An Overview of Quantum Algorithms: from Quantum Supremacy to Shor Factorization

Subhasree Patro
QuSoft, CWI
Amsterdam, The Netherlands
Email: subhasree.patro@cwi.nl

Alvaro Piedrafita
QuSoft, CWI
Amsterdam, The Netherlands
Email: piedrafita@cwi.nl

***Abstract*—Recently, a team of scientists from Google claims to have carried a computation on their noisy, intermediate-scale quantum (NISQ) computer which no regular computer can achieve. A feat that is sometimes referred as *quantum supremacy*. In the first part of this work, we explain their approach, their randomised circuit construction and the consequences of their work. Having achieved this milestone, the natural question becomes: what else can we do with a quantum computer? We answer this question in the second part of this work and give an overview of the most widely used quantum primitives. We will see how one can implement them using quantum circuits and how these primitive circuits are composed to create fast algorithms capable of solving commercially relevant problems like simulation of complicated quantum systems for chemistry and material science, Monte Carlo simulation, solving large systems of linear equations, or breaking widely used cryptography like RSA.**

I. Introduction

Nowadays, computation is all built around *bits*, strings of bits, and the operations that we can do on them. A bit is a finite field of order 2, often denoted as $\mathbb{F}_2$, which is nothing more than $\{0, 1\}$ with multiplication and addition modulo two. The content of a bit is its *bit value*, and all computation in a regular computer reduces to representing things as strings of zeroes and ones and then performing Boolean operations on them. A quantum computer does not work like that. In quantum computing, one has qubits instead of bits, quantum states instead of bit values, and applies unitary maps acting on complex vector spaces instead of Boolean operations acting on binary strings. For a physicist, a qubit is a two-level quantum system, and a quantum state is a vector that describes a physical system. For a mathematician or a computer scientist, a qubit is simply the two-dimensional complex Hilbert space $\mathbb{C}^2$, and an n-qubit quantum state is a unit-norm vector in a Hilbert space $H = (\mathbb{C}^2)^{\otimes n}$, where $\otimes$ denotes the tensor product of Hilbert spaces. In this work we will use the Dirac notation to denote quantum states and operations. We will use $|v\rangle$, read as ”ket v”, to denote a vector in H. We denote the dual form of $|v\rangle$ as $\langle v|$, read ”bra v”, which is a map from H to $\mathbb{C}$. Throughout this paper we use the word *classical*, in opposition to quantum, to refer to usual computers and computation done on them. Circuits will be collections of horizontal wires, one for each qubit, advancing from left to right and upon which we perform gates. Vertical wires connecting a qubit to a gate are quantum control wires. Wires with a dash crossing them denote many qubits at once. Tensor products have sometimes been omitted when no confusion arises, e.g. we write $|\varphi\rangle |\psi\rangle$ to denote $|\varphi\rangle \otimes |\psi\rangle$. With the notation laid out, the rest of the paper will deal with four quantum algorithms or primitives, starting with the one that will require most of our attention: quantum random circuits and the recent claim of quantum supremacy.

II. Quantum supremacy

A. The problem

The extended Church-Turing hypothesis (ECT) states that classical computers can simulate any physical process with only a polynomial overhead. Quantum supremacy, short for quantum computational supremacy, is the goal of demonstrating that quantum computers can efficiently perform some computations which would take an exponential amount of time (or resources) if computed by a classical computer, thus disproving the ECT hypothesis.

There are many approaches (not necessarily useful computational tasks) to demonstrate quantum supremacy, such as integer factorization [1], boson sampling [2], instantaneous quantum polynomial-time (IQP) circuits [3], random circuit sampling [5], etc. Table I shows an overview of their main strengths and weaknesses. Some approaches like integer factorization are currently very hard to run on a quantum computer, while the rest of them are fairly straightforward to simulate and can be simulated on near-term devices [4]. However, the claim of quantum supremacy for all these approaches relies on various complexity-theoretic assumptions [6], assumptions for which we have strong but not definitive evidence. Just as we still cannot prove $\mathsf{P} \neq \mathsf{NP}$, we cannot yet unconditionally claim that quantum computers are not efficiently simulatable by classical computers.

