Persistent Homology of De Bruijn Graphs with Applications to Genomics

Ian Michael McKay

Spring 2015
Abstract

Persistent homology is a valuable mathematical tool used to gain insight about the shape of a data set. One type of data that biologists and bioinformaticians are interested in is DNA. De Bruijn graphs are an essential tool for assembling genomes, and so the study of the structure of such graphs can help elucidate the underlying structure of the DNA itself. In this thesis, we briefly review the concepts of simplicial homology, de Bruijn graphs, and persistent homology. The persistent homology framework is then applied to a number of bacterial genomes, in order to distinguish various classes of organisms based on taxonomy. Simplicial complexes are formed directly from the de Bruijn graphs of the organisms, as well as from an embedding of the graphs into Euclidean space. The betti numbers for each organism are explored and compared to the barcodes obtained via the aforementioned embedding into Euclidean space. Clusters of organisms based on two distance metrics, Earth Mover’s distance and Bottleneck distance, suggest that the homological structure of the DNA adequately reflects differences in taxonomy.
To David Koslicki, for all the time and sweat he invested in me and this project. I couldn’t have done it without him.

To Chelsea Evans, for pushing me to be the best person that I can be, for encouraging me when times were tough, and for always lending a hand or an ear when I needed it most. You’re my everything, Chelsea.

To Jim and Sally McKay, for raising me, for instilling confidence in me, and for their unwaivering support in the pursuit of my dreams.

To Bill Bogley and Ren Guo, for their guidance, and for sparking my interest in all things topological.
Contents

1 Introduction .......................... 1  
  1.1 Simplicial Homology .................. 1  
  1.2 De Bruijn Graphs ...................... 7  
  1.3 Persistent Homology .................. 13  

2 Application to Bacterial Genomes 21  
  2.1 Data ................................. 22  
  2.2 Preparation of Data .................. 22  
  2.3 Selection of Algorithms ............... 28  
     2.3.1 Bcalm ............................ 28  
     2.3.2 High Dimensional Embedding ....... 29  
     2.3.3 Spring Electrical Embedding ...... 30  
     2.3.4 JavaPlex .......................... 30  
     2.3.5 Bottleneck Distance ............... 31  
     2.3.6 Earth Mover’s Distance .......... 32  
  2.4 Results .............................. 33  
     2.4.1 Betti Numbers .................... 34  
     2.4.2 Validity of the Spring Electrical Embedding 35  
     2.4.3 Persistent Homology in Dimension 2 37  
     2.4.4 Clustering Based on Taxonomy ...... 38  
  2.5 Discussion ........................... 41
List of Figures

1.1 Two DNA sequences and their corresponding de Bruijn Graphs. The image on the left is circular, while the one on the right is more knotted. ......................................................... 9
1.2 The four distinct complete directed graphs on 4 vertices up to isomorphism. ......................................................... 12
1.3 Vietoris-Rips complexes of a point cloud representing an annulus for various values of $\epsilon$. Image due to Robert Ghrist in [14]. . . . 16
1.4 The barcode for $H_*(V)$ for the example in Figure 1.3. Image courtesy of Robert Ghrist in [14]. ................................. 19

2.1 The de Bruijn graph of Acetobacter pasteurianus 386B after running Bealim. The graph is now ready to be turned into a point cloud. ......................................................... 24
2.2 Three barcodes of organisms of varying complexity. Top: Aciduliprofundum MAR08 339. Center: Clavibacter michiganensis nebraskensis NCPPB 2581. Bottom: Chlamydia psittaci RD1. 26
2.3 Example of the Bealim reduction process. The resulting de Bruijn graph has fewer vertices, but retains the information of the original graph. ......................................................... 28
2.4 High Dimensional Embedding applied to Acetobacter pasteurianus 386B. The larger loops are clumped together, causing unwanted persistent homology classes. ................................. 29
2.5 Spring Electrical Embedding applied to Acetobacter pasteurianus 386B. Compared to High Dimensional Embedding, this embedding provides a more natural representation of the graph. . . . . 30
2.6 Persistence diagram of two organisms along with flow indicating how far each point traveled in EMD. Circles: Acidothermus cellulolyticus 11B. Pluses: Aciduliprofundum boonei T469. . . . 33
2.7 Histogram of the betti numbers of the 399 organisms sampled. . 34
2.8 Histogram of the betti numbers of 2768 bacterial genomes, including the 399 organisms used in the experiment. ................. 35
2.9 The de Bruijn graph for Aciduliprofundum boonei T469, as well as a comparison of the spring electrical barcode (right) with the explicit (left). ......................................................... 36
2.10 The de Bruijn graph for Aciduliprofundum MAR08 339, as well as a comparison of the spring electrical barcode (right) with the explicit (left) ......................................................... 37
2.11 The de Bruijn graph and barcode for Bdellovibrio bacteriovorus Tiberius. Highlighted is a 2-cycle which persists. This particular cycle consists of 108 triangular faces.

2.12 Example of a minimal nontrivial 2-dimensional homology class present in the data.

2.13 The Dendrogram of the 399 organism subset using EMD. The highlighted regions show that organisms whose de Bruijn graphs are alike are clustered together.

2.14 The Dendrogram of the 399 organism subset using Bottleneck distance. The highlighted regions show that organisms whose de Bruijn graphs are alike are clustered together.
List of Tables

2.1 Average sensitivity and specificity of each taxonomic rank using EMD. .................................................. 40
2.2 Average sensitivity and specificity of each taxonomic rank using Bottleneck distance. ............................... 41
Chapter 1

Introduction

Like geometry, topology is the study of shape. One of the motivations behind studying topology is the ability to distinguish some shapes from others. Homology is one of the most intuitive and powerful invariants used for this purpose. In Section 1.1, we outline basic homological concepts. Using the language of simplicial homology, we then convert biological information into mathematical objects called de Bruijn graphs in Section 1.2. Finally, we extend simplicial homology to persistent homology in Section 1.3, and examine the biological ramifications of applying this process to bacterial genomes in Chapter 2.

1.1 Simplicial Homology

The fundamentals of simplicial homology are reviewed in this section in order to apply them to persistent homology, and later genomics. The definitions and concepts here are similar to those found in [23] or in [15]. We begin by defining one of the basic building blocks of simplicial homology: an m-simplex.

Definition 1.1.1. An m-simplex $\sigma = [p_0, ..., p_m]$ is the convex hull of $m + 1$ affinely independent points in a real vector space.

Intuitively, a 0-simplex can be thought of as a point in space, a 1-simplex as a line segment joining two points, a 2-simplex as a filled-in triangle, a 3-simplex as a solid tetrahedron, and so on.
Simplices are defined in terms of a set of points, so if $\sigma = [p_0, ..., p_m]$ is an $m$-simplex, then the vertex set of $\sigma$ is denoted $Vert(\sigma) = \{p_0, ..., p_m\}$. An orientation is assigned to a simplex via an ordering of the vertices.

**Definition 1.1.2.** Given a simplex $\sigma$, a **face** of $\sigma$ is itself a simplex $\sigma'$ such that $Vert(\sigma') \subseteq Vert(\sigma)$.

Simplicial structure alone has little value to real world applications, and so it is necessary to create a new “complex” structure out of a collection of simplices taken as a whole.

**Definition 1.1.3.** A **finite simplicial complex** $K$ is a finite collection of simplices in some euclidean space such that:

(i) if $\sigma \in K$, then every face of $\sigma$ also belongs to $K$;

(ii) if $\sigma, \tau \in K$, then $\sigma \cap \tau$ is either empty or a common face of $\sigma$ and $\tau$.

A simplicial complex may contain many simplices in several different dimensions, and so this information is encoded into algebraic objects called **chain groups**, one for each dimension. For a particular dimension $i$, the $i$-chains, which are elements of the chain group, are simply formal sums of oriented simplices in dimension $i$. So an $i$-chain can be written as $\Sigma_{\alpha} n_{\alpha} \sigma_\alpha$, where the coefficients $n_{\alpha} \in \mathbb{Z}$, each $\sigma_\alpha$ is an $i$-simplex in the simplicial complex $K$, and $\alpha$ is an index. The important thing to note is that the collection of these $i$-chains forms an additive group. More can be said, however, as the simplices in that dimension form a basis, and so:

**Definition 1.1.4.** The **$i$-th Chain Group of $K$**, denoted $C_i(K)$, is the free abelian group generated by the oriented simplices of $K$ in dimension $i$.

It is worth noting that the coefficients $n_{\alpha}$ are defined in this section to be integer-valued, which gives way to the canonical definition of chains, cycles, boundaries, and homology groups. In fact, any abelian group can be used in place of $\mathbb{Z}$ (see [12]), and later we will use coefficients in the field $\mathbb{Z}_2$ to further ease computation.
Now for a simplicial complex $K$, each dimension has associated to it a free abelian group $C_i(K)$, but these are all separate entities with no way of communicating with one another.

