Quantum Framework for Topological Data Analysis

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Bernardo Ameneyro

May 2025

© by Bernardo Ameneyro, 2025

All Rights Reserved.

Acknowledgements

I would like to thank my advisors Vasileios Maroulas, George Siopsis and Rebekah Herrman for their guidance throughout this research project as well as my graduate studies in general, along with the rest of my committee Abner Salgado and Vyron Vellis for their time and useful feedback. I also want to recognize the many comments made by the reviewers of the papers that form this work as they greatly improved my results. In addition, my research assistantship has been funded by the National Science Foundation award DMS-2012609. Finally, I want to thank all my family and friends who encourage me to keep doing what I love.

Abstract

Topological Data Analysis (TDA) methods combine tools from statistics and machine learning with concepts from algebraic topology usually with the purpose of classifying data based on its shape. These techniques provide advantages such as dimensionality reduction and resilience to noise. In addition, they can often recover information from signals or other complex dynamical systems that traditional methods fail to capture. In recent years, TDA methods have seen applications in many different classification problems from biology, materials science, robotics, and computer vision, among others. However, extracting topological features from a data set can be computationally expensive as in general it involves an NP problem. With the rapid development of quantum computers and their rise in popularity to deal with similar mathematical problems, it is natural to wonder if these new devices can provide an advantage for TDA. Indeed there have been several recent papers introducing quantum algorithms for TDA and discussing their potential to speedup or complement the current classical counterparts. This manuscript is a compilation of my works on the subject of quantum methods for TDA. My contributions include a novel quantum algorithm for persistent homology that is able to obtain topological features from a data set and track them through changes in resolution. In addition, the algorithm yields more information than other similar quantum algorithms and still has the potential to provide a quadratic speedup over classical counterparts. Furthermore, I introduce a subroutine that allows the aforementioned algorithm and similar ones to work with time series data sets like signals. Finally, I adapt techniques from quantum variational algorithms to estimate distances between persistence diagrams, so as to compare data sets through the topological features extracted by the persistent homology algorithm.

Table of Contents

1 Introduction **1**

2 Background **6**

2.1 Persistent Homology 7

2.2 Spectral Persistet Homology and Dirac Operators 11

2.3 Topological Data Analysis for Point Clouds 13

2.4 Topological Data Analysis for Time Series 14

2.5 Persistence Diagrams 14

2.6 Quantum Computing 17

3 Quantum Algorithm for Persistent Homology **20**

3.1 Representing simplices and chain complexes with quantum states 21

3.2 Projections 23

3.2.1 Membership Oracle 24

3.2.2 Grover's search algorithm 25

3.3 Persistent Dirac operator 25

3.4 Quantum Phase Estimation 29

3.4.1 Implementation of an exponential operator 30

3.5 An application 31

4 Quantum Persistent Homology for Time Series **35**

4.1 Encoding Data and Simplices 36

4.2 Quantum Delay Embedding 36

4.3 Quantum Algorithm for Time Series 38
4.4 Results 39
4.4.1 Periodic time series example 39
4.4.2 An electroencephalogram example 42

5 Quantum Distance Approximation for Persistence Diagrams 44
5.1 Quantum Algorithm for Wasserstein Distance 45
5.2 A More Efficient Distance 53
5.3 Implementations 57
5.3.1 An Illustrative Example 58
5.3.2 Circles vs Noisy Circles 64
5.3.3 Complexity and Comparison with Existing Methods 67

6 Discussion 69

Bibliography 72

Appendix 82

A Proofs of Theorems 5.3 and 5.4 83
A.1 Proof of Feasibility: Theorem 5.3 84
A.2 Proof of Completeness: Theorem 5.4 85

Vita 87

Chapter 1

Introduction

Topological Data Analysis (TDA) encompasses methods that use topological and geometrical properties –such as connected components, holes and voids– to characterize data based on its shape [1, 2, 3, 4]. TDA approaches can provide dimensionality reduction and robustness to noise for large and complex data sets, and they are also invariant under continuous transformations like scaling and rotations. Homology and persistent homology, the mathematical foundations of TDA, have been around for some time [5, 6, 7]. However, in recent years these topological techniques have come together with statistical machine learning tools, and the resulting methods have seen increasing use in tasks like data analysis, visualization, classification and clustering [8, 9, 10, 11, 12]. TDA has numerous applications in different fields ranging from biology and healthcare [13, 14, 15, 16, 17, 18, 19, 20, 21], to chemistry and material science [22, 23, 24, 25, 26, 27], as well as action recognition and robotics [28, 29], handwriting analysis [30], and even sensor networks [31, 32, 33, 34, 35, 36]. Moreover, topological features can represent multi-stability, periodicity, and chaos in dynamical systems. So, there are also TDA approaches for time series data sets [37, 38] tackling problems like signal analysis and identification [39, 40, 41, 42], which are able to capture features that traditional signal analysis techniques fail to detect.

Persistent homology utilizes algebraic groups to identify topological features in a data set. Initially, graph-like structures called simplicial complexes are constructed by connecting data points that are close to each other with respect to a fixed value often referred to as the scale or resolution. Features like connected components, holes and voids can be extracted from simplicial complexes and tracked through changes in the scale or resolution. Recently, persistent spectral approaches were introduced to extract even more information from the data [43]. Their methodology defines a persistent Laplacian operator such that its harmonic spectra encodes the persistent homology of the data. In addition, the non-harmonic spectra captures interesting geometric information that traditional methods for persistent homology cannot see. Various persistent Dirac operators, which act as the square root of the persistent Laplacian, have surfaced since then [44, 45, 46, 47]. These different variants of persistent spectral homology can consider more complex relationships between data points that conventional persistent homology fails to capture. The results are commonly

displayed as persistence diagrams, which are summaries consisting of points that represent the topological features along with the scales when they first appear and disappear [3, 4, 48]. But the process of obtaining persistence diagrams involves an NP-hard problem since given a data set with n points, there are potentially 2^n objects that contribute to the topology. In consequence, the efficient computation of persistent homology has become an active research topic. For example, popular software like Dionysus [49] and Ripser [50, 51, 52] take advantage of special properties of simplicial complexes to build persistence diagrams efficiently [53]. However, these classical algorithms are not easily scalable and tend to deteriorate with the increase in the number of data size. For this reason, classical implementations often restrict attention to features of lower dimensions. Persistence diagrams are useful because they can summarize large and high-dimensional data sets down to simple collections of 2-dimensional points, often used to perform machine learning tasks such as classification or clustering. Machine learning algorithms commonly use the Wasserstein or bottleneck distances to compare persistence diagrams [49, 54], but a more recent alternative distance introduced in [42] has shown better results for certain machine learning problems [12].

Recent advances in quantum computing have lead to the development of quantum methodologies for machine learning and data science, exploring potential advantages for computationally complex problems. TDA is of particular interest since obtaining the homology of a data set involves an NP-hard problem. As such, some works appeared discussing the quantum computation of Betti numbers [55, 56, 57, 58, 59, 60], which are the number of holes in a data set at a fixed scale. However, in order to track topological features across different scales one must compute the persistent Betti numbers instead, i.e., the number of holes that persist from an initial scale to a later scale. So not long after, a few papers extended quantum methodologies for TDA to compute persistent Betti numbers [44, 61, 62, 63]. Complexity analysis suggest the exponential speedup promised in [55] is unlikely in practice, but that Grover-like or quadratic advantage over classical algorithms is possible [64, 65, 66]. On the other hand, some of the hardware required by these algorithms (quantum memories for example) or the techniques they rely on (like Grover's search) either will likely not be available in the near future or will not be efficient in current

noisy quantum computers (NISQ). Hence, a few of these papers have focused on developing methods that support NISQ devices [60, 58, 59]. In addition, quantum approaches to compute the distance between persistence diagrams have surfaced as well [67, 68, 69]. These frameworks formulate a quadratic unconstrained binary optimization problem (QUBO), then encode the QUBO into a Hamiltonian operator and obtain the distance by finding its ground state. The ground state of the Hamiltonian can be estimated using quantum annealing [67] or by means of variational algorithms [68, 69] inspired by the Quantum Approximate Optimization Algorithm (QAOA) [70].

This manuscript outlines my contributions to the development of a quantum framework for TDA. These contributions include a novel quantum algorithm for persistent homology that can estimate persistent Betti numbers and could provide a Grover-like speedup over classical algorithms [44]. My method also provides an advantage over similar quantum algorithms [61, 62] as it can not only estimate the persistent Betti numbers but also the non-zero eigenvalues of the persistent combinatorial (Hodge) Laplacian [43, 71], which can also be used for data analysis tasks, e.g. see [43, 72, 73]. While some of the specific tools and implementations, for instance quantum memory and Grover's search, are not compatible with NISQ devices, other works [60, 58, 59] have suggested alternatives that can be substituted into the algorithm to remedy this issue. In addition, quantum TDA algorithms often assume access to a membership oracle that uses pairwise distances between data points to determine which simplices are present at a given scale. However, the membership oracles previously discussed in the literature [55, 61] were limited to point cloud data sets, so I constructed one based on Takens' delay embedding that enables quantum algorithms for persistent homology [61, 62, 44] to work with time series data sets like signals [63]. Finally, my latest article [69] provides a quantum approach to compute distances between persistence diagrams. This QAOA inspired method formulates the distance as a constrained binary optimization problem, it then encodes the cost of the problem as a cost Hamiltonian and uses control clauses [74] to incorporate the constraints into the mixing Hamiltonian. Furthermore, unlike previous approaches [67, 68] this work also considers a new distance proposed in [42] that

has been shown to outperform the more popular Wasserstein distance in certain machine learning tasks [12].

The layout of this work is as follows. It starts with Chapter 2 which provides relevant background material regarding TDA as well as basic concepts of quantum computing. Then, Chapter 3 describes my quantum algorithm for persistent homology and shows a simple illustrative example. Chapter 4 follows up with the membership oracle for time series data sets along with some examples of the quantum persistent homology algorithm utilizing it. Later, Chapter 5 introduces quantum hybrid algorithms that compute two different distances between persistence diagrams. Finally, Chapter 6 concludes with a discussion of the results shown here and their overall importance. In addition, Appendix A contains relegated proofs of Theorems.

Chapter 2

Background

This chapter briefly introduces basic concepts from TDA and quantum computing that are necessary to understand the main results presented later in the text. In particular, Section 2.1 covers persistent homology which is the main mathematical tool behind TDA methods, and further details on the topic can be found in [5, 6, 7]. Next, Section 2.3 describes the classical computation of persistent homology information for point cloud data sets, and Section 2.4 follows with an embedding approach for time series. Then, Section 2.5 depicts persistence diagrams as a way to display the information obtained from persistent homology algorithms. Any reader who wishes for more details on computational topology methods may consult [1, 3, 4]. Last, Section 2.6 provides some basic concepts and notation from quantum information theory, as well as general aspects of quantum algorithms with relevant examples. Readers unfamiliar with the field of quantum computing may find [75, 76] to be useful references.

2.1 Persistent Homology

Persistent homology studies objects called simplicial complexes. The classical algorithms that perform topological data analysis use the data to construct simplicial complexes, e.g., by connecting all the points in a point cloud within a certain distance from each other (Vietoris-Rips complex) and then varying the distance to obtain a filtration of simplicial complexes. After that, the algorithms proceed to compute the eigenvalues and eigenvectors of linear operators, such as the boundary operator and the persistent combinatorial Laplacian that act on the complexes.

We start by defining simplicial complexes and homology, an algebraic descriptor for coarse shape in topological spaces, and in turn persistent homology, which harnesses the power of homology to the description of subspace filtrations of topological spaces.

Definition 2.1. *A k-dimensional manifold X is a topological space such that if every point x that belongs to X has a homeomorphic neighborhood to an open neighborhood in the k-dimensional Euclidean space.*

Next, we expand the definition of the k-dimensional manifold to k-simplex which may be topologically treated as a k-dimensional manifold including its boundary.

Definition 2.2. *A k-simplex is defined by $k+1$ linearly independent vertices as the collection of all their convex combinations:*

$$\sigma = [v_0, \ldots, v_k] = \left\{ \sum_{i=0}^{k} \alpha_i v_i : \sum_{i=0}^{k} \alpha_i = 1 \text{ and } \alpha_i \geq 0 \text{ for all } i \right\}. \tag{2.1}$$

An oriented simplex is expressed as the ordered list of its vertices, such as $[v_0, v_1, v_2]$. The faces of a simplex consist of all the simplices generated by a subset of its vertex set.

Fig. 2.1 shows the simplices of dimensions zero (vertex), one (edge), two (triangle), and three (tetrahedron).

Definition 2.3. *A simplicial complex $\mathcal{K}$ is a collection of simplices satisfying the following*
(i) if $\sigma \in \mathcal{K}$, then all its faces $\tau \subset \sigma$ are also in $\mathcal{K}$, and
(ii) the intersection $\sigma_1 \cap \sigma_2$ of any pair of simplices $\sigma_1, \sigma_2 \in \mathcal{K}$ is another simplex in $\mathcal{K}$.
The collection of k-simplices within $\mathcal{K}$ is written here as $\mathcal{K}_k$.

A commonly used simplicial complex is the Vietoris-Rips (VR) complex, which for a fixed positive number ϵ is defined by all the simplices with a diameter of at most ϵ (see Definition 2.7). Furthermore, we can use these basis-like collections to define a space of formal sums of simplices. These spaces are called chain groups and are akin to vector spaces, an important feature since quantum states are elements of a Hilbert space over $\mathbb{C}$.

Definition 2.4. *Let us denote with $C_k(\mathcal{K})$ the chain group of dimension k on a simplicial complex $\mathcal{K}$, which is defined by*

$$C_k(\mathcal{K}) = \left\{ \sum_{\sigma \in \mathcal{K}_k} n_\sigma \sigma : n_\sigma \in \mathbb{C} \right\}. \tag{2.2}$$

Relying on Def. 2.4, one understands that chain groups give an algebraic way to describe subsets of simplices as a formal sum. For example, the boundary of a triangle (Fig. 2.2) is considered the sum of its three edges, and the boundary of an edge yields the sum of its

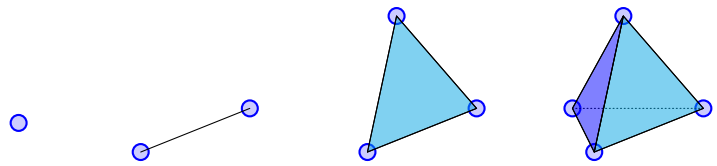

Figure 2.1: Simplices of dimension 0 through 3.

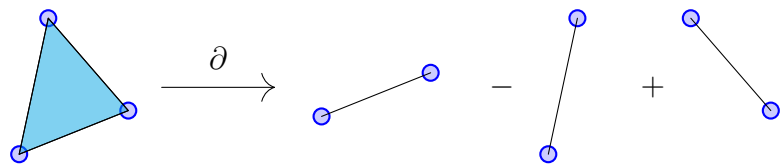

Figure 2.2: Boundary of a triangle

endpoints. The sign of a simplex in one of these sums specifies its orientation. This notion of a boundary is fundamental to persistent homology and is formalized as a map between chain groups.

Definition 2.5. *The k-th boundary map is a homomorphism on the chain groups $\partial_k : C_k(\mathcal{K}) \to C_{k-1}(\mathcal{K})$ defined by its action on each simplex:*

$$\partial_k[v_0, \ldots, v_k] = \sum_{n=0}^{k} (-1)^n [v_0, \ldots, v_{n-1}, v_{n+1}, \ldots, v_k], \tag{2.3}$$

as an alternating sum over the faces of dimension $k-1$.

One may obtain a chain complex by taking into account chain groups (and their boundary maps) of any dimension

$$\ldots \xrightarrow{\partial_{k+1}} C_k(\mathcal{K}) \xrightarrow{\partial_k} C_{k-1}(\mathcal{K}) \xrightarrow{\partial_{k-1}} \ldots \xrightarrow{\partial_1} C_0(\mathcal{K}) \xrightarrow{\mathbf{0}} 0. \tag{2.4}$$

As it can be shown, e.g., see in [3], the composition of subsequent boundary maps $\partial_{k+1}\partial_k = 0$, which yields that $\mathrm{Im}(\partial_{k+1}) \subset \mathrm{Ker}(\partial_k)$. The k-Betti number, denoted by β_k, is the dimension of the homology group $H_k(\mathcal{K}) = \mathrm{Ker}(\partial_k)/\,\mathrm{Im}(\partial_{k+1})$. Homology groups are generated by the topological features of the complex $\mathcal{K}$, i.e., generators for the 0-homology group correspond to connected components, generators of 1-homology group correspond to holes in $\mathcal{K}$, etc.

However data analysis often requires to observe how topological features persist as the resolution of the data changes (see Section 2.3). So, we must extend the notions above to track features across multiple simplicial complexes. Consider a nested sequence or filtration of simplicial complexes

$$\emptyset \subseteq \mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \ldots \subseteq \mathcal{K}_n. \tag{2.5}$$

Each complex $\mathcal{K}_i$ in Eq. (2.5) has associated chain groups $C_k(\mathcal{K}_i)$, boundary operators ∂_k^i and Homology groups H_k^i as before. But now for $i < j$ the inclusion maps $\iota : \mathcal{K}_i \to \mathcal{K}_j$ between complexes induce homomorphisms $h_k^{i,j} = \iota_* : C_k(\mathcal{K}_i) \to C_k(\mathcal{K}_j)$ between their corresponding chain groups.

Definition 2.6. *For two nested simplicial complexes $\mathcal{K}_i \subseteq \mathcal{K}_j$ we define their k-th persistent Homology group $H_k^{i,j}$ as*

$$H_k^{i,j} = Im(h_k^{i,j}) = Ker(\partial_k^i)/(Im(\partial_{k+1}^j) \cap Ker(\partial_k^i)) \tag{2.6}$$

where $Ker(\partial_k^i)$ is viewed as a subgroup of $Ker(\partial_k^j)$. The dimension of $H_k^{i,j}$ is the k-th persistent Betti number $\beta_k^{i,j}$.

Let $\tilde{C}_{k+1}^{i,j} = \{x \in C_{k+1}(\mathcal{K}_j) : \partial_{k+1}^j x \in C_k(\mathcal{K}_i)\}$, that is, the subgroup of $C_{k+1}(\mathcal{K}_j)$ defined by the $(k+1)$-simplices in $\mathcal{K}_j$ with boundary in $\mathcal{K}_i$. Then we can define the k-th persistent combinatorial Laplacian

$$\mathcal{L}_k^{i,j} = \tilde{\partial}_k^{i,i*}\tilde{\partial}_k^{i,i} + \tilde{\partial}_{k+1}^{i,j}\tilde{\partial}_{k+1}^{i,j\ *}, \tag{2.7}$$

where $\tilde{\partial}_{k+1}^{i,j}$ is the restriction of the boundary operator ∂_{k+1}^j to $\tilde{C}_{k+1}^{i,j}$. Using [43], one may show that the dimension of the kernel of $\mathcal{L}_k^{i,j}$ is the k-th persistent Betti number $\beta_k^{i,j}$.

2.2 Spectral Persistet Homology and Dirac Operators

The spectral theory of persistent homology in [43, 71] is based on the k-th persistent combinatorial Laplacian $\mathcal{L}_k^{i,j}$ introduced in Eq. (2.7). The harmonic spectra of the Laplacian correspond to the topological features of conventional persistent homology. However, the non-harmonic spectra can provide useful geometric information that conventional persistent homology methods cannot capture. So, spectral approaches for persistence homology have gained traction in recent years [44, 45, 46, 47].

There is a close relationship between Dirac operators and Laplacian operators, with the former serving a role similar to the square root of the latter. Therefore, Dirac operators can have an important place in persistent spectral homology where the spectra of a persistent Laplacian is used to extract topological and geometrical information from data. For instance, the structure of the Dirac operator for simplicial homology allows an efficient quantum implementation that is useful for quantum algorithms that aim to compute the homology of a simplicial complex [55, 56]. Furthermore, the work in [77] introduces topological Dirac

equations, which are wave equations defined with respect to Dirac operators on networks, simplicial complexes and multiplex networks. This methodology was later utilized to create an algorithm for processing of higher-order topological signals [78].

The quantum approach in [55] was recently extended to the persistent homology of a filtration of simplicial complexes [44]. The harmonic spectra of the persistent Dirac operator we introduced holds information about the persistence of topological features through the filtration. Similarly, this persistent Dirac operator can be efficiently implemented on a quantum computer and can be used in quantum algorithms that compute the persistent homology of a filtration of simplicial complexes.

The persistent Dirac operator has also found interesting applications that rely on mathematical representations of molecular structures [45]. The authors introduce weighted versions of the persistent Dirac operators that extend spectral theory to weighted simplicial complexes. A weighted approach can be useful when the simplicial complexes are constructed from data points that may have different properties, as is the case for atoms in molecular structures. Moreover, their results suggest that molecular structures can be accurately characterized using descriptors based on spectral properties of weighted persistent Dirac operators.

A common limitation of simplicial TDA methods is their inability to consider asymmetric or directional relationships between data points. Such data sets are better modeled by structures like directed graphs and hypergraphs. The work in [46] introduces a persistent Dirac operator for a filtration of path complexes that arise from directed graphs and hypergraphs. The authors also discuss applications of these new Dirac operators to analyze molecular structures.

On the other hand, the chain complexes introduced in Section 2.1 are induced by a boundary map ∂ that satisfies $\partial^2 = 0$. But [47] introduces Mayer Dirac and persistent Mayer Dirac operators for N-chain complexes induced by a map d such that $N > 2$ is the smallest integer for which $d^N = 0$. While persistent Mayer homology is different than the usual persistent homology, their results suggest that it can also be used to analyze molecular structures and other data sets using geometric and topological features. Moreover, the

computation of persistent Mayer homology could be significantly faster than that of the regular persistent Laplacian [79].

2.3 Topological Data Analysis for Point Clouds

When analyzing a discrete data set $\boldsymbol{x} = \{v_i\}_{i=1}^N$ that belongs to a metric space (X, d) it is not enough to consider the set itself as a simplicial complex for its homology would simply yield the number of data points in the set. Instead, we take advantage of the metric d to obtain more information. Indeed, we fix a radius $r > 0$ and consider the collection of neighborhoods (think of them as balls centered at each datum) $U = \{U_i\} = \{B(v_i, r)\}$ along with its union $\mathcal{U}_r = \cup_i B(v_i, r)$. The collection of sets $\{\mathcal{U}_r\}_{r\in\mathbb{R}^+}$ provides information about the arrangement of the dataset $\boldsymbol{x}$ at different scales r. To make homological computations tractable for $\mathcal{U}_r$, one may consider the Vietoris-Rips (VR) complexes.

Definition 2.7. *The Vietoris-Rips complex of the point cloud data $\boldsymbol{x} = \{v_i\}$ at scale ϵ, denoted by $S^\epsilon(\boldsymbol{x})$ (or simply S^ϵ), is the simplicial complex where a k-simplex $\sigma = [v_{i_0}, \ldots, v_{i_k}]$ is in S^ϵ if and only if $diam(\sigma) \leq \epsilon$, where $diam(\sigma) := \max_{j,j'} \left\{ d(v_{i_j}, v_{i_{j'}}) \right\}$ denotes the diameter of the simplicial complex S^ϵ defined as the maximal distance between $v_{i_j}, v_{i_{j'}} \in S^\epsilon$.*

Given a point cloud data set $\boldsymbol{x}$, one can construct a filtration of VR complexes $\emptyset \subseteq S^{\epsilon_1} \subseteq S^{\epsilon_2} \subseteq \ldots \subseteq S^{\epsilon_N} = S$ by choosing an increasing sequence of scales $0 < \epsilon_1 \leq \epsilon_2 \leq \ldots \epsilon_N$, see Section 3.5 for an example. Indeed, it follows from Def. 2.7 that if $\epsilon \leq \epsilon'$ then $S^\epsilon \subseteq S^{\epsilon'}$. Moreover, one can check that there is a maximum of 2^n simplices that can be built from a data set with n points. So, there is a maximal VR complex S of size 2^n that contains all VR complexes S^ϵ.