The quantum-AI group at Google demonstrated quantum supremacy by using random circuit sampling (RCS) as proposed by Boixo et al. [5]. In a nutshell, a random quantum circuit generates a probability distribution that is far from uniform, yet very unstructured

Approaches	**Implementation difficulty**	**Classical hardness assumptions**	**Verification**	**Usefulness**
Integer factorization [1]	hard	RSA secure	easy	yes
Boson sampling [2]	easy	PH infinite or approx. counting $\neq$ exact counting [4]	hard	no
IQP [3]	moderately easy		moderate	no
Random circuit sampling [5]	moderately easy	QUATH [6], #P-hard [7]	hard	no

Table I. Examples of some approaches to quantum supremacy.

and hard to sample from. The reason why random circuits are suitable for a quantum supremacy experiment are many-fold. To mention a few, they can be demonstrated using noisy, intermediate-scale quantum (NISQ) devices and do not require fault tolerance. They also satisfy average-case computational hardness, which means that sampling from the probability distribution of the output of a typical quantum circuit classically is as hard as computing them in the worst case. RCS also satisfies an anti-concentration property, making it the first candidate with average-case hardness and anti-concentration [7].

B. Google's quantum supremacy experiment

Google demonstrated quantum supremacy with a programmable 53-qubit superconducting processor known as the Sycamore processor. One can refer to [8], [9] for the details of the experiment and their quantum processor.

a) The setup: The quantum processor consists of a two-dimensional array of 54 qubits arranged in a rectangular lattice (one qubit did not work properly). Every qubit is (tunably) coupled with four nearest neighbours using a total of 86 couplers. This structure enables for error correction using surface codes. Their setup is based on superconducting qubits and non-commuting gates. The technology and design of the Sycamore processor is quite advanced, and achieves fast, high-fidelity single-qubit and two-qubit operations even when gate operations are performed on many qubits simultaneously.

b) The experiment: The first step of the experiment is to generate a random quantum circuit C of depth 20, consisting of single-qubit and two-qubit gates. This circuit C is then applied to an all zero $|0\rangle^{\otimes 53}$ initial state and the outcome is measured in the computational basis $\{|0\rangle, |1\rangle\}$. The output of the measurement, which is a 53-bit string is recorded and the process is repeated millions of times.

c) Verification: For a randomly chosen circuit C, the measurements in process (b) have generated a sample containing millions of 53-bit long strings approaching the theoretical probability distribution of that circuit's output. This distribution needs to be further analysed to check whether the output is consistent with the outcome of a well-behaving quantum circuit. Google uses the method of cross-entropy benchmarking [5]. First, they benchmark the single-qubit operations using the cross-entropy benchmarking protocol to calibrate and benchmark the processor at a component level. Using the component level fidelities, they proceed to accurately estimate the performance of the whole system. Under some assumptions made about their experimental device, this measure sufficiently certifies that quantum supremacy is achieved [7]. However, this assumption is only empirically validated for a few qubits.

C. Discussion

The Sycamore processor takes around 200 seconds for the entire process in (b), while Google's current benchmarks indicate that the classical computers would take 10,000 years for the same task. Boixo et al. [5] suggest and give numerical evidence that these random quantum circuits generate distributions that are hard to simulate classically, and, no classical simulation is known that takes less than a petabyte of storage. Most importantly, this experiment shows the high level of control and fidelity achievable on the Sycamore chip and suggests that there is no fundamental reason why a quantum computer cannot be built.

D. Applications

The applications of the Sycamore processor and the work done towards demonstrating quantum supremacy are varied. For example, the benchmarking task mentioned in (c) has direct application in generating certifiable random numbers [10]. Additionally, the structure of the Sycamore processor is designed to be compatible with error correction using surface codes, making it a good candidate to run experiments on error correction. Last but not least, the ability to manage, for the first time, fast, high-fidelity operations on 53 qubits may result in near term applications in the field of optimization, machine learning, material sciences and chemistry.

