Two methods for breaking down a quantum algorithm

Miguel Murça,[1,2,3,*] Duarte Magano,[1,4] and Yasser Omar[1,2,3]
[1]*Instituto Superior Técnico, Universidade de Lisboa, Lisboa 1049-001, Portugal*
[2]*Centro de Física e Engenharia de Materiais Avançados (CeFEMA), Physics of Information and Quantum Technologies Group, Lisboa 1049-001, Portugal*
[3]*PQI – Portuguese Quantum Institute, Lisboa 1600-531, Portugal*
[4]*Instituto de Telecomunicações, Lisboa 1049-001, Portugal*

(Received 26 May 2023; accepted 9 January 2024; published 9 February 2024)

Despite the promise that fault-tolerant quantum computers can efficiently solve classically intractable problems, it remains a major challenge to find quantum algorithms that may reach computational advantage in the present era of noisy, small-scale quantum hardware. Thus, there is a substantial ongoing effort to create new quantum algorithms (or adapt existing ones) to accommodate depth and space restrictions. By adopting a hybrid query perspective, we identify and characterize two methods of breaking down quantum algorithms into rounds of lower (query) depth, designating these approaches as "parallelization" and "interpolation." To the best of our knowledge, these had not been explicitly identified and compared side by side, although one can find instances of them in the literature. We apply them to two problems with known quantum speedup: calculating the k-threshold function and computing a NAND tree. We show that for the first problem parallelization offers the best performance, while for the second interpolation is the better choice. This illustrates that no approach is strictly better than the other, and that there is more than one good way to break down a quantum algorithm into a hybrid quantum-classical algorithm.

DOI: 10.1103/PhysRevA.109.022412

I. INTRODUCTION

Algorithms that combine classical processing with limited quantum computational resources hold an attractive promise: To provide computational advantage over completely classical computation, while remaining compatible with the technological landscape of quantum computing. The appeal of this kind of algorithm is well reflected in some of the key modern proposals for quantum advantage, usually based on variational principles [1]. Prominent examples include the quantum approximate optimization algorithm [2], the variational quantum eigensolver [3–6], and some versions of quantum machine learning [7–11]. All of these attempt to exploit circuits of limited coherence to obtain computational advantage. However, variational approaches often cannot offer theoretical performance guarantees, as discussed in Refs. [12,13].

Consider instead a setting where we are given a quantum algorithm with guaranteed advantage for a certain computational problem, but the available hardware is too noisy to execute the algorithm with a reasonable fidelity. We would need to limit the circuit depths to values much shorter than the ones prescribed by the original algorithm to prevent errors from dominating the calculations. Is it still possible to guarantee some quantum advantage? We may phrase this question more precisely. Say we are faced with a computational problem f that can be solved by a classical computer in time $C(f)$, and we know a quantum algorithm that solves f with complexity $Q(f)$ [$Q(f) < C(f)$] by running quantum circuits of depth D; what is the best we can do if we are only permitted to run quantum circuits up to a depth D' smaller than D? The expectation is that the best strategy yields an algorithm with a complexity between $C(f)$ and $Q(f)$. We believe that understanding this question may contribute to finding practical but provable advantages in near-term quantum computers.

For oracular problems, the notion of limited coherence is captured by the hybrid query (or decision tree) complexity $Q(f;D)$, introduced by Sun and Zheng [14]. In this setting, only the input accesses (or queries) contribute to the complexity count, while the intermediate computations are free. $Q(f;D)$ is defined as the minimum number of queries required to solve f when limited to running quantum circuits of depth D. That is, we can only perform D queries before being forced to measure the state of the circuit and restart it.

It is known that quantum decision trees are strictly more powerful than hybrid decision trees, which are strictly more powerful than classical decision trees. Concretely, there is a problem f for which $C(f)$ and $Q[f, O(1)]$ are (super) exponentially separated [15], and similarly there is a problem f for which $Q[f, O(1)]$ and $Q(f)$ are exponentially separated [14]. There are also problems f that exhibit a continuous tradeoff between speedup and circuit depth [16], i.e., $Q(f;D) < Q(f;D+1)$ for every D between 1 and $Q(f)$.

While theoretically useful, as illustrated by these separations between quantum, classical, and hybrid query complexity, it should be noted that an advantage under the query model does not directly translate to an advantage in circuit depth. Nevertheless, an upper bound for the query model usually implies an upper bound for the gate

*Corresponding author: miguel.murca@tecnico.ulisboa.pt

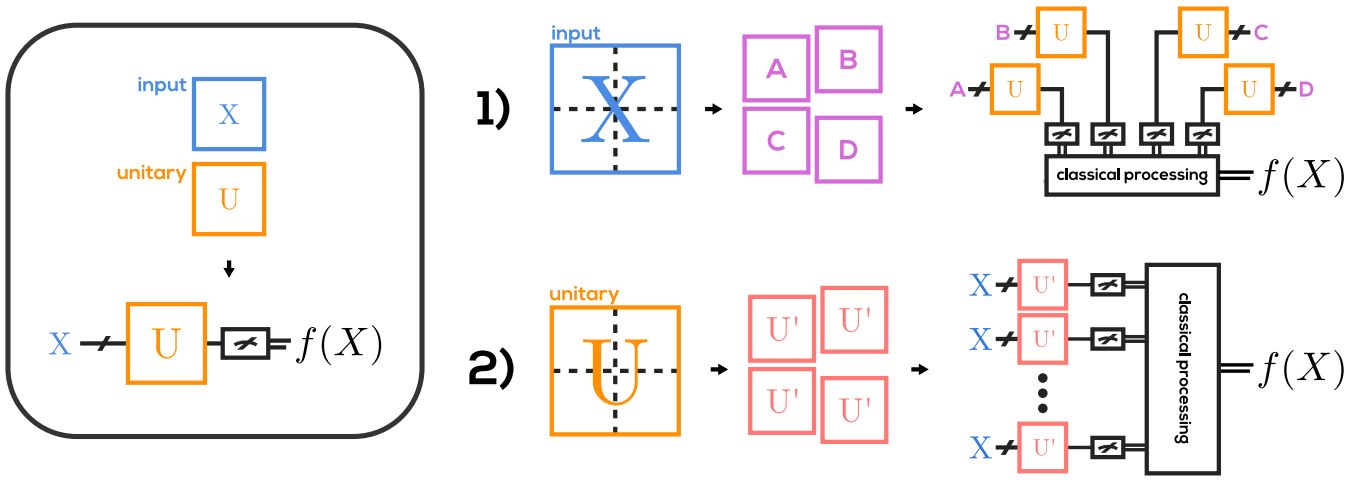


FIG. 1. Diagrammatic representation of the procedures of parallelization (procedure 1) and interpolation (procedure 2), as defined and exemplified in this paper. For both procedures, the overall goal is to carry out a quantum algorithm (as described by some unitary U) for some input X to calculate a property f of X, as shown in the box to the left. However, this unitary may require a prohibitive depth (modeled by us as a prohibitive amount of coherent quantum queries). In the case of parallelization (procedure 1), this is dealt with by identifying independent, smaller instances of the same problem that can be dealt with within the query constraints; in other words, by partitioning the input appropriately into subproblems. Interpolation (procedure 2) involves, instead, considering multiple repetitions of some unitary (or sequence of unitaries) that require, individually, less coherent queries, but that collectively yield the same information as a single run of U. For both of these approaches, breaking up the original algorithm may come at a cost of more overall queries—indeed, we expect that this will be the case for most algorithms. We show that no method is strictly better than the other, and that the best choice depends on the problem at hand.

model. For example, in the context of quantum singular value transformations (QSVTs) [17], discussed in detail in Sec. II B, we can block-encode an efficiently row-accessible sparse matrix with $O[\text{polylog}(n)]$ elementary gates [18], meaning that the corresponding upper bounds for the query and gate models differ only in polylogarithmic factors. Given this, and the theoretical tools available to deal with query complexity, we pose that the point of view of query complexity is a good first step to rigorously understanding quantum-classical hybrid algorithms. This is in similarity to what was historically done for fully quantum algorithms, where the Deutsch-Jozsa algorithm [19] first separated quantum and classical algorithms in the query model, before proposals like Shor's algorithm [20] provided a tentative absolute separation. Nonetheless, one must remember, when comparing in the query model, for example, QSVT-based algorithms with algorithms in a different framework, that the *caveats* of query vs time complexity apply.

Regardless, and despite the aforementioned results in hybrid query complexity, and knowledge of quantum algorithms with (time complexity) advantage, it is nonetheless not obvious how to, in general, optimally break up a quantum algorithm into circuits of smaller sizes. This problem has been tackled: For example, when given a quantum circuit that is too deep, Pérez-Salinas *et al.* [21] propose a heuristic algorithm where one performs intermediate measurements in a parametrized basis, as given by a shallow variational circuit, optimized to minimize the effect of measuring and restarting the quantum operation. But, the expressiveness of the shallow circuit determining the measurement basis and the difficulty of minimizing the cost function may limit the success of this approach, and theoretical guarantees are again lost.

In this paper, we identify and discuss two general strategies with theoretical guarantees to limit the number of queries performed coherently in an algorithm to some specified value D. In other words, we discuss two methods to deal with limits on the allowed query depth, modeling the more practical case of limits in circuit depth. We refer to these two methods as "parallelization" and "interpolation."

