Grover's Algorithm

Yinzhen Li[1], Varun Vora[1], and Jacob White[1]

Purdue University
{li2850, vora18, white570}@purdue.edu

Abstract. Grover's algorithm is a relatively simple and well-known quantum unstructured search algorithm that has an extremely wide variety of applications. In this report, we briefly introduce the basics of quantum computing and key details for how Grover's algorithm works. Then, we use clarifying examples to demonstrate how quantum computers can solve some problems more efficiently than their classical counterparts, presenting insights, limitations, and further questions that highlight the importance of Grover's algorithm across a wide variety of applications and even entire subfields of research.

1 Introduction

In quantum computing, the quantum unstructured search algorithm refers to finding a unique input to some black-box oracle function f that produces a particular output with high probability. A quantum computer can achieve this by using just $O(\sqrt{N})$ evaluations of the black box function, where $N = 2^n$ is the size of the function's domain and n is the number of bits used to represent its elements. This algorithm was introduced by Lov Grover in 1996 [Gro96] and is commonly referred to as Grover's algorithm. It was also proved that any solution to the problem is required to evaluate the function at least $\Omega(\sqrt{N})$ times[Ben+97], showing that Grover's algorithm is asymptotically optimal.

2 Background

The idea behind quantum computing is to take advantage of quantum mechanical phenomena to perform some computation, applying rules of probability theory and linear algebra to modify the final state in such a way that it is likely to produce the correct result once measured. The current understanding of quantum physics allows for the state of a perfectly isolated system to be modelled by a unit vector of complex numbers.

2.1 Quantum Bits

In classical computers, bits can be in state 0 or 1. In contrast, however, quantum computer's state is represented by quantum bits– or qubits– which can simultaneously exist in a superposition of states. One common way of representing a qubit is by a unit vector of complex numbers. State 0 and State 1 qubits are called the computational basis states.

$$|0\rangle := [1, 0] \qquad |1\rangle := [0, 1]$$

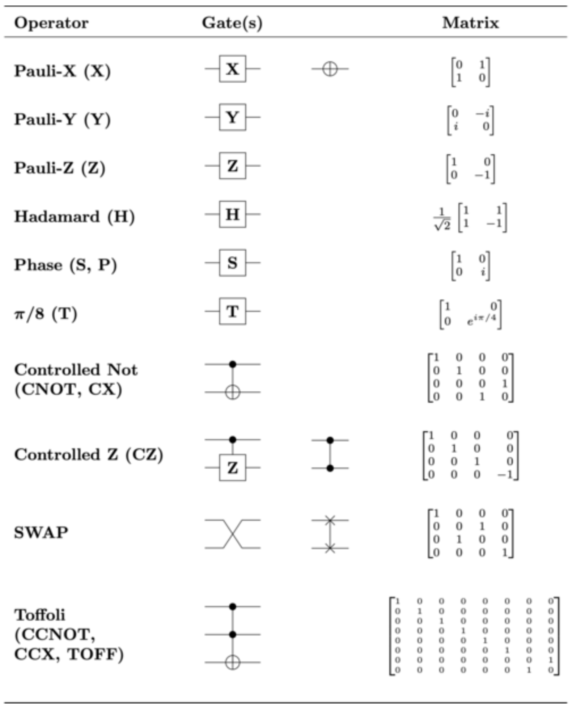
All other qubits can be represented as a linear combination of these two basis states.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where α and β are the probability amplitudes of the two basis states. When a qubit is measured, however, it collapses into one of these basis states. The probability that it collapses to the $|0\rangle$ state is $|\alpha|^2$ and the probability that it collapses to the $|1\rangle$ state is $|\beta|^2$. Since the qubit is a unit vector, it also follows that $|\alpha|^2 + |\beta|^2 = 1$ [Aar16].

2.2 Quantum Gates

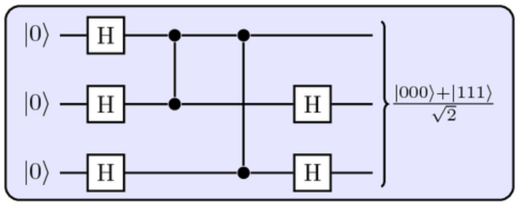
Similar to logic gates in classical computing, quantum gates can be arranged to manipulate qubits to perform quantum computations. These quantum gates are often represented by unitary matrices, which offer two conceptually different but logically equivalent representations for designing quantum algorithms. The following figure shows common quantum gates with their name, abbreviation, circuit form(s) and the corresponding matrix representation.