E. Criticisms

Soon after Google's paper came out, IBM [11] claimed it was possible to simulate their circuit on the largest existing supercomputer in a couple of days. In principle, this does not refute Google's claim to quantum supremacy because the amount of resources required to classically simulate random circuits still scales exponentially with the number of qubits. However, there have been concerns about the cross-entropy statistical measure which Google uses for verification. Even though it has been suggested that it is difficult to achieve high cross-entropy using classical circuits [12], this claim is not supported by any complexity-theoretic evidence. In [7], the authors comment on how cross-entropy might not be sufficient for certifying whether a device has achieved quantum supremacy without making some assumptions on the internal operations of the quantum device. To this day, no verification protocol is known to unconditionally certify the classical hardness of a sampling task.

III. Three algorithmic primitives

A. Quantum Fourier Transform

Given a vector $\mathbf{v} \in \mathbb{R}^N$, $N = 2^n$, the Fourier transform of $\mathbf{v}$ is defined as $\hat{\mathbf{v}} = F_N \mathbf{v}$, where $F_N(i,j) = \frac{1}{\sqrt{N}} \omega_N^{ij}$ and $\omega_N = e^{\frac{2\pi i}{N}}$.

The fast Fourier transform (FFT) algorithm of Cooley and Tukey [13] computes $\hat{\mathbf{v}}$ in time $O(N \log N)$. This algorithm is much faster than the naive $O(N^2)$ algorithm and is widely used in computer science. However, its scaling is still polynomial in the size of the input, which makes it incapable of dealing with very large inputs. There is a quantum algorithm, called quantum Fourier transform (QFT) that computes a quantum version of the Fourier transform in time polylogarithmic in the size of the vector.

1) The problem: The quantum Fourier transform problem can be phrased in the following way. Assume one is given a vector of amplitudes of the form $|v\rangle = \sum_{i=0}^{N-1} v_i |i\rangle$ encoding a vector $\mathbf{v} = (v_0, \ldots, v_{N-1}) \in \mathbb{R}^N$. The goal is to produce a state $|\hat{v}\rangle = \sum_{i=0}^{N-1} \hat{v}_i |i\rangle$ such that $(\hat{v}_0, \ldots, \hat{v}_{N-1}) =: \hat{\mathbf{v}} = F_N \mathbf{v}$.

2) The circuit: The circuit in fig. 1 implements the QFT in time $O(n^2)$ [14]. The gates H and R_n are defined as $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ and $R_n = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^n}} \end{pmatrix}$.

Notice how the QFT circuit and the FFT algorithm actually do different things. While the FFT produces a vector $\hat{\mathbf{v}}$ that is written *somewhere* and can easily be read, the QFT encodes $\hat{\mathbf{v}}_i$ as amplitudes. In term of readout one can roughly think of amplitudes as encoding the probabilities of obtaining result $|i\rangle$ upon measurement, and cannot be easily read off of a register. Despite this limitation, the QFT has many interesting and potentially game-changing applications.

3) Applications: A widely used application of QFT is a subroutine called *phase estimation* [14]. Let U be an n-qubit unitary, and $|u\rangle$ be an eigenvector of U with eigenvalue $e^{2\pi i \varphi}$. Using the QFT one can obtain $|\tilde{\varphi}\rangle |\psi\rangle$ with $O(t^2)$ queries to U, where $|\tilde{\varphi}\rangle$ is a t-bit approximation of φ. There is also a variant of the phase estimation subroutine called *amplitude estimation* [15].

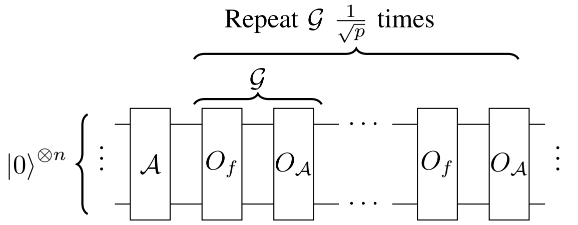
However, the most important application of the QFT is Shor's algorithm, which solves two specific instances of the hidden subgroup problem (HSP) for Abelian groups. This rather abstract and convoluted problem is at the heart of modern cryptography because all public key cryptography relies on the computational hardness of factoring (RSA) and discrete logarithms (Diffie-Hellman), which can be reduced to HSP. An efficient, polynomial-time quantum algorithm for HSP like Shor's means that our current cryptography would cease to be secure against adversaries with a quantum computer. The vulnerability of current cryptography to quantum computers has attracted much attention to the field of quantum computation, opening new research fields such as Quantum Key Distribution (QKD) and post-quantum cryptography.