Classical algorithms for persistent homology consider a filtration or nested sequence of VR complexes $S^{\epsilon_0} \subseteq \cdots \subseteq S^{\epsilon_N}$ and construct a boundary matrix in which the simplices are ordered according to the sequence [48]. Then this boundary matrix is diagonalized to extract topological features like the number of connected components, holes, voids, and k dimensional holes in general. In particular, the position of the non-zero entries of the final

matrix reflect the persistence information. Persistence diagrams are often used to display the results of these algorithms (see Section 2.5).

2.4 Topological Data Analysis for Time Series

Consider a time series x_t for $t = 1, 2, \ldots, T$. Notice that applying the techniques from Section 2.3 to a data set like this would often lead to loss of the sequential time structure. Instead, a common approach is to consider a higher dimensional point cloud in phase space that encodes the time structure as spatial features. The Takens's delay embedding of the time series x_t is given by the vector

$$v_i = (x_i, x_{i+\tau}, \ldots, x_{i+(d-1)\tau}) \tag{2.8}$$

for $i = 1, \ldots, T - \tau d$, where d is the dimension of the point cloud, and τ is the delay parameter. The Takens's delay embedding theorem guarantees that the map from the time series to point cloud is an embedding and as such all the relevant topological information is retained [80]. Then, the methodology from Section 2.3 can be performed on the resulting point cloud to obtain persistence information from the initial time series. The parametric choices of the dimension, d, and the delay, τ, has been widely discussed in the literature, e.g. see [41] and references therein. For example, the time series is typically embedded into a $d = 2, 3$, space, and the delay parameter may be selected based on the autocorrelation.

2.5 Persistence Diagrams

Persistent homology considers a sequence or filtration of simplicial complexes from data sets as in Figures 2.3a and 2.3b. The filtration is built by choosing an increasing sequence of positive numbers representing different scales or resolutions of the data, and connecting the data points that are close to each other with respect to each of these numbers. One can then track how the topological features of different dimensions, like connected components, holes, and voids, appear and disappear as the scale increases. This information is used to

construct persistence diagrams that consist of points (b_η, d_η), where b_η is the scale at which feature η first appears in the filtration, and d_η is the last scale at which it is present. See Figures 2.3c and 2.3d for examples.

Persistence diagrams can be useful for classification of their corresponding data sets since they summarize their shape while being invariant to any continuous transformations such as rotating, stretching or shrinking of the data. For example, the data in Fig. 2.3a clusters around five different locations and this information can be recovered from the five connected components (H_0) away from the diagonal in the corresponding persistence diagram in Fig. 2.3c. On the other hand, Fig. 2.3b shows a data set arranged into two separate circles while the corresponding persistence diagram in Fig. 2.3d has two connected components (H_0) and two one-dimensional holes (H_1) away from the diagonal. Of course, in order to compare data sets using their diagrams one must define a distance on the space of persistence diagrams. The most popular such distance is the Wasserstein distance [3, 4] defined below.

Definition 2.8. *Let $\mathcal{D}_1, \mathcal{D}_2$ be two persistence diagrams, and Δ_1, Δ_2 be their projections to the diagonal. Then, their Wasserstein distance is defined as*

$$d_p^W(\mathcal{D}_1, \mathcal{D}_2) = \inf_{\phi : \tilde{\mathcal{D}_\infty} \leftrightarrow \tilde{\mathcal{D}_\in}} \left(\sum_{x \in \tilde{\mathcal{D}_\infty}} \|x - \phi(x)\|_q^p \right)^{\frac{1}{p}} \tag{2.9}$$

where q is usually chosen as ∞, i.e. the supremum norm, and ϕ are bijections between $\tilde{\mathcal{D}_\infty} = \mathcal{D}_1 \cup \Delta_2$ and $\tilde{\mathcal{D}_\in} = \mathcal{D}_2 \cup \Delta_1$.

Based on Definition 2.8, the Wasserstein distance computes the optimal matching of the points in two diagrams by penalizing any unmatched points with their distance from the diagonal. Figure 2.4a shows two persistence diagrams, one consists of two blue dots and the other of a single orange triangle. In addition, each point has a corresponding disk with a radius equal to its distance from the diagonal, which represents the penalty for unmatched points. Since two disjoint disks are equivalent to the distance between the corresponding points being larger than their combined distances to the diagonal, one can disregard any bijections ϕ in Def. 2.8 that match points with disjoint disks. Notice that one could have two different diagrams that consist of completely disjoint disks so that all of their points are

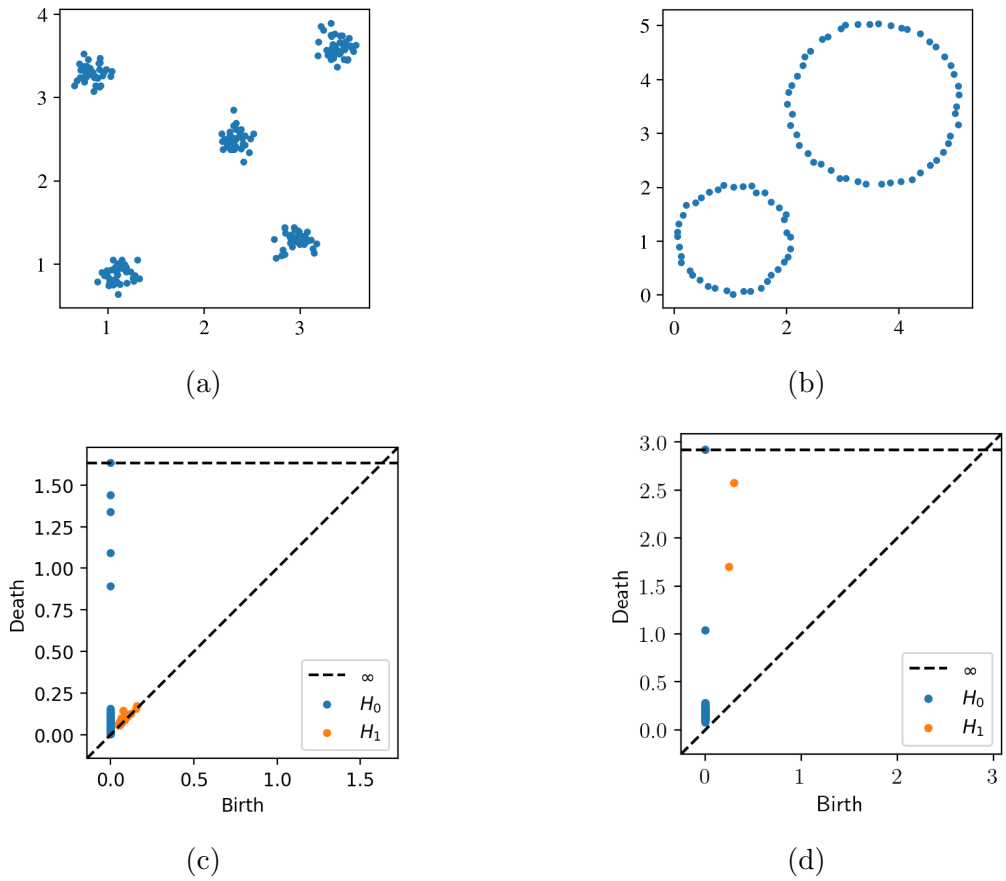

Figure 2.3: Examples of persistence diagrams for point cloud data sets.

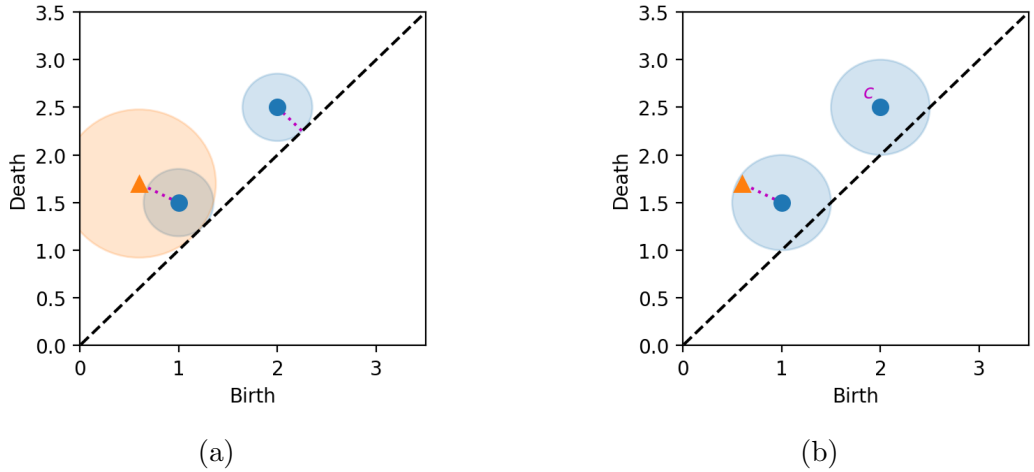

Figure 2.4: Example of the Wasserstein (a) and d_p^c (b) distances between a persistence diagram with two points (blue dots) and another with one point (orange triangle). Disks around the points illustrate the penalization mechanism of each distance. The optimal matching in each case is given by the dashed lines.

matched to the diagonal, however if all of these points are also very close to the diagonal then the Wasserstein distance would be rather small even though the diagrams are different. This could play an important role in supervised or unsupervised learning [12], so one could instead consider the following distance in such a case as it was introduced in [42].

Definition 2.9. *Consider two persistence diagrams $\mathcal{D}_1, \mathcal{D}_2$ with respective cardinalities n, m such that $n \leq m$. Then, their d_p^c distance is defined as*

$$d_p^c(\mathcal{D}_1, \mathcal{D}_2) = \left(\frac{1}{m} \left(\inf_{\phi:\mathcal{D}_1 \hookrightarrow \mathcal{D}_2} \sum_{x \in \mathcal{D}_1} \min\left(c, \|x - \phi(x)\|_q\right)^p + c^p |n - m| \right) \right)^{\frac{1}{p}} \tag{2.10}$$

where q is usually chosen as ∞, i.e. the supremum norm, and ϕ are one-to-one functions from $\mathcal{D}_1$ to $\mathcal{D}_2$.

The d_p^c distance still seeks to match points from two persistence diagrams such that the distance is minimized, but it penalizes unmatched points with a constant value c rather than their distance to the diagonal. Fig. 2.4b shows the same two persistence diagrams that appear in Figure 2.4a, but now there are only two disks of radius c around the blue points. In order to find the optimal matching that yields the d_p^c distance, it suffices to consider only the functions ϕ in Def. 2.9 that match points from the largest diagram to points that lie within their corresponding disk of radius c. In addition, any points that are isolated with respect to the penalizing constant c can be completely ignored when computing the distance and simply added later. Notice that choosing a large enough c solves the issue that the Wasserstein distance has when points in the diagrams are close to the diagonal. Moreover, the results provided in [42] show that its accuracy for classification tasks is comparable or better (in the data problems of that manuscript) than that of the Wasserstein distance.

2.6 Quantum Computing

Quantum theory relies on linear algebra of complex vector spaces. In particular, a quantum system is described by a Hilbert space $\mathcal{H}$ over the complex numbers. Vectors of the Hilbert

space $\mathcal{H}$ are called quantum states and are written using a ket $|\cdot\rangle$ as $|\psi\rangle \in \mathcal{H}$. We may write any quantum state $|\psi\rangle$ as a linear combination with respect to the standard basis $\{|\boldsymbol{k}\rangle\}_k$

$$|\psi\rangle = \sum_{i \in k} \psi_i \, |\boldsymbol{i}\rangle \,, \tag{2.11}$$

where the coefficients $\psi_i \in \mathbb{C}$ called amplitudes are normalized so that $\sum_i |\psi_i|^2 = 1$. Linear combinations of quantum states like Eq. (2.11) with at least two $\psi_i \neq 0$ are called quantum superpositions. Elements of the dual space are represented using bra $\langle\cdot|$ notation, for instance the dual vector induced by $|\psi\rangle$ from Eq. (2.11) is $\langle\psi| = \psi_0^* \langle\mathbf{0}| + \psi_1^* \langle\mathbf{1}| + \ldots$, with ψ_k^* denoting the complex conjugate.

Classical computers have a fundamental unit of information called bit, which can take a value of 0 or 1. Similarly, quantum computers consider a fundamental unit of information called quantum bit or qubit. A qubit is the simplest example of a quantum system and is described by the Hilbert space over the complex numbers with standard basis $\{|0\rangle, |1\rangle\}$. While a qubit is generally in a state given by a vector $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, an observer cannot see this superposition and instead must perform a measurement to extract useful information. Measuring $|\psi\rangle$ with respect to the standard basis for example will result in an outcome of 0 with probability $|\alpha|^2$, or an outcome of 1 with probability $|\beta|^2$. However, the state of a qubit collapses after a measurement to the state that was observed, so in order to recover a superposition one must prepare it and measure several times. We can construct larger quantum systems by considering arrays or registers of qubits, which are characterized by the tensor product of the Hilbert spaces for each qubit. Indeed, a register of n qubits is described by the Hilbert space over the complex numbers where the standard basis is given by vectors of the form $|b_0\rangle \otimes |b_1\rangle \otimes \cdots \otimes |b_{n-1}\rangle$ with $b_k = 0, 1$. To avoid cumbersome notation these vectors are often written as $|b_0 \ldots b_{n-1}\rangle$ instead, or even just enumerated as $|\mathbf{0}\rangle, \ldots, |\mathbf{2^n - 1}\rangle$ when the structure of the qubits is not relevant.

The state of a quantum system can be modified via unitary operators. For example, Pauli operators (X, Y, Z) are some of the most common single qubit gates, and they can be defined by their actions on the standard basis $\{|0\rangle, |1\rangle\}$. The bit-flip operator, X, is the

quantum equivalent of the classical NOT gate, it sends the state $|0\rangle$ to $|1\rangle$ and vice versa. On the other hand, the phase-flip operator, Z, leaves the state $|0\rangle$ unchanged but flips the sign of the state $|1\rangle$. For more details see [75].

Quantum algorithms consist of a series of unitary operations $U_1, \ldots, U_n$ applied to an initial state $|\psi^{\text{in}}\rangle$. The result is an output quantum state $|\psi^{\text{out}}\rangle = U_n U_{n-1} \cdot U_1 |\psi^{\text{in}}\rangle$ which can be either passed to another quantum algorithm or measured. Since current and near future quantum hardware is not fault tolerant, hybrid quantum-classical algorithms are often more attractive than pure quantum ones. The Quantum Approximate Optimization Algorithm (QAOA) [70] is such an example of a hybrid algorithm. QAOA approximates the ground state of a Hamiltonian by alternating two unitary operators U_C and U_M applied to a suitable initial state. The unitary $U_C = e^{-i\gamma H_C}$ is defined by the problem Hamiltonian H_C and a parameter $\gamma \in [0, 2\pi)$, while the other operator $U_M = e^{-i\beta H_M}$ consists of a mixing Hamiltonian H_M and a parameter $\beta \in [0, 2\pi)$.

Here, the problem Hamiltonian H_C encodes the cost of the optimization problem to be solved, while the mixing Hamiltonian H_M is used to produce a superposition over all the quantum states of interest (those encoding possible solutions). After p iterations of the unitaries applied to the initial state, one obtains the trial state $|\psi(\beta, \gamma)\rangle_p$. The goal of the algorithm is to find parameters β and γ that maximize the expected value of the problem Hamiltonian: ${}_p\langle\psi(\beta, \gamma)| \, |H_C|\psi(\beta, \gamma)\rangle_p$. QAOA has been used to solve combinatorial optimization problems over binary variables, such as MaxCut problems which involve partitioning the vertices of a graph into two sets in a way that maximizes the number of edges between sets [81, 74]. Chapter 5 displays the process of how computing the distance between two persistence diagrams can be viewed as a combinatorial optimization problem over binary variables.

Chapter 3

Quantum Algorithm for Persistent Homology

Now I introduce one of the main results of my work, a quantum algorithm that estimates the persistent Betti numbers along with the non-harmonic spectra of the persistent combinatorial Laplacian in Eq. (2.7). The following is an extract from a paper published by myself and my advisors [44].

The quantum algorithm proceeds by first mapping the simplices and other concepts from persistent homology onto quantum states and Hermitian operators. We choose a direct mapping between the n vertices of a point cloud and n qubits. In this manner, all 2^n possible simplices can be mapped onto quantum computational basis states in an n-qubit Hilbert space, using exponentially fewer memory space than the classical case. This allows us to use the fermionic representation of the boundary operator [58, 82] which is defined in terms of Pauli operators, see Eq. (3.2). Moreover, the chain groups are identified by closed subspaces of the n-qubit Hilbert space, so we can use their respective orthogonal projection to restrict the boundary operator as in Eq. (2.7) for the persistent combinatorial Laplacian. To implement these projections we suggest the use of a quantum memory along with Grover's search, however NISQ friendly alternatives have been discussed in other works [58, 60, 59]. Finally, our algorithm uses quantum phase estimation to recover the spectra of the persistent combinatorial Laplacian, including the persistent Betti numbers. Our main contributions include the implementation of a projection onto the closed subspace that encodes the chain group $C^{\epsilon,\epsilon'}$. In addition, we introduce the persistent Dirac operator, which allows our algorithm to recover the full spectrum of the persistent combinatorial Laplacian, in contrast to other quantum algorithms for persistent homology [61, 62] that can only obtain the kernel.

3.1 Representing simplices and chain complexes with quantum states

A k-simplex $\sigma = [v_{i_1}, \ldots, v_{i_k}]$ can be stored in a n-qubit register as $|\sigma\rangle = |v_1\rangle \otimes \cdots \otimes |v_n\rangle$, with 1s at the positions of its $k+1$ vertices $v_{i_1}, \ldots, v_{i_k}$ and 0s elsewhere. Let S denote the collection of all possible quantum basis states of n qubits given by a string of n 0s and 1s, then S effectively encodes the maximal VR complex described in Section 2.3 with all possible

simplices that can be formed with n vertices or data points. Moreover, $\mathcal{H}$, the Hilbert space over $\mathbb{C}$ with basis S encodes the chain group of that complex.

We denote by S_k the subset of states in S encoding k-simplices, and by $\mathcal{H}_k$ the corresponding closed subspace of $\mathcal{H}$ which encodes the k-th chain group defined in Eq. (2.2). Notice that the dimension of a simplex can be recovered from its quantum state encoding by counting the number of qubits in state 1.

Similarly, we write S^ϵ for the subset of S encoding simplices of diameter at most ϵ, in other words the VR complex at scale ϵ as in Def. 2.7, and $\mathcal{H}^\epsilon$ for the corresponding closed subspace of $\mathcal{H}$. We also need to consider the closed subspace of $\mathcal{H}$ that encodes the chain group $\tilde{C}^{\epsilon,\epsilon'}$ of elements present at scale ϵ' with boundary in scale ϵ. This subspace is given by

$$\mathcal{H}^{\epsilon,\epsilon'} = \{|\psi\rangle \in \mathcal{H}^{\epsilon'} : \partial |\psi\rangle \in \mathcal{H}^{\epsilon}\}. \tag{3.1}$$

A boundary operator can be defined on $\mathcal{H}$ using Pauli gates X, Y, Z as

$$\partial := \sum_{i=0}^{n-1} Z^{\otimes(n-1-i)} \otimes X^+ \otimes I^{\otimes i} \ , \tag{3.2}$$

where $X^\pm = \frac{1}{2}(X \pm iY)$. Since $X^+ |1\rangle = |0\rangle$ and $X^+ |0\rangle = 0$, it maps $\mathcal{H}_k$ to $\mathcal{H}_{k-1}$. This representation of the boundary map introduced in [58] has various properties. In particular, notice that ∂ is a bounded continuous linear operator on $\mathcal{H}$ such that $\partial^2 = 0$. The reader may refer to [82] for further details on these properties, and its advantages over the representation introduced in [55]. Mapping $\mathcal{H}_{k-1}$ to $\mathcal{H}_k$, its adjoint is given by $\partial^* := \sum_{i=0}^{n-1} Z^{\otimes(n-1-i)} \otimes X^- \otimes I^{\otimes i}$. In order to obtain a Hermitian version of the boundary operator in Eq. (3.2) that can act on the subspaces considered by the persistent combinatorial Laplacian in Eq. (2.7), we attach two ancillary qubits and define the Dirac operator

$$B = (|0\rangle\langle 1| + |1\rangle\langle 2|) \otimes \partial + (|1\rangle\langle 0| + |2\rangle\langle 1|) \otimes \partial^* \tag{3.3}$$

on the Hilbert space spanned by states $\{|0\rangle, |1\rangle, |2\rangle\}$, where $|0\rangle = |0\rangle \otimes |0\rangle$, $|1\rangle = |0\rangle \otimes |1\rangle$, and $|2\rangle = |1\rangle \otimes |0\rangle$.

3.2 Projections

To implement a projection P_k onto the subspace $\mathcal{H}_k$ generated by simplices of dimension k it suffices to count the number of qubits in state 1. Indeed, computational basis states in $\mathcal{H}_k$ are exactly those that have $k+1$ qubits in state 1. The reader may refer to [58, 59] for details on efficient implementations.

On the other hand, for a scale ϵ, we assume the existence of an oracle

$$\mathcal{O}^\epsilon \left|\sigma\right\rangle \left|0\right\rangle = \left|\sigma\right\rangle \left|a_\sigma^\epsilon\right\rangle \ , \tag{3.4}$$

where $a_\sigma^\epsilon = 1_{\sigma \in S^\epsilon}$ determines the membership in S^ϵ, which in turn encodes the simplices in the ϵ-complex. For completeness we provide a construction of such oracle in Subsection 3.2.1.

The oracle in Eq. (3.4) can be used for the implementation of the projection operators,

$$P^\epsilon = \sum_{\sigma \in S^\epsilon} \left|\sigma\right\rangle \left\langle\sigma\right| , \tag{3.5}$$

onto $\mathcal{H}^\epsilon$, the subspace of $\mathcal{H}$ spanned by the simplices in S^ϵ, which will be needed to construct the persistent Dirac operator later in Section 3.3. To implement the projection P^ϵ, we perform amplitude amplification [83] based on Grover's search algorithm [84] (for details, see Subsection 3.2.2).

For a NISQ implementation of the projections P^ϵ, i.e. an implementation that doesn't require a fault tolerant computer like Grover's search algorithm or a quantum memory, see the approach by Ubaru et. al. [58]. This approach involves classical computation of the pairwise distance of the vertices as well as the $n \times n$ ϵ-adjacency matrix $\{A_{i,j}\}$. The quantum circuit is then built by adding gates conditional to the values $A_{i,j}$.

