the value of w_j obtained in the first step. For each $j \in \{1, \ldots, d_{in}\}$, we perform a unitary U_j on $\mathcal{H}_j \otimes \mathcal{H}_{\mathbf{2}} \otimes \mathcal{H}_{\mathbf{3}}$ that maps $|w_j\rangle|w_j\rangle|0\rangle$ to $|w_j\rangle|0\rangle|j\rangle$. This can be done as follows:

1. Perform $|x\rangle|y\rangle \to |x\rangle|x \oplus y\rangle$ on $\mathcal{H}_j \otimes \mathcal{H}_{\mathbf{2}}$ where $\oplus$ denotes bitwise XOR.

2. Conditional on the second register being $|0\rangle$, add j to the third register.

3. Conditional on the third register not being $|j\rangle$, perform $|x\rangle|y\rangle \to |x\rangle|x \oplus y\rangle$ on $\mathcal{H}_j \otimes \mathcal{H}_{\mathbf{2}}$ to reverse the first step.

We then perform similar unitaries $U_{d_{in}+j}$ that map $\left|w'_j\right\rangle\left|w'_j\right\rangle|0\rangle$ to $\left|w'_j\right\rangle|0\rangle|d_{in}+j\rangle$ and, if $u = v_1$, we also perform a unitary $U_{d_{in}+d_{out}+1}$ that maps $|v_{V+1}\rangle|0\rangle$ to $|0\rangle|d_{in}+d_{out}+1\rangle$. Each of these unitaries requires $O(\log V)$ quantum gates and there are $O(d)$ of them. Thus, the overall complexity is $O(d \log V)$.

The third step is a unitary D on $\mathcal{H}_{\mathbf{3}}$ that depends on $d_{in}+d_{out}$ and on whether we have $u = v_1$. We can express D as $D = U_\psi(I - 2|0\rangle\langle 0|)U_\psi^{-1}$ where U_ψ is any transformation with $U_\psi|0\rangle = |\psi\rangle$.

Both U_ψ and $I - 2|0\rangle\langle 0|$ are simple transformations on a $\lceil \log(d+2) \rceil$ qubit register $\mathcal{H}_\mathbf{3}$ which can be performed with $O(\log d)$ gates.

Since there are d possible values for $d_{in} + d_{out}$ and 2 possibilities for whether $u = v_1$, we can try all possibilities one after another, checking the conditions and then performing the required unitary, if necessary, with $O(d \log d)$ gates. Checking $u = v_1$ requires $O(\log V)$ gates.

The fourth step is the reverse of the first two steps and can be performed with the same complexity.

The overall complexity is $O(d)$ queries and $O(d \log V)$ quantum gates. The transformation R_B can be performed with a similar complexity in a similar way.

6 Analysis of algorithm for DAG size estimation

This section is devoted to the analysis of Algorithm 1.

6.1 Spectral theorem

Our bounds on eigenvalues of $R_\mathsf{B} R_\mathsf{A}$ are based on the spectral theorem from [15].

Suppose that $\mathcal{X}$ is a subspace of a Hilbert space $\mathcal{H}$. Let $\mathrm{ref}_\mathcal{X}$ denote a reflection which leaves $\mathcal{X}$ invariant and negates all vectors in $\mathcal{X}^\perp$.

Let $Gram(w_1, \ldots, w_k)$ stand for the Gram matrix of vectors $w_1, \ldots, w_k$, i.e., the matrix formed by the inner products $\langle w_s, w_t \rangle$.

Definition 11 ([15, Definition 5])**.** Let $(\{w_1, \ldots, w_k\}, \{\tilde{w}_1, \ldots, \tilde{w}_l\})$ be an ordered pair of orthonormal systems. The discriminant matrix of this pair is

$$M = Gram(w_1, \ldots, w_k, \tilde{w}_1, \ldots, \tilde{w}_l) - I.$$

Let $\mathcal{A}$ and $\mathcal{B}$ be two subspaces of the same Hilbert space $\mathcal{H}$. Suppose that $\mathcal{A}$ is spanned by an orthonormal basis $w_1, \ldots, w_k$ and $\mathcal{B}$ is spanned by an orthonormal basis $\tilde{w}_1, \ldots, \tilde{w}_l$. Let M be the discriminant matrix of $(\{w_1, \ldots, w_k\}, \{\tilde{w}_1, \ldots, \tilde{w}_l\})$. The spectral theorem from [15] provides spectral decomposition for the operator $\mathrm{ref}_\mathcal{B}\mathrm{ref}_\mathcal{A}$ restricted to $\mathcal{A}+\mathcal{B} = (\mathcal{A}^\perp \cap \mathcal{B}^\perp)^\perp$; on the subspace $\mathcal{A}^\perp \cap \mathcal{B}^\perp$ (called the *idle subspace* in [15]) $\mathrm{ref}_\mathcal{B}\mathrm{ref}_\mathcal{A}$ acts as the identity.

Theorem 12 ([15, Spectral Theorem])**.** $\mathrm{ref}_\mathcal{B}\mathrm{ref}_\mathcal{A}$ *has the following eigenvalues on* $\mathcal{A} + \mathcal{B}$*:*

- *eigenvalue 1, with eigenvectors being all vectors in* $\mathcal{A} \cap \mathcal{B}$*; the space has the same dimension as the eigenspace of* M *associated to the eigenvalue 1.*
- *eigenvalue* $2\lambda^2 - 1 \mp 2i\lambda\sqrt{1-\lambda^2}$*, with corresponding eigenvectors* $\left(|\tilde{a}\rangle - \lambda \left|\tilde{b}\right\rangle\right) \pm i\sqrt{1-\lambda^2}\left|\tilde{b}\right\rangle$*, where* (a, b) *is an eigenvector of* M *with eigenvalue* $\lambda \in (0, 1)$ *and*

$$|\tilde{a}\rangle := \sum_{i=1}^{k} a_i w_i, \quad \left|\tilde{b}\right\rangle := \sum_{j=1}^{l} b_j \tilde{w}_j.$$

- *eigenvalue* -1*, with eigenvectors being all vectors in form* $|\tilde{a}\rangle$ *or* $\left|\tilde{b}\right\rangle$*, where* (a, b) *is an eigenvector of* M *with eigenvalue 0.*

Since

$$\mathrm{ref}_{\mathcal{A}^\perp} = -\mathrm{ref}_{\mathcal{A}}, \quad \mathrm{ref}_{\mathcal{B}^\perp} = -\mathrm{ref}_{\mathcal{B}}, \qquad \mathrm{ref}_{\mathcal{B}^\perp}\mathrm{ref}_{\mathcal{A}^\perp} = \mathrm{ref}_{\mathcal{B}}\mathrm{ref}_{\mathcal{A}},$$

we can restrict the operator $\mathrm{ref}_{\mathcal{B}^\perp}\mathrm{ref}_{\mathcal{A}^\perp}$ on $\mathcal{A}+\mathcal{B}$ and obtain its spectral decomposition in terms of the discriminant matrix of the pair $(\{w_1,\ldots,w_k\},\{\tilde{w}_1,\ldots,\tilde{w}_l\})$ (instead of forming the discriminant matrix of the orthogonal systems spanning $\mathcal{A}^\perp$ and $\mathcal{B}^\perp$).

6.2 1-eigenvectors of $R_{\mathsf{B}}R_{\mathsf{A}}$

In our setting $R_{\mathsf{B}} = \mathrm{ref}_{\mathcal{H}_{\mathsf{B}}^\perp}$ and $R_{\mathsf{A}} = \mathrm{ref}_{\mathcal{H}_{\mathsf{A}}^\perp}$.

From the spectral theorem it follows that all 1-eigenvectors of $R_{\mathsf{B}}R_{\mathsf{A}}$ belong to either $\mathcal{H}_{\mathsf{A}}^\perp \cap \mathcal{H}_{\mathsf{B}}^\perp$ or $\mathcal{H}_{\mathsf{A}} \cap \mathcal{H}_{\mathsf{B}}$. We start with characterizing these 1-eigenspaces as follows:

Lemma 13. *1. The starting state $|e_{T+1}\rangle$ is orthogonal to each state in $\mathcal{H}_{\mathsf{A}}^\perp \cap \mathcal{H}_{\mathsf{B}}^\perp$.*

2. $\dim(\mathcal{H}_{\mathsf{A}} \cap \mathcal{H}_{\mathsf{B}}) = 0$.

Proof. The first claim can be restated as

$$|e_{T+1}\rangle \in \left(\mathcal{H}_{\mathsf{A}}^\perp \cap \mathcal{H}_{\mathsf{B}}^\perp\right)^\perp = \mathcal{H}_{\mathsf{A}} + \mathcal{H}_{\mathsf{B}}.$$

To show that, notice that the following equality holds:

$$\sum_{v\in\mathsf{V}_{\mathsf{A}}}\sum_{e\in\mathsf{E}(v)}|e\rangle = \sum_{v\in\mathsf{V}_{\mathsf{B}}}\sum_{e\in\mathsf{E}(v)}|e\rangle, \tag{2}$$

since for every edge $e\in\mathsf{E}$ the state $|e\rangle$ is added both in the LHS and RHS of (2) exactly once: if e connects a vertex $u\in\mathsf{V}_{\mathsf{A}}$ and a vertex $v\in\mathsf{V}_{\mathsf{B}}$ (no edge can connect two vertices from V_{A} or from V_{B}), then $|e\rangle$ appears in the sum $\sum_{e'\in\mathsf{E}(u)}|e'\rangle$ in the LHS and in the sum $\sum_{e'\in\mathsf{E}(v)}|e'\rangle$ in the RHS (and, for any other vertex $w\notin\{u,v\}$, $|e\rangle$ is not contained in the sum $\sum_{e'\in\mathsf{E}(w)}|e'\rangle$).

Now from (2) we conclude that

$$|e_{T+1}\rangle = |e_{T+1}\rangle + \alpha\sum_{v\in\mathsf{V}_{\mathsf{A}}}\sum_{e\in\mathsf{E}(v)}|e\rangle - \alpha\sum_{v\in\mathsf{V}_{\mathsf{B}}}\sum_{e\in\mathsf{E}(v)}|e\rangle = |s_{v_1}\rangle + \alpha\sum_{v\in\mathsf{V}_{\mathsf{A}}\setminus\{v_1\}}|s_v\rangle - \alpha\sum_{v\in\mathsf{V}_{\mathsf{B}}}|s_v\rangle,$$

i.e., $|e_{T+1}\rangle\in\mathcal{H}_{\mathsf{A}}+\mathcal{H}_{\mathsf{B}}$ as claimed.