Operator	Gate(s)		Matrix
Pauli-X (X)	$\boxed{\mathbf{X}}$	$\oplus$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	$\boxed{\mathbf{Y}}$		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	$\boxed{\mathbf{Z}}$		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	$\boxed{\mathbf{H}}$		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)	$\boxed{\mathbf{S}}$		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)	$\boxed{\mathbf{T}}$		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)			$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	$\boxed{\mathbf{Z}}$		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP			$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)			$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

2.3 Quantum Circuit Model

The quantum logic gates can be arranged to build quantum circuits. Quantum circuits offer one model for quantum computation, where the horizontal axis is time. Quantum computations can be modelled by Quantum Turing Machines (QTMs) as well. Though there are many strategies for modelling quantum computations, they are all mathematically equivalent. The following figure shows a quantum circuit made up of Hadamard gates with three qubits in the input. Note in particular that, immediately before measuring the quantum circuit at the end, it can be reversed to provide the input again. This, along with its equivalent representation as a sequence of unitary matrices, is a necessary property required by every quantum circuit.



2.4 Notation

The n-dimensional *quantum state* is defined as a unit vector (aka "ket") in the Hilbert space $\mathbb{C}^n$. Suppose there are $N = 2^n$ distinguishable states $|1\rangle, |2\rangle, ..., |N\rangle$. The *quantum superposition* of these states is denoted by $|\psi\rangle = \alpha_1 |1\rangle + \alpha_2 |2\rangle + ... + \alpha_N |N\rangle$, with N states and $\sum_{i=1}^{N} |\alpha_i|^2 = 1$. The conjugate transpose (aka "bra" or Hermitian adjoint) of a state is denoted by $\langle\psi| = |\psi\rangle^\dagger = [\overline{\alpha_1} \dots \overline{\alpha_N}]$, where $\overline{\alpha_i}$ is the complex conjugate. Given two states $|\psi\rangle = \sum_{i=1}^{N} \alpha_i |i\rangle$ and $|\phi\rangle = \sum_{i=1}^{N} \beta_i |i\rangle$, the inner product (aka "braket") is denoted by $\langle\psi|\phi\rangle = \sum_{i=1}^{N} \overline{\alpha_i}\beta_i$. The outer product of the two states (aka "ketbra") is denoted by $|\psi\rangle\langle\phi|$, an $N \times N$ matrix with (i,j)-entry $\alpha_i\overline{\beta_j}$.[Aar16]

3 Grover's Algorithm

3.1 Search problem

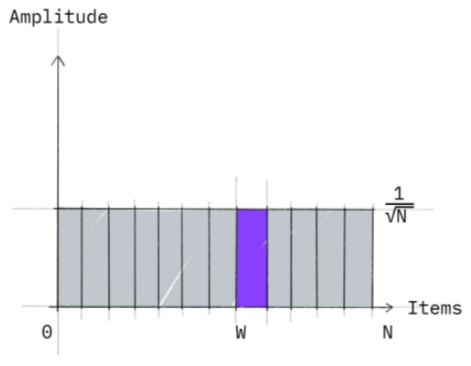
Consider the following search problem. Suppose there is a domain D of size $N = 2^n$, and a function f. For all but one element in this domain, $f(x) = 0$. For exactly one element, $f(\omega) = 1$. Now the task is to find this ω. How many times do we need to call f?

Since we assume the domain is unstructured, we are not allowed to leverage any properties of f or the problem statement that it encodes– that is, it is a "black-box" oracle. Classically, the only way to solvle this is to try every element in the search space by "brute-force". The worst case time for a classical algorithm is $N - 1$, where we check all but one elements and only the last one checked is the target. The average time is $\frac{N}{2}$, which is not much better. How can we improve on this using quantum computations?

