Chapter 3

Towards quantum advantage via topological data analysis

In this chapter we discuss the advantages that the quantum algorithm for Betti number estimation can achieve over classical algorithms. Firstly, in Section 3.1, we formally define the computational problems that the quantum algorithm for Betti number estimation can (efficiently) solve. In particular, it is clear that the techniques used in the quantum algorithm for Betti number estimation can also be used to estimate the number of small eigenvalues of arbitrary sparse Hermitian matrix, not just of combinatorial Laplacians. We take this as the starting point to define our natural generalization, which is called *low-lying spectral density* estimation (a version of which was also studied by Brandão [40]). Next, in Section 3.2, we show that this generalization is DQC1-hard, which suggests that the quantum-algorithmic methods behind the quantum algorithm for Betti number estimation may be a source of exponential separation between quantum and classical computers. We also discuss how to potentially close the gap between the topological data analysis problem of Betti number estimation and its generalization, which would show that the topological data analysis problem is itself classically intractable. Setting aside the complexity theory, in Section 2.2.3 we discuss the state-of-the-art classical algorithms for Betti number estimation and compare them with the quantum algorithms for Betti number estimation. We also discuss promising approaches for developing novel more efficient classical algorithms that take into account the specifics of the combinatorial Laplacian and we clearly delineate the theoretical hurdles that, at least currently, stymie such classical approaches. After discussing the strengths and weaknesses of the classical algorithms, we identify graphs for which the quantum algorithm can achieve (superpolynomial) speedups over the best known classical algorithms in Section 3.2.4.

3.1 Problem definitions

In this section we formally define the computational problems whose hardness we will study. We begin by defining the problems that capture the key steps of the quantum algorithm for Betti number estimation. Afterwards, we define the problems related to topological data analysis that the quantum algorithm for Betti number estimation aims to solve. We end this section by discussing the precise relationships between these problems.

The input matrices that we consider are sparse positive semidefinite matrices. We call a $2^n \times 2^n$ positive semidefinite matrix *sparse* if at most $\mathcal{O}(\mathrm{poly}(n))$ entries in each row are nonzero. A special class of sparse positive semidefinite matrices that we consider is the class of *log-local* Hamiltonians, i.e., n-qubit Hamiltonians that can be written as a sum

$$H = \sum_{j=1}^{m} H_j, \tag{3.1}$$

where each H_j acts on at most $\mathcal{O}(\log n)$ qubits and we assume that $m \in \mathcal{O}(\mathrm{poly}(n))$.

Our problems take as input a specification of a sparse positive semidefinite matrix, and we consider the following two standard cases. First, we consider the case where the input matrix is specified in terms of *sparse access*. That is, the input matrix $H \in \mathbb{C}^{2^n \times 2^n}$ is specified by quantum circuits that let us query the values of its entries, and the locations of the nonzero entries. More precisely, we assume that we are given classical descriptions of $\mathcal{O}(\mathrm{poly}(n))$-sized quantum circuits that implement the oracles O_H and $O_{H,\mathrm{loc}}$, which map

$$\begin{aligned} O_H &: |i,j\rangle\,|0\rangle \mapsto |i,j\rangle\,|H_{i,j}\rangle\,, \\ O_{H,\mathrm{loc}} &: |j,\ell\rangle\,|0\rangle \mapsto |j,\ell\rangle\,|\nu(j,\ell)\rangle\,, \end{aligned}$$

where $0 \leq i,j,\ell \leq 2^n - 1$, and $\nu(j,\ell) \in \{0,\dots,2^n-1\}$ denotes the location of the ℓ-th nonzero entry of the j-th column of H. Secondly, for log-local Hamiltonians, we also consider specifying the input matrix H by its *local-terms* $\{H_j\}$ as in Eq. (3.1).

In order to define the problem of generating approximations of eigenvalues that are sampled uniformly at random, we fix a suitable notion of an approximation of a probability distribution. In particular, this notion needs to take into account that the algorithm may err on both the estimation of the eigenvalue, and on the probability with which it provides such an estimation. For this we use the following definition presented in [200]. Let p be some probability distribution over the eigenvalues of a positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$. That is, sampling according to p will output an eigenvalue λ_k with probability $p(\lambda_k)$, and $\sum_{k=0}^{2^n-1} p(\lambda_k) = 1$. In this context, a probability distribution q with finite support $Y_q \subset \mathbb{R}$ is said to be an (δ,μ)-*approximation* of p if it satisfies

$$\sum_{y \in Y_q \,:\, |y-\lambda_k| < \delta} q(y) \geq (1-\mu)p(\lambda_k), \;\; \forall k \in \{0,\dots,2^{n-1}\}.$$

Intuitively, this means that if we draw a sample according to q, then this sample will be δ-close to an eigenvalue λ_k with probability at least $(1-\mu)p(\lambda_k)$[1] Using this

[1]This definition captures the distribution generated by quantum phase estimation: the eigenvector is chosen according to the distribution p specified by the input state, and the output is δ-close to

definition, we define the problem of generating approximations of eigenvalues that are sampled uniformly at random from the set of all eigenvalues as follows.

Sparse uniform eigenvalue sampling (SUES)[2]

Input:

1) A sparse positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$, with $||H|| \leq \text{poly}(n)$.
2) An estimation precision $\delta \in \Omega\left(1/\text{poly}(n)\right)$.
3) An error probability $\mu \in \Omega\left(1/\text{poly}(n)\right)$.

Output: A sample drawn according to a (δ, μ)-approximation of the uniform distribution over the eigenvalues of H.

In the quantum algorithm for Betti number estimation, samples from SUES are used to estimate the number of eigenvalues of the combinatorial Laplacian that are close to zero. Clearly, this same idea can be used to estimate the number of eigenvalues that lie in some given interval for arbitrary sparse positive semidefinite matrices. This is called the *eigenvalue count* [40], which for a positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$ and eigenvalue thresholds $a, b \in \mathbb{R}_{\geq 0}$ is given by

$$N_H(a,b) = \frac{1}{2^n} \sum_{k \,:\, a \leq \lambda_k \leq b} 1,$$

where $\lambda_0 \leq \cdots \leq \lambda_{2^n - 1}$ denote the eigenvalues of H. For a threshold $b \in \Omega\left(1/\text{poly}(n)\right)$, we shall refer to the quantity $N_H(0,b)$ as *low-lying spectral density.* This precisely captures our notion of the number of eigenvalues close to zero as discussed before. We define the problem of estimating the low-lying spectral density as follows.

Low-lying spectral density estimation (LLSD)[3]

Input:

1) A sparse positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$, with $||H|| \leq \text{poly}(n)$.
2) A threshold $b \in \Omega\left(1/\text{poly}(n)\right)$.
3) Precision parameters $\delta, \epsilon \in \Omega\left(1/\text{poly}(n)\right)$.
4) A success probability $\mu > 1/2$.

Output: An estimate $\chi \in [0,1]$ that, with probability at least μ, satisfies

$$N_H\left(0,b\right) - \epsilon \leq \chi \leq N_H(0, b+\delta) + \epsilon.$$

To provide some intuition behind this definition, note that it is supposed to precisely capture the problem that is solved by repeatedly sampling from SUES and computing the frequency of the eigenvalues that lie below the given threshold. We therefore require the precision parameter δ due to the imprecisions in the quantum phase estimation algorithm. Moreover, the precision parameter ϵ is necessary due to the sampling error we incur by estimating a probability by a relative frequency.

Now that we have formally defined the problems that capture the key steps of the

the corresponding eigenvalue with probability at least $(1-\mu)$.

[2] In view of noisy quantum computers, it is interesting to consider distributions that are close to these (δ, μ)-approximation in total variation distance. Sampling such distributions can be less demanding, however, the precise hardness remains to be analyzed.

[3] The exact version of this problem is closely related to #P [45].

quantum algorithm for Betti number estimation, we define the problems related to topological data analysis that they allow us to solve. For these problems we consider the adjacency matrix of the graph to be the input, as this is usually the input to the quantum algorithm for Betti number estimation. We define the problem of estimating Betti numbers as follows.

Betti number estimation (BNE)[4]
Input:
1) The adjacency matrix of a graph $G = ([n], E)$.
2) An integer $0 \leq k \leq n-1$.
3) A precision parameter $\epsilon \in \Omega(1/\text{poly}(n))$.
4) A success probability $\mu > 1/2$.