Take any basis element of the chain group $C_n(K)$. Such an element is an oriented $n$-simplex, so it can be written as $\sigma = [p_0, p_1, ..., p_n]$, and the order of the vertices yields a particular orientation for $\sigma$. Because of the way simplices are constructed, deleting any point $p_i$ from the list $[p_0, p_1, ..., p_n]$ will still be a simplex, and in fact will be an $(n-1)$-simplex. Moreover, this new simplex $\tau = [p_0, ..., \hat{p_i}, ..., p_n]$, where the hat indicates the vertex has been omitted, is actually contained in $K$, by property (i) of Definition 1.1.3. Each chain group is generated by simplices in a specific dimension, and so $\tau$ will not only be an element of $C_{n-1}(K)$, but a basis element.

Listing out each simplex obtained by successively deleting each point from the original $\sigma$ will create a list of every $n-1$ dimensional face of $\sigma$. The sum of all such simplices, with an appropriate sign to account for consistency of orientation, is called the boundary of $\sigma$.

**Definition 1.1.5.** Given an $n$-simplex $\sigma$, the **boundary** of $\sigma$, denoted $\partial(\sigma)$ is:

$$
\partial(\sigma) = \sum_{i=0}^{n} (-1)^i[p_0, ..., \hat{p_i}, ..., p_n]
$$

This definition also holds true to intuition, as the boundary of a solid tetrahedron should be only the outside shell, which is made up of all of its triangular faces. The boundary of a solid triangle should be all of its edges, and so on.

Let $n > 0$. Define a homomorphism $\partial_n : C_n(K) \to C_{n-1}(K)$ via the above boundary formula, setting $\partial_n(\sigma) = 0$ if $n > \text{dim}(K)$, and extending by linearity. This map is called the boundary homomorphism, and it allows each chain group to communicate with the chain groups of one dimension higher and lower. The diagram below illustrates this phenomenon.
Each chain group is no longer a separate entity, and they communicate with each other through appropriate composition of maps. For a given simplicial complex $K$, the collection of all the chain groups of $K$, paired with these boundary homomorphisms form what is known as the Simplicial chain complex of $K$, $C_*(K) = (\{C_i\}, \partial)$. In order for such a pair to be a chain complex, it needs to satisfy one more condition: the composition of two consecutive boundary homomorphisms must be zero. The proof of the $\partial^2 = 0$ lemma (as found in [23]) is a classic:

**Lemma 1.1.1.** $\forall n > 1$, the composition $\partial_{n-1} \circ \partial_n$ is the zero homomorphism.

**Proof.** Let $\sigma \in C_n(K)$. Since $C_n(K)$ is generated by oriented simplices, it suffices to assume $\sigma$ is a simplex. Thus $\sigma = [p_0, ..., p_n]$. Direct computation shows that:

$$\partial_{n-1} \circ \partial_n (\sigma) = \partial_{n-1} \left( \sum_{i=0}^{n} (-1)^i [p_0, ..., \hat{p}_i, ..., p_n] \right)$$

$$= \sum_{i \leq j} (-1)^{j+k-1} [p_0, ..., \hat{p}_i, ..., \hat{p}_j, ..., p_n]$$

$$+ \sum_{i > j} (-1)^{j+k} [p_0, ..., \hat{p}_j, ..., \hat{p}_i, ..., p_n]$$

Now in the second sum, interchanging roles of $i$ and $j$, we see that it is the negative of the first sum. Thus each pair of terms cancel, and $\partial_{n-1} \circ \partial_n = 0$.

Utilizing the group-theoretic properties of the chain complex, the chains in each dimension can be further characterized into smaller groups, called cycles and boundaries.

**Definition 1.1.6.** An **n-cycle** is an n-chain $\sigma \in C_n(K)$ such that $\partial \sigma = 0$. 

$$... \xrightarrow{\partial_{n+1}} C_n(K) \xrightarrow{\partial_n} C_{n-1}(K) \xrightarrow{\partial_{n-1}} C_{n-2}(K) \xrightarrow{\partial_{n-2}} ... \xrightarrow{\partial_0} C_0(K) \rightarrow 0$$
Thus the set of all n-cycles is simply the kernel of $\partial_n$. The kernel of a homomorphism is a subgroup of the domain $[16]$, and so it is clear that $\ker(\partial_n) \leq C_n(K)$.

**Example 1.1.1.** In graph theory, a cycle is a path which starts and ends at the same vertex $v_0$. When following the path, many edges may be included, and some repeated. The path corresponds here to a 1-chain, with the coefficients in the sum corresponding to the number of times an edge is repeated, and the simplices being the edges themselves. The analogy is not perfect, but consider taking the boundary of such a 1-chain. With just a single 1-simplex, $\sigma_1 = [p_0, p_1]$, the boundary will be $\partial_1(\sigma_1) = [p_1] - [p_0]$, using the formula for boundary in Definition 1.1.5. The way to decipher this result is to think of it as ‘head’ minus ‘tail’. Notice that if $\sigma'_1 = [p_1, p_0]$, where the order of the vertices has been reversed, then $\partial_1(\sigma'_1) = [p_0] - [p_1] = -\partial_1(\sigma_1)$, indicating the opposite orientation has been chosen. Thinking back to $\sigma_1$, if another 1-simplex $\sigma_2 = [p_1, p_2]$ is added to the chain, how does this affect the boundary?

Now $\partial_1(\sigma_1 + \sigma_2) = \partial_1(\sigma_1) + \partial_1(\sigma_2) = [p_1] - [p_0] + [p_2] - [p_1] = [p_2] - [p_0]$. The vertex $p_1$ cancels, and what is left is again the ‘head’ of the chain minus the ‘tail’ of the chain. Adding more simplices to the end of the chain will produce the same results, and so for any chain that resembles a path, only the starting and ending vertex contributes to the boundary. In the graph theory setting, we said that cycles start and end at the same vertex. In the chain setting, a cycle is a chain whose boundary is zero. Envision a cycle in a graph as a 1-chain, and take its boundary. The ‘head’ minus ‘tail’ motto reveals that if $\tau$ is the chain in question, with repeated vertex $p_0$, then $\partial_1(\tau) = [p_0] - [p_0] = 0$. Therefore $\tau$ is also a cycle in the chain sense.

Even though it is more difficult to visualize, an analogous result holds in higher dimensions.

\[ \text{Consider a 2-simplex, } \sigma = [p_0, p_1, p_2]. \text{ Remember that taking the boundary} \]
of \( \sigma \) will leave a 1-chain, \([p_1, p_2] - [p_0, p_2] + [p_0, p_1]\) which corresponds to the three 1-simplices on the boundary of the 2-simplex. Lemma 1.1.1 ensures that the boundary of this new 1-chain will be zero. This means that the chain is in fact a cycle, but it is somehow a different type of cycle than if the simplicial complex consisted only of the three 1-simplices and no 2-simplex at all. In the latter case, there is a hole in the middle of the simplicial complex, and the cycles are able to capture that information. In the former case however, there is no hole being measured, as the original simplicial complex was a solid simplex. One can imagine a 2-chain with many simplices surrounded by other simplices, thus creating numerous cycles that do not measure any voids or holes.

We wish to distinguish these two classes of cycles, and really only take into account the ones which capture information about the holes in each dimension.

Starting with the 2-simplex \( \sigma \), a chain was created when taking the boundary. Every time this happens, Lemma 1.1.1 forces this chain to be a cycle.

**Definition 1.1.7.** Chains which are the boundary of a simplex in one dimension higher are called **boundaries**.

Since each boundary came about by using the boundary homomorphism, then the set of all such boundaries in \( C_n(K) \) is \( \text{im}(\partial_{n+1}) \).

**Definition 1.1.8.** The set of all n-cycles in a simplicial complex \( K \) is denoted \( Z_n(K) = \ker(\partial_n) \), and the set of all n-boundaries in \( K \) is denoted \( B_n(K) = \text{im}(\partial_{n+1}) \).

Once more via Lemma 1.1.1, every boundary is a cycle, and so for all dimensions \( i \), \( B_i(K) \subseteq Z_i(K) \subseteq C_i(K) \). More can be said, however, as the boundaries actually form a subgroup of \( Z_i(K) \), which are in turn a subgroup of \( C_i(K) \).

**Definition 1.1.9.** For a simplicial complex \( K \), the **n-th simplicial homology group** of \( K \) is the quotient group: \( H_n(K) = Z_n(K)/B_n(K) \).

The definition above is quite abstract, but discussion of homology at any level will tell you that “Homology is the search for cycles that are not boundaries.” [3]
Definition 1.1.10. The n-th Betti number of a simplicial complex $K$ is $b_n(K) = \text{rank}(H_n(K))$.