3.2 Grover's Algorithm: an intuition

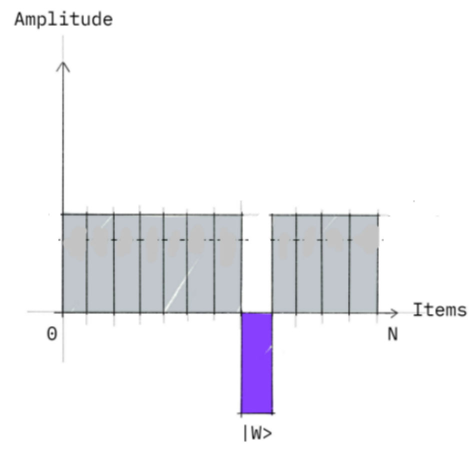
Quite simply, we call f on a superposition of states. Quantum mechanics allows us, in some sense, to call f for all inputs x simultaneously. Unfortunately, however, this is not quite as simple as calling f in a single step, as we shall soon see.

Because we don't yet know which of the elements in the domain have been marked with $f(x) = 1$, we begin by considering all candidate states to be equally likely to occur after being measured. That is, the algorithm begins by forming a uniform superposition of all N states in the domain D.



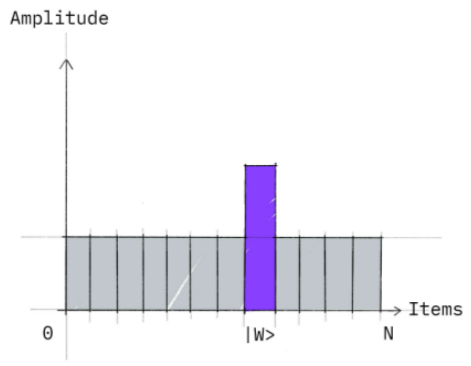
(Figures cited from Qiskit)

Then, informally speakaing, we call f on this superposition. More formally, we apply a quantum circuit, the *oracle*, which implements the function f on this superposition, resulting in a new superposition:

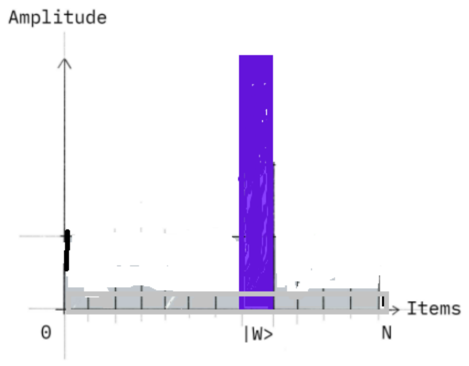


Here, the phase of the target is flipped, while the phase of all other candidates remain the same. Intuitively, this is like negating the input x iff $f(x) = 1$, doing nothing otherwise.

However, note that this is still a uniform superposition, since the amplitude of all candidates are still equal. Therefore, if we were to measure it now, all candidate states would still be equally likely to be obtained. While it is not immediately apparent, however, this step is crucial. Next, we use another quantum circuit g, the *diffuser*, to obtain yet another superposition:



Informally speaking, what this does is slightly reduce the amplitude of every non-target of f, adding them to the target of f. This result seems much more promising because, if we measure to measure the quantum state now, we would have a higher probability of obtaining the target. However, this is not sufficient, since the probability is still significantly less than 1. However, simply repeating the process by calling $f \circ g$ approximately $O(\sqrt{N})$ times will result in significant amplification of the amplitudes of f's targets.



Finally, we now measure this superposition, giving a very high probability of collapsing to the correct target state that makes $f(x) = 1$.

3.3 Grover's algorithm: formal description

For simplicity, suppose $N = 4$ and $n = 2$. Then we have 2 qubits, where the the candidate states are represented by $|00\rangle, |01\rangle, ..., |11\rangle$. Let us also assume the target $\omega = |11\rangle$.

As noted before, we start with the uniform superposition

$$|s\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

In this basic example, the quantum circuit for f can trivially be represented by the unitary matrix

$$U_f := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

Note that the position corresponds to the position of the desired target ω. In practice, while we are able to implement it by quantum circuit, we wouldn't actually know this position in the matrix where it is equal to -1. In a real application, this unitary matrix would be much too complicated to simply observe the the unitary matrix itself to find the target.