B. Quantum Search

The next primitive that we discuss is called quantum search. Despite not offering quantum speedups as dramatic as those of QFT for cryptography, quantum search is of great importance because it is much more widely applicable.

1) The problem: Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Any n-bit string x such that $f(x) = 1$ is called a *solution*. Our goal is to find a solution to f. The circuit that we will present here is a subroutine because it requires one to provide an algorithm that finds a solution with a certain probability (which could be simply brute force search), and constructs a circuit from it.

2) The circuit: Assume that we have a reversible algorithm, classical or quantum, that can find a solution to f with probability p on input $|0\rangle$. Let $\mathcal{A}$ be a circuit implementation of this algorithm. Assume also that we have a reversible algorithm O_f that checks a solution by doing the operation $|x\rangle \mapsto (-1)^{f(x)} |x\rangle$. Last, let $\mathcal{R}$ act as $|x\rangle \mapsto (-1)^{1-\delta_{x,0}} |x\rangle$, i.e. $\mathcal{R}$ is a reflection around the state $|0\rangle$. Define $O_{\mathcal{A}} = \mathcal{A}\mathcal{R}\mathcal{A}^{-1}$. Since the circuit $\mathcal{A}$ finds a solution with probability p, one would need to run it $O(1/p)$ times to find a solution with high probability. The following circuit finds a solution with only $O(1/\sqrt{p})$ applications of $\mathcal{A}$ and $\mathcal{A}^{-1}$.



3) applications: This primitive has a surprising number of applications thanks to the versatility that comes from the ability to choose the function f and the algorithm $\mathcal{A}$. The first application of this primitive was finding an element in an unstructured database, which is equivalent to finding a solution to a random function. Quantum search is in essence a generalization of two widely used subroutines called *amplitude estimation* and *amplitude amplification* [15]. Such subroutines are rather ubiquitous in the literature, forming the basis of span program quantum algorithms [16], [17], [18], [19], as well as quantum algorithms for formula satisfiability [20], quantum linear system solvers [21], triangle finding [22] etc. A broadly studied application

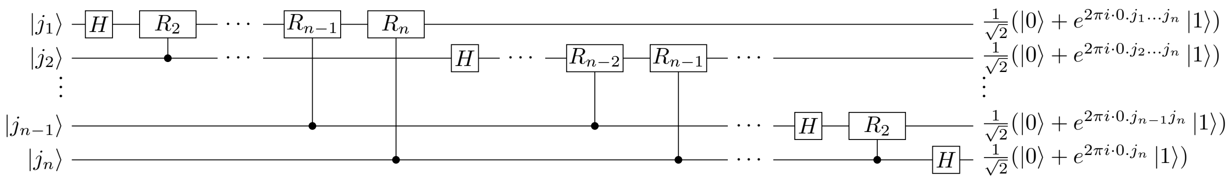

$|j_1\rangle$
$|j_2\rangle$
$|j_{n-1}\rangle$
$|j_n\rangle$
H
R_2
R_{n-1}
R_n
R_{n-2}
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \cdot 0.j_1 \ldots j_n}|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \cdot 0.j_2 \ldots j_n}|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \cdot 0.j_{n-1} j_n}|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \cdot 0.j_n}|1\rangle)$

Fig. 1. Circuit for the quantum Fourier transform.

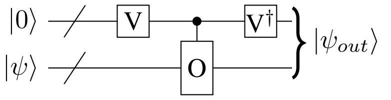
of quantum search is quantum random walks [23], [24] and random walk based algorithms [25], [26], [27], [28]. The quantum search primitive allows us to construct a quantum algorithm that walks on a graph and whose mixing time is quadratically smaller than the corresponding random walk on a regular computer. Conversely, one can run many random walks in quantum superposition and speed up some widely used types of Monte Carlo simulations [29], making quantum random walks of great practical importance.

C. Linear combination of unitaries (LCU)

This technique, first introduced by Berry et al. [30], [31], [32], allows the user to implement a Hermitian matrix M given as a linear combination of unitaries. This is a special case of a more general and involved primitive called "singular-value transformation" [33], that can implement any matrix M, be it non-Hermitian or even non-square.

