48th Annual IEEE Symposium on Foundations of Computer Science

Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer

Andris Ambainis[*,†] ambainis@math.uwaterloo.ca
Andrew M. Childs[†,‡] amchilds@uwaterloo.ca
Ben W. Reichardt[‡] breic@caltech.edu
Robert Špalek[§] spalek@eecs.berkeley.edu
Shengyu Zhang[‡] shengyu@caltech.edu

Abstract

For any AND-OR formula of size N, there exists a bounded-error $N^{\frac{1}{2}+o(1)}$-time quantum algorithm, based on a discrete-time quantum walk, that evaluates this formula on a black-box input. Balanced, or "approximately balanced," formulas can be evaluated in $O(\sqrt{N})$ queries, which is optimal. It follows that the $(2-o(1))$th power of the quantum query complexity is a lower bound on the formula size, almost solving in the positive an open problem posed by Laplante, Lee and Szegedy.

1 Introduction

Consider a formula φ on N inputs $x_1, \ldots, x_N$, using the gate set S either $\{\text{AND}, \text{OR}, \text{NOT}\}$ or equivalently $\{\text{NAND}\}$. That is, the formula φ corresponds to a tree where each internal node is a gate from S on its children. If the same variable is fed into different inputs of φ, we treat each occurrence separately, so that N counts variables with multiplicity. The variables x_i are accessed by querying a quantum oracle, which we can take to be the unitary operator $O_x : |b,i\rangle \mapsto (-1)^{bx_i}|b,i\rangle$, for $b \in \{0,1\}$ and $i \in \{1,\ldots,N\}$ the control qubit and query index, respectively. We will show:

Theorem 1. *After efficient preprocessing (i.e., preprocessing taking $\mathrm{poly}(N)$ steps) of the formula φ independent of x, $\varphi(x)$ can be evaluated with error $< 1/3$ using $N^{\frac{1}{2}+o(1)}$ time and queries to O_x.*

Our algorithm is inspired by the recent $N^{\frac{1}{2}+o(1)}$-time algorithm of Farhi, Goldstone and Gutmann [11] for the case in which $S = \{\text{NAND}\}$, each NAND gate in φ has exactly two inputs and φ is *balanced*—i.e., $N = 2^n$ and φ has depth n. Our algorithm requires no preprocessing in this special case of a balanced binary NAND tree. For a balanced, or even an "approximately balanced" NAND tree, our algorithm requires only $O(\sqrt{N})$ queries (Theorem 8), which is optimal. The correctness of our algorithm will turn on spectral analysis of a Hamiltonian similar to that introduced by Farhi et al., except in general weighted according to the formula's imbalances.

Our algorithm (almost) solves in the positive the open problem posed by Laplante, Lee and Szegedy [16], whether the quantum query complexity squared is a lower bound on the formula size. Theorem 1 implies that the formula size of a function f is at least $Q_2(f)^{2-o(1)}$. Note also that evaluating an AND-OR tree is the decision version of evaluating a MIN-MAX tree, and the latter can be solved using any algorithm for the former with a log factor slow down.

History of the problem

Grover showed in 1996 [12, 13] how to search a general unstructured database of size N, represented by a black-box oracle function, in $O(\sqrt{N})$ oracle queries and $O(\sqrt{N}\log\log N)$ time on a quantum computer. His search algorithm can be used to compute the logical OR of N bits in the same time; simply search for a 1 in the input string. Since Grover search can be used as a subroutine, even within another instance of Grover search, one can speed up the computation of more general logical formulas. For example, a two-level AND-OR tree (with one AND gate of fan-in $\sqrt{N}$ and $\sqrt{N}$ OR gates of the same fan-in as its children) can be computed in $O(\sqrt{N}\log N)$ queries. Here

[*]Department of Computer Science, University of Latvia.
[†]Department of Combinatorics & Optimization and Institute for Quantum Computing, University of Waterloo. A.A. supported by CIAR, ARO, NSERC, MITACS and IQC University Professorship.
[‡]Institute for Quantum Information, California Institute of Technology. Supported by NSF Grant PHY-0456720 and ARO Grant W911NF-05-1-0294. B.R. also supported by NSF Grant CCF-0524828.
[§]University of California, Berkeley. Supported by NSF Grant CCF-0524837 and ARO Grant DAAD 19-03-1-0082. Work conducted in part while visiting Caltech.

0272-5428/07 $25.00 © 2007 IEEE
DOI 10.1109/FOCS.2007.57

the log factor comes from amplifying the success probability of the inner quantum search to be polynomially close to one, so that the total error is at most constant. By iterating the same argument, regular AND-OR trees of depth d can be evaluated with constant error in time $\sqrt{N} \cdot O(\log N)^{d-1}$ [7].

Høyer, Mosca and de Wolf [14] showed that Grover search can be applied even if the input variables are noisy, so the log factor is not necessary. Consequently, a depth-d AND-OR tree can be computed in $O(\sqrt{N} \cdot c^d)$ queries, where c is a constant that comes from their algorithm. It follows that constant-depth AND-OR trees can be computed in $O(\sqrt{N})$ queries. Unfortunately, their algorithm is too slow for the balanced binary AND-OR tree of depth $\log_2 N$ (although it does give some speedup for sufficiently large constant fan-ins).

Classically, one can compute the value of a balanced binary AND-OR tree with zero error in expected time $O(N^{\log_2[(1+\sqrt{33})/4]}) = O(N^{0.754})$ [19, 17] using a technique called alpha-beta pruning, and this is optimal even for bounded-error algorithms [18].

For a long time, no quantum algorithm was known that performed better than the classical zero-error algorithm, despite the fact that the best known lower bound, from the adversary method, is only $\Omega(\sqrt{N})$ [3]. Very recently, Farhi, Goldstone and Gutmann [11] presented a groundbreaking quantum algorithm for the balanced case, based on continuous-time quantum walks. Their algorithm runs in time $O(\sqrt{N})$ in an unconventional, continuous-time query model. Childs, Cleve, Jordan and Yeung [8] shortly after pointed out that this algorithm can be discretized into the conventional oracle query model with a small slowdown, to run in time $N^{\frac{1}{2}+o(1)}$.

This paper is a merged version of technical reports [9, 2].

2 Summary of results and methods

We design an algorithm whose running time depends on the structure of the NAND tree. For arbitrary balanced trees, the algorithm uses $O(\sqrt{N})$ queries.

More generally, if the fanin of each NAND gate in our formula is bounded by a constant, our algorithm uses $O(\sqrt{N}d)$ queries, where d is the depth of the formula. If the depth is large, we use a rebalancing procedure of [6, 5] to construct an equivalent formula with depth $2^{O(\sqrt{\log N})}$. This implies that any NAND formula of size N can be evaluated using $N^{1/2+O(1/\sqrt{\log N})}$ queries.

Idea of the algorithm

Our algorithm can be described in terms of discrete-time quantum walks. With any NAND formula φ, we associate a tree $T = T(\varphi)$. Figure 1 gives an example for the case of the balanced binary tree of depth 3.

