Efficient simulation of Hamiltonians

by

Robin Kothari

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The problem considered in this thesis is the following: We are given a Hamiltonian H and time t, and our goal is to approximately implement the unitary operator $e^{-\mathrm{i}Ht}$ with an efficient quantum algorithm. We present an efficient algorithm for simulating sparse Hamiltonians. In terms of the maximum degree d and dimension N of the space on which the Hamiltonian acts, this algorithm uses $(d^2(d+\log^* N)\|Ht\|)^{1+o(1)}$ queries. This improves the complexity of the sparse Hamiltonian simulation algorithm of Berry, Ahokas, Cleve, and Sanders, which scales like $(d^4(\log^* N)\|Ht\|)^{1+o(1)}$. In terms of the parameter t, these algorithms are essentially optimal due to a no–fast-forwarding theorem.

In the second part of this thesis, we consider non-sparse Hamiltonians and show significant limitations on their simulation. We generalize the no–fast-forwarding theorem to dense Hamiltonians, and rule out generic simulations taking time $o(\|Ht\|)$, even though $\|H\|$ is not a unique measure of the size of a dense Hamiltonian H. We also present a stronger limitation ruling out the possibility of generic simulations taking time $\mathrm{poly}(\|Ht\|, \log N)$, showing that known simulations based on discrete-time quantum walks cannot be dramatically improved in general. We also show some positive results about simulating structured Hamiltonians efficiently.

Acknowledgements

First, I would like to thank my supervisor, Andrew Childs, for being a constant source of encouragement, ideas, advice and information. I am grateful that he worked with me closely when I needed his help, and yet let me work independently and choose my own problems when I wanted to. Most importantly, I would like to thank him for making me feel like a collaborator rather than a research assistant. I would also like to thank Andrew Childs again for working with me on all the problems considered in this thesis. All the results presented in this thesis are the outcome of joint work with him. Needless to say, this thesis would not be possible without him.

Second, I would like to thank Richard Cleve and John Watrous, the other two faculty members I had the chance to interact with on a regular basis. At their weekly group meetings I learned a great deal about sub-fields of quantum computation and information outside my area of research. These meetings also introduced me to some interesting people, who I had the chance to work with and learn from. Among these people, I especially wish to thank Tsuyoshi Ito, who always promptly answered my questions about complexity theory and quantum computation.

Next, I would like to thank my fellow graduate students at IQC, with whom I had many quantum and non-quantum discussions. I would also like to thank my new friends in Waterloo for making my stay here enjoyable and my old friends for many years of friendship and support. I am grateful to my office-mate, Christina Boucher, for providing help and advice, especially when I was new here.

Finally, I would like to thank my family, particularly my mother for making my stay in Waterloo extremely comfortable.

Contents

Chapter 1

Introduction

1.1 Why simulate Hamiltonians?

To motivate the problem considered in this thesis, let us start with a very fundamental question. What is the aim of physics? In other words, what are physicists trying to do? There are several good answers to this question, one of which is that physicists are trying to find a theory that explains how the universe works. This answer immediately leads to the more interesting question: what does it mean to *explain how the universe works*?

When is one theory said to explain how the universe works better than another theory? A pragmatic answer that I like is that the better theory is the one that predicts future events more accurately. The theory whose prediction more closely matches reality is the better theory. Conversely, if a theory is able to explain all logically consistent future outcomes equally well, including those that never happen, then it is a completely useless theory. This fact is sometimes counter-intuitive, as people often believe that a stronger theory should explain more things, such as apples falling upwards and like charges attracting each other. This is why a theory which allows all possible futures has zero informational content. To quote Eliezer Yudkowsky [36],

> *The strength of a theory is not what it allows, but what it prohibits; if you can invent an equally persuasive explanation for any outcome, you have zero knowledge.*

So predicting future events accurately is very important, and probably one of the most important problems ever. If we believed that our universe is completely deterministic, as was widely believed before the advent of quantum mechanics, our ideal theory would predict the future perfectly, given enough time to perform calculations (i.e., unbounded computational power). This is exactly what Laplace [29] wrote in 1825:

> *An intelligence that, at a given instant, could comprehend all the forces by which nature is animated and the respective situation of the beings that make it up, if moreover it were vast enough to submit these data to analysis, would encompass in the same formula the movements of the greatest bodies of the universe and those of the lightest atoms. For such an intelligence nothing would be uncertain, and the future, like the past, would be open to its eyes.*

In simpler words, he says that if we know the (classical) Hamiltonian of the system (*all the forces by which nature is animated*) and the initial conditions (*the respective situation of the beings that make it up*), and have unbounded computational power (*were vast enough to submit these data to analysis*), then the future can be perfectly predicted (*nothing would be uncertain, and the future, like the past, would be open to its eyes*).

Although Laplace said this about Newtonian mechanics, it still makes sense in the quantum world. If we know the Hamiltonian of the system and the initial quantum state, we can predict the final quantum state of the system. The equation that tells us how to do this is the Schrödinger equation. In this thesis, quantum systems will always be finite dimensional and the Hamiltonians considered will be time independent. In this case, the Schrödinger equation simplifies to

$$|\Psi_{\text{final}}\rangle = e^{-\mathrm{i}Ht} |\Psi_{\text{initial}}\rangle \,, \tag{1.1}$$

where H is the Hamiltonian (an $N \times N$ Hermitian matrix), $|\Psi_{\text{initial}}\rangle$ and $|\Psi_{\text{final}}\rangle$ are N-dimensional vectors representing the initial and final quantum states respectively, and t is the time difference between the initial and final states.

This is the Hamiltonian simulation problem: Given a Hamiltonian, a time and an initial state, produce the output state. It is *merely* the problem of exponentiating a matrix and multiplying it with a vector, which is not a difficult task at all. So why is this a hard problem? The hardness lies in the fact that although this is easy to do in principle, it is not clear how to do this in a reasonable amount of time. In 1929, Dirac [14] wrote

> *The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble.*

This quotation highlights an important point—just knowing equations is not sufficient to predict future events. We also need to be able to solve these equations in a reasonable amount of time. There would be no point in knowing how to predict the motion of a few particles if the amount of time it required to predict this exceeded the age of the universe.

So what is a reasonable amount of time to predict something? Is this a subjective concept or can one quantify the notion of an efficient method to solve a problem? The punch line is now obvious to everyone who has taken an algorithms course. Of course we know how to quantify efficient algorithms today, although it wasn't obvious at all before the invention of Turing machines and the papers of Cobham [12] and Edmonds [15]. This is the problem studied in this thesis: How efficiently can we solve the Hamiltonian simulation problem?

1.2 Why quantum computation?

The Hamiltonian simulation problem, or specifically the problem of simulating actual quantum systems is an important practical problem today. A significant fraction of the world's computing power is devoted to simulating quantum systems that arise in chemistry, materials science, condensed matter physics, nuclear physics, etc. Unfortunately, all the known classical algorithms for simulating quantum systems take exponential time.

The meaning of the previous sentence might not be clear, since I have not explained what it means to simulate a quantum system. Surely if one is required to output an actual quantum state this is obviously infeasible for a classical computer, which does not have the ability to process quantum information. If the output is to be a description of the quantum state, even approximately, this would require exponential space and time to write down.

However, the problem can be easily modified to make the inputs and outputs classical without losing the essence of the problem. For example, consider the following problem whose inputs and outputs are classical. Given a Hamiltonian H, and a time t, approximately sample from the probability distribution over measurement outcomes when the state $e^{-\mathrm{i}Ht}|0\rangle$ is measured in the computational basis.

Now the above-mentioned claim makes sense: all known classical algorithms for this problem run in exponential time. Moreover, if this problem had an efficient classical algorithm, then quantum computers, which are quantum systems too, would have efficient classical simulations.[1] In particular, this would mean that integer factorization has an efficient classical algorithm [35]. Indeed, it was the apparent exponential time complexity of simulating quantum systems on a classical computer that led Feynman to propose the idea of quantum computation [20].

In addition to predicting the behavior of physical systems, Hamiltonian simulation has algorithmic applications. For example, the implementation of a continuous-time quantum

[1] This statement can be made more formal by saying that the Hamiltonian simulation problem, even when the Hamiltonian is a sum of k-local terms, is BQP-hard. This follows from a construction of Feynman [21] that encodes a polynomial-size quantum circuit into such a Hamiltonian.

walk algorithm is a Hamiltonian simulation problem. Examples of algorithms that can be implemented using Hamiltonian simulation methods include unstructured search [19], adiabatic optimization [18], a quantum walk with exponential speedup over classical computation [8], and the recent NAND tree evaluation algorithm of Farhi, Goldstone and Gutmann [16].

As an example of how Hamiltonian simulation is used for algorithm design, consider the NAND tree evaluation algorithm. The algorithm encodes the input into a sparse Hamiltonian in such a way that the output can be obtained by simulating the Hamiltonian for some amount of time. Since the constructed Hamiltonian is sparse, it can be simulated efficiently, and the algorithm is efficient.

I believe that there are other algorithmic applications of Hamiltonian simulation waiting to be discovered, and Hamiltonian simulation is a useful tool to have at one's disposal when designing quantum algorithms. This is the main motivation for improving the efficiency of Hamiltonian simulation algorithms.

1.3 Notation and terminology

Most of the notation used in this thesis is standard in quantum computation, and is given in any standard textbook such as the books by Nielsen and Chuang [31] or Kaye, Laflamme and Mosca [26]. We will see several matrix norms in this thesis, all of which are explained in detail in Section 2.1. We will also need some terminology from graph theory, which is explained below. It is assumed that the reader is familiar with quantum computation, linear algebra and basic graph theory.

In this thesis, graphs will always be simple: no self-loops and no multiple edges. Thus, a directed graph G will be an ordered pair (V, E), such that $E \subseteq V \times V$ and for all vertices v, (v, v) is not an edge. An undirected graph has the additional constraint that for all edges (u, v), (v, u) is also an edge. Equivalently, E can be thought of as a set of unordered pairs, instead of a set of ordered pairs.

A graph is said to be d-sparse if its maximum degree is d. A *star graph* is a tree in which one vertex (called the center) is adjacent to all the other vertices. In other words, it is a complete bipartite graph $K_{1,r}$. We call a forest in which each tree is a star graph a *galaxy*. A graph is said to have *arboricity* k if its adjacency matrix can be written as the sum the adjacency matrices of k forests, but not $k-1$ forests.

A directed graph is a *directed forest* (*directed tree*) if its undirected graph is a forest (tree). A directed tree is an *arborescence* if it has a unique root v such that all edges point away from v. Alternately, there is exactly one directed path from v to any other vertex u.

In an arborescence, the edges are always directed from the parent to the child. A directed forest in which each tree is an arborescence is called a forest of arborescences.

A Hamiltonian, which is an $N \times N$ Hermitian matrix, can also be thought of as the weighted adjacency matrix of a graph on N vertices, where the weight of the edge (i, j) is H_{ij}. In this graph, the weights are complex numbers and the weight of an edge from u to v is the complex conjugate of the weight of the edge from v to u. Now if we make all the nonzero weights equal to 1, we get the adjacency matrix of an undirected graph. We call this the *graph of the Hamiltonian*. Notice that the graph of a Hamiltonian doesn't depend on the actual entries; it just depends on whether they are zero or not. Two Hamiltonians whose sets of nonzero entries are the same have the same graph. We often associate properties of the graph of a Hamiltonian with the Hamiltonian itself. For instance, we might say "H is a forest," meaning that the graph of H is a forest. Note that the graph of a Hamiltonian with at most d nonzero entries in each row is a d-sparse graph. We call such Hamiltonians d-sparse.

Finally, in this thesis $\log N$ will denote the base-2 logarithm of N. The function $\log^* N$ is defined by $\log^* N = 0$ if $N \leq 1$ and $\log^* N = 1 + \log^* \log N$ if $N > 1$.

1.4 The Hamiltonian simulation problem

In the Hamiltonian simulation problem, for a given Hamiltonian H and time t, our goal is to approximately implement the unitary operator $e^{-\mathrm{i}Ht}$ with an efficient quantum algorithm (using ancilla, if needed). An *approximate implementation* just means that the final state produced should not be too far away from the intended final state, in terms of the trace distance.[2] This seems like the right norm to use because the distinguishability of two states is exactly characterized by their trace distance, as originally observed by Helstrom [24].

Most importantly, we want this implementation to be efficient. Since a system with $\log N$ qubits is described by a Hamiltonian of size $N \times N$, we would like the running time of the algorithm to be polynomial in the size of the system, which is $\log N$. To impose a reasonable restriction on the dependence on ϵ, we could ask that the algorithm should be able to produce a constant approximation, such as $\epsilon = 0.01$, in polynomial time. However, this isn't good enough, since the Hamiltonian simulation algorithm might be used as a subroutine polynomially many times, and we may wish to have constant error at the end. Thus we should be able to guarantee inverse polynomial error in polynomial time. Putting these two constraints together, we need an algorithm with running time $\mathrm{poly}(\log N, 1/\epsilon)$.

Now we need to impose a reasonable restriction on the dependence on H and t. First let us consider the dependence on t. It seems reasonable to expect our algorithm's running

[2]The trace distance between ρ_0 and ρ_1 is $\|\rho_0 - \rho_1\|_{\mathrm{tr}}$, where $\|A\|_{\mathrm{tr}}$ is defined as $\mathrm{Tr}\left(\sqrt{A^\dagger A}\right)$.

time to be polynomial in the running time of the quantum system we are simulating. In particular, we would like it to be as close to linear as possible, since a sub-linear dependence on t is not possible in general (see Theorem 1 in Section 1.5). The dependence on H, however, is not as straightforward, since it depends on how H is represented. Clearly H cannot be written down as an $N \times N$ matrix, because an algorithm running in poly$(\log N)$ time cannot read the entire input.