To implement the projection $P^{\epsilon,\epsilon'}$ onto the subspace $\mathcal{H}^{\epsilon,\epsilon'}$ defined in Eq. (3.1), we introduce the operator $\mathcal{W} = (\left|0\right\rangle \left\langle 1\right| + \left|1\right\rangle \left\langle 0\right|) \otimes I - \left|0\right\rangle \left\langle 2\right| \otimes \partial - \left|2\right\rangle \left\langle 0\right| \otimes \partial^*$, defined with the aid of a pair of qubits, along with the projection $Q^{\epsilon,\epsilon'} = \left|0\right\rangle \left\langle 0\right| \otimes I + \left|1\right\rangle \left\langle 1\right| \otimes P^\epsilon + \left|2\right\rangle \left\langle 2\right| \otimes P^{\epsilon'}$. Then, we consider, $\mathcal{W}^{\epsilon,\epsilon'} = Q^{\epsilon,\epsilon'} (I - \mathcal{W}) Q^{\epsilon,\epsilon'}$, acting on a vector of the form $\left|\phi\right\rangle = \left|0\right\rangle \left|\phi_0\right\rangle + \left|1\right\rangle \left|\phi_1\right\rangle + \left|2\right\rangle \left|\phi_2\right\rangle$. Notice that $\mathcal{W}^{\epsilon,\epsilon'}$ acts as the identity on states that satisfy

$$\partial P^{\epsilon'} |\phi_2\rangle = P^{\epsilon} |\phi_1\rangle \tag{3.6a}$$

$$(I - P^{\epsilon}) |\phi_1\rangle = P^{\epsilon} |\phi_0\rangle \tag{3.6b}$$

$$(I - P^{\epsilon'}) |\phi_2\rangle = P^{\epsilon'} \partial^* |\phi_0\rangle . \tag{3.6c}$$

In particular, (3.6b) and (3.6c) only hold if both sides are equal to zero. Thus, we may conclude that $|\phi_1\rangle \in \mathcal{H}^{\epsilon}$ and $|\phi_2\rangle \in \mathcal{H}^{\epsilon'}$. It follows then from Eq. (3.6a) that $\partial |\phi_2\rangle \in \mathcal{H}^{\epsilon}$, so that $|\phi_2\rangle \in \mathcal{H}^{\epsilon,\epsilon'}$. Therefore, the projection operator $P^{\epsilon,\epsilon'}$ can be implemented by projecting onto the eigenspace of $\mathcal{W}^{\epsilon,\epsilon'}$ with eigenvalue 1.

3.2.1 Membership Oracle

We can encode the order k of a simplex σ in a state $|k\rangle$ ($k = 0, 1, \ldots, n-1$) by starting from $|0\rangle$ and performing permutations $0 \to 1 \to \cdots \to n-1 \to 0$, conditional upon the corresponding digit of σ being 1. Thus, we perform k permutations mapping $|0\rangle \to |k\rangle$. This can be implemented efficiently because the permutation is a 1-sparse matrix.

To encode the scale ϵ we need information on the data points that can be stored in quantum parallel in QRAM, if it is available, and accessed efficiently [85, 86]. For any $i, j = 1, 2, \ldots, n$, QRAM $|i\rangle |j\rangle |0\rangle = |i\rangle |j\rangle |d(i,j)\rangle$, where $d(i,j)$ is the distance between points i and j. Notice that the size of the memory is only logarithmic on the number of data points. We introduce a register of qubits to record the parameter ϵ as $|\epsilon\rangle$. We need to know when $d(i,j) \leq \epsilon$ to form a VR complex. This information will be stored in a qubit initially in the state $|0\rangle$, and flipped if the membership condition is satisfied. This is implemented with a unitary test that uses the qubit registers storing $d(i,j)$ and ϵ as controls to flip the last qubit,

$$U^{\epsilon}_{\text{test}} |d(i,j)\rangle |\epsilon\rangle |0\rangle = |d(i,j)\rangle |\epsilon\rangle |a^{\epsilon}(i,j)\rangle \ , \quad a^{\epsilon}(i,j) = \begin{cases} 0 & , \quad d(i,j) > \epsilon \\ 1 & , \quad d(i,j) \leq \epsilon \end{cases} \tag{3.7}$$

Next, in order to know if $\sigma \in S^\epsilon$, we must check if $d(i,j) \leq \epsilon$ for all (i,j) pairs such that $v_i = v_j = 1$. To this end, we make $\mathcal{O}(k^2)$ calls to QRAM, where k is the dimension of σ. For each pair (i,j), we use $|\sigma\rangle$ as control to call QRAM and apply the test provided $v_i = v_j = 1$, $\text{QRAM}^\dagger U^\epsilon_{\text{test}} \text{QRAM} \, |\sigma\rangle \, |i\rangle \, |j\rangle \, |0\rangle \, |\epsilon\rangle \, |0\rangle = |\sigma\rangle \, |i\rangle \, |j\rangle \, |0\rangle \, |\epsilon\rangle \, |a^\epsilon(i,j)\rangle$. The membership of σ in the VR complex, S^ϵ, is decided if for all (i,j) we end up with $a^\epsilon(i,j) = 1$.

3.2.2 Grover's search algorithm

Here we review the salient features of amplitude amplification [83] and Grover's search algorithm [84] which are needed for the implementation of the projection P^ϵ (Eq. (3.5)), for completeness.

Let $|\Psi_k\rangle \in \mathcal{H}_k$ be a state in the span of the k-simplex states. We wish to construct the normalized projected state $|\Psi^\epsilon_k\rangle = \frac{P^\epsilon|\Psi_k\rangle}{\|P^\epsilon|\Psi_k\rangle\|} \in \mathcal{H}^\epsilon_k$, assuming that it exists. To this end, we introduce the unitary operator $U_G = -U_{\Psi_k} U^\epsilon$, with $U_{\Psi_k} = I - 2\,|\Psi_k\rangle\,\langle\Psi_k|$ and $U^\epsilon = I - 2P^\epsilon$. Since $\mathcal{H}^\epsilon_k$ is a closed subspace, we may write $|\Psi_k\rangle$ as $|\Psi_k\rangle = \sin\theta\,|\Psi^\epsilon_k\rangle + \cos\theta\,|\bar{\Psi}^\epsilon_k\rangle$, where $|\Psi^\epsilon_k\rangle \in \mathcal{H}^\epsilon_k$ and $|\bar{\Psi}^\epsilon_k\rangle \in \mathcal{H}^{\epsilon\perp}_k$. Notice that $\sin\theta = \|P^\epsilon\,|\Psi_k\rangle\|$. We can think of $|\Psi_k\rangle$ as the vector $(\sin\theta, \cos\theta)^T$ in the two-dimensional space spanned by $\{|\Psi^\epsilon_k\rangle, |\bar{\Psi}^\epsilon_k\rangle\}$, then U_G acts as a rotation by an angle 2θ. Applying it K times, we obtain the state $U^K_G\,|\Psi_k\rangle = \sin(2K+1)\theta\,|\Psi^\epsilon_k\rangle + \cos(2K+1)\theta\,|\bar{\Psi}^\epsilon_k\rangle$. This is close to the desired state for $(2K+1)\theta \approx \frac{\pi}{2}$. Therefore, the number of Grover steps needed is $K = \lfloor \frac{\pi}{4\theta} \rfloor$. As discussed in [66], K could be exponential on the number of data points if $\|P^\epsilon\,|\Psi_k\rangle\|$ is small, for example when the number of simplices present in S^ϵ_k is only polynomial on the number data points.

3.3 Persistent Dirac operator

For persistent homology, we need to restrict the space on which the Dirac operator acts. Precisely, the Dirac operator should act on quantum states of the form $|0\rangle\,|\psi_0\rangle + |1\rangle\,|\psi_1\rangle + |2\rangle\,|\psi_2\rangle$, where $|\psi_0\rangle \in \mathcal{H}^\epsilon_{k-1}$, $|\psi_1\rangle \in \mathcal{H}^\epsilon_k$, and $|\psi_2\rangle \in \mathcal{H}^{\epsilon,\epsilon'}_{k+1}$. Moreover, we want the boundary operator and its adjoint to act like the restricted boundary and coboundary in Eq. (2.7) of

the Laplacian. To that end, we introduce the persistent Dirac operator which plays a central role in our quantum algorithm.

Definition 3.1. *Let P^ϵ and $P^{\epsilon,\epsilon'}$ be the respective orthogonal projections onto the closed subspaces $\mathcal{H}^\epsilon$ and $\mathcal{H}^{\epsilon,\epsilon'}$ of $\mathcal{H}$, defined in Section 3.2. Then, the persistent Dirac operator $B^{\epsilon,\epsilon'}$ is defined as $B^{\epsilon,\epsilon'} = -PBP$, where B is the Dirac operator in Eq. (3.3) and*

$$P = (|0\rangle\langle 0| - |1\rangle\langle 1|) \otimes P^\epsilon + |2\rangle\langle 2| \otimes P^{\epsilon,\epsilon'}. \tag{3.8}$$

We can visualize this operator as a block matrix

$$B^{\epsilon,\epsilon'} = \begin{pmatrix} 0 & P^\epsilon \partial P^\epsilon & 0 \\ P^\epsilon \partial^* P^\epsilon & 0 & P^\epsilon \partial P^{\epsilon,\epsilon'} \\ 0 & P^{\epsilon,\epsilon'} \partial^* P^\epsilon & 0 \end{pmatrix} \tag{3.9}$$

The importance of the persistent Dirac operator is that we can use it to recover the eigenvalues of the persistent combinatorial Laplacian from Eq. (2.7). This is summarized below in Theorem 3.1.

Theorem 3.1. *The positive eigenspaces of the persistent combinatorial Laplacian introduced in Eq. (2.7) have a one-to-one correspondence with the eigenspaces of the persistent Dirac operator introduced in Def. 3.1 corresponding to positive eigenvalues. That is, $\lambda > 0$ is an eigenvalue of the persistent Dirac operator, defined in Eq. (3.9), with eigenvector $|0\rangle|\psi_0\rangle + |1\rangle|\psi_1\rangle + |2\rangle|\psi_2\rangle$ if and only if $|\psi_0\rangle, |\psi_1\rangle \in \mathcal{H}^\epsilon$ and $|\psi_2\rangle \in \mathcal{H}^{\epsilon,\epsilon'}$, and $|\psi_1\rangle$ is an eigenvector of the persistent combinatorial Laplacian $\mathcal{L}^{\epsilon,\epsilon'}$ from Eq. (2.7) with eigenvalue λ^2. Moreover, $|\psi_0\rangle$ and $|\psi_2\rangle$ are uniquely defined by $|\psi_1\rangle$.*

Proof. The vector $|0\rangle |\psi_0\rangle + |1\rangle |\psi_1\rangle + |2\rangle |\psi_2\rangle$ is an eigenstate of the persistent Dirac operator if and only if it satisfies the conditions

$$P^\epsilon \partial P^\epsilon |\psi_1\rangle = \lambda |\psi_0\rangle \tag{3.10a}$$

$$P^\epsilon \partial P^{\epsilon,\epsilon'} |\psi_2\rangle + P^\epsilon \partial^* P^\epsilon |\psi_0\rangle = \lambda |\psi_1\rangle \tag{3.10b}$$

$$P^{\epsilon,\epsilon'} \partial^* P^\epsilon |\psi_1\rangle = \lambda |\psi_2\rangle \tag{3.10c}$$

Notice that the left hand side of Equations (3.10a) and (3.10b) are contained in $\mathcal{H}^\epsilon$ because of the projections, implying that $|\psi_0\rangle, |\psi_1\rangle \in \mathcal{H}^\epsilon$. Similarly, Eq. (3.10c) is satisfied when $|\psi_2\rangle \in \mathcal{H}^{\epsilon,\epsilon'}$. Moreover, since $\lambda \neq 0$, Eqs. (3.10a) and (3.10c) express $|\psi_0\rangle$ and $|\psi_2\rangle$ in terms of $|\psi_1\rangle$, and substituting into Eq. (3.10b), we obtain $\left(P^\epsilon \partial^* \partial P^\epsilon + \partial P^{\epsilon,\epsilon'} \partial^* P^\epsilon\right) |\psi_1\rangle = \lambda^2 |\psi_1\rangle$. □

However, the above theorem does not hold for $\lambda = 0$, because the states $|\psi_0\rangle$ and $|\psi_2\rangle$ are no longer uniquely defined by $|\psi_1\rangle$. In particular, the kernel of the persistent Dirac operator contains elements of $\mathcal{H}^{\epsilon\perp}$ and $\mathcal{H}^{\epsilon,\epsilon'\perp}$. To recover information about the kernel of the persistent combinatorial Laplacian, i.e., the persistent Betti numbers, we must introduce a gap in the spectrum by considering a shifted version of the persistent Dirac operator.

Definition 3.2. *Let $B^{\epsilon,\epsilon'}$ denote the persistent Dirac operator introduced in Def. 3.1, and take $\xi > 0$. Then we define the ξ-shift persistent Dirac operator $B^{\epsilon,\epsilon'}[\xi]$ as $B^{\epsilon,\epsilon'}[\xi] := B^{\epsilon,\epsilon'} - \xi P$, where P is defined in (3.8).*

Based on Definition 3.2, one may rewrite $B^{\epsilon,\epsilon'}[\xi]$ as a block matrix

$$B^{\epsilon,\epsilon'}[\xi] = \begin{pmatrix} -\xi P^\epsilon & P^\epsilon \partial P^\epsilon & 0 \\ P^\epsilon \partial^* P^\epsilon & \xi P^\epsilon & P^\epsilon \partial P^{\epsilon,\epsilon'} \\ 0 & P^{\epsilon,\epsilon'} \partial^* P^\epsilon & -\xi P^{\epsilon,\epsilon'} \end{pmatrix}. \tag{3.11}$$

Next, by shifting the persistent Dirac operator, we introduce a gap in its spectrum which is needed to avoid overcounting of vectors in the kernel of the persistent combinatorial Laplacian. Theorem 3.2 states that we can use the positive eigenvalues of the shifted persistent Dirac operator to recover the full spectrum of the persistent combinatorial Laplacian $\mathcal{L}^{\epsilon,\epsilon'}$.

Theorem 3.2. *The eigenspaces of the persistent combinatorial Laplacian introduced in Eq. (2.7) have a one-to-one correspondence with the eigenspaces of the ξ-shift persistent Dirac operator introduced in Def. 3.2 corresponding to a positive eigenvalue. In particular, for any eigenvalue γ of the persistent combinatorial Laplacian $\mathcal{L}^{\epsilon,\epsilon'}$ with corresponding eigenvector $|\psi_1\rangle$, there is a unique choice of $\lambda > 0$, $|\psi_0\rangle$ and $|\psi_2\rangle$ such that the quantum state $|0\rangle|\psi_0\rangle + |1\rangle|\psi_1\rangle + |2\rangle|\psi_2\rangle$ is an eigenvector of $B^{\epsilon,\epsilon'}[\xi]$ with eigenvalue λ. Moreover, $|\psi_0\rangle, |\psi_1\rangle \in \mathcal{H}^{\epsilon}$ and $|\psi_2\rangle \in \mathcal{H}^{\epsilon,\epsilon'}$, and the eigenvalues satisfy $\lambda^2 - \xi^2 = \gamma$.*

Proof. The vector $|0\rangle|\psi_0\rangle + |1\rangle|\psi_1\rangle + |2\rangle|\psi_2\rangle$ is an eigenstate of the ξ-shift persistent Dirac operator if and only if it satisfies the conditions

$$-\xi P^{\epsilon}|\psi_0\rangle + P^{\epsilon}\partial P^{\epsilon}|\psi_1\rangle = \lambda|\psi_0\rangle \tag{3.12a}$$

$$P^{\epsilon}\partial P^{\epsilon,\epsilon'}|\psi_2\rangle + \xi P^{\epsilon}|\psi_1\rangle + P^{\epsilon}\partial^* P^{\epsilon}|\psi_0\rangle = \lambda|\psi_1\rangle \tag{3.12b}$$

$$P^{\epsilon,\epsilon'}\partial^* P^{\epsilon}|\psi_1\rangle - \xi P^{\epsilon,\epsilon'}|\psi_2\rangle = \lambda|\psi_2\rangle \tag{3.12c}$$

Notice that for $\lambda > 0$ the left hand side of Eqs. (3.12a) and (3.12b) are again contained in $\mathcal{H}^{\epsilon}$, similarly the left side of Eq. (3.12c) is in $\mathcal{H}^{\epsilon,\epsilon'}$. This implies that the corresponding eigenvectors must satisfy $|\psi_0\rangle, |\psi_1\rangle \in \mathcal{H}^{\epsilon}$ and $|\psi_2\rangle \in \mathcal{H}^{\epsilon,\epsilon'}$. Therefore, we may rewrite the conditions as

$$P^{\epsilon}\partial P^{\epsilon}|\psi_1\rangle = (\lambda + \xi)|\psi_0\rangle \tag{3.13a}$$

$$P^{\epsilon}\partial P^{\epsilon,\epsilon'}|\psi_2\rangle + P^{\epsilon}\partial^* P^{\epsilon}|\psi_0\rangle = (\lambda - \xi)|\psi_1\rangle \tag{3.13b}$$

$$P^{\epsilon,\epsilon'}\partial^* P^{\epsilon}|\psi_1\rangle = (\lambda + \xi)|\psi_2\rangle \tag{3.13c}$$

Furthermore, since $\lambda \neq -\xi$, we can express $|\psi_0\rangle$ and $|\psi_2\rangle$ in terms of $|\psi_1\rangle$ using Equations (3.13a) and (3.13c), respectively, and then substitute into Eq. (3.13b) to obtain $\left(P^{\epsilon}\partial^*\partial P^{\epsilon} + \partial P^{\epsilon,\epsilon'}\partial^* P^{\epsilon}\right)|\psi_1\rangle = (\lambda^2 - \xi^2)|\psi_1\rangle$. Therefore, $|\psi_1\rangle$ is an eigenvector of the persistent combinatorial Laplacian $\mathcal{L}^{\epsilon,\epsilon'}$ with eigenvalue $\gamma = \lambda^2 - \xi^2$. In particular, the kernel of $\mathcal{L}^{\epsilon,\epsilon'}$ and hence the persistent Betti numbers are given by $\lambda = \xi$.

□

3.4 Quantum Phase Estimation

We may recover the persistent Betti number $\beta_k^{\epsilon,\epsilon'}$ as the dimension of the eigenspace of the ξ-shift persistent Dirac operator $B^{\epsilon,\epsilon'}[\xi]$ introduced in Def. 3.2 with eigenvalue $\lambda = \xi$. To estimate the dimensions of the eigenspaces of the shifted persistent Dirac operator we use a quantum phase estimation algorithm described below. This method not only yields the persistent Betti numbers but also the non-harmonic spectra of the Laplacian in Eq. (2.7).

For explicit calculations, we restrict $B^{\epsilon,\epsilon'}[\xi]$ to the desired dimension k, by restricting attention to the Hilbert space of states $|\psi\rangle = |0\rangle\,|\psi_0\rangle + |1\rangle\,|\psi_1\rangle + |2\rangle\,|\psi_2\rangle$, where $|\psi_0\rangle \in \mathcal{H}_{k-1}$, $|\psi_1\rangle \in \mathcal{H}_k$, $|\psi_2\rangle \in \mathcal{H}_{k+1}$ (see Subsection 3.4.1). Starting with the uniform state in Eq. (3.19), we copy the basis states to create the maximally entangled state $|\tilde{\boldsymbol{s}}\rangle_{12} = \frac{1}{\sqrt{\boldsymbol{N}}}\sum_{a=1}^{\boldsymbol{N}} |e_a\rangle_1\,|e_a\rangle_2$. Since $B_k^{\epsilon,\epsilon'}[\xi]$ is self-adjoint, we can find an orthonomral basis Λ of eigenvectors $|\lambda\rangle$ and rewrite $|\tilde{\boldsymbol{s}}\rangle_{12} = \frac{1}{\sqrt{\boldsymbol{N}}}\sum_{|\lambda\rangle\in\Lambda} |\lambda\rangle_1\,|\lambda\rangle_2$.

Next, we introduce a register of qubits R in the state $|R\rangle_R = \frac{1}{\sqrt{M}}\sum_{y=0}^{M-1} |y\rangle_R$ and entangle the registers 2 and R by applying U_B, a unitary version of $B_k^{\epsilon,\epsilon'}[\xi]$ given by

$$U_B = e^{2\pi i l y B_k^{\epsilon,\epsilon'}[\xi]/M}, \tag{3.14}$$

where M and l are positive integers that can be adjusted at will. We obtain the quantum state $U_B\,|\tilde{\boldsymbol{s}}\rangle_{12}\,|R\rangle_R = \frac{1}{\sqrt{\boldsymbol{N}M}}\sum_{|\lambda\rangle\in\Lambda}\sum_{y=0}^{M-1} e^{2\pi i l y\lambda/M}\,|\lambda\rangle_1\,|\lambda\rangle_2\,|y\rangle_R$. For details of the implementation of Eq. (3.14), see Subsection 3.4.1.

Finally, we perform the quantum Fourier transform $U_{QFT}\,|y\rangle = \frac{1}{\sqrt{M}}\sum_{p=0}^{M-1} e^{-2\pi i p y/M}\,|p\rangle$ on the register R, and the state of the system becomes

$$U_{QFT}U_B\,|\tilde{\boldsymbol{s}}\rangle_{12}\,|R\rangle_R = \frac{1}{M\sqrt{\boldsymbol{N}}}\sum_{\lambda\in\Lambda}\sum_{y=0}^{M-1}\sum_{p=0}^{M-1} e^{2\pi i(l\lambda-p)y/M}\,|\lambda\rangle_1\,|\lambda\rangle_2\,|p\rangle_R\,. \tag{3.15}$$

Notice that we can sum over y to rewrite the coefficients in Eq. (3.15) as $\dfrac{e^{2\pi i l\lambda}-1}{e^{2\pi i(l\lambda-p)/M}-1}$. Moreover, these coefficients are strongly peaked at $p \approx l\lambda$, and at the peaks, the coefficients are approximately equal to M. Therefore, $U_{QFT}U_B\,|\tilde{\boldsymbol{s}}\rangle_{12}\,|R\rangle_R \approx \frac{1}{\sqrt{\boldsymbol{N}}}\sum_{|\lambda\rangle\in\Lambda} |\lambda\rangle_1\,|\lambda\rangle_2\,|l\lambda\rangle_R$.

A measurement of the register R yields p with probability $\mathcal{P}(p) = \|_R \langle p| U_{QFT} U_B |\tilde{\boldsymbol{s}}\rangle_{12} |R\rangle_R\|^2$ which can be written as

$$\mathcal{P}(p) = \frac{1}{\boldsymbol{N}} \sum_{\lambda} g_\lambda(p) \ , \ \text{with} \quad g_\lambda(p) = \frac{1}{M^2} \frac{\sin^2 \pi l \lambda}{\sin^2 \frac{\pi(l\lambda - p)}{M}} \ . \tag{3.16}$$

Approximately, the probability $\mathcal{P}(p)$ vanishes for all p, except at $p \approx l\lambda$. At the peaks, each eigenvalue contributes $1/\boldsymbol{N}$. Therefore, each peak is approximately proportional to the multiplicity of the corresponding eigenvalue. In particular, for $p = l\xi$, which corresponds to the eigenvalue of interest $\lambda = \xi$, the probability in Eq. (3.16) becomes

$$\mathcal{P}(l) = \frac{\beta_k^{\epsilon,\epsilon'}}{\boldsymbol{N}} + \frac{1}{M^2 \boldsymbol{N}} \sum_{\lambda \neq \xi} \frac{\sin^2 \pi l \lambda}{\sin^2 \frac{\pi l(\lambda - 1)}{M}}. \tag{3.17}$$

In the limit of Eq. (3.17) as $M \to \infty$, the sum over eigenvalues $\lambda \neq \xi$ vanishes, and we obtain $\beta_k^{\epsilon,\epsilon'} = \boldsymbol{N}\mathcal{P}(l\xi)$.

If one is interested in the whole spectrum of the persistent combinatorial Laplacian, M and l must be chosen in a way that the positive spectrum of the shifted Dirac operator is covered with the different values of p. Notice that each eigenvalue is approximated as $\lambda \approx \frac{p}{l}$. If we are only interested in the harmonic spectra of the persistent combinatorial Laplacian, we only need to capture the peak at $p \approx l\xi$ and make sure it is resolved from other neighboring peaks.

