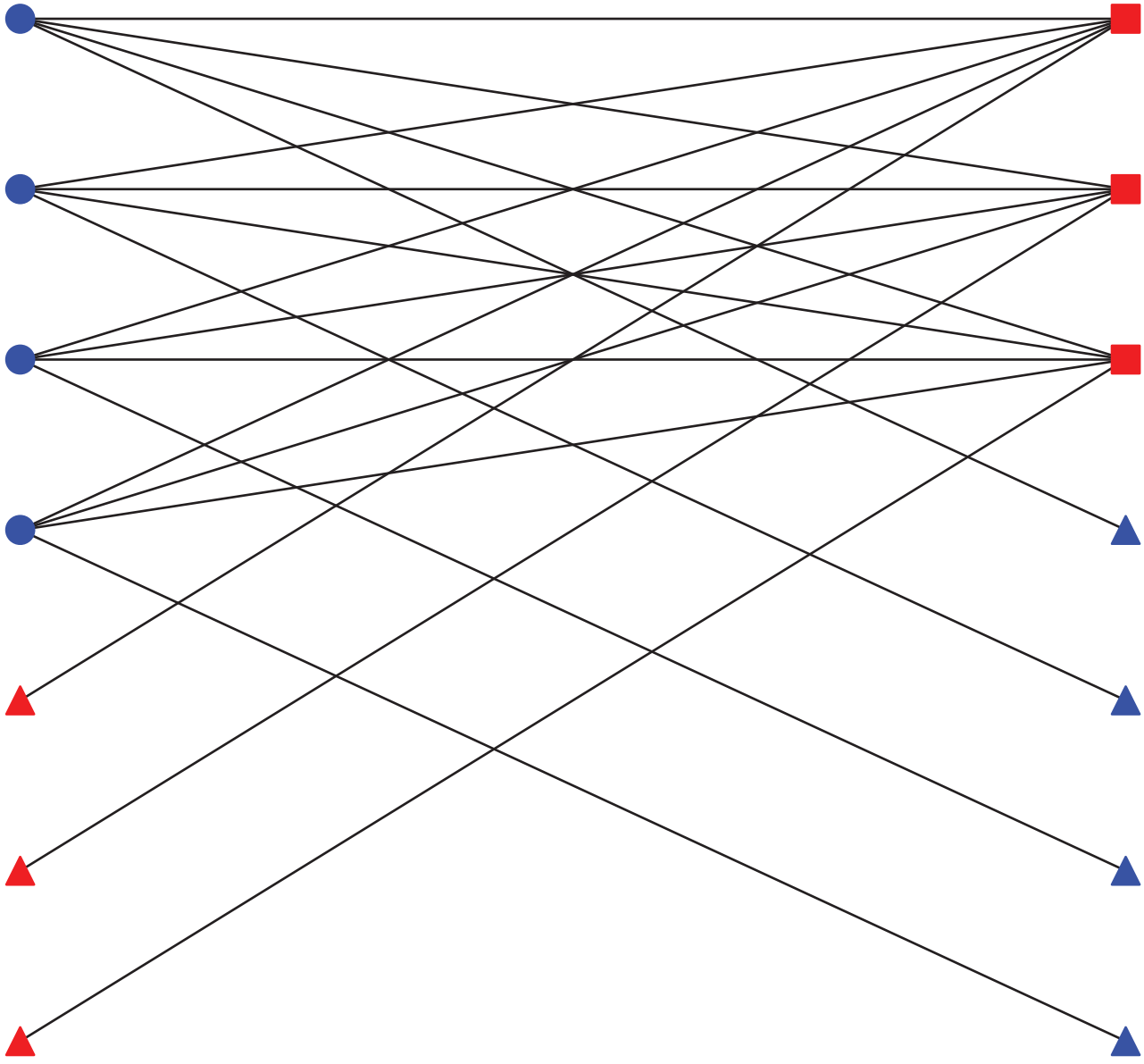
The Mathematics of Quantum-Enabled Applications on the D-Wave Quantum Computer



Jesse J. Berwald

Introduction

Over half a century ago, a groundbreaking technology, the microchip, started appearing in computers and research facilities around the world. Today there is no question of its importance. Yet in 1968, ten years after its invention, it was still a novelty to some: An IBM engineer famously asked, "But what... is it good for?"[1] Recent advances in the development of quantum computers in some ways mirror this evolution, though time, experience, and feverish media coverage ensure that few will ask the same naive question. The similarity comes from the observation that quantum computers are on a similar cusp, that of having broad societal impact, as the microchip was in the last century.

After some reflection, mathematicians and scientists may find themselves asking related questions. For instance, *What are quantum computers good for today? As a mathematician, what's in it for me?* and of course, *How do they do work?* Other than the last question, there are few definitive answers available. This article attempts to guide the reader towards her own intuition regarding the first two questions, but limits the "how" to a cursory glance and a host of references.

This article covers quantum computing from the angle of *adiabatic quantum computing* [7, 13], which has proven to have the shortest horizon to real-world applications, partly due to a slightly easier path to development[2] than alternative approaches such as *gate-model* quantum computers.

In this article we cover background on quantum annealing computing generally, the canonical problem formulation necessary to program the D-Wave quantum processing unit (QPU), and discuss how such a problem is compiled onto the QPU. We also cover recent joint work solving a problem from topological data analysis on the D-Wave quantum computer. The goal of the article is to cover the above from a mathematical viewpoint, accessible to a wide range of levels, and introduce as many people as possible to a small portion of the mathematics encountered in this industry.

Quantum Computing Background

Historical background. Richard Feynman is credited with the initial ideas for computing with quantum mechanics, presented in a seminal talk and subsequent paper from 1982, titled "Simulating Physics with Computers" [8]. Significant progress over the past decade has brought the quantum computer industry into what some term the *noisy intermediate-scale quantum* (NISQ) era [19]. While quantum computers have yet to show an undeniable advantage over classical systems, their theoretical advantages are well documented. Particularly noteworthy are Shor's algorithm [20] and Grover's search algorithm [10] for gate-model quantum computers. Quantum annealing, the model adopted by D-Wave Systems, also promises quantum speedup [22]. Already, in a number of narrowly defined use cases, improvements over classical computers have been observed on the D-Wave quantum computer [16, and references therein; 22].

