Homology Computation of Large Point Clouds Using Quantum Annealing

Raouf Dridi, Hedayat Alghassi
Abstract

Homology is a tool in topological data analysis which measures the shape of the data. In many cases, these measurements translate into new insights which are not readily available by other means. To compute homology, we rely on mathematical constructions which scale exponentially with the size of the data. Therefore, for large point clouds, the computation is infeasible using classical computers. In this paper, we present a quantum annealing pipeline for computation of homology of large point clouds. It is designed to work concurrently with resizable cloud computing platforms. The pipeline takes as input a witness graph approximating the given point cloud. It uses quantum annealing to compute a clique covering of the graph and then uses this cover to construct a Mayer-Vietoris complex. The pipeline terminates by performing a simplified homology computation of the Mayer-Vietoris complex in parallel. We have designed three different clique coverings and their quantum annealing formulation with which our algorithm exhibits an exponential speed-up over classical implementations. In fact, not only the computation is simplified but also the simplicial complex construction itself is greatly simplified. We have also included tests using D-Wave 2X quantum processor.

Keywords: homology, topological data analysis, quantum annealing

1 Introduction

The abundance of data, of all sorts, represents undoubtedly an exceptional and unprecedented amount of knowledge for humanity to benefit from. Yet, the extent of such abundance combined with the inherent complexity of the data make the deciphering and extraction of this knowledge tremendously difficult. Therefore, today, a data scientist faces two non trivial challenges. First, design models and algorithms appropriate to the complexity of the data and then leveraging them to large scales. In our opinion, the appropriate algorithmics are to be found in advanced mathematics where concepts like “correct glueing of local statistical information into a global insight” are captured, precisely defined and solved. Topological data analysis (TDA) is one of these mathematics. It uses algebraic topology, a branch of modern mathematics which investigates topological features through algebraic lenses. In this work, we leverage TDA algorithms to large scales using quantum annealing.

The main concept in TDA is homology which is an invariant that consists of a sequence of vector spaces measuring the shape of a given point cloud. These measurements usually translate into valuable insights about the data which are not readily available by other means. An excellent survey of TDA and its applications can be found in [7] and [8]; also we refer to [9] and [10] for the notions of homotopy equivalence, simplicial complexes and their homologies. The objective of the paper is to propose and test a quantum algorithm for computing the homology of large point clouds.

Our approach is based on the parallelization of homology computation using Mayer-Vietoris complexes. This approach starts with approximating the given point cloud $X$ with a simplicial complex $K$. We choose here to use witness complexes for this first step. This particular instance of simplicial complexes reduces both the dimensionality and size of the data while preserving the topology. Figure 1, below, shows two examples of such complexes (precise definitions are given in the next section).

Next, we construct a Mayer-Vietoris complex on top of $K$ as in ([11], [12]). This consists of covering the witness complex $K$ with smaller subcomplexes and then blowing up their overlaps. The blow-up operation is a homotopy equivalence which implies that the two complexes have the same homology. The parallelization of the homology computation of the Mayer-Vietoris complex is straightforward.

The efficiency of this parallelization depends on the covering step, which is done in two stages. First, cover the edges of the 1-skeleton of $K$ and then extend this cover to the simplicial complex $K$. The extension is an inexpensive operation and can easily be parallelized. The important problem is then to cover the 1-skeleton (in a way which makes the subsequent computations efficient). This problem has two important parameters: the structure of the covering subgraphs and their number. Here, we investigate the structure rather than the number of subgraphs which is justified by the availability of resizable cloud computing where the number of slave machines is not a practical constraint. The problem reduces then to covering the 1-skeleton with subgraphs whose structure minimizes the computational load assigned to each slave machine of the computing cluster. Solving this problem is NP-hard and is the focus of the present paper.
Figure 1: The left graph is the 1-skeleton of the witness complex of the unit torus with 100 landmarks and 100 000 data points. The corresponding witness complex is the 1-skeleton in addition to all its cliques. The right graph is the 1-skeleton of the witness complex (with 80 landmarks) of the NKI data set ([?]) that contains 24 496 gene expressions of 295 female patients diagnosed with breast cancer.