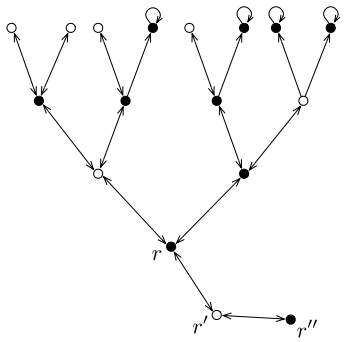

Figure 1: The balanced binary NAND formula of depth 3, $\varphi = \big((x_1 \barwedge x_2) \barwedge (x_3 \barwedge x_4)\big) \barwedge \big((x_5 \barwedge x_6) \barwedge (x_7 \barwedge x_8)\big)$—where $a \barwedge b = \overline{(a \wedge b)}$—is represented by the balanced subtree rooted at r. The input is fed in at the leaves (top row), and internal vertices are evaluated as NAND gates on their children. Here a vertex v is filled or not according to whether its evaluation $\barwedge(v)$ is 1 or 0, respectively; in this example, the input is $x = 00010111$ and overall, $\varphi(x) = \barwedge(r) = 1$. Below r, we have additionally attached two vertices, r' and r''. Modify the classical uniform random walk on this tree by marking leaf vertices evaluating to 1 as probability sinks, and adding a certain bias to the transition probabilities at r'. Then the corresponding quantum walk can be used to evaluate the formula in $O(\sqrt{N})$ time.

Consider the classical uniform random walk on this tree; at vertex v with degree $d(v)$, flip an unbiased $d(v)$-sided coin to decide which neighbor to step to next. (The coin at r' will have a certain bias, as will all coins in the case of an unbalanced tree.) Incorporate the input into the walk by making all vertices evaluating to 1 into probability sinks. The corresponding discrete-time quantum walk U works on a vertex register and a coin register. Instead of randomizing the coin register between steps, a Grover diffusion operator is applied to the coin. This quantum walk U, started at r'', can be used with phase estimation to evaluate a general NAND formula in only $N^{\frac{1}{2}+o(1)}$ time, or $O(\sqrt{N})$ queries in the balanced or approximately balanced case. The general algorithm is described in Section 7, while Figure 2 presents the full algorithm for the balanced case.

The correctness proof uses Szegedy's correspondence between classical random walks and discrete-time quantum walks (Theorem 6). Note that Szegedy's formulation can also be viewed as establishing a correspondence between discrete- and continuous-time quantum walks (Remark 2). In particular, the spectrum and eigenvectors of the unitary operator describing the discrete-time quantum walk can be related to those of the Hamiltonian for a continuous-time quantum walk on a closely related tree. We use this by first designing a continuous time quantum walk algorithm and then converting it into a discrete-time algorithm.

1. **Initialization.** Let $T = 320\lfloor\sqrt{N}\rfloor$. Prepare three quantum registers in the state

$$\Big(\frac{1}{\sqrt{T}}\sum_{t=0}^{T-1}(-i)^t|t\rangle\Big)\otimes|r''\rangle|\text{left}\rangle \ .$$

The first register is a counter for quantum phase estimation, the second register holds a vertex index, and the third register is a qutrit "coin" holding 'down', 'left' or 'right' in this order.

2. **Quantum walk.** If the first register is $|t\rangle$, perform t steps of the following discrete-time quantum walk U. Denote the last two registers by $|v\rangle|c\rangle$.

   - **Diffusion step.**
     (a) If v is a leaf, apply a phase flip $(-1)^{x_v}$ using one controlled call to the input oracle.
     (b) If v is an internal degree-three vertex, apply the following diffusion operator on coin $|c\rangle$:

$$\text{Reflection}_{|u\rangle} = 2|u\rangle\langle u| - 1 \ ,$$

where $|u\rangle = \frac{1}{\sqrt{3}}(|\text{down}\rangle + |\text{left}\rangle + |\text{right}\rangle)$.
     (c) If $v = r'$, apply the following diffusion operator on $|c\rangle$:

$$\text{Reflection}_{|u'\rangle} = 2|u'\rangle\langle u'| - 1 \ ,$$

where $|u'\rangle = \frac{1}{\sqrt[4]{N}}|\text{down}\rangle + \sqrt{1-\frac{1}{\sqrt{N}}}|\text{left}\rangle$.
     (d) If $v = r''$, do nothing.
   - **Walk step.**
     (a) If $c =$ 'down', then walk down to the parent of v and set c to either 'left' or 'right' depending on which child v is.
     (b) If $c \in \{$'left', 'right'$\}$, then walk up to the corresponding child of v and set c to 'down'.

   Note that the walk step operator is a permutation that simply flips the direction of each oriented edge.

3. **Quantum phase estimation.** Apply the inverse quantum Fourier transform (modulo T) on the first register and measure it in the computational basis. Return 0 if and only if the outcome is 0 or $\frac{T}{2}$.

Figure 2: An optimal quantum algorithm to evaluate the balanced binary NAND formula using $O(\sqrt{N})$ queries. The algorithm runs quantum phase estimation on top of the quantum walk of Figure 1.

Organization

The presentation of the algorithm and the proof of its correctness are organized as follows. Section 3 defines and puts weights on the tree $T(\varphi)$, where the weights of edges to the leaves depend on the input x. Section 4 shows that when $\varphi(x) = 0$, there exists a zero-energy eigenstate (i.e., eigenvector with eigenvalue zero) of the weighted adjacency matrix of $T(\varphi)$ and a short tail, having substantial overlap with a known initial state. Conversely, if $\varphi(x) = 1$, then any zero-energy eigenstates can be neglected, as they have no overlap on the initial state. In the case $\varphi(x) = 1$, Section 5 shows that eigenvectors with small nonzero eigenvalues can also be neglected. This is argued by connecting the NAND formula's evaluation to the ratio of amplitudes from a child to its parent. We then construct an algorithm using the observations of Sections 4 and 5. Section 6 reviews Szegedy's theorem relating the spectrum and eigenvectors of a discrete-time quantum walk to those of the Hamiltonian for a continuous-time quantum walk. We apply this relationship in Section 7 to show that phase estimation on a certain discrete-time quantum walk can be used to evaluate φ.

We conclude the paper by describing some applications to evaluating iterated functions in Section 8, and by presenting some open problems in Section 9.

3 Weighted NAND formula tree

Consider a NAND formula of size N—i.e., on N variables, counting multiplicity. (The NAND gate on inputs $y_1, \ldots, y_k \in \{0,1\}$ evaluates to $1 - \prod_{i=1}^k y_i$. In particular, the NAND gate on a single input is simply the NOT gate.)

Represent the formula φ by a rooted tree $T = T(\varphi)$, in which the leaves correspond to variables, and other vertices correspond to NAND gates on their children. (Because φ is a formula, not a circuit, each gate has fan-out one, so there are no loops in the associated graph.) Attach below the root r a "tail" of two vertices r' and r'' as in Figure 1.