Parallelization applies when a problem can be broken down into a number of smaller, independent subproblems, such that the algorithm that solves these subproblems fits the permitted query limits. In contrast, with interpolation the entire problem is tackled at each circuit run. We show that interpolation applies whenever there is a trade-off between the information content of the measurement and the query depth of the corresponding circuit. In these cases, we may compensate for the information loss caused by shortening the circuit depth with repeated runs of the shorter circuit. Intuitively, we can say that interpolation methods break up the unitary that solves the problem instead of breaking up the problem itself. See Fig. 1 for an illustration of these notions.

To the best of our knowledge, neither of these methods had been explicitly identified and compared side by side, even though several works that fit into these labels can be found in the literature. For example, parallelization approaches are present in Refs. [22,23], while Refs. [16,24–26] describe interpolation methods. We more finely characterize each of the methods in Secs. III A (parallelization) and III B (interpolation), further discussing how the above cited methods in the literature may fit into these concepts. Note that our proposed notions of parallelization and interpolation are not meant to be new methods improving over the cited methods. We furthermore do not provide a general statement on which method is the best for any given problem. Rather, we are seeking to identify two common approaches to breaking up quantum algorithms into multiple rounds of limited quantum computation (here, limited in the number of queries), and

provide a unified description and distinction of these methods. We also rule out the possibility that either method is strictly better than the other.

We illustrate these methods with two well-known problems: The k-threshold function and perfectly balanced NAND trees. These problems are known to exhibit quantum speedups (see Refs. [27] and [2,28,29]), but, to the best of our knowledge, neither has been discussed in the context of a quantum-classical hybrid computing model. Both problems are amenable to both parallelization and interpolation. We show that for the k-threshold function parallelization offers the best performance (query wise), while for evaluating perfectly balanced NAND trees the interpolation method is the most efficient (*idem*). This reinforces the relevance of the distinction between parallelization and interpolation, and demonstrates that no technique is *a priori* better than the other, as the best option depends on the problem at hand.

II. PRELIMINARIES

A. Hybrid query model

We will be working mostly within the query model of quantum computing. Here, we quickly review the main concepts, referring to Ambainis [30] for a more in-depth discussion.

The quantum query complexity model, a generalization of decision tree complexity [31], is widely used to study the power of quantum computers. On one hand, it captures the important features of most quantum algorithms, including search [32], period finding [20], and element distinctness [33]. On the other hand, it is simple enough to make the proof of lower bounds attainable [27,34].

In the query model, the goal is to compute a Boolean function $f(x_1, \ldots, x_N)$ of variables $x_i \in \{0, 1\}$. The function can be total (defined on $\{0, 1\}^N$) or partial (defined on a subset of $\{0, 1\}^N$). We only get information about the input variables by querying a black-box quantum operator O acting as

$$O|i\rangle|b\rangle = |i\rangle|b \oplus x_i\rangle \tag{1}$$

for every $b \in \{0, 1\}$ and $i \in \{0, 1\}^N$. A quantum query algorithm is specified by a set of input-independent unitaries $U_0, U_1, \ldots, U_T$. The algorithm consists of performing the transformation

$$U_T\, O\, U_{T-1}, \ldots, U_2\, O\, U_1\, O\, U_0|0\rangle \tag{2}$$

and measuring the result, which is then converted into the answer of the problem according to a predefined rule. In the query model, the algorithm's complexity increases with each query, while the intermediate computations are free. That is, the complexity of the algorithm corresponding to transformation (2) is T, independent of how the unitaries U_i are chosen.

We say that a quantum algorithm computes f with bounded error if, for all $x \in \{0, 1\}^N$, the answer of the algorithm agrees with $f(x)$ with probability at least $2/3$, where the probability is over the randomness of the algorithm's measuring process. The minimum query complexity of any bounded-error algorithm computing f is the quantum (bounded-error) complexity of f, denoted as $Q(f)$.

The hybrid query model introduced by Sun and Zheng [14] captures the idea of restricted-depth computation in an oracular setting. Hybrid algorithms are in direct correspondence with hybrid decision trees. A hybrid decision tree is similar to a (classical) decision tree, but the decision at each node is determined by the output of a quantum algorithm with query complexity no more than a value D, which we refer to as the depth of the hybrid algorithm. The hybrid algorithm's answer is the output of the algorithm at the leaf node. More plainly, a hybrid algorithm works by running and measuring sequences of circuits like (2) with $T \leqslant D$, using the intermediate measurements to decide what quantum circuit to run next.

A hybrid algorithm computes f with bounded error if, for all $x \in \{0, 1\}^N$, the answer of the algorithm agrees with $f(x)$ with probability at least $2/3$, where the probability is over the randomness of the internal measurements. The complexity of a path in a hybrid tree is the sum of the complexities of the algorithms associated with each node in the path. The complexity of a hybrid algorithm that computes a function f is the maximal complexity of any path that connects the root and a leaf, that is, it is the total number of queries needed to evaluate f in the worst case. The minimum query complexity of any bounded-error hybrid algorithm computing f is the hybrid (bounded-error) complexity of f, denoted as $Q(f; D)$.

B. Quantum singular value transformations

In this paper we make extensive use of QSVTs [17,18,35]. As a generalization of the work on quantum signal processing [36], QSVTs have provided a unifying description of several algorithms, including amplitude estimation, quantum simulation, and quantum methods for linear systems.

By the singular value decomposition theorem, an arbitrary matrix A of rank r can be written as

$$A = \sum_{i=1}^{r} \sigma_i |w_k\rangle\langle v_k|, \tag{3}$$

where $\{w_k\}_k$ and $\{v_k\}_k$ are orthogonal sets (known as the left and right singular values of A, respectively) and $\{\sigma_k\}_k$ are positive real numbers (known as the singular values of A). For functions $P : \mathbb{R} \to \mathbb{C}$, we call

$$P^{(\mathrm{SV})}(A) := \sum_{i=1}^{r} P(\sigma_i)|w_k\rangle\langle v_k| \tag{4}$$

a singular value transformation of A.

When considering performing such transformations on arbitrary matrices with quantum computers, we are immediately faced with the difficulty that quantum states evolve according to unitary transformations. The introduction of *block encodings* overcomes this apparent limitation [37]. Let Π and $\tilde{\Pi}$ be orthogonal projectors and U be a unitary; we say that Π, $\tilde{\Pi}$, and U form a block encoding of the operator A if

$$A = \tilde{\Pi} U \Pi. \tag{5}$$

Based on this concept, the main theorem of QSVTs can be phrased as follows.

Theorem 1 (QSVTs [17]). Let Π, $\tilde{\Pi}$, and U be a block encoding of a matrix A, and let $P : [-1, 1] \to [-1, 1]$ be a polynomial of degree d. Then, we can implement a unitary U_P such that Π, $\tilde{\Pi}$, and U_P form a block encoding of

$P^{(\mathrm{SV})}(A)$ using $O(d)$ calls to $U, U^\dagger$ and $\Pi/\tilde{\Pi}$-controlled-NOT operations [38].

A transformation that will be particularly useful for us is the step (or Heaviside) function,

$$\sigma \mapsto \begin{cases} 1, & \text{if } \sigma \geqslant \mu \\ 0, & \text{if } \sigma < \mu \end{cases}, \tag{6}$$

for some $\mu \in [-1, 1]$. References [17,39] show that we can approximate this transformation up to arbitrary accuracy by a polynomial approximation of the error function, defined as

$$\mathrm{erf}(x) := \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \tag{7}$$

The result is stated below.

Theorem 2 (Polynomial approximation of step function [17]). There is a polynomial $P_{\delta,\eta,\mu}(\lambda) : [-1, 1] \to [-1, 1]$ of degree

$$O\left[\frac{1}{\delta} \ln\left(\frac{1}{\eta}\right)\right] \tag{8}$$

satisfying

$$|P_{\delta,\eta,\mu}(\sigma)| \leqslant \eta, \quad \forall \sigma \in [-1, \mu - \delta] \tag{9}$$

$$P_{\delta,\eta,\mu}(\sigma) \geqslant 1 - \eta, \quad \forall \sigma \in [\mu + \delta, 1]. \tag{10}$$

We will also be interested in performing a step transformation on the modulus of the singular values (also known as a window function due to the shape of its plot),

$$\sigma \to \begin{cases} 1, & \text{if } |\sigma| \leqslant \mu \\ 0, & \text{if } |\sigma| > \mu \end{cases}. \tag{11}$$

Noting that $P_{\delta,\eta,-\mu} - P_{\delta,\eta,\mu}$ is a polynomial with the same degree as $P_{\delta,\eta,\mu}$, we immediately derive the following.

Corollary 1 (Polynomial approximation of window function). There is a polynomial $P'_{\delta,\eta,\mu}(\lambda) : [-1, 1] \to [-1, 1]$ of degree

$$O\left[\frac{1}{\delta} \ln\left(\frac{1}{\eta}\right)\right] \tag{12}$$

satisfying

$$|P'_{\delta,\eta,\mu}(\sigma)| \leqslant \eta, \quad \forall \sigma \in [-1, -\mu - \delta] \cup [\mu + \delta, 1] \tag{13}$$

$$P'_{\delta,\eta,\mu}(\sigma) \geqslant 1 - \eta, \quad \forall \sigma \in [-\mu + \delta, \mu - \delta]. \tag{14}$$

Combining Theorems 1 and 2, we find a method to distinguish the singular values of a block-encoded matrix that are above or below a given threshold. Similarly, from Theorem 1 and Corollary 1 we can distinguish the singular values of a block-encoded matrix whose modulus are above or below a given threshold.