We solve the above problem using quantum annealing and the latest D-Wave Systems’ D-Wave 2X quantum processor in particular. We begin by reformulating the work of [?] in the language of quantum annealing, which is quadratic unconstrained binary optimization (QUBO), a special case of quadratic pseudo-boolean optimization---see [?] and [?] for more on the subject. This does not solve the covering problem as it is stated above but provides a way to cover any simplicial complex, not necessarily a witness complex. We then present three novel approaches that take advantage of the clique-based structure of the witness complex. The first uses vertex clique cover, the second is based on edge clique cover, and the third is an iterative method to compute a specific edge clique cover, that we call edge disjoint edge clique cover. Since the scenario involves a resizable computing cluster, the changing number of slave machines is not problematic. We express the three methods as QUBOs. We then explain that these three new covering approaches reduce the computational load assigned to the local machines of the computing cluster. Indeed, our pipeline exhibits an exponential speed-up over classical implementations for dense graphs. We also explain how just the 1-skeleton of the initial simplicial complex $K$ suffices to carry on the homology computation (in contrast to the $k$-cut method [?] where prior construction of $K$ is needed to proceed). This is because a large portion of simplices (cliques) is computed with the covering step i.e., a large fraction of simplices are confined inside the covering cliques.

The practical end result of this paper is a pipeline that a data scientist can use as follows. First, represent the given point cloud as a witness graph which reduces the dimensionality and size of the data. At this stage, the graph can be visualized for possible early insights. Next, use the quantum annealer to cover the graph. The cover can be also analyzed for potential additional insights. We usually track the dimensions of the homology spaces (called Betti numbers) over a range of values of a persistent parameter $\epsilon$ (see definition next section). We compute the so-called bar codes. Meaningful insights persist over a long range of $\epsilon$; on the contrary, noise don't. On the other hand, a sudden change in the bar codes might point to an outlier. Real-life applications for this pipeline would be subpopulation detection in cancer genomics, fraud detection in financial transactions ([?]), brain networks ([?]), and robot sensor networks ([?]), to name a few.

The present work is the first quantum algorithm for homology computation using quantum annealing. A quantum algorithm based on the gate model, has been recently proposed in [?]. The key point there is the compression of the simplicial complex $K$ into a quantum state in a $\log_2(|K|)$-dimensional Hilbert space, spanned by the simplices in $K$. Within this space, Betti numbers are computed, in polynomial time, using quantum phase estimate. It is interesting to mention that, in our paper, we also have some form of compression: the covering cliques compress a lots of simplicial data.

We finally point out that there are mathematical constructions for the parallel homology computation other than Mayer-Vietoris complexes, including the spectral sequence algorithm introduced by [?] and further developed by [?].

In Section 2.1, we review the nerve and witness complexes, two efficient ways to map a given data set into a simplicial complex. In Section 2.2, we discuss Mayer-Vietoris complex and explain how the homology computation can be parallelized. Section 2.3 contains our quantum annealing-based solutions to the problem stated above. We have included tests using the D-Wave 2X processor as well as a basic description of how such quantum processors work.
2 Results

2.1 From data sets to simplicial complexes

There are various ways to map a given data set into a simplicial complex. Let us start by listing some efficient examples of such conversion and for which our algorithm can be applied. The first example is the nerve complex. Let $X$ be a data set seen as a discrete topological space, and let $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ be a covering of $X$. The nerve of the cover $\mathcal{U}$, denoted by $N\mathcal{U}$, is the simplicial complex with vertex set $A$, where a family $\{\alpha_0, \ldots, \alpha_k\}$ spans a $k$-simplex if and only if the intersection $U_{\alpha_0} \cap \ldots \cap U_{\alpha_k}$ is not empty. This construction is important because of the nerve theorem which guarantees that complex $N\mathcal{U}$ is homotopy equivalent to the underlying space $X$ provided the overlaps between the covering sets $U_\alpha$ are contractible. It is also important practically since it reduces substantially the size and dimension of the problem.