Definition 1. *For a vertex v, let s_v be the number of inputs of the subformula under v, counting multiplicity (in particular, $s_r = s_{r'} = s_{r''} = N$). Let $\overline{\wedge}(v)$ denote the value of the subformula at v ($\overline{\wedge}(r) = \overline{\wedge}(r'') = \varphi(x)$).*

Definition 2. *Let H be a symmetric, weighted adjacency matrix of the graph consisting of T and the attached tail:*

$$H|v\rangle = h_{pv}|p\rangle + \sum_c h_{vc}|c\rangle \ , \qquad (1)$$

where p is the parent of v and the sum is over v's children. (If v has no parent or no children, the respective terms are

zero.) The edge weights depend on the structure of the tree, and are given by

$$h_{pv} = \left(\frac{s_v}{s_p}\right)^{1/4} , \tag{2}$$

with two exceptions:

1. *If a leaf v evaluates to $\overline{\wedge}(v) = 1$, set $h_{pv} = 0$, i.e., effectively remove the edge (p, v) by setting its weight to zero.*

2. *Set $h_{r''r'} = 1/(\sqrt{\sigma_-(r)}N^{1/4})$ (see Definition 3).*

Definition 3. *To track error terms through the analysis, it will be helpful to define*

$$\begin{aligned} \sigma_-(v) &= \max_{\xi} \sum_{\substack{w\in\xi:\\ \overline{\wedge}(w)=0}} \frac{1}{\sqrt{s_w}} \\ \sigma_+(v) &= \max_{\xi} \sum_{\substack{w\in\xi:\\ \overline{\wedge}(w)=1}} s_w \end{aligned} \tag{3}$$

with the maximum in each case taken over all paths ξ from v up to a leaf. Letting d_r be the depth of r, clearly $\sigma_-(v) \leq \sigma_-(r) \leq d_r$ and $\sigma_+(v) \leq \sigma_+(r) \leq Nd_r$.[1] We call formula φ "approximately balanced" if $\sigma_-(r) = O(1)$, $\sigma_+(r) = O(N)$ and $\|H\| = O(1)$, where $\|\cdot\|$ denotes the spectral norm.

The simplest example of an approximately balanced tree is the balanced binary tree. More generally, a tree is approximately balanced if s_v decreases sufficiently rapidly from the root toward the leaves (see Section 7).

Our algorithm will depend on the following properties of H, which we will prove in the next two sections:

Theorem 2. *If $\varphi(x) = 0$, then there exists a zero-energy eigenvector $|a\rangle$ of H with $\||a\rangle\| = 1$ and overlap $|\langle r''|a\rangle| \geq 1/\sqrt{2}$. If $\varphi(x) = 1$, then every eigenvector with support on r' or r'' has corresponding eigenvalue at least $\frac{1}{9\sigma_-(r)\sqrt{\sigma_+(r)}}$ in absolute value.*

Remark 1. *For a leaf v evaluating to 0, it is sufficient that h_{pv} satisfies $h_{pv} \geq 1/s_p^{1/4}$.*

One can also verify that Theorem 2 holds if the weights h_{pv} are defined by, for an arbitrary fixed β, $h_{pv} = {s_v}^{\beta}/{s_p}^{\frac{1}{2}-\beta}$ and $h_{r''r'} = 1/(\sqrt{\sigma_-(r)}N^{\frac{1}{2}-\beta})$ with $\sigma_-(v) = \max_{\xi} \sum_{w\in\xi:\,\overline{\wedge}(w)=0} {s_w}^{-2\beta}$. We fix $\beta = 1/4$ to simplify notation.

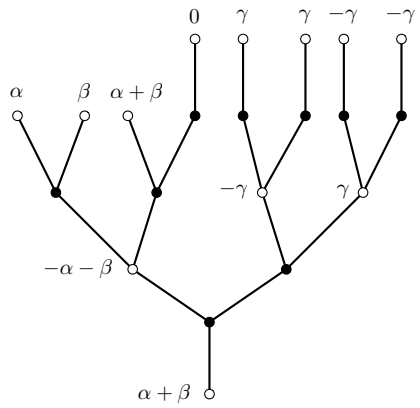

Figure 3: An example NAND tree to illustrate Lemmas 3 and 4. As in Figure 1, a vertex is filled or not according to whether it evaluates to 1 or 0, respectively. The amplitudes $\langle v|a\rangle$ of a zero-energy eigenstate $|a\rangle$ for the adjacency matrix H are also labeled, with α, β, γ free variables, assuming $h_{pv} = 1$ for every edge (p, v). The amplitudes of the neighbors of any vertex sum to zero. The existence of such an $|a\rangle$ is promised by Lemma 4. As required by Lemma 3, $\langle v|a\rangle = 0$ if $\overline{\wedge}(v) = 1$, so vertices evaluating to 1 are not labeled.

4 Zero-energy eigenstates of H

Recall that in a NAND tree T, internal vertices are interpreted as NAND gates on their children. As Definition 2 puts zero weight on the parental edge of a leaf evaluating to one, such leaves can be regarded as disconnected. Then all leaves connected to the root component can be interpreted as zeros.

Definition 4. *By T_v, we mean the subtree of T consisting of v and all its descendants. The restriction to T_v of a vector $|a\rangle$ on T will be denoted $|a_{T_v}\rangle$. That is, for a subset S of the vertices, define the projector $\Pi_S = \sum_{s\in S} |s\rangle\langle s|$; then $|a_{T_v}\rangle = \Pi_{T_v}|a\rangle$. We will also write a_v for $\langle v|a\rangle$. Finally, let $H_S = \Pi_S H$.*

Lemma 3. *For an internal vertex p in NAND tree T, if $\overline{\wedge}(p) = 1$ and $H_{T_p}|a\rangle = 0$, then $a_p = 0$.*

Proof. Since $\overline{\wedge}(p) = 1$, there exists a child v of p having $\overline{\wedge}(v) = 0$. If v is a leaf, then $0 = \langle v|H|a\rangle = h_{pv}a_p$, as asserted. Otherwise, all children c of v must have $\overline{\wedge}(c) = 1$, implying by induction that $a_c = 0$. Then

$$0 = \langle v|H|a\rangle = h_{pv}a_p + \sum_c h_{vc}a_c = h_{pv}a_p \ ,$$

as asserted. □

[1] In fact, $\sigma_-(r) = O(\sqrt{d_r})$, because s_w must increase by at least one every two levels down (two NOT gates in a row would be redundant). Slightly stronger bounds can be given for trees preprocessed according to the rebalancing procedure of Lemma 9, but $\mathrm{poly}(d_r)$ and $\mathrm{poly}(\log N)$ factors here won't significantly change the running time.

Lemma 3 constrains the existence of zero-energy eigenstates supported on the root r when the NAND formula evaluates to 1. However, there may be zero-energy eigenstates that are not supported on the root (for example, consider the right subtree in Figure 3).

Lemma 4. *Consider a vertex p in NAND tree T. If $\overline{\wedge}(p) = 0$, then there exists an $|a\rangle = |a_{T_p}\rangle$ with $H_{T_p}|a\rangle = 0$, $\||a\rangle\| = 1$, and $a_p \geq 1/(\sqrt{\sigma_-(p)}s_p^{1/4})$.*

Proof. Since $\overline{\wedge}(p) = 0$, for all children v of p, $\overline{\wedge}(v) = 1$. So each v has a child c^v satisfying $\overline{\wedge}(c^v) = 0$.

Construct $|a\rangle$ as follows. Set $a_v = 0$ for all children v. Set $|a_{T_c}\rangle = 0$ for all grandchildren $c \notin \{c^v\}$. By induction, for each v construct $|\tilde{a}_{T_{c^v}}\rangle$ satisfying $\||\tilde{a}_{T_{c^v}}\rangle\| = 1$, $H_{T_{c^v}}|\tilde{a}_{T_{c^v}}\rangle = 0$ and $\tilde{a}_{c^v} \geq 1/(\sqrt{\sigma_-(c^v)}s_{c^v}^{1/4})$.