Output: An estimate $\chi \in [0,1]$ that, with probability at least μ, satisfies

$$\left| \chi - \frac{\beta_k^G}{\dim \mathcal{H}_k^G} \right| \leq \epsilon.$$

As discussed in Section 2.2.2, the quantum algorithm for Betti number estimation does not precisely solve the above problem. Namely, due to the lack of knowledge regarding lower bounds on the smallest nonzero eigenvalue of the combinatorial Laplacian, we are not always able to estimate the number of eigenvalues that are exactly equal to zero. Nonetheless, the quantum algorithm for Betti number estimation is still able to estimate the number of eigenvalues of the combinatorial Laplacian that are close to zero, which we called approximate Betti numbers. We define the problem of estimating approximate Betti numbers as follows.

Approximate Betti number estimation (ABNE)
Input:
1) The adjacency matrix of a graph $G = ([n], E)$.
2) An integer $0 \leq k \leq n-1$.
3) Precision parameters $\delta, \epsilon \in \Omega(1/\text{poly}(n))$.
4) A success probability $\mu > 1/2$.

Output: An estimate $\chi \in [0,1]$ that, with probability at least μ, satisfies

$$\frac{\beta_k^G}{\dim \mathcal{H}_k^G} - \epsilon \leq \chi \leq N_{\Delta_k^G}(0, \delta) + \epsilon.$$

We are now set to outline the problem that the quantum algorithm for Betti number estimation can efficiently solve. As discussed in Section 2.2.2, the quantum algorithm for Betti number estimation can efficiently solve ABNE, but only in certain regimes. In particular, one has to be able to efficiently prepare the maximally mixed state over all cliques of a given size from the adjacency matrix of the graph. As mentioned in Section 2.2.2, the efficiency of this state preparation depends on the graph's clique-density (i.e., probability that a uniformly random subset of vertices is a clique), or the graph's arboricity (which up to a factor 1/2 is equivalent to the

[4]The exact version of this problem is NP-hard [14]

maximum average degree of a subgraph). In short, the problem that the quantum algorithm for Betti number estimation can efficiently solve is a restriction of ABNE, where one is promised that the input graph is such that one can efficiently prepare the maximally mixed state over all cliques of a given size from the adjacency matrix (e.g., if the graph is sufficiently clique-dense or if it has a sufficiently bounded arboricity). We discuss this in more detail in Section 3.2.4, where we outline sufficient conditions on the graph's clique-density or arboricity that allow the quantum algorithm to efficiently solve ABNE.

Next, we will study the complexity of LLSD as it is a generalization of the problem that the quantum algorithm for Betti number estimation efficiently solves. Namely, as we will show in the following section, we can use LLSD to directly solve the problem that the quantum algorithm for Betti number estimation efficiently solves. Note that the input to the quantum algorithm for Betti number estimation is the adjacency matrix, and not the combinatorial Laplacian. Therefore, in order to use LLSD to solve the problem that the quantum algorithm for Betti number estimation efficiently solves, one first has to construct the appropriate input to LLSD. As it is computationally too expensive to enumerate all cliques in your graph, we cannot take the straightforward approach of first computing the combinatorial Laplacian to construct the desired input to LLSD. Fortunately, we can still use LLSD to efficiently solve the problem that the quantum algorithm for Betti number estimation efficiently solves by simulating sparse access to a matrix that is obtained by padding the combinatorial Laplacian with all-zeros columns and rows (see Section 3.1.1 for more details).

3.1.1 Relationships between the problems

In the previous section we have formally defined the computational problems whose complexity we will study. In this section we examine the reductions between LLSD and the problems related to topological data analysis in order to elucidate the precise relationships. An overview of the reductions can be found in Figure 3.1.

First, we discuss the relationship between LLSD and ABNE. It is clear that LLSD with a combinatorial Laplacian as input produces a solution to the corresponding instance of ABNE. It is also clear that LLSD can be used to solve ABNE if given the input of ABNE (i.e, the adjacency matrix), we can efficiently implement sparse access to a matrix such that an estimate of its low-lying spectral density allows us to recover an estimate of the low-lying spectral density of the combinatorial Laplacian. Interestingly, it turns out that we can do so if the input graph is clique-dense (i.e., in precisely the regime that is efficiently solvable by the quantum algorithm for Betti number estimation). Namely, we can efficiently implement sparse access to the $\binom{n}{k+1} \times \binom{n}{k+1}$-sized matrix Γ_k^G whose columns and rows are indexed by $(k+1)$-subsets of vertices, and whose entries are given by

$$\left(\Gamma_k^G\right)_{i,j} = \begin{cases} (\Delta_k^G)_{i,j} & \text{if } i \text{ and } j \text{ are } (k+1)\text{-cliques,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

In other words, the entries of the columns and rows that correspond to $(k+1)$-cliques are equal to the corresponding entries of the combinatorial Laplacian, and all other entries are equal to zero. After subtracting the extra nullity caused by adding the $\binom{n}{k+1} - \chi_k$ all-zeros columns and rows, and renormalizing the eigenvalue count by a factor $\binom{n}{k+1}/\chi_k$, the low-lying spectral density of this Γ_k^G is equal to the low-lying spectral density of the combinatorial Laplacian. In equation form, we have that

$$N_{\Delta_k^G}(0,b) = \frac{\binom{n}{k+1}}{\chi_k} N_{\Gamma_k^G}(0,b) - \frac{\binom{n}{k+1} - \chi_k}{\chi_k}. \tag{3.3}$$

From Eq. (3.3), it is clear that an estimate of $N_{\Gamma_k^G}(0,b)$ up to additive inverse polynomial precision allows us to obtain an estimate of $N_{\Delta_k^G}(0,b)$ up to additive inverse polynomial precision, assuming indeed that the graph is clique-dense – i.e., that $\chi_k/\binom{n}{k+1} \in \Omega\left(1/\text{poly}(n)\right)$. Note that this also requires us to have an estimate of $\chi_k/\binom{n}{k+1}$. Since the graph is clique-dense, it suffices to estimate $\chi_k/\binom{n}{k+1}$ up to additive inverse polynomial precision. An estimate of $\chi_k/\binom{n}{k+1}$ up to additive error ϵ can be obtained by drawing $\mathcal{O}(\epsilon^{-2})$ many k-subsets of vertices uniformly at random, and computing the fraction of these subsets that constitute an actual k-clique.

We emphasize that the above reduction works in precisely the regime where the quantum algorithm for Betti number estimation can efficiently solve ABNE. In other words, LLSD can be used to directly solve the problem that the quantum algorithm for Betti number estimation can efficiently solve. In this regard, LLSD is indeed a generalization of the problem that the quantum algorithm for Betti number estimation can efficiently solve.

Finally, let us discuss the reductions between ABNE and BNE. It is clear that BNE is reducible to ABNE if the size of the smallest nonzero eigenvalue of the combinatorial Laplacian is at least inverse polynomial in n. The reverse direction is unclear, as for BNE the threshold on the eigenvalues is fixed to be exactly zero. A possible approach would be to first project the eigenvalues that lie below the given threshold to zero and then count the zero eigenvalues. However, using techniques inspired by ideas from [85, 121], we have only been able to project these eigenvalues close to zero, as opposed to exactly equal to zero, and we are not aware of any way to circumvent this.

3.2 Classical intractability of LLSD

To show that quantum computers have an advantage over classical computers in topological data analysis, one would have to show that Betti number estimation requires exponential time on a classical computer. In this section we study the classical hardness of the problem efficiently solved by the quantum algorithm for Betti number estimation. In particular, we show that the natural generalization of this problem (which we called low-lying spectral density estimation) is classically intractable under widely-believed complexity-theoretic assumptions by showing that it is hard for the one clean qubit model of computation. Afterwards, we discuss how to potentially close the gap between the classical intractability of low-lying spectral density and (approximate) Betti number estimation in order to show that the topological data analysis problem is itself classically intractable.