III. TWO APPROACHES TO RESTRICTED QUERY-DEPTH COMPUTATION

A. Parallelization

In many cases the problem at hand can be broken down into a number of smaller, independent subproblems. As an example, consider the problem of computing the OR function on N bits. We can partition the domain into p subdomains of size approximately N/p. If for any of those subdomains there is an index i for which $x_i = 1$, then we return 1; otherwise the answer is 0. In other words, the problem is reduced to evaluating p OR function on N/p bits. With Grover's algorithm [32] we can evaluate each subdomain with $O(\sqrt{N/p})$ queries. In total, this strategy has a query complexity of

$$O(\sqrt{pN}). \tag{15}$$

If we are limited to circuits of query depth D, we set $p = O(N/D^2)$, finding that

$$Q(\mathrm{OR}; D) = O\left(\frac{N}{D} + \sqrt{N}\right). \tag{16}$$

By Corollary 1.5 of Sun and Zheng [14], this is optimal.

We say that the algorithms that employ this kind of strategy—breaking the problem into smaller, independent problems that fit the permitted query depth—fall into the category of parallelization methods.

Note that this kind of procedure does *not* presume multiple quantum processors operating at the same time, even though it is amenable to it. The important point is that the different subproblems considered are independent and may be treated as such. This should be contrasted with the notion of parallel quantum algorithms as defined by Jeffery *et al.* [40], where a number of queries are realized at the same time (in parallel), but by a number of quantum registers that may be entangled with each other. This requires fully coherent operation, preventing a trade-off of resources. For our notion of parallelization, the quantum registers may *not* be entangled with each other, as, in principle, every round *could* be performed in series, with the same processor, halting and measuring between each round.

Arguably, parallelization as described above is the most natural approach to breaking up a quantum algorithm into independent circuits of lower quantum query depth, since there is an exploitation of the structure of the problem and its input to produce multiple problems, individually fitting the imposed constraints. One example of parallelization within the query model is Zalka [22], containing the OR function discussed above. Critically, Zalka performs parallelization "by assigning different parts of the search space to *independent* quantum computers." Another example is Grover and Radhakrishnan [23], who generalize the approach of Zalka and investigate the results of searching for marked elements over many copies of a database. Like Zalka, the action of the parallel processors is considered independent, and to each processor is given a copy of the database. Departing from the query model, in literature concerned with near-term quantum computing, where circuit depth limitation is a main concern, we again find proposals that divide a large problem into independent smaller instances by exploiting the structure of the input. For example, in the variational quantum eigensolver proposal [3], the (more demanding) problem of a phase estimation is replaced by the evaluation of multiple expectation values of observables that are easily implemented as measurements, by using the fact that the Hamiltonians under study are local. We may find generalizations of this approach, now exploiting, for example, the sparsity of the Hamiltonians [41].

B. Interpolation

Contrary to parallelization, interpolation methods do not distribute the problem into different subproblems. Instead, at each run the entire problem is tackled, but this is done over several quantum circuit runs. Since the circuit query depth is limited, each circuit measurement can only yield partial information about the answer to the problem; the definitive answer is recovered by repeating the computation multiple times.

We illustrate this approach with an information-theoretic argument (similar to that of Wang *et al.* [24]). Say that we have a quantum routine $\mathcal{A}$ that prepares the state

$$|0^n\rangle \xrightarrow{\mathcal{A}} \sqrt{1-p}|\psi_0\rangle + \sqrt{p}|\psi_1\rangle \tag{17}$$

for some unknown $p \in [0, 1]$, and assume that we can efficiently distinguish between $|\psi_0\rangle$ and $|\psi_1\rangle$. The goal is to estimate p, noting that many query problems can be reduced to estimating an amplitude. With Grover's iterator [32], we can prepare the state

$$\cos[(1+2k)\theta]|\psi_0\rangle + \sin[(1+2k)\theta]|\psi_1\rangle, \tag{18}$$

where $\theta = \arcsin(\sqrt{p})$, with $O(k)$ calls to $\mathcal{A}$. Now suppose that we prepare and measure the state (18) in the $\{|\psi_0\rangle, |\psi_1\rangle\}$ basis l times, recording the outcomes. The Fisher information associated with this experiment is

$$\begin{aligned} I(\pi) &:= l \sum_{i=0,1} \frac{1}{\mathbb{P}[|\psi_i\rangle|\pi]} \left(\frac{\partial}{\partial\pi}\mathbb{P}[|\psi_i\rangle|\pi]\right)^2 \\ &= \frac{l(1+2k)^2}{\pi(1-\pi)}, \end{aligned} \tag{19}$$

where $\mathbb{P}[|\psi_i\rangle|\pi]$ is the probability of observing outcome $|\psi_i\rangle$ in a single trial assuming that $p = \pi$. Expression (19) reveals that the measurement is more informative the larger the value of k (in particular, that it grows quadratically with k, justifying the quadratic speedup of Grover's algorithm).

References [16,25,26] have suggested different schemes to harness the enhanced information of deeper circuits. Outside the query model, circuit cutting [42,43] is also an example of a scheme to approximate a large quantum circuit by independent runs of multiple smaller quantum circuits, with a resulting classical overhead, growing as the quantum circuit is more broken up. Here, we adopt the query model, as has been done so far, and take the perspective put forward by Magano and Murça [26], according to which QSVTs constitute a natural framework for interpolation methods. The idea is to trade off the quality of the polynomial approximation to a target function by statistical sampling. In QSVT, a greater number of coherent quantum queries directly relates to the ability to realize a polynomial transformation over a matrix of larger degree. The scheme of Ref. [26] compensates for the usage of polynomials of lower degree (corresponding to a smaller number of coherent queries in the circuit) by running the quantum circuits more times. The result is a continuous trade-off between circuit depth and quantum speedup, without ever needing to identify independent subproblems.

In the subsequent sections, we consider both approaches (interpolation and parallelization) to achieve computation with query advantage, in a setting where the number of coherent quantum queries allowed are limited. We will illustrate their distinction by presenting examples where either one is advantageous.

IV. SOMETIMES PARALLELIZATION IS BETTER: THRESHOLD FUNCTION

Consider the k-threshold function, a total symmetric Boolean function defined as follows:

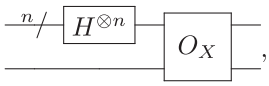
$$\text{Threshold}_k(x_1, \ldots, x_N) = \begin{cases} 0 & \text{if } \sum_{i=1}^N x_i \leqslant k \\ 1 & \text{otherwise} \end{cases}. \tag{20}$$

This function admits a quantum query speedup: Whereas in the classical case $\Theta(N)$ queries are required (easily concluded by an adversarial argument), the quantum query complexity is $\Theta[\sqrt{N \min(k, N-k)}]$ (as follows from Beals *et al.* [27]), resulting in the aforementioned quadratic speed-up when $\min(k, N-k) = O(1)$, and no speedup when $\min(k, N-k) = \Omega(N)$. For simplicity, we assume from now on that $k \leqslant N/2$.

We approach the problem from the perspective of QSVTs. This is a departure from the original proof of Beals *et al.* [27], where the problem of evaluating any totally symmetric Boolean function is reduced to quantum counting. Arguably, QSVTs permit tackling the k-threshold problem more directly, while also offering a more natural route towards interpolation. We show in Appendix B that our approach can also be generalized to any totally symmetric Boolean function, although in that case the proof resembles more closely that of Beals *et al.* [27].

We start by making the (trivial) observation that the k-threshold function can be written as a function of the Hamming weight of the input, which we denote by $|x|$. The first step of our algorithm will be to block-encode $\sqrt{|x|/N}$ (or, more technically, to block-encode the 1×1 matrix whose only entry is $\sqrt{|x|/N}$). Then, we will perform a QSVT on this value to prepare the desired function of $|x|$.

Consider the unitary transformation

$U =$ 

, (21)

where $n = \log_2(N)$—assuming, without loss of generality, that N is exactly a power of two—and O is our query operator (defined in Sec. II A). We have that

$$\begin{aligned} U|0^{n+1}\rangle &= \sqrt{\frac{1}{N}}\left(\sum_{i:\,x_i=0}|i\rangle|0\rangle + \sum_{i:\,x_i=1}|i\rangle|1\rangle\right) \\ &= \sqrt{1-\frac{|X|}{N}}|\phi_0\rangle|0\rangle + \sqrt{\frac{|X|}{N}}|\phi_1\rangle|1\rangle, \end{aligned} \tag{22}$$

where $|\phi_0\rangle$ and $|\phi_1\rangle$ are normalized states. Choosing

$$\Pi = |0^{n+1}\rangle\langle 0^{n+1}| \text{ and } \tilde{\Pi} = \mathrm{I}_{2^n} \otimes |1\rangle\langle 1|, \tag{23}$$

we find that

$$\tilde{\Pi} U \Pi = \sqrt{\frac{|x|}{N}}. \tag{24}$$

That is, $\tilde{\Pi}$, Π, and U form a block encoding of $\sqrt{|x|/N}$.

