A Graph Theoretical Approach to DNA Fragment Assembly

Jonathan Kaptcianos‡
Saint Michael's College
Colchester, Vermont 05439 USA

Received: September 19, 2007 *Accepted: April 1, 2008*

ABSTRACT

Built on prior work on sequencing by hybridization, fragment assembly is a newly explored method of determining whether or not a reassembled strand of DNA matches the original strand. One particular way to analyze this method is by using concepts from graph theory. By constructing data models based on these ideas, it is possible to come to various conclusions about the original problem regarding reassembled strands of DNA. In this paper we will detail this approach to DNA fragment assembly and present some related graph theoretical proofs in the process, including the BEST theorem. Further, we will explore the Eulerian superpath problem and its role in aiding fragment assembly, in addition to other recent applications of graph theory in the field of bioinformatics.

I. Introduction

In recent years, scientists and researchers have emphasized DNA sequencing and fragment assembly with the hopes of enhancing their abilities to reconstruct full strands of DNA based on the pieces of data they are able to record. Complications arise with fragment assembly due to imperfect data sets. Strands are often riddled with repeats and come in varying sizes. As a result, configuring the image of the original genome is not as easy as fitting one puzzle piece into the next.

We will discuss a recently discovered approach to DNA fragment assembly that uses components from graph theory, including de Bruijn graphs and Eulerian circuits, to successfully compensate for some of these errors. Specifically, this approach, originating from work done with sequencing by hybridization, consists of constructing various directed graphs based on provided DNA data, and counting possible Eulerian circuits in these graphs.

This paper will provide some preliminary background information from graph theory. In addition to several definitions, we will prove two theorems, the BEST theorem and the matrix tree theorem, both of which are used in counting Eulerian circuits in a directed graph.

Further, this paper will briefly survey the current state of DNA sequencing and fragment assembly, such as the problems that arise, and how they are addressed using graph theory. We will see how graphs are being constructed using DNA data, and how these graphs inform the problem. Our primary emphasis falls on the Eulerian superpath problem of Pevzner, Tang and Waterman in [1-2], which is a more recent attempt to simplify DNA graphs through a series of transformations on the graph. Such transformations include different edge detachments for cases of single and multiple edges in a directed Eulerian graph.

The concluding section discusses the decomposition of DNA sequencing with nanopores, as detailed in Bokhari and Sauer [3], and the role that graph theory plays in resolving problems that arise. This is a more recent application of graph theory being put to use in the field of bioinformatics.

II. Graph Theory Definitions

Graph theory is a field of mathematics which has many applications in the world of science. We assume the reader is familiar with the basic foundations of graph theory, such as that found in Tucker

‡ jkaptcianos@smcvt.edu

[4], and West [5]. The basic terminology here follows that of West [5].

One can move through the elements of a graph in several different ways. A walk is any journey through a graph which produces a list of vertices and edges such that an edge connects the vertices on either side of it. A trail is a walk where no edges are repeated. A closed trail is a circuit, and the list of vertices is expressed in cyclic order. Further, a path is a trail in which no vertices are repeated such that the vertices can be listed in consecutive order as one is adjacent to the next. In the case of a directed graph, the edges of a trail or circuit must be consistently oriented.

A trail or circuit which traces every edge in a graph once and exactly once is considered Eulerian. Eulerian trails and circuits are allowed to pass through vertices in a graph more than once. The number of Eulerian circuits in a graph can be computed with a formula generated from the BEST theorem, which will be proven later in this paper.

A final definition from the field of graph theory which we use in this paper is that of de Bruijn graphs. A de Bruijn graph is a directed graph with vertices that represents sequences of symbols from an alphabet, and edges that indicate where the sequence may overlap, as defined in de Bruijn [6].

The construction of this graph is dependent on the set of *l*-length pieces, or fragments, from the particular sequence at hand. Each vertex is labeled by a fragment of length $l - 1$ and the directed edge existing between two vertices represents one of the *l*-length fragments. Specifically, the first symbol in the fragment comes from the vertex that sends the edge, and similarly, the last symbol comes from the receiving vertex. Thus, the remaining symbols that both vertices contain are found labeling the edges.

For example, we construct a de Bruijn graph for the sequence "0110101" using fragments of length $l = 3$. The four triples that are present in the sequence are 011, 110, 101 (which appears twice), and 010. Figure 1 shows the de Bruijn graph that coincides with this sequence. Notice the vertices represent the first and last subsequence of length 2 in each triple, and the directed edge between two vertices represent the respective triple.

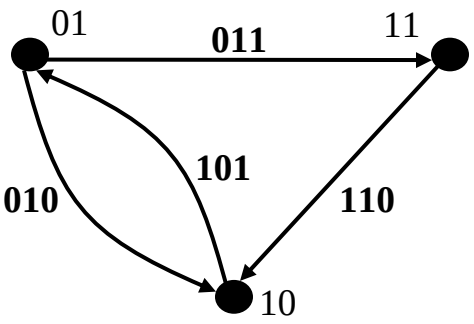

Figure 1. A de Bruijn Graph for the sequence "0110101" with fragments of length 4.

III. DNA Sequencing and Fragment Assembly

For simplicity, consider DNA sequencing to be much like the process of constructing a children's puzzle, as expressed by Pevzner, Tang, and Waterman [1]. The only difference lies in the fact that scientists are dealing with hundreds of pieces of varying sizes, and some pieces are exactly identical to one another. The "puzzle pieces" are various strands of DNA reads and the completed puzzle is what the entire original strand, or genome, looks like. The provided information on this aspect of Bioinformatics comes from various sections of the text by Jones and Pevzner [7].

Specifically, DNA fragment assembly is a lab technique which determines the configuration of an entire genome given a series of viewable reads from that particular genome. Since a full strand of DNA consists of millions of nucleotides, there is no way to visibly capture the structure in its entirety. With today's technology, scientists and researchers are able to look at fragments of DNA anywhere from 500 to 1200 nucleotides long. These fragments all consist of the assortment of the four types of bases which make up DNA: adenine (A), thymine (T), guanine (G), and cytosine (C). There could be multiple ways to reconstruct

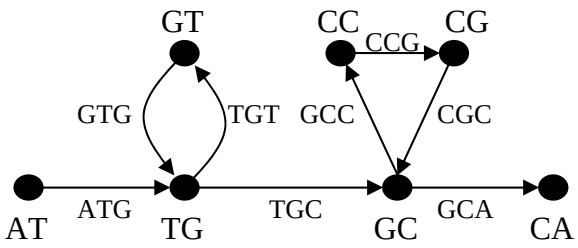

Figure 2. A de Bruijn graph for the sequence ATGTGCCGCA.

the original strand out of the fragment pieces but only one of them is correct.

There are many problems and complications that arise in fragment assembly which make it much more difficult than simply constructing a puzzle. Within the portrayal of the viewable reads there exists a rate in which errors are produced by sequencing machines. Anywhere from 1% to 3% of DNA reads resulting from sequencing may contain errors and do not appropriately represent a part of the full strand.

Further, another problem lies within the fact that DNA is a double-stranded structure. Based on the work of Watson and Crick [8], we know that with each single strand of DNA there exists a complement, as A matches with T, and C is complementary with G. Thus, upon looking at reads from a strand of DNA, we are unable to determine whether or not a fragment is coming from the desired strand, or its complement.