The betti number in each dimension serves as an estimate of the number of holes that $K$ contains in each dimension. For example, $b_0(K)$ measures the number of connected components of the simplicial complex, $b_1(K)$ counts the number of 1 dimensional loops, $b_2(K)$ the number of voids, and so on. The barcodes defined in Section 1.3 will serve as an analogue to the betti numbers.

1.2 De Bruijn Graphs

The homology groups defined in the previous section allow one to gain information about the shape of a simplicial complex. The focus of our study will be on forming a simplicial complex from de Bruijn graphs.

De Bruijn graphs are a particular kind of graph originally introduced in the 1940s by the Dutch mathematician Nicolaas Govert de Bruijn. De Bruijn initially constructed these graphs to prove a conjecture about the existence of what are now called de Bruijn sequences. The use of de Bruijn graphs has become ubiquitous in the field of bioinformatics, aiding in the tasks of de novo genome assembly, and SNP detection [17], [18].

Despite their frequent use in this field, many of the properties unique to de Bruijn graphs have remained unexplored. The definitions and basic concepts of de Bruijn graphs (as can be found in [8] and [4]) are introduced in this section.

Definition 1.2.1. An alphabet is a set $\mathcal{A}$ consisting of a finite number of symbols.

Definition 1.2.2. Words of length $k$, called k-mers, are of the form $x = x_1x_2...x_k$, where $x_i \in \mathcal{A} \forall 1 \leq i \leq k$. The set of all k-mers is denoted $\mathcal{A}^k$.

Example 1.2.1. Consider $\mathcal{A} = \{0,1\}$. Then all 3-mers would be the set $\{000,001,010,011,100,101,110,111\}$. In a de Bruijn graph, the vertices will
consist of all k-mers for a fixed k, and a directed edge is formed when there is sufficient overlap between two k-mers.

\[ \text{Definition 1.2.3.} \text{ The k-dimensional de Bruijn graph, } B_k(A^k), \text{ is the directed graph with vertex set } V = A^k, \text{ and edge set } E = \{ (v, w) \in V \times V | v_2...v_k = w_1...w_{k-1} \}. \]

\[ \text{Example 1.2.1 (continued). With } A = \{0,1\} \text{ and } k = 3, \text{ then there would be an edge from 001 to 010, but no edge from 001 to 101.} \]

This notion can also be extended to undirected graphs in the following manner:

\[ \text{Definition 1.2.4. The k-dimensional symmetric de Bruijn graph, } B^*_k(A^k), \text{ is the undirected graph with vertex set } V = A^k, \text{ and edge set } E = \{ (v, w) \in V \times V | v_2...v_k = w_1...w_{k-1} \text{ or } v_1...v_{k-1} = w_2...w_k \}. \]

For our purposes, we set \( A = \{A,C,G,T\} \), which corresponds to the nucleotides adenine, cytosine, guanine, and thymine respectively. Given an alphabet, there is only one de Bruijn graph for any fixed dimension k, and so it will be useful to focus on particular subgraphs of the full de Bruijn graph in order to model biological data. These de Bruijn graphs will be the link between the biological structure (DNA), and the mathematical structure (homology). Biologists are able to extract short sequences of nucleotides, called reads, contained in a strand of DNA from a particular organism, and then attempt to reconstruct the full sequence by piecing together the reads, much like a jigsaw puzzle. The difficulty is that even the simplest organisms such as bacteria have millions, or even tens of millions of base pairs in their genome. De Bruijn graphs have proven quite useful in this context [8].

\[ \text{Example 1.2.2. Suppose the sequence } AAATAGACTCGGAAA \text{ represents the genome of a particular organism. Letting } k = 4, \text{ we can fragment this} \]
particular sequence into 4-mers in order to construct a de Bruijn graph. The vertex set consists of all 4-mers present in the full sequence, and so we see that $$V = \{AAAT, AATA, ACTC, AGAC, ATAG, CGGA, CTCG, GAAA, GACT, GGAA, TAGA, TCGG\}$$. Next, the edges are formed when the suffix of one 4-mer is the prefix of the next (See Figure 1.1). With the graph $$G = (V, E)$$ in hand, the full sequence can be reconstructed by finding a Hamiltonian cycle in $$G$$. One begins at vertex $$AAAT$$, and traverses the edges to each vertex, spelling out the entire sequence: $$AAAT \rightarrow AATA \rightarrow ATAG \rightarrow TAGA \rightarrow ... \rightarrow GAAA \rightarrow AAAT$$. Each edge is an amalgamation of the two vertices, and so $$AAAT \rightarrow AATA$$ spells out the 5-mer $$AAATA$$. Since computing Hamiltonian cycles is NP-complete [13], finding such cycles is difficult algorithmically, particularly when there are millions or billions of vertices involved. Work has been done to convert this type of problem into finding Eulerian cycles [8].

![Figure 1.1: Two DNA sequences and their corresponding de Bruijn Graphs. The image on the left is circular, while the one on the right is more knotted.](image)

One issue with applying de Bruijn graphs to genome assembly is that for small values of $$k$$, there are only $$|A|^k$$ possible vertices, and yet there could be billions of reads, leading to numerous $$k$$-mers in the sequence being associated
to the same vertex. This is taken into account by weighting the vertices based on how many of each \( k \)-mer are present in the sequence. Vertices in a de Bruijn graph can have degree at most \( 2|\mathcal{A}| \), and it is quite common for vertices in the graph to have degree higher than 2 (see Figure 1.1). Since reconstruction of the genetic sequence relies on finding cycles in the graph, vertices with higher degree introduce a choice, and thus possible errors, in the reconstruction. There are also physical limitations to consider when choosing a higher value of \( k \). The goal is to minimize such errors while using the methods of DNA sequencing currently available.

Since de Bruijn graphs are a valuable tool for assembling genomes, the study of the structure of such graphs can help elucidate the underlying structure of the DNA itself. The main goal will be to turn a de Bruijn graph into a simplicial complex. This can be done in a number of different ways. The first, and simplest method would be to take the graph itself as a collection of 0- and 1-simplices, i.e. each \( v \in V \) is taken directly as a 0-simplex, and each \( e \in E \) is taken to be a 1-simplex. It is clear that \( K \) satisfies the definition of a simplicial complex, since each edge contains all of its faces (the vertices), and simplices are either disjoint or meet at a common face, and so homology groups can be computed. While this is a very concrete way of turning \( G \) into a simplicial complex, it may not be the best method. The fact that there are only 0- and 1-simplices means that all chain groups above dimension 1 are trivial, and homology groups can only be examined in dimensions 0 and 1. This is acceptable for a variety of applications, as there is a clear biological interpretation of the connected components of \( G \), and 1-dimensional cycles in \( G \). Another approach would be to form a clique complex from the graph \( G \).

**Definition 1.2.5.** A **clique**, \( C \), in a graph \( G \), is a finite collection of vertices such that the induced subgraph of \( G \) is complete.

Complete graphs are such that every vertex is adjacent to every other vertex. Thus if \( G \) is a complete graph with \( n \) vertices \( v_1, ..., v_n \), any subcollection of \( m \)
vertices $G' \subseteq Vert(G)$ forms a clique of size $m \leq n$. This means that if we take the collection of all cliques of a graph $G$, then each clique can be taken as a simplex, such that if a clique $C$ has the property that $|C| = k$, then $C$ is a simplex with $dim(C) = k - 1$. The resulting object is a simplicial complex, and portions of the graph with high connectivity correspond to simplices of high dimension. Clique complexes are examples of what are called flag complexes.

**Definition 1.2.6.** A **flag complex** is a simplicial complex $K$ that is entirely determined by its 1-skeleton, in that an $m$-simplex is formed whenever an $m + 1$ collection of vertices in $Vert(K)$ contains all of its 1-dimensional faces.

We will see later that the use of flag complexes can be computationally advantageous when dealing with graphs that have many vertices and edges. This new simplicial complex alleviates the problem before, where homology groups could only be calculated in dimensions 0 and 1. However, in the context of de Bruijn graphs, another problem arises when computing the clique complex.