3.4.1 Implementation of an exponential operator

Relying on [87], we review the construction of the exponential operator $e^{itB_k^{\epsilon,\epsilon'}[\xi]}$, where the shifted Dirac matrix $B_k^{\epsilon,\epsilon'}[\xi]$ is defined in Eq. (3.11). This construction is needed for the phase estimation algorithm described in Section 3.4 (See Eq. (3.14)). We start by constructing the SWAP_B operator from the shifted Dirac operator $B_k^{\epsilon,\epsilon'}[\xi]$,

$$\mathcal{S} \equiv \text{SWAP}_B = \sum_{\psi,\phi} B_k^{\epsilon,\epsilon'}[\xi](\psi,\phi) \, |\phi\rangle \langle\psi| \otimes |\psi\rangle \langle\phi| \ , \quad B_k^{\epsilon,\epsilon'}[\xi](\psi,\phi) = \langle\psi| \, B_k^{\epsilon,\epsilon'}[\xi] \, |\phi\rangle \ , \tag{3.18}$$

where $|\psi\rangle = |0\rangle |\psi_0\rangle + |1\rangle |\psi_1\rangle + |2\rangle |\psi_2\rangle$, with $|\psi_0\rangle \in \mathcal{H}_{k-1}$, $|\psi_1\rangle \in \mathcal{H}_k$, $|\psi_2\rangle \in \mathcal{H}_{k+1}$, and similarly for $|\phi\rangle$. Let $\boldsymbol{N}$ be the dimensionality of the Hilbert space in which $|\psi\rangle$ and $|\phi\rangle$ live and $\{|e_a\rangle, a = 1, \ldots, \boldsymbol{N}\}$ an orthonormal basis for the Hilbert space under consideration. With the choice $\xi = 1$, all matrix elements of the $\boldsymbol{N} \times \boldsymbol{N}$ matrix $B_k^{\epsilon,\epsilon'}[\xi](e_a, e_b) \in \{0, \pm 1\}$ and the matrix $\mathcal{S}$ can be efficiently constructed. Then we construct the exponential SWAP_B operator $e^{i\Delta t \mathcal{S}}$, which can be done efficiently because $\mathcal{S}$ is a one-sparse matrix. Next, we act on the state $|\boldsymbol{s}\rangle \otimes |\Psi\rangle$, where $|\boldsymbol{s}\rangle$ is the uniform state

$$|\boldsymbol{s}\rangle = \frac{1}{\sqrt{\boldsymbol{N}}} \sum_{a=1}^{\boldsymbol{N}} |e_a\rangle , \tag{3.19}$$

and $|\Psi\rangle$ is an arbitrary state in the subspace on which $B_k^{\epsilon,\epsilon'}[\xi]$ acts. After tracing over the space in which $|\boldsymbol{s}\rangle$ lives, we obtain $\mathrm{tr}_1 \left[e^{-i\Delta t \mathcal{S}} |\boldsymbol{s}\rangle \langle \boldsymbol{s}| \otimes |\Psi\rangle \langle \Psi| e^{i\Delta t \mathcal{S}} \right] = e^{-iB_k^{\epsilon,\epsilon'}[\xi]\Delta t/\boldsymbol{N}} |\Psi\rangle \langle \Psi| e^{iB_k^{\epsilon,\epsilon'}[\xi]\Delta t/\boldsymbol{N}} + \mathcal{O}(\Delta t^2)$, which projects onto the state $e^{-iB_k^{\epsilon,\epsilon'}[\xi]\Delta t/\boldsymbol{N}} |\Psi\rangle$ up to second order in Δt. The desired state $e^{-itB_k^{\epsilon,\epsilon'}[\xi]} |\Psi\rangle$ for finite t can be obtained by repeating the above construction as many times as needed.

3.5 An application

This Section demonstrates how our algorithm computes persistence Betti numbers, which track topological features across different scales. This extends previous work in [55] and [56] where the proposed quantum algorithms calculated Betti numbers only without addressing persistence features. To do this, we apply our method to the data set suggested in [88] and described below for the sake of completeness.

Consider a point cloud of 8 points consisting of two well-separated squares, as in Figure 3.1a. The smaller square has sides of length 1, while the larger square has sides of length $\sqrt{2}$. The distance between the two squares exceeds 2. It is easy to see that the smaller square produces a loop in the VR complex at scale 1 (Figure 3.1b) which disappears at scale $\sqrt{2}$, while at the same time the larger square produces a new loop (Figure 3.1c). It follows that for $1 < \epsilon_1 < \sqrt{2} < \epsilon_2 < 2$, the one-dimensional persistent Betti numbers

corresponding to the point cloud of Figure 3.1 are

$$\beta_1^{\epsilon_1,\epsilon_1} = 1 \ , \ \beta_1^{\epsilon_2,\epsilon_2} = 1 \ , \ \beta_1^{\epsilon_1,\epsilon_2} = 0 \ . \tag{3.20}$$

It should be pointed out that the algorithms proposed in [55, 56] can detect the number of loops in the VR complex at scales ϵ_1 and ϵ_2 by computing $\beta_1^{\epsilon_1,\epsilon_1}$ and $\beta_1^{\epsilon_2,\epsilon_2}$, respectively. However these Betti numbers do not hold any persistence information. The algorithms in [55, 56] cannot calculate the persistence Betti number $\beta_1^{\epsilon_1,\epsilon_2}$ which holds the persistence information (number of loops present at scale ϵ_1 that persist to scale ϵ_2). Thus, these algorithms cannot track topological features across different scales. Moreover, even though $\beta_1^{\epsilon_1,\epsilon_1} = \beta_1^{\epsilon_2,\epsilon_2}$, the persistence Betti number cannot be deduced from $\beta_1^{\epsilon_1,\epsilon_1}$ and $\beta_1^{\epsilon_2,\epsilon_2}$. This is because it indicates that the loops at scales across $\sqrt{2}$ are different, which is additional information to the existence of a loop encoded in the other two Betti numbers.

Figure 3.3 shows the quantum circuit for the calculation of Betti number $\beta_1^{\epsilon_1,\epsilon_2}$. The Dirac operator $B_1^{\epsilon_1,\epsilon_2}[\xi]$ acts on an 12-dimensional Hilbert space. For $\xi = 1$, the eigenvalues are $\lambda = -1, \pm\sqrt{5}$, each of degeneracy 4. Notice that 1 is not an eigenvalue, therefore $\beta_1^{\epsilon_1,\epsilon_2} = 0$. This is confirmed by the probabilities from the quantum algorithm depicted in Figure 3.2 as explained below.

A register of 4 qubits initially in the state $|0000\rangle_1$ is brought into the state $\frac{1}{4}\sum_{x=0}^{15} |x\rangle_1$, by acting with the Hadamard matrix on each qubit. Then we use them as control to apply CNOT on each of 4 qubits in an additional register in the state $|0000\rangle_2$, thus entangling them to the state $\frac{1}{4}\sum_{x=0}^{15} |x\rangle_1 |x\rangle_2$. We introduce a third register of 4 qubits (choosing $M = 16$) in the state $|0000\rangle_R$ and act on each with the Hadamard gate to bring them into the state $\frac{1}{4}\sum_{y=0}^{15} |y\rangle_R$. We then use them as control to act on register 2 with the exponential of the shifted Dirac operator (see Section 3.4 for details). Finally, we measure all 4 qubits in the register R. The result is the probability distribution $\mathcal{P}(p)$, where $p = 0, 1, \ldots, 15$ (Eq. (3.16)) depicted in Figure 3.2c. With the choice $l = 3$, $M = 16$, a measurement of the register of 4 qubits yields no peak at $p = 3$, showing that $\beta_1^{\epsilon_1,\epsilon_2} = 0$.

The persistent Betti number $\beta_1^{\epsilon_1,\epsilon_1}$ is calculated using an eight-dimensional Hilbert space. The eigenvalues of the persistent Dirac operator for $\xi = 1$ are $\pm 1, \pm\sqrt{3}, \pm\sqrt{5}$. Two of the

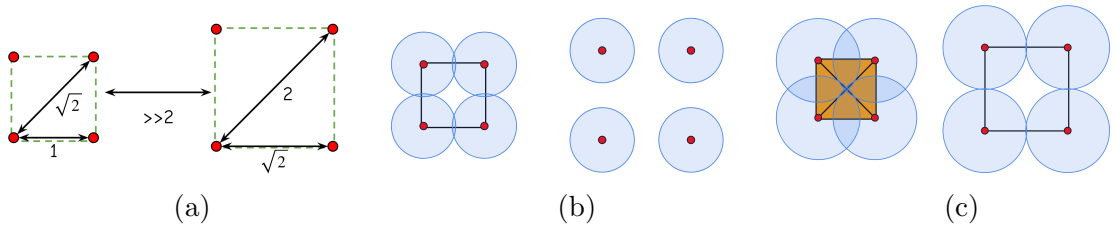

Figure 3.1: (a) A two square point cloud data set with the same characteristics as featured in [88]; Construction of the VR at two different scales. Precisely, (b) at ϵ_1; and (c) at $\epsilon_2 > \epsilon_1$.

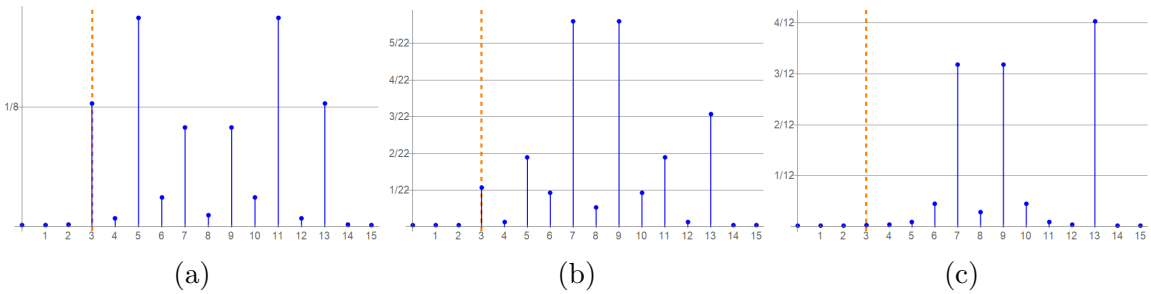

Figure 3.2: Probability density corresponding to the persistent Dirac operator (a) $B_1^{\epsilon_1,\epsilon_1}[\xi]$, (b) $B_1^{\epsilon_2,\epsilon_2}[\xi]$, and (c) $B_1^{\epsilon_1,\epsilon_2}[\xi]$, with $\xi = 1$, $l = 3$, $M = 16$. The heights at $p = 3$, multiplied by the dimension of the Hilbert space (8, 22, and 12, respectively), yield the Betti numbers $\beta_1^{\epsilon_1,\epsilon_1} = \beta_1^{\epsilon_2,\epsilon_2} = 1$, and the persistent Betti number $\beta_1^{\epsilon_1,\epsilon_2} = 0$.

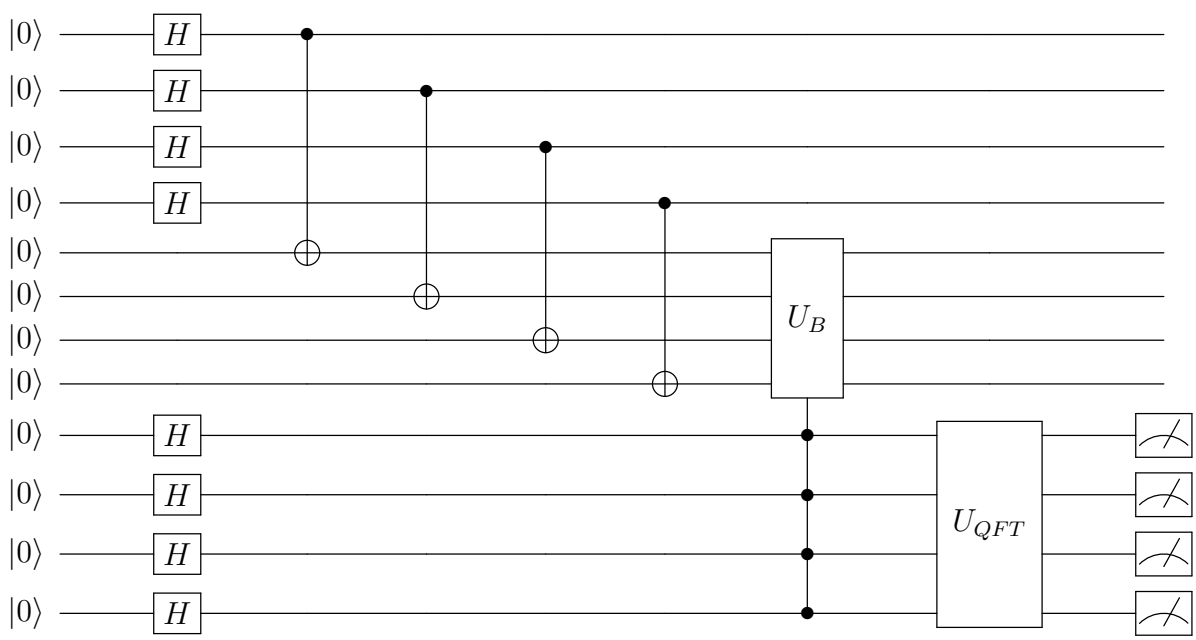

Figure 3.3: Quantum circuit for the calculation of Betti number $\beta_1^{\epsilon_1,\epsilon_2}$.

eigenvalues ($\pm\sqrt{3}$) are degenerate with multiplicity 2. We are interested in the multiplicity of the eigenvalue $\lambda = \xi = 1$, which is shown to be 1 by the peak at $p = 3$ of height 1/8 (see Fig.3.2a). The closest eigenvalue to $\lambda = 1$ is $\lambda = \sqrt{3}$ which is near $p = 5$. Thus, the peak at the nearest eigenvalue is well separated from the one of interest (in Figure 3.2a), one can see a dip at $p = 4$, and the height $\mathcal{P}(5) \approx \frac{2}{8}$, confirming the double degeneracy of the eigenvalue $\lambda = \sqrt{3}$. The quantum circuit flows as the one in Figure 3.3 but with registers 1 and 2 having 3 qubits each.

The calculation of Betti number $\beta_1^{\epsilon_2,\epsilon_2}$ proceeds similarly. The Hilbert space is 22-dimensional and for $\xi = 1$, the eigenvalues are the same as for $\beta_1^{\epsilon_1,\epsilon_1}$, but with degeneracies 1, 3, 2, 7 for $\lambda = 1, -1, \pm\sqrt{3}, \pm\sqrt{5}$, respectively. The distance from the eigenvalue of interest ($\lambda = 1$) to its closest one ($\lambda = \sqrt{3}$) is same as before, therefore we can choose $l = 3, M = 16$, again. The quantum circuit is similar to the one in Figure 3.3 except that registers 1 and 2 need 5 qubits each. The resulting probability distribution is depicted in Figure 3.2b showing that $\beta_1^{\epsilon_2,\epsilon_2} = 1$.

Chapter 4

Quantum Persistent Homology for Time Series

My next contribution extends the quantum algorithm described in Chapter 3 as well as similar ones [61, 62] to consider time series data set by defining a membership oracle based on Takens' delay embedding (see Section 2.4). This work was published in [63].

4.1 Encoding Data and Simplices

Consider a discrete time series x_t, $t = 1, 2 \ldots, T$, such that $T > 0$ is an integer. The data are stored in a QRAM, allowing for their quantum parallel access. The QRAM maps a quantum state $|t\rangle |0\rangle$ to the state $|t\rangle |x_t\rangle$ where $|x_t\rangle$ is a state normalized to reflect the value x_t. Such states can be encoded using $O(\log_2 T)$ qubits.

Recall that the point cloud relates to the time series via the vectors v_i given in Eq. (2.8). We use $T - \tau d$ qubits to encode these points so that any simplex σ formed with these points is represented by a quantum state $|\sigma\rangle$, with the qubits corresponding to points in the simplex in state $|1\rangle$, and all other qubits in state $|0\rangle$. A balanced superposition of the states represents the maximal simplicial complex S. Define $\mathbb{Z}_3 = \{-1, 0, 1\}$. Using these quantum states as a basis of the generated Hilbert space over $\mathbb{Z}_3$ encodes the chain complex needed to extract the desired topological features.

4.2 Quantum Delay Embedding

Considering the time series data x_t, along with the embedding parameters, the dimension, d, of the space to which the times series is embedded, and the delay parameter, τ, one may check for membership in a simplicial complex by first computing the distances between points v_i in the embedded point cloud, defined in Eq. (2.8). The distance between two points v_i and v_j of the delay embedded time series is given by

$$D(v_i, v_j) = \max_{0 \leq t < d} |x_{i+t\tau} - x_{j+t\tau}|. \tag{4.1}$$

With the ability to create quantum states encoding the values x_t, we can construct a quantum circuit that takes input states of the form $|t_1\rangle |t_2\rangle |0\rangle$ and returns the output state

$|t_1\rangle\,|t_2\rangle\,||x_{t_1}-x_{t_2}|^2\rangle$, where the last register contains an estimate of the difference between x_{t_1} and x_{t_2}. To obtain an estimate with accuracy δ it is necessary to make $O(\delta^{-1})$ calls to the QRAM and $O\left(\delta^{-1}(\log_2 T)^2\right)$ quantum operations [89, 55]. Moreover, this circuit can also operate in quantum parallel.

Notice that a simplex σ is in the VR complex S^ϵ if and only if for each pair of vertices v_i and v_j in the simplex, $D(v_i, v_j) \leq \epsilon$, where the distance, D, is defined in Eq. (4.1). Being able to compute the difference $|x_{t_1} - x_{t_2}|$ allows us to create an oracle $\mathcal{O}^\epsilon$ that verifies if the difference $|x_{t_1} - x_{t_2}| \leq \epsilon$. Such an oracle takes as input the state $|t_1\rangle\,|t_2\rangle\,|1\rangle$ and returns the state $|t_1\rangle\,|t_2\rangle\,|a^\epsilon\rangle$, where

$$a^\epsilon = \begin{cases} 1 & , \quad |x_{t_1} - x_{t_2}| \leq \epsilon \\ 0 & , \quad |x_{t_1} - x_{t_2}| > \epsilon \end{cases} . \tag{4.2}$$

Given a simplex σ encoded as the quantum state $|\sigma\rangle$, where the qubits in state $|1\rangle$ correspond to the vertices of σ, we can make repeated calls to the oracle $\mathcal{O}^\epsilon$ to check if the simplex belongs to the VR complex at scale ϵ. In particular, we need to verify the inequality for all $t_1 = i + t\tau$ and $t_2 = j + t\tau$ with $0 \leq t < d$ and $i < j$ such that v_i and v_j are vertices of σ, where d and τ as in Eq. (2.8) must be chosen according to the data. The vertices v_i, v_j are obtained from the quantum state $|\sigma\rangle$, and they correspond to the qubits in state $|1\rangle$. One concludes that σ is in S^ϵ if all the calls to the oracle $\mathcal{O}^\epsilon$ yield an output with the last qubit in state $|1\rangle$. However, if at least one of the calls returns an output with the last qubit in state $|0\rangle$, it means the simplex is not yet present at that scale.

To that end, for fixed d and τ, a membership oracle $\mathcal{O}^\epsilon_{d,\tau}$ is constructed such that it acts on a quantum state $|\sigma\rangle\,|1\rangle$ according to

$$\mathcal{O}^\epsilon_{d,\tau}\,|\sigma\rangle\,|1\rangle = \begin{cases} |\sigma\rangle\,|1\rangle & , \quad \sigma \in S^\epsilon \\ |\sigma\rangle\,|0\rangle & , \quad \sigma \notin S^\epsilon \end{cases} . \tag{4.3}$$

The membership oracle, $\mathcal{O}^\epsilon_{d,\tau}$, makes at most $dk(k+1)/2$ calls to the oracle $\mathcal{O}^\epsilon$ in order to determine whether a simplex σ of dimension k is present in the VR complex at scale ϵ.

4.3 Quantum Algorithm for Time Series

Having established a membership oracle $\mathcal{O}^{\epsilon}_{d,\tau}$, defined in Eq. (4.3), allows to consider a quantum persistent homology algorithm for time series. First, one uses $T - \tau d$ qubits to represent the point cloud points, v_i, defined in Eq. (2.8), which result from the embedding. Then, a simplex σ is represented by the quantum state $|\sigma\rangle$, where the qubits corresponding to the vertices of σ are in state $|1\rangle$, and all others are in state $|0\rangle$. The boundary map ∂ is essential to extract the topological features and it can be encoded using Pauli X operators, which are switches that change state $|0\rangle$ into state $|1\rangle$ and vice versa. The boundary operator given in Eq. (3.2) maps a simplex $|\sigma_k\rangle$ of dimension k into a superposition of the $(k-1)$-simplices that conform its boundary, which are none other than the simplices obtained by removing each of the vertices in σ_k.

Given two fixed scales $\epsilon \leq \epsilon'$, the membership oracle in Eq. (4.3) is used to implement the projection operators, P^{ϵ} and $P^{\epsilon'}$, onto the subspaces that encode the VR complexes S^{ϵ} and $S^{\epsilon'}$. The projections are in turn used to build a persistent Dirac operator as in Section 3.3. In order to obtain all persistent Betti numbers at various scales and represent them onto persistence diagrams, one needs to consider an increasing sequence of scales $\epsilon_0 < \cdots < \epsilon_n$, chosen according to the data. Then one implements the aforementioned quantum methodology to obtain the persistent Betti numbers $\beta_k^{\epsilon_i,\epsilon_j}$ for each pair ϵ_i, ϵ_j with $0 \leq i \leq j \leq n$. Since the estimations are independent, this can be done in parallel. Finally, once we have all possible Betti numbers, and the scales, which appear and disappear, one may depict them onto persistence diagrams. Indeed, if $\mu_k^{\epsilon,\epsilon'}$ denotes the number of $k-$dimensional holes that appear in scale ϵ and disappear at scale ϵ', we have $\mu_k^{\epsilon_i,\epsilon_j} = \beta_k^{\epsilon_i,\epsilon_{j-1}} - \beta_k^{\epsilon_i,\epsilon_j} - \left(\beta_k^{\epsilon_{i-1},\epsilon_{j-1}} - \beta_k^{\epsilon_{i-1},\epsilon_j}\right)$.

4.4 Results

We consider two time series and use the methodology described above to discover their shape properties. The first data is a simulated periodic discrete time series, and the second is a segment from an electroencephalography (EEG) measurement of a brain taken while the subject listened to music [90]. In both cases the task is to employ the quantum framework of Section 4.3 by first embedding the time series into a point cloud using the quantum delay embedding, and then examining the topological features of the corresponding embedded point clouds using the quantum persistent homology algorithm. Precisely, we simulate the quantum states and membership oracle Eq. (4.3) to construct a matrix representation of the persistent Dirac operator introduced in Section 3.3. Then we obtain the eigenvalues of this matrix as in Section 3.4 to estimate the persistent Betti numbers.

4.4.1 Periodic time series example

Consider the discrete time series based on the periodic function $\sin(2\pi t)$. Using a delay $d = 2$, $\tau = 1$, the time series is embedded into the 2-dimensional point cloud as shown in Fig. 4.1. Notice that this function has one maximum (or minimum) per period, which translates to one hole in the corresponding embedded point cloud, constructed by the quantum Taken's delay embedding.

To construct the persistence diagram, an increasing sequence of twenty-five different scales starting at $\epsilon_0 = 0.0 < \epsilon_1 = 0.1 < \cdots < \epsilon_{24} = 2.4$ with step size 0.1 is taken into account. Using the quantum persistent homology algorithm, the persistent Betti numbers of dimensions 0 and 1 for each possible pair of scales are computed, and depicted in Fig. 4.3. As expected (see Fig. 4.2), a single one dimensional hole exists due to periodicity, and this is born at scale 1 and died at scale 2. On the other hand, the number of connected components, which is initially four, is reduced to one when the one dimensional hole is formed at scale 1.