We would like to distinguish between cases where $\sqrt{|x|/N}$ is smaller than or equal to $\sqrt{k/N}$ and those where it is larger than $\sqrt{k/N}$. From the results on QSVTs (Theorems 1 and 2) we can perform the transformation

$$|0^{n+1}\rangle \to P_{\delta,\eta,\mu}\left(\sqrt{\frac{|x|}{N}}\right)|\phi_1\rangle|1\rangle + |\perp_1\rangle, \tag{25}$$

where $|\perp_1\rangle$ is such that $\tilde{\Pi}|\perp_1\rangle = 0$, using $O[(1/\delta)\ln(1/\eta)]$ calls to U. As U only calls the query operator O once, the operation (25) only involves $O[(1/\delta)\ln(1/\eta)]$ queries. We choose the parameters as

$$\eta = 1/8, \tag{26}$$

$$\delta = \frac{1}{2}(\sqrt{(k+1)/N} - \sqrt{k/N}) = \Theta\left(\frac{1}{\sqrt{kN}}\right), \tag{27}$$

$$\mu = \tfrac{1}{2}(\sqrt{(k+1)/N} + \sqrt{k/N}), \tag{28}$$

in which case the operation (25) consumes $O(\sqrt{kN})$ queries. For the derivation of Eq. (27), see Appendix C. The final step is simply to measure the last qubit of the resulting state, outputting 0 if we measure $|0\rangle$ and outputting 1 if we measure $|1\rangle$. To verify that this yields the desired answer, consider the two possible scenarios:

(i) $\text{Threshold}_k(x_1, \ldots, x_N) = 0$. Then, $\sqrt{|x|/N} \leqslant \sqrt{|k|/N}$, which means that $P_{\delta,\eta,\mu}(\sqrt{|x|/N}) \leqslant \eta = 1/8$. So, the probability of measuring the last qubit in state $|1\rangle$ is less than $1/3$.

(ii) $\text{Threshold}_k(x_1, \ldots, x_N) = 1$. Then, $\sqrt{|x|/N} > \sqrt{|k|/N}$, which means that $P_{\delta,\eta,\mu}(\sqrt{|x|/N}) \geqslant 1 - \eta = 7/8$. So, the probability of measuring the last qubit in state $|1\rangle$ is greater than $2/3$.

If instead $k > N/2$, the algorithm does not change significantly: Denote the logical negation of x by $\bar{x}$, and note that $\text{Threshold}_k(x_1, \ldots, x_N) = 1 - \text{Threshold}_k(\bar{x}_1, \ldots, \bar{x}_N)$. It follows that we just need to evaluate the threshold function on $\bar{x}$, whose Hamming weight is $|\bar{x}| = N - |x|$. Looking at expression (22), we see that U already provides a block encoding of the $\sqrt{|\bar{x}|/N}$: We just need to replace $\tilde{\Pi}$ with $\mathrm{I}_{2^n} \otimes |0\rangle\langle 0|$. Everything else follows as before.

We now proceed to adapt this algorithm to a limitation on the number of allowed coherent quantum queries, according to the schemes laid out in Secs. III B and III A.

A. Interpolation

Consider the approach described in Sec. III B. Recall that the idea is now to implement, with QSVT, a rougher polynomial approximation to a target function. Considering a rougher approximation will allow us to satisfy the coherent quantum query limitations, and will be compensated for by performing a larger number of measurements.

Concretely, the trade-off between circuit depth and repetitions of the circuit can be controlled by the parameter η, which we had previously fixed to be $O(1)$ [cf. (26)]. Now we choose

$$\eta = \Theta(2^{-\delta D}) \tag{29}$$

in such a way that the circuit depth associated with the transformation by $P_{\delta,\eta,\mu}$ is upper bounded by D. If we measure the last qubit of state (25), the probability that we see $|1\rangle$ is

$$\leqslant \eta^2, \quad \text{if } \text{Threshold}_k(x) = 0, \text{ or} \tag{30}$$

$$\geqslant (1-\eta)^2, \quad \text{if } \text{Threshold}_k(x) = 1. \tag{31}$$

So, the problem is reduced to distinguishing the bias of a Bernoulli distribution with precision $1 - 2\eta$. It is well known that $\Theta[1/(1-\eta)^2]$ samples are sufficient (and necessary) to achieve such a precision with bounded-error probability. That is, we prepare and measure state (25)

$$O\left(\frac{1}{(1-\eta)^2}\right) \overset{\text{Eq. (29)}}{=} O\left(\frac{1}{[1-\Theta(2^{-\delta D})]^2}\right) \overset{\delta D = O(1)}{=} O\left(\frac{1}{(\delta D)^2}\right) \tag{32}$$

times. The total number of queries to O is

$$O\left(\underbrace{\frac{1}{(1-\eta)^2}}_{O(1/(\delta D)^2)} \times \underbrace{\frac{1}{\delta}}_{O(1/\delta)} \times \underbrace{\ln\frac{1}{\eta}}_{O(\delta D)}\right) = O\left(\frac{1}{\delta^2 D}\right). \tag{33}$$

Replacing the definition of (27) in Eq. (33), and by means of the calculations given in Appendix C, we conclude that

$$Q(\text{Threshold}_k; D) = O\left(\frac{kN}{D}\right) \quad \text{if } D = O(\sqrt{kN}). \tag{34}$$

Based on the proof above, it might seem that also for $D = \Omega(\sqrt{kN})$, the query complexity would be smaller than $O(\sqrt{kN})$. But it is well known, as stated, that the quantum query complexity for the problem of calculating a threshold function is $Q(\text{Threshold}_k) = \Theta[\sqrt{N\min(k, N-k)}]$. Note the condition in Eq. (32) that $\delta D = O(1)$. If this is not the case, as happens for $D = \Omega(\sqrt{kN})$, then the higher power terms of (δD) will dominate in the expansion of $2^{-\delta D}$, and the conclusion of Eq. (34) is no longer valid. Thus, the query complexity is "saturated" at $O(\sqrt{kN})$, at which point the known amplitude estimation approach is optimal. We conclude

$$Q(\text{Threshold}_K; D) = O\left(\frac{kN}{D} + \sqrt{kN}\right). \tag{35}$$

B. Parallelization

The approach of Ref. [26] was originally developed in the context of phase estimation. In phase estimation the parameter ϕ to be estimated is accessed via a black-box oracle that changes the phase of a particular state by an angle proportional to ϕ. In that case, the interpolation is likely optimal. However, the threshold problem has more structure than phase estimation. Indeed, we can choose to query only a subset of the input variables, in which case the block encoding holds information about the Hamming weight of that subset of input variables, whereas we cannot choose to query a "fractional phase."

It is the parallelization approach that yields the optimal algorithm for evaluating the threshold function in a restricted query-depth setting. To show this, we follow a procedure similar to that of Grover and Radhakrishnan [23]. First, we partition the set $\{1, 2, \ldots, N\}$ into p disjoint subsets

$V_1, \ldots, V_p$ of size N/p (to simplify the notation, we assume that N/p is an integer). Then, for each subset V_i, we prepare the uniform superposition $\sqrt{p/N}\sum_{j\in V_i}|j\rangle|0\rangle$ and apply to it the query operator O. The resulting state is

$$\sqrt{\frac{p|x/\!\!/V_i|}{N}}|\phi_1'\rangle|1\rangle + \sqrt{1-\frac{p|x/\!\!/V_i|}{N}}|\phi_0'\rangle|0\rangle, \tag{36}$$

where $|\phi_0'\rangle$, $|\phi_1'\rangle$ are normalized states and $|x/\!\!/V_i| := |\{x_j \in x : j \in V_i\}|$. If we run the amplitude estimation algorithm of Brassard *et al.* [44] for D steps, we get an estimate of the amplitude $\sqrt{p|x/\!\!/V_i|/N}$ up to precision

$$O\left(\frac{1}{D}\sqrt{\frac{p|x/\!\!/V_i|}{N}}\right) \tag{37}$$

with a constant probability. To lower the probability that the algorithm fails to $1/p$, we repeat the amplitude amplification routine $O(\ln p)$ times; this guarantees a bounded probability that all the amplitude estimations succeed in returning a precision as in (37). We set

$$D = \begin{cases} O\left(\sqrt{\frac{N\ln p}{p}}\right) & \text{if } k \leqslant p\ln p \\ O\left(\frac{\sqrt{Nk}}{p}\right) & \text{if } k \geqslant p\ln p. \end{cases} \tag{38}$$

Then, for every subset V_i, we are estimating $|x/\!\!/V_i|$ with precision

$$\epsilon_i = \begin{cases} O\left(\sqrt{\frac{|x/\!\!/V_i|}{\ln p}}\right) & \text{if } k \leqslant p\ln p \\ O\left(\sqrt{\frac{|x/\!\!/V_i|}{k/p}}\right) & \text{if } k \geqslant p\ln p. \end{cases} \tag{39}$$

We estimate $|x|$ as the sum of our estimates for $|x/\!\!/V_i|$. If it exceeds k, we output 1, and otherwise we output 0.