For each v, in order to satisfy $\langle v|H|a\rangle = 0$, we need $h_{pv}a_p = -h_{vc^v}a_{c^v}$. To satisfy all these equations, we *rescale* the vectors $|\tilde{a}_{T_v}\rangle$. Let $a_p = 1$, and let $|a_{T_{c^v}}\rangle = -\frac{h_{pv}}{h_{vc^v}}\frac{1}{\tilde{a}_{c^v}}|\tilde{a}_{T_{c^v}}\rangle$. It only remains to verify that $\||a\rangle\|^2 \leq \sqrt{s_p}\sigma_-(p)$, so that renormalizing, $a_p/\||a\rangle\| = 1/\||a\rangle\|$ is still large. Indeed,

$$\begin{aligned}
\||a\rangle\|^2 &= a_p^2 + \sum_v \frac{h_{pv}^2}{h_{vc^v}^2(\tilde{a}_{c^v})^2}\||a_{T_{c^v}}\rangle\|^2 \\
&\leq 1 + \sum_v \frac{h_{pv}^2}{h_{vc^v}^2}\sqrt{s_{c^v}}\sigma_-(c^v) \\
&\leq 1 + \sum_v \frac{s_v}{\sqrt{s_p}}\sigma_-(c^v) \\
&\leq \sqrt{s_p}\Big(\frac{1}{\sqrt{s_p}} + \max_v \sigma_-(c^v)\Big) \\
&\leq \sqrt{s_p}\sigma_-(p) \ . \qquad \square
\end{aligned}$$

(The key step in the above proof, which motivates the choice of weights h_{pv}, is $\sum_v s_v = s_p$.)

Lemma 4 is a strong converse of Lemma 3, as it does not merely assert that a_p can be set nonzero; it also puts a quantitative lower bound on the achievable magnitude. Lemma 4 lets us say that there *exists* an energy-zero eigenstate with large overlap with the root r when $\overline{\wedge}(r) = 0$.

Now in the case $\varphi(x) = \overline{\wedge}(r) = 0$, let us extend $|a_{T_r}\rangle$ from Lemma 4 into a zero-energy eigenvector $|a\rangle$ over the whole graph, to see that the overlap $|\langle r''|a\rangle|/\||a\rangle\|$ is large. In order to satisfy $H|a\rangle = 0$, we must have $a_{r'} = 0$ and $-a_{r''} = \frac{h_{r'r}}{h_{r''r'}}a_r = \sqrt{\sigma_-(r)}N^{1/4}a_r \geq 1$. Therefore, we lower bound

$$\frac{|\langle r''|a\rangle|}{\||a\rangle\|} \geq \frac{1}{\sqrt{1 + \||a_{T_r}\rangle\|^2}} = \frac{1}{\sqrt{2}} \ .$$

This completes the proof of the first half of Theorem 2.

5 Spectral gap of H in the case $\varphi(x) = 1$

To prove the second half of Theorem 2, we must consider the case $\varphi(x) = 1$, and investigate the eigenvectors of H corresponding to energies E close to zero. As T is a bipartite graph, the spectrum of H is symmetric around zero. Let

$$|E\rangle = \sum_v \alpha_v |v\rangle$$

be an eigenvector of H with eigenvalue $E > 0$.

From Eq. (1) we obtain

$$\langle v|H|E\rangle = E\alpha_v = h_{pv}\alpha_p + \sum_c h_{vc}\alpha_c \ . \qquad (4)$$

The analysis depends on the fact that α_v/α_p is either large or small in magnitude depending on whether $\overline{\wedge}(v) = 0$ or 1.

Lemma 5. *Let $0 < E \leq \frac{1}{5\sigma_-(r)\sqrt{\sigma_+(r)}}$. For vertices $v \neq r''$ in T, define y_{0v} and y_{1v} by*

$$\begin{aligned}
y_{0v} &= (1 + k_v)\sigma_-(v)\sqrt[4]{s_v s_p} \\
y_{1v} &= (1 + k_v)\sigma_-(v)\frac{s_v^{3/4}}{s_p^{1/4}} \ ,
\end{aligned} \qquad (5)$$

where p is the parent of v, and k_v is defined by

$$k_v = 4E^2\sigma_-^2(v)\sigma_+(v).$$

Then for every vertex $v \neq r''$ in T, either $\alpha_v = \alpha_p = 0$, or

$$\begin{aligned}
\overline{\wedge}(v) = 0 \quad &\Rightarrow \quad 0 < \alpha_p/\alpha_v \leq y_{0v}E \\
\overline{\wedge}(v) = 1 \quad &\Rightarrow \quad 0 > \alpha_v/\alpha_p \geq -y_{1v}E \ .
\end{aligned} \qquad (6)$$

Note that, because of our assumption on E, we always have $k_v \leq \frac{4}{25} = 0.16$.

Proof. By induction. Base case: for every leaf v, $\overline{\wedge}(v) = 0$ and by Eq. (4), $E\alpha_v = h_{pv}\alpha_p$. Thus either $\alpha_v = \alpha_p = 0$, or

$$\frac{\alpha_p}{\alpha_v} = \frac{E}{h_{pv}} = \sqrt[4]{\frac{s_p}{s_v}}E = \sqrt[4]{s_p s_v}E \leq y_{0v}E \ .$$

Induction step:

- If $\overline{\wedge}(v) = 0$, then all children c evaluate to $\overline{\wedge}(c) = 1$. First assume $\alpha_v \neq 0$. Dividing both sides of Eq. (4) by α_v, using the induction hypothesis, and rearranging terms gives

$$\begin{aligned}
\frac{\alpha_p}{\alpha_v} &= \frac{1}{h_{pv}}\Big(E - \sum_c h_{vc}\frac{\alpha_c}{\alpha_v}\Big) \\
&\leq \frac{1}{h_{pv}}\Big(1 + \sum_c h_{vc}y_{1c}\Big)E \ .
\end{aligned}$$

Using the inductive assumption about y_{1c} and substituting the expressions for h_{pv}, h_{vc} in terms of α_p, α_v, α_c, we can upper bound the coefficient of E by

$$\frac{1}{h_{pv}} + \sum_c (1+k_c)\sigma_-(c)\frac{s_c s_p^{1/4}}{s_v^{3/4}} \ .$$

Since $\sum_c s_c = s_v$ and $k_c \le k_v$ for any c, this is at most

$$\begin{aligned}(1+k_v)&\Big(\sqrt[4]{\frac{s_p}{s_v}} + \max_c \sigma_-(c)\sqrt[4]{s_v s_p}\Big) \\ &= (1+k_v)\sqrt[4]{s_v s_p}\Big(\max_c \sigma_-(c) + \frac{1}{\sqrt{s_v}}\Big) \\ &= (1+k_v)\sigma_-(v)\sqrt[4]{s_v s_p} \ .\end{aligned}$$

The induction hypothesis also gives that $\alpha_p/\alpha_v \ge E/h_{pv} > 0$. If $\alpha_v = 0$, then the induction hypothesis gives that all α_c are zero, so also $\alpha_p = 0$ by Eq. (4).