To show the second claim, suppose a vector $|x\rangle$ is contained both in $\mathcal{H}_{\mathsf{A}}$ and $\mathcal{H}_{\mathsf{B}}$. Then $|x\rangle$ can be expressed in two ways via the vectors $|s_v\rangle$, i.e., there are scalars η_i, $i\in[V]$, such that

$$\sum_{i=A+1}^{A+B}\eta_i\sum_{e\in\mathsf{E}(v_i)}|e\rangle = \sum_{i=1}^{A}\eta_i\sum_{e\in\mathsf{E}(v_i)}|e\rangle + \eta_1\alpha^{-1}|e_{T+1}\rangle. \tag{3}$$

Clearly, for any adjacent vertices $v_i\sim v_j$ we must have $\eta_i=\eta_j$, since the corresponding basis state $|e\rangle$ (where e is the unique edge between v_i and v_j) has coefficient η_i in the LHS of (3) (supposing that v_i is in V_{A}) and coefficient η_j in the RHS of (3). However, $\mathcal{G}$ is connected, therefore we must have $\eta_1=\eta_2=\ldots=\eta_V$. It remains to notice that $\eta_1=0$, since the LHS of (3) is orthogonal to $|e_{T+1}\rangle$. We conclude that only the null vector belongs to the subspace $\mathcal{H}_{\mathsf{A}}\cap\mathcal{H}_{\mathsf{B}}$. □

An immediate consequence of this Lemma is that all 1-eigenvectors $|\psi\rangle$ of $R_{\mathsf{B}}R_{\mathsf{A}}$ are orthogonal to the starting state $|e_{T+1}\rangle$.

6.3 Eigenvalue closest to 1

The spectral decomposition for $R_{\mathsf{B}}R_{\mathsf{A}}$ (restricted to $\mathcal{H}_{\mathsf{A}}+\mathcal{H}_{\mathsf{B}}$) will be obtained from the discriminant matrix of two orthonormal systems spanning $\mathcal{H}_{\mathsf{A}}$ and $\mathcal{H}_{\mathsf{B}}$; in $(\mathcal{H}_{\mathsf{A}}+\mathcal{H}_{\mathsf{B}})^{\perp}=\mathcal{H}_{\mathsf{A}}^{\perp}\cap\mathcal{H}_{\mathsf{B}}^{\perp}$ the operator $R_{\mathsf{B}}R_{\mathsf{A}}$ acts as the identity.

From Theorem 12 and Lemma 13 it follows that the discriminant matrix does not have the eigenvalue 1. Let $\lambda\in(0,1)$ be the maximal eigenvalue of the discriminant matrix and $\theta=2\arccos\lambda$, then $e^{\pm i\theta}$ is the eigenvalue of $R_{\mathsf{B}}R_{\mathsf{A}}$ which is closest to 1.

To describe the discriminant matrix, we introduce the following notation. We define a $(T+1)\times A$ matrix M_{a} as follows:

- the elements of the first column are defined by

$$M_{\mathsf{a}}[i,1]=\begin{cases}1, & e_i\in\mathsf{E}(v_1),\\ \alpha^{-1}, & i=T+1,\\ 0, & \text{otherwise;}\end{cases}$$

- the elements of the j^{th} column, $j=2,3,\ldots,A$, are defined by

$$M_{\mathsf{a}}[i,j]=\begin{cases}1, & e_i\in\mathsf{E}(v_j),\\ 0, & \text{otherwise.}\end{cases}$$

A $(T+1)\times B$ matrix M_{b} is defined by

$$M_{\mathsf{b}}[i,j]=\begin{cases}1, & e_i\in\mathsf{E}(v_{A+j}),\\ 0, & \text{otherwise,}\end{cases}\qquad i\in[T+1],\ j\in[B].$$

Then $\mathcal{H}_{\mathsf{A}}$ and $\mathcal{H}_{\mathsf{B}}$ can be identified with the column spaces of M_{a} and M_{b}, respectively. Let $a\in\mathbb{R}^A$ and and $b\in\mathbb{R}^B$ be vectors defined by

$$a[1]=\sqrt{d_1+\alpha^{-2}},\ a[j]=\sqrt{d_j},\ j\in[2\,..\,A]\quad\text{ and }\quad b[j]=\sqrt{d_{A+j}},\ j\in[B].\tag{4}$$

By M_{A} we denote the matrix $M_{\mathsf{a}}\operatorname{diag}(a)^{-1}$ and by M_{B} we denote the matrix $M_{\mathsf{b}}\operatorname{diag}(b)^{-1}$. Notice that columns of M_{A} and M_{B} are orthonormal vectors. The corresponding vectors

$$\sum_{i\in[T+1]}M_{\mathsf{A}}[i,j]\,|i\rangle=\frac{\left|s_{v_j}\right\rangle}{\left\|s_{v_j}\right\|},\ j\in[A],\quad\text{ and }\quad\sum_{i\in[T+1]}M_{\mathsf{B}}[i,j]\,|i\rangle=\frac{\left|s_{v_{A+j}}\right\rangle}{\left\|s_{v_{A+j}}\right\|},\ j\in[B]\tag{5}$$

form orthonormal bases of $\mathcal{H}_{\mathsf{A}}$ and $\mathcal{H}_{\mathsf{B}}$, respectively.

Let $L=M_{\mathsf{A}}^{*}M_{\mathsf{B}}$. Then the discriminant matrix of the pair of orthonormal systems spanning $\mathcal{H}_{\mathsf{A}}$ and $\mathcal{H}_{\mathsf{B}}$ has the following block structure:

$$\begin{pmatrix}\mathbf{0}_{A,A} & L\\ L^{*} & \mathbf{0}_{B,B}\end{pmatrix}.$$

It is easy to check that $\left(\begin{smallmatrix}u\\ v\end{smallmatrix}\right)$ is an eigenvector of the discriminant matrix with an eigenvalue $\lambda>0$ iff u is a left-singular vector and v is the right-singular vector of L with singular value λ.

Therefore, if $\lambda_L = \cos\frac{\theta}{2}$ is the largest singular value of L, then $e^{\pm i\theta}$ are the eigenvalues of $R_{\mathsf{B}}R_{\mathsf{A}}$ that are closest to 1.

Let $K = (I - LL^*)^{-1}$ and λ_K be the maximal eigenvalue of K. Since $\lambda_L = \cos\frac{\theta}{2} < 1$ is the maximal singular value of L, it holds that $\lambda_K = \left(1 - \lambda_L^2\right)^{-1} = \sin^{-2}\frac{\theta}{2}$ is the maximal eigenvalue of K. We show the following characterization of the entries of K:

Lemma 14. *For all $i, j \in [A]$ the following inequalities hold:*

$$\alpha^2 a[i]a[j] \le K[i,j] \le \left(\alpha^2 + n\right) a[i]a[j].$$

Moreover, when $i = 1$ or $j = 1$, we have $K[i,j] = \alpha^2 a[i]a[j]$.

Proof in Sections 6.5 - 6.8.

Lemma 14 now allows to estimate the maximal eigenvalue of K:

Lemma 15. *The entries $K[i,j], i, j \in [A]$, satisfy*

$$\alpha^2\sqrt{d_i d_j} \le K[i,j] \le \sqrt{d_i d_j}\left(\alpha^2 + n\right). \tag{6}$$

Furthermore, λ_K, the maximal eigenvalue of K, satisfies

$$\alpha^2 T \le \lambda_K \le (\alpha^2 + n)T. \tag{7}$$

Proof. Since $a[i] = \sqrt{d_i}$ for all $i \in [2\,..\,A]$, inequalities (6) immediately follow from Lemma 14 when $i, j \ge 2$. Suppose that $i = 1$ (since K is symmetric), then, again by Lemma 14, $K[1,j] = \alpha^2 a[1]a[j]$. Since $a[j] \ge \sqrt{d_j}$ for all $j \in [A]$, the first inequality in (6) is obvious. It remains to show

$$\alpha^2 a[1]a[j] \le \sqrt{d_1 d_j}\left(\alpha^2 + n\right), \quad j \in [A]. \tag{8}$$

However, from the definition of a we have $\alpha a[j] \le \sqrt{\alpha^2 d_j + 1}$ for all $j \in [A]$. Hence the LHS of (8) is upper-bounded by $\sqrt{(\alpha^2 d_1 + 1)(\alpha^2 d_j + 1)}$, which, in turn, is upper bounded by the RHS of (8). This proves (6).

Let K' be a symmetric $A \times A$ matrix, defined by $K'[i,j] = \sqrt{d_i d_j}$. Then (6) can be restated as

$$\alpha^2 K'[i,j] \le K[i,j] \le (\alpha^2 + n)K'[i,j], \qquad i, j \in [A].$$

Now, from [7, Theorem 8.1.18] we have that

$$\lambda(\alpha^2 K') \le \lambda_K \le \lambda\left((\alpha^2 + n)K'\right),$$

where by $\lambda(M)$ we denote the spectral radius of a matrix M (the maximum absolute value of an eigenvalue of M), since $\lambda_K = \lambda(K)$. On the other hand, K' is a rank-1 matrix, thus its spectral radius is $\lambda(K') = \sum_{j=1}^{A}(\sqrt{d_j})^2 = \sum_{j=1}^{A} d_j = T$ (on each side of the last equality every edge in E is counted exactly once), hence

$$\alpha^2 T = \lambda(\alpha^2 K') \le \lambda_K \le \lambda\left((\alpha^2 + n)K'\right) = (\alpha^2 + n)T.$$

□

6.4 Phase estimation for θ

We now show that Lemma 15 implies Lemmas 4 and 5.