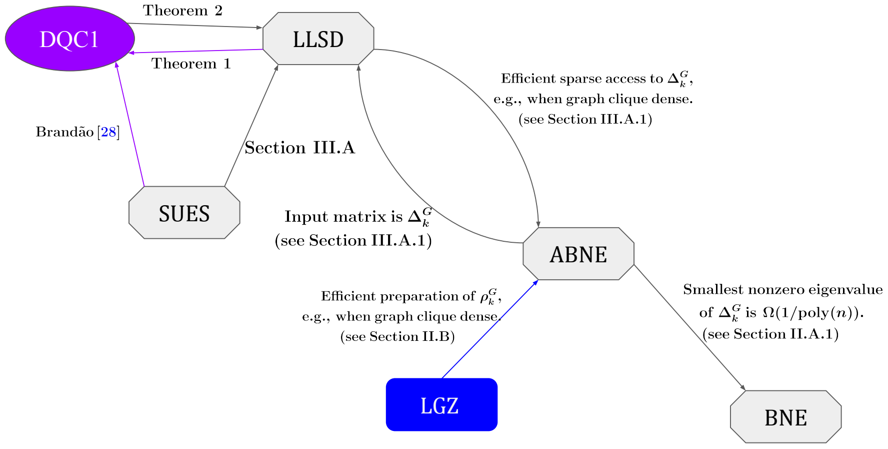
DQC1
Theorem 2
LLSD
Theorem 1
Brandão [28]
Section III.A
SUES
Efficient sparse access to Δ_k^G,
e.g., when graph clique dense.
(see Section III.A.1)
Input matrix is Δ_k^G
(see Section III.A.1)
ABNE
Efficient preparation of ρ_k^G,
e.g., when graph clique dense.
(see Section II.B)
Smallest nonzero eigenvalue
of Δ_k^G is $\Omega(1/\text{poly}(n))$.
(see Section II.A.1)
LGZ
BNE

Figure 3.1: Overview of the relations between the problems (octagons), algorithm (rectangle) and complexity class (ellipse) studied in this thesis. A $\xrightarrow{\text{C}}$ B stand for: "A can efficiently solve B if condition C holds". The algorithm studied is that by Lloyd, Garnerone and Zanardi (LGZ) as described in Figure 2.5. The problems are sparse uniform eigenvalue sampling (SUES), low-lying spectral density estimation (LLSD), approximate Betti number estimation (ABNE), and Betti number estimation (BNE) as defined in Section 3.1 The class DQC1 is defined in Section 3.2.1.

3.2.1 The one clean qubit model of computation

In the next section we will show that the complexity of the problems defined in Section 3.1 are closely related to the one clean qubit model of quantum computation [122]. In this model we are given a quantum register that is initialized in a state consisting of a single 'clean' qubit in the state $|0\rangle$, and $n-1$ qubits in the maximally mixed state. We can then apply any polynomially-sized quantum circuit to this register, and measure only the first qubit in the computational basis. Following [122], we will refer to the complexity class of problems that can be solved in polynomial time using this model of computation as $\mathsf{DQC1}$ – "deterministic quantum computation with a single clean qubit".

We will refer to a problem as $\mathsf{DQC1}$-hard if any problem in $\mathsf{DQC1}$ can be reduced to it under polynomial time truth-table reductions. That is, a problem L is $\mathsf{DQC1}$-hard if we can solve any problem in $\mathsf{DQC1}$ using polynomially many nonadaptive queries to an oracle for L, together with polynomial time preprocessing of the inputs and postprocessing of the outcomes. Technically, instead of containing the problem of estimating a given quantity up to additive inverse polynomial precision, $\mathsf{DQC1}$ contains the decision problem of deciding whether this quantity is greater than $1/2+\sigma$ or less than $1/2-\sigma$, where σ is some inverse polynomial gap. However, as the estimation versions of these problems are straightforwardly reduced to their decision version using binary search, we will bypass this point from now on and only consider the problems of estimating a given quantity up to inverse polynomial precision [178].

It is widely believed that the one clean qubit model of computation is more powerful than classical computation. For instance, estimating quantities that are supposedly hard to estimate classically, such as the normalized trace of a unitary matrix corresponding to a polynomial-depth quantum circuit and the evaluation of a Jones polynomial at a root of unity, turn out to be complete problems for $\mathsf{DQC1}$ [178]. Moreover, it has been shown that classical computers cannot efficiently sample from the output distribution of the one clean qubit model up to constant total variation distance error, provided that some complexity theoretic conjectures hold [141, 142].

3.2.2 Hardness of LLSD for the one clean qubit model

Recall that in order to show that quantum computers have an advantage over classical computers in topological data analysis, one would have to show that the problem that the quantum algorithm for Betti number estimation can efficiently solve is hard for classical computers. In Section 3.1, we pointed out that the problem that the quantum algorithm for Betti number estimation can efficiently solve is a restriction of ABNE to clique-dense graphs (i.e., graphs which satisfy Eq. (2.31)). Moreover, we showed in Section 3.1.1 that LLSD is a generalization of this version of ABNE. This motivates us to study the classical hardness of LLSD. In this section we present our results, which show that the complexity of LLSD is intimately related to the one clean qubit model.

Our first and main result is that LLSD is hard for the class $\mathsf{DQC1}$[5], even when the input is restricted to log-local Hamiltonians. As the one clean qubit model of

[5]Throughout this section we mean $\mathsf{DQC1}$-hardness with respect to *Turing reductions*. We believe that our approach could be modified to a Karp reduction, but since this reduction is not vital for our claim, we leave this question open for future work.

computation is widely believed to be more powerful than classical computation, this shows that LLSD is likely hard for classical computers. We discuss the implications of this result on the classical hardness of the problem that the quantum algorithm for Betti number estimation can efficiently solve in Section 3.2.3.

Theorem 5. LLSD *is* DQC1*-hard. Moreover,* LLSD *with the input restricted to log-local Hamiltonians remains* DQC1*-hard.*

We now give a sketch of our proof of the above theorem, the complete proof can be found in Appendix A.1. The main idea behind the proof is to show that we can use LLSD to estimate a quantity similar to a normalized subtrace – or more precisely, a normalized sum of eigenvalues below a given threshold – which has been shown to be DQC1-hard by Brandão [40]. We estimate this normalized subtrace by constructing a histogram approximation of the low-lying eigenvalues, and afterwards computing the mean of this histogram. To construct this histogram, we use LLSD to estimate the number of eigenvalues that lie in each bin. To avoid double counting of eigenvalues due to imprecisions around the thresholds of the bins, we subtract the output of LLSD with the eigenvalue threshold set to the lower-threshold of the bin from the output of LLSD with the eigenvalue threshold set to the upper-threshold of the bin. By doing so, we obtain an estimate of the number of eigenvalues within the bin, and misplace eigenvalues by at most one bin.

Our second result shows that the complexity of LLSD is more closely related to DQC1 than just hardness. Namely, we point out that if the input to LLSD is restricted to log-local Hamiltonians (or more generally, any type of Hamiltonian that allows for efficient Hamiltonian simulation using $\mathcal{O}(\log(n))$ ancilla qubits), then it can be solved using the one-clean qubit model. From this it follows that LLSD is DQC1-complete if the input is restricted to log-local Hamiltonians. The main idea behind why we can solve these instances of LLSD using the one clean qubit model is that the one-clean qubit model can simulate having access to up to $\mathcal{O}(\log(n))$ pure qubits [178]. These pure qubits allow for Hamiltonian simulation techniques based on the Trotter-Suzuki formula [129] and for quantum phase estimation up to the required precision. We summarize this in the following theorem, the proof of which can be found in Appendix A.2.

Theorem 6. LLSD *with the input restricted to log-local Hamiltonians is* DQC1*-complete.*

As an added result, we find that the complexity of SUES with the input restricted to log-local Hamiltonians is also closely related to DQC1. The complexity of this instance SUES was stated as an open problem by Wocjan and Zhang [200]. Moreover, we believe that it is interesting to study the complexity of SUES, as this problem can potentially find practical applications beyond both LLSD and Betti number estimation. We remark that SUES with the input restricted to log-local Hamiltonians was already shown to be DQC1-hard by Brandão [40]. Here we point out that the complexity of this instance of SUES is more closely related to the one clean qubit model than just hardness, as it can also be solved using $\mathsf{DQC1}_{\log n}$ circuits, that is, DQC1 circuits where we are allowed to measure logarithmically many of the qubits in the computational basis at the end (to read out the encoding of the eigenvalue). The proof of the following proposition can be found in Appendix A.2.

Proposition 7. SUES *with the input restricted to log-local Hamiltonians can be solved in polynomial time by the one clean qubit model with logarithmically many qubits measured at the end.*

3.2.3 Closing the gap for classical intractability of ABNE

The results discussed in the previous section are not sufficient to conclude that ABNE and BNE are hard for classical computers, because for these problems the family of input matrices is restricted to combinatorial Laplacians. Nonetheless, because LLSD is a generalization of the problem that the quantum algorithm for Betti number estimation can efficiently solve, our result shows that – aside from the matter regarding the restriction to combinatorial Laplacians – the quantum algorithm for Betti number estimation solves a classically intractable problem which in some cases captures interesting information concerning an underlying graph. Moreover, our result eliminates the possibility of certain routes for dequantization, namely those that are oblivious to the particular structure of the input matrix, which in particular eliminates the approaches of Tang et al. [60].