**Lemma 1.2.1.** Let $G$ be a de Bruijn graph. There are no cliques $C$ in $G$ such that $|C| \geq 4$.

**Proof.** First notice that if $G$ contains a clique with more than 4 vertices, $C = \{p_1, ..., p_i\}, k > 4$, then any subset of $C$ will also be a clique, since the induced subgraph of $G$ is complete. Thus if $i > 4$, then $G$ will also contain a clique with exactly 4 vertices. We now show that $G$ cannot contain a clique with 4 vertices, and the result follows. Suppose by way of contradiction that $G$ does contain a clique of 4 vertices, namely $C = \{a, b, c, d\}$, where $a, b, c, d \in G$ are distinct, and $a = a_1a_2...a_k, b = b_1b_2...b_k$, and so on. Consider the induced subgraph $C' \subset G$. Since $C'$ is complete, it must contain $(4 - 1)! = 6$ edges. In order for $a$ and $b$ to be adjacent in $C'$, then either $a_2...a_k = b_1...b_{k-1}$ or $b_2...b_k = a_1...a_{k-1}$. Choosing one of these two possibilities, and later choosing the other reduces the claim to only directed de Bruijn graphs; if a directed edge exists in a de Bruijn graph, so too must its analogue in the undirected case. A similar situation occurs for all edges. There are 64 complete directed graphs on
4 vertices, and so we must show that in each case, we arrive at a contradiction. Luckily, up to isomorphism, there are only 4 cases to check (See Figure 1.2).

Figure 1.2: The four distinct complete directed graphs on 4 vertices up to isomorphism.

**Case 1:** \( E = \{(a, b), (b, c), (c, d), (d, a), (c, a), (d, b)\} \)

Since \((a, b) \in E\), then \(a_2\ldots_a_k = b_1\ldots b_{k-1}\). Similarly, each edge in \( E \) indicates:

\[
\begin{align*}
  b_2\ldots b_k &= c_1\ldots c_{k-1} \\
  c_2\ldots c_k &= d_1\ldots d_{k-1} \\
  d_2\ldots d_k &= a_1\ldots a_{k-1} \\
  c_2\ldots c_k &= a_1\ldots a_{k-1} \\
  d_2\ldots d_k &= b_1\ldots b_{k-1}
\end{align*}
\]

Using the above, we have \(b_1\ldots b_{k-1} = a_2\ldots a_k = d_2\ldots d_k = a_1\ldots a_{k-1} = c_2\ldots c_k = d_1\ldots d_{k-1}\). Thus \(a_1\ldots a_{k-1} = d_1\ldots d_{k-1}\) and \(a_2\ldots a_k = d_2\ldots d_k\), so then \(a = d\). This is a contradiction because \(a, b, c\) and \(d\) were chosen to be distinct.

**Case 2:** \( E = \{(b, a), (b, c), (c, d), (d, a), (c, a), (d, b)\} \)

Using a similar argument to Case 1, we see that \(b_2\ldots b_k = a_1\ldots a_{k-1} = b_1\ldots b_{k-1} = c_2\ldots c_k = d_1\ldots d_{k-1}\). Thus \(a_1\ldots a_{k-1} = d_1\ldots d_{k-1}\) and \(a_2\ldots a_k = d_2\ldots d_k\), so then \(a = c\). This again, this contradicts the fact that the vertices are distinct.

**Case 3:** \( E = \{(b, a), (b, c), (c, d), (d, a), (a, c), (b, d)\} \)

In this case, we have: \(b_2\ldots b_k = a_1\ldots a_{k-1} = a_2\ldots a_k = c_2\ldots c_k = d_1\ldots d_{k-1} = c_1\ldots c_{k-1} = d_2\ldots d_k\), and so \(a = c = d\).

**Case 4:** \( E = \{(b, a), (c, b), (c, d), (d, a), (c, a), (b, d)\} \)

Finally, we have: \(b_2\ldots b_k = a_1\ldots a_{k-1} = c_2\ldots c_k = d_1\ldots d_{k-1} = b_1\ldots b_{k-1} =\)
$d_2 \ldots d_k$, and so $b = d$.

Since a clique with 4 vertices cannot occur in any of the cases, then $G$ cannot contain such a clique, nor any clique with more 4 vertices.

The lemma above indicates that in a de Bruijn graph, there are only cliques of sizes 1, 2, and 3. Therefore forming the clique complex from a de Bruijn graph, will yield simplices (and thus potentially nontrivial homology groups) in dimensions 0, 1 and 2 only. This is a step up from using the graph as a simplicial complex explicitly, but it leaves much to be desired for those interested in higher dimensional phenomena. In order to introduce other methods to turn biological data into a simplicial complex using de Bruijn graphs, we turn our attention to persistent homology.

1.3 Persistent Homology

As technological resources have continued to improve, the field of computational topology has burst wide open. New techniques, algorithms, and tools have allowed us to perform computations that 60 years ago would have been unfathomable. One such tool is persistent homology, which applies to a broad class of problems in data analysis. The motivating factor here is that a set of data, no matter how large or small, has shape. Often times data might be inherently high dimensional.

Example 1.3.1. An auto insurance company has 1,000 customers, and each customer has their age, the number of accidents they have been in during the last 2 years, the number of years they have been with that company, their gender, the color of their car, and so on. If we manage to embed the set of customers as points in Euclidean space, then using just the attributes listed, each customer is a 5 dimensional point in space. The more points and the higher the dimension, the more impossible it is to visualize the overall shape of the data.
In order to gain insight about the shape features, the data is converted to a point cloud.

**Definition 1.3.1.** A **point cloud** is a collection of points in Euclidean space, $\mathbb{E}^n$.

Some problems are easily embedded into Euclidean space, and others (such as gender, or color of car above) are more difficult to embed while still retaining the original shape. The focus here will be on obtaining a point cloud from a de Bruijn graph, so that later the shape of bacterial genomes can be determined.

There is a multitude of ways in which a point cloud can be transformed into a simplicial complex, as outlined below. These complexes are similar to the complexes formed in Section 1.1, with the difference being that they have a 'persistence parameter'.

**Definition 1.3.2.** Given a point cloud $X$, and $\epsilon > 0$, the Čech complex of $X$, with filtration parameter $\epsilon$, denoted $C_\epsilon(X)$, consists of the following:

(i) Each point $x \in X$ is a 0-simplex in $C_\epsilon(X)$;

(ii) For each $x_0, x_1 \in X$, if $B_\epsilon(x_0) \cap B_\epsilon(x_1) \neq \emptyset$, then $[x_0, x_1] \subset C_\epsilon(X)$ is a 1-simplex, where $B_\epsilon(x_i)$ is the Euclidean ball centered at $x_i$ with radius $\epsilon$;

(iii) For each $x_0, ..., x_k \in X$, if $B_\epsilon(x_0) \cap ... \cap B_\epsilon(x_k) \neq \emptyset$, then $[x_0, ..., x_k]$ is a $k$-simplex in $C_\epsilon(X)$.

Another name for the Čech complex is the “Nerve” of $X$. The Čech complex has been shown (by the Nerve Theorem, see [2] or [5]) to be topologically faithful to the structure of the point cloud fattened by Euclidean balls. However, with more than a few vertices in $X$, computation can rapidly become expensive. This is because in each dimension, pairwise intersections must be checked for all relevant subsets of points in $X$. As stated earlier, even the smaller bacterial genomes consist of thousands of nucleotides, and so creating a Čech complex of such data would be infeasible.
In order to combat this computational issue, another option would be to use a Vietoris-Rips complex.

**Definition 1.3.3.** Let \(d(x, y)\) denote the Euclidean distance between points \(x, y \in \mathbb{R}^n\). A **Vietoris-Rips complex** of a point cloud \(X\), with filtration parameter \(\epsilon\), denoted \(V_\epsilon(X)\), is the simplicial complex formed as follows:

(i) Each point \(x \in X\) is a 0-simplex in \(V_\epsilon(X)\);

(ii) For any subset \(x_0, ..., x_k\) of \(k+1\) points in \(X\), if \(d(x_i, x_j) \leq \epsilon\) for all \(i, j \leq k\), then \(\sigma = [x_0, ..., x_k]\) is a \(k\)-simplex in \(V_\epsilon(X)\).

The advantages of using a Vietoris-Rips complex are twofold. The definition allows complexes to be formed in any metric space, rather than only restricting to Euclidean space. This means that the Vietoris-Rips complex does not depend on the embedding of \(X\) into \(\mathbb{R}^n\), but merely on the distances between points (see [9]). The second advantage is that the Vietoris-Rips complex is a flag complex, drastically decreasing the number of computations required over the Čech complex. The consequence of this is that for a particular value of \(\epsilon\), the Vietoris-Rips complex can be quite different than the Čech complex corresponding to the same \(\epsilon\), and so it may not faithfully represent the data. The solution is to choose two values \(\epsilon, \epsilon'\) with \(\epsilon' < \epsilon\). The following theorem (as proven in [9]) indicates that geometric properties of Čech complex may be inferred from the two corresponding Vietoris-Rips complexes (See Figure 1.3).

**Theorem 1.3.1.** Let \(X\) be a point cloud in \(\mathbb{R}^d\), and \(C_\epsilon(X)\) be the Čech complex of the cover of \(X\) by balls of radius \(\epsilon/2\). Then there is a chain of inclusions \(R_{\epsilon'}(X) \subset C_\epsilon(X) \subset R_\epsilon(X)\) whenever \(\frac{\epsilon}{\epsilon'} \geq \sqrt{\frac{2d}{d+1}}\).