After its application, our superposition becomes

$$|s_1\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)$$

As discussed in the Section 3.2, the next step is to apply a general quantum gate g, whose matrix representation is given by

$$U_g := \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

After applying $U_g U_f$, our superposition becomes $|s_2\rangle = |11\rangle$, which is exactly the target.

As a final remark, note that the diffuser U_g and its oracle U_f are entirely separate; we don't leverage the structure of f whatsoever to find the target. Furthermore, wherever the target is, this matrix will have same desired effect, namely, giving us the clear state of the target. This is the precise meaning of "unstructured", as highlighted in Section 3.2. It will be very helpful to compare this with the matrix representation of f, which strongly depends on the position of the target.

This is for the case of $N = 4$. This case, to be clear, is a very easy example: after one iteration, we arrive exactly at the target state. For much larger quantum circuits, which work with n qubits and $N = 2^n$ states, it is more efficient to follow this process of using Grover's algorithms to sequentially lead the probability of the target being very close to 1, so that if we measure the superposition we will have very high probability of obtaining the target. In general, we need $\Theta(\sqrt{N})$ iterations to arrive at such a good state. The process is a simple generalization of the example given above so we omit the details.

3.4 Number of executions

We have outlined how Grover's algorithm works. However, we have (intentedly) ignored a very important point so far. It's difficult to fully address this issue in this 10-page paper, but we find it necessary to at least point it out here.

Ask yourself the question: why not just finish the whole process in one single step? I mean, we can define a function h as the $\sqrt{N}$ times composition of $f \circ g$. This h is a valid quantum gate, and if we apply it on the uniform superposition at the very beginning, we will immediately arrive at the final ideal superposition. Why can't we just do that and claim that we have discovered an exciting $O(1)$ algorithm?

We can only provide a partial answer here: quantum mechanics allows us to directly call f and g on a superposition, but it does not allow us to directly call this h. In other words, while it is quite reasonable to say we can implement f and g in $O(1)$ time, we have to use $O(\sqrt{N})$ to implement h. To fully understand this, one have to do some research on how we build a quantum circuit to implement f and g in reality, which we don't have time to cover here.

4 Applications

Many of the following applications rely on extensions of Grover's algorithm; see Section 5.1.

4.1 Finding a satisfying assignment for SAT

Recall that Grover's algorithm is able to search for a satisfying input to f (that is, a marked element x^* such that $f(x^*) = 1$) without relying on the particular structure of the oracle for the problem or its instances. And so, Grover's algorithm can easily be leveraged to find a satisfying assignment for Boolean formulas in SAT and many of its variants. All this requires is to relax the assumption that there exists a *unique* input to f, instead allowing for any number of elements to be marked. Then, Grover's algorithm almost trivially becomes a decider for SAT (albeit exponential-time). In particular, if we let $x \in \{0,1\}^n$ represent a particular assignment for n Boolean variables in ϕ and implement the oracle f as a quantum circuit or unitary matrix representing the statement ϕ, Grover's algorithm simply needs to find an x such that $f(x) = 1$ after at most $O(\sqrt{N})$ queries, which it does with high probability. By the amplification lemma, this can be repeated as many times as necessary to lower the error bound further, if desired, before accepting the input ϕ iff it most often chose assignments x such that $f(x) = 1$.

An interesting consequence of Grover's algorithm providing a quadratic speedup for unstructured search is that, because SAT is NP-Complete, Grover's algorithm works for any NP problem where the search domain contains the problem's instances. As a result, Grover's applies very broadly.

4.2 Cryptanalysis on one-way functions

Grover's algorithm, in essence, can be considered a function inversion algorithm– that is, finding a pre-image x such that $f(x) = 1$– and so applies broadly to *one-way functions* which are otherwise assumed difficult to invert. While the existence of one-way functions would imply, among other things, that $P \neq NP$ [Rud88], conjectured one-way functions are useful in many areas of cryptography. From this perspective, the existence of an algorithm (i.e. Grover's) that can broadly invert them in at most $O(2^{n/2})$ time instead of $O(2^n)$ is of particular concern to cryptography as a whole.