While the error rates and complement confusions do hinder some steps in DNA sequencing and fragment assembly, the major area of concern and confusion comes from repeated sequences. The human genome has a large number of sequences that repeat multiple times. Further, if a repeating sequence is larger than the size of the viewable reads, it would make construction of the genome almost impossible. For example, consider a particular genome which has a repeating sequence spanning a stretch of 2000 nucleotides. No sequencing machines today can view a read of this length, so there is no way to ever see the repeat in its entirety.

Most attempts at correcting errors in fragment assembly aim to attack the problem of repeats. Previous approaches for doing so followed the "overlap-layout-consensus" algorithm. The first step involves matching all possible reads and finding any overlapping. This is done by looking at the beginning sequence of one read and the ending of another. The layout step is the construction of the strand, and it proves to be the most difficult. Keeping repeats in mind, an attempt is made to find the order of reads along a full strand of DNA and put them together. The last component of this algorithm is deriving how the sequence will appear based on the layout produced in the previous step. The approach we will discuss in this paper abandons the traditional outline of this three step process, and in doing so, becomes a matter of probability.

IV. Resembling Strands of DNA with de Bruijn Graphs

Sequencing by hybridization is a lab technique which similarly looks at fragments of DNA, known as DNA arrays, but is not to be confused with fragment assembly, as they are different. Specifically, sequencing by hybridization (SBH) relies mostly on the binding of an unknown target fragment of DNA with an array of shorter synthetic fragments known as probes. These probes are anywhere between 8 and 30 nucleotides in length, and they bind to the target based on the Watson-Crick complements, as noted in Pevzner [9].

However, it was a graph theory approach to SBH by Idury and Waterman [10] that introduced similar approaches in the field of DNA fragment assembly. In their article, the authors defined the rules to construct a directed graph based on a sequence of DNA. More specifically, these

rules were for constructing a de Bruijn graph based on DNA pieces of k-length. The graph had vertices labeled by fragment of length *k*-1, and edges labeled by fragments of length *k*, which connected two adjacent vertices in the sequence.

Figure 2 shows a de Bruijn graph for the simple sequence ATGTGCCGCA, as it was used by Idury and Waterman [10]. Here, the desired fragments of DNA are of length 3. Thus, the vertices are labeled by fragments of length 2 and edges labeled by fragments of length 3, representing the particular triple from the sequence. Notice how there is only one possible Eulerian trail in the graph above, as this is just a simple case. Various work done by Pevzner in [9, 11] has developed these ideas further.

Now, we will construct another de Bruijn graph using the same rules as above. Given the set of fragments *S* = {ATG, TGG, TGC, GTG, GGC, GCA, GCG, CGT}, the graph would similarly possess vertices of length 2 and edges of length 3. However, the graph produced for this set of reads is more complex than the previous graph. Specifically, there are two different closed trails through the entire graph, producing the sequences ATGGCGTGCA and ATGCGTGGCA. See Figure 3, and note that the extra dotted edge is added to connect the first and last vertices AT and CA, creating an Eulerian digraph.

As previously mentioned, this approach to DNA fragment assembly, referred to by some as the Eulerian path approach, creates a question of probability. With this example above, we see that this given set of fragments constructs two different sequences, based on there being two differing Eulerian circuits. Therefore, there is a ½ possibility that we will resemble the actual genome, depending on which Eulerian circuit we follow. Since we added the extra edge making the graph closed, we are able to count Eulerian circuits as opposed to trails.

More generally, when we are given complete and errorless data, we find that the probability of reconstructing the original strand of DNA given a set of fragments is:

$$\frac{1}{\text{total \# of Eulerian circuits in the de Bruijn graph}}$$

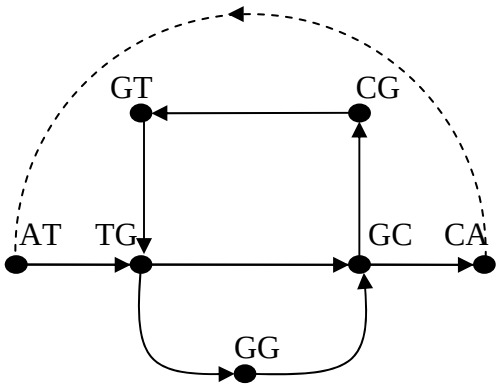

Figure 3. The Graph for the set of reads **S**.

With that, there is a theorem used to count Eulerian circuits in a directed graph, known as the BEST theorem, which we will prove, followed by another related theorem and proof.

V. The BEST Theorem

There exists a formula for computation of Eulerian circuits in a directed graph. Named after its inventors, de Bruijn, van Aardenne-Ehrenfest, Smith, and Tutte, [12-13], the BEST Theorem reads as follows:

Theorem 5.1. Given a connected directed graph G and set of vertices $V(G) = \{\nu_1, \ldots, \nu_n\}$ all of even degree, the number of Eulerian circuits $|s(G)|$ is expressed as the following, where $|t_i(G)|$ is the number of spanning trees rooted towards any vertex ν_i in G:

$$\left|s(G)\right| = \left|t_i\left(G\right)\right| \prod_{j=1}^{n} (d^+(v_j) - 1)!\,.$$

The following proof for this theorem is indicated in Bollobás [14].

Proof: We are given a directed multigraph G with a vertex set $V(G) = \{\nu_1, \ldots, \nu_n\}$ and the outdegree and indegree of a vertex ν_i being equivalent, which is denoted $d^+(\nu_i) = d^-(\nu_i)$. If these conditions are met, then we know that G has at least one directed Euler circuit, by a theorem found in Bollobás [14].

Further, let's assign E as the set of directed Euler circuits, and E_i as the set of

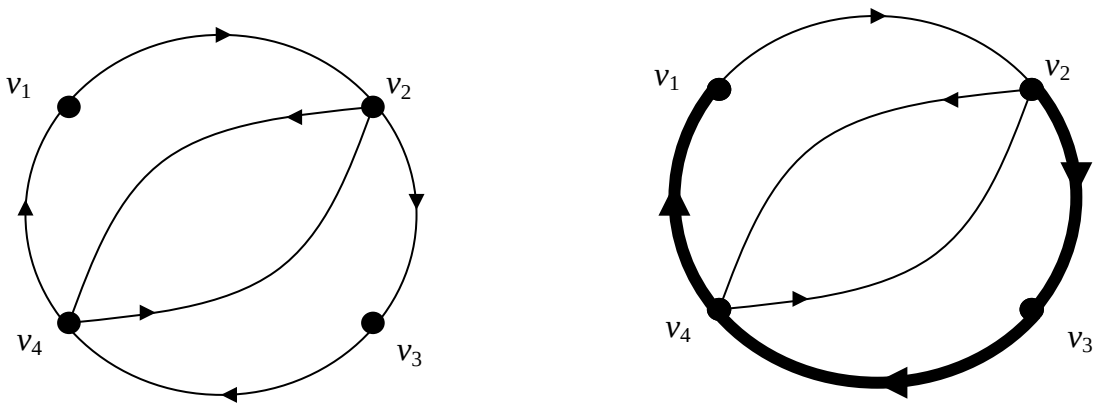

Figure 4. By using the Eulerian cycle $\nu_1\nu_2\nu_4\nu_2\nu_3\nu_4\nu_1$ in graph *G*, this map produces the spanning tree oriented toward ν_1 indicated in bold in the graph on the right.

directed Euler trails starting and ending with vertex ν_i. Since the number of times that each Eulerian circuit will pass through vertex ν_i is equivalent to indegree (or outdegree, equivalently) of ν_i, then the number of Euler trails starting with ν_i can be expressed as

$$|E_i| = d^+(\nu_i)|E| = d^-(\nu_i)|E|$$

The above equation states that the number of Eulerian trails starting and ending with v_i is equivalent to the either the indegree or outdegree of ν_i multiplied by the total number of Eulerian circuits in *G*.