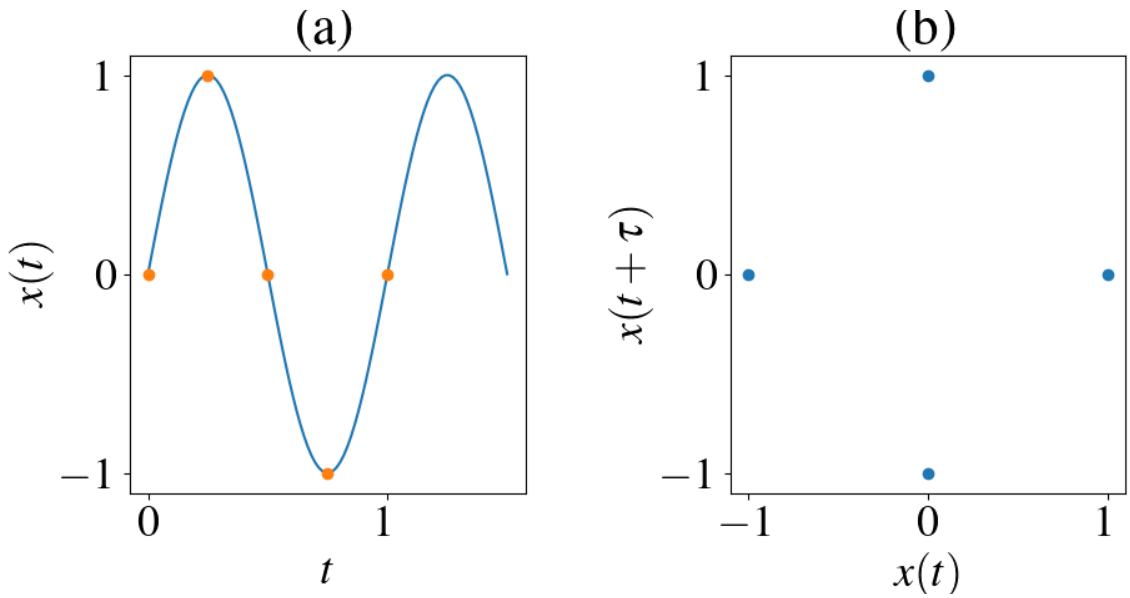

Figure 4.1: (a) The graph of $\sin(2\pi t)$ (blue line) along with a discrete time series (orange dots) given by $t = 0, 1/4, 1/2, 3/4, 1$. (b) The point cloud obtained by Takens's delay embedding using $\tau = 1$ and $d = 2$.

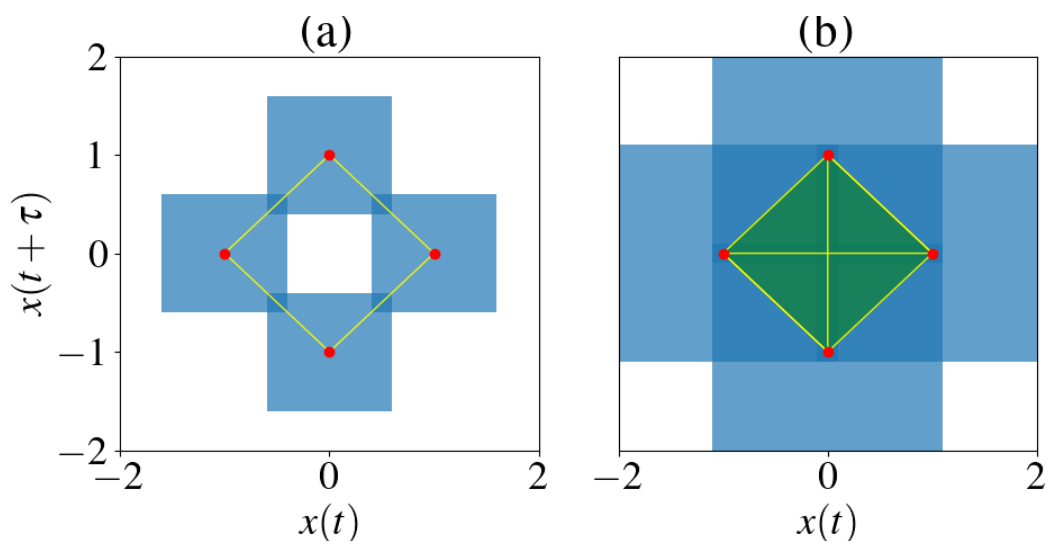

Figure 4.2: Vietoris Rips complexes of the embedded point cloud associated with the sinusodial signal of Fig. 4.1 at scales (a) $\epsilon_1 = 1.2$, the points are pairwise connected, which results in a hole in the middle; and (b) $\epsilon_2 = 2.2$, the points become totally connected and the hole disappears. The blue squares are the balls of diameter ϵ_i around each point.

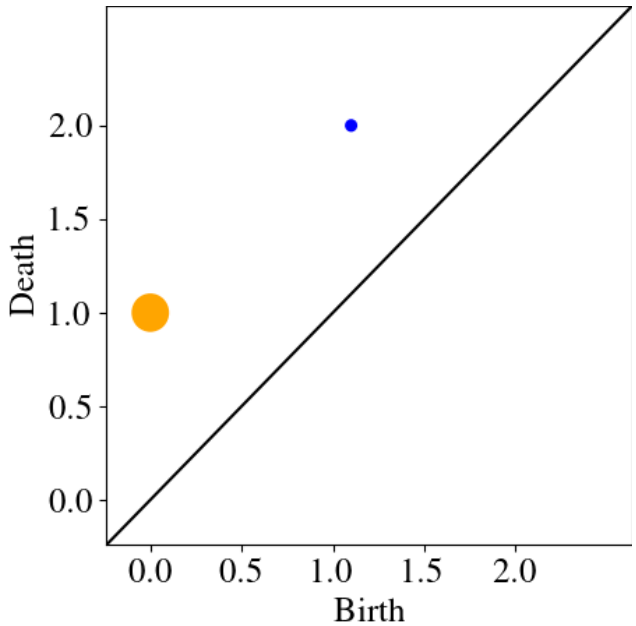

Figure 4.3: Persistence diagram for the time series in Fig 4.1. The horizontal axis marks the scales at which the topological features are born, while the vertical axis marks the scales at which they disappear. The size of the dots represents the number of features that appear and disappear at the same scales. Finally, the orange dots along the vertical axis represent the connected components, while the blue dots closer to the diagonal line are the one dimensional holes.

4.4.2 An electroencephalogram example

A segment of fifty measurements from an electroencephalogram (EEG) signal while the subject listened to music is considered in Fig. 4.4.

The point cloud obtained using the quantum delay embedding algorithm for a delay $\tau = 8$ and $d = 2$ is shown in Fig. 4.4. Next, the persistence diagrams are constructed using the quantum persistent homology algorithm for an increasing sequence of scales $\epsilon_k = k$, for $k = 0, \ldots, 15$. The persistence diagram of dimensions 0 (connected components) and 1 (holes) are depicted in Fig. 4.5. In this case one may see a number of small one dimensional holes that appear and disappear very quickly indicating that the segment of the signal is not periodic. The number of connected components is reduced quickly as shown by the large circles near the origin. Nevertheless, a few survive past scale 6 which indicates the presence of clusters in Fig. 4.4 (b).

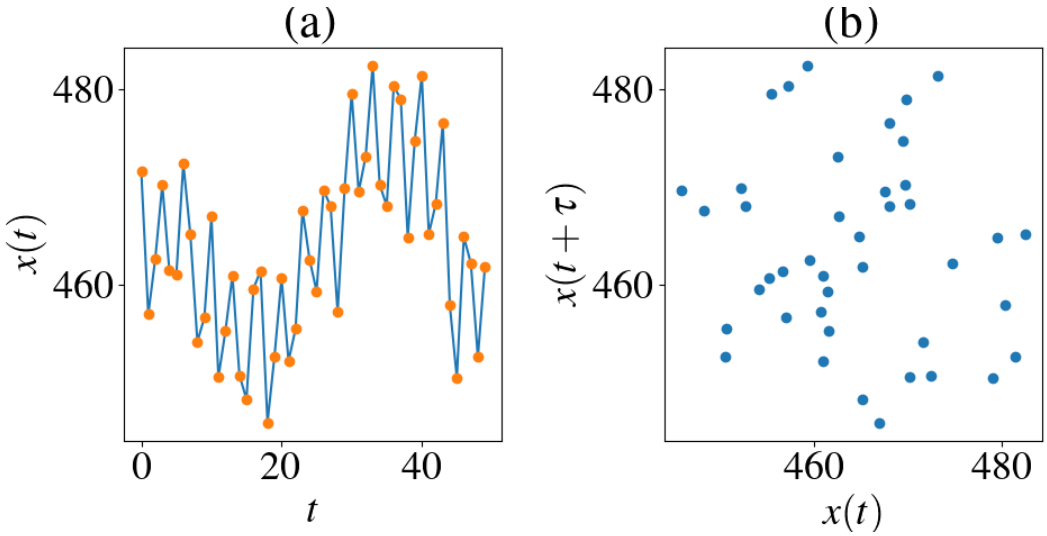

Figure 4.4: (a) A segment of an EEG signal measured while the subject listened to music [90]. (b) The point cloud obtained by the Takens's delay embedding using $\tau = 8$ and $d = 2$.

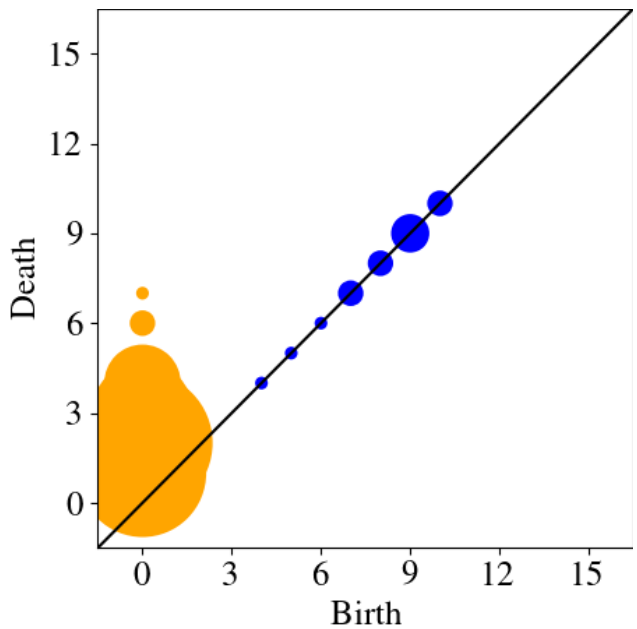

Figure 4.5: Persistence diagram for the time series in Fig 4.4. The horizontal axis states the scales at which the topological features are born, while the vertical axis marks the scales at which they disappear. The size of the dots represents the number of features that appear and disappear at the same scales. Finally, the orange dots along the vertical axis represent the connected components, while the blue dots closer to the diagonal line are the one dimensional holes.

Chapter 5

Quantum Distance Approximation for Persistence Diagrams

Finally, I formulate combinatorial optimization problems over binary variables to compute the distances between persistence diagrams defined in Section 2.5, and then construct quantum algorithms to estimate the solution to the optimization problems, which provides a quantum approach to compare persistence diagrams. This result is published in [69].

5.1 Quantum Algorithm for Wasserstein Distance

We begin by creating an optimization problem that encodes the Wasserstein distance between persistence diagrams introduced in Def. 2.8. Let $\mathcal{D}_1, \mathcal{D}_2$ be persistence diagrams and consider a weighted graph $G = (V, E)$. The vertex set V consists of the points in $x_i \in \mathcal{D}_1$ and $y_j \in \mathcal{D}_2$ along with an auxiliary vertex $\tilde{x}_i$ or $\tilde{y}_j$ for each point. These auxiliary vertices represent the projection of a point to the diagonal of the diagram and are meant to penalize unmatched points when the diagrams have different cardinality. The edge set E on the other hand is formed by all the possible edges (x_i, y_j) between points $x_i \in \mathcal{D}_1$ and $y_j \in \mathcal{D}_2$, as well as the edges between each point and their auxiliary vertex $(x_i, \tilde{x}_i)$ or $(\tilde{y}_j, y_j)$. A *main edge* is defined to be an edge connecting some $x_i \in \mathcal{D}_1$ and $y_j \in \mathcal{D}_2$. The main edges produce a complete bipartite graph between the points in the persistence diagrams with leaves to vertices that represent the closest point on the diagonal to each point. See Fig. 5.1 for a visual representation of this graph.

Consider the optimization problem over binary variables

$$\text{minimize} \quad C = \sum_{e \in E} w_e \delta_e \tag{5.1}$$

$$\text{subject to} \quad \sum_{e \sim v} \delta_e = 1 \ , \quad \forall v \in \mathcal{D}_1, \mathcal{D}_2 \tag{5.2}$$

where $\delta_e \in \{0, 1\}$ is 1 if edge e is present in the matching and 0 if it is not. The weight of edge e is represented by w_e, and $e \sim v$ denotes the edges incident to vertex v.

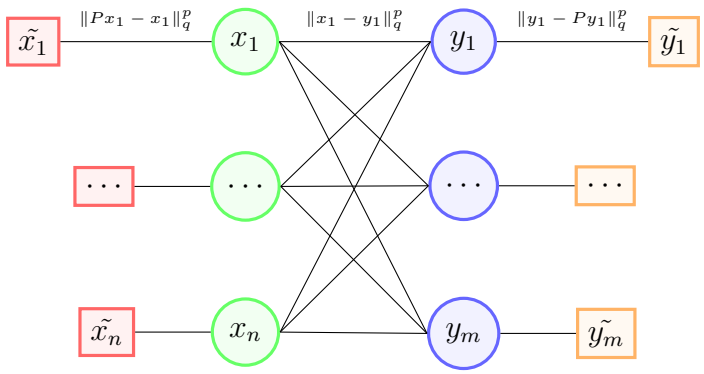

Figure 5.1: Graph connecting the points of two persistence diagrams.

Lemma 5.0.1. *The solution to the optimization problem as defined in Eqs.* (5.1) *and* (5.2) *using the weights*

$$w_e = \begin{cases} \|x_i - y_j\|_q^p & e = (x_i, y_j) \\ \|x_i - Px_i\|_q^p & e = (x_i, \tilde{x}_i) \\ \|Py_j - y_j\|_q^p & e = (\tilde{y}_j, y_j) \end{cases} \tag{5.3}$$

is the distance $d_p^W(\mathcal{D}_1, \mathcal{D}_2)$ *introduced in Def. 2.8. Here,* $Px = \frac{1}{2}(a+b, a+b)^T$ *denotes the projection to the diagonal of* $x = (a, b)^T$.

Proof. The cost function in Eq. (5.1) represents the sum in Def. 2.8, while the constraints in Eq. (5.2) ensure that each vertex is matched exactly once in order to encode the bijections between $\tilde{\mathcal{D}_\infty}$ and $\tilde{\mathcal{D}_\in}$. Notice that the auxiliary vertices represent the projection of each point to the diagonal, and the weight that is assigned to the corresponding edges is simply the distance to the diagonal. For a choice of δ_e's that satisfies the constraints and with ϕ the bijection that corresponds to it, the cost in Eq. (5.1) is simply $\mathrm{Cost}[\phi] = \sum_{x \in \tilde{\mathcal{D}}_1} \|x - \phi(x)\|_q^p$. Since taking the minimum over the different solutions to the constraints, Eq. (5.2) is the same as taking the minimum over the bijections, and thus the result of the optimization problem must be d_p^W. □

Now we describe the quantum algorithm to compute the optimization problem within a quantum framework given in Eq. (5.1) with the constraints as in Eq. (5.2). We use qubits $|e\rangle$ to encode the edges $e \in E$ such that the state of $|e\rangle$ corresponds to the binary variable $\overline{\delta_e} = 1 + \delta_e \mod 2$. Thus a qubit $|e\rangle$ in state $|0\rangle$ means that edge e is present. In general we can use $n \times m$ qubits to identify the edges between points in the diagrams, along with $n + m$ more qubits for the edges to auxiliary vertices. The quantum states $\bigotimes_{e \in E} |e\rangle$ encode all the different sub-graphs of G. Throughout this paper, the quantum states are arranged as in Eq. (5.4), where each row will correspond to a point in $\mathcal{D}_1$ and each column to a point in $\mathcal{D}_2$. In particular this makes it easier to recover the respective matching.

$$\left|\begin{array}{cccc} (x_1, y_1) & \cdots & (x_1, y_m) & (x_1, \tilde{x}_1) \\ \vdots & \ddots & \vdots & \vdots \\ (x_n, y_1) & \cdots & (x_n, y_m) & (x_n, \tilde{x}_n) \\ (\tilde{y}_1, y_1) & \cdots & (\tilde{y}_m, y_m) & \end{array}\right\rangle \tag{5.4}$$

In order to construct a Hamiltonian that encodes the cost function in Eq. (5.1) we need to modify it to use the variables $\overline{\delta}_e$ that correspond to the states of the qubits. Notice that $C = \sum_{e\in E} w_e - \sum_{e\in E} w_e \overline{\delta_e}$, where w_e are the weights introduced in Eq. (5.3). Since the first term is constant, minimizing Cost$[\phi]$ is equivalent to maximizing $g = \sum_{e\in E} w_e \overline{\delta_e}$. So we may use the cost Hamiltonian $H_g = \frac{1}{2}\sum_{e\in E} w_e(I - Z_e)$ whose ground state maximizes g. The introduction of weights into the cost Hamiltonian H_g inspired by [81] is essential to capture the distances between points of the persistence diagrams. This introduction of weights can be recast as other QAOA variants, such as multi-angle QAOA [91]. This yields our unitary cost operator

$$U_g = e^{i\frac{\gamma}{2}\sum_{e\in E} w_e Z_e} = \prod_{e\in E} e^{i\frac{\gamma}{2} w_e Z_e} = \prod_{e\in E} R_{Z_e}(-\gamma w_e) \ , \tag{5.5}$$

where Z_e and $R_{Z_e}(-\gamma w_e)$ are the Pauli-Z operator and the single qubit rotation over the z-axis, respectively, acting on qubit $|e\rangle$. We omit the constant term $\frac{1}{2}\sum_{e\in E} w_e I$ in the Hamiltonian since it only adds a global phase $e^{-i\frac{1}{2}\sum_{e\in E} w_e}$ to the unitary operator.

To encode the constraints in Eq. (5.2), we use control clauses f as in [74], to build an individual mixing Hamiltonian $M_e = f(e)X_e$ for each edge, where X_e is the Pauli-X operator acting on qubit $|e\rangle$.

Definition 5.1. *We define the control clause f as the binary function on the set of edges E such that*

$$f(e) = \begin{cases} \overline{\delta_e} \prod_{l\neq i} \overline{\delta_{x_l,y_j}} \prod_{k\neq j} \overline{\delta_{x_i,y_k}} & e = (x_i, y_j) \\ \delta_e \overline{\prod_{k=1}^{|\mathcal{D}_2|} \overline{\delta_{x_i,y_k}}} & e = (x_i, \tilde{x}_i) \\ \delta_e \overline{\prod_{l=1}^{|\mathcal{D}_1|} \overline{\delta_{x_l,y_j}}} & e = (\tilde{y}_j, y_j) \end{cases} . \tag{5.6}$$

Notice that for an edge $e = (x_i, y_j)$, the control clause $f(e)$ will be 1 whenever both points x_i and y_j are unmatched. On the other hand, for an edge to an auxiliary vertex $e = (x_i, \tilde{x}_i)$ the control clause $f(e)$ will be 0 either when the point x_i is unmatched, or when its corresponding auxiliary vertex $\tilde{x}_i$ is unmatched. Similarly, for an edge $e = (\tilde{y}_j, y_j)$, $f(e)$ is 0 if either y_j or $\tilde{y}_j$ are unmatched. So this control clause captures the idea behind our constraints of matching each point exactly once.

We use the individual mixing Hamiltonians M_e to build an individual mixing unitary operator for each edge

$$U_{M,e}(\beta) = e^{-i\beta M_e} = \bigwedge_{f(e)} e^{-i\beta X_e} = \bigwedge_{f(e)} R_{X_e}(\beta), \tag{5.7}$$

and these individual unitaries are applied in succession to build the complete mixing unitary operator

$$U_M(\beta) = \prod_{e \in E} U_{M,e}(\beta) = \prod_{e \in E} e^{-i\beta M_e} = \prod_{e \in E} \bigwedge_{f(e)} R_{X_e}(\beta). \tag{5.8}$$

Note however that the order in which one applies the individual operators matters, we apply the operators corresponding to the main vertices first and those corresponding to auxiliary vertices last. Our goal is to create a mixing operator that produces only quantum states that satisfy the constraints of the problem. But the mixing unitary operator in Eq. (5.8) creates some quantum states that do not satisfy the constraints in Eq. (5.2), i.e. matchings that do not represent bijections. So, we must show that all of these problematic quantum states have no effect on the optimization process. Indeed, as stated in Thm. 5.1, we may relax the constraints of the problem so that any extra quantum states produced by our mixing unitary satisfy these new constraints. Moreover, matchings corresponding to extra quantum states are the result of adding edges from auxiliary vertices to the matchings that do represent bijections. In particular, their cost will be greater and thus the minimum will remain the same.

Theorem 5.1. *The mixing operator $U_M(\beta)$ preserves the feasibility of quantum states. That is, if the quantum state $|s\rangle$ represents a matching that satisfies the constraints*

$$\begin{aligned} \sum_{(i,j)\sim v} \delta_{(i,j)} \leq 1 \ , \quad \text{for all } v \in \mathcal{D}_1, \mathcal{D}_2 \\ \sum_{e\sim v} \delta_e \geq 1 \ , \quad \text{for all } v \in \mathcal{D}_1, \mathcal{D}_2 \end{aligned} \tag{5.9}$$

then $U_M(\beta)\,|s\rangle$ also satisfies these constraints. Moreover, the minimum of the cost function in Eq. (5.1) over these constraints is the same as the minimum over the constraints in Eq. (5.2).

Proof. We first give some intuition about these new constraints and show that the minimum remains the same. Notice that the first constraint asks that a point v in one of the persistence diagrams is matched at most once to points in the other diagram. The second constraint requires that every point in the persistence diagrams is matched at least once. This means that points in a persistence diagram $\mathcal{D}_i, i \in \{1, 2\}$ could be incident to more than one edge, but only one of the edges will connect it to a point in the other diagram and all other edges connect to auxiliary vertices. So the solutions to these constraints will be matchings that represent a bijection or matchings that result of adding connections to auxiliary vertices from a bijection. In particular, any solution to the constraints in Eq. (5.2) is also a solution to Eq. (5.9), and any solution to the latter that is not also a solution to the former must have a larger cost.

Now, given a quantum state $|s\rangle$ which satisfies the constraints in Eq. (5.9), we show that $U_{M,e}(\beta)\,|s\rangle$ also satisfies the constraints for arbitrary $e \in E$. This in turn proves that $U_M(\beta)$ preserves the feasibility of quantum states as it is defined as a product of the individual unitaries.

From Eq. (5.7) note that there are only two possible cases for $U_{M,e}(\beta)\,|s\rangle$ depending on the value of $f(e)$. First, if $f(e) = 0$ the output state is simply $|s\rangle$. On the other hand, when $f(e) = 1$ the resulting state is $\cos\beta I\,|s\rangle - i\sin\beta X_e\,|s\rangle$. The first term is just $|s\rangle$ scaled by a constant, so we only need to verify that the second term is feasible.