Since the unitary $e^{-\mathrm{i}Ht}$ is a function of the product of H and t only, and not H or t individually, this gives some restriction on how the algorithm should scale with H. For example, we can rescale t by rescaling H, by mapping the pair (H, t) to $(Ht, 1)$, without affecting the unitary $e^{-\mathrm{i}Ht}$. Thus our algorithm's running time should be bounded by a polynomial in some measure of the size of H. As explained in Section 2.1, the obvious choice here is a matrix norm. When H is sparse, most of its matrix norms have comparable values and so the complexity of simulating H is not very sensitive to how its size is quantified. It is conventional to require that the scaling be polynomial in $\|H\|$, the spectral norm of H. (As discussed in Chapter 4, this might not be appropriate for non-sparse Hamiltonians.)

Finally, we can put these constraints together. We say that a Hamiltonian H can be simulated efficiently if there exists a quantum circuit which uses at most poly$(\log N, \|Ht\|, 1/\epsilon)$ one- and two-qubit gates that approximates the unitary operator $e^{-\mathrm{i}Ht}$, such that the maximum error in the final state, as quantified by the trace distance, is at most ϵ. Furthermore, just knowing that these circuits exist is not sufficient; we would like that these circuits can be generated in polynomial time. So a uniformity condition is needed, which would state that given access to the inputs H, t and ϵ, there is an efficient procedure to produce the quantum circuit. (I have deliberately not explained how the input H is given. This will depend on context and will be described when needed.)

1.5 Previous work

Various classes of Hamiltonians are known to have efficient simulations. As a trivial example, consider a k-local Hamiltonian, where k is some constant. A Hamiltonian is called k-local if it acts non-trivially on at most k qubits. Such a Hamiltonian can be simulated in polynomial time because we can read the input to find out which k qubits are affected by this Hamiltonian and then transform those qubits appropriately. Since k is a constant, any unitary operation on k qubits can be implemented with a constant number of one- and two-qubit gates.

As another example, consider a diagonal Hamiltonian, which is a Hamiltonian whose off-diagonal entries are zero. Assuming we have an efficient procedure to compute the diagonal entries of H, such a Hamiltonian is easy to implement. If H is diagonal, so is $e^{-\mathrm{i}Ht}$, and if the j^{th} diagonal entry of H is h_j, then the j^{th} diagonal entry of $e^{-\mathrm{i}Ht}$ is

$e^{-\mathrm{i}h_j t}$. So the transformation to be performed is $|j\rangle \to e^{-\mathrm{i}h_j t}|j\rangle$. This operation is easy to implement: first compute h_j in an ancillary register, then perform a set of controlled phase gates on the first register controlled by the ancilla, and then uncompute the ancilla. More succinctly, $|j,0\rangle \to |j,h_i\rangle \to e^{-\mathrm{i}h_j t}|j,h_j\rangle \to e^{-\mathrm{i}h_j t}|j,0\rangle$. (For more details, and an efficient circuit implementation, see Rule 1.6 in Ref. [10].)

As a non-trivial example, consider the class of Hamiltonians that can be represented as the sum of polynomially many local Hamiltonians. Although local Hamiltonians can be simulated trivially, simulating a sum of local Hamiltonians is not necessarily easy. Note that $e^{-\mathrm{i}(A+B)t}$ is, in general, not equal to $e^{-\mathrm{i}At}e^{-\mathrm{i}Bt}$, and thus a sum of local Hamiltonians cannot, in general, be simulated by simulating them one after the other. As Lloyd observed, such Hamiltonians can be simulated efficiently using the Lie–Trotter formula [30], given an explicit list of the local Hamiltonians. (Notice that the size of the input is polynomial in $\log N$, since the input consists of polynomially many constant-sized Hamiltonians.)

This was later generalized by Aharonov and Ta-Shma [1] to the case of sparse (and efficiently row-computable) Hamiltonians. A Hamiltonian of size $N \times N$ is sparse if it has at most $\mathrm{poly}(\log N)$ nonzero entries in any row. It is efficiently row-computable if there is an efficient procedure to determine the location and matrix elements of the nonzero entries in each row. This is indeed a generalization of the Hamiltonians considered by Lloyd, since they are also sparse and efficiently row-computable.

These conditions lead to a very convenient black-box formulation of the problem, which allows us to abstract away the details of the efficient procedure for computing locations and matrix elements. We assume there is a black box that can be queried with a row index j and another index i to obtain the value and location of the i^{th} nonzero entry in the j^{th} row. This black box can be implemented efficiently when H is efficiently row-computable. Now we can describe the complexity of simulating sparse Hamiltonians in terms of the number of queries made to this black box.

A series of results decreased the number of black-box queries, in terms of N, from the original $O(\log^9 N)$ [1], to $O(\log^2 N)$ [10], to $O(\log^* N)$ [5]. In particular, Berry, Ahokas, Cleve, and Sanders [5] presented an almost linear-time algorithm for simulating sparse Hamiltonians with query complexity

$$(\log^* N) d^4 \|Ht\| \left(\frac{\|d^2 Ht\|}{\epsilon} \right)^{o(1)}, \tag{1.2}$$

where d is the maximum number of nonzero entries in any row and ϵ is the maximum permitted error in the final state.

The dependence of (1.2) on the simulation time is nearly optimal, since it is not possible to simulate a general sparse Hamiltonian for time t using $o(t)$ queries. Intuitively, this means there is no generic way to fast-forward through the time evolution of quantum systems. More formally, we have the following theorem [5, Theorem 3].

Theorem 1 (No–fast-forwarding theorem)**.** *For any positive integer N there exists a row-computable sparse Hamiltonian H with $\|H\| = 1$ such that simulating the evolution of H for time $t = \pi N/2$ within precision $1/4$ requires at least $N/4$ queries to H.*

Although the dependence on t cannot be substantially improved, we would like to improve the query complexity of simulating sparse Hamiltonians in some other way. In Chapter 3, we show how to simulate sparse Hamiltonians with query complexity

$$(d + \log^* N)d^2\|Ht\|\left(\frac{\|dHt\|}{\epsilon}\right)^{o(1)}, \tag{1.3}$$

which is at least as good as (1.2), and is strictly better when $d = \omega(1)$.

The simulation of Ref. [5] has also been improved using a completely different approach [7, 4], giving an algorithm with query complexity

$$O\left(\frac{\|Ht\|}{\sqrt{\epsilon}} + d\max(H)t\right), \tag{1.4}$$

where $\max(H) = \max_{ij}|H_{ij}|$. This improves (1.2) in some ways, but its dependence on the error threshold ϵ is worse, and thus the two algorithms are incomparable.

Moving on to more general Hamiltonians, recently methods have been presented for simulating a Hamiltonian H that is not necessarily sparse. Of course, we do not expect to efficiently simulate a general Hamiltonian, simply because there are too many Hamiltonians to consider (just as we cannot hope to efficiently implement a general unitary operation [28]). Moreover, a general Hamiltonian does not even have a polynomial-size description. However, we can conceivably efficiently simulate non-sparse Hamiltonians with a suitable concise description.

In particular, by applying phase estimation to a discrete-time quantum walk derived from H, one can simulate H for time t in a number of walk steps that grows only linearly with t [7, Theorem 5].

Theorem 2. *For any Hermitian matrix H, there is a discrete-time quantum walk on the graph of nonzero entries of H such that $e^{-\mathrm{i}Ht}$ can be simulated with error at most ϵ using $O(\|\mathrm{abs}(Ht)\|/\epsilon)$ steps of the walk, where $\mathrm{abs}(H)$ is the matrix with entries $\mathrm{abs}(H)_{jk} = |H_{jk}|$.*

Of course, to apply this result, we must implement the discrete-time quantum walk derived from H. This can be done efficiently for various concisely specified non-sparse Hamiltonians [7]. Note that the same theorem holds with $\|\mathrm{abs}(H)\|$ replaced by $\|H\|_1$ (a matrix norm defined in Section 2.1); this quantity is generally larger than $\|\mathrm{abs}(H)\|$, but the resulting walk may be easier to implement.

Notice that the overhead of this simulation is proportional not to the spectral norm $\|H\|$, but to a measure of the size of H that can be much larger when some entries of H are negative (or more generally, complex). This naturally raises the question of whether an improved simulation is possible. In Chapter 4, we examine this possibility. Unfortunately, we show that there is no general Hamiltonian simulation algorithm that uses only $\mathrm{poly}(\|Ht\|, \log N)$ steps (Theorem 8), even if the algorithm has access to additional structural information about the Hamiltonian, including information that allows a discrete-time quantum walk on the graph of nonzero entries of H to be performed efficiently. This suggests that Theorem 2 cannot be substantially improved.

1.6 Summary of results

The results stated in this thesis also appear in published articles and preprints, as indicated below.

In Chapter 3, we present an algorithm for simulating sparse Hamiltonians that improves on the best known algorithm for high precision simulation of sparse Hamiltonians. We improve the complexity from $(\log^* N) d^4 \|Ht\| \left(\frac{\|d^2 Ht\|}{\epsilon}\right)^{o(1)}$ to $(d + \log^* N) d^2 \|Ht\| \left(\frac{\|dHt\|}{\epsilon}\right)^{o(1)}$, providing the best known method for high-precision simulation of sparse Hamiltonians. The results of Chapter 3 appear in the following preprint:

> Andrew M. Childs and Robin Kothari. Simulating sparse Hamiltonians with star decompositions. *Arxiv preprint arXiv:1003.3683*, 2010.

In Chapter 4, we consider the problem of simulating dense Hamiltonians. In Section 4.1 we describe how the no–fast-forwarding theorem (Theorem 1) can be modified to give a lower bound that depends on the spectral norm rather than various smaller measures of the size of a Hamiltonian. We then strengthen this result in Section 4.2 and present an example of a family of Hamiltonians with $\|\mathrm{abs}(H)\| \gg \|H\|$ that cannot be simulated in time $\mathrm{poly}(\|Ht\|, \log N)$.

In Chapter 5, we investigate how certain structured Hamiltonians can be simulated in time $O(\|Ht\|)$, even though a general Hamiltonian cannot. In particular, we give a positive result on the simulation of Hamiltonians whose graphs have small arboricity. The results of Chapters 4 and 5 appear in the following published article:

> Andrew M. Childs and Robin Kothari. Limitations on the simulation of non-sparse Hamiltonians. *Quantum Information and Computation* **10** (2010) 669–684.

Chapter 2

Measuring simulation complexity

2.1 Measures of simulation complexity

As one can see from Section 1.5, upper and lower bounds on the complexity of simulating a Hamiltonian H depend on some measure of the size of H. As discussed in Section 1.4, since $e^{-\mathrm{i}Ht}$ depends only on the product Ht, the complexity of simulating H for time t is some function of Ht. For example, the no–fast-forwarding theorem clearly cannot be circumvented by simply multiplying H by a constant. Similarly, simulation results such as those for sparse Hamiltonians, using $\|Ht\|^{1+o(1)}$ operations, and Theorem 2, using $O(\|\mathrm{abs}(Ht)\|)$ operations, depend on various measures of the size of Ht.

In this section, we take a step back and consider properties of various measures of the size of H that may play a role in the complexity of simulating it. Let $\nu(Ht)$ be a function that measures the complexity of simulating H for time t. We can infer various properties of $\nu(\cdot)$ as follows. Since it is trivial to simulate the identity operation, $\nu(0) = 0$. On the other hand, if $H \neq 0$, then it requires some work to simulate, so $\nu(H) > 0$. It is also plausible to suppose that $\nu(tH) = |t|\nu(H)$. We clearly have $\nu(Ht) \leq |t|\nu(H)$ for $t \in \mathbb{Z}$, since Ht can be simulated using $|t|$ exact simulations of H. On the other hand, the no–fast-forwarding theorem suggests that this is the best possible way to simulate Ht in general. Finally, since the Lie product formula can be used to simulate $H + K$ using simulations of H and K, we expect that $\nu(H + K) \lessapprox \nu(H) + \nu(K)$ (up to the fact that a bounded-error simulation requires a slightly superlinear number of operations).

These properties are reminiscent of the axioms for matrix norms, suggesting that it may be reasonable to quantify the complexity of simulating H in terms of some matrix norm $\nu(H)$. Indeed, results on the simulation of sparse Hamiltonians are typically stated in terms of the spectral norm $\|H\|$, and Theorem 2 also involves matrix norms. We now introduce various matrix norms relevant to Hamiltonian simulation.

2.2 Matrix norms

Definition 1 (Spectral norm)**.** The spectral norm of a matrix H is defined as

$$\|H\| := \max_{v \neq 0} \frac{\|Hv\|}{\|v\|} = \max_{\|v\|=1} \|Hv\|, \tag{2.1}$$

where $\|v\|$ is the standard Euclidean vector norm defined as $\|v\| := \sqrt{\sum_i |v_i|^2}$.

The spectral norm, also known as the operator norm or induced Euclidean norm, is equal to the largest singular value of the matrix. For Hermitian matrices it is also equal to the magnitude of the largest eigenvalue. This norm arises in the complexity of sparse Hamiltonian simulation algorithms, and in Theorem 2 as the spectral norm of $\mathrm{abs}(H)$, the matrix with entries $\mathrm{abs}(H)_{jk} = |H_{jk}|$.

Definition 2 (Induced 1-norm)**.** The induced 1-norm of a matrix H is defined as

$$\|H\|_1 := \max_{v \neq 0} \frac{\|Hv\|_1}{\|v\|_1} = \max_j \sum_i |H_{ij}|, \tag{2.2}$$

where $\|v\|_1$ is the vector 1-norm defined as $\|v\|_1 := \sum_i |v_i|$.