The next useful complexes are the so-called witness complexes introduced by [?]. Suppose we are given a set $\mathcal{L} \subset X$, called the landmark set, and a parameter $\epsilon > 0$. For every point $x \in X$, we let $m_x$ denote the distance from this point to the set $\mathcal{L}$, i.e., $m_x = \min_{l \in \mathcal{L}} \{d(x, l)\}$. The witness complex attached to $(X, \mathcal{L})$ is then the complex $W(X, \mathcal{L}, \epsilon)$ whose vertex set is $\mathcal{L}$, and where a collection $\{l_0, \ldots, l_k\}$ spans a $k$-simplex if and only if there is a point $x \in X$ (the witness) such that $d(x, l_i) \leq m_x + \epsilon$ for all $i$. It is not hard to see that the witness complex $W(X, \mathcal{L}, \epsilon)$ is actually a nerve complex and thus, by the nerve theorem $W(X, \mathcal{L}, \epsilon)$ is homotopy equivalent to $X$. One can relax the previous definition by considering the clique complex (also known flag complex) of the abstract graph defined by the 1-simplices as above. Then the lazy witness complex, denoted by $W_{\mathcal{L}}(X, \mathcal{L}, \epsilon)$, is defined by taking the 1-simplices as above, but a family $\{l_0, \ldots, l_k\}$ is declared a $k$-simplex if and only if all of the pairs $(l_i, l_j)$ are 1-simplices. Due to its being a relaxation, $W_{\mathcal{L}}(X, \mathcal{L}, \epsilon)$ is not necessarily homotopy equivalent to $X$; however, it works well in many cases with a careful selection of landmarks (an important topic which we will not discuss here; see [?] Section 2.3 for practical guidelines). Figure 1 illustrates two witness complexes of the unit torus and cancer genomic data. Another example of common complexes in TDA, and for our algorithm can also be applied, is Vietoris-Rips complex which is the clique complex of the neighbourhood graph: two points in the data set are connected if their distance is less than $\epsilon$.

2.2 Parallelization of homology computation

We describe now how the homology computation of a simplicial complex $K$, which approximates a given point cloud $X$, is made parallel. The construction described here is valid for any abstract simplicial complex. We recall the key definition of Mayer-Vietoris blow-up complex and then describe the parallelization via a simple example. Let $\mathcal{C} = \{K^i\}_{i \in I}$ be a cover of $K$ by simplicial subcomplexes $K^i \subseteq K$. For $J \subseteq I$, we define $K^J = \bigcap_{j \in J} K^j$. The Mayer-Vietoris blow-up complex ([?]) of the simplicial complex $K$ and cover $\mathcal{C}$ is defined by

$$K^C = \bigcup_{J \subseteq I} \bigcup_{\sigma \in K^J} \sigma \times J. \quad (2.1)$$

A basis for the $k$-chains $C_k(K^C)$ is $\{\sigma \times J \in K^C \mid \dim \sigma + \card J = n\}$. The boundary of a cell $\sigma \times J$ is given by:

$$\partial(\sigma \times J) = \partial \sigma \times J + (-1)^{\dim \sigma} \sigma \times \partial J.$$  

We will not provide a proof here, but it is a fact that the projection $K^C \to K$ is a homotopy equivalence and induces an isomorphism $H_*(K^C) \cong H_*(K)$.

The definition above boils down to the following: The simplicial complex $K^C$ is the set of “original” simplices in addition to the ones we get by blowing up common simplices. These are of the form $\sigma \times J$ in the definition above. In Figure 2, the vertex $d$ common to the two subcomplexes $\{K_1, K_2\}$ is blown up into an edge $v \times 12$ and the edge $bc$ is blown up into the triangle-like $bc \times 01$. In Figure 3, the vertex $a$ common to three subcomplexes $\{K_0, K_1, K_2\}$ is blown up into the triangle $a \times 012$.

Now, the key point in the parallelization is that the boundary map of the simplicial complex $K^C$ (which replaces $K$ by the homotopy equivalence) has a block form suitable for parallel rank computation. As an example, let us consider again the simplicial complex $K$ depicted in Figure 2. First, the space $C_0(K^C)$ is spanned by the vertices

$$a \times 0, b \times 0, c \times 0, b \times 1, c \times 1, d \times 1, d \times 2, e \times 2.$$  

That is, all vertices of $K$ take into account the partition to which they belong. The space of edges $C_1(K^C)$ is spanned by $ab \times 0, bc \times 0, be \times 1, cd \times 1, de \times 2, b \times 01, c \times 01, d \times 12$. That is, first the “original” edges (i.e., those of the form $\sigma \times j$, with $j \in J = \{0, 1, 2\}$ and $\sigma$ being an edge in $K$) are constructed. Then, the new edges that result from blow-ups (i.e., those of the form $v \times ij$, where $v$ is a vertex in $K^J \cap K^I$; if the intersection is empty the value of boundary map is 0), are constructed. The matrix $\partial_0$ with respect to the given ordering is then:
Figure 2: Top: The simplicial complex $K$ is the depicted graph. Middle: $K$ is covered with $K_0$, $K_1$, and $K_2$. Bottom: The blow-up complex of the cover depicted in the middle image. After the blow-up, the edges $b \otimes 01$, $c \otimes 01$, $d \otimes 12$, and the 2-simplex $bc \otimes 01$ appear.