Let u be the left-singular vector and v be the right-singular vector of L corresponding to the largest singular value λ_L. By Theorem 12, the corresponding eigenvectors of $R_{\mathsf{B}}R_{\mathsf{A}}$ are

$$|\Psi_{\pm}\rangle = \frac{1}{\sqrt{2(1-\lambda_L^2)}}\,|\tilde{a}\rangle - \frac{\lambda_L}{\sqrt{2(1-\lambda_L^2)}}\left|\tilde{b}\right\rangle \mp \frac{i}{\sqrt{2}}\left|\tilde{b}\right\rangle,$$

where $|\tilde{a}\rangle \in \mathcal{H}_{\mathsf{A}}$, $\left|\tilde{b}\right\rangle \in \mathcal{H}_{\mathsf{B}}$ are the unit vectors associated to $M_{\mathsf{A}}u$ and $M_{\mathsf{B}}v$, i.e.,

$$|\tilde{a}\rangle = \sum_{j=1}^{A} \frac{\left|s_{v_j}\right\rangle}{\left\|s_{v_j}\right\|}u[j], \quad \left|\tilde{b}\right\rangle = \sum_{j=1}^{B} \frac{\left|s_{v_{A+j}}\right\rangle}{\left\|s_{v_{A+j}}\right\|}v[j].$$

The two-dimensional plane $\Pi = \operatorname{span}\{|\Psi_+\rangle, |\Psi_-\rangle\}$ is also spanned by

$$|q_1\rangle = \left|\tilde{b}\right\rangle, \quad |q_2\rangle = \frac{1}{\sqrt{1-\lambda_L^2}}\,|\tilde{a}\rangle - \frac{\lambda_L}{\sqrt{1-\lambda_L^2}}\left|\tilde{b}\right\rangle. \tag{9}$$

We claim that

Lemma 4. *If $\alpha \geq \sqrt{2n}$, we have*

$$\langle e_{T+1}\,|q_2\rangle \geq \frac{2}{3}$$

for the state $|q_2\rangle \in \operatorname{span}\{|\Psi_+\rangle, |\Psi_-\rangle\}$, defined by (9).

Proof. Since $|e_{T+1}\rangle \perp \mathcal{H}_{\mathsf{B}}$, we have $\langle e_{T+1}\left|\tilde{b}\right\rangle = 0$ and

$$\langle e_{T+1}\,|q_2\rangle = \frac{1}{\sqrt{1-\lambda_L^2}}\,\langle e_{T+1}\,|\tilde{a}\rangle = \sqrt{\lambda_K}\,\langle e_{T+1}\,|\tilde{a}\rangle\,.$$

Since $\langle e_{T+1}\left|s_{v_j}\right\rangle = 0$ unless $j = 1$, we have

$$\langle e_{T+1}\,|\tilde{a}\rangle = \sum_{j=1}^{A} \frac{\langle e_{T+1}\left|s_{v_j}\right\rangle}{\left\|s_{v_j}\right\|}u[j] = \frac{u[1]}{\sqrt{1+\alpha^2 d_1}}.$$

Consequently,

$$\langle e_{T+1}\,|q_2\rangle = \frac{u[1]\sqrt{\lambda_K}}{\sqrt{1+\alpha^2 d_1}}. \tag{10}$$

We now lower bound this expression. Since u is an eigenvector of K, we have

$$\lambda_K u[i] = \sum_{j=1}^{A} u[j]K[i,j], \quad i \in [A].$$

From (6) it follows that

$$\lambda_K u[i] \le (\alpha^2+n)\sqrt{d_i}\left(\sum_{j=1}^{A}\sqrt{d_j}u[j]\right), \quad i \in [A].$$

Denote $\mu = \sum_{j=1}^{A}\sqrt{d_j}u[j]$; then the previous inequality can be rewritten as

$$u[i] \le \mu \cdot \frac{\alpha^2+n}{\lambda_K}\sqrt{d_i}, \quad i \in [A].$$

On the other hand, u is a unit vector, thus

$$1 = \sum_{i=1}^{A} u^2[i] \le \mu^2 \left(\frac{\alpha^2+n}{\lambda_K}\right)^2 \sum_{i=1}^{A} d_i = \mu^2 \left(\frac{\alpha^2+n}{\lambda_K}\right)^2 T.$$

Now we conclude that

$$\mu \ge \frac{\lambda_K}{(\alpha^2+n)\sqrt{T}}.$$

From Lemma 14 it follows that $K[1,j] = \alpha^2 a[1]a[j] \ge \alpha^2\sqrt{d_1+\alpha^{-2}}\sqrt{d_j}$ for all $j \in [A]$. That allows to estimate the RHS of the equation

$$u[1] = \frac{1}{\lambda_K}\sum_{j=1}^{A} u[j]K[1,j]$$

more precisely:

$$u[1] \ge \frac{\alpha^2\sqrt{d_1+\alpha^{-2}}}{\lambda_K}\sum_{j=1}^{A}\sqrt{d_j}u[j] = \frac{\alpha^2\sqrt{d_1+\alpha^{-2}}}{\lambda_K}\mu \ge \frac{\alpha^2}{\alpha^2+n}\cdot\frac{\sqrt{\alpha^2 d_1+1}}{\alpha\sqrt{T}}.$$

Combining this with (10) and the estimate $\sqrt{\lambda_K} \ge \alpha\sqrt{T}$ (which follows from (7)) yields

$$\langle e_{T+1}\,|q_2\rangle = \frac{u[1]\sqrt{\lambda_K}}{\sqrt{1+\alpha^2 d_1}} \ge \frac{\alpha^2}{\alpha^2+n} = 1 - \frac{n}{\alpha^2+n} \ge \frac{2}{3},$$

which completes the proof. □

Lemma 5. *Suppose that $\delta \in (0,1)$. Let $\alpha = \sqrt{2n\delta^{-1}}$ and $\hat{\theta} \in (0;\pi/2)$ satisfy*

$$\left|\hat{\theta}-\theta\right| \le \frac{\delta^{1.5}}{24\sqrt{3nT}}.$$

Then

$$(1-\delta)T \le \frac{1}{\alpha^2\sin^2\frac{\hat{\theta}}{2}} \le (1+\delta)T.$$

Proof. We start with

Lemma 16. *Suppose that $\hat{\theta} \in (0; \pi/2)$ satisfies*

$$\left|\theta - \hat{\theta}\right| \leq \epsilon \sin\frac{\theta}{2}$$

for some $\epsilon \in (0,1)$. Then

$$\left|\frac{1}{\sin^2\frac{\hat{\theta}}{2}} - \frac{1}{\sin^2\frac{\theta}{2}}\right| \leq \frac{6\epsilon}{\sin^2\frac{\theta}{2}}.$$

Proof. Let $\sigma = \sin\frac{\theta}{2}$, then $\left|\theta - \hat{\theta}\right| \leq \epsilon\sigma$ and we have to show that

$$\left|\frac{1}{\sin^2\frac{\hat{\theta}}{2}} - \frac{1}{\sin^2\frac{\theta}{2}}\right| \leq \frac{6\epsilon}{\sigma^2}.$$

Since the sin function is Lipschitz continuous with Lipschitz constant 1, we have

$$\left|\sin\frac{\hat{\theta}}{2} - \sin\frac{\theta}{2}\right| \leq \frac{\epsilon\sigma}{2}$$

and $\sigma\left(1 - \frac{\epsilon}{2}\right) \leq \sin\frac{\hat{\theta}}{2} \leq \sigma\left(1 + \frac{\epsilon}{2}\right)$. On the other hand,

$$\left|\sin^2\frac{\hat{\theta}}{2} - \sin^2\frac{\theta}{2}\right| = \left|\sin\frac{\hat{\theta}}{2} - \sin\frac{\theta}{2}\right|\left|\sin\frac{\hat{\theta}}{2} + \sin\frac{\theta}{2}\right| \leq \frac{\epsilon\sigma}{2}\cdot 2\max\left\{\sin\frac{\hat{\theta}}{2}, \sin\frac{\theta}{2}\right\} \leq \epsilon\left(1 + \frac{\epsilon}{2}\right)\sigma^2.$$

We conclude that

$$\left|\frac{1}{\sin^2\frac{\hat{\theta}}{2}} - \frac{1}{\sin^2\frac{\theta}{2}}\right| = \frac{\left|\sin^2\frac{\hat{\theta}}{2} - \sin^2\frac{\theta}{2}\right|}{\sin^2\frac{\hat{\theta}}{2}\cdot\sin^2\frac{\theta}{2}} \leq \frac{\epsilon\left(1+\frac{\epsilon}{2}\right)\sigma^2}{\sigma^4\left(1 - \epsilon + \epsilon^2/4\right)} \leq \frac{\epsilon(1+\epsilon/2)}{\sigma^2\left(1-\epsilon+\epsilon^2/4\right)} < \frac{6\epsilon}{\sigma^2},$$

where the last inequality is due to $2x(1+x)/(1-x)^2 < 12x$, which is valid for all $x \in (0, 1/2)$, and, in particular, for $x = \epsilon/2$. □

From Lemma 15 it follows that $\sigma := \sin\frac{\theta}{2}$ satisfies

$$\alpha^2 T \leq \sigma^{-2} \leq (\alpha^2 + n)T. \tag{11}$$

Notice that

$$\left|\theta - \hat{\theta}\right| \leq \frac{\delta^{1.5}}{24\sqrt{3nT}} = \frac{\delta}{24}\cdot\sqrt{\frac{\delta}{3}}\cdot\frac{1}{\sqrt{nT}} < \frac{\delta}{24}\cdot\sqrt{\frac{\delta}{2+\delta}}\cdot\frac{1}{\sqrt{nT}} = \frac{\delta}{24\sqrt{\left(\frac{2n}{\delta} + n\right)T}}.$$

Since $\frac{2n}{\delta} = \alpha^2$, the RHS of the last equality is upper-bounded by (11) as $\frac{\delta}{24\sqrt{(\alpha^2+n)T}} \leq \frac{\delta\sigma}{24}$. An application of Lemma 16 (with $\epsilon = \delta/24 < 1$) now gives

$$\left|\frac{1}{\sin^2\frac{\hat{\theta}}{2}} - \sigma^{-2}\right| \leq \frac{\delta\sigma^{-2}}{4}$$

or

$$\sigma^{-2}\left(1-\frac{\delta}{4}\right) \le \frac{1}{\sin^2\frac{\hat{\theta}}{2}} \le \sigma^{-2}\left(1+\frac{\delta}{4}\right).$$

From (11) it follows that

$$\left(1-\frac{\delta}{4}\right)\alpha^2 T \le \frac{1}{\sin^2\frac{\hat{\theta}}{2}} \le \left(1+\frac{\delta}{4}\right)(\alpha^2+n)T.$$

Consequently,

$$\left(1-\frac{\delta}{4}\right)T \le \frac{1}{\alpha^2\sin^2\frac{\hat{\theta}}{2}} \le \left(1+\frac{\delta}{4}\right)\left(1+\frac{n}{\alpha^2}\right)T = \left(1+\frac{\delta}{4}\right)\left(1+\frac{\delta}{2}\right)T.$$

It remains to notice that

$$\left(1+\frac{\delta}{4}\right)\left(1+\frac{\delta}{2}\right) = 1+\frac{3}{4}\delta+\frac{\delta^2}{8} < 1+\delta.$$

Hence

$$\left(1-\frac{\delta}{4}\right)T \le \frac{1}{\alpha^2\sin^2\frac{\hat{\theta}}{2}} \le (1+\delta)\,T,$$

and the claim follows.