- If $\overline{\wedge}(v) = 1$, then there is at least one child c with $\overline{\wedge}(c) = 0$. We may assume $\alpha_v \neq 0$ since otherwise $\alpha_v/\alpha_p = 0$ and the inequality holds trivially. Then, again dividing Eq. (4) by $\alpha_v h_{pv}$, using the induction hypothesis, and multiplying by E,

$$\begin{aligned}\frac{\alpha_p}{\alpha_v} &= \frac{E}{h_{pv}} - \sum_c \frac{h_{vc}}{h_{pv}}\frac{\alpha_c}{\alpha_v} \\ &\le \frac{E}{h_{pv}} + \sum_{c:\overline{\wedge}(c)=1} \frac{h_{vc} y_{1c}}{h_{pv}} E \\ &\qquad - \sum_{c:\overline{\wedge}(c)=0} \frac{h_{vc}}{h_{pv} y_{0c} E} \ .\end{aligned} \tag{7}$$

Because of the previous case, we can upper bound the first sum by

$$\begin{aligned}\sum_c \frac{h_{vc} y_{1c}}{h_{pv}} E &\le \max_c (1+k_c)\sigma_-(c)\sqrt[4]{s_v s_p} E \\ &\le (1+k_v)\sigma_-(v)\sqrt[4]{s_v s_p} E \ .\end{aligned} \tag{8}$$

We lower bound the second sum by one of its terms (since there is at least one c with $\overline{\wedge}(c) = 0$):

$$\frac{h_{vc}}{h_{pv} y_{0c} E} \ge \frac{s_p^{1/4}}{(1+k_c)\sigma_-(v) s_v^{3/4} E} \ .$$

Finally, the first term on the right hand side of Eq. (7) is $\frac{E}{h_{pv}} = \sqrt[4]{\frac{s_p}{s_v}} E$, which is less than the right hand side of Eq. (8). Therefore, Eq. (7) is at most

$$\begin{aligned}2(1+k_v)\sigma_-(v)\sqrt[4]{s_v s_p} E - \frac{s_p^{1/4}}{(1+k_c)\sigma_-(v) s_v^{3/4} E} \\ = \frac{-s_p^{1/4}\Big(1 - 2(1+k_v)(1+k_c)\sigma_-^2(v) s_v E^2\Big)}{(1+k_c)\sigma_-(v) s_v^{3/4} E} \ .\end{aligned}$$

Let $\delta = \sigma_-^2(v) s_v E^2$. Since $k_c \le k_v \le 0.16$, we can lower bound the expression in brackets by

$$1 - 2 \cdot 1.16^2 \delta \ge 1 - 2.7\delta \ .$$

This means that

$$\frac{\alpha_p}{\alpha_v} \le \frac{-s_p^{1/4}}{(1+k_c)\sigma_-(v) s_v^{3/4} E}(1 - 2.7\delta) \ .$$

To complete the proof that $\frac{\alpha_p}{\alpha_v} \le -\frac{1}{y_{1v} E}$ (and, hence, by taking inverses, $\frac{\alpha_v}{\alpha_p} \ge -y_{1v} E$), it suffices to show that

$$\frac{1+k_c}{1-2.7\delta} \le 1 + k_v \ .$$

We have

$$\begin{aligned}\frac{1+k_c}{1-2.7\delta} &= 1 + k_c + (1+k_c)\left(\frac{1}{1-2.7\delta} - 1\right) \\ &\le 1 + k_c + 1.16\left(\frac{1}{1-2.7\delta} - 1\right) \ .\end{aligned} \tag{9}$$

We now observe that $\delta \le \sigma_-^2(v)\sigma_+(v)E^2 \le 0.04$. If $0 \le \delta \le 0.04$, the last term of (9) is always upper bounded by 4δ. Therefore, the entire right hand side of (9) is upper bounded by

$$\begin{aligned}1 + k_c + 4\delta &= 1 + 4\sigma_-^2(v)(\sigma_+(c) + s_v)E^2 \\ &\le 1 + k_v \ .\end{aligned}$$

□

Now we are ready to complete the proof of the second half of Theorem 2. Assume $|E\rangle$ is an eigenvector of H with energy $E \in (0, \frac{1}{5\sigma_-(r)\sqrt{\sigma_+(r)}}]$. We want to show $\alpha_{r'} = \alpha_{r''} = 0$. We have

$$\begin{aligned}E\alpha_{r'} &= h_{r'r''}\alpha_{r''} + h_{rr'}\alpha_r \\ &\ge \frac{h_{r'r''}^2}{E}\alpha_{r'} - h_{rr'} y_{1r} E \alpha_{r'} \ ,\end{aligned}$$

with the inequality following from $E\alpha_{r''} = h_{r'r''}\alpha_{r'}$ and Lemma 5. If $\alpha_{r'} \neq 0$, we can divide both sides by $E\alpha_{r'}$. Then, moving the second term from the right hand side to the left gives us

$$1 + h_{rr'} y_{1r} \ge \frac{h_{r'r''}^2}{E^2} \ .$$

Substituting the values of $h_{rr'}$ and $h_{r'r''}$ and applying the assumed upper bound on E gives us

$$1 + y_{1r} \ge \frac{25\sigma_-(r)\sigma_+(r)}{\sqrt{N}} \ge 25\sigma_-(r)\sqrt{N} \ . \tag{10}$$

By Lemma 5, we have

$$y_{1r} = (1 + k_r)\sigma_-(r)\frac{s_{r'}^{3/4}}{s_r^{1/4}} \leq 1.16\sigma_-(r)\sqrt{N} \ .$$

Substituting this into (10) gives a contradiction.

Therefore, $\alpha_{r'} = 0$ and, because of $E\alpha_{r''} = h_{r'r''}\alpha_{r'}$, we also have $\alpha_{r''} = 0$. This completes the proof of Theorem 2.

6 Discrete-time quantum walk

To construct an algorithm from Theorem 2, we first briefly review Szegedy's procedure for quantizing classical random walks. Theorem 6, adapted from [20], relates the eigensystem of the discrete-time quantum walk to that of the original classical walk.

Theorem 6 ([20]). *Let $\{|v\rangle : v \in V\}$ be an orthonormal basis for $\mathcal{H}_V$. For each $v \in V$, let $|\tilde{v}\rangle = |v\rangle \otimes \sum_{w\in V}\sqrt{p_{vw}}|w\rangle \in \mathcal{H}_V \otimes \mathcal{H}_V$, where $p_{vw} \geq 0$ and $\langle\tilde{v}|\tilde{v}\rangle = \sum_w p_{vw} = 1$. Let $T = \sum_v |\tilde{v}\rangle\langle v|$ and $\Pi = TT^\dagger = \sum_v |\tilde{v}\rangle\langle\tilde{v}|$ be the projection onto the span of the $|\tilde{v}\rangle$s. Let $S = \sum_{v,w}|v,w\rangle\langle w,v|$, a swap. Let $M = T^\dagger ST = \sum_{v,w}|v\rangle\langle\tilde{v}|S|\tilde{w}\rangle\langle w| = \sum_{v,w}\sqrt{p_{vw}p_{wv}}|v\rangle\langle w|$ a real symmetric matrix, and take $\{|\lambda_a\rangle\}$ a complete set of orthonormal eigenvectors of M with respective eigenvalues λ_a.*

Let $U = (2\Pi - 1)S$, a swap followed by reflection about the span of the $|\tilde{v}\rangle$s. Then the spectral decomposition of U is determined by that of M as follows: Let $R_a = \mathrm{span}\{T|\lambda_a\rangle, ST|\lambda_a\rangle\}$. Then $R_a \perp R_{a'}$ for $a \neq a'$; let $R = \oplus_a R_a$. U fixes the spaces R_a and is $-S$ on $R^\perp$. The eigenvalues and eigenvectors of U within R_a are given by $\beta_{a,\pm} = -\lambda_a \pm i\sqrt{1-\lambda_a^2}$ and $(1 + \beta_{a,\pm}S)T|\lambda_a\rangle$, respectively.