Now, we let T_i be the set of spanning trees oriented towards v_i and use it in the mapping function $f_i: E_i \rightarrow T_i$. This map has an input of the set of Euler trails starting and ending with ν_i in *G* and an output of the set of spanning trees oriented towards ν_i in *G*. For simple notation in this proof, we'll use $i = 1$, so $f_1: E_1 \rightarrow T_1$.

Given *S* as an Euler trail from the set E_1, this map constructs a set of edges e_j in the following manner. The first edge in *S* exiting ν_1 is not drawn, meaning that $j = 2,\ldots,n$ for each edge e_j. Further, each edge e_j appears as *S* passes through a vertex ν_j for the last time and doesn't return to it again in the Eulerian trail. The result from this map is a directed tree *T*, with vertices $\nu_1,\ldots, \nu_n$ and edges $e_2,\ldots,e_n$ that is in the set of spanning trees oriented towards vertex v_1. Figure 3 shows an example of this map based on a specified Eulerian cycle in *G*.

Since here we are claiming that $T \in T_1$, we must prove that a) *T* is indeed a tree, and that b) *T* is oriented towards the vertex ν_1.

To show (a) we first assume that *T* contains a cycle *C*, which would not contain ν_1. Also, since *C* is a cycle, each vertex in *C* has an outdegree of 1. Assume that edge e_g is the last edge of Euler trail *S* on cycle *C*, which connects the vertices ν_g and ν_h. If this is the case, then *S* is arriving back to ν_h despite having left it for the last time after traveling through edge *eh* previously. This contradicts the corresponding map, which states that an edge e_j follows the last visit to vertex ν_j, and will not return to ν_j again. Therefore, *T* is a tree.

Now, we prove (b) by asking whether or not *T* is oriented towards ν_1. We answer this by first assuming that T contains a path $\nu_k\nu_{k-1}\ldots\nu_1$. The edge between vertices ν_2 and ν_1 is e_2, as there is no e_1. Further, the edge between vertices ν_3 and ν_2 could be either e_2 or e_3, but since e_2 has already been assigned, it must be e_3. As we proceed in this same manner between all vertices in the path, we this prove that the path $\nu_k\nu_{k-1}\ldots\nu_1$ is indeed oriented towards the vertex ν_1.

Thus, the digraph *T* is in the set of spanning trees oriented towards vertex ν_i in the original graph *G*. Therefore, we can conclude that $T \in T_1$ and continue with this proof.

To get the desired map f_1: $E_1 \rightarrow T_1$, we must set $f_1(S)=T$. Now, working backwards, the set $f_1^{-1}(T)$ describes the all the Euler trails about vertex ν_1 that give rise to the spanning tree T.

Thus, using the fact that $f_1(S) = T$ and $S \in E_1$, we can draw an Euler trail S in the following manner. We begin at an edge which leaves vertex ν_1. Once we return to ν_1, we leave it through another unused edge, which also must exit ν_1; when there are no untouched edges exiting ν_1, the trail ends. Now, upon arriving at ν_j (where $j > 1$), leave this vertex by an edge that is not e_j and also not in the tree T; once these edges have been exhausted, then leave ν_j through the edge e_j.

Based on the fact that the indegree and outdegrees of any vertex ν_j in G are equivalent, we have successfully found an Euler trail S which is in the set E_1 such that $f_1(S) = T$. Note that, for some spanning trees T, there will be more than one Euler trail S that can be produced.

Using the graph G as an example, we show the process of using the spanning tree T (as shown darkened in Figure 3) to construct the Euler trails S that gives rise with it. Start at the edge which leaves ν_1 towards ν_2. Once at ν_2, we must use the edge towards ν_4, as the one towards ν_3 is the "e_j" edge used in the tree T. Notice at ν_4, there are two outgoing edges, since the one towards ν_1 is also part of T, we must use the edge going back to v_2. Having returned to ν_2, the only remaining edge is the one from T, so we use that. The same follows for ν_4, as the only remaining edge, also from T, completes the Eulerian trail S.

The primary concept in this process is as follows: once we enter any vertex in the graph, we exit that vertex by a non-tree edge for as long as possible, until we are only left with a tree edge. Continuing to do so for each vertex produces an Eulerian trail in the graph.

In general, we can express the number of Euler trails S that can be produced with a given spanning tree T oriented towards ν_i (denoted as $|f_1^{-1}(T)|$) as follows:

$$\left| f_1^{-1}(T) \right| = d^+(v_i)! \prod_{j=2}^{n} \left(d^+(v_j) - 1 \right)! \quad .$$

The factorial of the outdegree of vertex ν_1 considers number of different ways the Euler trail can begin (which explains why the size of E_1 is greater than E), and the product factorial considers that there is one less available outgoing edge at vertex ν_j as the Euler trail S passes through.

Furthermore, upon focusing on the entire set of Eulerian circuits E and disregarding a set vertex ν_i to start from, the number of said circuits is as follows:

$$|E| = |T_1| \prod_{j=1}^{n} \left(d^+(v_j) - 1 \right)! \quad .$$

Here, the product factorial for outdegrees of all vertices in G is multiplied by the total number of spanning trees T_1 oriented towards the vertex ν_1.

With this series of equations, we have thus proven and verified the accuracy of the formula which is associated with the BEST theorem. ■

The BEST theorem relies on the number of spanning trees which exist in a particular Eulerian digraph. We now need a theorem which enables us to determine the number of spanning trees in a directed Eulerian graph. This theorem has been referred to as the matrix tree theorem, and a full understanding of its proof requires some further definitions. The following terms and definitions follow from texts by Fleischner [15-16], Bollobas [14], and Pevzner [9].

Given a directed Eulerian graph G, it is possible to construct various matrices which encode information about the properties of the graph. One matrix, the adjacency matrix, lists the number of edges between various vertices in the graph. It is an $n \times n$ matrix where n is the total number of vertices in G. If a_{ij} is the number of directed edges that leave a vertex ν_i and enter a vertex ν_j in G, then a_{ij} is the i,j-th entry of the matrix. The i,i-th entries, which are the diagonal of the matrix, will be zero unless G has loops.

Recall the graph G from Figure 3 that we just used previous. Its adjacency matrix is denoted here as A(G).

$$\mathrm{A}(G) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

Notice the following: a) ν_1 sends an edge to ν_2, hence a 1 in the 1,2-entry; b) ν_2 sends edges to ν_3 and ν_4, hence 1's in the 2,3 and 2,4 entries; c) ν_3 sends an edge to ν_4, hence a 1 in the 3,4-entry; and d) ν_4 sends edges to ν_1 and ν_2, hence 1's in the 4,1 and 4,2 entries. Also, since there are no multiple edges of the same direction in *G*, there are only 1's in the entries. Further, the diagonal contains all zeros because there are no loops in *G*.