The open question regarding the classical hardness of ABNE and the problem that the quantum algorithm for Betti number estimation can efficiently solve is whether LLSD remains classically hard when restricted to combinatorial Laplacians of arbitrary or clique-dense graphs, respectively. Even though these restrictions on the input seem quite stringent, note that our result shows that LLSD is already DQC1-hard for the restricted family of log-local Hamiltonians obtained from Kitaev's circuit-to-Hamiltonian construction[6]. Moreover, there exists a family of combinatorial Laplacians that can encode DQC1-hard Hamiltonians, but not all of those are combinatorial Laplacians of clique complexes [48]. One way we tried to close this gap was by investigating whether we could encode Hamiltonians obtained from Kitaev's circuit-to-Hamiltonian construction into combinatorial Laplacians of sufficiently large graphs. While indeed various matrices related to quantum gates can be found as submatrices of combinatorial Laplacians, we did not succeed in finding an explicit embedding. In our view, this remains a promising way of showing that LLSD remains classically hard when restricted to combinatorial Laplacians (if indeed this claim is true at all).

Besides the above approach based on the Kitaev circuit-to-Hamiltonian construction, there are many other constructions that could potentially be used to show that LLSD remains classically hard when restricted to combinatorial Laplacians (again, if indeed this claim is true at all). In particular, there are several constructions used to prove QMA-hardness of the ground-state energy problem for certain families of Hamiltonians (i.e., deciding if the smallest eigenvalue lies above or below some thresholds)[7]. All of these constructions typically take as input a (verification) circuit and produce a Hamiltonian that has a small eigenvalue if and only if there exists a quantum state (also called a witness) that makes the circuit accept (i.e., if on this input it is more

[6]In [51, 40] and our case it is unclear whether this holds for k-local Hamiltonians with constant k, as the standard constructions of these local Hamiltonians involve a clock register that is too large.

[7]For an overview of circuit-to-Hamiltonian constructions see [39].

likely to output 1 on the first qubit). A special property of the Kitaev construction is that for every input to the circuit, there exists a state whose energy with respect to the corresponding Hamiltonian is close to the acceptance probability of the circuit (i.e., not just that there exists small eigenvalue if and only if there exists a state that makes the circuit accept). This property allowed Brandão to prove that normalized sub-trace estimation for these Hamiltonians is DQC1-hard [40], which is at the core of our proof of DQC1-hardness of LLSD. Hence, a promising approach to show DQC1-hardness of LLSD for a family of Hamiltonians is to look at existing circuit-to-Hamiltonian constructions used to prove QMA-hardness of versions of the ground-state energy problem and investigate whether they also have this special property that the Kitaev construction has (or to see if they can be equipped with it). This is particularly interesting for the constructions used to show QMA-hardness of the Bose-Hubbard model [61], or the Fermi-Hubbard model [149]. The reason for this is that both of these Hamiltonians exhibit similarities to the Hamiltonian of the hardcore fermion model, which is equal to the combinatorial Laplacian of a clique complex [48]. Specifically, the Hamiltonian H_G of the fermion hardcore model on a graph $G = ([n], E)$ is given by

$$H_G = \sum_{(i,j)\in E} P_i a_i a_j^\dagger P_j + \sum_{i\in V} P_i, \tag{3.4}$$

where $P_i = \prod_{(i,j)\in E}(I - n_j)$, a_i denotes the fermionic annihilation operator, and n_j denotes the fermionic number operator [48]. For this Hamiltonian H_G it holds that

$$H_G = \bigoplus_{k=0}^{n-1} \Delta_k^{\bar{G}}, \tag{3.5}$$

where $\bar{G}$ denotes the complement graph of G, and Δ_k^G denotes the k-th combinatorial Laplacian. Continuing along these lines, the authors of [67] established a circuit-to-Hamiltonian mapping onto combinatorial Laplacians that allowed them to show that deciding whether a Betti number is zero or not is QMA_1[8]-hard (though it has not yet lead to DQC1-hardness of ABNE).

Finally, instead of trying to show that the family of combinatorial Laplacians is sufficiently rich, we could also generalize this family of matrices while still remaining relevant to topological data analysis. For example, one could consider generalizations of combinatorial Laplacians, such as weighted combinatorial Laplacians [105] or persistent combinatorial Laplacians [195], and show that these generalized families are sufficiently rich as to contain DQC1-hard instances. Besides all the approaches discussed above, other routes such as proving classical hardness of LLSD when restricted to other sets of matrices such as $\{0, \pm 1\}$-matrices, or by going through the discrete structures related to Tutte and Jones polynomials [16, 178] could all be possible as well.

The open questions regarding the classical hardness of BNE are the same as those regarding the classical hardness of ABNE, except that there is one additional open

[8]QMA_1 is the one-sided error version of QMA.

question. Namely, assuming that ABNE is classically hard, the remaining open question regarding the classical hardness of BNE is whether estimating the number of eigenvalues exactly equal to zero is at least as hard as estimating the number of eigenvalues below a given inverse polynomially small threshold. This question was already addressed in Section 3.1.1 when we examined the reductions between ABNE and BNE. As discussed there, one approach would be to project the eigenvalues below the given threshold to zero, and afterwards count only the zero eigenvalues.

Regardless, even if LLSD does not remain classically hard when restricted to combinatorial Laplacians, we can envision practical generalizations of the quantum-algorithmic methods used by the algorithm for Betti number estimation that go beyond Betti numbers, as we will discuss in more detail in Section 3.3. Specifically, in Section 3.3 we provide efficient quantum algorithms for two concrete examples of such practical generalizations, together with complexity-theoretic evidence of their classical hardness. The first example we discuss is numerical rank estimation, an important problem in machine learning, data analysis and many other applications. The second example is spectral entropy estimation, which can be used as a tool in complex network analysis.

3.2.4 Graphs with quantum speedup

In Section 2.2.2, we outlined criteria that the graph has to satisfy in order for the quantum algorithm to be able to efficiently estimate (approximate) Betti numbers. Specifically, the graph has to be such that one can efficiently prepare the input state in Eq. (2.30), e.g., by sampling uniformly at random from cliques of a given size. Afterwards, in Section 2.2.3, we discussed the best known classical algorithms and we outlined the regimes in which they require superpolynomial runtimes. In this section we put these two considerations together and we concretely characterize families of graphs for which the quantum algorithm achieves either a high-degree polynomial, or even a superpolynomial speedup over the best known classical algorithm. In particular, we identify families of graphs for which the quantum algorithm is efficient and for which the best known classical algorithms are unable to achieve competitive runtimes.

As discussed in Section 2.2.2, one way to efficiently prepare the input state is to use Grover's algorithm or rejection sampling to sample uniformly at random from cliques of a given size. Recall that for this to be efficient the graph has to be clique dense, i.e., it has to satisfy Eq. (2.31). To identify a family a clique-dense graphs, let us consider clique sizes $k \geq 3$, let $\gamma > \frac{k-2}{2(k-1)}$ be a constant, and consider any graph on n vertices with at least γn^2 edges. Suppose we want to estimate the k-th approximate Betti number of this graph, where k and the precision parameters are constant. The quantum algorithm for Betti number estimation can do so in time

$$\widetilde{\mathcal{O}}\left(\sqrt{n^{k+1}/\chi_k} + n^3\right),$$

where χ_k denotes the number of $(k+1)$-cliques. Having chosen the graph the way we did, the clique density theorem [162] now directly guarantees that our graph satisfies

$$\chi_k \in \Omega(n^{k+1}),$$

which is a phenomenon known as "supersaturation". In particular, this implies that our graph is clique-dense and that the quantum algorithm for Betti number estimation estimates the required approximate Betti number in time

$$\widetilde{\mathcal{O}}\left(n^3\right).$$

Moreover, as discussed in Section 2.2.3, the best known classical algorithm requires time

$$\mathcal{O}\left(n^{k+1}\right),$$

as the number of nonzero entries of the corresponding combinatorial Laplacian is at least χ_k. We conclude that in these instances the quantum algorithm for Betti number estimation achieves a $(k-2)$-degree polynomial speedup over the best known classical methods, which for large enough k might allow for runtime advantages on prospective fault-tolerant computers, even when all overheads are accounted for [27].

We can push the above separation between the best known classical algorithm and the quantum algorithm even further. Consider the same setting as above, but with $\gamma = \frac{k-1}{k}$ and we allow k to scale with n. Using a result of Moon and Moser [140, 132, 191], we can derive that in this setting the graph satisfies

$$\binom{n}{k+1}/\chi_k \in \mathcal{O}\left(k^k\right).$$

Therefore, the quantum algorithm can estimate the k-th approximate Betti number in time

$$\mathcal{O}\left(k^{2+k/2} + n^3\right).$$

On the other hand, the best known classical algorithm runs in time

$$\mathcal{O}\left(n^{k+1}/k^{2k}\right),$$

as the number of nonzero entries of the corresponding combinatorial Laplacian is at least $\chi_k \geq n^{k+1}/k^{2k}$. In particular, if we let k scale with n in an appropriate way, then the quantum algorithm achieves a superpolynomial speedup over the best known classical method. For example, if we let the clique size scale as $k \sim \log n$, then the quantum algorithm runs in time

$$2^{\mathcal{O}(\log n \log\log n)},$$

whereas the best known classical algorithm runs in time

$$2^{\mathcal{O}\left((\log n)^2\right)},$$

giving rise to a superpolynomial quantum speedup. Note that the graphs in the previous two settings are rather edge-dense (which occurs in topological data analysis if the grouping-scale ϵ approaches the maximum distance between two datapoints), and it is unknown whether better classical algorithms are possible in this regime.