Recalling Feynman's famous quote from [8], "Nature isn't classical... and if you want to make a simulation of Nature, you'd better make it quantum mechanical," precise control over annealing properties, as exists on the D-Wave quantum computer, allows for novel quantum material simulation. Recent work by Harris, et al. [11, and references therein] on phase transitions in spin glasses; and by King, et al. [15] on Kosterlitz-Thoulless phase transitions in exotic forms of matter show manifestations of the central thesis of Feynman's original paper.

Technical background. We absorb the majority of the technical description of quantum annealing into this section, saving a mathematical reformulation of certain aspects for the next two sections. The discussion below is general to quantum annealing.

To lay the groundwork, we briefly describe D-Wave's specific implementation of a QPU, though much of the technology described here is used by others exploring quantum computers using superconductivity. The D-Wave quantum computer is a programmable quantum computer whose QPU is composed of a network of superconducting flux qubits, each of which acts as a programmable Ising spin. Electrical current may travel in either direction in the qubit bodies, corresponding to up and down spins. Tunable mutual inductances between pairs of qubit bodies allow for in situ adjustment of magnetic coupling energy between these pairs. On the current model, the D-Wave 2000Q, the grid of qubits is arranged in tiles of $K_{4,4}$ bipartite graphs, known as unit cells. Qubits are connected across unit cells, giving each qubit a degree of at most six on the current hardware.

The D-Wave quantum computer implements a process known as quantum annealing [7, 13], which is independent of the chip architecture. The goal of a quantum annealing computer is to find a low-energy state of a *problem Hamiltonian*, H_P. The key is to initialize the system in a ground state of a driver Hamiltonian, H_D, that is computationally trivial to obtain, then evolve the system from that known state to the unknown ground state of H_P. The quantum adiabatic theorem guarantees that if the time evo-

Jesse J. Berwald is a Quantum Applications Engineer at D-Wave Systems. His email address is jberwald@dwavesys.com.

Communicated by Notices *Associate Editor Emilie Purvine.*

DOI: https://doi.org/10.1090/noti1893

[1] *Attributed to a particularly myopic engineer at the Advanced Computing Systems Division of IBM, 1968, commenting on the microchip.*

[2] *But by no means trivial, as D-Wave's nineteen-year journey can attest to.*

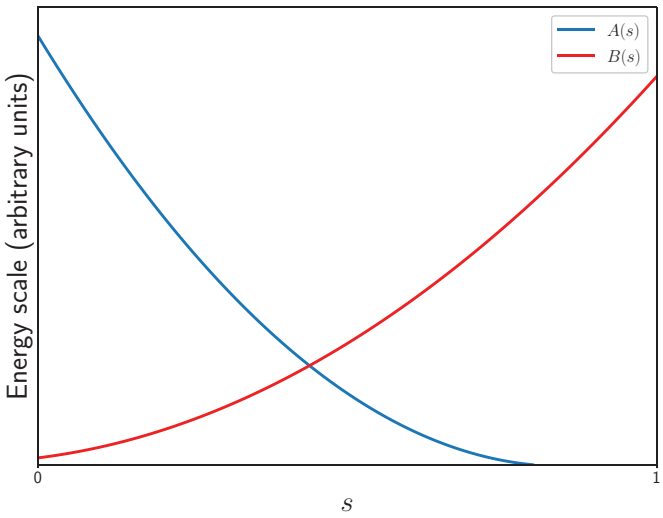


Figure 1. A typical annealing schedule. Scales are arbitrary. The parameter s evolves from 0 to 1, typically on the scale of microseconds. At $s = 0$ the system is in a ground state of the driver Hamiltonian, H_D, in which $A(0) \gg B(0)$ and is then evolved to a classical state such that $B(1) \gg A(1)$.

lution of a closed quantum system is slow enough, then the system will remain in its ground state throughout the process. Thus, at the end of a slow anneal process the ground state of the quantum state will also be the global minimum of the classical problem Hamiltonian. (See [7] for more details.) Note that the D-Wave computer actually runs the quantum annealing algorithm on an open quantum system, that is, one that is coupled to a thermal environment. Even with a slow annealing process, thermal excitations can result in a distribution of states with energies above the ground state. Sampling the system many times helps to mitigate such effects, which are fundamental to any open quantum system.

While not critical to understanding the mathematics in the following sections, we state the adiabatic theorem more precisely and describe its use in computation as this provides important context for the goal of a quantum annealing computer in general. Suppose the evolution of our quantum system is governed by the time-dependent Schrödinger equation [7],

$$i\frac{d}{dt}|\psi(t)\rangle = H(t)\,|\psi(t)\rangle\,,$$

where $|\psi(t)\rangle$ is a *state vector* in a complex n-dimensional Hilbert space. Studying the ground state of $|\psi(t)\rangle$ boils down to an eigenstate problem. We consider certain instantaneous eigenstates of H, $E_0(t) \le E_1(t) \le \cdots \le E_{n-1}(t)$, for fixed $0 \le t \le T$.[3] In the language of eigenstates, the adiabatic theorem guarantees that if $|\psi(0)\rangle$ is the ground state of $H(0)$, with eigenvalue $E_0(0)$, and if the spectral gap between the ground state and first excited state is positive for all $t \in [0, T]$, that is, $|E_0(t) - E_1(t)| > 0$, then the probability that $|\psi(T)\rangle$ is in the ground state is arbitrarily close to one. For technical caveats, see [7].

Consider the Hamiltonian,

$$H(s) = \frac{1}{2}A(s)H_D + \frac{1}{2}B(s)H_P, \tag{1}$$

that contains the problem Hamiltonian, H_P, a driver Hamiltonian, H_D, whose ground state is relatively easy to construct,[4] and $s \in [0, 1]$ (units are arbitrary). Typical curves, $A(s)$ and $B(s)$, governing the evolution, or annealing schedule, of the system are shown in Fig. 1. Eq. (1) allows us to leverage the adiabatic theorem for computation: At $s = 0$ the ground state is a global superposition of all computational basis states, obtained through application of a precise transverse magnetic field. From there the system is evolved to the ground state of the classical system, defined by H_P, at $s = 1$.