1) The problem: This primitive takes as inputs two strings $\alpha = (\alpha_1, ..., \alpha_m) \subset \mathbb{R}^+$, $(U_1, \ldots, U_m) \subset \mathcal{U}(2^n)$ and an n-qubit state $|\psi\rangle \in \mathbb{C}^{2n}$, and aims to prepare a state $\propto M|\psi\rangle$, where $M = \sum \alpha_i U_i$.

2) The circuit: In addition to the n-qubit register containing the state $|\psi\rangle$, we define an additional register $\mathcal{E}$ containing $\lceil \log m \rceil$ qubits. Let us assume that one can construct the unitary $O = \sum_{i=1}^{m} |i\rangle\langle i| \otimes U_i$ acting on $\mathcal{E} \otimes \mathbb{C}^{2n}$ and a unitary on $\mathcal{E}$ acting as $V : |0\rangle \mapsto \frac{1}{\|\alpha\|_1} \sum_{i=1}^{m} \sqrt{\alpha_i}|i\rangle$. Apply the following circuit to the state $|0\rangle|\psi\rangle$.



The output state of this circuit is of the form $|\psi_{out}\rangle = \frac{1}{\|\alpha\|_1}|0\rangle M|\psi\rangle + \sqrt{1 - \frac{\|M|\psi\rangle\|^2}{\|\alpha\|_1^2}}|\varphi\rangle$, where $(|0\rangle\langle 0| \otimes I)|\varphi\rangle = 0$. That is, it produces a linear combination of the state that we want to construct, including a flag, and some other garbage state. Applying the techniques described earlier in section III-B we can amplify this amplitude and approximate the desired state to precision ϵ in $\frac{1}{\epsilon}$ iterations of the circuit presented. Observe that LCU is a meta-algorithm, a subroutine that relies on the user to supply O and V. While the map V is efficiently implementable given the string α, the complexity of O can vary a lot and will ultimately be determined by the description complexity of the string $(U_0, \ldots, U_n)$.

3) Applications: This primitive can be used to simulate the physical evolution of quantum systems, a task often called Hamiltonian simulation. In Hamiltonian simulation, the goal is to approximately implement a unitary e^{iHt} generated by the Hermitian operator H, called the Hamiltonian. Often times, a Hamiltonian can be decomposed as a linear combination of unitary local terms $\sum \alpha_i H_i$, each H_i affecting only a few particles or degrees or freedom. Combining this fact with the Taylor expansion of the exponential function, allows us to write the unitary e^{iHt} as a linear combination of unitaries, which can be truncated and implemented using the LCU technique described above.

Another application of this method is the HHL algorithm for solving linear systems of equations [21]. The goal in this case is to find a quantum state $|x\rangle$ encoding in its amplitudes the solution to a linear system of the form $A\mathbf{x} = \mathbf{b}$. In particular, if this system has a single unique solution, then A is invertible and $\mathbf{x} = A^{-1}\mathbf{b}$. The algorithm works by constructing a unitary U, called a *block encoding* of A, that contains the invertible matrix A as a minor. Using a polynomial expansion of the inverse function to approximate A^{-1} by a linear combination of unitaries allows us to obtain an approximation of $|x\rangle$. While the original algorithm by Harrow, Hassidim and Lloyd relies on amplitude and phase estimation, this algorithm by Gylién et al. [33] has better scaling in every parameter.

IV. Final remarks

In this overview we have focused on quantum supremacy and quantum algorithms, but these topics are only a portion of the field of quantum information processing. We have left out entire avenues of research such as fault tolerance, error correction, circuit optimization, secure communication or multiparty computation on which much progress has been done recently.

The quantum supremacy experiments mark the beginning of the NISQ era, in which we will start to see applications of the three primitives discussed above being implemented on real hardware. While it might take a while to see a large number factorized by a quantum computer, quantum random walks, and especially, Hamiltonian simulation, may become a reality in the following years.

Acknowledgements. SP is supported by the Robert Bosch Stiftung, also additionally supported by NWO

Gravitation grants NETWORKS and QSC, and EU grant QuantAlgo.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, pp. 1484–1509, 1994.

[2] S. Aaronson and A. Arkhipov, "The computational complexity of linear optics," 2010. [Online]. Available: https://arxiv.org/abs/1011.3245v1

[3] D. Shepherd and M. J. Bremner, "Temporally unstructured quantum computation," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 465, no. 2105, pp. 1413–1439, 2009.