A second matrix that can be constructed from a graph is the diagonal matrix. This matrix gives the indegrees of each vertex in the graph along its diagonal. All other entries must be zero. Using the same graph *G*, its diagonal matrix $\mathrm{D}(G)$ is as shown:

$$\mathrm{D}(G) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}.$$

A third matrix combines both aspects of these two matrices $\mathrm{A}(G)$ and $\mathrm{D}(G)$ that we have described above. Some graph theorists will refer to this as the Kirchoff matrix, as seen in Fleischner [16], while others will call it the combinatorial Laplacian matrix, as seen in Bollobás [14]. For the sake of this paper, will we call it the Laplacian matrix, and denote it $\mathrm{L}(G)$. Specifically, this matrix is the result of subtracting $\mathrm{A}(G)$ from $\mathrm{D}(G)$.

Once again, using our graph *G* from Figure 3 as an example, the Laplacian matrix L(*G*) is as shown below:

$$\mathrm{L}(G) = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & -1 & 0 & 2 \end{pmatrix}.$$

This matrix is central to the matrix tree theorem. There are certain characteristics that all Laplacian matrices of Eulerian digraphs possess. First, the diagonal still represents the indegree of each vertex in *G*, having only removed any loops. In our example there are none, thus the diagonal stays the same. Also, for each edge that exists from vertex ν_i to ν_j, there is a -1 in the *i,j-th* entry of the matrix, where *i* and *j* are not equal.

Another characteristic of the Laplacian matrix is that columns and rows sum to zero. The reason for this is clear. Each column *i* accounts for two quantities. One of which, on the diagonal, is the indegree of vertex ν_i, excluding loops. The second is the number of edges that are received by ν_i from all other vertices in *G*, whose values here are negative. Thus, it is reasonable that by adding these together, all edges that enter vertex ν_i are being accounted for, which produces zero for the balanced digraphs we consider here.

On the other hand, each row *i* accounts for two different quantities. The first is the number of edges that leave vertex ν_i, excluding loops, and the second is the number of edges that are received by all other vertices from ν_i. These values, however, are all negative. Since we know *G* is an Eulerian digraph, the number of edges entering and number of edges exiting any vertex are equivalent. Because of this, we know that by adding the indegree of vertex ν_i, which is positive, to the summation of the edges which leave it, which is negative, will always result in zero for any $\mathrm{L}(G)$ where *G* is an Eulerian digraph.

We now recall the cofactor of a matrix. This value is found by removing the *i-th* row and *i-th* column of a given matrix, and then taking the determinant of the remaining matrix. We denote this as follows, where *k* is the *i-th* cofactor:

$$k = \det_{i,i} \mathrm{L}(G).$$

We have included the $\mathrm{L}(G)$ matrix in the formula above because, for the sake of this paper and ensuing theorem and proof, we will be considering the cofactors of the Laplacian matrices of various directed Eulerian graphs.

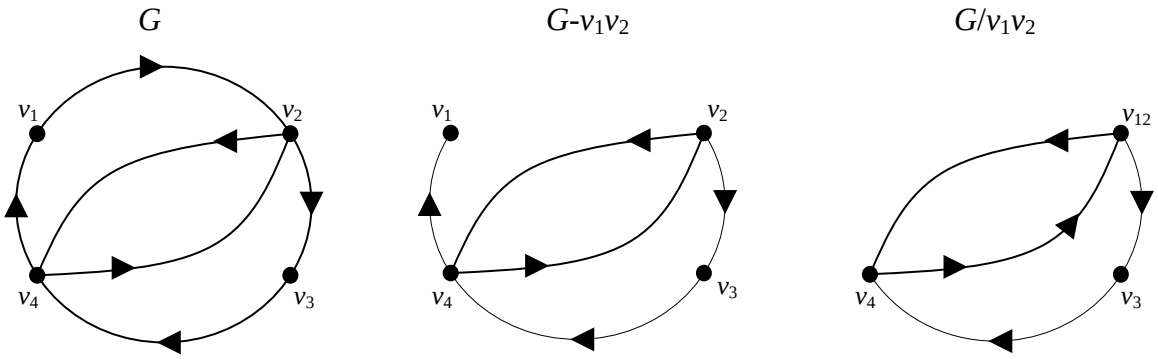

Figure 5. The graphs G, G-$\nu_1\nu_2$, and G/$\nu_1\nu_2$.

As an example, consider the Laplacian matrix for the graph G from Figure 3. The cofactor at the first row and first column is the determinant of the matrix below:

$$\begin{pmatrix} 2 & -1 & -1 \\ 0 & 1 & -1 \\ 1 & 0 & 2 \end{pmatrix}.$$

Taking the determinant of this matrix produces the value of $k = 2$. Further, if we found the other three cofactors from $\mathbb{L}(G)$, we would find all of them to be 2; the reason for this stems back to the fact that all rows and columns sum to zero, as this will hold true for all such matrices. Proof of this can be found in Fleischner [16]. It is important to note that matrices whose rows and columns do not add to zero may have different cofactors depending on which rows and columns are removed from the original matrix. This information allows us to move on to a theorem which is pivotal for full understanding of the former BEST theorem and other topics involving Eulerian graphs. As stated earlier, this theorem is referred to as the matrix tree theorem.

Theorem 5.2. Given a directed graph G with the set of vertices $V(G) = \{\nu_1,\ldots, \nu_n\}$ and a set of spanning trees $t_i(G)$ oriented towards the vertex ν_i, then $|t_i(G)|$ is equal to the cofactor of $\mathbb{L}(G)$ on the i-th row and i-th column. That is,

$$|t_i(G)| = \det_{i,i} \mathbb{L}(G).$$

The following proof for the matrix tree theorem was constructed with the aid of Bollobás [14], which proves a similar theorem involving undirected and weighted graphs for electrical networks.

Proof: We will use induction on the number of edges m in a graph for this proof. Further, we will assume that the given graph G is connected, as an unconnected graph has zero spanning trees, and that m is greater than one, as a graph with one edge only has one spanning tree.

Now, given two vertices ν_1 and ν_2 that are adjacent to each other in G, we will produce two new graphs from G which have less than m edges. The first is created by removing all the directed edges that leave vertex ν_1 and enter ν_2 in G, whose set we express as $\nu_1\nu_2$. This produces the graph G-$\nu_1\nu_2$. To make the second graph we contract these edges $\nu_1\nu_2$, producing a graph G/$\nu_1\nu_2$. In this case, the vertices ν_1 and ν_2 are fused, creating a new vertex ν_{12}. Also, the edges previously coming into each former vertex from a particular vertex ν_i are joined as one edge, with a weight totaling the summation of these individual edges. Further, all edges exiting each removed vertex into v_i are joined and weighted in the same manner. Note that both graphs G-$\nu_1\nu_2$ and G/$\nu_1\nu_2$ will have less than m edges.

Figure 5 depicts an example of producing these two new graphs from the

graph G. Notice how G-$\nu_1\nu_2$ just removes the single directed edge from ν_1 to ν_2, and $G/\nu_1\nu_2$ contracts the edge, leaving a multiple edge weighted at 2 going from the vertex ν_4 into the new vertex ν_{12}, which we express with two arrows on the one drawn edge. This compensates for the fact that ν_4 sent an edge to each of ν_1 and ν_2 in the original graph G. There are no multiple edges leaving ν_{12}, as there were no vertices in G that received an edge from both ν_1 and ν_2.