We now describe the problem Hamiltonian in greater detail. The structure of the operator

$$H_P = \sum_i h_i \sigma_i^z + \sum_{i,j} J_{i,j}\sigma_i^z\sigma_j^z \tag{2}$$

is defined physically by manipulating local, real-valued fields h_i and $J_{i,j}$ on the QPU; the σ^z are Pauli spin matrices. In quantum computation, the ith bit z_i is replaced by a *qubit*, $|z_i\rangle$. Each $|z_i\rangle$ represents the eigenstate of the σ_i^z operator, an observable state of the ith physical flux qubit in a D-Wave quantum computer. The eigenstates take values $|\uparrow\rangle$ or $|\downarrow\rangle$, with eigenvalues $+1$ and -1, indicating the "spin" of the quantum system to be either "up" or "down." Hence, a problem space with n qubits is spanned by a 2^n-dimensional Hilbert space.

Some Examples, Broadly Described

We now divorce ourselves from the physics and focus solely on the problem Hamiltonian, H_P, going forward. In the following subsections we discuss H_P from three distinct viewpoints. These short sections are meant to motivate mathematicians by highlighting areas of deeper mathematical formalism lurking behind quantum algorithms in general, and quantum annealing formulations in particular. In the first subsection, we consider H_P as a polynomial and point out some of the interesting consequences of this perspective. Next, we discuss one of the fundamental questions that arises with a limited, and rather sparse, chip topology: How does one fit the graphical structure of a general problem Hamiltonian onto the fixed graphical structure of the QPU? Lastly, we briefly follow up on the statement made in the subsection about sampling, and argue

[3] *The exact value of T is critical to the success of quantum annealing. A good derivation can be found in* [7] *and references therein.*

[4] *This in no way implies that construction of a quantum computing device is trivial, only that obtaining the ground state for H_D is easier than finding the ground state of H_P through non-adiabatic means.*

that there are possible benefits to this form of error correction.

Example 1: A polynomial viewpoint. Assume we have a quantum annealing computer, such as the D-Wave quantum computer, that is designed to seek a minimum energy solution to an Ising problem, Eq. (2), or equivalently a combinatorial optimization problem known as a *quadratic unconstrained binary optimization* (QUBO) problem. A simple transformation takes the variables in an Ising formulation to binary variables in a QUBO setting, using $y \mapsto \frac{y+1}{2}$, where y is a spin variable in H_P taking values in $\{-1, 1\}$.

We now describe the QUBO formulation of the problem Hamiltonian in more detail. We begin with a couple of definitions.

Definition 3. *Let $\mathbb{B}^n := \mathbb{Z}_2^n$, the n-dimensional hypercube of binary vectors.*

After a transformation of variables, we can define the problem Hamiltonian as a QUBO taking arguments from $\mathbb{B}^n$.

Definition 4. *Let*

$$H_P(\mathbf{x}) := \sum_{i=1}^{n} h_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} J_{i,j} x_i x_j$$

be a real-valued polynomial with arguments $\mathbf{x} \in \mathbb{B}^n$.

The polynomial H_P is an interesting mathematical object: The coefficients of H_P live in $\mathbb{Q}$, yet the variables are restricted to $\mathbb{B}^n$. To deal with this, at least notationally, we define a restriction to the polynomials over the rationals.[5]

Definition 5. *Define the set of polynomials with rational coefficient and binary variables, $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{B}^n$, as $\mathbb{Q}[\mathbf{x}|\mathbb{B}^n] \subset \mathbb{Q}[\mathbf{x}]$.*

Remark 6. *We can regard $\mathbb{Q}[\mathbf{x}|\mathbb{B}^n]$ as a quotient ring, $\mathbb{Q}[x_1, \dots, x_n]/\langle x_1^2 - x_1, \dots, x_n^2 - x_n\rangle$. The ideal $\langle x_1^2 - x_1, \dots, x_n^2 - x_n\rangle$ absorbs all polynomials in $\mathbb{Q}[\mathbf{x}]$ for which the binary constraint, $x_i^2 = x_i$, holds.*

This remark points to an elegant area of research. Dridi et al. [4] have leveraged the algebraic properties of H_P in developing a number of applications using the D-Wave quantum computer. For instance, in [4] they leverage Groebner bases to reduce the size of the problem prior to sending it to the quantum computer. In [5], Dridi et al. leverage computational algebraic geometry for the important problem of embedding the problem Hamiltonian onto the QPU, a topic described in the subsection "Technical background."

With Definition 3 and Definition 5, we can now state the problem solved by the quantum computer more precisely.

[5] *We could also take coefficients over $\mathbb{R}$.*

Definition 7. *Suppose we are given a Hamiltonian $H_P \in \mathbb{Q}[\mathbf{x}|\mathbb{B}^n]$ and a quantum computer, $\mathcal{Q}$, designed to implement the adiabatic theorem using the time-dependent Schrödinger equation Eq. (1). Then $\mathcal{Q}$ solves the combinatorial optimization problem*

$$\mathcal{H} \equiv \arg\min_{\mathbf{x}\in\mathbb{B}^n} H_P(\mathbf{x}), \tag{8}$$

given proper assumptions on the evolution of the system H.

The combinatorial optimization problem defined by $\mathcal{H}$ represents, abstractly, the problem to be solved. These are, in general, NP-hard problems, making the prospect of a quantum annealing computer that can solve the class of problems described by $\mathcal{H}$ enticing. Lucas [17] provides a thorough overview of methods for formulating a number of NP-hard problems as QUBOs.