The induced 1-norm is equal to the maximum absolute column sum of the matrix. As mentioned in Section 1.5, Theorem 2 holds with $\|\mathrm{abs}(H)\|$ replaced by $\|H\|_1$. This does not, however, lead to a superior simulation method since $\|H\|_1 \geq \|\mathrm{abs}(H)\|$, as shown in Lemma 1 below.

Definition 3 (Maximum column norm)**.** The maximum column norm of a matrix H is defined as

$$\mathrm{mcn}(H) := \max_j \sqrt{\sum_i |H_{ij}|^2} = \max_{v \neq 0} \frac{\|Hv\|}{\|v\|_1} = \max_j \|He_j\|, \tag{2.3}$$

where e_j is the j^{th} column of the identity matrix.

The maximum column norm is the maximum Euclidean norm of the columns of H. This norm appears in the complexity of an algorithm for simulating Hamiltonians whose graphs are trees [7, Theorem 4] and in the related Proposition 2 in Section 5.1.

Definition 4 (Max norm)**.** The max norm of a matrix H is defined as

$$\max(H) := \max_{i,j} |H_{ij}|. \tag{2.4}$$

The max norm is just the largest entry of H in absolute value. It is a matrix norm, and is typically much smaller than the other norms mentioned.

2.3 Relationships between matrix norms

The following lemma relates the various norms introduced above.

Lemma 1. *For any Hermitian matrix $H \in \mathbb{C}^{N\times N}$, we have the following inequalities:*

$$\max(H) \le \operatorname{mcn}(H) \le \|H\| \le \|\operatorname{abs}(H)\| \le \|H\|_1 \le \sqrt{N}\operatorname{mcn}(H) \le N\max(H). \tag{2.5}$$

Furthermore, each of these inequalities is the best possible.

Proof. The first inequality follows from the fact that the maximum element in any column cannot be greater than the Euclidean norm of that column. We have

$$\max(H) = \max_j \Big(\max_i |H_{ij}| \Big) \le \max_j \sqrt{\sum_i |H_{ij}|^2} = \operatorname{mcn}(H). \tag{2.6}$$

The next inequality follows from the observation that $\operatorname{mcn}(H)$ is defined by a maximum over the standard basis vectors e_j, whereas $\|H\|$ is defined by a maximum over all vectors with norm 1, which contains the set of all e_j. Thus

$$\operatorname{mcn}(H) = \max_j \|He_j\| \le \max_{\|v\|=1} \|Hv\| = \|H\|. \tag{2.7}$$

Using the triangle inequality with $\|H\| = \max_{\|v\|=1}(\sum_i |\sum_j H_{ij}v_j|^2)^{\frac{1}{2}}$, we get

$$\|H\| \le \max_{\|v\|=1} \left(\sum_i \left| \sum_j |H_{ij}||v_j| \right|^2 \right)^{\frac{1}{2}} = \max_{\substack{\|v\|=1 \\ v_j \ge 0}} \left(\sum_i \left| \sum_j \operatorname{abs}(H)_{ij} v_j \right|^2 \right)^{\frac{1}{2}}. \tag{2.8}$$

Now by maximizing over all v with $\|v\| = 1$ instead of only those with $v_j \ge 0$, we get

$$\max_{\substack{\|v\|=1 \\ v_j \ge 0}} \left(\sum_i \left| \sum_j \operatorname{abs}(H)_{ij} v_j \right|^2 \right)^{\frac{1}{2}} \le \max_{\|v\|=1} \left(\sum_i \left| \sum_j \operatorname{abs}(H)_{ij} v_j \right|^2 \right)^{\frac{1}{2}} = \|\operatorname{abs}(H)\|. \tag{2.9}$$

The inequality in (2.9) is actually an equality due to the Perron–Frobenius theorem.

Since $\operatorname{abs}(H)$ is a symmetric matrix, there is an eigenvector z with eigenvalue equal in magnitude to $\|\operatorname{abs}(H)\|$. Clearly this eigenvector satisfies $\|\operatorname{abs}(H)z\|_1 = \|\operatorname{abs}(H)\|\|z\|_1$. Using this and maximizing over all nonzero vectors, we have

$$\|\operatorname{abs}(H)\| = \frac{\|\operatorname{abs}(H)\|\|z\|_1}{\|z\|_1} = \frac{\|\operatorname{abs}(H)z\|_1}{\|z\|_1} \le \max_{v\ne 0} \frac{\|\operatorname{abs}(H)v\|_1}{\|v\|_1} = \|\operatorname{abs}(H)\|_1. \tag{2.10}$$

The inequality now follows from the fact that $\|H\|_1 = \|\mathrm{abs}(H)\|_1$, since

$$\|\mathrm{abs}(H)\|_1 = \max_j \sum_i |\,\mathrm{abs}(H)_{ij}| = \max_j \sum_i |H_{ij}| = \|H\|_1. \tag{2.11}$$

For the next inequality, we use the fact that $\|v\|_1 \leq \sqrt{N}\|v\|$ for all vectors v. This can be proved using the Cauchy–Schwarz inequality, $|\langle u, v\rangle| \leq \|u\|\|v\|$, by taking $u_i = v_i/|v_i|$. Let $j_{\max}$ be the index j that maximizes $\sum_i |H_{ij}|$. Thus $\|H\|_1 = \sum_i |H_{ij_{\max}}| = \|He_{j_{\max}}\|_1$. Using these two inequalities, it follows that

$$\|H\|_1 = \|He_{j_{\max}}\|_1 \leq \sqrt{N}\|He_{j_{\max}}\| \leq \sqrt{N}\max_j \|He_j\| = \sqrt{N}\,\mathrm{mcn}(H). \tag{2.12}$$

The last inequality is proved using the fact that for any j, $H_{ij} \leq \max_i H_{ij}$; thus

$$\mathrm{mcn}(H) = \max_j \sqrt{\sum_i |H_{ij}|^2} \leq \max_j \sqrt{N \max_i |H_{ij}|^2} = \sqrt{N}\max_{ij} |H_{ij}| = \sqrt{N}\,\mathrm{max}(H). \tag{2.13}$$

For each of these inequalities, there is a matrix that achieves equality. The first four inequalities are saturated when H is the identity matrix since the relevant norms are all equal to 1. The last two inequalities are satisfied with equality when H is the all-ones matrix (i.e., for all i, j, $H_{ij} = 1$), since then $\|H\|_1 = N$, $\mathrm{mcn}(H) = \sqrt{N}$, and $\mathrm{max}(H) = 1$. □

Since Theorem 2 involves $\|\mathrm{abs}(H)\|$, we would like to know the strongest inequality that relates $\|\mathrm{abs}(H)\|$ and $\|H\|$ directly. Lemma 1 gives $\|\mathrm{abs}(H)\| \leq \sqrt{N}\|H\|$, which is also the best possible inequality between the two norms. For example, when N is a power of 2, the matrix $H = R^{\otimes \log N}$ achieves equality, where $R := \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)/\sqrt{2}$ is the Hadamard matrix. It has $\|H\| = 1$, but $\|\mathrm{abs}(H)\| = \sqrt{N}$. This shows that Theorem 2 might not be as powerful as we would like, since for some Hamiltonians, the simulation method of Theorem 2 may be infeasible even when $\|H\|$ is small.

2.4 Relationships between matrix norms for sparse Hamiltonians

Although the above inequalities cannot be tightened in general, there can of course be stronger relationships among the various norms for special classes of Hamiltonians. When H is sparse, the norms mentioned above can differ at most by a factor of $\mathrm{poly}(\log N)$. Specifically, if H is d-sparse (i.e., it has at most d nonzero entries per row), then

$$\mathrm{max}(H) \leq \mathrm{mcn}(H) \leq \|H\| \leq \|\mathrm{abs}(H)\| \leq \|H\|_1 \leq \sqrt{d}\,\mathrm{mcn}(H) \leq d\,\mathrm{max}(H). \tag{2.14}$$

The first four inequalities are from Lemma 1. The inequality $\|H\|_1 \leq \sqrt{d}\,\mathrm{mcn}(H)$ follows from (2.12) using the fact that $\|v\|_1 \leq \sqrt{d}\|v\|$ when v has at most d non-zero entries (this can be proved using the Cauchy–Schwarz inequality as before). The last inequality follows from (2.13) and the inequality $\sum_i |H_{ij}|^2 \leq d \max_i |H_{ij}|^2$, which holds when H is d-sparse.

Just as in Lemma 1, for each of these inequalities and for all $d > 0$, there is a matrix that achieves equality. The identity matrix, which is d-sparse for all $d > 0$ saturates the first four inequalities. For the last two inequalities, choose the matrix which has its first $d \times d$ entries equal to 1, and the rest equal to zero. More precisely, the (i, j) entry of this matrix is 1 when i and j are both less than or equal to d. This matrix is d-sparse, and has $\|H\|_1 = d$, $\mathrm{mcn}(H) = \sqrt{d}$, and $\max(d) = 1$.

For sparse Hamiltonians, $d = \mathrm{poly}(\log N)$, which means that for sparse Hamiltonians all the above-mentioned norms are equivalent up to polynomial factors in $\log N$. Although we had used the spectral norm in the definition of an efficient simulation (recall that we required the scaling to be $\mathrm{poly}(\log N, \|Ht\|, 1/\epsilon)$), this lemma shows that if we quantify the complexity of simulating sparse Hamiltonians using a different norm, this will change the complexity by at most a $\mathrm{poly}(\log N)$ factor. Since we allow Hamiltonian simulation algorithms to run in time polynomial in $\log N$ anyway, this will not change the definition of an efficient simulation.

Chapter 3

Simulating sparse Hamiltonians

3.1 Problem description

In this chapter we consider the problem of simulating sparse Hamiltonians, which are specified by a black box which can compute the location and matrix entries of the nonzero elements in any row of the Hamiltonian, as discussed in Section 1.5.

More formally, the problem is to approximately implement the unitary $e^{-\mathrm{i}Ht}$ for a d-sparse and efficiently row-computable N-dimensional Hamiltonian H for time t. As input, we are given black-box access to H, and the values of d, t, ϵ, and N. We have to implement a unitary U, such that for all quantum states $|\psi\rangle$, the density operator corresponding to $U|\psi\rangle$ is ϵ-close to that of $e^{-\mathrm{i}Ht}|\psi\rangle$ in trace distance.

As mentioned in Section 1.5, the property of row computability allows a convenient black-box formulation of the problem that abstracts away the details of computing matrix entries and locations. The Hamiltonian is provided as a black-box function f, which accepts a row index and an integer $i \in \{1, 2, \ldots, d\}$ and outputs the column index and matrix element corresponding to the i^{th} nonzero entry in that row, if one exists. More precisely, if the nonzero elements in row x are $y_1, y_2, \ldots, y_{d_x}$, where $d_x \leq d$ is the degree of x, then $f(x, i) = (y_i, H_{x,y_i})$ for $i \leq d_x$ and $f(x, i) = (x, 0)$ for $i > d_x$.

For each row x, we allow the order in which the y_i are given by the oracle to be arbitrary (but fixed). We do not assume that there is a convenient ordering, such as the increasing order of labels (i.e., we do not assume that $y_1 < y_2 < \ldots < y_{d_x}$). To use the black box in a quantum circuit, we define an equivalent unitary matrix U_f which performs the operation $U_f|x, i, 0\rangle = |x, i, f(x, i)\rangle$.

Let us denote the minimum number of queries to U_f required to approximately simulate $e^{-\mathrm{i}Ht}$, up to error ϵ, by $Q_\epsilon(H, t)$. This is the query complexity of implementing the given Hamiltonian, and this is what we wish to minimize.

3.2 Previous best algorithm

The best known algorithm for high-precision simulation of sparse Hamiltonians, prior to our work, is the algorithm of Ref. [5]. A common approach to this problem, which is also the approach of Ref. [5], breaks the problem into two subproblems, which we call the Hamiltonian decomposition problem and the Hamiltonian recombination problem. First the Hamiltonian is decomposed into a sum of easy-to-simulate Hamiltonians; then these Hamiltonians are simulated for short times in a specific manner so that the overall simulation is approximately the same as that of H.

Since we will also follow the decomposition–recombination strategy, we review this approach as applied in Ref. [5]. The given Hamiltonian H is decomposed into a sum of m Hamiltonians, $H = \sum_{j=1}^{m} H_j$. Let $Q(H_j)$ denote the number of queries required to simulate H_j for time t_j given black-box access to H. (Note that we are given black-box access to H, not H_j.) In general, the number of queries required will depend on t_j and the error threshold, but in the simulations used here $Q(H_j)$ is independent of t_j and H_j will be simulated exactly. Specifically, $Q(H_j)$ includes the number of queries required to decompose H into H_j as well as to simulate H_j. In Ref. [5], the Hamiltonians H_j are 1-sparse, and their decomposition uses $O(\log^* N)$ queries to a black box for H_j. Since a 1-sparse Hamiltonian can be simulated for any time with 2 queries given an oracle for the 1-sparse Hamiltonian [8, 10], $Q(H_j) = O(\log^* N)$.

Theorem 3 (Hamiltonian edge decomposition [5])**.** *If H is an $N \times N$ Hamiltonian with maximum degree d, then there exists a decomposition $H = \sum_{j=1}^{m} H_j$, where each H_j is 1-sparse, such that $m = 6d^2$ and each query to any H_j can be simulated by making $O(\log^* N)$ queries to H.*

These Hamiltonians are then recombined using the Lie–Trotter formula, which expresses the time evolution due to H as a product of time evolutions due to the individual H_j. The unitary $e^{-\mathrm{i}Ht}$ is approximated by a product of exponentials $e^{-\mathrm{i}H_j t_j}$, such that the maximum error in the final state does not exceed ϵ. We need an upper bound on the number of exponentials required, N_{exp}, which is given by the following theorem.