Proof. First assume $a \neq a'$, and let us show $R_a \perp R_{a'}$. Indeed, $\langle\lambda_a|T^\dagger T|\lambda_{a'}\rangle = \langle\lambda_a|\lambda_{a'}\rangle = 0$, as $T^\dagger T = 1$. Since $S^2 = 1$, similarly, $ST|\lambda_a\rangle$ is orthogonal to $ST|\lambda_{a'}\rangle$. Finally, $\langle\lambda_a|T^\dagger ST|\lambda_{a'}\rangle = \langle\lambda_a|M|\lambda_{a'}\rangle = 0$. Therefore, the decomposition $\mathcal{H}_V \otimes \mathcal{H}_V = (\oplus_a R_a) \oplus R^\perp$ is well-defined.

R is the span of the images of ST and T. $2\Pi - 1$ is $+1$ on the image of T and -1 on its complement; therefore U is $-S$ on $R^\perp$.

Finally, $\Pi T = TT^\dagger T = T$ and $\Pi ST = TT^\dagger ST = TM$, so

$$U(ST|\lambda_a\rangle) = (2\Pi - 1)T|\lambda_a\rangle = T|\lambda_a\rangle$$
$$U(T|\lambda_a\rangle) = (2\Pi - 1)ST|\lambda_a\rangle = (2\lambda_a - S)T|\lambda_a\rangle \ ;$$

U fixes the subspaces R_a. To determine its eigenvalues on R_a, let $|\beta\rangle = (1 + \beta S)T|\lambda_a\rangle$. Then $U|\beta\rangle = (2\lambda_a + \beta)T|\lambda_a\rangle - ST|\lambda_a\rangle$ is proportional to $|\beta\rangle$ if $\beta(2\lambda_a + \beta) = -1$; i.e., $\beta \in \{-\lambda_a \pm i\sqrt{1-\lambda_a^2}\}$. (If $\lambda_a = \pm 1$, note that $T|\lambda_a\rangle = \pm ST|\lambda_a\rangle$, so R_a is one-dimensional, corresponding to a single eigenvector of U.) □

To connect this theorem to classical and quantum walks, start with an undirected graph $G = (V, E)$. Choose the $p_{v,w}$ to be the transition probabilities $v \to w$ of a classical random walk on this graph (i.e., with the constraint $p_{v,w} = 0$ if $(v, w) \notin E$). Then $U = (2\Pi - 1)S$ can be considered a quantization of the classical walk, taking place on the *directed edges* of G. First the swap S switches the direction of an edge. Then, for the first register fixed to be $|v\rangle$, $2\Pi - 1$ acts as a reflection about $|\tilde{v}\rangle = |v\rangle \otimes \sum_{w\sim v}\sqrt{p_{v,w}}|w\rangle$; it is a "coin flip" that mixes the directed edges leaving v. Therefore, although U acts on $\mathcal{H}_V \otimes \mathcal{H}_V$, it preserves the subspace spanned by $|v, w\rangle$ and $|w, v\rangle$ for $(v, w) \in E$. An alternative basis for this subspace is to give a vertex v together with an edge index to describe an edge leaving v. If the graph has maximum degree D, then U can be implemented on $\mathcal{H}_V \otimes \mathbb{C}^D$ instead of $\mathcal{H}_V \otimes \mathcal{H}_V$.

Discretization of continuous-time quantum walks

Szegedy's Theorem 6 relates the eigenvalues and eigenvectors of the quantum walk U to that of the matrix $M = \sum_{v,w}\sqrt{p_{v,w}p_{w,v}}|v\rangle\langle w|$. If $P = \sum_{v,w}\sqrt{p_{v,w}}|v\rangle\langle w|$ is the elementwise square root of the transition matrix of a classical random walk, then M is the elementwise product $P \circ P^T$. But M can also be regarded as the Hamiltonian for a continuous-time quantum walk on the vertices of the underlying graph.

In our case, we are given H, and desire a factorization $H = P \circ P^T$ such that P has all row norms exactly one. Then Theorem 6 with $M = H$ applies to relate the eigensystem of H to that of a certain discrete-time quantum walk. Such a factorization is possible for a large class of Hamiltonians:

Claim 7. *Let $H = \sum_{v,w} H_{v,w}|v\rangle\langle w|$ be the positive-weighted symmetric adjacency matrix of a connected graph G. Let $|\delta\rangle$ be the principal eigenvector of H, with $\langle v|\delta\rangle = \delta_v > 0$ for every v. Assume $\|H\| = 1$. Then $H = P \circ P^T$, where $P = \sum_{v,w}\sqrt{H_{v,w}\frac{\delta_w}{\delta_v}}|v\rangle\langle w|$ has all row norms equal to 1.*

Proof. Since H is nonnegative, its principal eigenvector $|\delta\rangle$ is also nonnegative, by the Perron-Frobenius Theorem. Since H is connected, $\delta_v > 0$ for every v; hence P is well-defined. By construction, $P_{v,w}P_{w,v} = H_{v,w}$, i.e., $P \circ P^T = H$. Furthermore, the squared norm of the vth row of P is $\sum_w P_{v,w}^2 = \frac{1}{\delta_v}\sum_w H_{v,w}\delta_w = \frac{(H\delta)_v}{\delta_v} = \|H\| = 1$, so P corresponds to a classical random walk. □

Remark 2. *Szegedy's Theorem 6, with Claim 7, serves as a general method for relating an arbitrary continuous-time quantum walk on the vertices of G to a discrete-time quantum walk on directed edges of G. In particular, the eigenvalues of the discrete walk $-iU$ are given by $\pm\sqrt{1-\lambda_a^2} + i\lambda_a$*

(i.e., $e^{i \arcsin \lambda_a}$ and $-e^{-i \arcsin \lambda_a}$), whereas the continuous walk e^{iM} has eigenvalues $e^{i\lambda_a}$. The spectral gaps from zero of the continuous walk and the discrete walk are equal up to third order.

7 The algorithm

Recall from Definition 3 that a NAND formula is "approximately balanced" if $\sigma_-(r) = O(1)$, $\sigma_+(r) = O(N)$ and $\|H\| = O(1)$. These conditions are satisfied, for example, by a balanced binary NAND tree, as $\sigma_-(r) < 2$ and $\sigma_+(r) < 2N$ will then both be geometric series. The definition is also satisfied if for a fixed $\epsilon \in (0, \frac{1}{2}]$, for every vertex p and every grandchild c of p, $s_c \leq (1-\epsilon)s_p$. Under this condition, $\sigma_-(r) = O(\frac{1}{\epsilon})$ and $\sigma_+(r) = O(\frac{N}{\epsilon})$.