As the new graphs are produced, the following relationship on the number of spanning trees is established, where a_{12} is the total number of edges $\nu_1\nu_2$ that existed in G:

$$\left|t_i(G)\right| = \left|t_i(G-\nu_1\nu_2)\right| + a_{12}\left|t_1(G/\nu_1\nu_2)\right|.$$

Specifically, the number of spanning trees in G oriented towards vertex ν_i is equal to such spanning trees in G-$\nu_1\nu_2$ added to the product of such spanning trees in $G/\nu_1\nu_2$ and the number of directed edges $\nu_1\nu_2$. The first term on the right side takes into account all the spanning trees oriented toward ν_i where the edges $\nu_1\nu_2$ have been removed. Therefore the second term must consider all the spanning trees that do have the edges $\nu_1\nu_2$ in them. Because of this, we can represent those edges by themselves with a_{12}. Removing this amount leaves all the spanning trees oriented towards ν_i in the graph where $\nu_1\nu_2$ has been removed. We know this to be $G/\nu_1\nu_2$.

By considering the various spanning trees oriented towards any of the four vertices in our example graph G, we find this equality to be true. Taking the vertex ν_4, for example, we see there are two spanning trees oriented towards it. The first contains the directed edges $\nu_1\nu_2$, $\nu_2\nu_3$, and $\nu_3\nu_4$, and the second contains the edges $\nu_1\nu_2$, $\nu_2\nu_4$, and $\nu_3\nu_4$. With the right side of our formula and the graph G -$\nu_1\nu_2$, we notice there are no spanning trees oriented towards ν_4. Therefore, both trees must be produced from the second term of our formula above. Using both of the multiple edges at $\nu_4\nu_2$, we are able to construct two sets of trees which touch all three vertices and are oriented towards ν_4. Multiplying this by the number of edges $\nu_1\nu_2$ we removed, 1, we produce the final value 2. This is indeed equivalent to the number of spanning trees in our original graph G oriented towards the vertex ν_4.

Now that we have established a relationship for the spanning trees in G, G-$\nu_1\nu_2$, and $G/\nu_1\nu_2$, we turn to the second part of this proof. We do so by constructing a similar relationship with the cofactors of each graph's Laplacian matrices, which is as follows:

$$\det_{i,i}\mathbb{L}(G) = \det_{i,i}\mathbb{L}(G-\nu_1\nu_2) + a_{12}\det_{i,i}\mathbb{L}(G/\nu_1\nu_2).$$

In order to complete this proof, we must prove that the formula above will always be true. By doing this we can establish equality between the elements on the right side of this equation and those from the right side of the earlier equation based on $|t_i(G)|$. Keeping in mind that we are proving this theorem via induction on the number of edges m, notice that these inequalities are based on graphs of less than m edges,

Like the equation in the first part of this proof, the relation among the cofactors uses the same summation with the graphs G-$\nu_1\nu_2$ and $G/\nu_1\nu_2$, and the number of edges $\nu_1\nu_2$ removed from the original graph G. Because of this, we can approach it in the same manner, by looking at each individual component, starting first with the Laplacian matrices of each graph.

As we stated prior to this proof, a Laplacian matrix for a directed graph enumerates the directed edges from a vertex ν_i to a vertex ν_j, which is represented as a negative number in the *i,j* entry of the matrix. We will denote this as $-a_{i,j}$, where $i \neq j$. On the diagonal in $\mathbb{L}(G)$ are the indegrees of each respective vertex in G, where d_i is at the *i,i* entry and is the indegree of vertex ν_i. Thus $\mathbb{L}(G)$ for a general directed graph G with n vertices can be expressed as follows:

$$\mathrm{L}(G)=\begin{pmatrix} d_1 & -a_{12} & -a_{13} & -a_{14} & \cdots & -a_{1n} \\ -a_{21} & d_2 & -a_{23} & -a_{24} & \cdots & -a_{2n} \\ -a_{31} & -a_{32} & d_3 & -a_{34} & \cdots & -a_{3n} \\ -a_{41} & -a_{42} & -a_{43} & d_4 & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ -a_{n1} & -a_{n2} & -a_{n3} & \cdots & \cdots & d_n \end{pmatrix}$$

Now, upon removing the all the directed edges $\nu_1\nu_2$ in our general graph, the respective Laplacian Matrix would be as shown below, keeping in mind that a_{12} is the number of directed edges that existed from ν_1 to ν_2:

$$\mathrm{L}(G-\nu_1\nu_2)=\begin{pmatrix} d_1 & -a_{12} & -a_{13} & -a_{14} & \cdots & -a_{1n} \\ -a_{21} & d_2-a_{12} & -a_{23} & -a_{24} & \cdots & -a_{2n} \\ -a_{31} & -a_{32} & d_3 & -a_{34} & \cdots & -a_{3n} \\ -a_{41} & -a_{42} & -a_{43} & d_4 & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ -a_{n1} & -a_{n2} & -a_{n3} & \cdots & \cdots & d_n \end{pmatrix}.$$

Notice how only two components of this matrix are different from the original matrix $\mathrm{L}(G)$, at the 1,2 and 2,2 entries, which represent the directed edges from vertices ν_1 to ν_2, and the indegree of ν_2, respectively. The indegree of ν_2 will be decreased as the edges coming from ν_1, expressed as a_{12}, have been removed. Further, the value in the *1,2* entry will clearly become zero under the same premise.

The third Laplacian matrix, for the graph $G/\nu_1\nu_2$, will be slightly different. Since the number of vertices is being reduced from the original graph G, it follows that $\mathrm{L}(G/\nu_1\nu_2)$ will be of a smaller dimension. Specifically, it will be of dimension (n-1)×(n-1), as two vertices in G are being fused, producing just one vertex.

$$\mathrm{L}(G/\nu_1\nu_2)=\begin{pmatrix} d_1-a_{21}+d_2-a_{12} & -(a_{13}+a_{23}) & -(a_{14}+a_{24}) & \cdots & -(a_{1n}+a_{2n}) \\ -(a_{31}+a_{32}) & d_3 & -a_{34} & \cdots & -a_{3n} \\ -(a_{41}+a_{42}) & -a_{43} & d_4 & & -a_{4n} \\ \vdots & \vdots & & \ddots & \vdots \\ -(a_{n1}+a_{n2}) & -a_{n3} & -a_{n4} & \cdots & d_n \end{pmatrix}$$

Comparing this matrix to the previous one, we see that the 1,1 entry is the summation of the entries from the 1,1, 1,2, 2,1, and 2,2 entries in $\mathrm{L}(G\text{-}\nu_1\nu_2)$. This is reasonable, as these vertices have been fused the directed edges from ν_1 to ν_2 where removed. Further, for the rest of the first row and first column, we see each entry is the summation of the respective entries from the first two rows and columns frm the previous matrices. Once again, this coincides with the actual graph. The edges that a vertex ν_i sent or received from both ν_1 and ν_2 are joined to create a multiple edge adjacent to the new matrix ν_{12}. Thus, these entries in the first row and column for the matrix $\mathrm{L}(G/\nu_1\nu_2)$ are indeed a summation of such edges. The rest of the matrix remains the same as in the previous two matrices.