Due to Grover's algorithm being able to run in $O(\sqrt{N})$ time, it is commonly accepted in the folklore of cryptographic literature that an adversary with access to sufficiently powerful quantum computation can essentially halve the number of bits they need to search through to find a desired secret value[1]. This is especially useful for breaking cryptosystems, which often assumes classical adversaries who can only brute-force search for these values in $\Omega(N)$ time[2]. Even if the cryptosystem itself would not be broken in other means by advances in quantum computing, then, the number of bits of security should regardless be at least doubled to ensure the same level of security as assumed in a classical model. Encryption and hashes are two classic examples that demonstrate its impact.

Recovering private keys. Arguably one of the most popular examples of one-way functions is in encryption schemes. While we will not go into detail about their many and precise security definitions and instantiations, at a high level they consist of three algorithms: $\mathsf{KeyGen} : 1^\lambda \mapsto k$, $\mathsf{Enc}_k : \mathcal{M} \to \mathcal{C}$, and $\mathsf{Dec}_k : \mathcal{C} \to \mathcal{M}$ where λ is the number of bits of security (often, $|k| := \lambda$), $\mathcal{M}$ is the message space, $\mathcal{C}$ is the ciphertext space. Dec is designed to only return the original message m if it uses the same key k used to encrypt it, and does so deterministically with $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m \; \forall m \in \mathcal{M}$. However, if the format of the message m is known by or can be influenced by the adversary[3], Grover's algorithm can recover the key k that decrypts the ciphertext c to a recognizable message m in $O(2^{\lambda/2})$ time without even leveraging any of the mathematical details behind its design! This time complexity even holds when extending block ciphers implemented with various encryption modes [CNS17]. If λ is chosen to be particularly small (e.g. 256-bits)– which is especially common when using elliptic curve cryptography– Grover's algorithm can thus grant significant computational power for an adversary to subvert security of cryptosystems[2].

Finding pre-images of hashes. Another very common example of one-way functions used in cryptography are hash functions $H : \{0,1\}^* \to \{0,1\}^m$ which, given any input, produce a "random-looking" m-bit output that varies drastically with even minor changes to the input. Because there could potentially be more than m bits passed as input, however, by the pigeonhole principle some pairs of inputs (x, x') must *collide*, producing the same hash $H(x) = H(x')$. While hashing algorithms are specifically designed to be resistant against such collisions, quantum algorithms such as Grover's have the potential to more quickly find some pre-image x' that collides. An interesting caveat here is that the size of the domain here is technically infinite, as it allows for an arbitrary-length bitstring to be passed as input. If we know the maximum size of the input being hashed,

[1] Note that some sources challenge the accuracy of this notion in either direction, arguing that the nature of this result is much more nuanced that the folklore result seems to suggest; see [Flu17; CNS17].

[2] Despite also being exponential-time, even a quadratic speedup for brute-force search can be hugely impactful on the practical security of a cryptosystem.

[3] This is a necessary part of the threat model for any modern and secure encryption scheme, and is by no means an unreasonable assumption: for example, the HTML format of a webpage, or a protocol-specific query.

however (e.g. password length, disk sector size, or the length m of another hash) we can reduce our search space to, say, $n \geq m$ bits and still be guaranteed to find one such pre-image x' that hashes to the same value $H(x') = H(x)$[4]. Grover's algorithm should find such a pre-image x' in at most $O(2^{n/2})$ time by encoding f to return $f(x') = 1$ iff the pre-image results in a collision [Zha18]. As an even tigher upper bound, if we take into account the fact that multi-target quantum search applies here and also consider potential time and space complexity from implementing the oracle f to realize a practical implementation of Grover's algorithm, in practice this gives a time and space complexity of $\tilde{O}(2^{2n/5})$ and $\tilde{O}(2^{n/5})$ respectively (relative to $\mathsf{poly}(n)$) using $O(n)$ qubits [CNS17].

4.3 Machine learning

In the field of machine learning, we often have a dataset of inputs and outputs and the goal is learn a model function that accurately maps the inputs to the output. Much of the effort goes into coming up with an algorithm to learn the model. Quantum computing offers many prospects for learning algorithms, ranging from reduced computation, hyperparameter tuning and improved generalization performance.