Next, we construct a family of graphs where (*i*) the Betti numbers are large, (*ii*) the clique-density is high, and (*iii*) the spectral gaps the combinatorial Laplacian are sufficiently large. Due to properties (*i*)-(*iii*) this family of graphs provides a great example of a family of graphs on which the quantum algorithm outperforms its classical counterpart. Let $K(m, k)$ be the k-partite complete graph, where each partition contains m vertices. That is, $K(m, k)$ consists of k clusters, each with m vertices; there are no edges within clusters, but all edges between clusters are included. Note $K(m, 1)$ is a collection of m points with no edges. $K(m, k)$ gives a useful example of a clique complex with a high Betti number [13]. It also has a Laplacian with a large spectral gap.

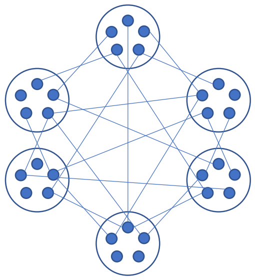

Figure 3.2: The graph $K(5, 6)$.

Proposition 8. *The $(k-1)^{th}$ Betti number of the clique complex of $K(m, k)$ is*

$$\beta_{k-1} = (m-1)^k. \tag{3.6}$$

Proposition 9. *The combinatorial Laplacian $\Delta^G_{k-1} = (\partial^G_{k-1})^\dagger \partial^G_{k-1} + \partial^G_k (\partial^G_k)^\dagger$ of the clique complex of $K(m, k)$ has spectral gap*

$$\lambda_{\min} = m. \tag{3.7}$$

We prove these in A.3 using techniques from simplicial homology. A further useful fact is that

$$|\mathrm{Cl}_k(K(m, k))| = m^k. \tag{3.8}$$

Standard classical approaches need to at least store a vector of this length, so we can give a classical complexity T_c of estimating normalized Betti numbers

$$T_c \sim e^{k \ln m}. \tag{3.9}$$

As a first approximation for the quantum cost T_q, we use the formula

$$T(G,k,r,\delta) = 3\pi|E|\frac{\ln(1/\delta)}{r}\sqrt{\frac{\binom{n}{k}}{\beta_{k-1}^G}}. \tag{3.10}$$

and consider just the square root factor and $|E|$. Stirling's approximation gives $\binom{n}{k} \sim \left(\frac{m^{1+1/m}}{m-1}\right)^n$, and Proposition 8 gives $\beta_{k-1} = (m-1)^{n/m}$, giving a quantum complexity scaling as

$$T_q \sim |E|\left(\frac{m^{1+1/m}}{(m-1)^{1+1/m}}\right)^{n/2} \sim n^2 e^{(k/2)(1+1/m)}. \tag{3.11}$$

Therefore, for constant m, there is a polynomial speedup by a $2\ln m$ root (ignoring n^2 and the $1/m$ term). Alternatively, taking k constant, the above formulae give

$$T_c = \mathcal{O}(n^k), \tag{3.12}$$
$$T_q = \mathcal{O}(n^2). \tag{3.13}$$

Then there is a polynomial speedup by a $k/2$ root. To obtain a superpolynomial speedup, m can be taken to increase close to linear in n, but k can be taken to also increase with n. Close to the best result is obtained for $k = c\ln^2 n$ with some constant c. Then the logs of the complexities are approximately

$$\ln T_c \sim c\ln^3 n, \tag{3.14}$$
$$\ln T_q \sim 2\ln n + (c/2)\ln^2 n. \tag{3.15}$$

That implies a speedup by a $2\ln n$ root, which is superpolynomial.

This is still not an exponential speedup, but as far as the graph is concerned this is the best speedup that could be obtained from this type of approach. This is because, with k constant, the quantum complexity ignoring the $|E|$ factor is $\mathcal{O}(1)$. The Betti number is already scaling the same as $\binom{n}{k}$, but the overhead from $|E|$ means that the speedup is not exponential.

As also discussed in Section 2.2.2, besides clique-density another important graph parameter that dictates the runtimes of specialized algorithms for uniform clique sampling is the so-called *arboricity.* The arboricity of a graph is equivalent (up to a factor 1/2) to the maximum average degree of a subgraph. For a graph with n vertices and arboricity α, near-optimal classical algorithms sample a k-clique uniformly at random in time [77]

$$\widetilde{\mathcal{O}}\left(k^k \cdot \max\left\{\left(\frac{(n\alpha)^{k/2}}{\chi_k}\right)^{\frac{1}{k-1}},\ \min\left\{n\alpha, \frac{n\alpha^{k-1}}{\chi_k}\right\}\right\}\right). \tag{3.16}$$

By also considering the algorithm of [77] (i.e., instead of rejection sampling or Grover's algorithm) we strictly expand the family of graphs for which the quantum algorithm achieves a superpolynomial speedup for ABNE. In particular, there exists a family of graphs for which the algorithm of [77] is superpolynomially more efficient[9] than Grover's algorithm and rejection sampling for the problem of uniform clique sampling. An example of such a family is as follows: consider the n-vertex graphs consisting of n/r cliques of size r (for simplicity we assume that n is a multiple of r), where each r-clique is fully-connected with d other r-cliques (i.e., all edges between the $2r$ vertices are present). In other words, consider a d-regular graph on n/r vertices, and replace each vertex with an r-clique and fully-connect all r-cliques that were connected according to the d-regular graph we started with. Now if we set $d, r = \log n$ and $k = \log\log n$, then the number of k-cliques (and thus also the runtime of the best known classical algorithm for ABNE) scales like $\log(n)^{\log\log(n)}$. Moreover, the clique-density (and thus also the runtime of rejection sampling and Grover's algorithm) scales like $n^{\log\log(n)}$. Finally, the runtime of the algorithm of [77] scales like $\log\log(n)^{\log\log(n)}$. In conclusion, for these graphs the algorithm of [77] is superpolynomially more efficient than rejection sampling and Grover's algorithm for the problem of uniform clique sampling. Moreover, for these graphs the quantum algorithm for ABNE achieves a superpolynomial speedup over the best-known classical algorithm for ABNE, but only if one uses the algorithm of [77] (i.e., this speedup goes away if one uses rejection sampling or Grover's algorithm). We again remark that we are dealing with special types of graphs, and it is unknown whether better classical algorithms are possible in this regime.

3.3 Quantum speedups beyond Betti numbers

In the previous section we provided evidence that the computational problems tackled by the quantum algorithm for Betti number estimation are likely hard for classical computers. Even though we fell short of showing that the topological data analysis problem of estimating (approximate) Betti number is classically intractable, we did provide evidence that the quantum algorithmic methods that underlie the quantum algorithm for Betti number estimation could give rise to a potential source of practical quantum advantage. In this section we demonstrate this by discussing extensions of the quantum-algorithmic methods behind the algorithm for Betti number estimation that go beyond Betti numbers. In particular, we provide efficient quantum algorithms for numerical rank estimation (an important problem in machine learning and data analysis) and spectral entropy estimation (which can be used to compare complex networks), together with complexity-theoretic evidence of their classical hardness.

3.3.1 Numerical rank estimation

In this section we identify a practically important application of the problem of estimating the number of small eigenvalues (which we called LLSD). Specifically, we consider the problem of *numerical rank estimation*. The numerical rank of a matrix $H \in \mathbb{C}^{2^n \times 2^n}$ is the number of eigenvalues that lie above some given threshold b, i.e.,

[9]We say that a runtime $t_1(n)$ is *superpolynomially more efficient* than a runtime $t_2(n)$ if $\log t_2(n)/\log t_1(n) \to \infty$ when $n \to \infty$.

it is defined as

$$r_H(b) = \frac{1}{2^n} \sum_{k\,:\,\lambda_k > b} 1,$$

where $\lambda_1 \leq \cdots \leq \lambda_{2^n-1}$ denote the eigenvalues of H. By the rank-nullity theorem we have that

$$r_H(b) = 1 - N_H(0, b),$$

which shows that we can estimate the numerical rank using low-lying spectral density estimation and that the error scaling is the same.