A third and fourth option would be to use a Witness complex, or a Lazy Witness complex. Both complexes involve extracting a dense core subset of the original point cloud. The subset consists of landmark points, which are used as 0-simplices, while the unused points can ‘witness’ a 1-simplex if they are within \(\epsilon\) of two landmark points. Higher dimensional simplices are inherited from the
Figure 1.3: Vietoris-Rips complexes of a point cloud representing an annulus for various values of \( \epsilon \). Image due to Robert Ghrist in [14].

1-skeleton, making both the Witness complex and Lazy Witness complex flag complexes. The value that these bring is that they are even less computationally expensive, since the number of simplices in a lazy witness complex is significantly fewer than the number for a Vietoris-Rips complex (see [1]). They are not without cost, however. With more parameters (choosing the landmark subset), it is much more difficult to pinpoint the shape features present in the Čech complex. So it is often the case that many witness complexes need to be constructed, with different numbers of landmark points, and different numbers of divisions in the filtration (see Definition 1.3.4), in order to find a decent approximation. Thus, by itself, the witness complex is too coarse of an approximation for our application.

The natural question at this stage is: how can we determine the optimal value for \( \epsilon \), so that there is as little noise in the data as possible, and so that the significant features are present? It is clear that for small enough \( \epsilon \), the point cloud \( X \) remains discrete, and yet too large of \( \epsilon \) leads to a single contractible mass where all of the shape features have been swallowed up [14]. This question
is answered by the notion of ‘persistence’.

The first step is creating a filtration (as found in [11]), which informally is a way of choosing many values of $\epsilon$ at once. Suppose there is a simplicial complex $K$, and some function $f : K \to \mathbb{R}$ which is monotonically non-decreasing as the faces in $K$ increase. In other words, if $\sigma, \tau \in K$ with $\sigma$ a face of $\tau$, then $f(\sigma) \leq f(\tau)$.

This requirement ensures that the inverse image $K(t) = f^{-1}(-\infty, t]$ is not only a simplicial complex, but in fact a subcomplex of $K$ for each $t$. In practice, only finite simplicial complexes are used, and so there is some $m \geq 0$ such that $m$ is the number of simplices in $K$. Each of these $m$ simplices has a corresponding function value, $f(\sigma_i) = t_i$, and thus a sequence of real numbers is obtained: $t_1 < t_2 < \ldots < t_m$. For notational simplicity, let $K_i = K(t_i)$, and set $t_0 = -\infty$. Choosing $n \leq m$, we obtain a chain of $n + 1$ simplicial complexes ordered by inclusion:

$$\emptyset = K_0 \subseteq K_1 \subseteq \ldots \subseteq K_n = K$$

**Definition 1.3.4.** The selection of subcomplexes of a simplicial complex $K$ using the construction above is a filtration of $K$. Higher values of $n$ correspond to finer filtrations, and lower values correspond to coarser filtrations.

Given a particular filtration of $K$, then for any $0 \leq i \leq j \leq n$, there is a natural inclusion $f^{i,j} : K_i \hookrightarrow K_j$. Note that this $f$ should not be confused with the $f$ defined for a filtration of $K$ above. The fact that $f$ is an inclusion means that we can think of a simplex $\sigma \in K_i$ as it would live in $K_j$, and so $f^{i,j}(\sigma) = \sigma \in K_j$. Therefore $f$ takes simplices to simplices, and is a simplicial map. For each dimension $p \leq n$, consider a $p$-chain $\sum_\alpha n_\alpha \sigma_\alpha$, where each $\sigma_\alpha \in K_i$ is a $p$-simplex. Such a $p$-chain is a linear combination of simplices, and thus mapping via $f$ induces a homomorphism $f^{i,j}_\# : C(K_i) \to C(K_j)$, which takes $p$-chains in $C(K_i)$ to $p$-chains in $C(K_j)$. Both complexes have equipped a boundary homomorphism, leading to the following commutative diagram:
Now we see that for any chain \( c \in C_p(K_i) \), that \( \partial_j \circ f_#(c) = f_# \circ \partial_i(c) \), and so \( f_# \) is a chain map.

Every chain map induces a homomorphism in homology, and so \( f_# \) induces \( f_{i,j}^p : H_p(K_i) \to H_p(K_j) \) for each \( p \) defined by \( f_{i,j}^p(\text{cls}(z)) = \text{cls}(f_{i,j}^p(z)) \), where \( \text{cls}(z) \) is the equivalence class of \( z \) in homology. Therefore we obtain from our filtration a sequence of maps \( 0 = H_p(K_0) \to H_p(K_1) \to ... \to H_p(K_n) = H_p(K) \).

Moving to the right in this sequence, some new homology classes are born, and some others may become trivial [11].

**Definition 1.3.5.** The \( p \)-th persistent homology groups are \( H_{i,j}^p = \text{im}(f_{i,j}^p) \).

Let \( z \in H_{i,j}^p \) for some particular \( i, j, \) and \( p \). Since \( z \in H_{i,j}^p \), it represents a homology class existing either before or beginning at \( K_i \) which still exists until \( K_j \) or after. Notice immediately that if \( i = j \), then \( H_{i,j}^p = H_p(K_i) \). We may alternatively define \( H_{i,j}^p = Z_p(K_i)/(Z_p(K_i) \cap B_p(K_j)) \). This definition makes sense, because if an element of \( Z_p(K_i) \) (a cycle), is in fact a boundary, then \( f \) will map this element to a boundary in \( K_j \). If such an element is not a boundary in \( K_i \), yet would be trivial in \( H_p(K_j) \), then that element is \( 0 \in H_{i,j}^p \), as it is a boundary in \( K_j \). Lastly, if the cycle in \( K_i \) persists until or after \( K_j \), then it will remain nontrivial in \( H_{i,j}^p \). If a homology class \( z \) appears between \( i \) and \( j \), then it must be the case that there exists \( l \), with \( i \leq l \leq j \) such that \( z \notin H_{l,i}^p \), but \( z \in H_{l,j}^p \).

**Definition 1.3.6.** For any \( i \leq j \), a class \( z \in H_{i,j}^p \) is born at \( i \) if \( z \in H_{i,i}^p \), but \( z \notin H_{i-1,i}^p \), and \( z \) dies entering \( j \) if \( f_{i,j}^p(z) \notin H_{i,j}^{p-1,j} \), yet \( f_{i,j}^p(z) \in H_{i,j}^{p-1,j-1} \).

In other words, the class of \( z \) becomes trivial passing from \( j-1 \) into \( j \). If \( z \) does not die during the course of the filtration, then \( z \in H_p(K) \), and we say that \( z \) persists to infinity.
Now for a particular homology class $z$ in dimension $p$ occurring somewhere in the filtration of $K$, $\exists i, j \leq n$ such that $z$ is born at $i$ and dies entering $j$ (or persists to infinity). Each $i, j$ corresponds to a real number from the selection of the filtration of $K$, which we will re-label as $t_i$ and $t_j$ respectively. The monotonicity requirement allows us to conclude that $t_i \leq t_j$, and so $[t_i, t_j] \subset \mathbb{R}$ is a real interval, called the persistence interval of $z$.

The collection of all persistence intervals for a particular dimension can be graphically represented by what is known as a barcode (See Figure 1.4). Barcodes have a filtration time axis along the bottom, and horizontal bars are placed alongside the axis such that the left endpoint of a bar represents the beginning of a persistence interval, and the right endpoint represents the end of the interval.

Figure 1.4: The barcode for $H_*(V)$ for the example in Figure 1.3. Image courtesy of Robert Ghrist in [14].

A feature of the barcode representation is that on any fixed filtration time $t_i$, a vertical line can be placed perpendicular to the time axis, allowing one to count how many bars intersect that vertical line. This number will be the number of persistent homology classes which are ‘alive’ at time $t$. More formally, this number is the rank of $H_p(K_i)$, which is the $p$-th persistence Betti number.
at $t_i$. This serves as an approximation to the number of ‘holes’ in the $p$-th dimension that occur in the simplicial complex at that time.

We have already seen that the Čech complex, the Vietoris-Rips complex, the witness complex, and the lazy witness complex all fit this persistence framework, since as $\epsilon$ increases in each case, the resulting simplicial complexes are ordered by inclusion. Also discussed earlier, it is possible to glean information about a Čech complex by ‘squeezing’ it in between two Vietoris-Rips complexes. This reinforces the fact that the Vietoris-Rips complex is a much more valuable tool as a persistence complex than as a simplicial complex taken with only a single value of $\epsilon$. 