Quantum K-nearest neighbors algorithm. The K-nearest neighbour (KNN) is one of the simplest algorithms for classifications problems with unsupervised learning. Without the outputs in the dataset, KNN attempts to classify the data into K clusters by measuring similarity between data points. However, KNN and its improved variants have require a large amount of computation and have a high complexity. The quantum KNN algorithm (QKNN) mitigates this problem using Grover's algorithm. It uses quantum superposition of states to achieve parallel computing of similarity using the quantum minimum search algorithm [Dan+18].

Quantum Genetic Algorithms. Genetic algorithms are a class of heuristic algorithms in Machine Learning inspired by the natural evolution. They are used in search and optimization problems using biologically-inspired operators such as mutation, crossover and selection. The chromosomes of an individual are usually represented as a bit-string. The aim of the heuristic is to find the desired solution by finding the fittest individual. Quantum search algorithms offer an optimization to this approach. Quantum genetic algorithms can use the qubit representation to encode an entire generation of individuals as a superposition. Finding the fittest individual from the superposition can be reduced to a Grover search [UPV06].

5 Discussion

5.1 Related work

Grover's algorithm is an optimal quantum unstructured search algorithm, requiring at least $\lceil \pi/4\sqrt{N} \rceil \in O(\sqrt{N})$ oracle lookups to have the greatest probability of finding the marked element [Ben+97; Boy+98; Zal99a]. However, it is only possible to parallelize Grover's algorithm by splitting the search space across multiple quantum computers [Zal99a], and so practical improvements to the core algorithm are highly unlikely.

There are many extensions and generalizations to Grover's algorithm with greatly expand the range of applications it has (see Section 4). For example quantum partial search, which finds blocks which contain a k-bit target, is much more efficient, requiring about $O(\sqrt{N/K})$ queries where $K = 2^k$ [GR05]. Performing a partial search, then, is significantly more efficient then searching for the entire marked element passed as input into f. Another extension involves utilizing randomness to

[4] The pre-image x' may or may not be equal to the original x that generated the hash.

construct a Quantum-RP algorithm which can find 1 of t marked elements in f for some unknown number of elements t, removing the simplifying assumption of the original algorithm that there must be a *unique* input x^* such that $f(x^*) = 1$. This requires about $\pi/4\sqrt{N/t}$ oracle queries, and is in fact more efficient than assuming a unique input [Zal99b].

Analysis of Grover's algorithm has shown that any quantum Turing machine relative to a bounded-error oracle caannot solve either NP nor $NP \cap co\text{–}NP$ in polynomial time [Ben+97]; while Grover's algorithm provides quadratic speedup over classical brute-force search, it's still exponential-time in the length of the input. This result in quantum computing is somewhat analogous to the theorem of Baker, Gill, and Solovay [BGS75], who proved that $P \overset{?}{=} NP$ in the classical world cannot be proven using relativization techniques with oracles. In a similar way, the optimality of Grover's algorithm for any black-box NP oracle implies that relativization can't be used to show that specifically $BQP = NP$. This, alongside the fact that no polynomial-time quantum algorithms are known for any NP-hard problem, lends credit to the conjecture that quantum computers do not have the power to efficiently solve NP-hard problems [Ben+97]. It is possible, however, that breakthroughs in quantum mechanics would allow for relaxations of a quantum-computing models which would result in BQP containing NP-Complete problems or even beyond [Aar+16].

Not every relaxation of constraints in the quantum computing model would allow for quantum computers to efficiently solve NP-hard problems, however. For example, even if it were possible to measure a quantum state without collapsing its superposition, Grover's algorithm could only be improved to require anywhere from $\Omega(N^{1/4})$ to $\tilde{O}(N^{1/3})$ queries. While this would make Grover's algorithm slightly more efficient, it would still require exponential time relative to an oracle, and so even a breakthrough such as this would not be sufficient to show that $NP \subseteq BQP$ [Aar+16].

5.2 Future work

While Grover's algorithm has been proven to be optimal for quantum unstructured search– at least to the extent of our current understanding of quantum mechanics– it would be interesting to see whether there exists another quantum algorithm with similar asymptotic performance, but is more efficient in practice. Such an algorithm could give opportunity for parallelization, smaller constants and/or number of qubits to implement, etc. The goal should be to remain just as generalizable as Grover's algorithm itself in performing black-box search, since other faster quantum algorithms already exist to solve more specific problems such as breaking the SHA-2 hashing algorithm [Ber09] or integer factorization [Sho94]. Finding a more parallelizable quantum black-box searching algorithm could be a PhD thesis project, depending on how bold of a practical improvement the authors would choose to strive for. Due to the complexity of quantum mechanics and especially quantum complexity as fields, and the sheer depth of existing literature on Grover's algorithm, many other open questions in this field would likely require decades of effort to conclusively answer.