Notice that the second term X_e flips the qubit $|e\rangle$ but it is only applied when $f(e) = 1$. If $e = (x_i, y_j)$ is an edge between points in the diagrams, $f(e) = 1$ if and only if $\delta_e, \delta_{i,k}, \delta_{l,j} = 0$ for all $l \neq i$ and $k \neq j$, that is, the edge $e = (x_i, y_j)$ is added to the matching if there are no other main edges containing vertices x_i or y_j. Since this operation adds an edge, the second constraint in Eq. (5.9) is trivially satisfied, moreover the first constraint is satisfied because the edge is only added if that sum is equal to zero. On the other hand, if $e = (x_i, \tilde{x}_i)$ (or $(\tilde{y}_j, y_j)$) is an edge to an auxiliary vertex, $f(e) = 1$ whenever $\delta_e = 1$ and at least one of $\delta_{i,k}$ for $1 \leq k \leq |\mathcal{D}_2|$ (or $\delta_{l,j}$ for $1 \leq l \leq |\mathcal{D}_1|$) is non-zero, in other words, we remove the edge $e = (x_i, \tilde{x}_i)$ (or $(\tilde{y}_j, y_j)$) from the matching if there is a main edge connecting the corresponding vertex x_i (or y_j) to a vertex in the other diagram. So, the first constraint in Eq. (5.9) is unaffected and the second one is still satisfied as we only remove an edge if the corresponding sum is at least 2. All in all, $U_{M,e}(\beta)\,|s\rangle$ is a superposition of quantum states whose corresponding matchings satisfy the constraints in Eq. (5.9). □

Another important property of our mixing operator is that it produces every solution to the relaxed constraits in Eq. (5.9) in a single iteration. Using the mixing operator again will change the amplitudes of the quantum states but it will not create any new states. For this reason we also use the mixing operator to initialize the algorithm with a superposition of only the necessary quantum states to solve the optimization problem.

Theorem 5.2. *Applying each of the $n \times m + n + m$ unitaries $U_{M,e}(\beta)$ in sequence starting with the initial state*

$$\left|\begin{matrix} 1 & \cdots & 1 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 1 & \cdots & 1 & 0 \\ 0 & \cdots & 0 & \end{matrix}\right\rangle \tag{5.10}$$

yields a superposition of all possible matchings that satisfy Eq. (5.9) with non-zero amplitudes.

Proof. Notice that one may obtain all solutions to Eq. (5.9) by finding every possible choice for the main $n \times m$ edges that satisfy the first constraint, and then for each of these obtaining the different combinations of the $n+m$ auxiliary edges that satisfy the second constraint. So,

we first apply the unitaries corresponding to the main edges to produce the solutions to the first constraint while keeping all the auxiliary edges on. After which we use the remaining unitaries to get the combinations of auxiliary edges. Since every unitary keeps the quantum state it acts on, either whole or scaled by $\cos\beta$, this process yields all matchings that satisfy Eq. (5.9).

Let $e_1, \ldots, e_N$ be any ordering of the $n \times m$ main edges, we want to prove that applying their corresponding unitaries in sequence will produce a superposition of all solutions to the first constraint in Eq. (5.9). We proceed by induction, where we will prove that applying U_{M,e_k} to the superposition of all feasible matches using only edges $e_1, \ldots, e_{k-1}$, yields a superposition of all feasible matchings using only edges $e_1, \ldots, e_k$. Since the case for $k = 1$ starts with the quantum state in Eq. (5.10), $f(e_1)$ will always equal 1 and the result of applying U_{M,e_1} is a superposition of this trivial matching and the matching which includes only edge e_1. For $k > 1$, assume we have a superposition of all feasible matchings with edges $e_1, \ldots, e_{k-1}$ and notice that applying U_{M,e_k} to any one of these matchings can have two possible outcomes. Indeed if $f(e_k) = 0$ the matching remains unchanged. On the other hand, if $f(e_k) = 1$ the output will be a combination of the input matching and the matching that results from adding edge e_k. Therefore, U_{M,e_k} will retain all the matchings with only edges $e_1, \ldots, e_{k-1}$ and produce all feasible matchings that result from adding edge e_k, which yields a superposition over all feasible matchings that use only edges $e_1, \ldots, e_k$.

Now let $e'_1, \ldots, e'_{N'}$ be any ordering of the $n + m$ auxiliary edges. We prove in a similar way that given the superposition that results of the previous step, applying the auxiliary unitaries in sequence yields a superposition of all possible solutions to Eq. (5.9). Indeed, applying U_{M,e'_1} to each of the matchings in the superposition from the previous step gives either the same matching when $f(e'_1) = 0$ or a combination of the input matching and the one that results of removing edge e'_1 when $f(e'_1) = 1$, producing all feasible choices for e'_1. In the same manner, applying U_{M,e'_k} to the superposition with all feasible choices for edges $e'_1, \ldots, e'_{k-1}$ will produce a superposition with all feasible combinations for edges $e'_1, \ldots, e'_k$. $\square$

Remark 5.1. *This quantum algorithm uses $n \times m + n + m$ logical qubits to encode the edges and a few extra ancillary qubits for the controls. One must apply the individual mixing operator for each edge in sequence, which requires $n \times m + n + m$ multi-qubit controled rotation gates R_X, as well as $n \times m + n + m$ single qubit rotations R_Z for the cost operator. So the overall cost of the algorithm is $\mathcal{O}(nm)$ operations, which is better than $\mathcal{O}(n^2 m)$ operations needed by the Hungarian algorithm used to compute this distance classically.*

5.2 A More Efficient Distance

We proceed in a similar manner by describing the optimization problem and then building a quantum algorithm to solve it. Let $\mathcal{D}_1, \mathcal{D}_2$ be persistence diagrams such that their cardinalities satisfy the relationship, $|\mathcal{D}_1| \leq |\mathcal{D}_2|$. Consider a weighted graph $G = (V, E)$. This time the vertex set V consists of the points in $x_i \in \mathcal{D}_1$ and $y_j \in \mathcal{D}_2$ along with an auxiliary vertex $\tilde{y}_j$ for each point in $\mathcal{D}_2$. The auxiliary vertices still serve the purpose of penalizing any unmatched points due to a difference in cardinality, but now they also compute $\min(c, \|x_i - y_j\|_q)$. The edge set E on the other hand is formed by all the possible edges (x_i, y_j) between points $x_i \in \mathcal{D}_1$ and $y_j \in \mathcal{D}_2$, as well as the edges between each point in $\mathcal{D}_2$ and their auxiliary vertex $(\tilde{y}_j, y_j)$. This results in a smaller graph, which in turn will reduce the number of qubits and controls needed for the quantum algorithm. See Fig. 5.2 for a visual representation of this graph.

Now consider the following optimization problem over binary variables.

$$\text{minimize} \quad C = \sum_{e \in E} w_e \delta_e \tag{5.11}$$

$$\begin{aligned} \text{subject to} \quad & \sum_{e \sim v} \delta_e = 1 \ , \quad \forall v \in \mathcal{D}_2 \\ \text{and} \quad & \sum_{e \sim v} \delta_e \leq 1 \ , \quad \forall v \in \mathcal{D}_1 \end{aligned} \tag{5.12}$$

The first constraint ensures that every point in $\mathcal{D}_2$ is matched exactly once, either to a point in $\mathcal{D}_1$ or to its auxiliary vertex. On the other hand, the second constraint allows points

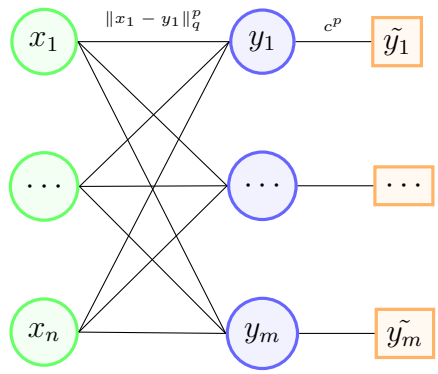

Figure 5.2: Graph connecting the points of two persistence diagrams.

in $\mathcal{D}_1$ to be matched once or to remain unmatched. These choices represent the different possible outcomes, c or $\|x_i - y_j\|$, of the corresponding minimum that appears in the sum at Def. 2.9. As before, we show that for a particular choice of weights, the solution to this optimization problem yields the distance $d_p^c(\mathcal{D}_1, \mathcal{D}_2)$.

Lemma 5.2.1. *The solution to the optimization problem described in Eqs.* (5.11) *and* (5.12) *using the weights*

$$w_e = \begin{cases} \|x_i - y_j\|_q^p & e = (x_i, y_j) \\ c^p & e = (\tilde{y}_j, y_j) \end{cases} \tag{5.13}$$

is the distance $d_p^c(\mathcal{D}_1, \mathcal{D}_2)$ introduced in Def. 2.9.

Proof. Notice that in order to compute the sum in Def. 2.9 one must choose for each point in $\mathcal{D}_2$ wether to match it to an unmatched point in $\mathcal{D}_1$ or penalize it with the parameter c. The auxiliary vertices serve the purpose of penalizing any unmatched points in $\mathcal{D}_2$. On the other hand, the constraints ensure that each point in $\mathcal{D}_2$ is matched exactly once, either to a point in $\mathcal{D}_1$ or to an auxiliary vertex, and that each point in $\mathcal{D}_1$ is matched at most once.

Given a solution to the constraints, an edge of the form $e = (x_i, y_j)$ with $\delta_e = 1$ represents matching vertices x_i and y_j with weight $\|x_i - y_j\|_q^p$. On the other hand, an edge of the form $e' = (\tilde{y}_j, y_j)$ represent the vertex y_j penalized with the weight c^p instead.

Then the cost in Eq. (5.11) will be the sum that appears in Def. 2.9 with either c or $\|x - \gamma(x)\|_q$ for each point x instead of the minimum. However, since this is a minimization problem, the algorithm will prefer the matching with the weight that corresponds to $\min(c, \|x - \gamma(x)\|_q)$. Thus, minimizing over all solutions to the constraints yields the distance d_p^c. □

We use again qubits $|e\rangle$ to encode the edges $e \in E$ of the graph. We still require $n \times m$ qubits to identify the main edges between points in the diagrams, but now we only need m more qubits for the edges to auxiliary vertices. The resulting quantum states $\bigotimes_{e \in E} |e\rangle$ encode all the different sub-graphs of G and are often arranged throughout this text as in Eq. (5.14) below,

$$\left| \begin{matrix} (x_1, y_1) & \cdots & (x_1, y_m) \\ \vdots & \ddots & \vdots \\ (x_n, y_1) & \cdots & (x_n, y_m) \\ (\tilde{y}_1, y_1) & \cdots & (\tilde{y}_m, y_m) \end{matrix} \right\rangle . \tag{5.14}$$

To construct a Hamiltonian for the cost function in Eq. (5.11), we must once more transform it into $g = \sum_{e \in E} w_e \overline{\delta_e}$ to avoid unnecessary gates. Note that the weights w_e are now the ones introduced in Eq. (5.13). This results in a unitary cost operator of the same form as in Eq. (5.5) for the Wasserstein distance, but with the edges and weights corresponding to the d_p^c distance. To encode the constraints in Eq. (5.12) we use similar control clauses to build individual mixing Hamiltonians $M_e = f(e) X_e$ for each edge.

Definition 5.2. *We define the control clause f as the binary function on the set of edges E such that*

$$f(e) = \begin{cases} \overline{\delta_e} \prod_{l \neq i} \overline{\delta_{x_l, y_j}} \prod_{k \neq j} \overline{\delta_{x_i, y_k}} & e = (x_i, y_j) \\ \delta_e \overline{\prod_{l=1}^{|\mathcal{D}_1|} \overline{\delta_{x_l, y_j}}} & e = (\tilde{y}_j, y_j) \end{cases} . \tag{5.15}$$

This control clause is similar to the one described in Def. 5.1, except we no longer have auxiliary vertices for the points in $\mathcal{D}_1$. The resulting individual mixing unitary operators $U_{M,e}$ are the same as for the Wasserstein case in Eq. (5.7). Furthermore, the whole mixing unitary operator U_M is constructed by applying them in sequence as in Eq. (5.8). Note however that since there are fewer edges in this case, there are also fewer individual unitaries. We apply the operators corresponding to the main vertices first and those corresponding to auxiliary vertices last. Since this mixing operator also produces some quantum states that do not satisfy the constraints in Eq. (5.12), we follow the same steps as for the Wasserstein distance and we relax the constraints to ensure that the mixing operator preserves the feasibility of the quantum states.

Theorem 5.3. *The mixing operator $U_M(\beta)$ preserves the feasibility of quantum states. That is, if $|\boldsymbol{s}\rangle$ represents a matching that satisfies the constraints*

$$\begin{aligned} \sum_{(i,j)\sim v} \delta_{(i,j)} \leq 1 \ , \quad \forall v \in \mathcal{D}_1, \mathcal{D}_2 \\ \sum_{e\sim v} \delta_e \geq 1 \ , \quad \forall v \in \mathcal{D}_2 \end{aligned} \tag{5.16}$$

then $U_M(\beta)\,|\boldsymbol{s}\rangle$ will also satisfy these constraints. Moreover, the minimum of the cost function in Eq. (5.11) over these constraints is the same as the minimum over the constraints in Eq. (5.12).

The mixing operator of Theorem 5.3 produces every solution to the relaxed constraints in Eq. (5.16) in a single iteration. Using the mixing operator again changes the amplitudes of the quantum states but it does not create any new states.

Theorem 5.4. *Applying each of the $n \times m + m$ unitaries $U_{M,e}(\beta)$ in sequence starting with the initial state*

$$\left| \begin{matrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \\ 0 & \cdots & 0 \end{matrix} \right\rangle \tag{5.17}$$

yields a superposition of all possible matchings that satisfy Eq. (5.16) with non-zero amplitudes.

The proofs of Theorem 5.4 and 5.3 are similar to those of Theorems 5.1 and 5.2, respectively, so we delegate them to Appendix A.

5.3 Implementations

In this section we provide a few implementations of our algorithm to some simple persistence diagrams. The first are persistence diagrams of samples from circles which we mostly use to illustrate the quantum algorithm for each distance and show the differences between them.

Next, we add some noise to the samples from these circles, which creates similar but slightly more complicated persistence diagrams, and compare the performance of the two methods.

5.3.1 An Illustrative Example

We first use a simple example shown in Fig. 5.3 to showcase the corresponding circuits for the Wasserstein and d_p^c distances. One of the data sets consists of points sampled from a circle, the corresponding persistence diagram of dimension 1 has only one point. The other data set has in addition samples from a second larger circle, and thus the corresponding diagram contains a second point. So both distances are obtained by matching the shared points x_1 and y_1 representing the small circle, and penalizing the extra point y_2 created by the large circle. For this example one iteration of the unitaries is more than enough to find the matching that minimizes the cost functions and represents the distance between the diagrams.

We can illustrate how the mixing unitary operator acts on the initial state using a tree in which each level represents the application of an individual mixing unitary to the current state of the system, see Fig. 5.4 for one such tree corresponding to the Wasserstein distance.

Fig. 5.5 shows the quantum circuit for the mixing unitary operator corresponding to the Wasserstein distance. Barriers separate the different individual mixing operators, and the qubits are arranged in a main register for the main edges between points in the diagrams, auxiliary registers for the edges to auxiliary vertices, and the ancillas needed to compute control clauses.

The quantum states for the Waserstein case are arranged as

$$\left| \begin{matrix} (x_1, y_1) & (x_1, y_2) & (x_1, \tilde{x}_1) \\ (\tilde{y}_1, y_1) & (\tilde{y}_2, y_2) & \end{matrix} \right\rangle, \tag{5.18}$$

and the ones encoding possible solutions to the optimizations problem are shown in Eq. (5.19). In particular, the quantum states $|\mathbf{0}\rangle, |\mathbf{7}\rangle, |\mathbf{8}\rangle$ correspond to matchings that satisfy the original constraints in Eq. (5.2), while the remaining states $|\mathbf{1}\rangle, \ldots, |\mathbf{6}\rangle$ only satisfy the relaxed constraints in Eq. (5.9).

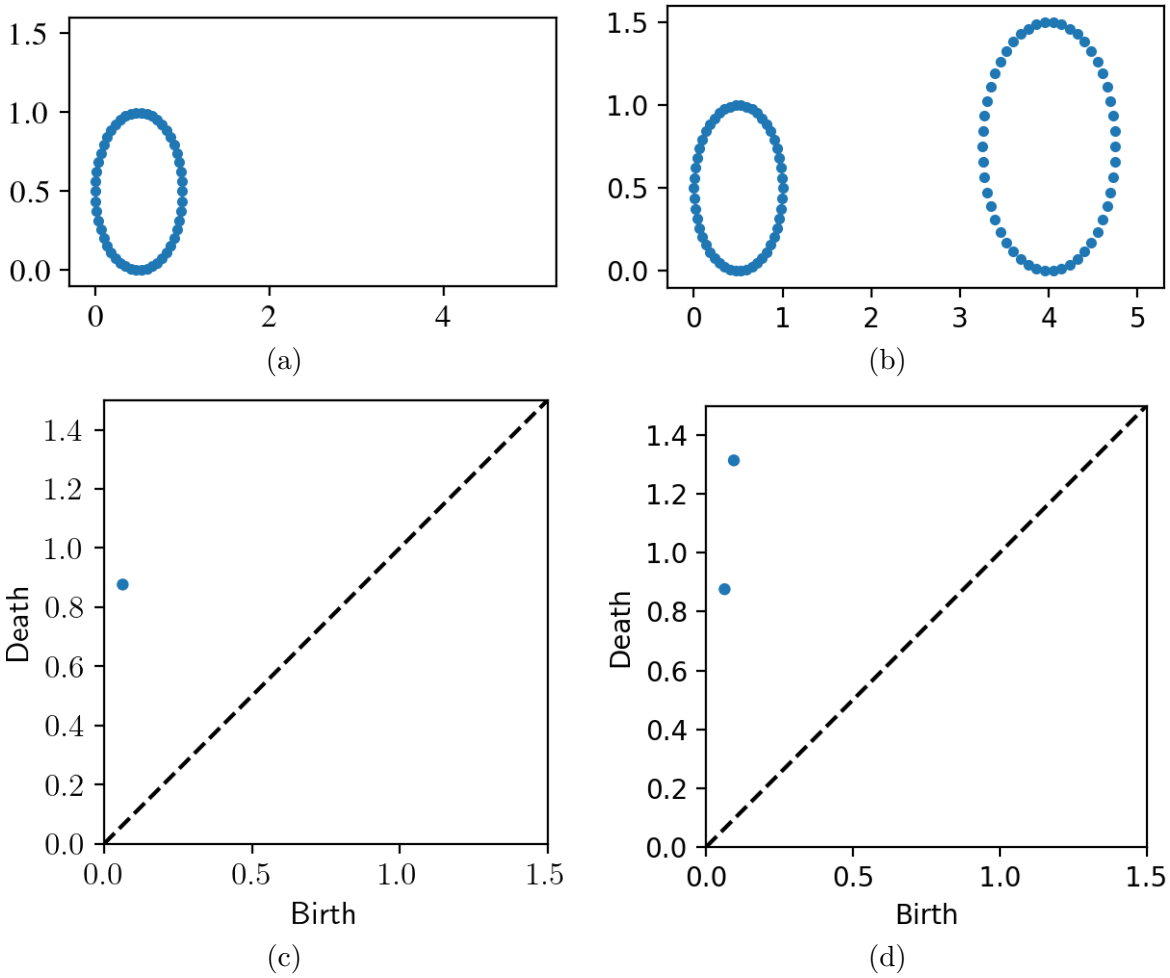

Figure 5.3: Two point clouds (top), the first (left) sampled from one circle and the second (right) sampled from two circles, along with their respective persistence diagrams (bottom) for dimension 1.

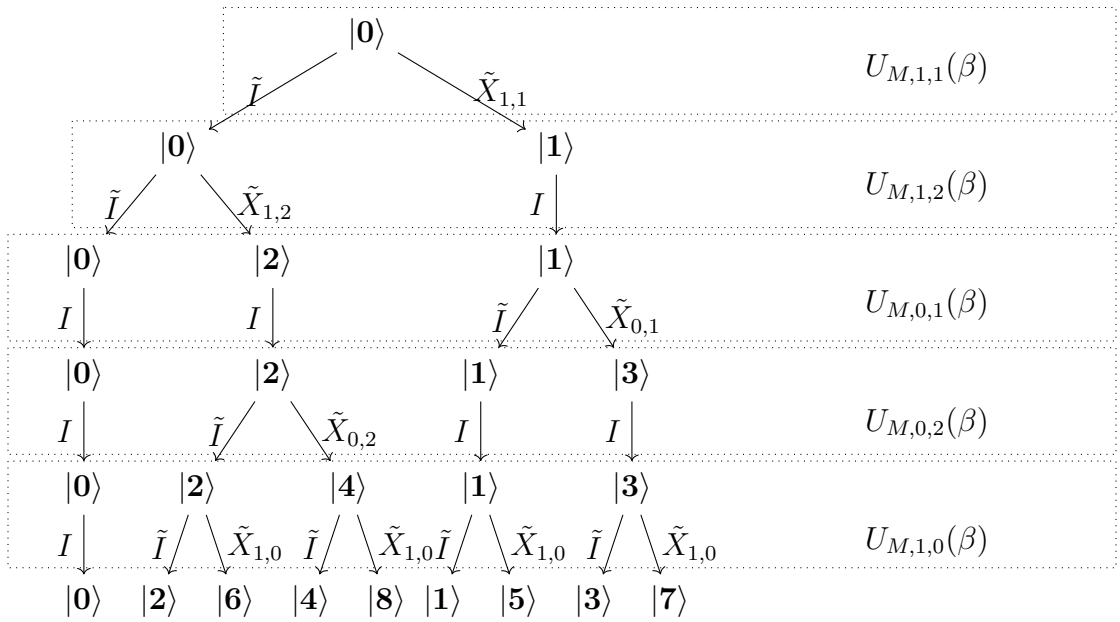

Figure 5.4: Construction tree for Wasserstein distance of the example in Fig. 5.3, where the quantum states are shown in Eq. (5.19).

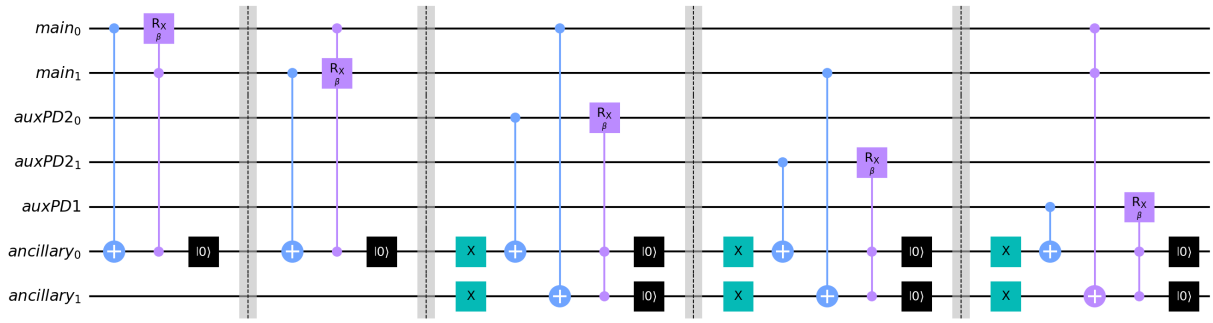

Figure 5.5: Quantum circuit for the mixing unitary operator of the Wasserstein distance between the persistence diagrams shown in Fig. 5.3.

$$
\begin{aligned}
|\mathbf{0}\rangle &\equiv \left|\begin{matrix} 1 & 1 & 0 \\ 0 & 0 & \end{matrix}\right\rangle & |\mathbf{1}\rangle &\equiv \left|\begin{matrix} 0 & 1 & 0 \\ 0 & 0 & \end{matrix}\right\rangle & |\mathbf{2}\rangle &\equiv \left|\begin{matrix} 1 & 0 & 0 \\ 0 & 0 & \end{matrix}\right\rangle \\
|\mathbf{3}\rangle &\equiv \left|\begin{matrix} 0 & 1 & 0 \\ 1 & 0 & \end{matrix}\right\rangle & |\mathbf{4}\rangle &\equiv \left|\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & \end{matrix}\right\rangle & |\mathbf{5}\rangle &\equiv \left|\begin{matrix} 0 & 1 & 1 \\ 0 & 0 & \end{matrix}\right\rangle \\
|\mathbf{6}\rangle &\equiv \left|\begin{matrix} 1 & 0 & 1 \\ 0 & 0 & \end{matrix}\right\rangle & |\mathbf{7}\rangle &\equiv \left|\begin{matrix} 0 & 1 & 1 \\ 1 & 0 & \end{matrix}\right\rangle & |\mathbf{8}\rangle &\equiv \left|\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & \end{matrix}\right\rangle
\end{aligned}
\tag{5.19}
$$

The mixing operator consists of a series of individual unitary operators corresponding to each edge. First, we apply those corresponding to the main edges, in this case (x_1, y_1) and (x_1, y_2). As shown on the first level of the tree in Fig. 5.4, applying the individual mixing unitary $U_{M,1,1}(\beta)$ defined in Eq. (5.7) to the initial state $|\mathbf{0}\rangle$ creates the superposition $\cos(\beta/2)\,|\mathbf{0}\rangle - i\sin(\beta/2)\,|\mathbf{1}\rangle$, with $|\mathbf{0}\rangle, |\mathbf{1}\rangle$ given in Eq. (5.19). Indeed, one may verify that for the initial state $|\mathbf{0}\rangle$ the control clause f introduced in Def. 5.1 has a value $f(x_1, y_1) = 1$. Next, the individual mixing unitary $U_{M,1,2}$ acts on the superposition generated by the previous step. Notice that the term involving $|\mathbf{1}\rangle$ remains unchanged since the control clause on this state is $f(x_1, y_2) = 0$. However, the control clause is $f(x_1, y_2) = 1$ on the state $|\mathbf{0}\rangle$, so the corresponding term is split into $\cos^2(\beta/2)\,|\mathbf{0}\rangle - i\cos(\beta/2)\sin(\beta/2)\,|\mathbf{2}\rangle$. Thus, after the second individual unitary the state of the system is given by a superposition of $|\mathbf{0}\rangle, |\mathbf{1}\rangle, |\mathbf{2}\rangle$ as can be seen on the second level in Fig. 5.4. Note that the initial state corresponds to the solution of the constraints in Eq. (5.2) that matches every point to the diagonal. Moreover, the individual unitaries for the main edges generate solutions to the relaxed constraints in Eq. (5.9) by adding the corresponding main edge to the matching.