□

6.5 Harmonic functions and electric networks

The next five subsections are devoted to the proof of Lemma 15. We start with the concept of harmonic functions which is linked to the connection between electric networks and random walks. More details on the subject can be found in [10, Sec. 4] (in the case of a simple random walk which is the framework we use below), [8, Lect. 9], [11, Chap. 2] and [9, Chap. 9] (in a more general setting with weighted graphs).

Throughout the rest of this subsection suppose that $\mathbf{P}$ is a transition matrix of an irreducible Markov chain with a state space Ω.

Definition 17. Consider a function $f : \Omega \to \mathbb{R}$. We say that f is *harmonic* for $\mathbf{P}$ at $x \in \Omega$ (or simply "harmonic at x") if [9, Eq. 1.28]

$$f(x) = \sum_{y\in\Omega} \mathbf{P}[x,y]f(y),$$

i.e., f has the *averaging property* at x. If f is harmonic for every $x \in \Omega' \subset \Omega$, then f is said to be harmonic on the subset Ω'.

It can be easily seen that a linear combination of harmonic functions is still harmonic. In particular, all constant functions are harmonic on Ω.

It is known that a harmonic (on Ω') function attains its maximal and minimal values (in the set Ω) on $\Omega \setminus \Omega'$ (it appears as an exercise in [9, Exercise 1.12]; in the context of weighted random walks it can be found in [13, Lemma 3.1]).

Lemma 18 (Maximum Principle)**.** *Let f be harmonic for $\mathbf{P}$ on $\Omega' \subset \Omega$, with $\Omega \setminus \Omega' \neq \emptyset$. Then there exists $x \in \Omega \setminus \Omega'$ s.t. $f(x) \geq f(y)$ for all $y \in \Omega'$.*

By applying the Maximum Principle for $-f$, one obtains a similar statement for the minimal values of f. In particular, if the function f is harmonic on Ω' and constant on the set $\Omega \setminus \Omega'$, then f is constant. A consequence of this is the Uniqueness Principle: if f and g are harmonic on Ω' and $f \equiv g$ on $\Omega \setminus \Omega'$, then $f \equiv g$ on Ω. Moreover, when f is harmonic everywhere on Ω, it is a constant function.

Further, suppose that $\mathbf{P}$ is a simple random walk on a finite connected undirected graph $\mathcal{G} = (\Omega, \mathcal{E})$, in the sense that

$$\mathbf{P}[x, y] = \frac{\mathbb{1}_{\{x \sim y\}}}{d(x)}, \quad d(x) := \Big| \{y \in \Omega \,\big|\, x \sim y\} \Big|,$$

where $\mathbb{1}$ stands for the indicator function. Then the Markov chain, corresponding to $\mathbf{P}$, is time-reversible; the graph $\mathcal{G}$ can be viewed as an electric network where each edge has unit conductance (this approach can be further generalized to weighted graphs where the weight $c(e)$ of an edge $e \in \mathcal{E}$ is referred to as conductance in the electric network theory, whereas its reciprocal $r(e) = c(e)^{-1}$ is called resistance).

For a subset $\Omega' \subset \Omega$ we call the set $\{x \in \Omega \setminus \Omega' \,\big|\, \exists y \in \Omega' : x \sim y\}$ the boundary of Ω' and denote by $\partial\Omega'$. The Maximum Principle can be strengthened as follows:

Lemma 18'. *Let f be harmonic for $\mathbf{P}$ on $\Omega' \subset \Omega$, with $\partial\Omega' \neq \emptyset$. Then there exists $x \in \partial\Omega'$ s.t. $f(x) \geq f(y)$ for all $y \in \Omega'$.*

Similarly,

- if the function f is harmonic on Ω' and constant on the boundary $\partial\Omega'$ (for example, when the boundary is a singleton), then f is constant on $\Omega' \cup \partial\Omega'$.
- if f and g are harmonic on Ω' and $f \equiv g$ on $\partial\Omega'$, then $f \equiv g$ on $\Omega' \cup \partial\Omega'$.

Let $s, t \in \Omega$ be two different vertices of the graph $\mathcal{G}$ and consider the unique function $\phi_{st} : \Omega \to \mathbb{R}$, which

- is harmonic on $\Omega \setminus \{s, t\}$,
- satisfies boundary conditions $\phi_{st}(s) = 1$, $\phi_{st}(t) = 0$.

From the Maximum Principle it follows that values of ϕ_{st} are between 0 and 1. In the electric network theory, the function $\phi_{st}(u)$, $u \in \Omega$, is interpreted as the voltage of u, if we put current through the electric network associated to $\mathcal{G}$, where the voltage of s is 1 and the voltage of t is 0 (see [10, p. 22] or [8, p. 60]; in the latter ϕ_{st} is denoted by $\tilde{V}$). For the random walk with the transition matrix $\mathbf{P}$, the value $\phi_{st}(u)$ is the probability that the random walk starting at u visits s before t.

Consider the quantity

$$R_{st} = \left(\sum_{u : u \sim t} \phi_{st}(u) \right)^{-1}.$$

The electrical connection is that this quantity, called the *effective resistance* (or simply resistance between s and t in [10]), is the voltage difference that would result between s and t if one unit of current was driven between them. On the other hand, it is linked to the "escape probabilities", because $(d(t)R_{st})^{-1}$ is the probability that a random walk, starting at t, visits s before returning back to t [8, p. 61].

An important result is the Rayleigh's Monotonicity Law [9, Theorem 9.12], from which it follows that adding edges in the graph cannot increase the effective resistance [10, Corollary 4.3], [9, Corollary 9.13]. More precisely, if we add another edge in $\mathcal{G}$, obtaining a graph $\mathcal{G}' = (\Omega, \mathcal{E}')$, and denote by R'_{st} the effective resistance between vertices s and t, then $R_{st} \geq R'_{st}$.

More generally, if we consider the same graph $\mathcal{G}$, but with different weights (or conductances) $c(x, y)$ and $c'(x, y)$, satisfying $c(x, y) \leq c'(x, y)$ for all $x, y \in \Omega$, then the Monotonicity Law says that the effective resistances satisfy the opposite inequality $R_{st} \geq R'_{st}$ for all distinct $s, t \in \Omega$.

We can view adding an edge as increasing its weight from 0 to 1, hence the claim about edge adding.

6.6 Extended DAG and an absorbing random walk

Let $\mathcal{G}$ and $\mathcal{G}'$ be as defined in Section 3.2. In this Section, we define an absorbing random walk on $\mathcal{G}''$, a slightly extended version of $\mathcal{G}'$.

Let Γ denote the adjacency matrix of the weighted graph $\mathcal{G}'$. We denote $\beta = \left(d_1\alpha^2 + 1\right)^{-1}$.

We introduce another vertex v_{V+2} and connect the vertex v_{V+1} with an edge $e_{T+2} = (v_{V+2}, v_{V+1})$. Let $\mathsf{V}'' = \mathsf{V}' \cup \{v_{V+2}\}$, $\mathsf{E}'' = \mathsf{E}' \cup \{e_{T+2}\}$ and $\mathcal{G}'' = (\mathsf{V}'', \mathsf{E}'')$.

Finally, let all edges in the original DAG $\mathcal{G}$ have weight 1, but the two additional edges have the following weights:

- the edge e_{T+1} has weight α^{-2};
- the edge e_{T+2} has weight d_1.

For any two vertices v_i, v_j, $i, j \in [V+2]$, we denote $v_i \sim v_j$ if there is an edge between v_i and v_j in the DAG $\mathcal{G}''$. For each $i \in [V+2]$ we denote by $d''(i)$ the degree of the vertex v_i in the weighted graph $\mathcal{G}''$. Then

- $d''(1) = d''(V+1) = d_1 + \alpha^{-2}$;
- $d''(i)$ for each $i \in [2\,..\,V]$ equals d_i, i.e., the degree of v_i in the unweighted graph $\mathcal{G}$;
- $d''(V+2) = d_1$.

Notice that

$$a[i] = \sqrt{d''(i)},\ i \in [A], \quad \text{and} \quad b[j] = \sqrt{d''(A+j)},\ j \in [B],$$

for vectors a and b defined with (4).

Consider a random walk on the graph $\mathcal{G}''$ with transition probabilities as follows:

- when at the vertex v_{V+2}, with probability 1 stay at v_{V+2};
- when at the vertex v_{V+1}, with probability $1 - \beta$ move to v_{V+2} and with probability β move to v_1;

- when at the vertex v_1, with probability β move to v_{V+1} and with probability $\frac{1-\beta}{d_1} = \beta\alpha^2$ move to any $v \in \mathsf{E}(v_1)$ (i.e., to any neighbor of the root, different from v_{V+1});
- at any vertex v_i, $i \in [2\,..\,V]$, with probability $\frac{1}{d''(i)}$ move to any neighbor of v_i.

In other words, at any vertex we move to any its neighbors with probability proportional to the weight of the edge, except for the vertex v_{V+2}, where we stay with probability 1. Moreover, this random walk ignores edge direction, i.e., we can go from a vertex of depth l to a vertex of depth $l-1$.

This way an absorbing random walk is defined; let $\{Y_k\}_{k=0}^{\infty}$ be the corresponding sequence of random variables, where $Y_k = j \in [V+2]$ if after k steps the random walk is at the vertex v_j (i.e., this sequence is the Markov chain, associated to the absorbing random walk).

Let P be the transition matrix for this walk; it has the following block structure:

$$P = \begin{pmatrix} & & & & & & 0 \\ & & & & & & 0 \\ & & Q & & & & 0 \\ & & & & & & \vdots \\ & & & & & & 1-\beta \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix},$$

where Q is a matrix of size $(V+1)\times(V+1)$ which describes the probability of moving from some transient vertex to another.

Define a $(V+1)\times 1$ vector $\mathbf{d}$ as follows:

$$\mathbf{d}[i] = a[i] = \sqrt{d''(i)},\ i \in [A], \quad \mathbf{d}[A+j] = b[j] = \sqrt{d''(A+j)},\ j \in [B],$$

and $\mathbf{d}[V+1] = \sqrt{d''(V+1)} = \mathbf{d}[1]$. Then $\mathrm{diag}(\mathbf{d})^2 Q$ is the adjacency matrix Γ for the graph $\mathcal{G}'$.