Now that we have established the form of these matrices for each specific graph and how they are related, we now must focus the cofactors of each. For notational sake, we will consider the cofactor found upon removing the first row and first column of each respective Laplacian matrix, which will be referred to as k_1. Specifically, each k_1 can be found with the following:

$$k_1(G)=\det\begin{pmatrix} d_2 & -a_{23} & -a_{24} & \cdots & -a_{2n} \\ -a_{32} & d_3 & -a_{34} & \cdots & -a_{3n} \\ -a_{42} & -a_{43} & d_4 & & -a_{4n} \\ \vdots & \vdots & & \ddots & \vdots \\ -a_{n2} & -a_{n3} & -a_{n4} & \cdots & d_n \end{pmatrix} \tag{1}$$

$$k_1(G-v_1v_2)=\det\begin{pmatrix} d_2-a_{12} & -a_{23} & -a_{24} & \cdots & -a_{2n} \\ -a_{32} & d_3 & -a_{34} & \cdots & -a_{3n} \\ -a_{42} & -a_{43} & d_4 & & -a_{4n} \\ \vdots & \vdots & & \ddots & \vdots \\ -a_{n2} & -a_{n3} & -a_{n4} & \cdots & d_n \end{pmatrix} \tag{2}$$

and

$$k_1(G/v_1v_2)=\det\begin{pmatrix} d_3 & -a_{34} & \cdots & -a_{3n} \\ -a_{43} & d_4 & & -a_{4n} \\ \vdots & & \ddots & \vdots \\ -a_{n3} & -a_{n4} & \cdots & d_n \end{pmatrix} \tag{3}.$$

Upon computing these values, we immediately notice a relationship that can be used to simplify the notation. If we call the entire right side of (3) D_0, it can then become one of a series of such determinants in computing the first k_1 value. Following D_0 are $D_1,\ldots,D_n$ where n is taken from the last term needed in finding the determinant. Using the rules for finding determinants from linear algebra verifies this. Notice that the terms in brackets below are exactly identical in the first two equations.

$$k_1(G)=d_2\cdot D_0-\left[\left(-a_{23}\cdot D_1\right)+\left(-a_{24}\cdot D_2\right)-\ldots-\left(-a_{2n}\cdot D_n\right)\right]$$

$$k_1(G-v_1v_2)=\left(d_2-a_{12}\right)\cdot D_0-\left[\left(-a_{23}\cdot D_1\right)+\left(-a_{24}\cdot D_2\right)-\ldots-\left(-a_{2n}\cdot D_n\right)\right]$$

$$k_1(G/v_1v_2)=D_0$$

If we use these values in their respective spots from the initial equation and simplify, we find both sides to be equal. Thus, the following equation will always be true:

$$k_1(G) = k_1(G - v_1v_2) + a_{12}k_1(G / v_1v_2).$$

A similar argument shows that, for any *i*,

$$\det_{i,i} \mathrm{L}(G) = \det_{i,i} \mathrm{L}(G - v_1v_2) + a_{12} \det_{i,i} \mathrm{L}(G / v_1v_2).$$

In establishing this equality, we have completed our proof of the matrix tree theorem. Since this proof was based on induction on the number of edges *m* in the graph G, we were able to find the following equality knowing that the number of spanning trees and cofactors for two graphs with less than m edges were equivalent.

$$\left|t_i(G - v_1v_2)\right| + a_{12}\left|t_i(G / v_1v_2)\right| = \det_{i,i} \mathrm{L}(G - v_1v_2) + a_{12} \det_{i,i} \mathrm{L}(G / v_1v_2).$$

And hence,

$$\left|t_i(G) = \det_{i,i} \mathrm{L}(G)\right|$$

Therefore, the number of spanning trees for a directed Eulerian graph G is indeed equivalent to the cofactor of its Laplacian matrix. ■

Having completed these two proofs and derived formulas from each, we can now state the following: Given a de Bruijn graph G generated from a set of DNA fragments, the probability of correctly assembling the original strand is

$$\frac{1}{t_i(G)\prod_{j=1}^{n}\left(d^+(v_j) - 1\right)!}.$$

I. The Eulerian Superpath Problem

The Eulerian superpath problem (referred to here as the ESP) aims to do the following: Given an Eulerian graph G with a set of paths P in the graph, find an Eulerian circuit in G that contains all components of P as subpaths.

We noted earlier in this paper that the technique of using graph theory for DNA fragment assembly is referred to be some authors as the Eulerian path approach. However, in the terminology we have adopted, a path does not repeat edges or vertices, so it cannot always complete an Eulerian journey through a graph. Thus, for the rest of this paper, this approach really refers to Eulerian trails and circuits, as opposed to paths.

Given error free data and perfect pieces of DNA data, we are able to compose a directed graph representing fragments from the strand of DNA. An Eulerian circuit in this graph completes a full sequence that can be drawn from this data. Finding the total number of Eulerian circuits in such a graph is known as the Eulerian path problem.

However, in cases of erroneous data with repeats and fragments of varying lengths, the typical Eulerian path problem becomes much more difficult. Fragments range from anywhere from 100 to 300 nucleotides in length, and attempting to compose a de Bruijn graph with 20-tuples produces a graph with millions of vertices and edges. Clearly a graph of this nature is not easy to work with.

In two articles by Pevzner, Tang, and Waterman [1-2], the authors detailed a computer package which handles graphs of this kind. This program, known as EULER, takes erroneous and incomplete data, and simplifies it based on a series of cuts and detachments on the graph. The random DNA fragments of various lengths are represented as individual paths found throughout the graph. These transformations on graphs in EULER are the components of what the authors define as the Eulerian superpath problem. Note that the original Eulerian path problem is a case of the ESP, in which each path is represented by one edge.

The approach to solving this problem is based on maintaining equivalence on the graph G and system of paths P from one transformation to the next. More specifically, the resulting graph and paths following the first transformation, represented as (G_1, P_1), corresponds

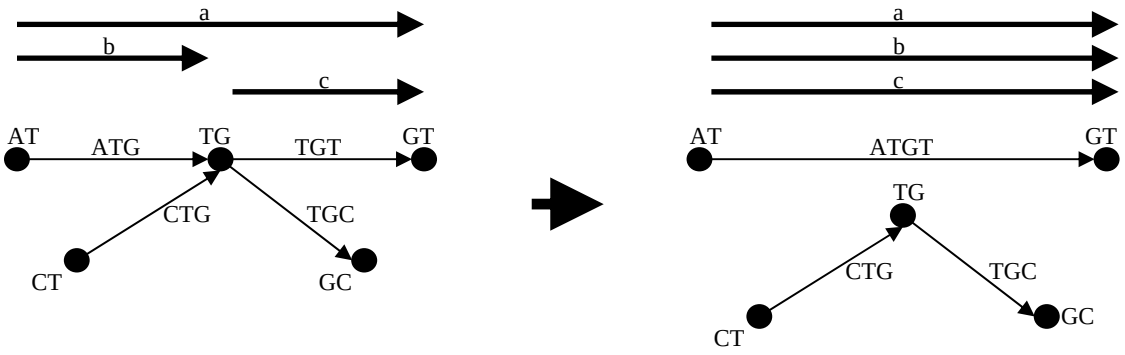

Figure 6. An *x*,*y*-detachment connecting the edges ATG and TGT.

equally with the original set (G, P). Therefore, once the final k-th transformation is completed, the resulting set (G_k, P_k) is a much neater representation of (G, P). It is with this final set that we have transformed the ESP to just an Eulerian path problem, and the Eulerian circuits that we find with (G_k, P_k) match what would be found in (G, P). The first transformation detailed by Pevzner, Tang, and Waterman [1-2] is the *x*,*y*-detachment, and it can be performed in the event that there are no multiple edges in the given graph G. Please note that our notation differs from that of the respective references, as it maintains the notation we have been using throughout this paper. If we have two adjacent edges x_1x_2 and x_2x_3 in G, and three complete sets of paths from P, one which contains both edges consecutively, one which ends with x_1x_2, and one which begins with x_2x_3, then this cut can be sufficiently executed. Specifically, the vertex connecting the two edges, x_2, is pulled away, bringing together a new edge, x_1x_3. This transformation removes an edge from the graph, and makes the vertex v_2 of lesser degree. The new graph is still equivalent to the original, but now computations are more simplified as a result of these transformations.