20
Chapter 2

Application to Bacterial Genomes

The persistent homology framework will now be applied to bacterial genomes. Given the numerous ways to form a simplicial complex as outlined in Section 1.3, we first explore which methods are most appropriate for the data at hand. The size and complexity of the different types of simplicial complexes give rise to a trade-off between computational capability and the amount of useful information that can be extracted. This trade-off is explored by first treating the de Bruijn graph of an organism as an explicit simplicial complex, and later treating the data as a point cloud. In the latter case, a few strategies for alleviating the computational burden are presented. Finally, a subset of bacteria organisms is selected. It will be shown that persistent homology of the associated de Bruijn graphs of the organisms is an acceptable tool for distinguishing taxonomic ranks of the organisms. Two distance metrics will be applied to the barcodes, offering further evidence to support this claim.
2.1 Data

All available bacterial whole genomes were downloaded from the National Center for Biotechnology Information (NCBI) website on January, 2015. The NCBI houses links to thousands of nucleotide sequences from a variety of different organisms, including the human genome, and the bacterial genomes used in this experiment. After obtaining the genomes, the plasmids were removed, leaving 2,768 organisms stemming from a variety of taxa. De Bruijn graphs, computations on the explicit simplicial complexes, as well as both embedding processes (see Section 2.3) were performed on all of the genomes. A subset of 399 organisms of small to moderate size were selected for persistent homology computations. A full list of the 399 organisms in this subset, as well as the custom code used can be found at https://github.com/mckayi/Homology.

2.2 Preparation of Data

First, de Bruijn graphs were formed for each of the genomes. The raw genome is taken from the DNA sequencer and encoded as a file with a ‘.fasta’ extension. Fasta files contain long sequences of nucleotides that were read directly from the sample. Jellyfish [19] was used to take each fasta file and produce a list of 30-mers, as well as how many times each 30-mer appeared in the file.

Directly forming a de Bruijn graph led to a large percentage of organisms having millions of vertices. Even using 512GB of RAM, JavaPlex (for a description of JavaPlex, see Section 2.3.4) quickly ran out of memory when forming the Vietoris-Rips complex. In order to reduce the complexity of the graph without losing any information about its shape, each file was run through a program called Bcalm. K-mers are input into Bcalm [6], and the output results in a generalization of a de Bruijn graph. This process eliminates numerous well-behaved vertices, while keeping the topology of the graph unchanged. A description of Bcalm can be found in Section 2.3.

The genomes could then successfully be converted into de Bruijn graphs via
a custom python script, which took each listing in a Bcalm file as a vertex, and formed edges whenever any two vertices overlapped in 29 places. Notice that since the de Bruijn graph was reduced through Bcalm, each vertex could correspond to a k-mer with potentially hundreds of nucleotides. Thus in order to maintain the structure of the graph, the edges needed to be formed when two k-mers overlapped in exactly 29 places, rather than $k - 1$ places. Three files were output by the python script: the first file was the de Bruijn graph of the sample, so that it could be read into JavaPlex. The next file was the de Bruijn graph in Mathematica format, so that the graph could be turned into a point cloud. The final file was a translator file, consisting of the vertices, as well as their id numbers.

Typically, the de Bruijn graph of a sample would consist of one very large connected component, and many small fragments with fewer than 10 vertices each. One reason for this is that some pieces of the genetic sequence of the sample would be unreadable, or scrambled in some way. The raw DNA file would put an $N$ in place of the unknown part of the sequence. When Jellyfish encounters an $N$ it then throws out the k-mer in question, leading to some broken pieces surrounding it. These fragments were not considered as important as the large connected component, and hindered the process of faithfully embedding the graph into Euclidean space. Since the ‘interesting’ parts of the graph were not in the fragments, only the largest graph component was used for each sample (See Figure 2.1). Next, a simplicial complex was formed from the de Bruijn graphs in two ways.

Each de Bruijn graph was first taken as an explicit simplicial complex (just 0- and 1-simplices from the vertices and edges of the graph itself), and JavaPlex was used to calculate betti numbers in dimensions 0 and 1. This acted as a baseline to ensure the next step in the process would be faithful to the biological interpretation of the data.

Next, the de Bruijn graph needed to be turned into a point cloud, so that a Vietoris-Rips complex could be formed, and persistent homology groups could
be calculated. Graphs were first embedded into 3 dimensional Euclidean space by Mathematica’s High Dimensional Graph Embedding algorithm [10]. This algorithm tended to clump points together in an unnatural way due to the fact that the graphs were inherently high dimensional, yet the embedding was low dimensional ($\mathbb{R}^3$). A description of this phenomenon appears in Section 2.3.2.

In order to alleviate this issue, the Mathematica spring electrical embedding was computed for each graph instead [10]. This algorithm can also be found in Section 2.3.3.

The spring electrical embedding yielded a list of 3 dimensional coordinates for each organism, and thus a point cloud in $\mathbb{R}^3$. Each point cloud could then be loaded into JavaPlex, and a Vietoris-Rips complex was formed in dimensions 0, 1, and 2. This calculation produced homology class representatives in each dimension, including the birth/death times, a barcode (See figure 2.2), and the
persistence intervals.

Once each genome had a barcode associated to it, they could then be compared to one another using two methods: Bottleneck distance and Earth Mover’s distance. The barcodes are compared so that shape similarities may be examined. Two organisms with ‘similar’ barcodes will have persistent homology classes which are born and die at similar filtration times, and will therefore have similar shape features present in the point cloud (e.g. number and size of loops). This comparison will later reveal that such organisms will also have a similar taxonomy.

The first method is quite popular in the persistent homology community, as it is stable under small perturbations of the barcodes themselves [7]. In order to calculate Bottleneck distance, a persistence diagram first had to be created for each barcode. The endpoints of each bar in a barcode are real numbers, and so one may take the birth time and the death time, and plot them in $\mathbb{R}^2$. This is yet another way to visualize the information provided by persistent homology groups. Each point in a persistence diagram represents a particular persistent homology class. Points close to the line $y = x$ represent classes which were only alive for a relatively short period of time, while points well above the diagonal represent those classes which persisted. Notice that since the death time is always at least as big as the birth time, then points cannot lie below the diagonal. A description of the Bottleneck distance algorithm can be found in Section 2.3.5.

The other distance used to compare barcodes was the Earth-Movers distance, or EMD. This metric was used not only to provide further evidence of the ability of persistent homology to distinguish organisms of different taxa, but also because EMD provides a natural way of considering distance between barcodes. EMD considers distances between all points in a persistence diagram, whereas Bottleneck distance only considers distance between the “best worst case” points.

In order to define EMD, persistence diagrams are once again used, with a
Figure 2.2: Three barcodes of organisms of varying complexity. Top: Aciduliprofundum MAR08 339. Center: Clavibacter michiganensis nebraskensis NCPPB 2581. Bottom: Chlamydophila psittaci RD1.
few minor modifications.

It is often the case that more than one point will have the same 2 dimensional coordinates in the persistence diagram, and this is taken into account by adding a weight to each point based on the number of points in the diagram with those coordinates. For example, if three points have the coordinates (2.5, 4), then the weight of that point in the diagram will be three. Once more, a full description of EMD can be found in Section 2.3.6.

The output of both EMD and the Bottleneck algorithm is a pairwise distance matrix $D$ for each dimension, where $D_{i,j}$ is the corresponding distance between the barcodes of organism $i$ and organism $j$. These matrices then underwent a clustering process. The first step was to create a dendrogram using the distance matrix $D$. A dendrogram is a tree diagram where the leaves represent the organisms sampled, and two organisms are children of the same parent node if they are within some distance in $D$ designated by the height of the parent node.

The information from dendrograms is obtained by making horizontal ‘cuts’ at height $h$ in the tree, resulting in a partition of the data into clusters where each organism is within distance $h$ of the other organisms in that cluster. Fixing the number of clusters one wishes to obtain specifies the height at which the ‘cut’ needs to be made.

Next, the number of unique members of each taxonomic rank (kingdom, phylum, class, etc.) was determined and used as the number of clusters into which the data would be partitioned. Once the data was clustered, each cluster needed to represent a member of the taxonomic rank in question. We chose a “majority vote” method to determine cluster identity.

Example 2.2.1. Suppose one were to cluster by phylum. There are 23 unique phyla that the organisms could belong to, so fix the number of clusters to be 23. Each cluster contains one or more of the organisms in the sample, and is labeled as the phylum that the majority of the organisms in the cluster belong to.
Once this process is complete, and each cluster has an associated set of organisms belonging to it, and a taxonomic branch, then the sensitivity and specificity were calculated for each unique element in that taxonomic rank. For the example above, there would be 23 different sensitivity and specificity calculations, based on the number of samples identified as belonging to that particular phylum, as well as the organisms which actually belong to that phylum (true positive, true negative, false positive, false negative). The sensitivity and specificity of each taxonomic rank was averaged, weighted by the number of samples. Again, this was done for each dimension, and for both distance metrics (Bottleneck distance and EMD).