[4] A. W. Harrow and A. Montanaro, "Quantum computational supremacy," *Nature*, vol. 549, p. 203–209, 2017.

[5] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, no. 6, p. 595–600, 2018.

[6] S. Aaronson and L. Chen, "Complexity-theoretic foundations of quantum supremacy experiments," 2016. [Online]. Available: https://arxiv.org/abs/1612.05903v2

[7] A. Bouland, B. Fefferman, C. Nirkhe, and U. Vazirani, "Quantum supremacy and the complexity of random circuit sampling," 2018. [Online]. Available: https://arxiv.org/abs/1803.04402v1

[8] J. M. M. et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505–510, 2019.

[9] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, and et al., "Supplementary information for 'quantum supremacy using a programmable superconducting processor'," 2019. [Online]. Available: https://arxiv.org/abs/1910.11333v1

[10] S. Aaronson, "Manuscript in preparation."

[11] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, and R. Wisnieff, "Leveraging secondary storage to simulate deep 54-qubit sycamore circuits," 2019. [Online]. Available: https://arxiv.org/abs/1910.09534v2

[12] S. Boixo, V. N. Smelyanskiy, and H. Neven, "Fourier analysis of sampling from noisy chaotic quantum circuits," 2017.

[13] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," 1965.

[14] M. A. Nielsen and I. L. Chuang, "Quantum information and quantum computation," 2001.

[15] G. Brassard, P. F. Hoyer, M. Mosca, A. T. D. U. de Montreal, B. U. of Aarhus, and C. U. of Waterloo, "Quantum amplitude amplification and estimation," 2000.

[16] T. Ito and S. Jeffery, "Approximate span programs," *Algorithmica*, vol. 81, pp. 2158–2195, 2018.

[17] A. Belovs and B. Reichardt, "Span programs and quantum algorithms for st-connectivity and claw detection," in *ESA*, 2012.

[18] M. Jarret, S. Jeffery, S. Kimmel, and A. Piedrafita, "Quantum algorithms for connectivity and related problems," in *26th Annual European Symposium on Algorithms (ESA 2018)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 112, 2018, pp. 49:1–49:13.

[19] C. Cade, A. Montanaro, and A. Belovs, "Time and space efficient quantum algorithms for detecting cycles and testing bipartiteness," *Quantum Information and Computation*, vol. 18, pp. 18–50, 2016.

[20] S. Arora and B. Barak, "Computational complexity - a modern approach," 2009.

[21] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations." *Physical review letters*, vol. 103 15, p. 150502, 2009.

[22] F. L. Gall, "Improved quantum algorithm for triangle finding via combinatorial arguments," in *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, Oct 2014, pp. 216–225.

[23] F. Magniez, A. Nayak, J. Roland, and M. Santha, "Search via quantum walk," in *STOC*, 2007.

[24] M. Santha, "Quantum walk based search algorithms," in *TAMC*, 2008.

[25] A. Belovs, "Learning-graph-based quantum algorithm for k-distinctness," *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pp. 207–216, 2012.

[26] S. Jeffery, R. Kothari, and F. Magniez, "Nested quantum walks with quantum data structures," in *SODA*, 2013.

[27] A. Ambainis, "Quantum walk algorithm for element distinctness," *45th Annual IEEE Symposium on Foundations of Computer Science*, pp. 22–31, 2003.

[28] F. L. Gall, "Improved quantum algorithm for triangle finding via combinatorial arguments," *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 216–225, 2014.

[29] A. Montanaro, "Quantum speedup of monte carlo methods," in *Proceedings. Mathematical, physical, and engineering sciences*, 2015.

[30] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, "Exponential improvement in precision for simulating sparse hamiltonians," in *STOC*, 2014.

[31] D. W. Berry, A. M. Childs, and R. Kothari, "Hamiltonian simulation with nearly optimal dependence on all parameters," *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pp. 792–809, 2015.

[32] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, "Simulating hamiltonian dynamics with a truncated taylor series." *Physical review letters*, vol. 114 9, p. 090502, 2015.

[33] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics," in *STOC*, 2018.