Let

$$N = \mathbf{I}_{V+1} + Q + Q^2 + Q^3 + \ldots = (\mathbf{I}_{V+1} - Q)^{-1}$$

be the fundamental matrix of this walk. An entry of the fundamental matrix $N[i,j]$, $i,j \in [V+1]$, equals the expected expected number of visits to the vertex j starting from the vertex i, before being absorbed, i.e.,

$$N[i,j] = \mathbb{E}\left[\sum_{k=0}^{\infty} \mathbb{1}_{\{Y_k=j\}} \,\middle|\, Y_0 = i\right].$$

Notice that $NQ = QN = Q + Q^2 + Q^3 + \ldots = N - \mathbf{I}_{V+1}$. It follows that

$$N[i,j] = \sum_{l=1}^{V+1} N[i,l]Q[l,j] + \delta_{ij} = \sum_{l=1}^{V+1} Q[i,l]N[l,j] + \delta_{ij} \quad \text{for all } i,j \in [V+1], \tag{12}$$

where by δ_{ij} we denote the Kronecker symbol.

6.7 Entries of the fundamental matrix

The purpose of this section is to obtain expressions for entries of the fundamental matrix N. In the next subsection, we will relate those entries to entries of the matrix K from Lemma 15. This will allow us to complete the proof of Lemma 15.

Lemma 19.

$$N[1,V+1] = N[V+1,1] = N[V+1,V+1] = \frac{1}{1-\beta}, \quad N[1,1] = \frac{1}{\beta(1-\beta)}.$$

Proof. We have

$$N[V+1,V+1] = \mathbb{E}\left[\sum_{k=0}^{\infty} \mathbb{1}_{\{Y_k=V+1\}} \,\middle|\, Y_0 = V+1\right].$$

Notice that

$$\sum_{k=0}^{\infty} \mathbb{1}_{\{Y_k=V+1\}} = \sum_{k=0}^{\infty} \mathbb{1}_{\{Y_k=V+1 \text{ and } Y_{k+1}=1\}} + 1,$$

since from the vertex v_{V+1} one either moves to v_{V+2} and gets absorbed or moves to v_1 and returns back to v_{V+1} later. From the linearity of expectation it follows that

$$\begin{aligned} N[V+1,V+1] &= 1 + \sum_{k=0}^{\infty} \mathbb{E}\left[\mathbb{1}_{\{Y_k=V+1 \text{ and } Y_{k+1}=1\}} \,\middle|\, Y_0 = V+1\right] \\ &= 1 + \sum_{k=0}^{\infty} \mathbb{P}\left[Y_k = V+1, Y_{k+1} = 1 \,\middle|\, Y_0 = V+1\right]. \end{aligned}$$

Since

$$\begin{aligned} &\mathbb{P}\left[Y_k = V+1, Y_{k+1} = 1 \,\middle|\, Y_0 = V+1\right] \\ &= \mathbb{P}\left[Y_{k+1} = V+1 \,\middle|\, Y_k = V+1, Y_0 = V+1\right] \cdot \mathbb{P}\left[Y_k = V+1 \,\middle|\, Y_0 = V+1\right] \\ &= \mathbb{P}\left[Y_{k+1} = V+1 \,\middle|\, Y_k = V+1\right] \cdot \mathbb{P}\left[Y_k = V+1 \,\middle|\, Y_0 = V+1\right] \\ &= Q[V+1,1]\,\mathbb{P}\left[Y_k = V+1 \,\middle|\, Y_0 = V+1\right], \end{aligned}$$

we obtain

$$N[V+1,V+1] = 1 + \beta \sum_{k=0}^{\infty} \mathbb{P}\left[Y_k = V+1 \,\middle|\, Y_0 = V+1\right].$$

On the other hand,

$$N[V+1,V+1] = \mathbb{E}\left[\sum_{k=0}^{\infty} \mathbb{1}_{\{Y_k=V+1\}} \,\middle|\, Y_0 = V+1\right] = \sum_{k=0}^{\infty} \mathbb{P}\left[Y_k = V+1 \,\middle|\, Y_0 = V+1\right],$$

hence

$$N[V+1,V+1] = 1 + \beta N[V+1,V+1],$$

from which the equality $N[V+1,V+1] = (1-\beta)^{-1}$ follows.

From (12) it follows that

$$N[V+1,V+1] = \sum_{l=1}^{V+1} Q[V+1,l]N[l,V+1] + 1 = \sum_{l=1}^{V+1} N[V+1,l]Q[l,V+1] + 1.$$

Since $Q[l,V+1]$ and $Q[V+1,l]$ is nonzero only for $l=1$, we have

$$N[V+1,V+1] = \beta N[1,V+1] + 1 = \beta N[V+1,1] + 1.$$

From that we conclude

$$N[V+1,1] = N[1,V+1] = \frac{1}{\beta}\left(N[V+1,V+1]-1\right) = \frac{1}{1-\beta}.$$

Finally, again from (12) we obtain

$$N[V+1,1] = \sum_{l=1}^{V+1} Q[V+1,l]N[l,1] = \beta N[1,1],$$

thus $N[1,1] = (\beta(1-\beta))^{-1}$. □

Define the matrix $\tilde{N} = N\,\mathrm{diag}(\mathbf{d})^{-2}$, then

$$\tilde{N}[i,j] = \frac{N[i,j]}{d''(j)}, \quad i,j \in [V+1].$$

We note that $\tilde{N}$ is a symmetric matrix, since

$$\tilde{N} = (\mathbf{I}_{V+1} - Q)^{-1}\,\mathrm{diag}(\mathbf{d})^{-2} = \left(\mathrm{diag}(\mathbf{d})^2 - \mathrm{diag}(\mathbf{d})^2 Q\right)^{-1}$$

and $\mathrm{diag}(\mathbf{d})^2 Q$ is symmetric. Moreover, from the symmetry we also have $d''(l)Q[l,j] = d''(j)Q[j,l]$ for $j,l \in [V+1]$. Then, since

$$\sum_{l=1}^{V+1} N[i,l]Q[l,j] = \sum_{l=1}^{V+1} \tilde{N}[i,l]d''(l)Q[l,j] = \sum_{l=1}^{V+1} \tilde{N}[i,l]Q[j,l]d''(j),$$

we can rewrite (12) as

$$\tilde{N}[i,j] = \sum_{l=1}^{V+1} \tilde{N}[i,l]Q[j,l] + \frac{\delta_{ij}}{d''(j)} = \sum_{l=1}^{V+1} Q[i,l]\tilde{N}[l,j] + \frac{\delta_{ij}}{d''(j)} \quad \text{for all } i,j \in [V+1]. \tag{13}$$

It follows that for all $i \in [V+1]$, the function $f_i : [V+1] \to \mathbb{R}$ defined by

$$f_i(l) = \tilde{N}[i,l] = \tilde{N}[l,i], \quad l \in [V+1],$$

is harmonic on the set $[V] \setminus \{i\}$ (the function is well defined, since $\tilde{N}[i,l] = \tilde{N}[l,i]$ due to the symmetry of $\tilde{N}$).

In particular, f_{V+1} is harmonic on the set $[V]$, whose boundary is the singleton $\{V+1\}$. Hence f_{V+1} is a constant function. Similarly, f_1 is constant on $[V]$, because it is harmonic on $[2\,..\,V]$, whose boundary is the singleton $\{1\}$.

Corollary 1. *For all $i \in [V+1]$ we have*

$$\tilde{N}[i,V+1] = N[V+1,i] = \frac{1}{d_1}$$

and for all $i \in [V]$ we have

$$\tilde{N}[i,1] = N[1,i] = \alpha^2 + \frac{1}{d_1}.$$

Proof. We already concluded that f_{V+1} is a constant function, i.e., the value

$$f_{V+1}(j) = \tilde{N}[V+1, j] = \tilde{N}[j, V+1]$$

does not depend on j. By Lemma 19,

$$\tilde{N}[V+1, V+1] = \frac{1}{d''(V+1)} N[V+1, V+1] = \frac{1}{d_1 + \alpha^{-2}} \cdot \frac{d_1\alpha^2 + 1}{d_1\alpha^2} = \frac{1}{d_1},$$

and the first claim follows.

The other claim follows from the fact that f_1 is constant on $[V]$ and

$$f_1(1) = \frac{1}{d_1 + \alpha^{-2}} N[1,1] = \frac{1}{d_1 + \alpha^{-2}} \cdot \frac{\left(d_1\alpha^2 + 1\right)^2}{d_1\alpha^2} = \alpha^2 + \frac{1}{d_1}.$$

□

It remains to describe the values of $\tilde{N}[i,j]$ when $2 \le i, j \le V$. For each $i \in [2\,..\,V]$ define ϕ_i to be the unique function which

- is harmonic on $[2\,..\,V] \setminus \{i\}$,
- satisfies $\phi(1) = 1$, $\phi(i) = 0$.

Let $R : [2\,..\,V] \to \mathbb{R}$ be defined by

$$R(i) = \left(\sum_{j : v_j \sim v_i} \phi_i(j) \right)^{-1}.$$

Lemma 20. *For all $i \in [2\,..\,V]$, $j \in [V]$, it holds that*

$$\tilde{N}[i,j] = \alpha^2 + \frac{1}{d_1} + (1 - \phi_i(j))\, R(i).$$

Proof. Fix $i \in [2\,..\,V]$. Let $m = f_i(1)$ (we already have $m = \alpha^2 + {d_1}^{-1}$) and $M = f_i(i) - m$ (M to be described). Then f_i is the unique function which is harmonic on $[2\,..\,V] \setminus \{i\}$ and satisfies the boundary conditions $f_i(1) = m$, $f_i(i) = m + M$. Clearly, $0 \neq M$, since otherwise f_i must be a constant (and therefore harmonic) function, but from (13) it follows that f_i is not harmonic at i.