Theorem 4 (Hamiltonian recombination [5])**.** *Let k be any positive integer. If $H = \sum_{j=1}^{m} H_j$ is a Hamiltonian to be simulated for time t by a product of exponentials $e^{-\mathrm{i}H_j t_j}$, and the permissible error (in terms of trace distance) is bounded by $\epsilon \leq 1 \leq 2m5^{k-1}\|Ht\|$, then the number of exponentials required, N_{exp}, is bounded by*

$$N_{\mathrm{exp}} \leq 5^{2k} m^2 \|Ht\| \left(\frac{m\|Ht\|}{\epsilon} \right)^{1/2k}. \tag{3.1}$$

Using the upper bound on the number of exponentials and the maximum number of queries needed to simulate any exponential, the total number of queries needed to simulate the Hamiltonian H satisfies $Q_\epsilon(H,t) \leq N_{\exp} \times \max_j Q(H_j)$. With $Q(H_j) = O(\log^* N)$ and $m = 6d^2$, we get

$$Q_\epsilon(H,t) = O\left(5^{2k} d^4 (\log^* N) \|H\| t \left(\frac{d^2 \|Ht\|}{\epsilon}\right)^{1/2k}\right). \tag{3.2}$$

We see that $Q_\epsilon(H,t)$ is almost linear in t, which is almost optimal due to the no–fast-forwarding theorem (Theorem 1). However, the dependence on d is not optimal. In the next section we improve the dependence on d without affecting the other terms.

We propose a new strategy for decomposing sparse Hamiltonians and solving the Hamiltonian decomposition problem, and reuse the Hamiltonian recombination theorem (Theorem 4). Our strategy breaks up the Hamiltonian into only $m = 6d$ parts, but increases $Q(H_j)$ to $O(d + \log^* N)$, improving the overall dependence on d and N.

3.3 Our algorithm

In this section we exhibit a different strategy for solving the Hamiltonian decomposition problem which improves Theorem 3. The Hamiltonian decomposition problem is the problem of decomposing a Hamiltonian H into a sum of m Hamiltonians H_j such that given a label $1 \leq j \leq m$ and a time t_j, the unitary $e^{-\mathrm{i}H_j t_j}$ can be efficiently simulated.

We solve this problem by decomposing the Hamiltonian into $m = 6d$ galaxies. To achieve this, we first decompose the given graph into d forests using the forest decomposition technique of Paneconesi and Rizzi [33]. The idea is to assign one of at most d colors to each edge of the graph (not necessarily a proper edge coloring) such that the edges of any particular color form a forest. Not only is this decomposition possible, but it has some special properties that are required later in Lemma 3.

Lemma 2 (Forest decomposition)**.** *For any Hamiltonian H of maximum degree d, there exists a decomposition $H = \sum_{c=1}^{d} H_c$ and an assignment of directions to the edges such that each H_c is a forest of arborescences. Furthermore, given a color c and a vertex v, we can determine v's parent in H_c with one query (or determine that it is a root) and with $O(d)$ queries we can determine the list of edges in H_c incident on v.*

Proof. We first describe a procedure that assigns a color c to each edge. H_c then consists of all edges colored c. To color the edges, every vertex proposes a color for each edge incident on it using the oracle in the following way: if y is x's i^{th} neighbor, then x proposes color

i for the edge xy. More formally, if $f(x, i) = (y, H_{x,y})$, then x proposes color i for the edge xy. Similarly, y proposes a color for the edge xy. The edge is now colored using the proposal of the vertex with higher label (i.e., if $x > y$ then the edge xy is colored with x's proposal). This coloring uses d colors, which is optimal up to constants since a d-sparse graph can have $dn/2$ edges, but forests have at most $n - 1$ edges.

Now we assign directions to the edges and show that each H_c has no cycles, which shows that each H_c is a directed forest. The edge xy is directed from x to y if $x < y$. This choice of directions results in a directed acyclic graph, which has no directed cycles. To rule out non-directed cycles, we note that any such cycle must contain a vertex v for which both the edges of the cycle point toward v. This means the label of v is greater than those of its two neighbors. Thus the color of these edges was decided by v, which cannot happen since a vertex always proposes different colors for all the edges incident on it.

To show that each tree in H_c is an arborescence, let us show that it has a unique root. If a directed tree has more than one root, it must have a vertex with more than one parent. This again leads to the situation where a vertex has two incoming edges of the same color, which is not possible since these edges are colored by this vertex's proposal.

To show that the parent of a vertex can be determined with one query, note that if p_v is the parent of vertex v in H_c, then the edge from p_v to v must be directed toward v. Thus the color of this edge is decided by v. If this edge is in H_c, it is colored c. So if v has a parent, it must be the c^{th} neighbor of v. With one query to the oracle, we can determine the c^{th} neighbor of v. If there is no such neighbor, this vertex has no parent and is a root in H_c. Otherwise the output of the oracle contains the label of the parent.

Finally, we show how to determine the list of edges in H_c incident on x with $O(d)$ queries. First we query the oracle at most d times to get the labels of all the neighbors of x. For a neighbor y where $y < x$, the edge between x and y is colored by c only if y is x's parent in H_c. Thus we can discard all edges xy where $y < x$ but y is not the parent of x. When $y > x$, an edge between x and y is colored c only if x is y's parent in H_c, and it takes one query to verify this for each y. Thus with at most d additional queries we can determine if all such edges are colored c. □

We have now decomposed the Hamiltonian into a sum of d directed forests. Let T be such a forest. We will decompose T into a sum of 6 galaxies, $T = T_1 + T_2 + \cdots + T_6$. This is achieved by using an extension of the "deterministic coin tossing" protocol of Cole and Vishkin [13] by Goldberg, Plotkin and Shannon [22]. Their protocol gives a proper vertex coloring of an arborescence using only 6 colors making $O(\log^* N)$ queries. Vertex coloring a forest of arborescences gives a galaxy decomposition of the forest, since all the edges that point to vertices of a particular color form a galaxy.

Lemma 3 (Vertex coloring a forest)**.** *If T is a forest of arborescences, and the parent of a vertex can be determined with one query to an oracle for T, then there exists a proper*

vertex coloring of T using 6 colors, such that the color of any vertex can be determined by making $O(\log^* N)$ queries.

Proof. We first describe the vertex-coloring procedure for the forest. A simple observation is that we already possess a vertex coloring of the forest: the labels of the vertices. This is a trivial proper vertex coloring using N colors. Now we use a procedure that decreases the number of colors used to $O(\log N)$. Then we can run several rounds of this procedure to decrease the number of colors down to 6. Let $c_j(x)$ be the color assigned to vertex x at the beginning of the j^{th} round of the procedure. At the beginning of the first round, we have $c_1(x) = x$.

Let x be a vertex with parent p_x. Assume that we started with a proper vertex coloring at the beginning of round j. Since we have a proper coloring, $c_j(x) \neq c_j(p_x)$. Let k be the index of the first bit at which x and p_x differ, and let b be the value of the k^{th} bit of x. The new color for vertex x is the concatenation of k and b, denoted (k, b). If x is the root, we take $k = 0$. We claim that if each vertex performs this procedure, the result is a proper vertex coloring.

For a contradiction, suppose there are two adjacent vertices that have been assigned the same color in round j. Without loss of generality, one of them is the parent of the other, so let them be y and its parent p_y. Since we started with a proper coloring at the beginning of round j, $c_j(y) \neq c_j(p_y)$, but now $c_{j+1}(y) = c_{j+1}(p_y)$. Let $c_{j+1}(y) = (k, b)$ where, by definition, k is the bit at which $c_j(y)$ and $c_j(p_y)$ differ, and b is the value of the k^{th} bit of $c_j(y)$. Since $c_{j+1}(p_y)$ also equals (k, b), the k^{th} bit of $c_j(p_y)$ is b. But $c_j(y)$ and $c_j(p_y)$ are supposed to differ at the k^{th} bit, so this is a contradiction. Therefore the coloring procedure is valid.

It remains to show that if the colors of the vertices are updated in this way, we reduce the number of colors to 6 in $O(\log^* N)$ rounds. Let L_j be the number of bits used to represent colors at the beginning of round j. Since we start with N colors, $L_1 = \lceil \log N \rceil$. In any step, the new color is (k, b), where the size of k is the logarithm of the size of the previous color and b is just one bit. Thus $L_{j+1} = \lceil \log(L_j) \rceil + 1$. This gives us the following recursion relation:

$$L_1 = \lceil \log(N) \rceil \qquad L_{j+1} = \lceil \log(L_j) \rceil + 1.$$

This recurrence relation can be solved by noting that $L_j \leq 2\lceil \log^{(j)}(N) \rceil$ for all $j \geq 1$, where $\log^{(j)}(x)$ is defined by $\log^{(1)} x = \log x$ and $\log^{(j)} x = \log(\log^{(j-1)} x)$. This yields $L_j = 3$ when $j = \log^* N$ [22].

Further rounds cannot decrease L_j below 3, since $L_{j+1} = L_j$ when $L_j = 3$. A length of 3 bits allows the use of at most 8 colors. Now we run the procedure once more. Since there are 3 possible values for k, and 2 for b, there are at most 6 different colors now. The total number of rounds is now $\log^* N + 1$.

To show that the color of a vertex can be determined with $O(\log^* N)$ queries, we note that the color of vertex x at the end of the first round depends solely on the labels x and p_x. In general, the color of vertex x at the end of j rounds depends only on the labels of its first j ancestors. To determine x's color after $\log^* N + 1$ rounds, we only need the labels of its $\log^* N + 1$ ancestors, which can be found with $\log^* N + 1$ queries, since the parent of a vertex can be found with one query. □

We have shown that a Hamiltonian can be decomposed into d forests of arborescences, each of which can be vertex-colored with 6 colors. If we consider all the edges of one of the d forests that point to a vertex of a particular color, this graph is a galaxy. So this decomposes the original Hamiltonian into $6d$ galaxies. For this particular decomposition of the Hamiltonian to be useful, we need to show that galaxies can be simulated easily.

Theorem 5 (Galaxy simulation)**.** *If H_j is a Hamiltonian whose graph is a galaxy of maximum degree d, and the oracle can identify which vertices are centers of stars, then the unitary operator $e^{-\mathrm{i}H_j t}$ can be simulated using $O(d)$ calls to an oracle for H_j.*

Proof. The key idea is that given a vertex v, we can learn everything about the star to which v belongs with $O(d)$ queries. If v is the center of the star, the oracle identifies it as the center, so we can query all its neighbors to learn everything about the star with at most d queries. If v is not the center, we can determine the center, which is the only neighbor of v, with only one query, and then learn the rest of the star with at most d queries.

Let $R(x)$ denote all the information about the star to which x belongs: the label of the center, the labels of the other vertices in some fixed order, and the weights of all the edges. It is essential that $R(x)$ depend only the star and not the particular vertex x chosen from the star, so that if x and y belong to the same star then $R(x) = R(y)$. Since we know that $R(x)$ can be computed with $O(d)$ queries, we can implement the unitary U given by $U|x, 0\rangle = |x, R(x)\rangle$ with $O(d)$ queries.

The Hamiltonian we are trying to simulate, H_j, is a galaxy. Thus, if c is the center of a star, and its neighbors are y_i with edge weights w_i, then $H_j|c\rangle = \sum_i w_i|y_i\rangle$. If x is not the center of a star, and the edge between x and the center c has weight w_x, then $H_j|x\rangle = w_x|c\rangle$. Let K be a Hamiltonian which is similar to H_j, but acts on the input state $|x, R(x)\rangle$ instead of $|x\rangle$. That is, $K|c, R(c)\rangle = \sum_i w_i|y_i, R(y_i)\rangle$ when c is the center, and $K|x, R(x)\rangle = w_x|c, R(c)\rangle$ otherwise. Note that although the second register looks different, it is unaffected by K since $R(x)$ depends only on the star and not the vertex. Combining K with the unitary U above, we see that $H_j = U^\dagger K U$. In words, U first computes all the information about the star in another register, K performs the required Hamiltonian, and the $U^\dagger$ uncomputes the second register, which was unaffected by K.

This simulation is efficient since K can be simulated efficiently. More importantly for our purposes, K requires no queries to implement, since all the information about the star

is already present in the second register. Thus the operation $H_j = U^\dagger K U$ requires only as many queries as U and $U^\dagger$ require, which is $O(d)$. □

Combining Lemma 2, Lemma 3, and Theorem 5 gives our Hamiltonian decomposition theorem.

Theorem 6 (Hamiltonian star decomposition)**.** *For a d-sparse Hamiltonian H, there exists a decomposition $H = \sum_{j=1}^{m} H_j$, where each H_j is a galaxy, such that $m = 6d$ and each galaxy H_j can be simulated for time t_j using $Q(H_j) = O(d + \log^* N)$ queries to an oracle for H.*

Proof. From Lemma 2, Lemma 3, and Theorem 5, we know that the claimed decomposition is possible and the resulting galaxies can be simulated. It remains to show that any H_j can be simulated for time t_j using $O(d + \log^* N)$ queries to an oracle for H (not the H_j). Thus the number of queries includes the cost of the decomposition and the cost of simulating H_j.

To show this, let us implement H_j on the basis state $|x\rangle$. If the implementation is correct on all basis states, it is correct for all input states by linearity. We are given an index j, which represents two indices, $1 \leq c \leq d$ and $1 \leq t \leq 6$. We want to simulate the galaxy formed by edges in H_c directed toward vertices colored t by the vertex coloring algorithm of Lemma 3.

From the proof of Theorem 5, it is clear that if we can compute $R(x)$, then we can implement U, and thereby simulate the desired Hamiltonian. $R(x)$ contains all the information about the star to which x belongs. Using the result of Lemma 2, we can determine the list of x's neighbors in H_c using $O(d)$ queries. By the result of Lemma 3, with $O(\log^* N)$ queries we can determine x's color according to the vertex coloring algorithm. We now have x's color and a list of its neighbors in H_c.