Many machine learning and data analysis applications deal with high-dimensional matrices whose relevant information lies in a low-dimensional subspace. To be specific, it is a standard assumption that the input matrix is the result of adding small perturbations (e.g., noise in the data) to a low-rank matrix. This small perturbation turns the input matrix into a high-rank matrix, that can be well approximated by a low-rank matrix. Techniques such as principle component analysis [111] and randomized low-rank approximations [99] are able exploit this property of the input matrix. However, these techniques often require as input the dimension of this low-dimensional subspace, which is often unknown. This is where numerical rank estimation comes in, as it can estimate the dimension of the relevant subspace by estimating the number of eigenvalues that lie above the "noise-threshold". In addition, being able to determine whether the numerical rank of a matrix is large or small enables one to assert whether the above low-rank approximation techniques is applicable at all, or not.

From Theorem 5 it directly follows that quantum computers achieve an exponential speedup over classical computers for numerical rank estimation of matrices specified via sparse access (unless the one clean qubit model can be efficiently simulated on a classical computer). Still, it is also interesting to consider settings where the matrix is specified via a different input model. In the remainder of this section we study two examples of different input models. Firstly, motivated by a more practical perspective we consider a seemingly weaker input model that is more closely related to the input models that appear in classical data analysis settings. Secondly, we consider a likely stronger input model that appears throughout quantum machine learning literature, which is more informative from a complexity-theoretic perspective.

In typical (classical) applications, matrices are generally not specified via sparse access. Here we consider an input model that is more closely related to what is encountered in a typical classical setting. Specifically, we consider the case where a sparse matrix A of size $2^n \times 2^n$ is specified as a list of triples

$$\big\{(i_k, j_k, A_{i_k,j_k}) \mid A_{i_k,j_k} \neq 0\big\},$$

which is sorted lexicographically by column and then row. Storing matrices in this type of memory structure is very natural when dealing with matrices with a limited number of nonzero entries (which we denote by nnz). Now, for the quantum analogue we consider the same specification but we suppose that it is stored in a QRAM-type memory, only additionally allowing us to query it in superposition as follows:

$$\sum_k \alpha_k |k\rangle |0\rangle \mapsto \sum_k \alpha_k |k\rangle |i_k, j_k, A_{i_k,j_k}\rangle .$$

Since the list is sorted, and since A is sparse, we can still simulate column-wise sparse access in $\mathcal{O}(\log \mathsf{nnz})$ queries, essentially by using binary search. Therefore, if A is Hermitian, then the quantum algorithm can estimate its numerical rank in time $\mathcal{O}(\mathrm{poly}(n, \log \mathsf{nnz}))$. On the other hand, the best known classical algorithms run in time $\mathcal{O}(\mathsf{nnz})$ [190, 59, 70, 124]. Consequently, the quantum algorithm achieves a speedup over the best known classical algorithm if nnz is at least a high-enough degree polynomial in n (and it achieves an exponential speedup if nnz is itself exponential). For the case where A is not Hermitian, recall that we also need sparse access to $A^\dagger$. For this issue we found no general method that can do so in time less than $\mathcal{O}(\mathsf{nnz})$, without assuming a high sparsity. However, the high sparsity then exactly offsets any potential quantum advantage in the full algorithm complexity.

Next, we consider a likely stronger input model which is widely-studied in the quantum machine learning literature. Specifically, we study the quantum-accessible data structure introduced in [118, 119], which can generate quantum states proportional to the columns of the input matrix, together with a quantum state whose amplitudes are proportional to the 2-norms of the columns. When the input matrix is provided in this quantum-accessible data structure, the quantum-algorithmic methods of [85, 55] can be used to estimate its numerical rank in time $\mathcal{O}(\mathrm{poly}(A_{\max}, n))$, where $A_{\max} = \max_{i,j} |A_{ij}|$.

The classical analogue of this quantum-accessible data structure is the sampling and query access model introduced in [186], which brought forth the "dequantization" methods discussed in [60]. At present it is not clear whether assuming sampling and query access allows us to efficiently estimate the numerical rank using dequantizations, or other methods. Here both possibilities are interesting. Firstly, if numerical rank estimation remains equally hard with sampling and query access, then it shows that quantum algorithms relying on the methods of LGZ have a chance of maintaining their exponential advantage in more general scenarios. Secondly, if an efficient classical algorithm for numerical rank estimation is possible with sampling and query access, then this leads to new insights regarding the hardness of the one clean qubit model. Recall that we have shown that estimating the numerical rank of matrices specified via sparse access is $\mathsf{DQC1}$-hard (in the sense that, if a classical algorithm could do so efficiently given analogous access, then it can be used to efficiently solve all problems in $\mathsf{DQC1}$). Now for the sparse matrix case, the only difference between sparse access and sampling and query access is that the latter allows one to sample from a distribution whose probabilities are proportional to the 2-norms of the columns. Indeed, the other part (i.e., sampling from distributions whose probabilities are proportional to the squared entries of the columns) is straightforward when the matrix is specified via sparse access. This implies that, if sampling and query access allows us to efficiently estimate the numerical rank of sparse matrices, then producing samples according to the 2-norms of the columns of a sparse matrix is $\mathsf{DQC1}$-hard. This also holds for the log-local Hamiltonian setting, so it would also follow that sampling from a distribution proportional to the 2-norms of the columns of log-local Hamiltonians is $\mathsf{DQC1}$-hard. We summarize this observation in the proposition below.

Proposition 10. *Suppose there exists an efficient classical algorithm for numerical rank estimation (or, equivalently* LLSD*) for matrices provided by sampling and query access. Then, sampling from a distribution whose probabilities are proportional to the 2-norms of the columns of a sparse Hermitian matrix is* $\mathsf{DQC1}$*-hard (with respect to Turing reductions).*

3.3.2 Combinatorial Laplacians beyond Betti numbers

In the previous section we discussed a practical application of the quantum-algorithmic methods behind the algorithm for Betti number estimation by using the same methods, but changing the family of input matrices (i.e., going beyond combinatorial Laplacians). In this section we take a different approach, namely we again consider the combinatorial Laplacians, but investigate applications beyond Betti number estimation (i.e., beyond estimating its nullity) relying on different algorithms than the one for low-lying spectral density estimation. Moreover, we will again find regimes where the same type of evidence of classical hardness can be provided, further motivating investigations into quantum algorithms that operate on the combinatorial Laplacians.

The eigenvalues and eigenvectors of the combinatorial Laplacian have many interesting graph-oriented applications beyond the applications in topological data analysis discussed in Section 2.2. The intuition behind this is that the combinatorial Laplacian can be viewed as a generalization of the standard graph Laplacian. For example, there exist generalizations of spectral clustering and label propagation (important techniques in machine learning that are used for dimensionality reduction and classification) which utilize the eigenvalues and eigenvectors of the combinatorial Laplacians [152]. Moreover, the eigenvalues of a normalized version of the combinatorial Laplacian convey information about the existence of circuits of cliques (i.e., ordered lists of adjacent cliques that cover the whole graph) and about the chromatic number [105]. Lastly, Kirchhoff's matrix tree theorem – which relates the eigenvalues of the standard graph Laplacian to the number of spanning trees – turns out to have a generalization to higher-order combinatorial Laplacians [75].

The specific problem that we study in this section is that of sampling from a distribution over the eigenvalues whose probabilities are proportional to the magnitude of the eigenvalues. In particular, we give a quantum algorithm that efficiently samples from an approximation of these distributions. Moreover, we show that sampling from these distributions for arbitrary sparse Hermitian matrices is again as hard as simulating the one clean qubit model, which shows that it is classically intractable (unless the one clean qubit model can be efficiently simulated on a classical computer). Finally, we discuss how this quantum algorithm can speed up spectral entropy estimation, which when applied to combinatorial Laplacians can be used to compare complex networks.

We define the problem that we study in this section as follows.

Sparse weighted eigenvalue sampling (SWES)
Input:
1) A sparse positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$, with $||H|| \leq 1$ and $\mathrm{tr}\{H\}/2^n \in \mathcal{O}\left(\mathrm{poly}(n)\right)$.
2) An estimation precision $\delta \in \Omega\left(1/\mathrm{poly}(n)\right)$.
3) A sampling error probability $\mu \in \Omega\left(1/\mathrm{poly}(n)\right)$.

Output: A sample drawn from a (δ, μ)-approximation of the distribution $p(\lambda_j) = \lambda_j / \mathrm{tr}\{H\}$.

Using the subroutines of the quantum algorithm for Betti number estimation (i.e., Hamiltonian simulation and quantum phase estimation), we can efficiently sample from an approximation of the distribution of SWES defined above. In fact, we can efficiently implement *purified quantum query-access* to $p(\lambda_j)$ [84]. To be precise, we can implement an approximation of the unitary U_H (and its inverse) which acts as

$$U_H \left|0\right\rangle_A \left|0\right\rangle_B = \left|\psi_H\right\rangle = \sum_{j=0}^{2^n-1} \sqrt{p\left(\lambda_j\right)} \left|\psi_j\right\rangle_A \left|\phi_j\right\rangle_B , \tag{3.17}$$

such that $\mathrm{Tr}_B\left(\left|\psi_H\right\rangle\left\langle\psi_H\right|\right) = H/\mathrm{tr}\{H\}$. Purified quantum query-access has been shown to be more powerful than standard classical sampling access, as it can speedup the postprocessing of the samples when trying to find out properties of the underlying distribution [84].