Example 2: Compiling H_P – a graph minor embedding problem. Much like a classical computer converts high-level, abstract, and human-readable languages to machine instructions, Eq. (8) must be converted to a *quantum machine instruction* (QMI) that will run on the quantum computer. There are numerous steps in this process, one of which, *embedding*, we touch on briefly in this section. It is convenient to view the problem Hamiltonian, H_P, as a weighted graph. In the subsection "The Wasserstein Graph as a QUBO," we construct a specific problem Hamiltonian to make this connection more clear. Define $G = \langle V, E\rangle$, where the node set

$$V = \{(x_1, h_1), \dots, (x_n, h_n) \mid h_i \neq 0\}$$

is composed of nodes that are in direct correspondence with each binary variable x_i, where $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{B}^n$. Each node is weighted by the bias on the qubit, h_i. Similarly, the edge set is composed of weighted edges defined by the coupling terms in H_P, so that

$$E = \{(x_i, x_j, J_{i,j}) \mid x_i, x_j \in V \text{ and } J_{i,j} \neq 0\}.$$

This definition of E encodes the variable coupling in H_P.

The graph G must be embedded onto the hardware to solve H_P. Embedding the *logical graph*, G, onto the *hardware graph*, K, amounts to finding a *minor embedding*. A *minor* of a graph K is a subgraph of K obtained by contracting or deleting edges, and omitting isolated vertices. A minor embedding is a function that maps the vertices of G to the power set of the vertices of K,

$$\psi : V_G \to 2^{V_K},$$

such that for each $u \in V_G$, the graph induced in K by $\psi(u) \in V_K$ is connected. These connected components within the hardware graph are termed *chains*. Embeddings for which $\psi(u)$ is a singleton for all u are called *native* embeddings. Lastly, there exists an edge between $\psi(u)$ and $\psi(v)$ whenever u and v are adjacent in the logical graph. The map ψ is the minor embedding we seek. Whether

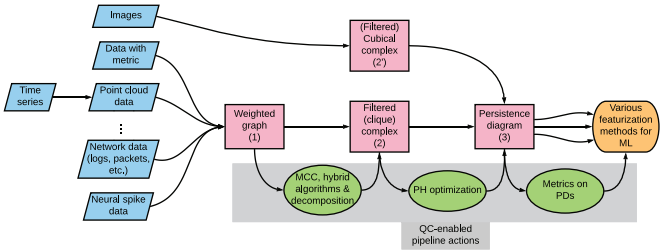


Figure 2. The topological data analysis pipeline for persistent homology (PH). Transformations containing portions amenable to quantum computation are highlighted in the green ovals.

G can be embedded as a minor contained in K is known to be NP-hard. Elegant, polynomial-time algorithms exist for embedding problem Hamiltonians onto the the QPU architecture [2]. The more efficient the embedding—the shorter the chains—the larger the problem that can be solved on the QPU.

Example 3: Distributions of solutions through sampling. Quantum computers are inherently probabilistic, with samples being drawn from an approximate Boltzmann distribution in the case of D-Wave's implementation. Hence, it is necessary to sample the energy landscape of the problem many times to obtain a distribution of solutions. This is one form of error correction. For example, Shor's algorithm [20] is designed to return the prime factors of a number with *high probability*. Repeated sampling will provide many possible factor pairs, leaving the final confirmation of a much smaller set of possibilities up to a classical computer.

On a quantum annealing computer, once the energy landscape is defined by H, and the QUBO is embedded on the QPU, a collection of hundreds or thousands of samples, $\{\mathbf{x}_i\}$, can be gathered quickly by repeatedly annealing the problem and reading out the answer. Each of the $\mathbf{x}_i$ could be located at or near a local minimum, using Hamming distance as a metric; or one may find solutions with similar energy at opposite corners of the hypercube $\mathbb{B}^N$. Combined with reverse annealing, a local search capability available on the D-Wave quantum computer, this feature provides a powerful avenue to explore regions of high probability (low energy) in multimodal systems, especially in the realm of neural networks and genetic algorithms as discussed in [3]. We provide a brief example of the sampling aspect of quantum computation applied to Wasserstein distance in the subsection "Sampling solutions."

A Mathematical Application

Many users of the D-Wave quantum computer in recent years have focused on hybrid workflows. In the context of quantum computing, these are software pipelines that use classical computers for a majority of their work, inserting quantum computation at compute-intensive bottlenecks [18, 24]. This is a fruitful area to focus research efforts as there will always be vast amounts of pre- and post-processing within real-world pipelines. Much of that processing is not amenable to quantum acceleration, yet alleviating bottlenecks has the potential to yield significant computational gains.

In general, users seeking quantum speedup tend to isolate the tight "inner loop" of their problem, the bottleneck where computing this loop in one step will reduce the complexity of the problem by orders of magnitude. Recent work has looked at specific methods to speed up the inner loop of topological data analysis pipelines [12, 25]. In Fig. 2, we show a typical topological data analysis (TDA) pipeline. The blue boxes on the left represent various potential data sources, while the pink boxes in the middle, labeled 1, 2, 2′, and 3, show computational bottlenecks in the TDA pipeline. The green ovals highlight the algorithms that could potentially run on the quantum computer to alleviate bottlenecks. Lastly, the final box on the right assumes further processing using the features extracted by the TDA pipeline. We used the scikit-tda package for the TDA portion of our analysis. In the next two subsections we provide a brief summary of persistent homology (PH) and Wasserstein distance. In the subsection "The Wasserstein Graph as a QUBO," we translate the Wasserstein distance to a QUBO to compare the topological signature of point clouds. While not a bottleneck per se—polynomial-time algorithms exist to compute Wasserstein distance—it provides an instructive case for translating a general problem to a QUBO and hence into a quantum annealing framework. Our work shows the interplay between the underlying mathematics of the Wasserstein distance, the construction of a QUBO to solve the combinatorial optimization problem in Definition 7, and provides an example of the additional configurations returned as a result of the probabilistic nature of quantum computation.

Persistent homology. Our interest in computing Wasserstein distances is rooted in the search for robust features in noisy, real-world data. This section provides a brief overview of the PH pipeline [6]. PH, one of the most widely used tools in the field of TDA, is based on the idea that analyzing noisy data through a sequence of resolutions enables one to robustly identify and quantify structure in such data. Suppose we have a set of data points in a metric space. PH uses a filtered topological space, such as a simplicial or cubical complex, to study these data at various resolutions. A typical assumption in PH is that the data under study represents a random sample of points taken from some distribution on a manifold embedded in a nice ambient space like $\mathbb{R}^d$. It is the job of PH to discover the homology of the underlying manifold from the data.