The actual behavior of the algorithm depends on how the 1-input variables are distributed among the subsets $V_1, \ldots, V_p$. In the worst-case scenario, all the ones are concentrated in a single bin. However, this scenario is extremely unlikely. Raab and Steger's "balls into bins" theorem [45] states that, with probability greater than 2/3,

$$\max_i |x/\!\!/V_i| = \begin{cases} O(\ln p) & \text{if } |x| \leqslant p\ln p \\ O\left(\frac{|x|}{p}\right) & \text{if } |x| \geqslant p\ln p. \end{cases} \tag{40}$$

Using this result, we show in Appendix A that there is a choice for the constant factors in (38) that guarantees that our estimate for $|x|$ is larger than k if $\text{Threshold}_k(x) = 1$ and smaller than or equal to k if $\text{Threshold}_k(x) = 0$.

Putting everything together, we conclude that

$$Q(\text{Threshold}_k; D) = O\left[\frac{N}{D}\ln^2\left(\frac{N}{D}\right) + \sqrt{Nk}\ln k\right]. \tag{41}$$

Comparing with the upper bound that we derived with the interpolation method [Eq. (35)], we see that parallelization offers the best performance. Indeed, for short circuit query depths the complexity of the parallelization method is smaller by a factor of k (up to logarithmic factors).

V. SOMETIMES INTERPOLATION IS BETTER: NAND TREES

We now apply the interpolation and parallelization techniques for the problem of evaluating a balanced binary NAND formula. This problem has been widely studied in the literature: Farhi *et al.* [46] proposed a quantum walk algorithm that runs in $O(N^{1/2})$ time with an unconventional, continuous-time query model. Later, Childs *et al.* [28] understood that this algorithm could be translated into the discrete query model (as presented in Sec. II A) with just an $O(N^{o(1)})$ overhead. Finally, Ambainis *et al.* [29] presented an optimal $O(N^{1/2})$-time algorithm on the conventional query model. We adapt their approach to a restricted query-depth setting.

Let Φ be a Boolean function on N inputs $x_1, \ldots, x_N$ expressed with NAND gates. We treat each occurrence of a variable separately, in that N is counting with the variables' multiplicity. Equivalently, we could be considering a formula expressed in terms of the gate set {AND, OR, NOT}. The input is accessed via the conventional query operator O as defined in Sec. II A.

The formula Φ can be represented by a tree, where the internal nodes are NAND gates acting on their children and the leafs hold the input variables. Here, we restrict our attention to formulas that are represented by perfectly balanced binary trees. We note that Ambainis *et al.*'s algorithm can be applied to general formulas after a proper rebalancing of the corresponding tree [47,48]. Similarly, our arguments could also be extended to the general case.

Ambainis *et al.* [29] prove that (after efficient classical preprocessing) $\Phi(x)$ can be evaluated with bounded-error probability using $\sqrt{N}$ queries to O. The main idea is to build a weighted graph whose adjacency matrix, denoted as H, has a spectrum that relates to the value of $\Phi(x)$. Then, one simulates a discrete-time quantum walk on this graph. By applying a phase estimation on this process for a special starting state, one is able to infer the value of $\Phi(x)$.

We present a succinct definition of H, referring the reader to the original paper [29] for a more detailed explanation. We construct a symmetric weighted graph from the formula's tree, attaching to the root node (call it r) a tail of two nodes, r' and r''. For each node v, let s_v be the number of variables of the subformula rooted at v. The weights on the graph are defined in the following manner. If p is the parent of a node v, then

$$\langle v|H|p\rangle := \left(\frac{s_v}{s_p}\right)^{1/4}, \tag{42}$$

with two exceptions:

(1) if v is a leaf reading 1, then $\langle v|H|p\rangle := 0$ [effectively removing the edge (v, p) from the graph];

(2) $\langle r'|H|r''\rangle := 1/(\sqrt{2}N^{1/4})$.

The spectrum of H has the following properties [[29], Theorem 2]:

(1) if $\Phi(x) = 0$, then there is a zero-eigenvalue eigenstate $|g\rangle$ of H $|\langle r''|g|\geqslant 1/\sqrt{2}$;

(2) if $\Phi(x) = 1$, then every eigenstate with support on $|r''\rangle$ has eigenvalue at least $1/(18\sqrt{2N})$ in absolute value.

That is, we can evaluate Φ by determining whether $|r''\rangle$ has a large zero-eigenvalue component.

A. Interpolation

Starting on the graph construction of Ambainis *et al.* [29], we present a different, QSVT-based approach to infer the value of $\Phi(x)$, circumventing the quantum walk and phase estimation steps. With the aforementioned principle of trading off lower degree polynomial approximations by longer statistical sampling, this will allow us to derive an interpolating algorithm for evaluating general NAND trees.

We begin by constructing a block encoding of H. As H has a bounded degree and the weights of its edges are upper bounded by 1, we can use standard block-encoding techniques for sparse matrices [18,37]. Namely, for projectors

$$\Pi, \tilde{\Pi} = |0^m\rangle\langle 0^m|, \tag{43}$$

with $m = O(\ln N)$, there is a unitary U_H that block-encodes $H/3$ with $O(1)$ calls to O. By definition, the unitary U_H is such that, for an arbitrary state $|\psi\rangle$,

$$U_H|0^m\rangle|\psi\rangle = |0^m\rangle\left(\frac{H}{3}|\psi\rangle\right) + |\perp\rangle, \tag{44}$$

where $|\perp\rangle$ is orthogonal to $|0^m\rangle$.

We would like to distinguish between the eigenstates of $H/3$ whose eigenvalue is close to zero and those whose eigenvalue is larger than

$$\frac{1}{3} \times \frac{1}{18\sqrt{2N}} =: \delta \tag{45}$$

in absolute value. We treat this as a QSVT problem, as discussed in Sec. II B. Indeed, let $\{\lambda_i, |v_i\rangle\}_i$ be an eigenvalue decomposition of $H/3$ and $|\psi\rangle = \sum_i \alpha_i|v_i\rangle$ be an arbitrary state. From Theorem 1 and Corollary 1, we can perform the transformation

$$\begin{aligned} |0^m\rangle|\psi\rangle &= |0^m\rangle\left(\sum_i \alpha_i|v_i\rangle\right) \\ &\to |0^m\rangle\left(\sum_i P'_{\delta,\eta,\mu}(\lambda_i)\alpha_i|v_i\rangle\right) + |\perp\rangle, \end{aligned} \tag{46}$$

where $P'_{\delta,\eta,\mu}$ is an approximation to the window function (as defined in Corollary 1), with $O[(1/\delta)\ln(1/\eta)]$ queries to O.

We now have all the necessary tools to solve the problem. We start by preparing the state $|r''\rangle$ (this does not involve any oracle queries). We then transform $|r''\rangle$ as in (46). We measure the m first qubits (i.e., the block-encoding register) of the resulting state, assigning an outcome "yes" if we observe $|0^m\rangle$ and an outcome "no" otherwise. From the spectral properties of H we known that

$$\mathbb{P}[\text{"yes"}]\begin{cases} \geqslant \frac{(1-\eta)^2}{2}, & \text{if } \Phi(x) = 0 \\ \leqslant \eta^2, & \text{if } \Phi(x) = 1 \end{cases}. \tag{47}$$

So, we need to determine the bias of a Bernoulli distribution with precision no larger than $(1-\eta)/4$. It is well known that $O[1/(1-\eta)^2]$ samples are sufficient (and necessary) to achieve such a precision with bounded-error probability. In summary, we can evaluate $\Phi(x)$ with bounded-error probability by running $O[(1/\delta)\ln(1/\eta)]$-deep circuits $O[1/(1-\eta)^2]$ times, amounting to a total of

$$O\left[\frac{1}{(1-\eta)^2} \times \frac{1}{\delta}\ln\left(\frac{1}{\eta}\right)\right] \tag{48}$$

queries to O.

We have purposely left η as a free parameter in our algorithm. We get the best possible complexity by choosing $\eta = 1 - \Omega(1)$, in which case the algorithm's query complexity is [using definition (45)]

$$O\left(\frac{1}{\delta}\right) = O(\sqrt{N}), \tag{49}$$

recovering the scaling of Ambainis *et al.* [29]. But this choice of η requires running circuits of (query) depth also in $O(\sqrt{N})$. Suppose now that we want to limit the circuit depth to some maximum value D. We can run the same algorithm, setting this time η to be

$$\eta = O(2^{-\delta D}). \tag{50}$$

Replacing into expression (48), we find that

$$Q(\Phi; D) = O\left(\frac{N}{D} + \sqrt{N}\right). \tag{51}$$

B. Parallelization

The problem of evaluating NAND trees is also amenable to parallelization. The key observation is that, if for any given level of the tree we know the logical value of all the nodes at that level, then we can infer $\Phi(x)$ without performing any more queries to the input. Therefore, we solve the problem if, for every node v at that level, we run the quantum algorithm for evaluating the NAND tree rooted at v.