Figure 3: The triangle $a \otimes 012$ appears after blowing up the cover of the middle picture.
We see that one can now row-reduce each coloured block independently. There might be *remainders*, i.e., zero rows except for the intersection part. We collect all such rows in one extra matrix and row-reduce it at the end and aggregate. For the second boundary matrix, we need to determine $C_2(K^c)$. The 2-simplices are of three forms: the original ones (those of the form $σ ⊗ j$ with $σ ∈ C_2(K)$; in this example there are none); those of the form $σ × \{i,j\}$, with $σ ∈ K^1 ∩ K^2$; and those of the form $v ⊗ \{i,j,k\}$, with $v ∈ K^1 ∩ K^2 ∩ K^k$ (there are none in this example but Figure 3 has one). We get $C_2(K^c) = \langle be ⊗ 01 \rangle$ and thus there is no need for parallel computation.

### 2.3 Covering using quantum annealing and speed-up

Now we turn to the problem of covering a given witness complex in such a way that considerably reduces the local rank computation, described earlier. Before doing so, and for completeness, we first present a quantum formulation of the minimum $k$-cut based covering algorithm [21], which is valid for any simplicial complex. The subgraphs in this covering do not have any specific structure. This means that, generally, the local rank computation can not be simplified. Later we present our novel clique based covering methods (specific to witness complexes): vertex clique cover, edge clique cover and edge disjoint-edge clique cover, all of them implemented using quantum annealing. We also discuss the simplified simplicial complex construction and local computation for these three covering methods. We conclude this section by testing all methods using D-Wave 2X quantum annealing processor, as a proof of concept.

For the remainder of this section, $G$ is the 1-skeleton of the simplicial complex $K$, i.e., $G$ is the set of all simplices of dimension $\leq 1$. We also denote by $V(G)$ and $E(G)$ the vertex set and edge set of $G$, respectively. Finally, $n := |V(G)|$ and $m := |E(G)|$.

#### 2.3.1 Covering using minimum $k$-cut

For this part, the simplicial complex $K$ doesn’t need to be a witness complex. We implement quantum mechanically the work of [5]. Therein, METIS graph partitioning library ([2]) has been used to heuristically partition the 1-skeleton $G$ using the minimum $k$-cut. This problem, which is one of Karp’s 21 NP-complete problems ([19]), consists of partitioning the vertex set of $G$ into $k$ non-empty and fixed-sized subsets so that the total weight of edges connecting distinct subsets is minimized. The QUBO formulation of the minimum $k$-cut problem is formulated in ([19]) (see also[2]). For each partition $U_i$ in the graph, for $0 ≤ i ≤ k − 1$, we need $k$ binary decision vectors of size $n$. The row vector $x_i = (x_{i1}, x_{i2}, \ldots , x_{in})$ is part of the solution that indicates that if the vertex $v_j$ of $G$ belongs to the $i$th partition, the $x_{ij}$ is equal to 1; otherwise it is 0. Let $X$ denotes:

$$X = [x_0 \ x_1 \ \cdots \ x_{k-1}]^T$$

The minimum $k$-cut graph partitioning problem is then the QUBO:

$$\min X^T (Q_{main} + αQ_{orth} + βQ_{card}) X$$

$$Q_{main} = -I_k \otimes A$$

$$Q_{orth} = (J_k - 2I_k) \otimes I_n$$

$$Q_{card} = I_k \otimes (J_n - 2s_{av}I_n)$$

The matrix $I_k$ is the $k × k$ identity matrix, $J_k$ is the $k × k$ one matrix with all entries 1, $s_{av} = \frac{1}{k} \left(\frac{n_k}{n} + \frac{n_{k-1}}{n} \right)$ represents the average size (cardinality) of partitions. Also $α$ and $β$ are the orthogonality and cardinality constraint balancing factors.

Once we pass this QUBO to the quantum annealer we obtain subgraphs $\{U_i\}_{0 ≤ i ≤ k − 1}$ which, in addition to an extra subgraph containing all of the edges between them (now minimized), define a covering for $G$. To execute the parallel computation of the previous section, we need to complete this graph covering into a cover of $K$ in terms of subcomplexes for which we have $K = \bigcup_{i=1}^{k} K_i$. For this, we can use the procedure described in [13], which assigns the
subcomplex $K_i$ to the simplex $\sigma \in \mathcal{K}$ if its vertices belong to the subgraph $U_i$. Otherwise, the simplex is put in the extra cover $K_k$. In the covering solutions that follow, this completion step is simplified substantially. In fact, the beforehand knowledge of the whole simplicial complex $\mathcal{K}$, needed here, is not required for our methods. The reason is that the covering cliques confine a large portion of simplices.