If x's color is not t, then x must be the center of a (possibly empty) star in H_j. The only edges in this star point toward vertices of color t, so we compute the colors of all the children of x in H_c. These can be computed using only the labels of their $\log^* N + 1$ nearest ancestors, which are all common ancestors. Thus we can compute the colors of all of x's children using $O(\log^* N)$ queries in total. Now we know the star around x, and thus $R(x)$, using $O(d + \log^* N)$ queries.

If x's color is t, then x's parent is the center of star. The parent of x, p_x, can be determined with one query. Since x and p_x are in the same star, $R(x) = R(p_x)$. Since p_x is the center of a star, we can compute $R(p_x)$ as described above; thus we can also compute $R(x)$.

We have shown that for any x, we can compute $R(x)$ with $O(d + \log^* N)$ queries. Thus the unitary U in the proof of Theorem 5 can be simulated with $O(d + \log^* N)$ queries. By Theorem 5, this means we can implement H_j with $O(d + \log^* N)$ queries, as claimed. □

Now we can use our Hamiltonian decomposition theorem, instead of Theorem 3, with the Hamiltonian recombination theorem (Theorem 4). Since we have $Q(H_j) = O(d + \log^* N)$ from Theorem 6 and $m = 6d$ from Lemma 2 and Lemma 3, we get our final result using $Q_\epsilon(H,t) \leq N_{\exp} \times \max_j Q(H_j)$:

$$Q_\epsilon(H,t) = O\left(5^{2k} d^2 (d + \log^* N) \|H\| t \left(\frac{d\|Ht\|}{\epsilon}\right)^{1/2k}\right). \tag{3.3}$$

When compared with the query complexity of (3.2), we see that this improves the scaling with d. The query complexity of (3.3) is always at least as good as that of (3.2). Furthermore, when $d = \Omega(\log^* N)$, which is likely to be the case when d is not constant, (3.3) has no $\log^* N$ term: the scaling (in terms of d and N) is $(d^3)^{1+o(1)}$, as compared to (3.2) which scales like $(d^4 \log^* N)^{1+o(1)}$.

3.4 Remarks and conclusion

An important observation here is that there is a trade-off between two parameters in this framework, the number of Hamiltonians H_j, which is the parameter m, and the maximum complexity of simulating each H_j, which is $\max_j Q(H_j)$. The overall query complexity is roughly $O(m^2 \max_j Q(H_j))$. At one end, we can choose $m = 1$, which means we are not decomposing the Hamiltonian at all. This choice does not lead to anything. At the other end, we can choose a large value for m, but a small one for $\max_j Q(H_j)$. This is the choice of Ref. [5], since they choose $\max_j Q(H_j)$ to be $O(\log^* N)$, which is almost a constant, but m to be $6d^2$, making the overall query complexity scale like $O(d^4 \log^* N)$. Our choice of parameters is somewhere in the middle, since we choose m to be $6d$, but the complexity of each simulation increases to $O(d + \log^* N)$. But since the scaling is quadratic in m, but only linear in $\max_j Q(H_j)$, the trade-off is favorable.

So far, we have measured the size of H using the spectral norm $\|H\|$, due to convention. However, if we express the simulation complexity in terms of a different norm, then both (3.2) and (3.3) can be improved to give slightly better bounds.

In the proof of Theorem 4, $\|H\|$ is used as a simple upper bound for $\max_j \|H_j\|$. However, omitting this step gives a slightly stronger version of Theorem 4 with $\|H\|$ replaced by $\max_j \|H_j\|$. For a 1-sparse Hamiltonian, we know from (2.14) that $\|H_j\| = \max(H_j) \leq \max(H)$, so $\|H\|$ can be replaced by $\max(H)$ in (3.2). This makes the final query complexity of their algorithm $(d^4 (\log^* N) \max(H))^{1+o(1)}$.

However, this also leads to an improvement of our algorithm. When H_j is a galaxy, $\|H_j\| = \mathrm{mcn}(H_j)$ (see (5.1) in Chapter 5), and since H_j is entry-wise upper bounded by

H, $\mathrm{mcn}(H_j) \leq \mathrm{mcn}(H)$. Thus $\|H\|$ can be replaced with $\mathrm{mcn}(H)$ in (3.3). This makes our algorithm scale like $(d^2(d + \log^* N)\,\mathrm{mcn}(H))^{1+o(1)}$.

Now the algorithms are difficult to compare, since they are stated in terms of different norms. To directly compare the two simulations, we can apply the bound $\mathrm{mcn}(H) \leq \sqrt{d}\max(H)$ from (2.14) to express both query complexities in terms of $\max(H)$. In these terms, we still find that star decomposition improves over edge coloring: our algorithm uses at most $(d^{2.5}(d + \log^* N)\max(H))^{1+o(1)}$ queries, whereas the algorithm of Ref. [5] scales like $(d^4(\log^* N)\max(H))^{1+o(1)}$.

Chapter 4

Simulating dense Hamiltonians

4.1 A no–fast-forwarding theorem for dense Hamiltonians

The no–fast-forwarding theorem (Theorem 1) establishes a lower bound on the simulation of sparse Hamiltonians. Let us restate the theorem, as stated in Section 1.5:

Theorem 1 (No–fast-forwarding theorem)**.** *For any positive integer N there exists a row-computable sparse Hamiltonian H with $\|H\| = 1$ such that simulating the evolution of H for time $t = \pi N/2$ within precision $1/4$ requires at least $N/4$ queries to H.*

Although the theorem is stated with $\|H\| = 1$, any of the norms in Lemma 1 could have been used, since the Hamiltonian used in the proof of the no–fast-forwarding theorem is 2-sparse, and by (2.14) the norms differ at most by a factor of 2. In particular, the theorem could be restated with $\max(H) \leq 1$ or $\|H\|_1 \leq 2$.

Since the choice of norm is unclear, it is conceivable that there are Hamiltonian simulation algorithms that run in time $O(\max(Ht))$ or $O(\mathrm{mcn}(Ht))$. To distinguish between the norms, we require a dense Hamiltonian. The aim of this section is use the proof techniques of Theorem 1 to establish a similar theorem for dense Hamiltonians. In particular, we show that there does not exist an algorithm for simulating dense Hamiltonians in time $O(\max(Ht))$ or $O(\mathrm{mcn}(Ht))$. However, this does not appear to rule out $O(\|Ht\|)$ simulations, which we rule out in the next section with a different idea (Theorem 7).

Although Theorem 8 in the next section is stronger than Theorem 7 below, we briefly present this straightforward generalization of the no–fast-forwarding theorem to show the extent of that approach as applied to the non-sparse case.

As in Theorem 1, we consider a black-box formulation of the problem of simulating dense Hamiltonians. We consider a different and stronger black box here, because the black box used in Theorem 1 is too weak for dense Hamiltonians (i.e., it is too easy to prove lower bounds). Instead we assume that there is a black box that can be queried with a row index j, which outputs the entire j^{th} row, as opposed to giving just one element of the j^{th} row. Obviously any lower bounds proved for this black box still hold for the one used for sparse Hamiltonian simulation.

Since the Hamiltonian is dense, the description of the j^{th} row can be exponentially large. This is not a problem, however, since our goal is to find a lower bound on query complexity, not time complexity. Even though each query takes exponential space, it counts as only one query.

In terms of this black-box model, we have the following:

Theorem 7. *For any positive integer N, there exists a non-sparse Hamiltonian H such that simulating the evolution of H for time $t = \pi N/2$ within precision $1/4$ requires at least $N/4$ queries to H. This Hamiltonian has $\|H\| = 1$, $\mathrm{mcn}(H) = \Theta(1/\sqrt{N})$, and $\max(H) = \Theta(1/N)$.*

Proof. The main idea, as in the proof of Theorem 1 [5], is to construct a Hamiltonian whose simulation for time $t = \pi N/2$ determines the parity of N bits. Since we know that computing the parity of N bits requires at least $N/2$ queries [3, 17], this Hamiltonian cannot be simulated with $o(N)$ queries. Moreover, we want this Hamiltonian to be non-sparse.

To motivate the construction of the Hamiltonian H, we start with a simple sparse Hamiltonian H_1 whose graph is just a line with $N+1$ vertices. Consider the Hamiltonian acting on vectors $|i\rangle$ with $i \in \{0, \ldots, N\}$. The nonzero matrix entries of H_1 are

$$\langle i \,|H_1|\, i+1\rangle = \langle i+1 \,|H_1|\, i\rangle = \sqrt{(N-i)(i+1)}/N \tag{4.1}$$

for $i \in \{0, 1, \ldots, N-1\}$. This Hamiltonian has $\|H_1\| = 1$, and simulating H_1 for $t = \pi N/2$ starting with the state $|0\rangle$ gives the state $|N\rangle$ (i.e., $e^{-iH_1 t}|0\rangle = |N\rangle$). If one has not seen this Hamiltonian before, it is not clear why $e^{-iH_1 t}|0\rangle = |N\rangle$ and how the entries of H_1 are chosen. This Hamiltonian is a multiple of the angular momentum operator J_x of a system with spin $N/2$. The equation $e^{-iH_1 t}|0\rangle = |N\rangle$ is analogous to the equation $e^{-i\pi J_x}|j, -j\rangle = |j, +j\rangle$, with $j = N/2$.

Now consider the Hamiltonian H_2, which is the same one chosen in Ref. [5], generated from an N-bit string $S_0 S_1 \ldots S_{N-1}$. H_2 acts on vertices $|i, j\rangle$, with $i \in \{0, \ldots, N\}$ and $j \in \{0, 1\}$. The nonzero matrix entries of this Hamiltonian are

$$\langle i, j \,|H_2|\, i+1, j \oplus S_i\rangle = \langle i+1, j \oplus S_i \,|H_2|\, i, j\rangle = \sqrt{(N-i)(i+1)}/N \tag{4.2}$$

for all i and j.

By construction, $|0,0\rangle$ is connected to either $|i,0\rangle$ or $|i,1\rangle$ for any i; it is connected to $|i,j\rangle$ if and only if $j = S_0 \oplus S_1 \oplus \ldots \oplus S_{i-1}$. Thus $|0,0\rangle$ is connected to either $|N,0\rangle$ or $|N,1\rangle$, and determining which is the case determines the parity of S. The graph of this Hamiltonian consists of two disjoint lines, one of which contains $|0,0\rangle$ and either $|N,0\rangle$ or $|N,1\rangle$ depending on the parity of S. Just as for H_1, starting with the state $|0,0\rangle$ and simulating H_2 for time $t = \pi N/2$ will give either $|N,0\rangle$ or $|N,1\rangle$, which determines the parity of S. Note that since H_2 is a permutation of $H_1 \oplus H_1$, $\|H_2\| = \|H_1\| = 1$. This is the Hamiltonian used in the proof of Theorem 1.

Finally, we construct the dense Hamiltonian H that has the properties stated in the theorem. As before, H is generated from an N-bit string $S_0 S_1 \ldots S_{N-1}$. Now H acts on vertices $|i,j,k\rangle$, with $i \in \{0,\ldots,N\}$, $j \in \{0,1\}$, and $k \in \{0, N-1\}$. The nonzero entries of H are given by

$$\langle i,j,k\,|H|\,i+1, j\oplus S_i, k'\rangle = \langle i+1, j\oplus S_i, k'\,|H|\,i,j,k\rangle = \sqrt{(N-i)(i+1)}/N^2 \qquad (4.3)$$

for all i, j, k, and k'. The graph of H is similar to that of H_2, except that for each vertex in H_2, there are now N copies of it in H. This Hamiltonian is dense because it has $\Theta(N^2)$ vertices and each vertex is connected to all N copies of its neighboring vertices, which gives at least N nonzero entries in each row.

Now we simulate the Hamiltonian starting from the uniform superposition over the copies of the $|0,0\rangle$ state, i.e., from the state $\frac{1}{\sqrt{N}}\sum_k |0,0,k\rangle$. The subspace $\mathrm{span}\{\sum_k |i,j,k\rangle\}$ of uniform superpositions over the third register is an invariant subspace of this Hamiltonian. Since the initial state lies in this subspace, the quantum walk remains in this subspace. In other words, the quantum walk on this dense graph starting from the chosen state reduces to the quantum walk on H_2 starting from the $|0,0\rangle$ state.

Now, just as before, the parity of S can be determined by simulating H for time $t = \pi N/2$. This gives the lower bound of $N/2$ queries.

To calculate the norms of this Hamiltonian, we observe that $H = H_2 \otimes J/N$, where J is the all-ones matrix of size $N \times N$. This gives $\|H\| = \|H_2\| \cdot \|J\|/N = 1$. Direct computation shows that $\max(H) = \Theta(1/N)$ and $\mathrm{mcn}(H) = \Theta(1/\sqrt{N})$. □

This theorem rules out algorithms that make only $O(\max(Ht))$ or $O(\mathrm{mcn}(Ht))$ queries, since for this Hamiltonian with $t = \pi N/2$ we have $\max(Ht) = \Theta(1)$ and $\mathrm{mcn}(Ht) = \Theta(\sqrt{N})$, both of which are disallowed by the lower bound of $\Omega(N)$. However, this does not distinguish between $\|H\|$ and $\|\mathrm{abs}(H)\|$ (since $\mathrm{abs}(H) = H$), and in fact $\|H\|_1 \sim 1$ as well. In the next section we construct examples with $\|\mathrm{abs}(H)\| \gg \|H\|$ in order to show that a general simulation using $O(\|Ht\|)$ steps (or even $\mathrm{poly}(\|Ht\|, \log N)$ steps) is not possible.