### 2.3 Selection of Algorithms

#### 2.3.1 Bcalm

Bcalm is a program which reduces a de Bruijn graph by eliminating well-behaved vertices while still maintaining the topological features of the graph [6]. Vertices in the de Bruijn graph begin as $k$-mers. If a vertex has degree exactly 2, then it is a candidate for reduction. Each edge in a de Bruijn graph can be interpreted as a $k+1$-mer. For example, if vertex AAAC is adjacent to vertex AACT, then the edge between them can be thought of as AAACT. So if $(v_0, v_1), (v_1, v_2) \in E$, and $deg(v_1) = 2$, then $v_1$ can be removed from the graph entirely, resulting in a relabeling of $v_0$ from a $k$-mer, to a $k+1$-mer (See Figure 2.3).

![Figure 2.3](image)

Figure 2.3: Example of the Bcalm reduction process. The resulting de Bruijn graph has fewer vertices, but retains the information of the original graph.
2.3.2 High Dimensional Embedding

This algorithm uses principal component analysis to embed graphs into a higher dimensional space such that the distance between each point is preserved as much as possible [10]. Naturally, the higher the target dimension, the better this preservation process performs. Considering the number of vertices in each de Bruijn graph after running Bcalm (typically a few thousand), 3 dimensional space seemed too low to properly embed the graph. Once JavaPlex formed the Vietoris-Rips complex of the high dimensional embedding, unwanted homology classes began to appear in dimension two: the points which formed the simplices involved in such a class were nowhere near each other on the graph itself. The cause of this was that the high dimensional embedding tended to clump many of the larger loops together (See Figure 2.4).

![Figure 2.4: High Dimensional Embedding applied to Acetobacter pasteurianus 386B. The larger loops are clumped together, causing unwanted persistent homology classes.](image-url)
2.3.3 Spring Electrical Embedding

The spring electrical embedding is an iterated algorithm which minimizes the energy of the system at each iteration [10]. The coordinates found using this method appeared to be a much more natural embedding (See Figure 2.5). The output from Mathematica was a list of 3-tuples, representing the $x$, $y$, and $z$ coordinates of each vertex embedded in $\mathbb{R}^3$.

![Spring Electrical Embedding](image)

Figure 2.5: Spring Electrical Embedding applied to Acetobacter pasteurianus 386B. Compared to High Dimensional Embedding, this embedding provides a more natural representation of the graph.

2.3.4 JavaPlex

JavaPlex is a MATLAB tool used to create various streams of simplicial complexes, including the Vietoris-Rips complex, the witness complex, the lazy witness complex, and explicit simplex streams (manual input of simplices and their birth times) [1]. For each bacterial genome, an explicit stream was created by
taking each vertex in the de Bruijn graph as a 0-simplex, and each edge as a 1-simplex. This was done in order to determine the actual betti numbers for each graph. After the graphs underwent the embedding process, Vietoris-Rips complexes could be formed using the point cloud data. The Vietoris-Rips algorithm takes the following parameters: Point cloud, maximum dimension, maximum filtration time, number of divisions. The max filtration time was chosen to be 8, as this allowed much of the topological activity to transpire, while still remaining within the realm of computability. The number of divisions was chosen to be 100. This number corresponds to how many simplicial complexes were used in the filtration, and 100 was more than adequate to get a fine approximation of the topological activity. $\mathbb{Z}_2$ coefficients were used when computing the persistent homology of the simplex stream.

2.3.5 Bottleneck Distance

Computing Bottleneck distance is a popular method of comparing the similarity/dissimilarity between two persistence diagrams, as it is stable under small perturbations of points in the diagram [7]. The definition and algorithm for computing Bottleneck distance is outlined below.

In each dimension, each sample’s barcode is pairwise matched to another, and the two persistence diagrams are combined, with points in sample A’s diagram labeled as A, and points in sample B’s diagram labeled as B.

**Definition 2.3.1.** The **Bottleneck distance** between persistence diagrams $A$ and $B$ is $d_{BN}(A, B) = \min_{a \in A}\{\max\{\rho(a, b) | b \in B\}\}$ where $\rho(a, b)$ is the Euclidean distance between points $a$ and $b$ in the persistence diagram.

To compute the bottleneck distance between $A$ and $B$, points from the larger of the two sets are projected to the line $y = x$ until $|A| = |B|$. Padding the diagonal with additional points does not affect the structure of the barcode, as these points represent intervals with the same birth time and death time. Thus
no actual measurable intervals were added to either set. Each point is then
given a unique identifier, so that repeated points are not counted as one.

A bipartite graph \( G = (A, B, E) \) is then formed between vertices in \( A \) and
\( B \). This ensures that for each \( a \in A \), there is an edge between it and each point
in \( B \). Weights were assigned to each edge in \( G \), with \( w(a, b) = \rho(a, b) \), where
again \( \rho \) is Euclidean distance between points in the diagram.

From here, edges are successively deleted if they have maximal weight, i.e.
delete edge \((a, b) \in E \) if \( w(a, b) = \max_{(c, d) \in E} (w(c, d)) \). At each stage, the
subgraph \( G' \subset G \) is formed where the edge has been deleted, and checked to
see if there is still a maximal matching. If there is, the process is continued. If
deleting edge \((a, b) \in E \) causes the graph to no longer have a maximal matching,
then \( d_{BN}(A, B) = w(a, b) \).

This satisfies the definition of Bottleneck distance, and each pairwise distance
between barcodes was stored in a matrix for further analysis. Again, this was
done in each dimension (0, 1, and 2).

### 2.3.6 Earth Mover’s Distance

The intuition behind Earth Mover’s distance comes from a humble analogy.
Suppose a farmer had some number of piles of dirt, with some piles bigger than
others, and he wanted each pile to be the same size. Taking dirt from the large
piles and adding it to the small piles, the farmer eventually accomplishes this
task. Moving dirt from one pile to another has a cost, however, and the cost
is determined by how much dirt needed to be moved, as well as how far the
dirt had to travel between piles. The Earth Mover’s distance would then be the
total cost of evening out the piles in the most efficient way possible.

This analogy can be applied to persistence diagrams, by taking the weight
of each point as the size of the pile, and the Euclidean distance between points
to be the distance between each pile (See Figure 2.6).

Consider the sets \( A \) and \( B \) described above, which contain points correspond-
Figure 2.6: Persistence diagram of two organisms along with flow indicating how far each point traveled in EMD. Circles: Acidothermus cellulolyticus 11B. Pluses: Aciduliprofundum boonei T469.

To persistence intervals for a sample. Both sets are converted to histograms. The bins represent the distinct points in A or B, and the bin height keeps track of the repeats. The EMD is then the sum of the Euclidean distances each point in A moves to mimic B. In other words, the EMD is calculated by turning the histogram for A into the histogram for B (or vice versa), weighted by a cost function which takes into account the distance between points in the histogram as well as the bin height. The MATLAB package FastEMD (see [21] and [22]) was used to calculate all pairwise distances in each dimension, yielding a distance matrix $D$.

2.4 Results

The results of the experiment are divided into four sections: an examination of the betti numbers in the explicit case, establishing the validity of the spring elec-
metrical embedding, discussing the biological interpretation of persistent homology classes in dimension 2, and the results of the clustering process.

2.4.1 Betti Numbers

In Figure 2.7, we showcase the histogram of the betti numbers of the explicit simplex stream in dimension 1 for the subset of 399 organisms. Recall that only the largest connected component was used, and so in dimension 0, every organism will have $b_0 = 1$. Also note that since the graphs themselves have trivial homology groups for dimensions $n \geq 2$, then all betti numbers will be zero in those dimensions. As expected, the majority of organisms sampled had relatively small betti numbers, since most bacteria have simple genomes (e.g. circular genomes).

![Histogram of the betti numbers of the 399 organisms sampled.](image)

Below, in Figure 2.8, is the histogram of the betti numbers of the 2,786 bacterial organisms (no plasmids) downloaded from the NCBI. It is worth not-
ing that both histograms follow similar exponential decay, with a few outliers. This shows that the 399 organisms selected are representative of the bacteria population as a whole, rather than a targeted sample of unrealistically diverse organisms.

Figure 2.8: Histogram of the betti numbers of 2768 bacterial genomes, including the 399 organisms used in the experiment.

2.4.2 Validity of the Spring Electrical Embedding

In this section, we examine the betti numbers of the de Bruijn graph of the organisms when the graph itself is considered as a simplicial complex, and compare them to the betti numbers obtained from the Vietoris-Rips stream. Since the spring electrical embedding algorithm was used to turn each graph into a point cloud, this comparison will show that the embedding does indeed reflect the structure of the graph itself.