2.3.2 Covering using cliques

The simplicial complex $\mathcal{K}$ is assumed to be a witness complex (or a Vietoris-Rips complex). Given the clique-based structure of $\mathcal{K}$, the main idea below is to use cliques to cover the 1-skeleton $G$.

1. Vertex clique cover. A natural first approach is to compute a Vertex Clique Cover (VCC) for the graph $G$. This problem, which is also among the Karp’s 21 NP-complete problems [?], consists of covering the vertex set with cliques such that each vertex is assigned to a unique clique. It can be translated into a vertex colouring problem where we cover $\mathcal{G}$, the complement of $G$, with a given number of colours such that no edge connects two vertices of the same colour. Let $A$ be the adjacency matrix of the graph $G$, the QUBO problem formulation of the vertex clique cover is ([?]):

$$
\begin{align*}
\min \quad & \mathbf{x}^T (Q_{\text{main}} + \alpha Q_{\text{orth}}) \mathbf{x} \\
Q_{\text{main}} &= I_k \otimes (J_n - I_n - A) \\
Q_{\text{orth}} &= (J_k - 2I_k) \otimes I_n
\end{align*}
$$

where $k$, the number of cliques in the problem, is chosen greater than or equal to the clique covering number of $G$, $\theta(G)$ (equal to $\chi(\mathcal{G})$, the chromatic number of $\mathcal{G}$). An upper bound for $\theta(G)$ is $d = \delta(G)$, which is the minimum degree of graph $G$ (Brooks’ theorem [?]).

The resulting covering $\{U_i\}_{i=0}^k$ consists of $k$ cliques, in addition to an extra set containing the edges between those cliques. The boundary matrix of the Mayer-Vietoris complex has the form

$$
\begin{pmatrix}
A_0 & 0 & \cdots & 0 & B_0 \\
0 & A_1 & \cdots & 0 & B_1 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & A_k & B_k
\end{pmatrix}
$$

(2.2)

Here $A_i$ is the boundary matrix of the covering subcomplex $K_i$. For $i \in \{0, \cdots, k-1\}$, each subcomplex $K_i$ is merely the power set of the set $U_i$ (clique complex of a clique). The last subcomplex $K_k$ is the clique complex of the graph $U_k$ whose vertex set $V(U_k)$ is the set of all the vertices of the connecting edges and its set of edges $E(U_k)$ is the restriction of $E(G)$ to $V(U_k)$. Compared to the minimum $k-$cut, this time the rank of the $k-1$ first matrices $A_i$ is known. Also, the passage matrix $P_i$, which makes $A_i$ upper triangular ($A_i P_i$ is upper triangular) is also known. This is because the homology of an $n$-clique is the homology of the $n$-dimensional ball $\mathbb{B}^n$, which is always trivial except for the zeroth homology where it is one-dimensional (this gives for instance, rank $\partial_k = \sum_{\alpha=0, \cdots, \ell} (-1)^{\ell-\alpha} \binom{m}{\alpha}$). Now, the remaining question is to find the remainders (see Section 2.2). Suppose $r_i = \text{rank} A_i$ is the precomputed rank of $A_i$. The remainder is then given by the product $B_i[r_i+1, \text{end}] P_i$, where $B_i[r_i+1, \text{end}]$ is the submatrix of $B_i$ containing the rows $r_i+1$ downward.

2. Edge clique cover. The second method uses Edge Clique Cover (ECC), which is another NP-hard problem from Karp’s list. The problem now is to cover the set of edges of $G$ using a given number of cliques $k$. Let us define

$$
\mathbf{X} = \begin{bmatrix}
\mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_{k-1} & \mathbf{e}_0 & \mathbf{e}_1 & \cdots & \mathbf{e}_{k-1}
\end{bmatrix}^T
$$

where each $\mathbf{e}_i$ is a row vector with $m$ edge variables (the variables $\mathbf{x}_i$ are as in the $k-$cut). The edge clique cover QUBO formulation is then [?]:

$$
\begin{align*}
Q_{\text{main}} &= \begin{bmatrix}
I_k \otimes (J_n - I_n) & -2I_k \otimes B & \end{bmatrix} \\
&= \begin{bmatrix}
-I_k \otimes B^T & (J_k + 3I_k) \otimes I_m
\end{bmatrix}
\end{align*}
$$
where B is the incidence matrix of G. Here, for simplicity, we have presented only the version in which each edge is assigned to at most two cliques. This version requires \( k(n + m) \) variables which is higher than the previous methods. To increase the bound on number of cliques that can cover an edge, we should add extra slack variables as many as \( \left\lfloor \frac{n}{2} \right\rfloor m \). The smallest number of cliques that cover E(G) is called edge clique cover number or intersection number \( \theta_e(G) \). An upper bound for \( \theta_e(G) \) is \( 2e^2(d + 1)^2 \log_e(n) \) with \( d = \Delta(G) \) is the maximum degree of complement of G [?]. If we assume the graph is dense, which is the case for many hard homology problems, \( d \) is small and the intersection number is on the order of \( \log(n) \).

Similar to VCC, since the subgraphs \( \{U_i\}_{i \in \{0, \ldots, k-1\}} \) are cliques, their simplicial subcomplexes \( K_i \) are power sets. The last simplicial subcomplex \( K_k \) is the clique complex of the subgraph \( U_k \) defined as follows. Its vertex set \( V(U_k) \) is the set of vertices inside the pair-wise intersections between the covering cliques \( \{U_i\}_{i=0, \ldots, k-1} \). The edge set \( E(U_k) \) is the restriction of \( E(G) \) to \( V(U_k) \).

3. Edge disjoint-edge clique cover. The third clique covering method is a variation of the edge clique cover, we call it Edge Disjoint-Edge Clique Cover (ED-ECC). Here, the covering subgraphs intersect only at vertices. The algorithm takes as input the graph \( G \) and a stopping criterion. The idea is to iteratively find the maximum clique and each time remove the clique edges from the graph of the previous iteration. Each run gives one maximum clique. At step \( i \), we get a new graph \( G_i \) with adjacency matrix \( A_i \). We stop when the clique is small (stopping criterion 1) or after a certain number of cliques computation (stopping criterion 2). The QUBO formulation for finding the maximum clique (at iteration \( i \)) is

\[
\text{min } \mathbf{x}^T (A^{(i)} - I_n) \mathbf{x},
\]

where \( \mathbf{x} = (x_1, \ldots, x_n)^T \), \( A^{(i)} \) is the updated adjacency matrix at step \( i \), and \( n \) is the dimension of \( A^{(i)} \). The adjacency matrix of the maximum clique is then \( C^{(i)} = \mathbf{x}^{(i)} \mathbf{x}^{(i) T} \circ (J_n - I_n) \), where \( \circ \) is the Hadamard product and \( \mathbf{x}^{(i)} \) is the solution of the QUBO problem at iteration \( i \). There is an obvious gain in terms of the size of the problems that we can handle. Indeed, the number of variables involved here is only \( n \), making this covering method more practical considering current limitation on the size of quantum annealing processor. Note also that using stopping criterion 2, we have more control over the number of local machines, although this is not particularly problematic, as explained earlier.

By construction, the covering subcomplexes \( \{K_i\}_{i=0, \ldots, k} \) (defined in the same way as in ECC) intersect only at vertices; however, a vertex can be blown up into a high-dimensional simplex if it belongs to several covering subgraphs. This translates into a considerable reduction in the size of the matrices \( B_i \), which now involve only simplices of the form \( v \otimes J \), where \( v \) is a vertex in \( G \) and \( J \) is the set of all subcomplexes containing the vertex \( v \). The matrices \( A_i \) are as in the vertex cover and the same reduced computation can be carried out now with much smaller matrices \( B_i \).