4.2 A stronger limitation for dense Hamiltonians

As discussed in Section 1.5, there are dense Hamiltonian simulation algorithms that use $O(\|\mathrm{abs}(Ht)\|)$ or $O(\|Ht\|_1)$ steps of a discrete-time quantum walk. However, in light of the no–fast-forwarding theorem for dense Hamiltonians, it might be reasonable to hope that dense Hamiltonians can be simulated in $O(\|Ht\|)$ steps, or at least in $\mathrm{poly}(\|Ht\|, \log N)$ steps. Indeed, if such simulations existed they could be applied to give new quantum algorithms for various problems [7]. In this section, we show that such simulations are, unfortunately, not possible in general.

The currently known dense Hamiltonian simulation algorithms rely on certain properties of the Hamiltonian that we call its *structural properties*. By this we mean the location of nonzero entries in H, which correspond to the location of edges in the graph of the Hamiltonian, and the magnitudes of the edge weights. (The remaining information about the Hamiltonian is the phase of each matrix entry H_{ij}.)

Given the structural information, the currently known algorithms can simulate dense Hamiltonians using $O(\|\mathrm{abs}(Ht)\|)$ or $O(\|Ht\|_1)$ calls to an oracle that gives the phase of the matrix entry at H_{ij}. We call this the *matrix entry phase oracle*. This oracle provides the value of $H_{ij}/|H_{ij}|$ when queried with the input (i, j). (The oracle may return any complex number of unit modulus—say, 1—when $H_{ij} = 0$.)

Note that in this formulation of the problem, the algorithm has access to a lot of information about the Hamiltonian it has to simulate. It has complete structural information about the Hamiltonian, which is already an exponential amount of information. It might seem that given so much information, an algorithm can simulate the Hamiltonian with very few queries to the matrix entry phase oracle. Indeed, with this information there does exist an algorithm which simulates the Hamiltonian with $O(\|\mathrm{abs}(Ht)\|)$ queries [7].

However, we show that given a matrix entry phase oracle and complete structural information, there exist some Hamiltonians that cannot be simulated with $\mathrm{poly}(\|Ht\|, \log N)$ queries. The following theorem is our main result.

Theorem 8. *No quantum algorithm can simulate a general Hamiltonian $H \in \mathbb{C}^{N\times N}$ for time t with $\mathrm{poly}(\|Ht\|, \log N)$ queries to a matrix entry phase oracle, even when given complete structural information about the Hamiltonian.*

Proof. The proof of the theorem is divided into two parts. First we show that there exists a set of Hamiltonians of size $N \times N$ that is hard to simulate on average for a particular time t. Specifically, we show that simulating a Hamiltonian selected uniformly at random from this set for a chosen time has average-case query complexity $\Omega(\sqrt{N/\log N})$. Then we show that a Hamiltonian simulation algorithm that makes $\mathrm{poly}(\|Ht\|, \log N)$ queries

would violate this lower bound. The lemmas used in this section are proved in the next section.

To show the lower bound, we need a black-box problem with an $\Omega(\sqrt{N/\log N})$ average-case lower bound, and a set of Hamiltonians whose simulation would solve this problem. We consider the problem of distinguishing strings $s \in \{-1,+1\}^M$ that have sum $-B$ or $+B$, given a black box for the entries of the string. When queried with an index $i \in \{1,2,\ldots,M\}$, the black box returns the value of $s_i \in \{-1,+1\}$, where $s = s_1 s_2 \ldots s_M$. The following lemma characterizes the query complexity of this problem.

Lemma 4. *Suppose we are given black-box access to a string $s \in \{-1,+1\}^M$, where s is chosen uniformly at random from the set of strings with $\sum_i s_i \in \{-B,+B\}$. Then determining $\sum_i s_i$ has average-case quantum query complexity $\Theta(M/B)$.*

Thus, determining whether the sum is $-\sqrt{M \log M}$ or $+\sqrt{M \log M}$, with the promise that one of these is the case, requires $\Omega(\sqrt{M/\log M})$ quantum queries on average. For each string s, we construct a Hamiltonian H_s whose simulation for a particular time allows us to distinguish the two possible cases (assuming s satisfies the promise).

Let H_s be a symmetric circulant matrix of size $N \times N$, where $N = 2M+1$ is odd. A circulant matrix is a matrix in which each row is rotated one element to the right relative to the preceding row. (The rotation "wraps around," i.e., the last element of a row becomes the first element of the next row.) Consequently, a circulant matrix is completely specified by its first row. However, since H_s is a *symmetric* circulant matrix, it is completely specified by just the first $M+1$ entries of the first row. Let the first entry of the first row be 0, and the next M entries of the first row be $s_1, s_2, \ldots, s_M$. In other words, the first $M+1$ entries of the first row of H_s are 0 followed by the string s. Then the fact that H_s is symmetric forces the remaining entries of the first row to be $s_M, s_{M-1}, \ldots, s_1$.

Given a black box for the entries of s, we can easily construct a black box for the entries of H_s. Indeed, one query to H can be simulated with at most one query to the string s. Sometimes no query to s is needed, since the diagonal entries of H_s are always 0.

Since H_s is a circulant matrix, it is diagonalized by the discrete Fourier transform. Its eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$ are

$$\lambda_k = 2\sum_{j=1}^{M} s_j \cos\left(\frac{2\pi jk}{N}\right), \quad \text{and in particular,} \quad \lambda_0 = 2\sum_{j=1}^{M} s_j. \tag{4.4}$$

Thus the time evolution of H_s can be used to learn whether $\sum_j s_j$ is $-\sqrt{M \log M}$ or $+\sqrt{M \log M}$. Since $\lambda_0 = 2\sum_j s_j$, the two cases can be distinguished by determining the sign of λ_0. Note that we know the eigenvector corresponding to λ_0: it is the first column of the discrete Fourier transform matrix, i.e., the uniform superposition over all computational basis states.

Consider the eigenvalues and eigenvectors of the unitary matrix $e^{-\mathrm{i}H\tau}$ that corresponds to evolving H for time $\tau = \pi/4\sqrt{M \log M}$. The eigenvectors of this matrix are the same as those of H, and each eigenvalue λ_k of H corresponds to the eigenvalue $\exp(-\mathrm{i}\lambda_k\tau)$ of $e^{-\mathrm{i}H\tau}$. Thus the uniform superposition is an eigenvector of $e^{-\mathrm{i}H\tau}$ with eigenvalue $\exp(-\mathrm{i}\lambda_0\tau) = \exp\left(-\mathrm{i}\pi \sum_i s_i/2\sqrt{M \log M}\right)$. Since $\sum_i s_i/\sqrt{M \log M}$ is either ± 1, the two possible eigenvalues are $\pm\mathrm{i}$. Because the eigenvector is known, the two possibilities can be easily distinguished by phase estimation [27, 11] on the unitary $e^{-\mathrm{i}H\tau}$. Since the problem of distinguishing these two cases has an $\Omega(\sqrt{M/\log M})$ average-case lower bound by Lemma 4, we get an $\Omega(\sqrt{M/\log M})$ average-case lower bound for simulating such Hamiltonians for time $\tau = \pi/4\sqrt{M \log M}$.

Now we want to show that a $\mathrm{poly}(\|Ht\|, \log N)$ Hamiltonian simulation algorithm violates this average-case lower bound. Since we are dealing with average-case complexity, we need to know the typical behavior of $\|H_s\|$ when s satisfies the promise. Let $\mathcal{S} := \{-1, +1\}^M$ and let $\mathcal{P}$ be the subset of strings in $\mathcal{S}$ that satisfy the promise that $\sum_j s_j$ is either $-\sqrt{M \log M}$ or $+\sqrt{M \log M}$. As a first step, let us see the behavior of $\|H_s\|$ for all strings $s \in \mathcal{S}$, not just those that satisfy the promise.

Lemma 5. *Let $H_s \in \mathbb{R}^{N \times N}$ be a symmetric circulant matrix of size $N = 2M + 1$ with the first $M + 1$ entries of the first row given by 0 followed by a string $s \in \mathcal{S}$. If s is chosen uniformly at random from $\mathcal{S}$, denoted $s \in_{\mathrm{R}} \mathcal{S}$, then for any $d > 0$,*

$$\Pr_{s \in_{\mathrm{R}} \mathcal{S}} \left(\|H_s\| \geq 4d\sqrt{M \log M} \right) \leq \frac{4 + o(1)}{M^{2d^2-1}}. \tag{4.5}$$

In fact, even stronger results of this kind are known [34, 23], but the above bound is easy to prove and sufficient for our purposes. Using Lemma 5, we wish to bound the spectral norm of H_s when $s \in_{\mathrm{R}} \mathcal{P}$. We can do so by first calculating the probability that a randomly selected string satisfies the promise.

Lemma 6. *If $s \in_{\mathrm{R}} \mathcal{S}$, then the probability that s satisfies the promise is*

$$\Pr_{s \in_{\mathrm{R}} \mathcal{S}} (s \in \mathcal{P}) = \Theta(1/M). \tag{4.6}$$

Using Lemmas 5 and 6, we can upper bound the probability that $\|H_s\|$ is large when s is chosen uniformly at random from $\mathcal{P}$. If X is the event that $\|H_s\| \geq 4d\sqrt{M \log M}$ and Y is the event that $s \in \mathcal{P}$, then $\Pr(X)$ is given by Lemma 5 and $\Pr(Y)$ is given by Lemma 6. In these terms, we can compute an upper bound for $\Pr(X|Y)$ as follows:

$$\Pr_{s \in_{\mathrm{R}} \mathcal{P}} \left(\|H_s\| \geq 4d\sqrt{M \log M} \right) = \Pr(X|Y) = \frac{\Pr(X \cap Y)}{\Pr(Y)} \leq \frac{\Pr(X)}{\Pr(Y)} = O(M^{2-2d^2}). \tag{4.7}$$

To achieve a contradiction, assume that for some constant $c > 0$, there exists a Hamiltonian simulation algorithm using $O\left((\|Ht\| \log M)^c\right)$ queries to simulate H for time t. By (4.7), we know that H_s almost always has spectral norm smaller than $4d\sqrt{M \log M}$ when $s \in_{\mathrm{R}} \mathcal{P}$. Since $\|H_s\| \leq \|H_s\|_1 = 2M$, we can compute the average-case query complexity of the claimed algorithm for simulating a uniformly random Hamiltonian H_s chosen from the promised set for time $\tau = \pi/4\sqrt{M \log M}$ as follows:

$$\begin{aligned}
\mathop{\mathbb{E}}_{s \in_{\mathrm{R}} \mathcal{P}} \left((\|Ht\| \log M)^c\right) &\leq \Pr_{s \in_{\mathrm{R}} \mathcal{P}} \left(\|H_s\| < 4d\sqrt{M \log M}\right) O\left(\left(4d\sqrt{M \log M}\tau \log M\right)^c\right) \\
&\quad + \Pr_{s \in_{\mathrm{R}} \mathcal{P}} \left(\|H_s\| \geq 4d\sqrt{M \log M}\right) O((2M\tau \log M)^c) \qquad (4.8) \\
&\leq O((d \log M)^c) + O\left(M^{c/2-2d^2+2} (\log M)^{c/2}\right), \qquad (4.9)
\end{aligned}$$

and by choosing $2d^2 > c/2 + 2$, we have

$$\mathop{\mathbb{E}}_{s \in_{\mathrm{R}} \mathcal{P}} \left((\|Ht\| \log M)^c\right) = O\left((\log M)^c\right). \qquad (4.10)$$

Thus the average-case query complexity of the claimed algorithm is $O\left((\log M)^c\right)$, which violates the lower bound of $\Omega(\sqrt{M/\log M})$. □

The proof technique above can be extended to rule out algorithms with query complexity sub-exponential in $(\|Ht\|, \log N)$ as well, by changing the promised set (i.e., the value of B used in Lemma 4) and choosing a larger value of d in Lemma 5. Exponential functions of $(\|Ht\|, \log N)$ cannot be ruled out, of course, since any Hamiltonian can be simulated by making $O(N^2)$ queries, which is exponential in $\log N$. On the other hand, if we insist that the query complexity of an algorithm depends only on $\|Ht\|$ (and not $\log N$), then the proof above can be modified to rule out algorithms whose time complexity is an arbitrary function of $\|Ht\|$. For example, there exists no Hamiltonian simulation algorithm that makes $\exp(\exp(\|Ht\|))$ queries.

Finally, we emphasize that even though the above proof uses average-case complexity and distributions over inputs, Theorem 8 is a statement about the worst-case complexity of simulating Hamiltonians.

4.3 Proofs of lemmas

In this section, we prove Lemmas 4, 5, and 6, which were used in the previous section.

Lemma 4. *Suppose we are given black-box access to a string $s \in \{-1, +1\}^M$, where s is chosen uniformly at random from the set of strings with $\sum_i s_i \in \{-B, +B\}$. Then determining $\sum_i s_i$ has average-case quantum query complexity $\Theta(M/B)$.*

Proof. We prove this by first showing the same lower bound for the worst-case problem using the quantum adversary method [2] and then reducing the worst-case problem to the average-case problem.

For the worst-case lower bound, we use the original quantum adversary method of Ambainis [2, Theorem 2]:

Theorem 9 (Ambainis)**.** *Let $f(x_1, \ldots, x_N)$ be a function of n $\{0,1\}$-valued variables and X, Y be two sets of inputs such that $f(x) \neq f(y)$ if $x \in X$ and $y \in Y$. Let $R \subset X \times Y$ be such that*

1. *For every $x \in X$, there exist at least m different $y \in Y$ such that $(x, y) \in R$.*
2. *For every $y \in Y$, there exist at least m' different $x \in X$ such that $(x, y) \in R$.*
3. *For every $x \in X$ and $i \in \{1, \ldots, n\}$, there are at most l different $y \in Y$ such that $(x, y) \in R$ and $x_i \neq y_i$.*
4. *For every $y \in Y$ and $i \in \{1, \ldots, n\}$, there are at most l' different $x \in X$ such that $(x, y) \in R$ and $x_i \neq y_i$.*

Then, any quantum algorithm computing f uses $\Omega\left(\sqrt{\frac{mm'}{ll'}}\right)$ queries.