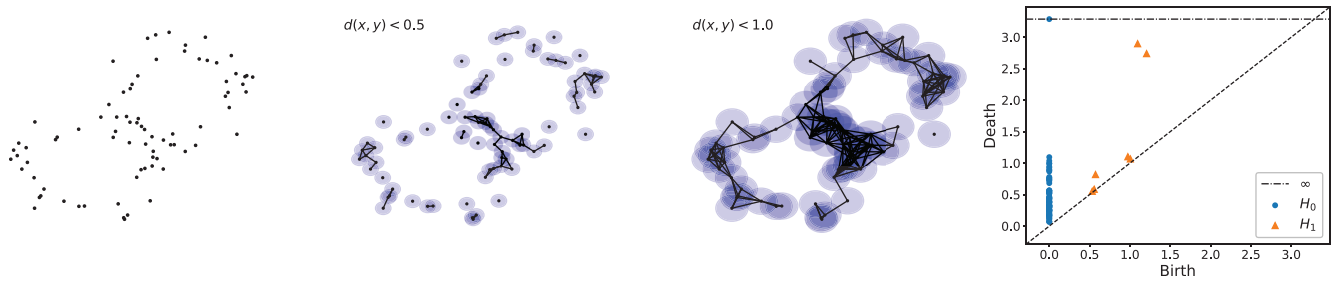


Figure 3. Snapshots of a filtration, with the associated persistence diagram on the right. The original data set in the left figure is generated by sampling points from two circles. A couple of steps in the filtration are shown with disks of radius $r = 0$, $r = 0.25$, and $r = 0.65$. For H_0, at $r \approx 0.55$ we are left with a single connected component that persists forever. In the case of H_1, the persistence diagram shows two long-lived generators, both born shortly after $r = 1$. When r is just under 3 the central holes fill in completely, causing each to become homeomorphic to a solid disk, and killing off those generators.

As an example, consider the point cloud, L, on the left of Fig. 3. We take this as a collection of points in $\mathbb{R}^2$ sampled from two copies of S^1 and then perturbed. The concept of resolution is parameterized through the length parameter r. Given any r, we produce a *Vietoris-Rips complex*, V_r, from the data. A 1-simplex, or edge, is added between two points, $x, y \in L$, whenever $d(x, y) < r$. Supposing x and y are not part of the same connected component already, this effectively merges two connected components into one, with the corresponding disappearance of a generator in the zeroth simplicial homology group, H_0. Higher-dimensional simplices of dimension $k > 0$ are also included in our accounting when all $k + 1$ vertices of the simplex, $\{x_0, \ldots, x_k\}$, are within distance r of each other. As is clear from Fig. 3, we get that $V_r \subset V_{r'}$, for $r < r'$. This inclusion allows the PH algorithm to track the birth and death of homology generators in H_k as r grows (see [6]).

Given a fixed dimension k, we obtain a set (possibly a multiset) of intervals, $I_k = \{(b, d) \mid b, d \in \mathbb{R}\}$, defining the *birth* and *death* (appearance and disappearance) of homology classes in H_k as r increases. The lifetime, $b - d$, of a generator is used to infer the robustness of the corresponding topological feature. We visualize each I_k by plotting the points on a *persistence diagram*. The persistence diagram on the right-hand side of Fig. 3 shows intervals for H_0 and H_1, denoted by ● and ▲, respectively. All connected components are born at $r = 0$, and merge into one component at $r \approx 1.2$. This component lives forever, as indicated by the line representing infinity. The diagram for H_1 indicates two long-lived homology classes at $(1.2, 2.7)$ and $(1.1, 2.9)$. These correspond to generators for the two large circles. The points just off the diagonal represent short-lived generators that correspond to small, insignificant cycles that had short lifetimes. These are often treated as noise. The intuition underlying PH is that a point in the persistence diagram far from the diagonal represents a homology class that appeared early in the filtration and died late. Such a homology class represents a robust topological feature within the noisy data.

Due to its abstract nature, PH tends towards a broad user base, with successful applications showing up in a wide variety of fields. For instance, mathematics and material engineering merge nicely in the analysis of time series obtained from rotating machines in [14]; financial crashes produce *persistence landscapes* different from stable market periods, as shown in [9]; and in [21] the authors describe a robust method for detecting holes in sensor networks.

All of these studies rely on the ability to understand trends and structures in data, which in turn requires a metric with which to compare two or more data sets. Two primary metrics used in PH are the *bottleneck distance* and the *Wasserstein distance*. For a finite dimension k, these metrics compute the distance between two data sets by comparing their persistence diagrams. We focus on the Wasserstein distance as we formulate an example of a quantum annealing-enabled algorithm below.

Wasserstein distance as a graph matching problem. In full generality, a persistence diagram is a finite multiset of points in the plane. First, define the region in the plane occupied by the persistence intervals as $\mathbb{R}^2_+ := \{(b, d) \mid d > b \text{ and } b \geq 0\}$. Second, for technical reasons, each diagram also includes an additional set of countably infinite copies of each point on the diagonal, $\Delta := \{(d, d) \mid d \geq 0\}$. The reason for this becomes clear when we define the Wasserstein distance for discrete data sets. Combining these two sets, a persistence diagram is a collection of points $\{a_1, \cdots, a_n\} \cup \Delta$, where each $a_i \in \mathbb{R}^2_+$ may occur repeatedly.