2.3.3 Implementation on D-Wave quantum processor

We have tested our algorithm on the D-Wave 2X machine over many instances of the solid torus, as a proof of concept. Here, we report some statistics. Due to the embedding limitations of the D-Wave 2X processor (Appendix A.2), some of the instances were not successfully embedded into the processor, thus we could not calculate their Betti numbers. Columns of tables below represent 1) \( n \) the number of vertices, 2) \( m \) the number of edges, 3) \( D \) the density of the graph , 4) \( k \) is respectively: the number of partitions in minimum \( k \)-cut, the clique number \( \theta(G) \) in VCC and the intersection number \( \theta_e(G) \) in ECC, 5) \( |P| \) the problem size (i.e., the number of binary variables in the QUBO), 6) Embed the embedding and solving status and 7) Betti the Betti number calculated status. The samples are sorted based on number of vertices and problem size, so the reader can see the border of embeddable graphs for each method. Since some of the QUBO’s are more sparse than others, they can be embedded in higher problem sizes.
Minimum $k$--Cut | VCC
---|---
| $n$ | $m$ | $D$ | $k$ | $|P|$ | Embed | Betti | $n$ | $m$ | $D$ | $\theta$ | $|P|$ | Embed | Betti |
| 12 | 24 | 0.36 | 4 | 48 | | &check; | &check; | 12 | 36 | 0.55 | 3 | 36 | | &check; | &check; |
| 12 | 36 | 0.55 | 4 | 48 | | &check; | &check; | 12 | 24 | 0.36 | 4 | 48 | | &check; | &check; |
| 12 | 48 | 0.73 | 4 | 48 | | &check; | &check; | 15 | 45 | 0.43 | 4 | 60 | | &check; | &check; |
| 16 | 40 | 0.33 | 4 | 64 | $\times$ | NA | | 16 | 40 | 0.33 | 4 | 64 | $\times$ | NA |
| 16 | 56 | 0.47 | 4 | 64 | $\times$ | NA | | 16 | 56 | 0.47 | 4 | 64 | $\times$ | NA |

For all instances that the problem was successfully embedded into D-Wave 2X processor, our algorithm successfully calculated the Betti numbers. Note that we only observed the minimum energy solution, among many reads of each annealing process, since our task is to prove the concept. The reader should also note that, these tests only show the correctness of each method’s implemented algorithm. The performance comparison and scaling characteristics of discussed algorithms cannot be evaluated with current size of quantum annealing processor.

### 3 Discussion

In this paper we have discussed how quantum annealing can be used to speed-up the homology computation of large point clouds and presented proof-of-concept tests using the newest D-Wave 2X quantum processor. Additionally, we have presented our work as a complete data mining pipeline where we have supplemented the quantum part with practical tools such as witness complexes and resizable cloud computing platforms.

Our pipeline is dedicated to dense witness graphs; sparse cases should be treated classically. Clearly, the complexity of the pipeline is defined by the dimension of the submatrices $B_{ij}$ of the boundary map matrix (2.2). This dimension is given by the number of the blowup simplices, that is, simplices of the form $\sigma \otimes J$ with $|J| > 1$. Precisely, to compute the $i$th Betti number $\beta_i$, we count all $\ell + 1$ simplices of the form $\sigma \otimes J$, such that $dim(\sigma) + |J| - 1 = \ell + 1$ in addition to $|J| > 1$.

For the vertex and edge cover, the intersection between the subgraphs $(U_j)_{j=0,k}$ is always a clique. The column dimension of $B_{ij}$, needed for $\beta_i$, would be $\sum_{\ell=0}^{\omega} \binom{\kappa}{\ell + i + 2}$ with $\omega := \max \{|\cap_j U_j|\}$ the size of the maximum intersection and $\kappa$ is the maximum number of subgraph $U_j$ with non empty overlap (i.e., the size of the maximum simplex in the nerve complex $N\{U_j\}$). Obviously, $\kappa$ is less than $\theta_e(G)$ which, for dense graph, is in order of $O(\log(n))$ in the edge clique cover case. If $\kappa$ is very big then the subgraphs $U_j$ are small and thus $\ell$ is small ($\beta_i$ is also the Betti number of the initial witness complex $K$ and thus $\ell$ is limited by the size of the maximum subgraph). If $\omega$ is very big then the graph is covered with only a small number of cliques. Almost all other cliques are confined inside these covering cliques and thus in the image of the boundary map. This implies that, after we mod out by the image of the boundary map, we dont have enough simplices to bound high dimensional voids and thus $\ell$ is small. The conclusion of this is that our algorithm, using edge clique cover, is polynomial in time for graphs in which $\kappa$ and $\omega$ are not very big. These are the type of graphs which are intractable classically. For vertex clique, the argument and conclusion are the same with replacing $\theta_e(G)$ with $\theta(G)$. By Brooks’ theorem, $\theta(G)$ is bounded by the degree of the complement graph.