Say that we want to limit to at most D coherent calls to the query. We partition the input variables into $O(N/D^2)$ subsets of $O(D^2)$ variables each. To each subset of variables corresponds a subtree of the total tree. For each such subtree, we evaluate the logical value of the root node with an error probability bounded by D^2/N, which we can do with $O[\sqrt{D^2}\ln(N/D^2)]$ queries to O. Since we repeat this for all subtrees, the hybrid query complexity becomes

$$Q(\Phi; D) = O\left[\frac{N}{D}\ln\left(\frac{N}{D}\right) + \sqrt{N}\right]. \tag{52}$$

We find that both the interpolation and parallelization methods can be applied for evaluating balanced binary NAND trees. Although the resulting complexities are close, the parallelization approach comes with an extra $\ln(N/D)$ factor. This problem illustrates that there are also situations where interpolation is advantageous over parallelization.

VI. CONCLUSIONS

In this paper, we suggest two distinct approaches for adapting a quantum algorithm to a restricted query-depth setting: Parallelization and interpolation. An algorithm is said to be parallelizable whenever we can split its action into smaller, independent subproblems; and interpolatable if the loss of

information caused by shortening the circuit query depth can be compensated for by repeated runs of the shorter circuit. Therefore, informally, these two methods can be understood as either breaking up the input (for parallelization) or breaking up the unitary procedure (for interpolation).

We argue that QSVTs closely relate to the notion of interpolation, rather than parallelization. For QSVTs, a smaller circuit query depth, and thus a smaller circuit depth, correspond to a polynomial approximation to a target function of lower degree, which needs to be compensated for by longer statistical sampling.

We apply these approaches to two problems with known quantum speed-ups: The k-threshold function and perfectly balanced NAND trees. To the best of our knowledge, neither of these problems had been studied in a hybrid, restricted query-depth setting. For the k-threshold function, we show that parallelization offers the best performance by a factor of $\tilde{O}(k)$ (in terms of query complexity). In contrast, for evaluating perfectly balanced NAND trees the interpolation method is the most efficient, differing by a factor of $O[\ln(N/D)]$. This way, we demonstrate that no technique (parallelization or interpolation) is strictly better than the other—each one may be the best option depending on the problem at hand.

This shows that, when designing a quantum-classical hybrid algorithm obeying certain (query) depth limitations, both of the proposed techniques can be explored as a strategy for maintaining some of the speedup (over a fully classical approach) of a quantum unrestricted query-depth counterpart. While we have ruled out that one of the two methods is always optimal, it remains to understand whether there is a criterion to determine which method is optimal *a priori* for a given problem, which we leave as an open question. In any case, given the close connection between (depth unrestricted) algorithms formulated in terms of QSVTs and the interpolation method, our results imply that, when searching for hybrid quantum-classical algorithms for a particular problem, it may be a good option to start by formulating a (depth unrestricted) QSVT algorithm for the problem, and then seeking to interpolate it.

We note that we only offered an example of a problem (perfectly balanced NAND trees) where the interpolation beats parallelization by a logarithmic factor. It would be interesting to find a problem for which the interpolation procedure is polynomially more efficient than the corresponding parallelization, to rule out the possibility that parallelization, whenever applicable, is always optimal up to logarithmic factors. We leave the existence of such a problem as an open question. In any case, if disregarding the logarithmic separation, we nonetheless conclude that the method of interpolation may achieve the same performance (in the query model) as parallelization, while exploiting a different mechanism to fulfil the query depth limitations.

The definitions we have provided for the terms "parallelization" and "interpolation" are not strictly rigorous; they should be seen as general strategies, rather than formal notions. This does not preclude that in some situations these classifications may not apply. As such, we expect there is room for discussion on what other classes of methods may exist besides the ones discussed here, and for other systematic approaches to hybridization.

ACKNOWLEDGMENTS

We thank R. de Wolf for his comments on quantum query lower bounds for the problem of quantum counting, in the context of calculating the threshold function, and N. Stamatopoulos for his comments regarding the proof of the parallelization method for the threshold function. We also acknowledge support from FCT – Fundação para a Ciência e a Tecnologia (Portugal), namely, through Project No. UIDB/04540/2020, as well as from projects QuantHEP and HQCC supported by the EU QuantERA ERA-NET Cofund in Quantum Technologies and by FCT (QuantERA/0001/2019 and QuantERA/004/2021, respectively), and from the EU Horizon Europe Quantum Flagship project EuRyQa (Project No. 101070144). D.M. and M.M. acknowledge the support from FCT through scholarships 2020.04677.BD and 2021.05528.BD, respectively.

M.M. and D.M. contributed equally to this work.

APPENDIX A: THRESHOLD FUNCTION—PROOF OF PARALLELIZATION METHOD

From Brassard *et al.* [44], there is a constant c such that the error for our estimate of $|x/\!\!/V_i|$ is bounded as

$$\epsilon_i < c\frac{\sqrt{N|x/\!\!/V_i|/p}}{D}. \tag{A1}$$

We analyze separately the cases where $\text{Threshold}_k(x) = 0$ and $\text{Threshold}_k(x) = 1$.

If $\text{Threshold}_k(x) = 0$, the following possible relations between p, k, and $|x|$ need to be considered.

(1) $p\ln p \leqslant |x| \leqslant k$. From the result of Raab and Steger [Eq. (40)], we know that $|x/\!\!/V_i| = O(|x|/p)$ for all i. So, by our expression for the error (39), we see that $\epsilon_i = O(\sqrt{|x|/k}) = O(1)$.

(2) $|x| \leqslant p\ln p \leqslant k$. Now, we know that $|x/\!\!/V_i| = O(\ln p)$. So, for all i, $\epsilon_i = O(\sqrt{p\ln p/k}) = O(1)$.

(3) $|x| \leqslant k \leqslant p\ln p$. Equation (40) ensures that $|x/\!\!/V_i| = O(\ln p)$. From the expression for the error, we see that $\epsilon_i = O(\sqrt{\ln p/\ln p}) = O(1)$.

That is, there is a choice of constants that guarantees that $\epsilon_i < 1/2$ for all i with bounded probability. In that case, we estimate each $|x/\!\!/V_i|$ exactly, and so we exactly infer $|x|$ and consequently the value of $\text{Threshold}_k(x)$.

If $\text{Threshold}_k(x) = 1$, the proof is slightly different. Again, we consider three scenarios.

(1) $p\ln p \leqslant k \leqslant |x|$. Equation (40) tells us that $|x/\!\!/V_i| = O(|x|/p)$. Combining with (39), we see that there is a (controllable) constant C for which

$$\epsilon_i < C\sqrt{\frac{|x|}{k}} \tag{A2}$$

for all $|x|, k$. Unlike before, we cannot guarantee that ϵ_i is kept below $1/2$ for all $|x|$. But we can make sure that our estimate for the Hamming weight is always greater than k. Let X_j be the random variables corresponding to the estimations of each $|x/\!\!/V_j|$, and σ_j^2 the corresponding variances. From the

Chebyshev bound,

$$\Pr\left[\left|\sum_j X_j - |x|\right| > |x| - k\right] < \left|\frac{\sum_j \sigma_j}{|x|-k}\right|^2 < \left|C'\, p\,\frac{\sqrt{|x|/k}}{|x|-k}\right|^2 \tag{A3}$$

for some constant C'. Thus, we can attain with constant probability an estimation of $|x|$ with error within $|x| - k$ if there exists a constant C' such that there exists a value $|x|^*$ satisfying the following:

(i) If $|x| < |x|^*$, the error in the estimation of each $|x/\!\!/V_j|$ is less than $1/2$, such that the estimate of $|x|$ is exact,

(ii) If $|x| > |x|^*$, then $C'\sqrt{|x|/k} < (|x| - k)/p$, bounding the error probability to be constant.

Choosing $|x|^* = k + p \ln p$, one can check that $C = 1/4(C')$ satisfies the conditions above.

(2) $k \leqslant p \ln p \leqslant |x|$. Again, for all i, $|x/\!\!/V_i| = O(|x|/p)$. Combining this with the expression for the error (39), we get $\epsilon_i = O(|x|/p \ln p)$. The proof follows the same steps as the $p \ln p \leqslant k \leqslant |x|$ case.

(3) $k \leqslant |x| \leqslant p \ln p$. From Eq. (40) we know that $|x/\!\!/V_i| = O(\ln p)$ for all i. Then, $\epsilon_i = O(\sqrt{\ln p/\ln p}) = O(1)$. So, in this case we can also ensure that we estimate $\sum_i |x/\!\!/V_i|$ exactly.

APPENDIX B: TOTAL NONCONSTANT SYMMETRIC BOOLEAN FUNCTIONS

We have shown before how to interpolate the k-threshold function based on quantum singular value transformations. A similar interpolation scheme to the one we have shown can actually be applied to the calculation of any symmetric Boolean function, as we now show. Furthermore, we show that a similar difference exists between the scaling for this interpolation and the scaling for a parallelization procedure.

We start by reviewing an intermediate claim of Beals *et al.* [27]:

Lemma 1. (Part of Theorem 4.10 of Beals *et al.* [27]) For a symmetric Boolean function f, if given an algorithm that outputs $|X|$ if $|X| < [N - \Gamma(f)]/2$ or outputs "in" otherwise, with Q queries to the oracle, immediately there is an algorithm that computes f with Q queries to the oracle.