As mentioned above, in the analysis of data sets we are often interested in the distance between two persistence diagrams, X and Y. The metric used is a discrete analog of the more general Wasserstein metric, which computes the minimal work required to transport the mass of one (continuous) probability distribution to another probability distribution. In the discrete case, we are tasked with matching points from opposing diagrams most efficiently so as

to minimize the work[6] necessary to transport the configuration of points in X to match the configuration of points in Y. In this case, we model the p-Wasserstein distance as

$$d_p(X,Y) = \inf_{\phi: X \to Y} \left(\sum_{a \in X} \|a - \phi(a)\|_q^p \right)^{1/p}, \qquad (9)$$

where the infimum is taken over all bijections ϕ between points in diagrams X and Y, $p, q \in [1, \infty)$, and $\|\cdot\|_q$ is the Euclidean q-norm [1]. It is convenient to use $p = q = 2$. Given a specific ϕ, define the *cost* of the matching induced by ϕ as

$$C(X,Y) = \sum_{a \in X} \|a - \phi(a)\|_q^p, \qquad (10)$$

where we omit reference to ϕ, p, and q on the left-hand side.

In practice, this is often solved by translating the bijection problem to a matching problem on a bipartite graph. Suppose X and Y are two persistence diagrams. We describe a method to represent the possible bijections between the diagrams X and Y as a weighted bipartite graph representation, that we then use to reformulate the cost function Eq. (10) as a portion of a problem Hamiltonian.

First, denote by X_0 and Y_0 the off-diagonal points in X and Y. Define the orthogonal projections of points in X_0 and Y_0 onto the diagonal Δ by X_0' and Y_0', respectively. In the discrete setting, unequal numbers of points or indivisibility of mass make some matchings infeasible or impossible. In such cases the diagonal acts to absorb points in X_0 or Y_0 that cannot be matched. Then we can denote our diagrams by $X = X_0 \sqcup X_0'$ and $Y = Y_0 \sqcup Y_0'$, where we have abused notation by redefining X and Y to only consider the finite collection of points we will use to compute the Wasserstein distance. We now specify the graph used to construct a QUBO that we embed on the D-Wave QPU.

Definition 11. *Define the* **Wasserstein Graph** $W := \langle X \cup Y, E \rangle$, *where the weighted edges* $E := E_1 \cup E_2 \cup E_3$ *such that*

$$\begin{aligned} E_1 &= \{(u, v, \theta_{uv}) \mid u \in X_0, v \in Y_0\} \\ E_2 &= \{(u, u', \theta_{uu'}) \mid u \in X_0, u' \in X_0'\} \\ E_3 &= \{(v, v', \theta_{vv'}) \mid v \in Y_0, v' \in Y_0'\}, \end{aligned}$$

and the edge weights are defined by

$$\theta_{u,v} = \begin{cases} \|u - v\|_\infty & \text{if } v = u' \\ \|u - v\|_q & \text{otherwise.} \end{cases} \qquad (12)$$

We abbreviate edge membership, writing $uv \in E$ *for the edge* (u, v, θ_{uv}).

[6] *In the general case, the mass is variable, so transport between distributions involves the traditional* work = mass × distance *formulation. We still use this terminology, except* mass = 1, *so we neglect it.*

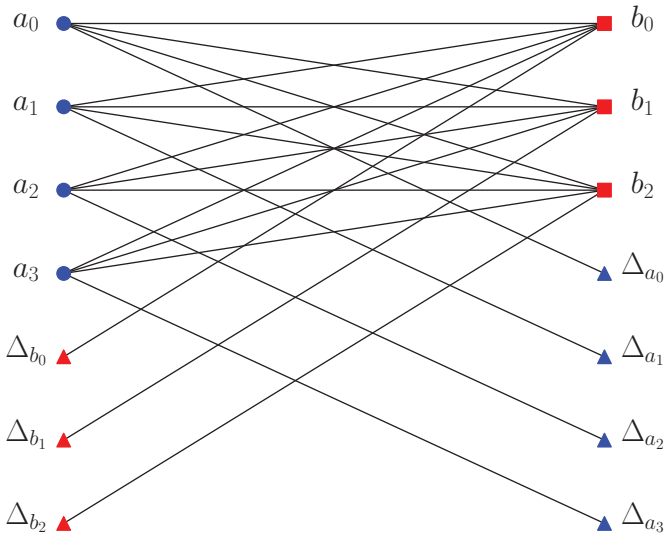


Figure 4. Bipartite graph with nodes on the left from X_0 and Y_0', respectively (labeled by ●'s and ▲'s); and nodes on the right from Y_0 and X_0', respectively (labeled by ■'s and ▲'s). The graph is complete only among off-diagonal points. (Edge weights are omitted to reduce clutter.)

Fig. 4 shows an example **Wasserstein Graph** for two persistence diagrams. In this case, $|X_0| = 4$ and $|Y_0| = 3$. Notice that off-diagonal and diagonal projection nodes are placed on opposing sides of the bipartite graph. Only the subgraph containing off-diagonal points in X_0 and Y_0 is complete. This limits the number of possible bijections between the two diagrams, a fact that helps to reduce the complexity of the QUBO as will be seen in the next section. We do not label edge weights in this example.

The Wasserstein Graph as a QUBO. The **Wasserstein Graph** provides a succinct example of how one might bridge mathematics and quantum computers. Given two persistence diagrams, we construct a QUBO from the associated **Wasserstein Graph**, W. The approach is straightforward. The QUBO must encode an objective function that minimizes the work, or cost, C, by "turning on" specific edges, while also enforcing certain constraints.

To make this precise, first we enumerate the edges that will map to the logical qubits. The number of edges in W is $N = mn + m + n$, where $m = |X_0|$ and $n = |Y_0|$. The weighted edges in W map to a set of tuples, $\{(x_{uv}, \theta_{uv}) \mid x_{uv} \in \mathbb{Z}_2\}$. An edge uv is *activated* if $x_{uv} = 1$, otherwise it is *inactivated*. We now rewrite Eq. (10) in terms that include the logical qubits,

$$H_{\text{cost}}(\mathbf{x}) = \sum_{uv \in E} \theta_{uv} x_{uv},$$

where $\mathbf{x} = (x_{uv}) \in \mathbb{B}^N$. Each $\mathbf{x}$ equates to a particular matching between diagrams induced by ϕ.