And since the complement is sparse, $\theta(G)$ is small. For edge disjoint edge clique covering, the column dimension of $B_{ij}$ reduces to $\omega \times \nu$ with $\nu := \text{card}\{J \subseteq \{0, \cdots, k\} \mid |J| = \ell + 2, \cap_{j \in J} C_j \neq \emptyset\}$, since the intersection is forced to be at the vertex level only. In this case, our algorithm takes $O(\omega^2 \times \nu^2)$ i.e., polynomial in the size of the graph since $\omega < n$. 

©2016 1QB Information Technologies
A Basic description of the D-Wave processor

Here we introduce the quantum annealing concept that ultimately solves a general Ising (QUBO) problem, then talk about the important topic of embedding a QUBO problem into the specific quantum annealer (D-Wave 2X processor).

A.1 Quantum annealing
Quantum annealing (QA), along with the D-Wave processor, have been the focus of much research. We refer the interested reader to [142, 147, 147, 166, 166, 176]. QA is a paradigm designed to find the ground state of systems of interacting spins represented by a time-evolving Hamiltonian:

\[ S(s) = \mathcal{E}(s)\mathcal{H}_P - \frac{1}{2} \sum_i \Delta(s) \sigma_i^z, \]

\[ \mathcal{H}_P = - \sum_i h_i \sigma_i^x + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z. \]

The parameters \( h_i \) and \( J_{ij} \) encode the particular QUBO problem \( P \) into its Ising formulation. QA is performed by first setting \( \Delta \gg \mathcal{E} \), which results in a ground state into which the spins can be easily initialized. Then \( \Delta \) is slowly reduced and \( \mathcal{E} \) is increased until \( \mathcal{E} \gg \Delta \). At this point the system is dominated by \( \mathcal{H}_P \), which encodes the optimization problem. Thus the ground state represents the solution to the optimization problem.

A.2 Embedding
An embedding is the mapping of the nodes of an input graph to the nodes of the destination graph. The graph representing the problem’s QUBO matrix needs to be embedded into the actual physical qubits on the processor in order for it to solve the QUBO problem. The specific existing connectivity pattern of qubits in the D-Wave chip is called the Chimera graph. Embedding an input graph (a QUBO problem graph) into the hardware graph (the Chimera graph) is in general NP-hard ([140]).

Figure 4 shows a sample embedding into the Chimera graph of the D-Wave 2X chip consisting of an \( 12 \times 12 \) lattice of \( 4 \times 4 \) bipartite blocks. The Chimera graph is structured so that the vertical and horizontal couplers in its lattice are connected only to either side of each bipartite block. Each node in this graph represents one qubit and each edge represents a coupling between two qubits. Adjacent nodes in the Chimera graph can be grouped together to form new effective (i.e., logical) nodes, creating nodes of a higher degree. Such a grouping is performed on the processor by setting the coupler between two qubits to a large negative value, forcing two Ising spins to align such that the two qubits end up with the same values. These effective qubits are expected to behave identically and remain in the same binary state at the time of measurement. The act of grouping adjacent qubits (hence forming new effective qubits) is called chain creation or identification.

An embedding strategy consists of two tasks: mapping and identification. Mapping is the assignment of the nodes of the input graph to the single or effective nodes of the destination graph. Solving such problems optimally is in general NP-hard, but one can devise various approximations and enhancement strategies to overcome these difficulties, for example, using statistical search methods like simulated annealing, structure-based methods, or a combination of both. For a better understanding of current embedding approaches, we refer the reader to [142, 147, 147, 166, 166, 176]. In Figure 4 (bottom), the blue lines indicate the identified couplers, the yellow lines indicates the problem couplers (i.e., the edges of the problem graph), and the grey lines indicate empty couplers.

Acknowledgements
We would like to thank M. Elsheikh for constructive discussions about the parallelization, S. S. Rezaei for his helpful comments, P. Haghnegahdar and P. Ronagh for helpful discussions on the topic, and M. Bucyk for proof reading the manuscript.
Figure 4: Top: A sample 1-skeleton, consists of five cliques of size 14 overlapping on five nodes. Bottom: The actual embedding of the corresponding edge disjoint edge clique cover problem inside the current D-Wave 2X Chimera graph. The QUBO problem has 50 variables, 280 quadratic terms used to map the problem and 586 quadratic terms used to map chains. The colouring of the nodes (and edges, respectively) represents the $h$ parameter values (the $J$ values, respectively) according to the colouring scheme on the interval [-1, 1] represented by a colour interval [blue, red] in the D-Wave 2X processor API.