Figure 6 details this *x,y*-detachment using a portion of a simple de Bruijn graph constructed with DNA data. In the event that we had three complete sets of various fragments, one set which contained the edges ATG and TGT consecutively (a), one set which ended at ATG (b), and one set which began with TGT (c), then the vertex TG would safely be pulled out and the edges ATG and TGT would become a new, longer edge, representing the piece ATGT. Notice how the resulting graph has fewer edges, and the degree of vertex TG has similarly been reduced. Further, now all paths which contained both edges ATG and TGT just contain edge ATGT, the paths which ended at ATG now do so at ATGT, and the paths which began with TGT now do so at ATGT.

The second detachment described concerns a graph G which contains multiple edges, which is referred to as a *x,y*$_1$-detachment. Here, we are given a multiple edge x_1x_2, and the edges x_2x_3 and x_2x_4 which are separately adjacent to and following it. We know that there will be an Eulerian circuit which visits the edge x_1x_2 on exactly two occasions, once moving to x_2x_3, and once to x_2x_4.

Much like the previous detachment, the *x,y*$_1$-detachment similarly depends on complete sets of paths. The first set is all the paths which contain one copy of the multiple edge x_1x_2 followed consecutively by x_2x_3. The second and third sets are those which end at the multiple edge x_1x_2, and those which begin with the edge x_2x_3.

Now this case becomes slightly more complicated, as it is dependent on whether or not there are any paths that satisfy the second set. Specifically, if there are no paths which end on x_1x_2, then we can safely remove one copy of x_1x_2, and create a new edge x_1x_3, which would further reduce the degree of the vertex x_2. However, if there is a set of paths which end on x_1x_2, then this detachment cannot safely be made,

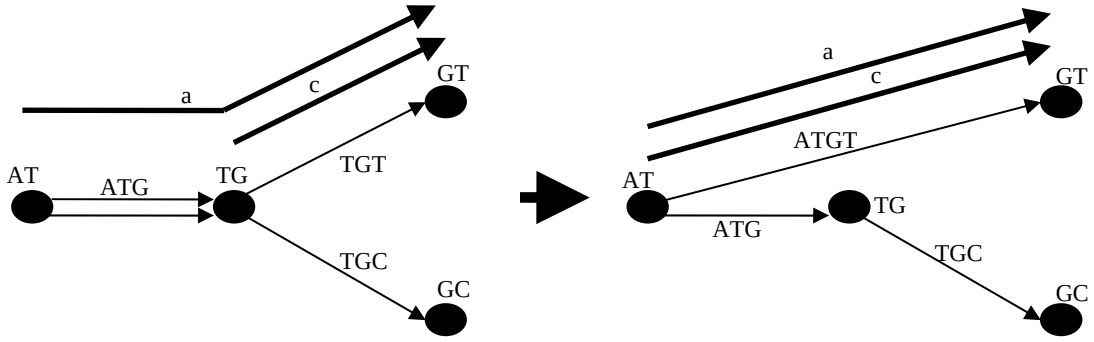

Figure 7. A successful x,y_1-detachment at the multiple edge ATG.

because we cannot assume that the desired Eulerian circuit will move to x_2x_3 or x_2x_4 next.

Once again, we will look at this detachment using a portion of a simplified de Bruijn graph with DNA data in Figure 7. Since there is a multiple edge ATG, we know this segment must repeat at some point in the entire sequence. Here, we have a complete set of DNA fragments which contain one copy of ATG followed consecutively by TGT (a), and such a set which begins at the TGT (c). For the sake of this example we will say there are no such fragments which stop at ATG, hence why there is no depiction of the paths (b) in our figure. Now, since these conditions are met, we can create a new edge ATGT connecting the vertices AT and GT, leaving one copy of ATG. Like the first detachment, this lessens the number of edges in our graph and reduces the degree of the vertex TG. Notice, once again, how all the paths in the set which contained ATG and TGT now contain ATGT, and such paths from the set which began with TGT now begin with ATGT.

The discussion of the ESP continues in the articles by Pevzner, Tang, and Waterman [1-2] with further cases of the problem, including an analysis on consistency between sets of paths. Further, the authors show another transformation on a graph, known as the *x*-cut, which allows us to completely remove an edge from certain sets of paths in our graph.

In the interest of understanding how the ESP works with a complete graph, we will demonstrate an example of detachments on a complete de Bruijn graph constructed with DNA data. Recall the graph in Figure 8, which represents a DNA de Bruijn graph that has two possible Eulerian circuits (given the additional edge which connects the first and last vertices). If we are given this graph, and a series of DNA fragments of varying size, we can use the different fragments to determine which Eulerian circuit represents the actual sequence.

Suppose we have the following fragments: (1) ATGG, (2) TGGCG, (3) GGCGTGG, and (4) CGTGCA. With these fragments, there are two critical single-edged detachments that we can make, and still maintain equality with the original graph. This transformation on the graph is shown in Figure 8.

The first detachment takes place at the vertex TG. Fragment (1) travels from AT to GG through vertex TG, whereas fragments (3) and (4) travel through TG from GT and then move on to GC. Because of this, we can safely create a new edge between the vertices AT and GG, and not disrupt any of the paths in doing so.

The second critical detachment takes place at the vertex GC. Here, both fragment (2) and (3) pass through GC from GG and then to CG. Further, fragment (4) passes through GC after from TG, and concludes at CA. Because of this, we are able to make another detachment which does not affect any of the paths by creating a new edge connecting vertices TG and CA.

Recall that the edges from the original graph represent fragments of length 3. But with these two detachments, the new edges formed, ATGG and TGCA, are of length 4. With these two transformations on our graph, it is now clear that the fragments

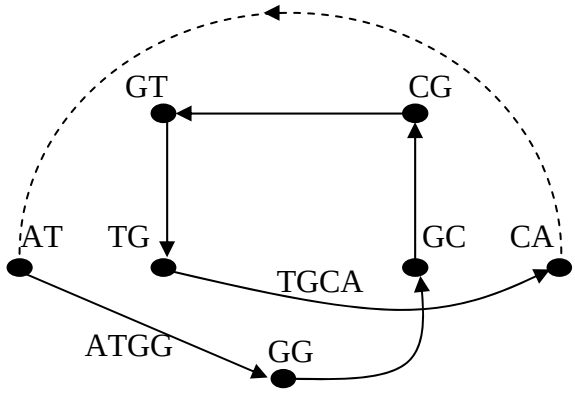

Figure 8. The graph which is produced after two detachments on the graph in Figure 3.

belong to the sequence ATGGCGTGCA, as that is the only Eulerian circuit that remains.