We require two sets of inputs X and Y that have different outputs. Let X be the set of all strings for which $\sum_i s_i = -B$, and Y be the set for which $\sum_i s_i = +B$. We define the relation R as follows. Let an element $x \in X$ be related to an element of $y \in Y$ if and only if y can be reached from x by changing exactly $B/2$ -1s to $+1$s in the string x. Note that a string in X has exactly $(M/2 + B/2)$ -1s and $(M/2 - B/2)$ $+1$s.

Using these sets X and Y, and the relation defined above, it is easy to see that

$$m = m' = \binom{M/2 + B/2}{B} \quad \text{and} \quad l = l' = \binom{M/2 + B/2 - 1}{B - 1}, \tag{4.11}$$

so

$$\frac{m}{l} = \frac{m'}{l'} = \frac{1}{2}\left(\frac{M}{B} + 1\right). \tag{4.12}$$

Theorem 9 now provides a lower bound of $\Omega\left(\sqrt{mm'/ll'}\right) = \Omega(M/B)$ for the worst-case query complexity of this problem.

The worst-case query complexity can now be reduced to the average-case query complexity under the uniform distribution over all input strings satisfying the promise. To do this, we first apply a uniformly random permutation to the input string, and then with

probability $\frac{1}{2}$ multiply all the entries by -1 (and leave them unchanged with probability $\frac{1}{2}$). The resulting distribution is now uniform over all input strings satisfying the promise. If the string is not multiplied by -1, then the output of the permuted string is the same as the input string. If the string is multiplied by -1, then the output of the modified string is the complement of that of the original input.

The lower bound is tight due to a matching upper bound provided by the algorithm for approximate quantum counting [6]. To distinguish the two types of inputs, we can approximately count the number of +1s to accuracy $\epsilon = B/2M$, which requires $O(M/B)$ queries. □

The lower bound can also be shown by reducing it to a well-known hard problem. The problem chosen in Lemma 4 looks similar to the problem of computing the majority of M bits, which is the problem of determining whether $\Sigma_i s_i \geq 0$. If we let $B = 1$, then our problem looks like the hardest case of the majority problem, since the sum is just above or below zero. Unsurprisingly, it can be shown that this problem is as hard as the majority problem, which is known to have a lower bound of $\Omega(M)$ [3]. This problem on an input of size of M/B bits can be reduced to our problem by duplicating all the inputs B times. Thus if the original problem had $\Sigma_i s_i = 1$, then after duplicating the bits, the new sum will be $+B$, and the input size will be M. This again gives a lower bound of $\Omega(M/B)$.

Lemma 5. *Let $H_s \in \mathbb{R}^{N\times N}$ be a symmetric circulant matrix of size $N = 2M + 1$ with the first $M + 1$ entries of the first row given by 0 followed by a string $s \in \mathcal{S}$. If s is chosen uniformly at random from $\mathcal{S}$, denoted $s \in_{\mathrm{R}} \mathcal{S}$, then for any $d > 0$,*

$$\Pr_{s\in_{\mathrm{R}}\mathcal{S}}\left(\|H_s\| \geq 4d\sqrt{M\log M}\right) \leq \frac{4 + o(1)}{M^{2d^2-1}}. \tag{4.13}$$

Proof. The eigenvalues of H_s are $\lambda_r = 2\sum_{j=1}^{M} s_j \cos\frac{2\pi jr}{N}$, where $r \in \{0, 1, \ldots, N-1\}$. We wish to bound the probability that λ_r is large, so as to bound the probability of $\|H_s\| = \max_r |\lambda_r|$ being large. This is achieved by applying Hoeffding's inequality [25, Theorem 2].

Theorem 10 (Hoeffding's inequality)**.** *If $X_1, X_2, \ldots, X_M$ are independent and $a_j \leq X_j \leq b_j$ for all $1 \leq j \leq M$, then for any $t > 0$, we have*

$$\Pr\left(X - \mathbb{E}\left(X\right) \geq Mt\right) \leq \exp\left(\frac{-2M^2t^2}{\sum_{j=1}^{M}(b_j - a_j)^2}\right) \tag{4.14}$$

where $X = \sum_{j=1}^{M} X_j$.

If we take $X_j = 2s_j \cos\frac{2\pi jr}{N}$, then $X = \lambda_r = 2\sum_{j=1}^{M} s_j \cos\frac{2\pi jr}{N}$, each X_j is between -2 and $+2$, and $\mathbb{E}(X) = \sum_{j=1}^{M} \mathbb{E}(X_j) = 0$. By choosing $t = 4d\sqrt{\log M/M}$, we get

$$\Pr\left(\lambda_r \geq 4d\sqrt{M\log M}\right) \leq \exp\left(\frac{-2M^2(16d^2\log M/M)}{\sum_{j=1}^{M} 4^2}\right) = \frac{1}{M^{2d^2}}. \tag{4.15}$$

Since a similar inequality holds when X_j is replaced by $-X_j$, we get

$$\Pr\left(|\lambda_r| \geq 4d\sqrt{M\log M}\right) \leq \frac{2}{M^{2d^2}}. \tag{4.16}$$

Finally, since $\|H_s\| = \max_r |\lambda_r|$, a union bound gives

$$\Pr\left(\|H_s\| \geq 4d\sqrt{M\log M}\right) \leq \frac{2N}{M^{2d^2}}, \tag{4.17}$$

which implies the desired result. □

Lemma 6. *If $s \in_{\mathrm{R}} \mathcal{S}$, then the probability that s satisfies the promise is*

$$\Pr_{s\in_{\mathrm{R}}\mathcal{S}}(s \in \mathcal{P}) = \Theta(1/M). \tag{4.18}$$

Proof. Of the 2^M strings of length M, those with sum $-\sqrt{M\log M}$ or $+\sqrt{M\log M}$ have either $\frac{1}{2}(M+\sqrt{M\log M})$ $+1$s or $\frac{1}{2}(M+\sqrt{M\log M})$ -1s. Thus the total number of such strings is

$$2\binom{M}{\frac{M+\sqrt{M\log M}}{2}}. \tag{4.19}$$

We can asymptotically approximate this expression using a well-known approximation for the binomial coefficients (see for example equations 4.5 and 4.10 of Ref. [32]), which states that

$$\binom{n}{k} \sim \frac{2^n \exp\left(-2(k-n/2)^2/n\right)}{\sqrt{\pi n/2}} \tag{4.20}$$

provided $|k - n/2| = o(n^{2/3})$. Applying this to (4.19), we get

$$2\binom{M}{\frac{M+\sqrt{M\log M}}{2}} = \Theta\left(2^M \exp(-\log M/2)/\sqrt{M}\right) = \Theta(2^M/M), \tag{4.21}$$

which proves the claim. □

Chapter 5

Simulating structured Hamiltonians

5.1 Structured Hamiltonians

As mentioned in Section 1.5 and Chapter 3, we know how to simulate general sparse Hamiltonians efficiently. As before, *efficient* means the running time of the quantum algorithm should be $\text{poly}(\log N, \|Ht\|, 1/\epsilon)$. On the other hand, we saw in Chapter 4 that even with considerable information about a non-sparse Hamiltonian, it is not possible to simulate it efficiently.

In algorithmic applications of Hamiltonian simulation, often the Hamiltonian to be simulated is known in great detail. For example, in the NAND tree evaluation algorithm [16], the Hamiltonian to be simulated is a tree. In the particular case of evaluating a balanced NAND tree, the Hamiltonian is a balanced binary tree (except the last level, which might have one or zero children instead of two). Balanced binary trees are 3-sparse, and thus one could use any black-box algorithm for simulating sparse Hamiltonians.

If we just use the algorithm of Ref. [5] whose query complexity is given by (1.2), the query complexity turns out to be about $(\log^* N \times t)^{1+o(1)}$. However, the $\log^* N$ arises only because the black-box algorithm has to decompose the tree into edges. But since we know the structure of the tree, this decomposition can be done by us, thus removing the $\log^* N$ term from the complexity of simulating such Hamiltonians. The moral of the story is that in algorithmic applications we often know useful information about the Hamiltonian that we are trying to simulate, which might help us speed up the simulation.

This chapter considers the problem of simulating dense Hamiltonians efficiently, given access to structural information about the Hamiltonian. In Chapter 4, we defined the *structural information* to be information about the magnitudes of all the entries in the Hamiltonian. Often we do not really require all that information to simulate dense Hamiltonians; all we need is to be able to perform the discrete-time quantum walk in Theorem 2.

In the following sections, we assume that we have enough structural information about the dense Hamiltonian we wish to simulate to perform the discrete-time quantum walk efficiently.

As Theorem 8 shows, we cannot hope for general Hamiltonian simulation algorithms that scale polynomially in the spectral norm of the Hamiltonian, even with access to the structural information of the Hamiltonian. Although we do know algorithms that scale like $O(\|\mathrm{abs}(H)t\|)$, Lemma 1 tells us that $\|\mathrm{abs}(H)\|$ could be exponentially larger than $\|H\|$. However, we can achieve better scaling for special classes of Hamiltonians. For example, we saw in Section 2.1 that much stronger bounds hold for sparse Hamiltonians.

5.2 Graphs of low arboricity

We can also improve the inequalities of Lemma 1 for certain classes of non-sparse Hamiltonians. For example, consider the class of Hamiltonians whose graphs are trees. Such Hamiltonians can be efficiently simulated (given enough structural information) even when they are not sparse: in this case, Theorem 2 gives a simulation using $O(\|Ht\|)$ steps of a discrete-time quantum walk, because when the graph of H is a tree, $\|\mathrm{abs}(H)\| = \|H\|$.

Proposition 1. *If the graph of a Hermitian matrix H is a tree, then there exists a unitary matrix U such that $UHU^\dagger = \mathrm{abs}(H)$. In particular, $\|\mathrm{abs}(H)\| = \|H\|$.*

Proof. The matrix U is diagonal. To define U_{ii}, we arbitrarily fix some vertex as the root and consider the unique path from the root to vertex i. Let the path contain the vertices $i_0, i_1, \ldots, i_{p-1}, i_p, i$, where i_0 is the root and i_p is the parent of i. For each nonzero entry of H, define $\alpha_{ij} := H_{ij}/|H_{ij}|$. Then let $U_{ii} := 1$ if i is the root and $U_{ii} := \alpha_{i_0 i_1}\alpha_{i_1 i_2}\cdots\alpha_{i_{p-1} i_p}\alpha_{i_p i}$ otherwise. (Since these α_{ij} correspond to edges, the corresponding H_{ij} is nonzero, and thus α_{ij} is well defined.)

Since U is diagonal, $(UHU^\dagger)_{ij} = U_{ii}H_{ij}U_{jj}^*$. If i and j are not adjacent in the tree, then $(UHU^\dagger)_{ij} = H_{ij} = 0$ as required. Otherwise, suppose without loss of generality that j is the parent of i. Then $U_{ii}U_{jj}^* = |\alpha_{i_0 i_1}|^2|\alpha_{i_1 i_2}|^2\cdots|\alpha_{i_{p-1} i_p}|^2\alpha_{ji} = H_{ji}/|H_{ij}|$, so $(UHU^\dagger)_{ij} = |H_{ij}|$ as claimed. □

Thus Theorem 2 gives a simulation using $O(\|Ht\|)$ steps of a discrete-time quantum walk. However, there is another simulation method for such Hamiltonians that uses only $\mathrm{mcn}(Ht)^{1+o(1)}$ steps [7, Theorem 4]. It seems from Lemma 1 that the discrete-time quantum walk might be inferior to this, but due to Proposition 2 below it is, in fact, superior to the $\mathrm{mcn}(Ht)^{1+o(1)}$ simulation (except with respect to error scaling), since $\|Ht\| \le 2\,\mathrm{mcn}(Ht)$ when the graph of the Hamiltonian H is a tree.

If H can be expressed as the sum of a small number of Hamiltonians, each of whose graph is a forest, then H can be efficiently simulated when $\|H\|$ is small. Recall that a graph is said to have arboricity k if its adjacency matrix can be written as the sum of the adjacency matrices of k forests, but not $k-1$ forests.

Proposition 2. *If the graph of a Hamiltonian H has arboricity k, then $\|\mathrm{abs}(H)\| \leq 2k\,\mathrm{mcn}(H)$. Moreover, $\|\mathrm{abs}(H)\| \leq 2k\|H\|$ and $\|H\| \leq 2k\,\mathrm{mcn}(H)$.*

Proof. We begin by considering the case of a star graph. We show that if S is a Hamiltonian whose graph is a star,

$$\mathrm{mcn}(S) = \|S\| = \|\mathrm{abs}(S)\|. \tag{5.1}$$

By permuting the vertices, the first vertex can be chosen to be the one with maximum degree. Now the first column of the matrix S completely determines the Hamiltonian. Let the first column be w. The matrix S has first column w and first row $w^\dagger$. It is easy to see that $\mathrm{mcn}(S) = \|w\|$. S has exactly two nonzero eigenvalues, $\pm\|w\|$, corresponding to the eigenvectors $\|w\|e_1 \pm w$, where e_1 is the first column of the identity matrix. Since $\|S\|$ is the maximum eigenvalue, $\|S\| = \mathrm{mcn}(S)$.