To avoid the case where setting $\mathbf{x} = \mathbf{0}$ minimizes the objective, we must add constraints. Each $u \in X_0 \cup Y_0$ must

have degree exactly one in order to avoid duplication of mass and to assure that every point is transported somewhere, either between diagrams or projected to the diagonal. The diagonal nodes can have degree zero or one, depending on whether or not their off-diagonal partner connects to another off-diagonal node. From these requirements we obtain

$$H_{\text{constraint}}(\mathbf{x}) = \sum_{u \in X_0} \left(1 - \sum_{uv \in E_1 \cup E_2} x_{uv}\right)^2 + \sum_{v \in Y_0} \left(1 - \sum_{uv \in E_1 \cup E_3} x_{uv}\right)^2,$$

where the two outer summations consider only edges emanating from off-diagonal nodes. The summand, $(1 - *)$, enforces the requirement that off-diagonal nodes have degree one. If each node in $X_0 \cup Y_0$ has degree one, then $H_{\text{constraint}} = 0$; otherwise, one or both of the terms in the expression will be positive, adding a penalty to the objective function.

The restriction of edges in $H_{\text{constraint}}$ reduces the complexity of the discrete problem significantly in both the classical and quantum computing cases by limiting the number of possible bijections. It is especially beneficial in quantum situations where physical qubits are at a premium.

Combining H_{cost} and $H_{\text{constraint}}$, we arrive back at the general definition of the problem in Eq. (8) with

$$H_P := H_{\text{cost}} + \gamma H_{\text{constraint}}, \tag{13}$$

where we have inserted the Lagrangian parameter γ to balance the magnitude of the terms. Quantum computers are analog physical devices that have limited accuracy and ranges for their parameters. Thus, determining correct parameters is essential for accurate solutions.

At this point, we can study $H_P \in \mathbb{Q}[\mathbf{x}|\mathbb{B}^n]$, and also note that the linear and quadratic terms in H_P define a logical graph G as discussed in the section "Example 2: Compiling H_P – a graph minor embedding problem." Quantum annealing requires that H_P be compiled to a QMI, so at this point software converts the logical graph to the hardware graph—the grid of qubits described earlier in this section—through a minor embedding.

Eq. (13) contains an important question. While the determination of the objective function and its constituent constraints is straightforward, it is not entirely clear that minimizing Eq. (13) yields the same value as Eq. (9). In [1, Sec. 4], we prove that minimizing Eq. (13) computes the p-Wasserstein distance, with the caveat that the minimizer of $\mathcal{H}$ provides an equivalent solution to Eq. (9) iff γ satisfies

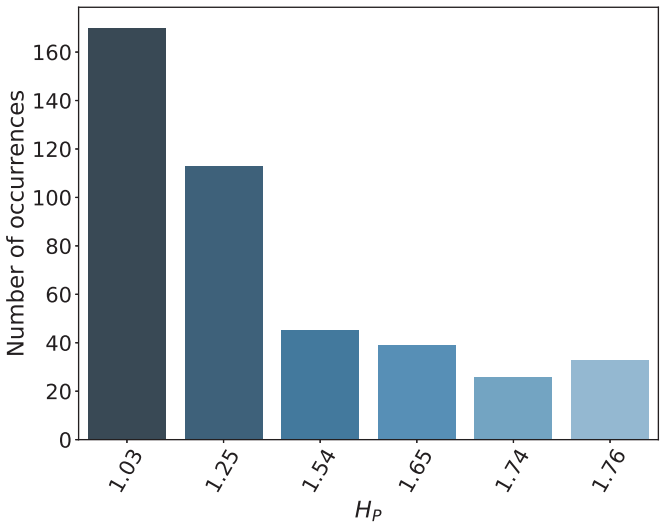


Figure 5. Frequency of costs, H_P, computed on the D-Wave quantum computer for different matchings between two persistence diagrams. The diagrams were computed from a torus and an annulus. For this example, $\gamma = 1$. Each distance on the x-axis corresponds to a specific $\mathbf{x} \in \mathbb{B}^N$. The occurrences on the y-axis denote the number of times a unique $\mathbf{x}$ was sampled. The matching corresponding to the Wasserstein distance, 1.03, is sampled most frequently.

$$\gamma > \max_{uv \in E} \theta_{uv}.$$

By keying the analysis of the quantum computational problem off of a known computable metric, we are able to determine exactly how to set hyperparameters properly. By contrast, it is often necessary in general problems to perform searches of the parameter space before a reasonable energy landscape, defined by H_P, can be processed accurately by the QPU.

Sampling solutions. As mentioned in the section "Example 3: Distributions of solutions through sampling," solutions returned by a quantum computer are probabilistic in nature. We obtain many samples by running the annealing procedure multiple times. In Fig. 5, the suite of samples we gather represents the cost of different possible matchings between the persistence diagrams of a torus and an annulus. We use $H_P(\mathbf{x})$ to compute the cost. The low-energy solutions represent valid matchings that do not violate constraints, e.g., $H_{\text{constraint}}(\mathbf{x}) = 0$. The minimum cost, 1.03, is the square of the Wasserstein distance, i.e., the infimum over all the possible valid matchings.

The different matchings and distances represent a distribution of low-energy solutions, each sample of which comes from a different ϕ and produces a different cost using Eq. (10). In fact, Fig. 5 represents a distribution of ϕ's sampled from an approximate Boltzmann distribution. In future work we plan to study the implications for statistics on persistence diagrams, along the lines outlined by Turner et al. in their work on Fréchet means in [23].

Conclusion

In this article we covered a number of mathematical aspects of quantum computing from a high level. Nevertheless, we have hardly scratched the surface of the subject. Interesting problems can be found in many different areas, from physical applications, to theoretical improvement of embedding QUBOs on the QPU, to decomposition of large problems into QPU-sized chunks.

Mathematicians and physicists have spent many years developing algorithms designed to run faster on quantum computers. The subtlety is that many of these methods require far more qubits than are available even on the 2000-qubit D-Wave quantum computer. Luckily, even before we reach that technological state, there is still exciting and effective research that can be accomplished in the current NISQ era. We hope that in touching on the mathematics involved in programming a D-Wave quantum computer we motivate interest in the myriad problems stemming from using this novel computational tool.