Define

$$g(j) = \frac{1}{M} \left(f_i(j) - m \right), \quad j \in [V].$$

Then g is harmonic on $[2\,..\,V] \setminus \{i\}$ and satisfies the boundary conditions $g(1) = 0$, $g(i) = 1$. By the Uniqueness Principle, $g \equiv 1 - \phi_i$, since $1 - \phi_i$ satisfies the same conditions. Hence

$$f_i(j) = m + M\left(1 - \phi_i(j)\right), \quad j \in [V].$$

From (13) we have

$$f_i(i) = \frac{1}{d''(i)} \left(1 + \sum_{j : v_j \sim v_i} f_i(j) \right) = m + M + \frac{1}{d''(i)} \left(1 - M \sum_{j : v_j \sim v_i} \phi_i(j) \right).$$

On the other hand, $f_i(i) = m + M$. Taking into account the definition of $R(i)$, we have

$$m + M = m + M + \frac{1}{d''(i)} \left(1 - \frac{M}{R(i)}\right)$$

or $M = R(i)$, which concludes the proof. □

Lemma 21. *Suppose that the original graph $\mathcal{G}$ is a tree. For all vertices $v_i, v_j \in \mathsf{V}$, let $\ell(i,j)$ be the distance from the lowest common ancestor of v_i, v_j to the root v_1.*

Then for all $i, j \in [V]$ it holds that

$$\tilde{N}[i,j] = \alpha^2 + \frac{1}{d_1} + \ell(i,j).$$

Proof. For $i = 1$ the claim follows from Corollary 1. Let $i \in [2\,..\,V]$, then from Lemma 20 we have

$$f_i(j) = \alpha^2 + \frac{1}{d_1} + (1 - \phi_i(j))R(i), \quad j \in [V].$$

Let us show that

$$\phi_i(j) = 1 - \frac{\ell(i,j)}{\ell(i)}, \quad j \in [V]. \tag{14}$$

and

$$R(i) = \ell(i). \tag{15}$$

In (14) the boundary conditions $\phi_i(1) = 1$, $\phi_i(i) = 0$ are satisfied. By the Uniqueness Principle it remains to show that the right-hand side of (14) defines a harmonic function in j on $[2\,..\,V] \setminus \{i\}$. Equivalently, we must show that $\ell(i,\cdot)$ is harmonic on $[2\,..\,V] \setminus \{i\}$.

Fix any $j \in [2\,..\,V] \setminus \{i\}$. There are two cases to consider:

- The vertex v_j is not on the path from the root to v_i; then the lowest common ancestor of v_i and v_j coincides with the lowest common ancestor of v_i and the parent of v_j or the lowest common ancestor of v_i and any child of v_j; hence $\ell(i,j) = \ell(i,k)$ for all vertices v_k, adjacent to v_j.
- The vertex v_j is on the path from the root to v_i. Let v_p be the parent of v_j and v_c be the unique child of v_j which also is on the path from the root to v_i. Then for each vertex $v_k \sim v_j$ we have

$$\ell(i,k) = \begin{cases} \ell(i,j), & k \notin \{p,c\}, \\ \ell(i,j) + 1, & k = c, \\ \ell(i,j) - 1, & k = p. \end{cases}$$

In both cases we obtain

$$\sum_{k : v_k \sim v_j} \frac{\ell(i,k)}{d''(j)} = \ell(i,j),$$

i.e., $\ell(i,\cdot)$ (and thus also $1 - \frac{\ell(i,\cdot)}{\ell(i)}$) is harmonic at every $j \in [2\,..\,V] \setminus \{i\}$. We conclude that (14) holds.

It remains to show (15). From the definition of R,

$$R(i)^{-1} = \sum_{j:v_j\sim v_i} \phi_i(j) = d''(i) - \frac{1}{\ell(i)} \sum_{j:v_j\sim v_i} \ell(i,j).$$

On the other hand, for every vertex $v_j \sim v_i$ we have

$$\ell(i,j) = \begin{cases} \ell(i) - 1, & v_j \text{ is the parent of } v_i, \\ \ell(i), & v_j \text{ is a child of } v_i. \end{cases}$$

Thus

$$\sum_{j:v_j\sim v_i} \ell(i,j) = d''(i)\ell(i) - 1 \quad \text{and} \quad R(i)^{-1} = \ell(i)^{-1}.$$

Now, by combining (14) and (15) with Lemma 20, we obtain the desired equality.

Remark. For another argument why $R(i) = \ell(i)$, see [9, Exercise 9.7]. □

Corollary 2. *Suppose that $\mathcal{G}$ is an arbitrary DAG satisfying the initial assumptions.*

Then for all $i, j \in [V]$ we have

$$\tilde{N}[i,j] - \left(\alpha^2 + \frac{1}{d_1}\right) \le \ell(i).$$

In particular,

$$0 \le \tilde{N}[i,j] - \left(\alpha^2 + \frac{1}{d_1}\right) \le n$$

for all $i, j \in [V]$. Moreover, when $i = 1$ or $j = 1$, the equality $\tilde{N}[i,j] = \alpha^2 + \frac{1}{d_1}$ holds.

Proof. When $i = 1$ or $j = 1$, the assertion holds (Lemma 1). Suppose that $i > 1$. Since $\tilde{N}[i,\cdot]$ is harmonic on $[2\,..\,V] \setminus \{i\}$, it attains its extreme values on the boundary $\{1, i\}$, i.e., it suffices to show the inequality for $j = i$. In view of Lemma 20, this becomes $R(i) \le \ell(i)$.

Let $\mathcal{T}$ be any spanning tree of $\mathcal{G}$ s.t. the shortest path between v_i and the root is preserved, i.e., distance from v_i to v_1 is still $\ell(i)$. By replacing $\mathcal{G}$ with $\mathcal{T}$, the value of $R(i)$ can only increase, since replacing $\mathcal{G}$ with $\mathcal{T}$ corresponds to deleting edges in $\mathcal{G}$ and this operation, by Rayleigh's Monotonicity Law, can only increase the effective resistance $R(i)$. However, Lemma 21 ensures that for the tree $\mathcal{T}$ the value of $R(i)$ equals $\ell(i)$. Thus in the graph $\mathcal{G}$ the value $R(i)$ is upper-bounded by $\ell(i) \le n$. □

6.8 Entries of the matrix K

Now we shall describe the matrix K, where K, L, M_{a}, M_{b}, a and b are defined as in Section 6.1. The adjacency matrix Γ of the graph $\mathcal{G}'$ has the following block structure:

$$\Gamma = \begin{pmatrix} \mathbf{0}_{A,A} & H & \alpha^{-2}\mathbf{e} \\ H^* & & \\ \alpha^{-2}\mathbf{e}^* & & \mathbf{0}_{B+1,B+1} \end{pmatrix},$$

where H is a matrix of size $A \times B$ and $\mathbf{e}$ stands for the column vector of length A, whose first entry is 1 and the remaining entries are 0.

We claim that

Lemma 22.

$$H = M_{\mathsf{a}}^* M_{\mathsf{b}}.$$

Proof. We have to show that for every $i \in [A]$, $j \in [A+1\,..\,V]$ it holds that

$$\Gamma[i,j] = \sum_{e \in [T+1]} M_{\mathsf{a}}[e,i] M_{\mathsf{b}}[e,j]. \tag{16}$$

Notice that $\Gamma[i,j] = 0$ unless $v_i \sim v_j$ (in that case $\Gamma[i,j] = 1$ for $i \in [A]$, $j \in [A+1\,..\,V]$). On the other hand, both $M_{\mathsf{a}}[e,i]$ and $M_{\mathsf{b}}[e,j]$ are simultaneously nonzero iff the edge e is incident both to v_i and v_j.

There are two cases to consider:

1. $v_i \sim v_j$; then $\Gamma[i,j] = 1$ and there is a unique edge $e \in [T+1]$ s.t. $M_{\mathsf{a}}[e,i] \neq 0$ and $M_{\mathsf{b}}[e,j] \neq 0$. Since $i \in [A]$, $j \in [A+1\,..\,V]$, we have $M_{\mathsf{a}}[e,i] = M_{\mathsf{b}}[e,j] = 1$ and (16) holds.
2. $v_i \not\sim v_j$; then $\Gamma[i,j] = 0$ and for each edge $e \in [T+1]$ either $M_{\mathsf{a}}[e,i] = 0$ or $M_{\mathsf{b}}[e,j] = 0$. Again, (16) holds.

□

Denote

$$\mathbf{D} = \operatorname{diag}(\mathbf{d})^{-1} \Gamma \operatorname{diag}(\mathbf{d})^{-1} = \operatorname{diag}(\mathbf{d}) Q \operatorname{diag}(\mathbf{d}),$$

then $\mathbf{D}$ has the following block structure:

$$\mathbf{D} = \begin{pmatrix} \mathbf{0}_{A,A} & \tilde{L} \\ \tilde{L}^* & \mathbf{0}_{B+1,B+1} \end{pmatrix},$$

where $\tilde{L}$ is an $A \times (B+1)$ matrix with the following block structure: $\tilde{L} = \begin{pmatrix} L & \beta \mathbf{e} \end{pmatrix}$. This follows from the fact that $\mathbf{d}$ has the following block structure:

$$\mathbf{d} = \begin{pmatrix} a \\ b \\ a[1] \end{pmatrix},$$

and from the equalities

$$L = \operatorname{diag}(a)^{-1} M_{\mathsf{a}}^* M_{\mathsf{b}} \operatorname{diag}(b)^{-1} = \operatorname{diag}(a)^{-1} H \operatorname{diag}(b)^{-1} \quad \text{and} \quad \beta = (\alpha a[1])^{-2}.$$

Now we can show the following characterization of the matrix $K = (\mathbf{I}_A - LL^*)^{-1}$:

Lemma 23. *Let* $\mathbf{N}$ *stand for the leading* $A \times A$ *principal submatrix of* $\tilde{N}$*, i.e., for the submatrix of* $\tilde{N}$*, formed by the rows indexed by* $[A]$ *and columns indexed by* $[A]$*. Then*

$$K = \operatorname{diag}(a) \left(\mathbf{N} - d_1^{-1} \mathbf{J} \right) \operatorname{diag}(a),$$

where $\mathbf{J}$ *is the* $A \times A$ *all-ones matrix.*

Proof. Since $Q = \operatorname{diag}(\mathbf{d})^{-1}\mathbf{D}\operatorname{diag}(\mathbf{d})$ and $N = (\mathbf{I}_{V+1} - Q)^{-1}$, we have

$$(\mathbf{I}_{V+1} - \mathbf{D})^{-1} = \operatorname{diag}(\mathbf{d})N\operatorname{diag}(\mathbf{d})^{-1} = \operatorname{diag}(\mathbf{d})\tilde{N}\operatorname{diag}(\mathbf{d}). \tag{17}$$

By the block-wise inversion formulas [2, Proposition 2.8.7], $(\mathbf{I}_{V+1} - \mathbf{D})^{-1}$ has the following block structure:

$$(\mathbf{I}_{V+1} - \mathbf{D})^{-1} = \begin{pmatrix} \left(\mathbf{I}_A - \tilde{L}\tilde{L}^*\right)^{-1} & -\left(\mathbf{I}_A - \tilde{L}\tilde{L}^*\right)^{-1}\tilde{L} \\ -\tilde{L}^*\left(\mathbf{I}_A - \tilde{L}\tilde{L}^*\right)^{-1} & \left(\mathbf{I}_{B+1} - \tilde{L}^*\tilde{L}\right)^{-1} \end{pmatrix}.$$

From (17) we conclude that

$$\left(\mathbf{I}_A - \tilde{L}\tilde{L}^*\right)^{-1} = \operatorname{diag}(a)\mathbf{N}\operatorname{diag}(a). \tag{18}$$

From the Sherman-Morrison formula [2, Fact 2.16.3] we have that for an invertible matrix W and a column vector w s.t. $1 + w^*W^{-1}w \neq 0$ the inverse of the updated matrix $W + ww^*$ can be computed as

$$(W + ww^*)^{-1} = W^{-1} - \frac{W^{-1}ww^*W^{-1}}{1 + w^*W^{-1}w}.$$

Take $W = \mathbf{I}_A - \tilde{L}\tilde{L}^*$ and $w = \beta\mathbf{e}$; then

- $\tilde{L}\tilde{L}^* = LL^* + \beta^2\mathbf{e}\mathbf{e}^*$;
- $\mathbf{e}\mathbf{e}^*$ is a matrix of size $A \times A$, whose only nonzero entry is 1 in the first row and column;
- $w^*W^{-1}w = \beta^2 W^{-1}[1,1] = \beta^2 d''(1)\tilde{N}[1,1] = \frac{1}{d_1\alpha^2}$ (by (18));
- $1 + w^*W^{-1}w = \frac{1}{1-\beta} \neq 0$;
- $\mathbf{I}_A - LL^* = W + ww^*$.