Proof. Let $\mathcal{A}$ be an algorithm as outlined in the lemma, requiring Q queries to the oracle. By definition of $\Gamma(f)$, f is constant for X such that $|X| \in \{[N - \Gamma(f)]/2, [N + \Gamma(f)]/2\}$. Therefore, let $\mathcal{A}'$ be an algorithm that runs $\mathcal{A}$, and then

(i) If $\mathcal{A}$ outputs "in", $\mathcal{A}'$ outputs $f\{[N - \Gamma(f)]/2\}$,

(ii) If $\mathcal{A}$ outputs $|X|$, $\mathcal{A}'$ outputs $f(|X|)$.

$\mathcal{A}'$ requires only as many queries as $\mathcal{A}$.

Now, departing from Beals *et al.*'s proof, we rephrase the construction of an algorithm matching the description of lemma 1 in terms of quantum singular value transformations.

We start with the following lemma of Low and Chuang [49]:

Lemma 2. [49] For a given $k \in \mathbb{R}$, $\delta \in [-1, 1]$ and $\epsilon \in [0, O(1)]$, there exists a real polynomial $p(x)$ satisfying

$$|p(x)| \leqslant 1\,, \quad x \in [-1, 1], \text{ and}$$
$$|p(x) - \mathrm{erf}[k(x - \delta)]| \leqslant \epsilon\,, \quad x \in [-1, 1]$$

with polynomial degree

$$\deg(p) = O\left(\sqrt{\left(\ln\frac{1}{\epsilon}\right)\left(k^2 + \ln\frac{1}{\epsilon}\right)}\right). \tag{B1}$$

From this lemma follows the already mentioned construction for a polynomial approximation to the threshold function, which we restate:

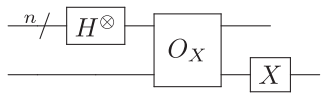
Corollary 2. [49] For a given $\delta \in [-1, 1]$, $\epsilon \in [0, O(1)]$, $\eta \in (0, 1/4)$, there exists a real polynomial p satisfying

$$|p(x)| \leqslant 1\,, \quad x \in [-1, 1]$$
$$|p(x) - 1| \leqslant \eta\,, \quad x \in [-1, \delta - \epsilon],$$
$$|p(x)| \leqslant \eta\,, \quad x \in [\delta + \epsilon, 1],$$

and with polynomial degree

$$\deg(p) = O\left(\frac{1}{\epsilon}\ln\frac{1}{\eta}\right).$$

To make use of these polynomial transformations, we also recall the block encoding of the quantities of interest, which are the same as for the k-threshold case; for unitary

$U =$ 

and $\Pi = |0^{n+1}\rangle\langle 0^{n+1}|$, we have that $(\mathrm{I}_{2^n} \otimes |1\rangle\langle 1|)U\Pi$ is a block encoding of $\sqrt{|X|/N}$, and $(\mathrm{I}_{2^n} \otimes |0\rangle\langle 0|)U\Pi$ is a block encoding of $\sqrt{(N - |X|)/N}$.

Now we first determine whether the Hamming weight of the input should produce output "in" or not, which is, essentially, the task of calculating the k-threshold function with $k = [N - \Gamma(f)]/2$ and with $k' = [N + \Gamma(f)]/2$. As stated in the body text, the case of threshold k' can be reduced to the case of threshold k calculated for the complement of the Hamming weight $N - |X|$, and so we conclude that this step requires $O\{2\sqrt{N[N - \Gamma(f)]}\} = O\{\sqrt{N[N - \Gamma(f)]}\}$ applications of the oracle.

In the event that we find $|X|$ to be smaller than $[N - \Gamma(f)]/2$, or larger than $[N + \Gamma(f)]/2$, it remains to output the Hamming weight of X, or of $\bar{X}$, respectively. We consider henceforth the case of $|X| < [N - \Gamma(f)]/2$, from which generalization is easy.

Note first that performing bisections on $|X|$ for $|X| \in \{0, [N - \Gamma(f)]/2\}$ corresponds to performing successive threshold operations for thresholds $k' < [N - \Gamma(f)]/2$, so, by binary search, we have that we may find $|X|$ with $O\{\sqrt{N[N - \Gamma(f)]}\ln^2[N - \Gamma(f)]\}$ applications of the oracle, where one of the ln factors is due to the binary search, and the other to error probability bounding. However, by making direct use of lemma 2, the ln factors can be significantly lowered. Consider the following lemma:

Lemma 3. Given the block encoding of a value $z \in [a, b] \subseteq [-1, 1]$, it is possible to determine $[a', b'] \subseteq [a, b]$

such that $z \in [a', b']$, and $(b' - a') \leqslant (b - a)/2$, with

$$D_{\text{round}} = O\left(\frac{1}{b-a}\right) \tag{B2}$$

coherent applications of the oracle, and

$$T_{\text{round}} = O\left(\frac{1}{b-a}\ln\frac{1}{E}\right) \tag{B3}$$

total applications of the oracle, with probability of error at most E.

Proof. Fix $\eta \in O(1)$, for example, $\eta = 1/8$. Using QSVT, create a block encoding of $P(z)$, where P is the polynomial approximating $\text{erf}[k(x-\delta)]$ up to absolute error ϵ (to be determined), with $k = \frac{2}{b-a}\text{erf}^{-1}(1-2\eta)$ and $\mu = (b-a)/2$. For a choice of σ, after $O[\ln(1/E)/\sigma^2]$ samples of this encoding, one obtains an estimate for $\text{erf}[k(z-\mu)]^2$ up to precision $\sigma + \epsilon^2$ with error probability E; denote this estimate $\tilde{p}$. This estimate implies a new window $[a', b']$ for z satisfying

$$\begin{aligned} b' - a' \leqslant \frac{1}{k}\{&\text{erf}^{-1}[2\sqrt{\tilde{p}} - 1 + 2(\sigma+\epsilon)] \\ &- \text{erf}^{-1}[2\sqrt{\tilde{p}} - 1 - 2(\sigma+\epsilon)]\}, \end{aligned} \tag{B4}$$

which in turn satisfies, with our choice of k,

$$\begin{aligned} \frac{b'-a'}{b-a} \leqslant &\frac{2}{\text{erf}^{-1}(1-2\eta)}(\sigma+\epsilon) \\ &\times \max_{y\in[-2,2]} (\text{erf}^{-1})'[2\sqrt{\tilde{p}} - 1 + y(\sigma+\epsilon)]. \end{aligned} \tag{B5}$$

Demanding that

$$\sigma + \epsilon \overset{!}{\leqslant} \eta/4, \tag{B6}$$

we have

$$\frac{b'-a'}{b-a} \leqslant \frac{\eta}{4}\frac{\sqrt{\pi}}{\text{erf}^{-1}(1-2\eta)} e^{[\text{erf}^{-1}(1-\eta)]2} \tag{B7}$$

which, for $\eta = 1/8$, has a right-hand side of less than one half.

Since $\eta \in O(1)$, we may choose $\sigma \in O(1)$ and $\epsilon \in O(1)$, satisfying the constraint (B6), and thus it follows that this procedure requires

$$M_{\text{round}} = O\left(\frac{1}{\sigma^2}\ln\frac{1}{E}\right) = O\left(\ln\frac{1}{E}\right) \tag{B8}$$

measurements of a circuit encoding a polynomial transformation of degree [cf. Eq. (B1)]

$$D_{\text{round}} = O(k) = O\left(\frac{1}{b-a}\right) \tag{B9}$$

or, equivalently, the same number of coherent queries. The total number of queries is therefore

$$T_{\text{round}} = D_{\text{round}} M_{\text{round}} = O\left(\frac{\ln E^{-1}}{b-a}\right) \tag{B10}$$

as claimed.

By repeating the procedure given in the lemma above, we may reduce the window for $\sqrt{|X|/N}$ until this value is unambiguous. This requires a final window of size $\Delta = \frac{1}{2}[\sqrt{N-\Gamma(f)} - \sqrt{N-\Gamma(f)-1}]$, which in turn requires $\ln[\sqrt{N-\Gamma(f)}/\Delta] = O(\ln\{\sqrt{N}[N-\Gamma(f)]\})$ rounds of application of the lemma.

Since we wish any of these rounds to fail with probability at most $1/3$, this requires that each round fails with probability at most $1/(3\ln\{\sqrt{N}[N-\Gamma(f)]\})$.

Therefore, overall, this procedure requires

$$D = O(\sqrt{N[N-\Gamma(f)]}) \tag{B11}$$

maximum coherent oracle calls, and a total number of oracle calls

$$T = O(\sqrt{N-\Gamma(f)}\ln\ln\{\sqrt{N}[N-\Gamma(f)]\}). \tag{B12}$$

Performing the interpolation now is straightforward: Instead of demanding the procedure from lemma 3 be repeated until the window is so small that the Hamming weight of X is unambiguous, we instead choose a final window size Δ' that respects the given coherent query limit. After this limit has been reached, we "switch" to statistical sampling until the final window size for the value of $\sqrt{|X|/N}$ is Δ. This procedure therefore is split into two steps; following an analysis analogous to the one for the unbound case, we conclude that for some choice of Δ', the first phase requires maximum coherent query depth

$$D_{\text{first}} = O\left(\frac{1}{\Delta'}\right) \tag{B13}$$

and total query count

$$T_{\text{first}} = O\left(\frac{\ln E^{-1}}{\Delta'}\right). \tag{B14}$$

Using the fact that $(\text{erf}^{-1})'(x) \geqslant \sqrt{\pi}/2$, one may then conclude that the second phase requires corresponding

$$D_{\text{second}} = O\left[\frac{1}{\Delta'}\sqrt{\ln\left(C\frac{\Delta'}{\Delta}\right)}\right] \tag{B15}$$

$$T_{\text{second}} = O\left[\frac{\Delta'}{\Delta^2}\sqrt{\ln\left(C\frac{\Delta'}{\Delta}\right)}\ln E^{-1}\right], \tag{B16}$$

where, again, E is the error probability, and C is a constant in $O(1)$.