Theorem 8. *After efficient (i.e., $\mathrm{poly}(N)$ time) classical preprocessing, independent of the input x, of an arbitrary formula φ of size N, $\varphi(x)$ can be evaluated with error $< 1/3$ using $N^{\frac{1}{2}+O(1/\sqrt{\log N})}$ queries to U_x. The running time is also $N^{\frac{1}{2}+O(1/\sqrt{\log N})}$ assuming unit-cost coherent access to the preprocessed string. For a formula that is approximately balanced according to Definition 3, the query complexity is only $O(\sqrt{N})$ and the running time is only $\sqrt{N}(\log N)^{O(1)}$.*

Proof. **Preprocessing:**

If $\sigma_-(r)\sqrt{\sigma_+(r)}\|H\| = N^{\frac{1}{2}+\Omega(1)}$, then preprocess the formula in two ways. First, expand out gates so each NAND gate has $O(1)$ fan-in. Since the edge weights are all ≤ 1, this ensures that $\|H\| = O(1)$. Also, apply the formula rebalancing procedure of [6, 5] with parameter k to be determined:

Lemma 9 ([5, Theorem 4])**.** *For all $k \geq 2$, one can efficiently construct an equivalent NAND formula φ with gate fan-ins at most two and satisfying*[2]

$$\mathrm{depth}(\varphi) \leq (9 \ln 2) k \log_2 N$$
$$\mathrm{size}(\varphi) \leq N^{1+1/\log_2 k} .$$

Let H be the Hamiltonian corresponding to a weighted adjacency matrix of the graph according to Definition 2. Compute a discrete-time quantum walk operator U that corresponds to $M = H/n(H)$ via Theorem 6 (where $n(H)$ is some upper bound on $\|H\|$, to ensure that all eigenvalues of M have $|\lambda_a| \leq 1$). Obtaining U takes a little care, since H depends on the oracle. Consider H_{0^N} the Hamiltonian from Definition 2 assuming that all leaves evaluate to 0. By applying Claim 7 as part of the preprocessing, we obtain a U_{0^N} corresponding to $H_{0^N}/\|H_{0^N}\|$. Let $U = O_x U_{0^N}$, where O_x is the input phase-flip oracle; conditioned on the current vertex being a leaf i, O_x adds a phase of $(-1)^{x_i}$. Then we claim that applying Theorem 6 to U gives $M = H/\|H_{0^N}\|$. Indeed, the only difference between U and U_{0^N} is on the $|\tilde{i}\rangle$s for leaf vertices i with $x_i = 1$; in U_{0^N}, $|\tilde{i}\rangle = |i, p\rangle$ (p being i's parent), whereas in U, $|\tilde{i}\rangle = |i, i\rangle$ (i.e., $p_{i,i} = 1$). Therefore the M from U only differs from H_{0^N} in the coefficients involving leaves i with $x_i = 1$, and $\langle i|M|p\rangle = \langle\tilde{i}|S|\tilde{p}\rangle = 0$, so $M = H/\|H_{0^N}\|$ as claimed.

Algorithm:

1. Prepare $|\tilde{r}''\rangle = |r'', r'\rangle$.
2. "Measure the energy according to H." In other words, apply quantum phase estimation to $-iU = -iO_x U_{0^N}$. Use precision $\delta_p = 1/(10\sigma_-(r)\sqrt{\sigma_+(r)})$ and error probability δ_e any constant less than $1/4$.
3. Output zero if and only if the measured phase is 0 or π.

Figure 2 lays out the steps of the algorithm in complete detail for the case of a balanced binary NAND tree. We did not use Claim 7 to derive U in Figure 2, because in this special case it is clear that applying Theorem 6 to U gives H, except with larger weights to leaves evaluating to 0 (see Remark 1).

Correctness:

The correctness follows from Theorems 2 and 6. If $\varphi(x) = 0$, then there exist two eigenvectors of U given by $(1 \pm iS)T|a\rangle$, with eigenvalues $\pm i$, respectively. Their overlaps with the initial state $|\tilde{r}''\rangle$ are $|\langle\tilde{r}''|(1 \pm iS)T|a\rangle| = |a_{r''} \pm i a_{r'} \frac{h_{r''r'}}{\|H\|}| \geq 1/\sqrt{2} - O(1/N^{1/4})$. Since the norm of $(1 \pm iS)T|a\rangle$ is at most $2\||a\rangle\| = 2$, we find that the probability of outputting 0 is at least $2\big(\frac{1}{2}(\frac{1}{\sqrt{2}} - o(1))\big)^2 = 1/4 - o(1)$.

Conversely, if $\varphi(x) = 1$, then every eigenstate of H with support on r' or r'' has energy at least $1/(5\sigma_-(r)\sqrt{\sigma_+(r)})$ in magnitude. Every eigenstate of U with support on $|r'', r'\rangle = |\tilde{r}''\rangle = T|r''\rangle$ must be of the form $(1 + \beta_{a,\pm}S)T|\lambda_a\rangle = (1 + (-\lambda_a \pm i\sqrt{1-\lambda_a^2})S)T|\lambda_a\rangle$. The terms which can overlap $T|r''\rangle$ are either $\langle r''|\lambda_a\rangle$ (via T) or $\langle r'|\lambda_a\rangle$ (via ST). But for $|\lambda_a| < 1/(10\sigma_-(r)\sqrt{\sigma_+(r)})$, both coefficients must be zero. Therefore, our algorithm outputs 0 with probability less than $\delta_e < 1/4$. This constant gap can be amplified as usual.

Query and time complexity:

Phase estimation requires applying $O(\|H_{0^N}\|/(\delta_p\delta_e)) = O(\sigma_-(r)\sqrt{\sigma_+(r)}\|H_{0^N}\|)$ controlled$U$ evolutions [10].

[2]The constant in the depth bound is $9 \ln 2$ instead of the $3 \ln 2$ in [5, Theorem 4] because we lose a constant converting an {AND, OR, NOT} formula to a NAND formula.

For an approximately balanced graph, this is $O(\sqrt{N})$. For a general graph, the number of controlled$U$ applications is $O(\sqrt{s_r}d_r^{3/2}\|H_{0^N}\|)$ due to the bounds on $\sigma_\pm(r)$ from Definition 3. For a rebalanced formula from Lemma 9 with parameter k, this is $O(N^{\frac{1}{2}+\frac{1}{2\log_2 k}}(k\log_2 N)^{3/2})$ since $\|H\| = O(1)$. Set $k = 2^{\sqrt{\frac{\log_2 N}{3}}}$ to optimize this bound to be $N^{\frac{1}{2}+\sqrt{(3+o(1))/\log N}}$ queries to O_x.

During the preprocessing phase, for each vertex v we compute a sequence of $O((\log N)^2 \log\log N)$ elementary gates that approximate to within $1/N$ the coin diffusion operator at v, by applying the Solovay-Kitaev Theorem [15]. (Using these approximations, the algorithm's total error probability will only increase by $o(1)$.) Store the descriptions of these gate sequences in a classical string, which we assume the algorithm can access coherently at unit cost. The algorithm at vertex v looks up the corresponding gate sequence and applies it to the coin register $|c\rangle$. The total running time is thus only polylogarithmically larger than the number of queries. □

8 Evaluating iterated functions

Our algorithm can be used to evaluate an arbitrary Boolean function by first writing a NAND formula for the function and then evaluating that formula. Of course, this approach will only be advantageous when the formula size is sufficiently small. This strategy is particularly natural in the case of a recursively defined function. In particular, it gives improved upper bounds for an iterated function studied in [1].