We obtain

$$K = (\mathbf{I}_A - LL^*)^{-1} = \left(\mathbf{I}_A - \tilde{L}\tilde{L}^*\right)^{-1}\left(\mathbf{I}_A - \beta^2(1-\beta)\mathbf{e}\mathbf{e}^*W^{-1}\right).$$

Applying (18) yields

$$K = \operatorname{diag}(a)\mathbf{N}U\operatorname{diag}(a),$$

where

$$\begin{aligned} U &:= \operatorname{diag}(a)\left(\mathbf{I}_A - \beta^2(1-\beta)\mathbf{e}\mathbf{e}^*W^{-1}\right)\operatorname{diag}(a)^{-1} \\ &= \mathbf{I}_A - \beta^2(1-\beta)\mathbf{e}\mathbf{e}^*\operatorname{diag}(a)W^{-1}\operatorname{diag}(a)^{-1} \\ &= \mathbf{I}_A - \beta^2(1-\beta)\mathbf{e}\mathbf{e}^*\operatorname{diag}(a)\mathbf{N} \\ &= \mathbf{I}_A - \beta^2(1-\beta)d''(1)\mathbf{e}\mathbf{e}^*\mathbf{N}. \end{aligned}$$

Here we have used the fact that $\mathbf{e}\mathbf{e}^*$ and $\operatorname{diag}(a)$ commute and $\mathbf{e}\mathbf{e}^*\operatorname{diag}(a)^2 = d''(1)\mathbf{e}\mathbf{e}^*$. It follows that

$$\mathbf{N}U = \mathbf{N} - \beta^2(1-\beta)d''(1)\mathbf{N}\mathbf{e}\mathbf{e}^*\mathbf{N}.$$

Furthermore, the row vector $\mathbf{e}^*\mathbf{N}$ equals the first row of $\mathbf{N}$ (and $\mathbf{N}$ is a symmetric matrix). All elements of the first row of $\mathbf{N}$ are equal to $\alpha^2 + d_1^{-1}$, hence

$$(\mathbf{N}\mathbf{e})\,(\mathbf{e}^*\mathbf{N}) = \left(\alpha^2 + d_1^{-1}\right)^2 \mathbf{J}.$$

It is straightforward to check that

$$\begin{aligned}
&\beta(1-\beta)d''(1)\left(\alpha^2 + d_1^{-1}\right) = 1,\\
&\beta^2(1-\beta)d''(1)\left(\alpha^2 + d_1^{-1}\right)^2 = \beta\left(\alpha^2 + d_1^{-1}\right) = d_1^{-1},
\end{aligned}$$

thus

$$\mathbf{N}U = \mathbf{N} - d_1^{-1}\mathbf{J} \quad \text{and} \quad K = \operatorname{diag}(a)\left(\mathbf{N} - d_1^{-1}\mathbf{J}\right)\operatorname{diag}(a).$$

□

We now continue with the proof of Lemma 14, restated here for convenience.

Lemma 14. *For all $i, j \in [A]$ the following inequalities hold:*

$$\alpha^2 a[i]a[j] \le K[i,j] \le \left(\alpha^2 + n\right) a[i]a[j].$$

Moreover, when $i = 1$ or $j = 1$, we have $K[i,j] = \alpha^2 a[i]a[j]$.

Proof. From Lemma 23 we have

$$K[i,j] = \left(\tilde{N}[i,j] - \frac{1}{d_1}\right) a[i]a[j], \quad i,j \in [A].$$

On the other hand, from Corollary 2 we have

$$\alpha^2 \le \tilde{N}[i,j] - \frac{1}{d_1} \le \alpha^2 + n, \quad i,j \in [A],$$

and $\tilde{N}[i,j] = \alpha^2 + \frac{1}{d_1}$ whenever $i = 1$ or $j = 1$. This proves the claim. □

7 Proofs for AND-OR tree evaluation

Proof of Lemma 10. **Correctness.** Children of a vertex v gets added to $\mathcal{T}'$ if the tree size estimate for $\mathcal{T}(v)$ is at least $\frac{2m}{3}$. If the tree size estimation in step 1 is correct, we have the following:

- if the actual size is at least m, the estimate is at least $(1-\delta)m = \frac{2m}{3}$;
- if the actual size is less than $\frac{m}{2}$, the estimate is less than $(1+\delta)\frac{m}{2} = \frac{2m}{3}$.

This means, if all the estimates are correct, then $\mathcal{T}'$ is a correct m-heavy element subtree.

To show that all the estimates are correct with probability at least $1-\epsilon$, we have to bound the number of calls to tree size estimation. We show

Lemma 24. *An m-heavy element subtree $\mathcal{T}'$ contains at most $n\frac{6T}{m}$ vertices.*

Proof. On each level, $\mathcal{T}'$ has at most $\frac{2T}{m}$ vertices x with $|\mathcal{T}(x)| \geq \frac{m}{2}$. Since the depth of $\mathcal{T}$ (and, hence, $\mathcal{T}'$) is at most n, the total number of such vertices is at most $\frac{2Tn}{m}$. For each such x, $\mathcal{T}'$ may also contain its children with $|\mathcal{T}(x)| < \frac{m}{2}$. Since each non-leaf x has two children, the number of such vertices is at most $\frac{4Tn}{m}$ and the total number of vertices is at most $\frac{6Tn}{m}$. □

Since the algorithm makes one call to tree size estimation for each x in $\mathcal{T}'$ and each call to tree size estimation is correct with probability at least $1 - \epsilon' = 1 - \frac{m}{6nT}\epsilon$, we have that all the calls are correct with probability at least $1 - \epsilon$.

Running time. Since the formula tree is binary, we have $d = O(1)$. Also, $\delta = \frac{1}{3}$ is a constant, as well. Therefore, each call to tree size estimation uses

$$O\left(\sqrt{nm}\log^2\frac{1}{\epsilon'}\right)$$

queries. Since tree size estimation is performed only for vertices that get added to $\mathcal{T}'$, it is performed at most $\frac{6Tn}{m}$ times. Multiplying the complexity of one call by $\frac{6Tn}{m}$ and substituting $\epsilon' = \frac{m}{6nT}\epsilon > \frac{\epsilon}{6T^2}$ gives the complexity of

$$O\left(\frac{n^{1.5}T}{\sqrt{m}}\log^2\frac{T}{\epsilon}\right).$$

□

Analysis of the main algorithm: correctness.

Lemma 25. *If* **Unknown-evaluate** *is used as a query, it performs a transformation* $|\psi_{start}\rangle \to |\psi\rangle$ *where* $|\psi\rangle$ *satisfies*

$$\|\psi - (-1)^T\psi_{start}\| \leq 2\sqrt{\epsilon}$$

Proof. We use induction over $i \in \{0, 1, \ldots, c\}$, with $i = 0$ being queries to one variable (which are invoked by **Unknown-evaluate**$(r, 1, \epsilon)$ in the 3rd step when $i = 1$) and $i = 1, \ldots, c$ being calls to **Unknown-evaluate** with the respective value of i.

For the base case ($i = 0$), a query to a variable produces the transformation $|i\rangle \to (-1)^{x_i}|i\rangle$. In this case, $T = x_i$, so, this is exactly the correct transformation.

For the inductive step ($i \geq 1$), we first assume that, instead of calls to **Unknown-evaluate** at the leaves, we have perfect queries with no error. Let $|\psi_{ideal}\rangle$ be the final state under this assumption. We first bound $\|\psi_{ideal} - (-1)^T\psi_{start}\|$ and then bound the difference between $|\psi_{ideal}\rangle$ and the actual final state $|\psi\rangle$.

Let $|\psi'\rangle = \sum_{\mathcal{T}',x} \alpha_{\mathcal{T}',x}|\mathcal{T}', x\rangle\left|\psi_{\mathcal{T}',x}\right\rangle$ be the state after the first three steps, with $\mathcal{T}'$ being the subtree obtained in the 1st or the 2nd step, x being the result obtained at the 3rd step and $\left|\psi_{\mathcal{T}',x}\right\rangle$ being all the other registers containing intermediate information. We express $|\psi'\rangle = |\psi_1\rangle + |\psi_2\rangle + |\psi_3\rangle$ where

- $|\psi_1\rangle$ contains terms where $\mathcal{T}'$ is a valid m-heavy element subtree and x is the correct answer;
- $|\psi_2\rangle$ contains terms where $\mathcal{T}'$ is a valid m-heavy element subtree but x is not the correct answer;
- $|\psi_3\rangle$ contains all the other terms.

By the correctness guarantees for steps 2 and 3, we have $\|\psi_2\| \leq \sqrt{\epsilon/5}$ and $\|\psi_3\| \leq \sqrt{\epsilon/5}$.

After the phase flip in step 4, the state becomes $|\psi''\rangle = (-1)^T|\psi_1\rangle + |\psi_2'\rangle + |\psi_3'\rangle$ with $|\psi_2'\rangle$ and $|\psi_3'\rangle$ consisting of terms from $|\psi_2\rangle$ and $|\psi_3\rangle$, with some of their phases flipped. Hence,

$$\|\psi'' - (-1)^T\psi'\| \leq \|\psi_2' + \psi_3'\| + \|\psi_2 + \psi_3\| \leq \|\psi_2'\| + \|\psi_3'\| + \|\psi_2\| + \|\psi_3\| \leq 4\sqrt{\frac{\epsilon}{5}}.$$

Reversing the first three steps maps $|\psi'\rangle$ back to $|\psi_{start}\rangle$ and $|\psi''\rangle$ to $|\psi_{ideal}\rangle$. Hence, we have the same estimate for $\|\psi_{ideal} - (-1)^T\psi_{start}\|$.