Choosing this intermediate window size Δ' to be $\Delta^{1-\alpha}$, for $\alpha \in [0, 1]$, we recover complexities analogous to those verified for α-quantum phase estimation [25,26]:

$$D(\alpha) = D_{\text{first}} + D_{\text{second}} = O[\Delta^{\alpha-1}\sqrt{\ln(C\Delta^{-\alpha})}] \tag{B17}$$

$$\begin{aligned} T(\alpha) &= T_{\text{first}} + T_{\text{second}} \\ &= O[\Delta^{-(1+\alpha)}\sqrt{\ln(C\Delta^{-\alpha})}\ln E^{-1}]. \end{aligned} \tag{B18}$$

Using again the fact that $\Delta^{-1} = O\{\sqrt{N[N-\Gamma(f)]}\}$, and with considerations as to the error probability identical to before, we finally have

$$D(\alpha) = \tilde{O}(\{N[N-\Gamma(f)]\}^{(1-\alpha)/2}) \tag{B19}$$

$$T(\alpha) = \tilde{O}(\{N[N-\Gamma(f)]\}^{(1+\alpha)/2}). \tag{B20}$$

APPENDIX C: PROOF OF $\sqrt{k+1}-\sqrt{k}=\Theta(\frac{1}{\sqrt{k}})$

We start with the lower bound:

$$\sqrt{k+1}-\sqrt{k}=\frac{\sqrt{k(k+1)}-k}{\sqrt{k}}. \tag{C1}$$

Clearly the numerator is positive and smaller than 1 for all $k=1,\ldots,N$. Thus,

$$\sqrt{k+1}-\sqrt{k}=O\left(\frac{1}{\sqrt{k}}\right). \tag{C2}$$

To show that the bound is tight, it suffices to show that

$$\frac{\frac{1}{\sqrt{k}}}{\sqrt{k+1}-\sqrt{k}}=\frac{1}{\sqrt{k(k+1)}-k}=O(1). \tag{C3}$$

The denominator in that expression is monotonically growing with k, for any integer $k>0$. Therefore,

$$\frac{1}{\sqrt{k(k+1)}-k}\leqslant\frac{1}{\sqrt{2}-1}=O(1). \tag{C4}$$

This concludes the lower bound proof.

The results stated in Eqs. (27) and (34) follow directly from the demonstration above.

[1] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, Noisy intermediate-scale quantum algorithms, Rev. Mod. Phys. **94**, 015004 (2022).

[2] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028.

[3] A. Peruzzo, J. McClean, P. Shadbolt, M. H. Yung, X. Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. **5**, 4213 (2014).

[4] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. **18**, 023023 (2016).

[5] D. Wecker, M. B. Hastings, and M. Troyer, Progress towards practical quantum variational algorithms, Phys. Rev. A **92**, 042303 (2015).

[6] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature (London) **549**, 242 (2017).

[7] E. Farhi and H. Neven, Classification with quantum neural networks on near term processors, arXiv:1802.06002.

[8] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, Quantum Sci. Technol. **4**, 043001 (2019).

[9] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, Variational quantum circuits for deep reinforcement learning, IEEE Access **8**, 141007 (2020).

[10] L. Banchi, Robust quantum classifiers via NISQ adversarial learning, Nat. Comput. Sci. **2**, 699 (2022).

[11] L. Buffoni and F. Caruso, New trends in quantum machine learning, Europhys. Lett. **132**, 60004 (2020).

[12] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nat. Commun. **9**, 4812 (2018).

[13] E. R. Anschuetz and B. T. Kiani, Quantum variational algorithms are swamped with traps, Nat. Commun. **13**, 7760 (2022).

[14] X. Sun and Y. Zheng, Hybrid decision trees: Longer quantum time is strictly more powerful, arXiv:1911.13091.

[15] S. Aaronson and A. Ambainis, Forrelation: A problem that optimally separates quantum from classical computing, SIAM J. Comput. **47**, 982 (2018).

[16] D. Wang, O. Higgott, and S. Brierley, Accelerated variational quantum eigensolver, Phys. Rev. Lett. **122**, 140504 (2019).

[17] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (ACM, New York, 2019).

[18] L. Lin, Lecture notes on quantum algorithms for scientific computation, arXiv:2201.08309.

[19] D. Deutsch and R. Jozsa, Rapid solution of problems by quantum computation, Proc. R. Soc. London A **439**, 553 (1992).

[20] P. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (IEEE Comput. Soc. Press, Santa Fe, NM, USA, 1994).

[21] A. Pérez-Salinas, R. Draškić, J. Tura, and V. Dunjko, Shallow circuits for deeper problems, Phys. Rev. A **108**, 062423 (2023).

[22] C. Zalka, Grover's quantum searching algorithm is optimal, Phys. Rev. A **60**, 2746 (1999).

[23] L. K. Grover and J. Radhakrishnan, Quantum search for multiple items using parallel queries, arXiv:quant-ph/0407217.

[24] G. Wang, D. E. Koh, P. D. Johnson, and Y. Cao, Minimizing estimation runtime on noisy quantum computers, PRX Quantum **2**, 010346 (2021).

[25] T. Giurgica-Tiron, I. Kerenidis, F. Labib, A. Prakash, and W. Zeng, Low depth algorithms for quantum amplitude estimation, Quantum **6**, 745 (2022).

[26] D. Magano and M. Murça, Simplifying a classical-quantum algorithm interpolation with quantum singular value transformations, Phys. Rev. A **106**, 062419 (2022).

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

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

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

[30] A. Ambainis, Understanding algorithms via query complexity, in *Proceedings of the International Congress of Mathematicians (ICM 2018)* (World Scientific, Singapore, 2019), pp. 3265–3285.

[31] H. Buhrman and R. de Wolf, Complexity measures and decision tree complexity: A survey, Theor. Comput. Sci. **288**, 21 (2002).
[32] L. K. Grover, Quantum mechanics helps in searching for a needle in a haystack, Phys. Rev. Lett. **79**, 325 (1997).
[33] A. Ambainis, Quantum walk algorithm for element distinctness, SIAM J. Comput. **37**, 210 (2007).
[34] A. Ambainis, Quantum lower bounds by quantum arguments, J. Comput. Syst. Sci. **64**, 750 (2002).
[35] J. M. Martyn, Z. M. Rossi, A. K. Tan, and I. L. Chuang, Grand unification of quantum algorithms, PRX Quantum **2**, 040203 (2021).
[36] G. H. Low and I. L. Chuang, Hamiltonian simulation by qubitization, Quantum **3**, 163 (2019).
[37] S. Chakraborty, A. Gilyén, and S. Jeffery, The power of block-encoded matrix powers: Improved regression techniques via faster Hamiltonian simulation, in *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)* (Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2019), pp. 33:1–33:14.
[38] By Π-controlled-NOT we mean an operation that acts as a NOT gate controlled on a given state being in the image of $\Pi/\tilde{\Pi}$.
[39] G. H. Low, Quantum signal processing by single-qubit dynamics, Ph.D. thesis, Massachusetts Institute of Technology, 2017.
[40] S. Jeffery, F. Magniez, and R. Wolf, Optimal parallel quantum query algorithms, Algorithmica **79**, 509 (2017).
[41] W. M. Kirby and P. J. Love, Variational quantum eigensolvers for sparse Hamiltonians, Phys. Rev. Lett. **127**, 110503 (2021).
[42] S. Bravyi, G. Smith, and J. A. Smolin, Trading classical and quantum computational resources, Phys. Rev. X **6**, 021043 (2016).
[43] T. Peng, A. W. Harrow, M. Ozols, and X. Wu, Simulating large quantum circuits on a small quantum computer, Phys. Rev. Lett. **125**, 150504 (2020).
[44] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, Quantum amplitude amplification and estimation, Quantum Inf. Comput. **305**, 53 (2002).
[45] M. Raab and A. Steger, 'Balls into bins" — A simple and tight analysis, in *Randomization and Approximation Techniques in Computer Science*, edited by M. Luby, J. D. P. Rolim, and M. Serna (Springer, Berlin, Heidelberg, 1998), pp. 159–170.
[46] E. Farhi, J. Goldstone, and S. Gutmann, A quantum algorithm for the Hamiltonian NAND tree, Theory of Computing **4**, 169 (2008).
[47] N. H. Bshouty, R. Cleve, and W. Eberly, Size-depth tradeoffs for algebraic formulas, SIAM J. Comput. **24**, 682 (1995).
[48] M. L. Bonet and S. R. Buss, Size-depth tradeoffs for Boolean formulae, Inf. Process. Lett. **49**, 151 (1994).
[49] G. H. Low and I. L. Chuang, Hamiltonian simulation by uniform spectral amplification, arXiv:1707.05391.