Another avenue of research, though, would be to find even more useful applications of Grover's algorithm, especially for how one would implement the quantum circuit for f in practice and its resulting impact on time and space complexity. This could very easily be a PhD thesis project.

References

[Gro96] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing.* STOC '96. Philadelphia, Pennsylvania, USA: ACM, 1996, pp. 212–219. DOI: `10.1145/237814.237866`.

[Ben+97] Charles H. Bennett et al. "Strengths and Weaknesses of Quantum Computing". In: *SIAM J. Comput.* 26 (1997), pp. 1510–1523.

[Aar16] Scott Aaronson. *The Complexity of Quantum States and Transformations: From Quantum Money to Black Holes.* 2016. arXiv: `1607.05256 [quant-ph]`.

[Rud88] Steven Rudich. "Limits on the Provable Consequences of One-way Functions." PhD thesis. EECS Department, University of California, Berkeley, 1988.

[Flu17] Scott Fluhrer. *Reassessing Grover's Algorithm.* Cryptology ePrint Archive, Report 2017/811. `https://ia.cr/2017/811`. 2017.

[CNS17] André Chailloux, María Naya-Plasencia, and André Schrottenloher. "An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography". In: *Advances in Cryptology – ASIACRYPT 2017.* Ed. by Tsuyoshi Takagi and Thomas Peyrin. Cham: Springer International Publishing, 2017, pp. 211–240.

[Zha18] Mark Zhandry. *Lecture notes for COS 597A: Quantum Cryptography: Lecture 6.* Ed. by Kevin Liu. 2018.

[Dan+18] Yijie Dang et al. *Image Classification Based on Quantum KNN Algorithm.* 2018. arXiv: `1805.06260 [cs.CV]`.

[UPV06] Mihai Udrescu, Lucian Prodan, and Mircea Vlăduţiu. "Implementing quantum genetic algorithms: a solution based on grover's algorithm". In: *Proceedings of the 3rd Conference on Computing Frontiers.* 2006, pp. 71–82.

[Boy+98] Michel Boyer et al. "Tight Bounds on Quantum Searching". In: *Fortschritte der Physik* 46.4-5 (June 1998), pp. 493–505. DOI: `10.1002/(sici)1521-3978(199806)46:4/5<493::aid-prop493>3.0.co;2-p`.

[Zal99a] Christof Zalka. "Grover's quantum searching algorithm is optimal". In: *Phys. Rev. A* 60 (4 Oct. 1999), pp. 2746–2751. DOI: `10.1103/PhysRevA.60.2746`.

[GR05] Lov K. Grover and Jaikumar Radhakrishnan. "Is Partial Quantum Search of a Database Any Easier?" In: *Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures.* SPAA '05. Las Vegas, Nevada, USA: ACM, 2005, pp. 186–194. DOI: `10.1145/1073970.1073997`.

[Zal99b] Christof Zalka. "A Grover-based quantum search of optimal order for an unknown number of marked elements". In: *arXiv e-prints*, quant-ph/9902049 (Feb. 1999), quant–ph/9902049. arXiv: `quant-ph/9902049 [quant-ph]`.

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. "Relativizations of the P =? NP Question". In: *SIAM J. Comput.* 4 (1975), pp. 431–442.

[Aar+16] Scott Aaronson et al. "The Space "Just Above" BQP". In: ITCS '16. Cambridge, Massachusetts, USA: Association for Computing Machinery, 2016, pp. 271–280. DOI: `10.1145/2840728.2840739`.

[Ber09] Daniel Bernstein. "Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete?" In: *Conference Proceedings for Special-purpose Hardware for Attacking Cryptographic Systems.* SHARCS '09. Lausanne, Switzerland: ECRYPT II, Jan. 2009, pp. 105–117.

[Sho94] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science.* Nov. 1994, pp. 124–134. DOI: `10.1109/SFCS.1994.365700`.