Acknowledgments

The author gratefully acknowledges support from the Institute for Mathematics and its Applications at the University of Minnesota. For helpful suggestions and discussions, the author would also like to thank T. Lanting, J. Gottlieb, E. Munch, S. Reinhardt, A. King, and the anonymous reviewers.

References

[1] Berwald J J, Gottlieb J M, Munch E. Computing Wasserstein Distance for Persistence Diagrams on a Quantum Computer, *ArXiv e-prints*, 2018-09, available at 1809.06433. https://arxiv.org/abs/1809.06433

[2] Cai J, Macready WG, Roy A. A practical heuristic for finding graph minors, 2014-06-10, available at 1406.2741v1.

[3] Chancellor N. Modernizing quantum annealing ii: Genetic algorithms with the inference primitive formalism, 2016-09-19, available at http://arxiv.org/abs/1609.05875v5.

[4] Dridi R, Alghassi H. Prime factorization using quantum annealing and computational algebraic geometry, *Scientific Reports*, no. 1 (7), 2017feb.

[5] Dridi R, Alghassi H, Tayur S. A novel algebraic geometry compiling framework for adiabatic quantum computations, 2018-10-02, available at http://arxiv.org/abs/1810.01440v1.

[6] Edelsbrunner H, Harer JL. *Computational topology: An introduction*, American Mathematical Society, Providence, RI, 2010. MR2572029

[7] Farhi E, Goldstone J, Gutmann S, Lapan J, Lundgren A, Preda D. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem, *Science*, no. 5516 (292):472–476, 2001, DOI 10.1126/science.1057726. MR1838761

[8] Feynman RP. Simulating physics with computers, *Internat. J. Theoret. Phys.*, no. 6-7 (21):467–488, 1981/82, DOI 10.1007/BF02650179. Physics of computation, Part II (Dedham, Mass., 1981). MR658311

[9] Gidea M, Katz Y. Topological data analysis of financial time series: landscapes of crashes, *Phys. A* (491):820–834, 2018, DOI 10.1016/j.physa.2017.09.028. MR3721543

[10] Grover LK. Quantum mechanics helps in searching for a needle in a haystack, *Physical Review Letters*, no. 2 (79):325–328, 1997jul.

[11] Harris R., Sato Y., Berkley A. J., Reis M., Altomare F., Amin M. H., Boothby K., Bunyk P., Deng C., Enderud C., Huang S., Hoskinson E., Johnson M. W., Ladizinsky E., Ladizinsky N., Lanting T., Li R., Medina T., Molavi R., Neufeld R., Oh T., Pavlov I., Perminov I., Poulin-Lamarre G., Rich C., Smirnov A., Swenson L., Tsai N., Volkmann M., Whittaker J., Yao J. Phase transitions in a programmable quantum spin glass simulator, *Science*, no. 6398 (361):162–165, 2018jul.

[12] Hylton A, Sang J, Henselman-Petrusek G, Short R. Performance enhancement of a computational persistent homology package. *2017 IEEE 36th international performance computing and communications conference (IPCCC)*: IEEE; 2017dec.

[13] Kadowaki T, Nishimori H. Quantum annealing in the transverse ising model, *Physical Review E*, no. 5 (58):5355–5363, 1998nov.

[14] Khasawneh FA., Munch E. Topological data analysis for true step detection in periodic piecewise constant signals, *Proc. A.*, no. 2218 (474):20180027, 24, 2018. MR3883592

[15] King AD., Carrasquilla J, Raymond J, Ozfidan I, Andriyash E, Berkley A, Reis M, Lanting T, Harris R, Altomare F, Boothby K, Bunyk PI., Enderud C, Fréchette A, Hoskinson E, Ladizinsky N, Oh T, Poulin-Lamarre G, Rich C, Sato Y, Smirnov AYu., Swenson LJ., Volkmann MH., Whittaker J, Yao J, Ladizinsky E, Johnson MW., Hilton J, Amin MH. Observation of topological phenomena in a programmable lattice of 1,800 qubits, *Nature*, no. 7719 (560):456–460, 2018aug.

[16] Li RY., Felice RD, Rohs R, Lidar DA. Quantum annealing versus classical machine learning applied to a simplified computational biology problem, *npj Quantum Information*, no. 1 (4), 2018feb.

[17] Lucas A. Ising formulations of many NP problems, *Frontiers in Physics* (2), 2014.

[18] Mott A, Job J, Vlimant J-R, Lidar D, Spiropulu M. Solving a higgs optimization problem with quantum annealing for machine learning, *Nature*, no. 7676 (550):375–379, 2017oct.

[19] Preskill J. Quantum computing in the NISQ era and beyond, *Quantum* (2):79, 2018aug.

[20] Shor PW. Algorithms for quantum computation: discrete logarithms and factoring, 35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994); 1994:124–134, DOI 10.1109/SFCS.1994.365700. MR1489242

[21] de Silva V, Ghrist R. Coverage in sensor networks via persistent homology, *Algebr. Geom. Topol.* (7):339–358, 2007, DOI 10.2140/agt.2007.7.339. MR2308949

[22] Somma RD., Nagaj D, Kieferová M. Quantum speedup by quantum annealing, *Physical Review Letters*, no. 5 (109), 2012jul.

[23] Turner K, Mileyko Y, Mukherjee S, Harer J. Fréchet means for distributions of persistence diagrams, *Discrete Comput. Geom.*, no. 1 (52):44–70, 2014, DOI 10.1007/s00454-014-9604-7. MR3231030

[24] Ushijima-Mwesigwa H, Negre CF. A., Mniszewski SM. Graph partitioning using quantum annealing on the d-wave system, 2017-05-04, available at `http://arxiv.org/abs/1705.03082v1`.

[25] Zomorodian A. The tidy set: a minimal simplicial set for computing homology of clique complexes [extended abstract], Computational geometry (SCG'10); 2010:257–266, DOI 10.1145/1810959.1811004. MR2742959

Jesse J. Berwald

Credits

Figures 1–5 are by the author.
Author photo is by Sarah Berwald.