We implement an approximation of the purified quantum-query access defined in Eq. (3.17) as follows:

1. Prepare the following input state by taking a maximally entangled state (which can always be expressed in the eigenbasis of H in one of its subsystems) and adding two ancillary registers

$$\left|\psi\right\rangle_{in} = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|0^t\right\rangle \otimes \left|0\right\rangle_{flag} ,$$

where $\{\left|\psi_k\right\rangle\}_{k=0}^{2^n-1}$ are orthonormal eigenvectors of H and $\{\left|\phi_k\right\rangle\}_{k=0}^{2^n-1}$ is an orthonormal basis of $\mathbb{C}^{2^n}$.

2. Use Hamiltonian simulation on H, and apply quantum phase estimation of the realized unitary to the first register to prepare the state

$$\begin{aligned}\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^t} \alpha_{k,j} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_{k,j}}\right\rangle \otimes \left|0\right\rangle_{flag} \\ \approx \frac{1}{\sqrt{N}} \sum_{k=0}^{2^n-1} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_k}\right\rangle \otimes \left|0\right\rangle_{flag} ,\end{aligned}$$

where the $\widetilde{\lambda_{k,j}}$ are t-bit strings, $|\alpha_{k,j}|^2$ is close to 1 if and only if $\lambda_k \approx \widetilde{\lambda_{k,j}}$, and $\widetilde{\lambda_k}$ denotes the best t-bit approximation of λ_k.

3. Use controlled rotations to "imprint" the t-bit approximations of the eigenvalues into the amplitudes of the flag-register to prepare the state

$$\begin{aligned}
&\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}\sum_{j=0}^{2^t}\alpha_{k,j}\,|\psi_k\rangle\,|\phi_k\rangle\otimes|\widetilde{\lambda_{k,j}}\rangle\\
&\qquad\otimes\left(\sqrt{\widetilde{\lambda_{k,j}}}\,|0\rangle_{flag}+\sqrt{1-\widetilde{\lambda_{k,j}}}\,|1\rangle_{flag}\right)\\
&\approx\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|\psi_k\rangle\,|\phi_k\rangle\otimes|\widetilde{\lambda_k}\rangle\\
&\qquad\otimes\left(\sqrt{\widetilde{\lambda_k}}\,|0\rangle_{flag}+\sqrt{1-\widetilde{\lambda_k}}\,|1\rangle_{flag}\right).
\end{aligned}$$

4. Use fixed point amplitude amplification to amplify states whose flag-register is in the state $|0\rangle$ to prepare an approximation of the state

$$\begin{aligned}
&\frac{1}{\sqrt{\mathrm{Tr}(H)}}\sum_{k=0}^{2^n-1}\sum_{j=0}^{2^t}\alpha_{k,j}\sqrt{\widetilde{\lambda_{k,j}}}\,|\psi_k\rangle\,|\phi_k\rangle\otimes|\widetilde{\lambda_{k,j}}\rangle\otimes|0\rangle_{flag}\\
&\approx\frac{1}{\sqrt{\mathrm{Tr}(H)}}\sum_{k=0}^{2^n-1}\sqrt{\widetilde{\lambda_k}}\,|\psi_k\rangle\,|\phi_k\rangle\otimes|\widetilde{\lambda_k}\rangle\otimes|0\rangle_{flag}.
\end{aligned}$$

5. Finally, uncompute and discard the eigenvalue- and flag-register to prepare the state

$$\begin{aligned}
|\psi_H\rangle&=\frac{1}{\sqrt{\mathrm{Tr}(H)}}\sum_{k=0}^{2^n-1}\sum_{j=0}^{2^t}\alpha_{k,j}\sqrt{\widetilde{\lambda_{k,j}}}\,|\psi_k\rangle\,|\phi_k\rangle\\
&\approx\frac{1}{\sqrt{\mathrm{Tr}(H)}}\sum_{k=0}^{2^n-1}\sqrt{\widetilde{\lambda_k}}\,|\psi_k\rangle\,|\phi_k\rangle.
\end{aligned}$$

Looking at the cost of the above algorithm, we note that Steps 2 and 3 can be implemented up to polynomial precision in time $\mathcal{O}(\mathrm{poly}(n))$. Also, note that Step 4 can be implemented up to polynomial precision in time $\mathcal{O}\left(\sqrt{2^n/\,\mathrm{tr}\{H\}}\right)$, which brings the total runtime to

$$\mathcal{O}\left(\mathrm{poly}(n)+\sqrt{2^n/\,\mathrm{tr}\{H\}}\right).$$

Besides being able to efficiently sample from an approximation of SWES on a quantum computer, we show that SWES requires superpolynomial time on a classical computer (unless the one clean qubit model can be efficiently simulated on a classical computer). To be precise, we show that sampling from SWES allows us to efficiently estimate the normalized subtrace discussed in Section 3.2.2, which is known to be

DQC1-hard [40]. We gather this in the following theorem, the proof of which can be found in the Supplementary Material.

Theorem 11. SWES *is* DQC1*-hard. Moreover,* SWES *with the input restricted to log-local Hamiltonians remains* DQC1*-hard.*

The above theorem motivates us to look for practical applications of SWES, or more specifically, of the purified quantum query-access described in Eq. (3.17). We end this section by discussing such an application called spectral entropy estimation, which when applied to combinatorial Laplacians can be used to compare complex networks. The classical hardness of SWES opens up another road towards practical quantum advantage, as it could be that combinatorial Laplacians arising in complex network analysis form a rich enough family for which SWES remains classically hard when restricted to them.

Spectral entropy estimation of the combinatorial Laplacian

Recently, several quantum information-inspired entropic measures for complex network analysis have been proposed [34, 68]. One example of these are spectral entropies of the combinatorial Laplacian, which measure the degree of overlapping of cliques within the given complex network [155, 179, 134]. Specifically, it has been shown that these entropic measures can be used to measure network centralization (i.e., how central is the most central node in relation to all other nodes) [179], network regularity (i.e., the difference in degrees among nodes) [155], and clique connectivity (i.e., the overlaps between communities in the network) [134].

If $\lambda_0, \dots, \lambda_{d_k^G - 1}$ denote the eigenvalues of a combinatorial Laplacian Δ_k^G (i.e., $d_k^G = \dim \mathcal{H}_k^G$), then its *spectral entropy* is defined by

$$S(\Delta_k^G) = -\sum_{j=0}^{d_k^G - 1} p(\lambda_j) \log(p(\lambda_j)), \tag{3.18}$$

where we define $p(\lambda_j) = \lambda_j / (\sum_k \lambda_k)$. This spectral entropy coincides with the von Neumann entropy of $\Delta_k^G / \operatorname{tr}\{\Delta_k^G\}$. Equivalently, it coincides with the Shannon entropy of the distribution $p(\lambda_j)$. Another entropy that is used in complex network analysis is the *α-Renyi spectral entropy*, which is given by

$$S_\alpha(\Delta_k^G) = \frac{1}{1-\alpha} \log \left(\sum_{j=0}^{d_k - 1} p(\lambda_j)^\alpha \right), \tag{3.19}$$

where $\alpha \geq 0$ and $\alpha \neq 1$. The limit for $\alpha \to 1$ is the spectral entropy as defined in Eq. (3.18).

To estimate the spectral entropy defined in Eq. (3.18), one can use techniques from [11, 192] to classically postprocess samples from $p(\lambda_j)$ that one obtains from the quantum algorithm for SWES described in the previous section. However, since we can implement purified quantum query-access using the algorithm described in the previous section, the postprocessing can be sped up quadratically using quantum methods [84]. This idea of speeding up the postprocessing of samples using quantum methods also holds for the α-Renyi entropy defined in Eq. (3.19), where one can either classically postprocess the samples [12], or use faster quantum methods [181].

Because we have shown that sampling from SWES is DQC1-hard, the above approach to spectral entropy estimation can not be done efficiently on a classical computer – i.e., it cannot be dequantized – when generalized to arbitrary sparse matrices (unless the one clean qubit model can be efficiently simulated on a classical computer). Moreover, as the α-Renyi entropy is the logarithm of the Schatten p-norm, and it is known that estimating Schatten p-norms is DQC1-hard [51], we find that computing α-Renyi entropy is classically intractable (again, unless the one clean qubit model can be efficiently simulated on a classical computer).