Define a function "all equal" as follows: $f(x_1,x_2,x_3) = 1$ if $x_1 = x_2 = x_3$ and $f(x_1,x_2,x_3) = 0$ otherwise. We define a sequence of iterated functions $f_1, f_2, \ldots$, with f_n being a function of 3^n variables. Let $f_1 = f$, and

$$\begin{aligned} f_n(x_1,\ldots,x_{3^n}) = f\big(&f_{n-1}(x_1,\ldots,x_{3^{n-1}}),\\ &f_{n-1}(x_{3^{n-1}+1},\ldots,x_{2\cdot 3^{n-1}}),\\ &f_{n-1}(x_{2\cdot 3^{n-1}+1},\ldots,x_{3^n})\big)\ . \end{aligned}$$

The functions f_n have attracted interest in the context of relating polynomial degree and quantum query complexity of Boolean functions. The polynomial degree of a Boolean function f is always a lower bound on its quantum query complexity [4]. The function f_n is one of the known functions for which this lower bound is not optimal. The polynomial degree of f_n is 2^n, while the quantum query complexity of f_n is lower bounded by $\Omega((\frac{3}{\sqrt{2}})^n) = \Omega(2.12^n)$ [1].

So far, there has been no quantum algorithm for f_n using $o(3^n)$ queries. Our approach gives the first non-trivial quantum algorithm for this function. To see this, note that both f and its negation can be represented by NAND formulas of size 6 as follows:

$$\begin{aligned} f(x_1,x_2,x_3) &= \overline{\wedge}(\overline{\wedge}(x_1,x_2,x_3),\overline{\wedge}(\bar{x}_1,\bar{x}_2,\bar{x}_2))\\ \bar{f}(x_1,x_2,x_3) &= \overline{\wedge}(\overline{\wedge}(x_1,\bar{x}_2),\overline{\wedge}(x_2,\bar{x}_3),\overline{\wedge}(x_3,\bar{x}_1))\ . \end{aligned}$$

Using these formulas, we can inductively construct NAND formulas for f^n and $\bar{f}^n$ of size 6^n. Namely, given NAND formulas for f^{n-1} and $\bar{f}^{n-1}$ of size 6^{n-1}, we can substitute those instead of variables into the NAND formulas for f and $\bar{f}$ to obtain NAND formulas for f^n and $\bar{f}^n$ of size 6^n. Thus, using our algorithm for NAND tree evaluation, we can evaluate f_n using $O(\sqrt{6^n}) = O(2.45^n)$ quantum queries.

Another interesting function for which our algorithm gives an improved upper bound is the majority function: $g(x_1,x_2,x_3) = 1$ if and only if $x_1 + x_2 + x_3 \geq 2$. The majority function can be expressed as $g(x_1,x_2,x_3) = (x_1 \wedge x_2) \vee ((x_1 \vee x_2) \wedge x_3)$, hence the iterated function g_n can be evaluated in $O(\sqrt{5^n}) = O(2.24^n)$ queries.

9 Open problems

We conclude by mentioning some open problems.

- Our algorithm needs to know the full structure of the formula beforehand to determine the quantum walk transition amplitudes (i.e., the biases of the "quantum coin") at each internal vertex. (The transition amplitudes are determined by the principal eigenvector of the graph's weighted adjacency matrix; they can also be solved for recursively from the leaves to r.) However, these coefficients need not be computed exactly, because there is some freedom in the recurrence on y_{0v} and y_{1v}. It would be interesting to know if a different choice of coefficients, or a relaxed calculation thereof, would allow for faster preprocessing. Furthermore, it would be interesting to know on what kinds of structured inputs the preprocessing can be done in time $N^{\frac{1}{2}+o(1)}$.

- Numerical simulations indicate that the formula can be evaluated by running the quantum walk from the initial state, and measuring whether the final quantum state has large overlap with $\frac{1}{\sqrt{2}}(|r',r''\rangle + |r'',r'\rangle)$ or $\frac{1}{\sqrt{2}}(|r',r''\rangle - |r'',r'\rangle)$. If this is true, then we can avoid the phase estimation on top of the quantum walk, simplifying the algorithm.

- What kinds of noisy oracles can this algorithm, or an extended version, tolerate? For example, [14] extends Grover search to the case where input values are computed by a bounded-error quantum subroutine.

- Are there hard instances of formulas for which the rebalancing provided by Lemma 9 is tight? Are these also hard instances for our algorithm? For example, the most unbalanced formula, $\varphi(x_1 \dots x_N) = x_1 \overline{\wedge}(x_2 \overline{\wedge}(x_3 \overline{\wedge}(\dots \overline{\wedge} x_N)))$, is not a hard instance. It can be rebalanced by a different procedure, giving an equivalent formula with depth $O(\log N)$ and size $O(N \log N)$, and can be evaluated with $O(\sqrt{N})$ queries.

Acknowledgments

Robert would like to thank the Institute for Quantum Information at Caltech for hospitality. We thank Ronald de Wolf for comments on an early draft of the paper, and thank Jérémie Roland for thoughtful remarks.

References

[1] A. Ambainis. Polynomial degree vs. quantum query complexity. *J. Comput. Syst. Sci.*, 72(2):220–238, 2006.

[2] A. Ambainis. A nearly optimal discrete query quantum algorithm for evaluating NAND formulas. quant-ph/0704.3628, 2007.

[3] H. Barnum and M. Saks. A lower bound on the quantum query complexity of read-once functions. *J. Comput. Syst. Sci.*, 69(2):244–258, 2004.

[4] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001.

[5] M. L. Bonet and S. R. Buss. Size-depth tradeoffs for Boolean formulae. *Information Processing Letters*, 49(3):151–155, 1994.

[6] N. H. Bshouty, R. Cleve, and W. Eberly. Size-depth tradeoffs for algebraic formulae. In *Proc. of 32nd IEEE FOCS*, pages 334–341, 1991.

[7] H. Buhrman, R. Cleve, and A. Wigderson. Quantum vs. classical communication and computation. In *Proc. of 30th ACM STOC*, pages 63–68, 1998.

[8] A. M. Childs, R. Cleve, S. P. Jordan, and D. Yeung. Discrete-query quantum algorithm for NAND trees. quant-ph/0702160, 2007.

[9] A. M. Childs, B. W. Reichardt, R. Špalek, and S. Zhang. Every NAND formula of size N can be evaluated in time $O(N^{1/2+\varepsilon})$ on a quantum computer. quant-ph/0703015, 2007.

[10] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proc. R. Soc. London A*, 454(1969):339–354, 1998.

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

[12] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. of 28th ACM STOC*, pages 212–219, 1996.

[13] L. K. Grover. Tradeoffs in the quantum search algorithm. quant-ph/0201152, 2002.

[14] P. Høyer, M. Mosca, and R. d. Wolf. Quantum search on bounded-error inputs. In *Proc. of 30th ICALP*, pages 291–299, 2003. LNCS 2719.

[15] A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, Rhode Island, 2002.

[16] S. Laplante, T. Lee, and M. Szegedy. The quantum adversary method and classical formula size lower bounds. *Computational Complexity*, 15:163–196, 2006. Earlier version in Complexity'05.

[17] M. Saks and A. Wigderson. Probabilistic Boolean decision trees and the complexity of evaluating game trees. In *Proc. of 27th IEEE FOCS*, pages 29–38, 1986.

[18] M. Santha. On the Monte Carlo decision tree complexity of read-once formulae. *Random Structures and Algorithms*, 6(1):75–87, 1995.

[19] M. Snir. Lower bounds on probabilistic linear decision trees. *Theoretical Computer Science*, 38:69–82, 1985.

[20] M. Szegedy. Quantum speed-up of Markov chain based algorithms. In *Proc. of 45th IEEE FOCS*, pages 32–41, 2004.