The second part of the mixing operator consists of the individual mixing unitaries that correspond to auxiliary edges, that is $(\tilde{y}_1, y_1)$, $(\tilde{y}_2, y_2)$, and $(x_1, \tilde{x}_1)$. The third and fourth levels in Fig. 5.4 show that the individual unitaries $U_{M,0,1}$ and $U_{M,0,2}$ generate quantum states $|\mathbf{3}\rangle$ and $|\mathbf{4}\rangle$ respectively. Finally, the individual unitary $U_{M,1,0}$ generates all the remaining solutions $|\mathbf{5}\rangle, |\mathbf{6}\rangle, |\mathbf{7}\rangle, |\mathbf{8}\rangle$. Notice that the unitaries for the auxiliary edges act by removing the corresponding edge from the matching.

Figure 5.6 displays a tree for the d_p^c distance and Figure 5.7 the quantum circuit for the corresponding mixing unitary operator. Notice that the last individual mixing unitary is not present in this case, which illustrates the possible advantage of using this distance since the individual mixing unitaries are by far the most complex part of the algorithm. Control clauses used to construct the mixing operator are the only sections of our algorithm that involve interactions between several qubits, thus removing some of these sections could provide an advantage when the connectivity between qubits is limited. Of course, as mentioned earlier the number of qubits and gates needed to compute the d_p^c distance is less than for the Wasserstein distance.

On the other hand, the quantum states for the d_p^c distance are given by

$$\left| \begin{matrix} (x_1, y_1) & (x_1, y_2) \\ (\tilde{y}_1, y_1) & (\tilde{y}_2, y_2) \end{matrix} \right\rangle, \tag{5.20}$$

and the possible solutions to the optimization problem are depicted in Eq. (5.21). The quantum states encoding solutions to the original constraints in Eq. (5.12) are $|\mathbf{0}\rangle, |\mathbf{3}\rangle, |\mathbf{4}\rangle$, and those that represent solutions of the relaxed constraints in Eq. (5.16) are $|\mathbf{1}\rangle, |\mathbf{2}\rangle$.

$$\begin{aligned} |\mathbf{0}\rangle &\equiv \left| \begin{matrix} 1 & 1 \\ 0 & 0 \end{matrix} \right\rangle \quad |\mathbf{1}\rangle \equiv \left| \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix} \right\rangle \quad |\mathbf{2}\rangle \equiv \left| \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix} \right\rangle \\ |\mathbf{3}\rangle &\equiv \left| \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \right\rangle \quad |\mathbf{4}\rangle \equiv \left| \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \right\rangle \end{aligned} \tag{5.21}$$

Similar to the Wasserstein case, the initial state $|\mathbf{0}\rangle$ represents penalizing all points with the constant c. The first and second level of the tree in Fig. 5.6 show that the individual mixing unitaries corresponding to main edges $U_{M,1,1}$ and $U_{M,1,2}$ generate the quantum states $|\mathbf{1}\rangle$ and $|\mathbf{2}\rangle$ respectively by adding main edges to the matching. Then, the last two levels in Fig. 5.6 show how individual operators corresponding to auxiliary edges $U_{M,0,1}$ and $U_{M,0,2}$ remove auxiliary edges to generate the states $|\mathbf{3}\rangle$ and $|\mathbf{4}\rangle$ respectively. Note that in this case the edge $x_1, \tilde{x}_1$ is not present, which results in one less individual unitary and significantly fewer possible solutions.

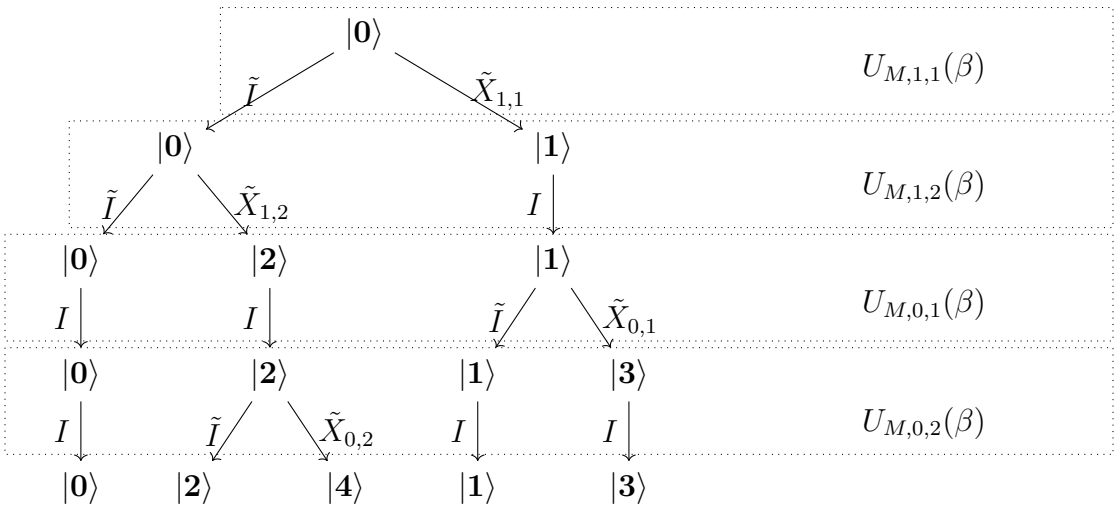

Figure 5.6: Construction tree for the example in Fig. 5.3.

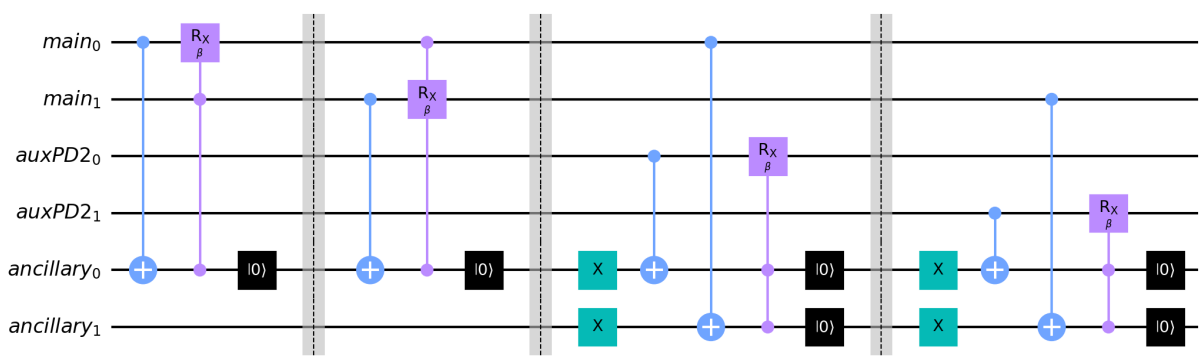

Figure 5.7: Quantum circuit for the mixing unitary operator of the d_p^c distance between the persistence diagrams shown in Fig. 5.3.

5.3.2 Circles vs Noisy Circles

Next, we added noise to the samples from the circles in Fig. 5.3a and 5.3b to create slightly more complex data sets (shown in Fig. 5.8a and 5.8b). Their corresponding persistence diagrams of dimension one have an extra point each generated by the noise, see Fig. 5.8c and 5.8d.

We used the Qiskit library for Python to simulate our quantum algorithms on the noisy and noiseless data sets. Only one iteration of the mixing and problem operators was used in the simulations. The rotation angles were obtained by minimizing the cost functions in Eq. (5.1) and (5.11). The results of the simulations using the optimal angles can be found in Fig. 5.9. Note that the parameters from Definitions 2.8 and 2.9 chosen for all simulations are $q = \infty$, $p = 2$, and $c = 0.2$.

Figure 5.9a shows the observed quantum states for the Wasserstein distance on the noiseless data sets from Fig. 5.3. The first observed state $|0\rangle$ corresponds to $\left|\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & \end{matrix}\right\rangle$ which matches the circle in Fig. 5.3a to the large circle in Fig. 5.3b. On the other hand, the state observed with the most frequency ($|1\rangle$) corresponds to $\left|\begin{matrix} 0 & 1 & 1 \\ 1 & 0 & \end{matrix}\right\rangle$, the matching that yields the Wasserstein distance.

Similarly, the results for the d_p^c distance on the noiseless data sets are depicted in Fig. 5.9c. As in the case of the Wasserstein distance, the observed quantum states are $|0\rangle$ corresponding to $\left|\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}\right\rangle$, and the one with the highest frequency, $|1\rangle$, equivalent to the matching that yields the d_p^c distance, $\left|\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix}\right\rangle$.

For the noisy data sets from Fig. 5.8 one iteration of the quantum algorithm was not good enough for the Wasserstein distance as shown in Fig. 5.9b. This time the most observed state $|5\rangle$ corresponds to the initial state $\left|\begin{matrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & \end{matrix}\right\rangle$ which is definitely not the matching that yields the Wasserstein distance. The other observed states are solutions to the relaxed constraints in Eq. (5.9) but not the original constraints in Eq. (5.2), and therefore also do

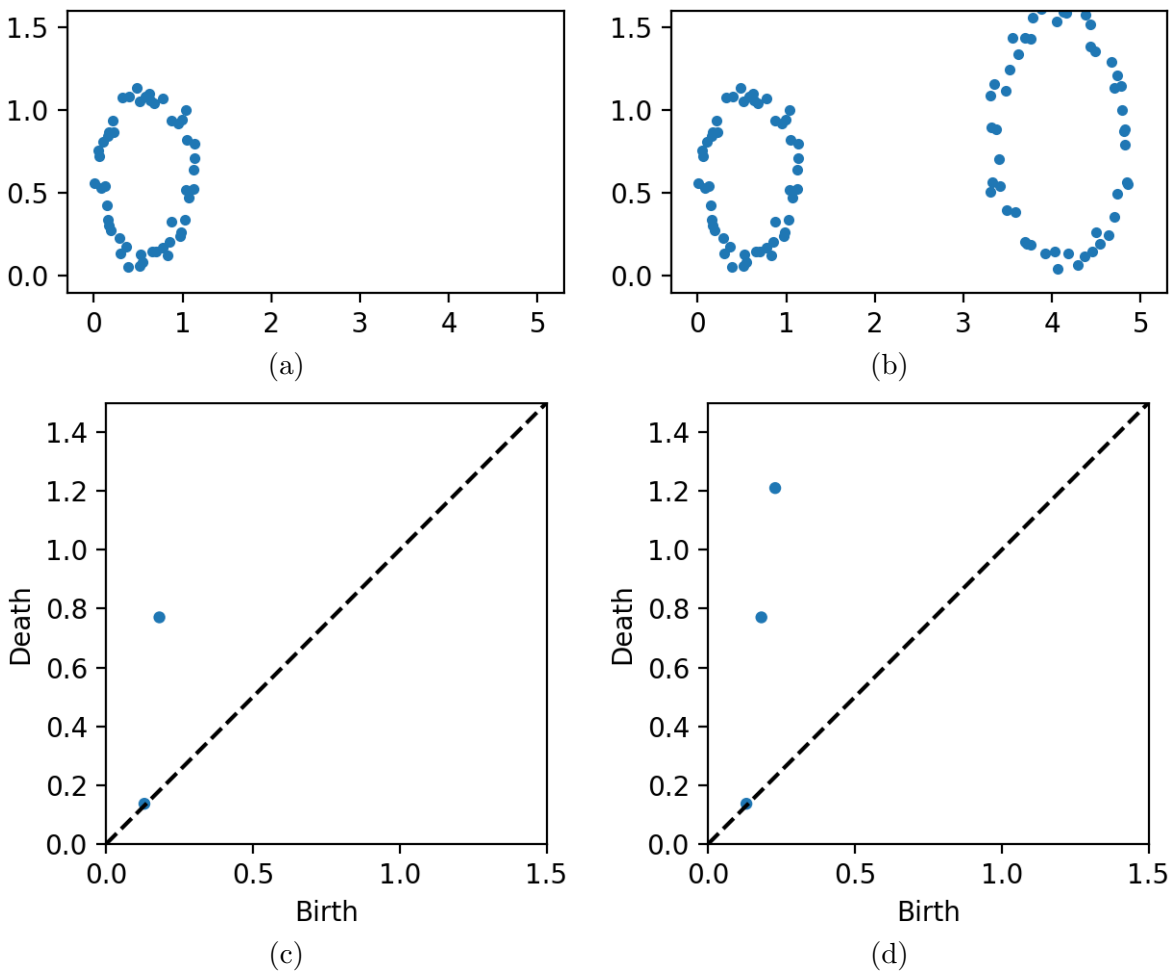

Figure 5.8: Two point clouds (top) resulting from adding noise to the data sets shown in Fig. 5.3a and 5.3b, along with their respective persistence diagrams (bottom) for dimension 1.

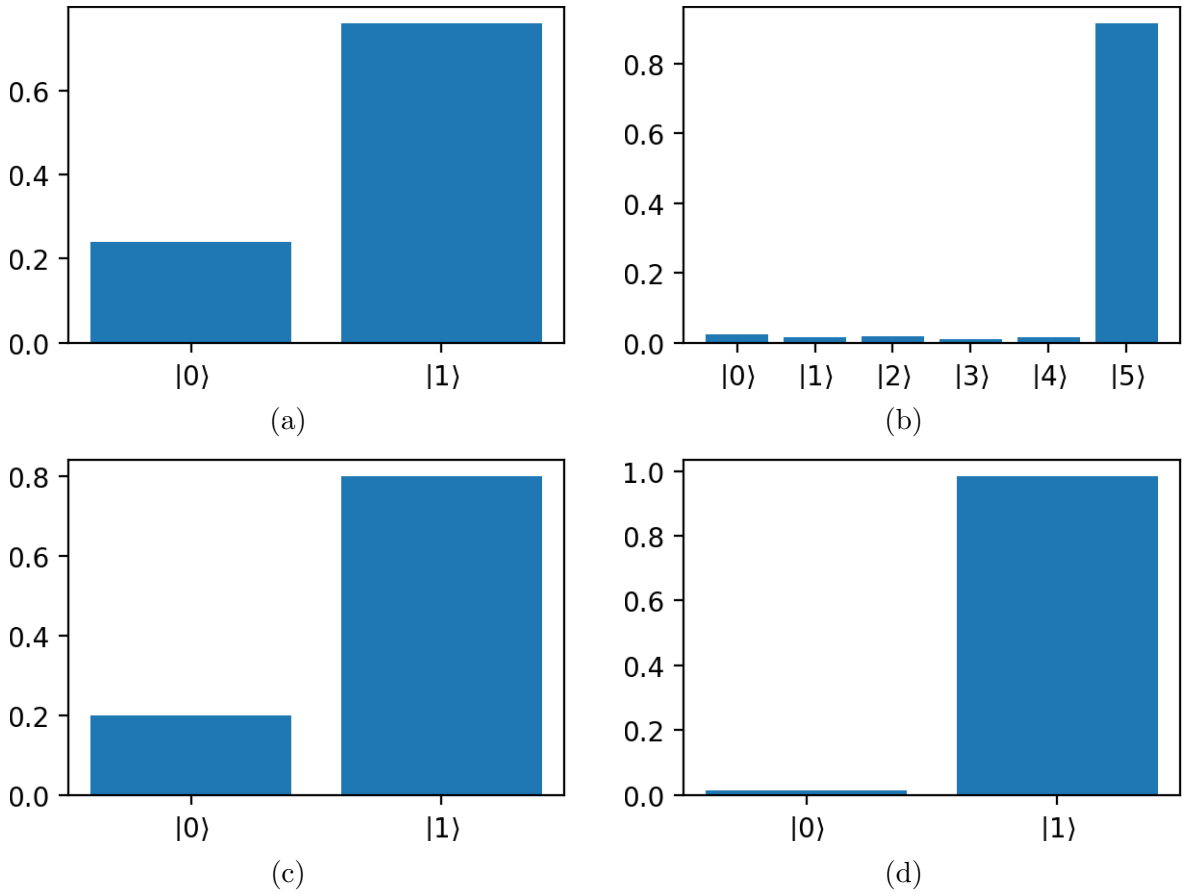

Figure 5.9: Frequency of measurements after one iteration of our QAOA algorithm for the Wasserstein (top) and d_p^c (bottom) distances on the data sets from Fig. 5.3 (left) and Fig. 5.8 (right).

not yield the Wasserstein distance. In contrast, the algorithm for the d_p^c distance did find the solution for the noisy data sets, as illustrated in Fig. 5.9d. The least observed quantum state $|0\rangle$ corresponds to the solution that matches the holes created by noise as well the circle in Fig. 5.8a to the large circle in Fig. 5.8b, $\left|\begin{matrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}\right\rangle$. The state with the highest frequency, $|1\rangle$, is equivalent to the matching that yields the d_p^c distance, $\left|\begin{matrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{matrix}\right\rangle$. This suggests that computing the d_p^c distance can be more efficient since the circuit for the Wasserstein distance already requires more iterations for such a simple example.

5.3.3 Complexity and Comparison with Existing Methods

Given that n is the number of points in the smaller diagram and m the number of points in the larger diagram, the unitary operator for the cost Hamiltonian of the Wasserstein distance requires the application of $n \times m + n + m$ single qubit rotation operators R_Z which can be performed in quantum parallel for a cost of $\mathcal{O}(1)$ operations. On the other hand, the mixing operator needs to perform single qubit rotations R_X with a control clause f for each edge in succession. This step requires $\mathcal{O}(nm)$ operations since there are $n \times m + n + m$ edges in the Wasserstein graph. As for the size of the quantum computer, the algorithm for the Wasserstein distance requires $n \times m + n + m$ qubits (one for each edge) and possibly some ancillary qubits to implement the control clause. Overall, the size and depth of the quantum circuit for the Wasserstein distance should scale as $\mathcal{O}(nm)$.

On the other hand, the cost Hamiltonian for the d_p^c distance requires $n \times m + m$ rotations R_Z that can still be done in quantum parallel for a cost of $\mathcal{O}(1)$ operations. But the mixing operator needs to perform fewer rotations R_X since there are only $n \times m + m$ edges in the d_p^c graph. In addition, the algorithm for the d_p^c distance only requires $n \times m + m$ qubits and likely fewer ancillary qubits for the control clauses. So, while the scaling remains the same

as for the Wasserstein case ($\mathcal{O}(nm)$), this alternative approach offers an advantage in both depth and size over the Wasserstein circuit.

Classical methods to compute the distance between persistence diagrams rely on the Hungarian algorithm which in general has a complexity of $\mathcal{O}(n^2m)$. Unfortunately, we were unable to find any classical implementations of the d_p^c distance introduced in [42] or discussions about its complexity. In the case of the Wasserstein distance, geometric properties of persistence diagrams have been exploited to reduce the number of matchings to be checked and improve the complexity to an estimated $\mathcal{O}(n^{1.6})$ [54]. However, this approach only approximates the distance between persistence diagrams rather than obtaining the exact value. In contrast, our quantum algorithms, as well as other classical implementations of the Hungarian algorithm with super-quadratic complexity, do find the exact distance between persistence diagrams. It is also possible that a similar approach could be applied to our quantum algorithms to reduce the number of qubits and R_X rotations required.

On the other hand, there are only a few previous quantum approaches for computing the distance between persistence diagrams [67, 68]. The quantum methodology from [67] is designed for a quantum annealing based device, whereas our algorithms and the one introduced in [68] would run on a quantum gate based computer. Both of these previous works have been somewhat vague with regards to complexity estimates. Nevertheless, all three approaches involve expressing the distance as an optimization problem over binary variables and identifying each edge of the graph with a qubit, so the overall complexity should be quite similar. The main difference between these previous approaches and ours is that they formulate an unconstrained optimization problem by encoding the constraints into the cost function. In contrast, we formulate an optimization problem with constraints and use control clauses to implement them. Moreover, we are the first to provide a quantum algorithm for the d_p^c distance, which is more efficient to implement and can even produce better results than the Wasserstein distance in some cases [42].

Chapter 6

Discussion

One of the main contributions of my work is a quantum algorithm for persistent homology that was shown to track topological features of a data set throughout changes in the resolution. This algorithm introduces a persistent Dirac operator that generalizes the Dirac operator previously used in [55, 56]. It only requires $\mathcal{O}(n)$ qubits to encode all possible simplices from a data set with n points, which is exponentially less memory than a classical algorithm would need. However, it is unlikely that the number of operations can be reduced exponentially as well [64, 65, 66]. In particular, while the dimension of the whole Hilbert space $\mathcal{H}$ is 2^n, the dimensions of the subspaces $\mathcal{H}^\epsilon$, $\mathcal{H}^{\epsilon'}$ and $\mathcal{H}^{\epsilon,\epsilon'}$ may not be exponential in n, so the Grover's search algorithm used to implement projections could have an exponential depth as it depends on the ratio of these dimensions [66]. But even though exponential speedup is unlikely, the algorithm could still provide at least a Grover-like (quadratic) speedup over classical counterparts. Moreover, my algorithm as well as other quantum algorithms for homology and persistent homology achieve the greatest advantage when computing features of higher dimensions [66], something that classical algorithms often struggle with. In addition, the methodology I developed allows for the computation of non-harmonic spectra of the persistent combinatorial Laplacian, in contrast to other approaches [61, 62] that can only estimate the dimension of its kernel. On the other hand, most quantum algorithms (including my own) assume access to an oracle that determines if a simplex is present in the complex at a given scale or resolution, but previous works only proposed constructions of these membership oracles for point cloud data sets. My design of a membership oracle based on Takens' delay embedding enables quantum algorithms for persistent homology to consider time series data sets such as signals.