In Figure 2.9, we examine the explicit betti numbers for Aciduliprofundum boonei T469, and contrast these with the barcode obtained from the spring
electrical embedding. The actual betti numbers are $b_0 = 1$ and $b_1 = 98$. The shape features are difficult to capture with a single filtration time, as there are numerous loops of widely varying sizes. At filtration time $t = 3.32$, we see that the Vietoris-Rips complex has become connected, and offers a betti number that encapsulates many of the cycles present in the graph. After this time, many of the classes of smaller loops die, while larger loops are born. The graphs with more uniformly sized loops (or no loops) contained a filtration time which yielded betti numbers matching the explicit case perfectly.

Figure 2.9: The de Bruijn graph for Aciduliprofundum boonei T469, as well as a comparison of the spring electrical barcode (right) with the explicit (left).

A similar examination is performed for Aciduliprofundum MAR08 339 in Figure 2.10. Here the explicit betti numbers were $b_0 = 1$ and $b_1 = 81$. The filtration time $t = 2.71$ yields persistent betti numbers which closely match the
explicit betti numbers. Again, the information is better absorbed as the full barcode, as the birth/death times for the persistent homology classes better reflect the shape of the graph than any one particular filtration time.

Figure 2.10: The de Bruijn graph for Aciduliprofundum MAR08 339, as well as a comparison of the spring electrical barcode (right) with the explicit (left)

Nevertheless, the spring electrical embedding has proven to be not only a visually appealing and natural way to form a point cloud, but also a very useful embedding for reflecting the taxonomy of the organism.

2.4.3 Persistent Homology in Dimension 2

The homology of the de Bruijn graph of an organism has a clear biological interpretation, as explained in Section 1.2. The homology groups obtained in this case are only potentially nontrivial in dimensions 0 and 1. The de Bruijn
graphs were projected into $\mathbb{R}^3$, however, granting potentially nontrivial persistent homology classes in dimension 2 as well. The consequence of this is that the interpretation of 2-cycles is somewhat unclear.

When examining the barcodes of the 399 organism subset, it was quite typical that any 2-dimensional persistent homology class representative had a relatively small persistence interval. This phenomenon often occurs naturally when forming a Vietoris-Rips complex, and so many of these intervals can be attributed to noise.

Occasionally an organism had a 2-dimensional class which did persist (see Figure 2.11). In such cases, the minimum number of triangular faces required to create the class was 8 (see Figure 2.12).

One possible explanation for these persistent classes is that they indicate the presence of a knotted center in the de Bruijn graph. Many of the de Bruijn graphs of the organisms examined have a few large loops connected to each other via a tangled clump. Throughout the course of the filtration, simplices in dimension 2 are more likely to be formed in this clump than on the larger loops, allowing for nontrivial homology in this dimension. De Bruijn graphs without this clump tended not to have any homological activity in dimension 2.

### 2.4.4 Clustering Based on Taxonomy

The idea behind clustering based on the taxonomy of an organism is that organisms which are taxonomically similar should have similar genetic makeup, hence similar de Bruijn graphs. With this in mind, the organisms are grouped based on their taxonomy in the hopes that the de Bruijn graphs in each group are similar. This is equivalent to saying that the distance between the barcodes of the organisms should correlate to the distances between the taxa of the organisms.

In Table 2.1, the sensitivity and specificity are shown for each taxonomic rank using the clustering process for EMD. The distance matrices in dimensions 0, 1, and 2 obtained from the EMD algorithm were summed to produce a single matrix, and then clustered by taxonomic rank.
Figure 2.11: The de Bruijn graph and barcode for Bdellovibrio bacteriovorus Tiberius. Highlighted is a 2-cycle which persists. This particular cycle consists of 108 triangular faces.

The dendrogram for the EMD clustering process is shown in Figure 2.13. Specific areas of the dendrogram are highlighted, with leaves labeled as the de Bruijn graphs of the corresponding organisms. It is clear that this process clusters similar de Bruijn graphs together effectively.

Table 2.2 displays the sensitivity and specificity for each taxonomic rank using the clustering process for Bottleneck distance. The dimension 0 results were not completed in time for analysis, and so are not included. Instead, the distance matrices in dimensions 1 and 2 were summed and clustered. In order
Figure 2.12: Example of a minimal nontrivial 2-dimensional homology class present in the data.

<table>
<thead>
<tr>
<th>Tax. Rank</th>
<th>Ave. Sensitivity</th>
<th>Ave. Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kingdom</td>
<td>95%</td>
<td>27%</td>
</tr>
<tr>
<td>Phylum</td>
<td>78%</td>
<td>88%</td>
</tr>
<tr>
<td>Class</td>
<td>79%</td>
<td>90%</td>
</tr>
<tr>
<td>Order</td>
<td>89%</td>
<td>95%</td>
</tr>
<tr>
<td>Family</td>
<td>91%</td>
<td>95%</td>
</tr>
<tr>
<td>Genus</td>
<td>96%</td>
<td>97%</td>
</tr>
<tr>
<td>Species</td>
<td>96%</td>
<td>98%</td>
</tr>
</tbody>
</table>

Table 2.1: Average sensitivity and specificity of each taxonomic rank using EMD.

to produce such similar results, more clusters were needed for class, order, and family, and fewer were required for phylum, genus, and species. This could partially be due to the lack of influence in dimension 0. The similarity between sensitivity/specificity using EMD and Bottleneck distance indicates that the effectiveness of persistent homology is independent of choice of distance metric.

Figure 2.14 shows the dendrogram of the 399 organism subset using Bottleneck distance. The distances between organisms is more contrasting than in the EMD case, leading to a variety of node heights. However, both distance metrics cluster the de Bruijn graphs effectively.

Due to the fact that so few organisms had any homological activity in dimension 2, the distance between each pair of organisms (EMD or Bottleneck) was much smaller than in other dimensions. Hence, clustering using the distance matrix in dimension 2 alone was much less effective. Summing the distance
<table>
<thead>
<tr>
<th>Tax. Rank</th>
<th>Ave. Sensitivity</th>
<th>Ave. Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kingdom</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>Phylum</td>
<td>70%</td>
<td>87%</td>
</tr>
<tr>
<td>Class</td>
<td>75%</td>
<td>93%</td>
</tr>
<tr>
<td>Order</td>
<td>94%</td>
<td>94%</td>
</tr>
<tr>
<td>Family</td>
<td>95%</td>
<td>94%</td>
</tr>
<tr>
<td>Genus</td>
<td>96%</td>
<td>94%</td>
</tr>
<tr>
<td>Species</td>
<td>97%</td>
<td>96%</td>
</tr>
</tbody>
</table>

Table 2.2: Average sensitivity and specificity of each taxonomic rank using Bottleneck distance.

matrices for each dimension produced the best results, as the disparity between taxa was more stratified.

Both Table 2.2 and Table 2.1 follow a similar pattern. Only 2 kingdoms were present in the data, and most organisms were clustered into a single kingdom regardless of distance metric, which explains the poor specificity in the kingdom row. Moving downward from kingdom, to phylum, to class, and so on, the results were progressively better. This is because phylum, class, and order are more broad categories that organisms may fall under, whereas family, genus, and species are more specific. In other words, organisms belonging to the same phylum may not be very taxonomically similar at all, and so their de Bruijn graphs may be very different. Organisms belonging to the same species, however, will have identical taxonomy in the higher ranks, and so the DNA (and therefore the de Bruijn graphs) of such organisms will be more closely related.

### 2.5 Discussion

In this experiment, we have explored de Bruijn graphs and persistent homology, and used them as a tool to distinguish the taxonomy of various bacterial organisms. After associating to each organism a de Bruijn graph, Vietoris-Rips complexes and barcodes were found, and later clustered based on both Earth Mover’s distance and Bottleneck distance. In both cases, we have shown evidence that persistent homology is a sufficiently powerful tool for distinguishing between organisms of different taxa. Not only does persistent homology
faithfully reflect the global structure of the genome, but it can also be used to cluster organisms into groups that reflect the taxonomy of the organisms. This has numerous consequences in the fields of computational biology, topological data analysis, and bioinformatics. Topological methods such as persistent homology have led researchers to discover a new subtype of breast cancer [20], and to further understand interactions between hereditary disease-genes [24]. This experiment could be extended by examining more complex organisms, or investigating embeddings into even higher dimensional Euclidean space.
Figure 2.13: The Dendrogram of the 399 organism subset using EMD. The highlighted regions show that organisms whose de Bruijn graphs are alike are clustered together.
Figure 2.14: The Dendrogram of the 399 organism subset using Bottleneck distance. The highlighted regions show that organisms whose de Bruijn graphs are alike are clustered together.
Bibliography