Moreover, since $\mathrm{abs}(S)$ is a Hamiltonian whose graph is a star, we have $\|\mathrm{abs}(S)\| = \mathrm{mcn}(\mathrm{abs}(S))$. For any matrix H, $\mathrm{mcn}(\mathrm{abs}(H)) = \mathrm{mcn}(H)$, since the norms of the columns depend only on the magnitude of each entry. This proves the desired result, $\mathrm{mcn}(S) = \|S\| = \|\mathrm{abs}(S)\|$. These results also hold for galaxies, since the above norms $\nu(\cdot)$ all have the property that $\nu(A_1 \oplus \ldots \oplus A_n) = \max(\nu(A_1), \ldots, \nu(A_n))$.

To show the result for graphs of arboricity k, we begin by showing how a rooted tree can be decomposed into the sum of two forests of stars. The first forest contains all the edges in which the parent vertex is at a even distance from the root. The second forest contains the rest of the edges. This decomposes a rooted tree into two forests of stars, and similarly decomposes a forest into two forests of stars. Since the Hamiltonian has arboricity k, it can be decomposed into k forests, which can be decomposed into $2k$ forests of stars.

Thus $H = \sum_{l=0}^{2k} S_l$, where each of the S_l is a Hermitian matrix whose graph is a forest of stars. Moreover, the matrices S_l have no overlapping edges, i.e., if $(S_l)_{ij} \neq 0$ for some l, then $(S_l)_{ij} = 0$ for all other l. Therefore, for all i, j, l, $H_{ij} \geq (S_l)_{ij}$, which implies $\mathrm{mcn}(H) \geq \mathrm{mcn}(S_l)$ for all l. This gives

$$\mathrm{mcn}(H) \geq \frac{1}{2k} \sum_l \mathrm{mcn}(S_l) = \frac{1}{2k} \sum_l \|S_l\|. \tag{5.2}$$

Using the triangle inequality, we find

$$\|\mathrm{abs}(H)\| \leq \sum_l \|\mathrm{abs}(S_l)\| = \sum_l \|S_l\|. \tag{5.3}$$

Combining (5.2) and (5.3) gives the main result. Using Lemma 1 and the main result gives $\|\mathrm{abs}(H)\| \leq 2k\|H\|$ and $\|H\| \leq 2k \,\mathrm{mcn}(H)$. □

This shows that Hamiltonians with low arboricity can be simulated efficiently, since the algorithm of Theorem 2 gives an efficient algorithm as $\|\mathrm{abs}(H)\| \leq 2k\|H\|$.

Chapter 6

Concluding remarks and open problems

In Chapter 3 we described a Hamiltonian decomposition technique that reduces the query complexity of simulating sparse Hamiltonians from the previous best $(d^4 \log^* N)^{1+o(1)}$ to $(d^2(d + \log^* N))^{1+o(1)}$ in terms of N and d, without changing the dependence on the other parameters, especially the error scaling. As stated in Section 1.5, if we are willing to tolerate much higher error then we can achieve the query complexity given by (1.4) and reduce the dependence on d and N. But how much can the dependence on d and N be reduced, keeping the same error scaling? Are there interesting trade-offs between these parameters suggesting that all the parameters cannot be simultaneously optimized?

On the other hand, we can establish some lower bounds on sparse Hamiltonian simulation using Theorem 8 of Chapter 4. Although the theorem is about dense Hamiltonians, we can consider a sparse Hamiltonian whose first $d \times d$ entries are nonzero (making it d-sparse) and apply the lower bound in Theorem 8 with d instead of N. This shows that some query complexities, like $\mathrm{poly}(\|Ht\|)$ (with no d dependence) cannot be achieved. By tweaking the argument of Theorem 8, we can also rule out algorithms that scale like $o\left(\sqrt{\frac{d}{\log d}}\|Ht\|\right)$ or $o(d \max(Ht))$. For constant error, the complexity of the algorithm of Ref. [4] given by (1.4) is indeed $O(d \max(Ht))$. Since $o(d \max(Ht))$ is disallowed by our bound, this algorithm is optimal in terms of these parameters.

As explained in Section 3.4, the Hamiltonian decomposition–recombination framework has an interesting trade-off that we exploited to improve the sparse Hamiltonian simulation algorithm. It would be interesting to see if this framework can be used to further reduce the dependence on d. It would also be interesting to establish stronger limitations on the simulation of sparse Hamiltonians taking error dependence into account, since all the present lower bounds only require constant-error simulation.

Another interesting direction for future research is to answer the following question: Can we simulate Hamiltonians with error dependence poly($\log(1/\epsilon)$) instead of poly($1/\epsilon$)? This would be an extremely high-precision algorithm for simulating sparse Hamiltonians. Currently no such algorithm is known, and it might be the case that this is impossible. An answer to this question either way would be interesting.

Finally, can the sparse Hamiltonian simulation techniques be extended to cover some classes of non-sparse Hamiltonians? This is a very open-ended question, but it is conceivable that ideas used for sparse Hamiltonian simulation might help with simulating some non-sparse Hamiltonians.

In Chapter 4 we ruled out the possibility of a generic Hamiltonian simulation algorithm using only poly($\|Ht\|, \log N$) operations, even with considerable extra information about the Hamiltonian. However, we can nevertheless hope that some nontrivial classes of Hamiltonians can be simulated in poly($\|Ht\|, \log N$) steps even though $\|H\| \ll \|\mathrm{abs}(H)\|$.

One approach is to consider changing the basis in which the Hamiltonian is simulated. Clearly, if unitary transformations U and $U^\dagger$ can be performed efficiently, then H can be simulated efficiently if and only if $UHU^\dagger$ can. There must exist bases in which $UHU^\dagger$ is sparse (such as the basis in which it is diagonal), which may lead to efficient simulations of H. Some trivial classes of Hamiltonians can be simulated in this way, such as Hamiltonians that are tensor products of small factors. For example, the Hamiltonian $R^{\otimes n}$, where $R := \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)/\sqrt{2}$ is the Hadamard matrix, has $\|R^{\otimes n}\| = 1$ and $\|R^{\otimes n}\|_1 = \|\mathrm{abs}(R^{\otimes n})\| = 2^{n/2}$, yet the evolution according to $R^{\otimes n}$ is easy to simulate. A similar simulation for a case where the Hamiltonian is not a tensor product was used in Ref. [9].

An alternative method is to investigate ways of decomposing a Hamiltonian as a sum of Hamiltonians that can be efficiently simulated. For example, we can simulate Hamiltonians whose graphs have polynomial arboricity by decomposing them into stars (although we saw in Proposition 2 that such Hamiltonians can already by simulated efficiently by the method of Ref. [7] since they satisfy $\|\mathrm{abs}(H)\| = O(\|H\|)$). More generally, other graph decompositions could give rise to new efficient simulations.

Another interesting problem is to find classes of Hamiltonians that can be simulated in sublinear time. These correspond to quantum systems whose time evolution can be fast-forwarded. Some Hamiltonians may even be simulated in constant time, if $e^{-\mathrm{i}H\tau} = I$ after some constant time τ (for example, the case $R^{\otimes n}$ mentioned above has $\tau = 2\pi$).

We can also consider the problem of simulating dense Hamiltonians when given access to a matrix entry oracle. This oracle returns H_{ij} when queried with (i, j). It is clear that many Hamiltonians require exponential time to simulate with this oracle, but the exact query complexity can still be investigated. For example, since a Hamiltonian is described by an $N \times N$ matrix, it can clearly be simulated with N^2 queries. However, we can do much better.

Let us try using the algorithm of Chapter 3 in this case. A dense Hamiltonian corresponds to $d = N$, and in this case the sparse Hamiltonian oracle and the matrix entry oracle become the same. Plugging in $d = N$ in (3.3), we see that the query complexity exceeds N^3, which is clearly unsatisfactory. Instead, let us use the discrete-time quantum walk based algorithm whose complexity is given by (1.4). This gives an algorithm with query complexity $O\left(\frac{\|Ht\|}{\sqrt{\epsilon}} + N \max(H)t\right)$. For constant error, constant time and $\max(H) = 1$, this gives an $O(N)$ algorithm, which is optimal.

It would be interesting to see if this can be improved. For example, can the dependence on ϵ be improved? Perhaps there exists an algorithm with query complexity $O(Nt \max(H) \operatorname{poly}(\log(1/\epsilon))$. On the other hand, maybe the dependence on N can be improved given a promise on the structure of the Hamiltonian. For example, Berry and Childs [4] show that if the Hamiltonian has the property that $\max(H) = 1$, and the 2-norm of every column and row is equal to 1, then such a Hamiltonian can be simulated for constant time with constant error using only $\tilde{O}(N^{2/3})$ queries. Such Hamiltonians arise when encoding a unitary matrix into a Hamiltonian. More precisely, if U is a unitary matrix, then the Hamiltonian $H := \left(\begin{smallmatrix} 0 & U \\ U^\dagger & 0 \end{smallmatrix}\right)$ has these properties. It is not known if $\tilde{O}(N^{2/3})$ is optimal for the problem of simulating Hamiltonians arising from unitaries. A lower bound of $\Omega(\sqrt{N})$ can be shown, however, by a reduction from the search problem.

Finally, it would be interesting to see Hamiltonian simulation used as a subroutine in new quantum algorithms. For example, several graph-theoretic problems have an obvious Hamiltonian associated with them — the adjacency matrix of the input graph. Simulating the adjacency matrix might conceivably yield useful information about the structure of the graph.

There are probably many algorithmic applications of Hamiltonian simulation waiting to be discovered, and finding them is also an important open problem.

References

[1] D. Aharonov and A. Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proc. 35th ACM Symposium on Theory of Computing*, pages 20–29. ACM, 2003. 7

[2] A. Ambainis. Quantum lower bounds by quantum arguments. *J. Comput. Syst. Sci.*, 64(4):750–767, 2002. 31

[3] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. 25, 32

[4] D. W. Berry and A. M. Childs. The quantum query complexity of implementing black-box unitary transformations. *ArXiv preprint arXiv:0910.4157*, 2009. 8, 38, 40

[5] D. Berry, G. Ahokas, R. Cleve, and B. Sanders. Efficient quantum algorithms for simulating sparse Hamiltonians. *Commun. Math. Phys.*, 270(2):359–371, 2007. 7, 8, 16, 22, 23, 25, 34

[6] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *Quantum Computation and Quantum Information Science, AMS Contemporary Mathematics Series*, 305:53–74, 2002. 32

[7] A. M. Childs. On the relationship between continuous- and discrete-time quantum walk. *Communications in Mathematical Physics*, 294(2):581–603, March 2010. 8, 11, 27, 35, 39

[8] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proc. 35th ACM Symposium on Theory of Computing*, pages 59–68. ACM, 2003. 4, 16

[9] A. M. Childs, L. J. Schulman, and U. V. Vazirani. Quantum algorithms for hidden nonlinear structures. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 395–404, Washington, DC, USA, 2007. IEEE Computer Society. 39

[10] A. M. Childs. *Quantum information processing in continuous time*. PhD thesis, Massachusetts Institute of Technology, 2004. 7, 16

[11] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proceedings: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998. 29

[12] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the 1964 International Conference for Logic, Methodology and Philosophy of Science*, pages 24–30. Elsevier/North-Holland, 1965. 3

[13] R. Cole and U. Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf. Control*, 70(1):32–53, 1986. 18

[14] P. A. M. Dirac. Quantum mechanics of many-electron systems. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 123(792):714–733, 1929. 2

[15] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17(3):449–467, 1965. 3

[16] E. Farhi, J. Goldstone, and S. Gutmann. A quantum algorithm for the Hamiltonian NAND tree. *Theory Of Computing*, 4:169–190, 2008. 4, 34

[17] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Limit on the speed of quantum computation in determining parity. *Physical Review Letters*, 81(24):5442–5444, 1998. 25

[18] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum computation by adiabatic evolution. *ArXiv preprint quant-ph/0001106*, 2000. 4

[19] E. Farhi and S. Gutmann. Analog analogue of a digital quantum computation. *Physical Review A*, 57(4):2403–2406, 1998. 4

[20] R. Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982. 3

[21] R. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16(6):507–531, 1986. 3

[22] A. V. Goldberg, S. A. Plotkin, and G. E. Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM Journal on Discrete Mathematics*, 1(4):434–446, 1988. 18, 19

[23] G. Halász. On a result of Salem and Zygmund concerning random polynomials. *Studia Sci. Math. Hungar.*, 8:369–377, 1973. 29

[24] C. Helstrom. Quantum detection and estimation theory. *Journal of Statistical Physics*, 1(2):231–252, 1969. 5

[25] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, pages 13–30, 1963. 32

[26] P. Kaye, R. Laflamme, and M. Mosca. *An Introduction to Quantum Computing*. Oxford University Press, Inc., New York, NY, USA, 2007. 4

[27] A. Kitaev. Quantum measurements and the Abelian stabilizer problem. *Arxiv preprint quant-ph/9511026*, 1995. 29

[28] E. Knill. Approximation by quantum circuits. *Arxiv preprint quant-ph/9508006*, 1995. 8

[29] P.-S. Laplace. *Philosophical Essay on Probabilities*. Springer-Verlag, New York, 1995. Translated by A. I. Dale from the 5th French edition of 1825. 1

[30] S. Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996. 7

[31] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000. 4

[32] A. M. Odlyzko. *Asymptotic Enumeration Methods*. MIT Press, Cambridge, MA, USA, 1995. 33

[33] A. Panconesi and R. Rizzi. Some simple distributed algorithms for sparse networks. *Distrib. Comput.*, 14(2):97–100, 2001. 17

[34] R. Salem and A. Zygmund. Some properties of trigonometric series whose terms have random signs. *Acta Mathematica*, 91(1):245–301, 1954. 29

[35] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. 3

[36] E. Yudkowsky. An Alien God. Less Wrong, November 2007. 1