Finally, my QAOA inspired approach to compute distances between persistence diagrams provides a quantum methodology to compute the Wasserstein distance as well as the d_p^c distance. Moreover, the results in Section 5.3 suggest that this alternative distance has a more efficient quantum, and perhaps even classical, implementation. Along with previous studies that show the d_p^c distance can outperform the Wasserstein distance for certain classification tasks [12, 42], this leads me to believe that it could be a great alternative for some cases and that it should be studied further. On the other hand, while the quantum advantage

of this algorithm alone over classical counterparts is not clear, it is important to remember that computing the distance between persistence diagrams is merely a subroutine of larger machine learning algorithms for classification an clustering. A complete framework for TDA involves creating topological summaries –like persistence diagrams– of data sets via persistent homology, then feeding these summaries into a machine learning algorithm for classification, which requires comparing the summaries using an algorithm like the one we provide here. Quantum algorithms for persistent homology promise an advantage for the process of obtaining persistence diagrams and other topological representations [44, 61, 62]. In addition, quantum approaches to K-Means, logistic regression, and Support Vector Machine algorithms promise similar results for classification and clustering tasks [92, 93, 94, 95, 96]. So, our quantum algorithm could be used as a subroutine of a larger quantum algorithm for classification to compare the results of quantum algorithms for persistent homology, creating a fully quantum framework for TDA. Without a subroutine like this one, the output of the first algorithm would need to be measured into a classical computer, then the distances between persistence diagrams would need to be computed classically and later loaded into the quantum algorithm for classification. The extra steps to send information between classical and quantum devices increase the number of operations and possible errors, therefore diminishing the potential advantage that quantum approaches can provide.

Bibliography

[1] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009. 2, 7

[2] Erik Carlsson, Gunnar Carlsson, and Vin De Silva. An algebraic topological method for feature identification. *International Journal of Computational Geometry*, 16(4):291–314, 2006. 2

[3] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Society, 2010. 2, 3, 7, 10, 15

[4] Gunnar Carlsson and Mikael Vejdemo-Johansson. *Topological data analysis with applications*. Cambridge University Press, 2021. 2, 3, 7, 15

[5] James R Munkres. *Elements of algebraic topology*. CRC press, 2018. 2, 7

[6] Marvin J Greenberg. *Algebraic topology: a first course*. CRC Press, 2018. 2, 7

[7] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002. 2, 7

[8] Genki Kusano, Yasuaki Hiraoka, and Kenji Fukumizu. Persistence weighted gaussian kernel for topological data analysis. In *International conference on machine learning*, pages 2004–2013. PMLR, 2016. 2

[9] Elizabeth Munch. *Applications of persistent homology to time varying systems*. PhD thesis, Duke University, 2013. 2

[10] Herbert Edelsbrunner. Persistent homology in image processing. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 182–183. Springer, 2013. 2

[11] Firas A Khasawneh and Elizabeth Munch. Chatter detection in turning using persistent homology. *Mechanical Systems and Signal Processing*, 70:527–541, 2016. 2

[12] Andrew Marchese, Vasileios Maroulas, and Josh Mike. K- means clustering on the space of persistence diagrams. In *Wavelets and Sparsity XVII*, volume 10394, pages 218–227. SPIE, 2017. 2, 3, 5, 17, 70

[13] Moo Chung, Jamie Hanson, Jieping Ye, Richard Davidson, and Seth Pollak. Persistent homology in sparse regression and its application to brain morphometry. *IEEE transactions on medical imaging*, 34(9):1928–1939, 2015. 2

[14] Saba Emrani, Thanos Gentimis, and Hamid Krim. Persistent homology of delay embeddings and its application to wheeze detection. *IEEE Signal Processing Letters*, 21(4):459–463, 2014. 2

[15] Cássio MM Pereira and Rodrigo F de Mello. Persistent homology for time series and spatial data clustering. *Expert Systems with Applications*, 42(15):6026–6038, 2015. 2

[16] Monica Nicolau, Arnold J Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011. 2

[17] Joshua Mike and Vasileios Maroulas. Combinatorial hodge theory for equitable kidney paired donation. *Foundations of Data Science*, 1(1):87–101, 2019. 2

[18] Joshua Mike, Colin D Sumrall, Vasileios Maroulas, and Fernando Schwartz. Nonlandmark classification in paleobiology: computational geometry as a tool for species discrimination. *Paleobiology*, pages 1–11, 2016. 2

[19] Vasileios Maroulas, Cassie Putman Micucci, and Farzana Nasrin. Bayesian topological learning for classifying the structure of biological networks. *Bayesian Analysis*, 17(3):711 – 736, 2022. 2

[20] Jacopo Binchi, Emanuela Merelli, Matteo Rucco, Giovanni Petri, and Francesco Vaccarino. jholes: A tool for understanding biological complex networks via clique weight rank persistent homology. *Electronic Notes in Theoretical Computer Science*, 306:5–18, 2014. 2

[21] Paul Bendich, James Stephen Marron, Ezra Miller, Alex Pieloch, and Sean Skwerer. Persistent homology analysis of brain artery trees. *The Annals of Applied Statistics*, 10(1):198–218, 2016. 2

[22] Na Gong, Farzana Nasrin, Yong Wang, Huibin Wu, David J Keffer, Vasileios Maroulas, and Orlando Rios. Persistent homology on electron backscatter diffraction data in nano/ultrafine-grained 18cr–8ni stainless steel. *Materials Science and Engineering: A*, 829:142172, 2022. 2

[23] Dong Chen, Kaifu Gao, Duc Duy Nguyen, Xin Chen, Yi Jiang, Guo-Wei Wei, and Feng Pan. Algebraic graph-assisted bidirectional transformers for molecular property prediction. *Nature Communications*, 12(1):3521, 2021. 2

[24] Theodore Papamarkou, Farzana Nasrin, Austin Lawson, Na Gong, Orlando Rios, and Vasileios Maroulas. A random persistence diagram generator. *Statistics and Computing*, 32(5):88, 2022. 2

[25] Vasileios Maroulas, Farzana Nasrin, and Christopher Oballe. A bayesian framework for persistent homology. *SIAM Journal on Mathematics of Data Science*, 2(1):48–74, 2020. 2

[26] J. Townsend, C.P. Micucci, J.H. Hymel, V. Maroulas, and K. D. Vogiatzis. Representation of molecular structures with persistent homology for machine learning applications in chemistry. *Nat Commun*, 11:3230, 2020. 2

[27] Kelin Xia, Xin Feng, Yiying Tong, and Guo Wei Wei. Persistent homology for the quantitative prediction of fullerene stability. *Journal of computational chemistry*, 36(6):408–422, 2015. 2

[28] Ramanarayan Vasudevan, Aaron Ames, and Ruzena Bajcsy. Persistent homology for automatic determination of human-data based cost of bipedal walking. *Nonlinear Analysis: Hybrid Systems*, 7(1):101–115, 2013. 2

[29] Vinay Venkataraman, Karthikeyan Natesan Ramamurthy, and Pavan Turaga. Persistent homology of attractors for action recognition. In *2016 IEEE international conference on image processing (ICIP)*, pages 4150–4154. IEEE, 2016. 2

[30] Aaron Adcock, Erik Carlsson, and Gunnar Carlsson. The ring of algebraic functions on persistence bar codes. *Homology, Homotopy and Applications*, 18(1):381–402, 2016. 2

[31] Vin De Silva and Robert Ghrist. Coordinate-free coverage in sensor networks with controlled boundaries via homology. *The International Journal of Robotics Research*, 25(12):1205–1222, 2006. 2

[32] Vin De Silva and Robert Ghrist. Homological sensor networks. *Notices of the American mathematical society*, 54(1), 2007. 2

[33] Pawel Dlotko, Robert Ghrist, Mateusz Juda, and Marian Mrozek. Distributed computation of coverage in sensor networks by homological methods. *Applicable Algebra in Engineering, Communication and Computing*, 23(1-2):29–58, 2012. 2

[34] Gunnar Carlsson and Vin De Silva. Zigzag persistence. *Foundations of computational mathematics*, 10(4):367–405, 2010. 2

[35] Gunnar Carlsson, Vin De Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 247–256. ACM, 2009. 2

[36] Vin De Silva and Robert Ghrist. Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, 7(1):339–358, 2007. 2

[37] Lee M Seversky, Shelby Davis, and Matthew Berger. On time-series topological data analysis: New data and opportunities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 59–67, 2016. 2

[38] Jose A Perea, Anastasia Deckard, Steve B Haase, and John Harer. Sw1pers: Sliding windows and 1-persistence scoring; discovering periodicity in gene expression time series data. *BMC bioinformatics*, 16(1):257, 2015. 2

[39] Jose A Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis. *Foundations of Computational Mathematics*, 15(3):799–838, 2015. 2

[40] Mijail Guillemard and Armin Iske. Signal filtering and persistent homology: an illustrative example, 2011. 2

[41] Andrew Marchese and Vasileios Maroulas. Topological learning for acoustic signal identification. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1377–1381. ISIF, 2016. 2, 14

[42] A. Marchese and V. Maroulas. Signal classification with a point process distance on the space of persistence diagrams. *Advances in Data Analysis and Classification*, 12(3):657–682, 2018. 2, 3, 4, 17, 68, 70

[43] Rui Wang, Duc Duy Nguyen, and Guo-Wei Wei. Persistent spectral graph. *International journal for numerical methods in biomedical engineering*, 36(9):e3376, 2020. 2, 4, 11

[44] Bernardo Ameneyro, Vasileios Maroulas, and George Siopsis. Quantum persistent homology. *Journal of Applied and Computational Topology*, pages 1–20, 2024. 2, 3, 4, 11, 12, 21, 71

[45] JunJie Wee, Ginestra Bianconi, and Kelin Xia. Persistent dirac for molecular representation. *Scientific Reports*, 13(1):11183, 2023. 2, 11, 12

[46] Faisal Suwayyid and Guo-Wei Wei. Persistent dirac of paths on digraphs and hypergraphs. *Foundations of Data Science*, 6(2):124–153, 2024. 2, 11, 12

[47] Faisal Abdulaziz Suwayyid and Guowei Wei. Persistent mayer dirac. *Journal of Physics: Complexity*, 2024. 2, 11, 12

[48] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33:249–274, 02 2005. 3, 13

[49] Dmitriy Morozov. Dionysus, a C++ library for computing persistent homology, 2007. https://github.com/mrzv/dionysus. 3

[50] Ulrich Bauer. Ripser, 2015. https://github.com/Ripser/ripser. 3

[51] Ulrich Bauer. Ripser: efficient computation of Vietoris-Rips persistence barcodes. *J. Appl. Comput. Topol.*, 5(3):391–423, 2021. 3

[52] Christopher Tralie, Nathaniel Saul, and Rann Bar-On. Ripser.py: A lean persistent homology library for python. *The Journal of Open Source Software*, 3(29):925, Sep 2018. 3

[53] Chao Chen and Michael Kerber. Persistent homology computation with a twist. In *Proceedings 27th European Workshop on Computational Geometry*, volume 11, 2011. 3

[54] Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithmics (JEA)*, 22:1–20, 2017. 3, 68

[55] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. Quantum algorithms for topological and geometric analysis of data. *Nature communications*, 7(1):1–7, 2016. 3, 4, 11, 12, 22, 31, 32, 37, 70

[56] George Siopsis. Quantum topological data analysis with continuous variables. *Foundations of Data Science*, 4(1):419–431, 2019. 3, 11, 31, 32, 70

[57] H. Huang and *et al.* Demonstration of topological data analysis on a quantum processor. *Optica*, 5:193, 2018. 3

[58] Shashanka Ubaru, Ismail Yunus Akhalwaya, Mark S. Squillante, Kenneth L. Clarkson, and Lior Horesh. Quantum topological data analysis with linear depth and exponential speedup, 2021. 3, 4, 21, 22, 23

[59] Dominic W. Berry, Yuan Su, Casper Gyurik, Robbie King, Joao Basso, Alexander Del Toro Barba, Abhishek Rajput, Nathan Wiebe, Vedran Dunjko, and Ryan Babbush. Analyzing prospects for quantum advantage in topological data analysis, 2023. 3, 4, 21, 23

[60] Ismail Yunus Akhalwaya, Shashanka Ubaru, Kenneth L. Clarkson, Mark S. Squillante, Vishnu Jejjala, Yang-Hui He, Kugendran Naidoo, Vasileios Kalantzis, and Lior Horesh. Towards quantum advantage on noisy quantum computers, 2022. 3, 4, 21

[61] Ryu Hayakawa. Quantum algorithm for persistent betti numbers and topological data analysis. *Quantum*, 6:873, 2022. 3, 4, 21, 36, 70, 71

[62] Sam McArdle, András Gilyén, and Mario Berta. A streamlined quantum algorithm for topological data analysis with exponentially fewer qubits, 2022. 3, 4, 21, 36, 70, 71

[63] Bernardo Ameneyro, George Siopsis, and Vasileios Maroulas. Quantum persistent homology for time series. In *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, pages 387–392, 2022. 3, 4, 36

[64] Marcos Crichigno and Tamara Kohler. Clique homology is qma1-hard. *arXiv preprint arXiv:2209.11793*, 2022. 3, 70

[65] Chris Cade and P Marcos Crichigno. Complexity of supersymmetric systems and the cohomology problem. *arXiv preprint arXiv:2107.00011*, 2021. 3, 70

[66] Alexander Schmidhuber and Seth Lloyd. Complexity-theoretic limitations on quantum algorithms for topological data analysis. *arXiv preprint arXiv:2209.14286*, 2022. 3, 25, 70

[67] Jesse J Berwald, Joel M Gottlieb, and Elizabeth Munch. Computing wasserstein distance for persistence diagrams on a quantum computer. *arXiv preprint arXiv:1809.06433*, 2018. 4, 68

[68] M. Saravanan and Mannathu Gopikrishnan. The wasserstein distance using qaoa: A quantum augmented approach to topological data analysis. In *2022 International Conference on Innovative Trends in Information Technology (ICITIIT)*, pages 1–8, 2022. 4, 68

[69] Bernardo Ameneyro, Rebekah Herrman, George Siopsis, and Vasileios Maroulas. Quantum distance approximation for persistence diagrams. *Journal of Physics: Complexity*, 6(1):015005, 2025. 4, 45

[70] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014. 4, 19

[71] Facundo Mémoli, Zhengchao Wan, and Yusu Wang. Persistent laplacians: Properties, algorithms and implications. *SIAM Journal on Mathematics of Data Science*, 4(2):858–884, 2022. 4, 11

[72] Zhenyu Meng and Kelin Xia. Persistent spectral–based machine learning (perspect ml) for protein-ligand binding affinity prediction. *Science Advances*, 7(19):eabc5329, 2021. 4

[73] Jiahui Chen, Yuchi Qiu, Rui Wang, and Guo-Wei Wei. Persistent laplacian projected omicron ba. 4 and ba. 5 to become new dominating variants. *Computers in Biology and Medicine*, 151:106262, 2022. 4

[74] Sagnik Chatterjee and Debajyoti Bera. Applying the quantum alternating operator ansatz to the graph matching problem, 2020. 4, 19, 48

[75] Eleanor Rieffel and Wolfgang Polak. An introduction to quantum computing for non-physicists. *ACM Comput. Surv.*, 32(3):300–335, sep 2000. 7, 19

[76] Anna Lopatnikova, Minh-Ngoc Tran, and Scott A Sisson. An introduction to quantum computing for statisticians and data scientists. *arXiv preprint arXiv:2112.06587*, 2021. 7

[77] Ginestra Bianconi. The topological dirac equation of networks and simplicial complexes. *Journal of Physics: Complexity*, 2(3):035022, 2021. 11

[78] Lucille Calmon, Michael T Schaub, and Ginestra Bianconi. Dirac signal processing of higher-order topological signals. *New Journal of Physics*, 25(9):093013, 2023. 12

[79] Li Shen, Jian Liu, and Guo-Wei Wei. Persistent mayer homology and persistent mayer laplacian. *arXiv preprint arXiv:2312.01268*, 2023. 13

[80] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981. 14

[81] Ruslan Shaydulin, Phillip C. Lotshaw, Jeffrey Larson, James Ostrowski, and Travis S. Humble. Parameter transfer for quantum approximate optimization of weighted MaxCut. *ACM Transactions on Quantum Computing*, feb 2023. 19, 48

[82] Ismail Yunus Akhalwaya, Yang-Hui He, Lior Horesh, Vishnu Jejjala, William Kirby, Kugendran Naidoo, and Shashanka Ubaru. Representation of the fermionic boundary operator. *Phys. Rev. A*, 106:022407, Aug 2022. 21, 22

[83] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002. 23, 25

[84] Lov K. Grover. Quantum computers can search rapidly by using almost any transformation. *Phys. Rev. Lett.*, 80:4329–4332, May 1998. 23, 25

[85] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Phys. Rev. Lett.*, 100:160501, Apr 2008. 24

[86] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Architectures for a quantum random access memory. *Physical Review A*, 78(5):052310, 2008. 24

[87] Patrick Rebentrost, Adrian Steffens, Iman Marvian, and Seth Lloyd. Quantum singular-value decomposition of nonsparse low-rank matrices. *Physical review A*, 97(1):012327, 2018. 30

[88] Sam Gunn and Niels Kornerup. Review of a quantum algorithm for betti numbers. *arXiv preprint arXiv:1906.07673*, 2019. 31, 33

[89] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Phys. Rev. Lett.*, 100:160501, Apr 2008. 37

[90] Jason Miller. Eeg micro-experiment, music vs. reading data sets. *kaggle* https://www.kaggle.com/datasets/millerintllc/eeg-microexperiment?resource=download, 2018. 39, 43

[91] Rebekah Herrman, Phillip C Lotshaw, James Ostrowski, Travis S Humble, and George Siopsis. Multi-angle quantum approximate optimization algorithm. *Scientific Reports*, 12(1):6781, 2022. 48

[92] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013. 71

[93] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. q-means: A quantum algorithm for unsupervised machine learning. *Advances in neural information processing systems*, 32, 2019. 71

[94] Hai-Ling Liu, Chao-Hua Yu, Yu-Sen Wu, Shi-Jie Pan, Su-Juan Qin, Fei Gao, and Qiao-Yan Wen. Quantum algorithm for logistic regression, 2019. 71

[95] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014. 71

[96] Zhaokai Li, Xiaomei Liu, Nanyang Xu, and Jiangfeng Du. Experimental realization of a quantum support vector machine. *Physical review letters*, 114(14):140504, 2015. 71

Appendix

Appendix A

Proofs of Theorems 5.3 and 5.4

A.1 Proof of Feasibility: Theorem 5.3

We first give some intuition about these new constraints and show that the minimum remains the same. Notice that the first constraint asks that a point v in one of the persistence diagrams is matched at most once to points in the other diagram. The second constraint requires that every point in the largest persistence diagram is matched at least once. This means that points in $\mathcal{D}_2$ could be matched more than once, but only one of the edges will go towards a point in the other diagram and any other edges connect to auxiliary vertices. So the solutions to these constraints will be matchings that represent a one-to-one function $\phi : \mathcal{D}_1 \to \mathcal{D}_2$ or matchings that result of adding connections to auxiliary vertices. In particular, any solution to the constraints in Eq. (5.12) is also a solution to Eq. (5.16), and any solution to the latter that is not also a solution to the former must have a greater cost.

Now, given a quantum state $|s\rangle$ which satisfies the constraints in Eq. (5.16), we show that $U_{M,e}(\beta)\,|s\rangle$ also satisfies the constraints for arbitrary $e \in E$. This in turn proves that $U_M(\beta)$ preserves the feasibility of quantum states as it is defined as a product of the individual unitaries.

From Eq. (5.7) note that there are only two possible cases for $U_{M,e}(\beta)\,|s\rangle$ depending on the value of $f(e)$. First, if $f(e) = 0$ the output state is simply $|s\rangle$. On the other hand, when $f(e) = 1$ the resulting state is $\cos\beta I\,|s\rangle - i\sin\beta X_e\,|s\rangle$. The first term is just $|s\rangle$ scaled by a constant, so we only need to verify that the second term is feasible.

Notice that the second term X_e flips the qubit $|e\rangle$ but it is only applied when $f(e) = 1$. If $e = (x_i, y_j)$ is an edge between points in the diagrams, $f(e) = 1$ if and only if $\delta_e, \delta_{i,k}, \delta_{l,j} = 0$ for all $l \neq i$ and $k \neq j$, that is, the edge $e = (x_i, y_j)$ is added to the matching if there are no other main edges containing vertices x_i or y_j. Since this operation adds an edge, the second constraint in Eq. (5.16) is trivially satisfied, moreover the first constraint is satisfied because the edge is only added if that sum is equal to zero. On the other hand, if $(\tilde{y}_j, y_j)$ is an edge to an auxiliary vertex, $f(e) = 1$ whenever $\delta_e = 1$ and at least one of $\delta_{l,j}$ for $1 \leq l \leq |\mathcal{D}_1|$ is non-zero, in other words, we remove the edge $(\tilde{y}_j, y_j)$ from the matching if there is a main edge connecting the corresponding vertex y_j to a vertex in the other diagram. So, the first

constraint in Eq. (5.16) is unaffected and the second one is still satisfied as we only remove an edge if the corresponding sum is at least 2.

All in all, $U_{M,e}(\beta)\,|s\rangle$ is a superposition of quantum states whose corresponding matchings satisfy the constraints in Eq. (5.16).

A.2 Proof of Completeness: Theorem 5.4

Notice that one may obtain all solutions to Eq. (5.16) by finding every possible choice for the main $n \times m$ edges that satisfy the first constraint, and then for each of these obtaining the different combinations of the m auxiliary edges that satisfy the second constraint. So, we first apply the unitaries corresponding to the main edges to produce the solutions to the first constraint while keeping all the auxiliary edges on. After which we use the remaining unitaries to get the combinations of auxiliary edges. Since every unitary keeps the quantum state it acts on, either whole or scaled by $\cos\beta$, this process yields all matchings that satisfy Eq. (5.16).

Let $e_1, \ldots, e_N$ be any ordering of the $n \times m$ main edges, we want to prove that applying their corresponding unitaries in sequence will produce a superposition of all solutions to the first constraint in Eq. (5.16). We proceed by induction, where we will prove that applying U_{M,e_k} to the superposition of all feasible matchings using only edges $e_1, \ldots, e_{k-1}$, yields a superposition of all feasible matchings using only edges $e_1, \ldots, e_k$. Since the case for $k = 1$ starts with the quantum state in Eq. (5.17), $f(e_1)$ will always equal 1 and the result of applying U_{M,e_1} is a superposition of this trivial matching and the matching which includes only edge e_1. For $k > 1$, assume we have a superposition of all feasible matchings with edges $e_1, \ldots, e_{k-1}$ and notice that applying U_{M,e_k} to any one of these matchings can have two possible outcomes. Indeed if $f(e_k) = 0$ the matching remains unchanged. On the other hand, if $f(e_k) = 1$ the output will be a combination of the input matching and the matching that results from adding edge e_k. Therefore, U_{M,e_k} will retain all the matchings with only edges $e_1, \ldots, e_{k-1}$ and produce all feasible matchings that result from adding edge e_k, which yields a superposition over all feasible matchings that use only edges $e_1, \ldots, e_k$.

Now let $e'_1, \ldots, e'_m$ be any ordering of the m auxiliary edges. We prove in a similar way that given the superposition that results of the previous step, applying the auxiliary unitaries in sequence yields a superposition of all possible solutions to Eq. (5.16). Indeed, applying U_{M,e'_1} to each of the matchings in the superposition from the previous step gives either the same matching when $f(e'_1) = 0$ or a combination of the input matching and the one that results of removing edge e'_1 when $f(e'_1) = 1$, producing all feasible choices for e'_1. In the same manner, applying U_{M,e'_k} to the superposition with all feasible choices for edges $e'_1, \ldots, e'_{k-1}$ will produce a superposition with all feasible combinations for edges $e'_1, \ldots, e'_k$.

Vita

Bernardo Ameneyro was born in Mexico City, but moved to Colima in the Pacific coast of Mexico at an early age. There, he eventually attended the University of Colima and received a Bachelor of Science degree in Mathematics. Before graduating he visited Arizona State University for a summer research program on mathematical biology and decided to attend graduate school afterwards. He then started a graduate program at the University of Tennessee in Knoxville, where he studied quantum methods for topological data analysis in pursue of a Doctor of Philosophy degree in Mathematics.