3.4 Possibilities and challenges for implementations

As near-term quantum devices are still limited, it is crucial to make sure to use them to their fullest extent when implementing a quantum algorithm. Near-term devices are limited in size, gates are error prone, qubits decohere, and their architectures are limited [160]. We are therefore interested in algorithms that require few gates (to minimize the effect of decoherence and gate errors), that are not too demanding regarding architecture, while achieving advantages with few qubits and being tolerant to noise (which will inevitably be present in the system regardless of the depth and gate count). The quantum algorithms we consider use Hamiltonian simulation and quantum phase estimation. Fortunately, both resource optimization [30] and error-mitigation [187, 38, 78, 136, 147] for these routines are important topics for the broadly investigated field of quantum algorithms for quantum chemistry and many-body physics, and any progress achieved for those purposes can be readily applied. Moreover, recent work has focused on reducing the depth of the quantum circuit required to implement the algorithm for (approximate) Betti number estimation [189]. In this section we will focus on the issues of size and noise. First, we investigate the required number of qubits and we propose methods on how to reduce this. Based on these methods, we provide an estimate of the number of qubits required to challenge classical methods. Finally, we discuss issues regarding robustness of the algorithm to noise in the quantum hardware.

To analyze the number of qubits required to implement Hamiltonian simulation of a $2^n \times 2^n$-sized input matrix, we consider two possible scenarios: the input matrix is either given to us as local terms, or it is specified via sparse access. If the input matrix is given to us as local terms, then we can implement Hamiltonian simulation based on the Trotter-Suzuki formula [129]. As this Hamiltonian simulation technique does not require ancillary qubits (assuming the available gate set can implement each of the Trotter steps without ancillary qubits) [51], we can implement it using only n qubits. On the other hand, if the input matrix is specified via sparse access, then we have to use more intricate Hamiltonian simulation techniques (e.g., based on quantum signal processing [133]). The downside of these methods is that they require an ancillary register to 'load' the queries to the sparse-access oracles onto. By having to add this ancillary register, the total number of qubits required to implement these Hamiltonian simulation techniques becomes $2n+r+1$, where r is the number of bits used to specify the entries of the input matrix. In other words, sparse-access oracles more than double the required number of qubits.

When possible it is therefore advantageous to avoid using sparse access when having first proof-of-principle demonstrations of quantum advantage in mind. One way of doing so is to add an extra precompilation step that finds a suitable decomposition of the input matrix. In particular, one can trade-off the required number of ancilla qubits for some amount of precompilation and some extra depth of the precompiled circuit, in the following two ways. First, one could decompose the input matrix in terms of a linear combination of unitaries, and use related techniques for Hamiltonian simulation of such input matrices [32]. This brings the required number of qubits down from $2n + r + 1$ to $n + \log(m)$, where m is the number of terms in the linear combination of unitaries. Secondly, one could decompose the input matrix in terms of a sum of local Hamiltonians and use Hamiltonian simulation based on the Trotter-Suzuki formula. This brings the required number of qubits down from $2n+r+1$ to n. Thus, both approaches can halve the number of required qubits, however, one has to be careful as finding such decompositions may constitute a dominating overhead.

In case of Betti number estimation, we note that such precompilation is in fact feasible and meaningful. This is due to the fact that in this case there is a direct way to decompose input matrix (i.e., the combinatorial Laplacian) as a sum of Pauli-strings in order to implement Hamiltonian simulation based on the Trotter-Suzuki formula. Specifically, due to the close relationship between combinatorial Laplacians and Hamiltonians of the fermion hardcore model (as described in Section 3.2.3) [48] we can decompose the combinatorial Laplacian into a sum of Pauli-strings by applying a fermion to qubit mapping such as the Jordan-Wigner or Bravyi-Kitaev transformations to Eq (3.4). Note however that this does not guarantee that Hamiltonian simulation based on the Trotter-Suzuki formula will be efficient as the decomposition might require exponentially many terms and the locality of the individual terms could be large. As can be seen in Eq. (3.4), the number of terms in the decomposition scales with the degree of the vertices in the complement of the graph. In particular, if the graph is such that any vertex is connected to all other vertices except for a constant number of them, then the number of terms in the decomposition scales polynomially. As discussed in Section 3.2.4, these are exactly the type of graphs where the quantum algorithm for Betti number estimation achieves a speedup over the best known classical algorithms, since these types of graphs are clique-dense (i.e., they satisfy Eq. (2.31)). The locality of the Pauli-strings in the decomposition can however not be guaranteed to be small, but this fortunately has less effect on the depth of the circuit. Finally, we remark that this decomposition also gives rise to a technique that allows one to control the depth of the circuit required for the Hamiltonian simulation. Namely, by dropping certain terms from the decomposition (e.g., terms with a small coefficient) one could reduce the depth of the circuit required for Hamiltonian simulation, while making sure to not perturb the matrix too much as to drastically change the low-lying spectral density.

Next, we focus on the number of qubits required for the quantum phase estimation. Standard quantum phase estimation requires an eigenvalue register of t qubits to estimate the eigenvalues up to t-bits of precision (which consequently determines the threshold in low-lying spectral density estimation). Fortunately, much improvement is possible in terms of the size of this eigenvalue register. First, as low-lying spectral density is only concerned with whether the t-bit approximation of an eigenvalue is zero or not, we can bring the size the of eigenvalue register down to $\log(t)$ by using a counter [163]. Moreover, we can bring the size of this eigenvalue register down to a single qubit at the expense of classical post-processing and qubit reinitialization methods [74, 148, 180].

We can now give the brief estimate of the number of qubits needed for demonstrations of quantum advantage (i.e., sizes needed to go beyond the best known classical methods). The best known classical methods for low-lying spectral density estimation, to our knowledge, are able to estimate the rank of a matrix in time linear in the number of nonzero entries [190, 59, 70, 124]. These methods are at most quadratically faster than exact diagonalization, which tends to hit a practical wall around matrices of size 2^{40}. We therefore look at how many qubits are required to estimate the low-lying spectral density below a threshold of about 10^{-9} (i.e., $t \approx \log\big(10^9\big) < 30$) of matrices of size around 2^{80} (i.e., $n \approx 80$). In this case, the required number of qubits for standard implementations is approximately

$$2n + r + 1 + t \approx 200.$$

If we precompile the input matrix through finding a decomposition in terms of local Hamiltonians, this can be reduced to

$$n + t \approx 110.$$

This can be further reduced to $n + \log(t)$ by using a counter in the eigenvalue register. Lastly, by using a single-qubit eigenvalue register (at the cost of classical postprocessing and qubit reinitialization) we bring the number of required qubits in the optimal case down to

$$n + 1 \approx 80,$$

which is tantalizingly close to what leading teams are expected to achieve in the immediate future in terms of qubit numbers alone.

When it comes to the robustness to noise in the hardware, we need to consider the type of algorithm that is being applied (i.e., how noise affects this algorithm in general) together with the specifics of the application. The algorithm we consider involves many iterations of Hamiltonian simulation and quantum phase estimation, where we are interested in the expected value of a two outcome measurement (designating the zero eigenvalues). As noted earlier, these routines are also crucial for quantum algorithms for quantum chemistry and many-body physics, and consequently, all error-mitigation methods developed for these purposes can be readily applied [187, 38, 78, 136, 147]. However, as in quantum chemistry and many-body physics one extracts the entire eigenvalues, as opposed to just the frequency of the zero eigenvalue, the application we consider is less demanding. Additional robustness properties van be inferred from the nature of the particular problem solved. For instance, in machine learning and data analysis applications, the fact that the algorithm serves the purpose of dealing with noise in the data might make noise in the hardware less detrimental compared to when solving more exact problems [73].

Unfortunately, this argument cannot be as readily applied to Betti number estimation, as noise in the data does not correspond to small perturbations of the simulated matrix (i.e., the combinatorial Laplacian), but rather to a completely different matrix altogether. In turn, small perturbations of the simulated matrix do not corresponds to any meaningful perturbation of the input data. However, we can still identify certain robust features by considering what perturbations of the combinatorial Laplacian entail for the final output, i.e., the low-lying spectral density. Specifically, if the combinatorial Laplacian is perturbed by a small enough matrix (e.g., in terms of operator norm or rank), then the low-lying spectral density remains largely unchanged as such perturbations will not push the low-lying eigenvalues above the threshold. These settings are often studied in the field of perturbation theory [114], which would allow us to make these arguments completely formal. Moreover, as a random matrix is likely of full rank [80], the perturbed combinatorial Laplacian is also likely of full rank, indicating that in the noisy setting we should focus on approximate Betti number estimation methods, as opposed to exact ones. Finally, there has been work verifying the robustness of the quantum algorithm for Betti number estimation in an experimental setting [106].