We now replace queries by applications of **Unknown-evaluate**. Let $t = O(\sqrt{sn}) = O(s)$ be the number of queries. Let $|\phi_i\rangle$ be the final state of the algorithm if the first i queries use the perfect query transformation and the remaining queries are implemented using **Unknown-evaluate**. Then, $|\psi_{ideal}\rangle = |\phi_t\rangle$ and $|\psi\rangle = |\phi_0\rangle$ and we have

$$\|\psi - \psi_{ideal}\| = \|\phi_0 - \phi_t\| \leq \sum_{j=0}^{t-1} \|\phi_j - \phi_{j+1}\| \leq \frac{2s\sqrt{\epsilon}}{s^{3/2}} = o(\sqrt{\epsilon}),$$

with the second inequality following from the inductive assumption. (The only difference between $|\phi_j\rangle$ and $|\phi_{j+1}\rangle$ is that, in the first case, we apply a perfect query transformation in the $(j+1)^{\text{st}}$ query and, in the second case, we apply **Unknown-evaluate**$(v, i-1, \epsilon/s^3)$ instead. The distance between the states resulting from these two transformations can be bounded by $\frac{2\sqrt{\epsilon}}{s^{3/2}}$ by the inductive assumption.) Therefore,

$$\|\psi - (-1)^T\psi_{start}\| \leq \|\psi_{ideal} - (-1)^T\psi_{start}\| + \|\psi - \psi_{ideal}\| \leq \frac{4}{\sqrt{5}}\sqrt{\epsilon} + o(\sqrt{\epsilon}) < 2\sqrt{\epsilon}.$$

This concludes the proof. □

For the case when **Unknown-evaluate** is used to obtain the final answer, let $|\psi'\rangle$ be the final state if, instead of calls to **Unknown-evaluate** at the next level, we had perfect queries and let $|\psi\rangle$ be the actual final state of the algorithm. We express $|\psi\rangle = |\psi_{cor}\rangle + |\psi_{inc}\rangle$ where $|\psi_{cor}\rangle$ ($|\psi_{inc}\rangle$) is the part of the state where the algorithm outputs correct (incorrect) answer. Let $|\psi'\rangle = |\psi'_{cor}\rangle + |\psi'_{inc}\rangle$ be a similar decomposition for $|\psi'\rangle$. Then, $\|\psi'_{inc}\| \leq \|\psi_2\| + \|\psi_3\| \leq \frac{2}{\sqrt{5}}\sqrt{\epsilon}$ and

$$\|\psi_{inc}\| \leq \|\psi'_{inc}\| + \|\psi_{inc} - \psi'_{inc}\| \leq \|\psi'_{inc}\| + \|\psi - \psi'\| \leq \frac{2}{\sqrt{5}}\sqrt{\epsilon} + o(\sqrt{\epsilon}) < \sqrt{\epsilon}.$$

The probability of **Unknown-evaluate** outputting an incorrect answer is $\|\psi_{inc}\|^2 < \epsilon$.

Analysis of the main algorithm: running time.

Lemma 26. *The number of queries made by* **Unknown-evaluate**(r, i, ϵ) *is of an order*

$$O\left(n^i\sqrt{T_iT^{1/c}}\left(\log T_i + \log\frac{1}{\epsilon}\right)^i\right).$$

Proof. Generating $\mathcal{T}'$ takes $O(T_1)$ steps if $i = 1$ and $O(\frac{n^{1.5}T_i}{\sqrt{T_{i-1}}}\log^2\frac{T_i}{\epsilon})$ steps if $i > 1$. Since $\frac{T_i}{\sqrt{T_{i-1}}} = \sqrt{T_iT^{1/c}}$, this is at most the bound in the statement of the lemma.

In the next step, the algorithm calls **Unknown-evaluate** for $O(\sqrt{ns}\log\frac{1}{\epsilon})$ subtrees where s is the size of $\mathcal{T}'$. Since $s = O\left(n\frac{T_i}{T_{i-1}}\right)$, this means $O\left(n\frac{\sqrt{T_i}}{\sqrt{T_{i-1}}}\log\frac{1}{\epsilon}\right)$ calls of **Unknown-evaluate**. For each of them, the number of queries is

$$O\left(n^{i-1}\sqrt{T_{i-1}T^{1/c}}\left(\log T_{i-1}+\log\frac{1}{\epsilon'}\right)^{i-1}\right) = O\left(n^{i-1}\sqrt{T_{i-1}T^{1/c}}\left(\log T_i+\log\frac{1}{\epsilon}\right)^{i-1}\right).$$

Multiplying the number of calls to **Unknown-evaluate** with the complexity of each call gives the claim. □

We now show how Lemma 26 implies Theorem 9. If $i = c$, the expression of Lemma 26 is equal to

$$O\left(n^c\sqrt{T^{1+1/c}}\left(\log T+\log\frac{1}{\epsilon}\right)^c\right) = O\left(T^{\frac{1}{2}+\frac{1}{2c}+o(1)}\right).$$

Since c can be chosen arbitrarily large, we can achieve $O(T^{1/2+\delta})$ for arbitrarily small $\delta > 0$. The total running time is $O(\log T) = O(T^{o(1)})$ times the number of queries.

8 Conclusion

Search trees of unknown structure (which can be discovered by local exploration) are commonly used in classical computer science. In this paper, we constructed three quantum algorithms for such trees: for estimating size of a tree, for finding whether a tree contains a marked vertex (improving over an earlier algorithm by Montanaro) and for evaluating an AND-OR formula described by a tree of unknown structure.

Some of possible directions for future study are:

1. **Space-efficient algorithm for AND-OR formula evaluation?** Our algorithm for evaluating AND-OR formulas in Section 3.5 uses a substantial amount of memory to store the heavy element subtrees. In contrast, the algorithm of [1] for evaluating formulas of known structure only uses $O(\log T)$ qubits of memory. Can one construct an algorithm for evaluating formulas of unknown structure with time complexity similar to our algorithm but smaller space requirements?

2. **Speeding up other methods for solving NP-complete problems.** Backtracking is used by SAT solvers and other algorithms for NP-complete problems. Our quantum algorithm for backtracking provides an almost quadratic quantum improvement over those algorithms. What other methods for solving NP-complete problems have faster quantum counterparts?

3. **Other parameter estimation problems.** The tree size estimation problem can be viewed as a counterpart of quantum counting, in a more difficult setting. What other problems about estimating size (or other parameters) of combinatorial structures would be interesting and what would they be useful for?

Acknowledgements

The authors would like to thank Mark Goh for his valuable comments on a previous version of the paper, in particular, for pointing out a mistake in the proof of Lemma 16. The corrected version requires adjusting the bound on $\left|\hat{\theta}-\theta\right|$ in Lemma 5 and choosing the parameter value $\delta_{min}=\frac{\delta^{1.5}}{24\sqrt{3nT_0}}$ (instead of $\delta_{min}=\frac{\delta^{1.5}}{4\sqrt{3nT_0}}$) in Algorithm 1.

This work has been supported by the ERC Advanced Grant MQC and Latvian State Research programme NexIT project No.1.

References

[1] A. Ambainis, A. Childs, B. Reichardt, R. Špalek, S. Zhang. Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. *SIAM Journal on Computing*, 39(6): 2513-2530, 2010. DOI:10.1137/080712167. Also arXiv:quant-ph/0703015.

[2] D. S. Bernstein. *Matrix mathematics. Theory, facts, and formulas*, 2nd edition. Princeton University Press, Princeton, 2009.

[3] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394-397, 1962. DOI:10.1145/368273.368557.

[4] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201-215, 1960. DOI:10.1145/321033.321034.

[5] E. Farhi, J. Goldstone, S. Gutman, A quantum algorithm for the Hamiltonian NAND tree. *Theory of Computing*, 4(1): 169-190, 2008. Available at `http://theoryofcomputing.org/articles/v004a008`. Also arXiv:quant-ph/0702144.

[6] L. K. Grover. A fast quantum mechanical algorithm for database search. *Proceedings of the ACM Symposium on the Theory of Computing (STOC'1996)*, pp. 212-219. DOI:10.1145/237814.237866. Also arXiv:quant-ph/9605043.

[7] R. A. Horn, C. R. Johnson. *Matrix analysis*, 2nd edition. Cambridge University Press, Cambridge, 2012. DOI:10.1017/CBO9781139020411.

[8] G. F. Lawler, L. N. Coyle. *Lectures on contemporary probability*, volume 2 of *Student Mathematical Library / IAS/Park City Mathematical Subseries*. American Mathematical Society, Providence, 1999. DOI:10.1090/stml/002

[9] D. A. Levin, Y. Peres, E. L. Wilmer. *Markov chains and mixing times. With a chapter by James G. Propp and David B. Wilson.* American Mathematical Society, Providence, 2009. Available at `http://www.ams.org/books/mbk/058/`. Also `http://pages.uoregon.edu/dlevin/MARKOV/markovmixing.pdf`.

[10] L. Lovász. Random walks on graphs: A survey. In *Combinatorics, Paul Erdös is eighty* (D. Miklós, V. T. Sós, T. Szőnyi, eds.), volume 2, pp. 353–397, János Bolyai Mathematical Society, Budapest, 1996.

[11] R. Lyons, Y. Peres. *Probability on trees and networks*. Cambridge University Press, New York, 2016. Available at `http://www.cambridge.org/9781107160156`. Also `http://pages.iu.edu/~rdlyons/`.

[12] A. Montanaro. Quantum walk speedup of backtracking algorithms. arXiv:1509.02374.

[13] R. Pemantle. Uniform random spanning trees. In *Topics in contemporary probability and its applications* (J.L. Snell, ed.), pp. 1–54, CRC Press, Boca Raton, 1995. Also arXiv:math/0404099.

[14] B. Reichardt. Faster quantum algorithm for evaluating game trees. *Proceedings of SODA'2011*, pp. 546-559. Available at `http://dl.acm.org/citation.cfm?id=2133079`. Also arXiv:0907.1623.

[15] M. Szegedy. Quantum speed-up of Markov chain based algorithms. *Proceedings of FOCS'2004*, pp. 32-41. DOI:10.1109/FOCS.2004.53