It is also important to recognize that the ESP has been found successful in realistic contexts. Specifically, Pevzner, Tang, and Waterman [1-2] apply the components from ESP and their EULER program and put it towards a project involving the sequencing of the *Neisseria meningitidis* [*NM*] genome, referred to as the NM project. Throughout the course of these two articles and further in Pevzner and Tang [17], the authors detail the impact that these concepts have when dealing with a bacterial genome that is rather difficult to reconstruct.

The *NM* genome, successfully reconstructed without errors, is represented in a de Bruijn graph that has 4,039,248 vertices, where each vertex is of length 20. However, construction of a graph based on the real provided DNA fragments originally contains 9,474,411 vertices. Following the error-correction procedure on repetitive and faulty data detailed by Pevzner, Tang, and Waterman [1-2], this number is reduced significantly to 4,081,857 vertices. Thus, there are still more than forty thousand vertices that separate the graph with experimental data from the graph representing the actual genome.

Using the concepts behind the ESP, the experimental graph of the *NM* genome is able to be simplified. Specifically, 18,026 of the 18,962 edges of length 21 in this graph are considered resolvable based on the detachments and decompositions as shown in the paper. The resulting de Bruijn graph has 126 repeats, of which further improvements can be implemented as shown by Pevzner and Tang [17]. Here the authors introduce the concepts of dealing with "double-barreled data" and a new algorithm, EULER-DB, which reduces this number of repeats significantly, also based on similar ideas.

In more recent years, other authors have continued to gain a better understanding of the ESP and how it can further be used. Zheng, Shi, and Zhou [18] used the concepts of the problem which enabled them to create an algorithm which would store all data found in a particular DNA de Bruijn graph. This proves to be crucial because these graphs are very large, and as a result are both time and memory consuming.

Additionally, as of last year, Zheng, Leong, and Tang [19] addressed the "spectral alignment problem" which was similarly developed by Pevzner, Tang, and Waterman [1-2] along with EULER and the ESP. Here, the authors propose an algorithm regarding coverage at the position on various fragments of DNA, which they refer to as the "coverage sensitive alignment" algorithm.

The Eulerian superpath approach on DNA fragment assembly doesn't necessarily eliminate all the discrepancies about the original construction of the genome, but just makes it easier to work with. With this approach and the EULER package, scientists and researchers are able to consider large groups of edges,

vertices, and paths as a significantly smaller number of elements, which enables easier computation of the Eulerian circuits in the graphs.

II. DNA Sequencing with Nanopores

In more recent years, concepts in graph theory have been used to aid other problems which occur in the field of bioinformatics. One of which can be found in Bokhari and Sauer [3]. Here, the authors use de Bruijn graphs representing DNA data and the Eulerian path approach upon determining an algorithm which aids the process of DNA sequencing with nanopores. Nanopore devices are used in DNA sequencing with the hopes of enabling large and complete strands of DNA to be read completely. These nanopores are a few atoms in diameter, and single stranded DNA passes through them by optical or electrical means. Through this passage, each individual base of the strand is identified, producing a seemingly endless chain of A, T, G, and C.

Bokhari and Sauer [3] identify and address the problems that occur in this form of sequencing, which are much similarly to those that we saw previous. The first issue lies on the fact that an entire single strand rarely makes it through the nanopore in one piece. Rather, substrings of lengths around 100,000 bases in length are produced. Further, as discussed earlier, the problem of the coinciding Watson-Crick complement causes confusion regarding whether or not the original strand or its coinciding complement is being passed through the nanopore. Similarly, there another issue involving the specified orientation of single strand as it passes through. Upon passage and decomposition into substrings, the direction (3'—5' or 5'—3') of the DNA is lost. The final problem is that of erroneous data. Not only may the DNA sequence itself be riddled with insertions, deletions, mutations, and repeats, but the nanopores themselves may also produce errors amidst the passing of a strand.

In formulating an algorithm to eliminate the discrepancies that coincide with these issues, the authors used the Eulerian path approach and provided new variations which allowed it to still be used successfully with long and erroneous strands of DNA. They addressed the problem of complements and orientation by considering all four variations of a substring as four different paths, all of which being contained in the graph.

Specifically, their de Bruijn DNA graph G must contain the paths $P_1,\ldots,P_4$ such that the union of all four paths is equivalent to G, denoted $P_1 \cup P_2 \cup P_3 \cup P_4 = G$. Further, no two paths have any equivalence in their intersections, as $P_i \cap P_j = \emptyset$ for all i, j, where $i \neq j$. This property was referred to by Bokhari and Sauer [3] as E^4, and the algorithm contains a function $E^4(G)$ that is agreed upon if the DNA graph G maintains these identities. Testing these paths with a permutation further determines whether or not that are, in fact, complements and reversals of each other. If this holds true, then the strand being examined comes from an authentic sequence of DNA data.

In the cases of repetitive data, it must first be determined whether or not the longest repeat in a set of DNA data is longer than the available substrings. If this is the case, then the algorithm is unable to produce a graph matching the E^4 identity. Also, in the case of the Eulerian path approach, repeats are defined to be sequences that match each other exactly. Thus, two long substrings which are 99% identical are not considered to be repeats upon constructing the de Bruijn graph, and the Eulerian path approach still functions properly with the provided data.

This graph decomposition algorithm was tested by Bokhari and Sauer [3] on the Mycoplasma pneumoniae genome, which is 816,394 bases in length. Upon sequencing with a nanopore device, this algorithm was able to produce the full sequence of this bacterium in less than an hour.

III. Conclusions

Various concepts from graph theory are used for applications in many different fields of study. In this paper, we demonstrated how de Bruijn graphs and Eulerian circuits are used to resolve some problems in DNA sequencing and fragment assembly. Further, it was the formula derived from both the BEST and matrix tree theorems that allowed us to arrive at a

constant formula which can be used to count the number of Eulerian circuits in a de Bruijn graph which has been constructed from DNA data. Such circuits dictate the probability of constructing the desired original strand of DNA based on a collection of fragments from that strand.

The progress being made with the Eulerian superpath problem and EULER package shows that scientists and researchers are continuing to find ways to resolve problems that arise in fragment assembly, especially those dealing with erroneous data sets. Hopefully, these transformations on directed Eulerian graphs represent the first of many steps which can be taken to simplify the extraordinarily complex world of DNA sequencing, as more of the like will come with time. Further, it is imperative to recognize that the field of bioinformatics is changing on a daily basis, as seen in the previous section. Much of this development is triggered with new advancements in technology and intelligence. In addition to the sources discussed in this paper, the reader is encouraged to look at other aspects of bioinformatics in which this approach has been involved. This includes, but is certainly not limited to, the following articles: Alekseyev and Pevzner [20], Myers [21], and Raphael, Zhi, Tang, and Pevzner [22].

Progress that has been achieved in this field through the use of graph theoretical concepts simply opens the door for more opportunities. Ultimately, problems with seemingly never-ending lines of partially corrupted DNA data can be simplified into various cases of a graph and paths contained within that graph. It is with these tools that the more complicated cases that may arrive can be approached with the hopes of more clarity and success.

ACKNOWLEDGEMENTS

Support made possible by the Vermont Genetics Network through Grant Number P20 RR16462 from the INBRE Program of the National Center for Research Resources (NCRR), a component of the National Institutes of Health (NIH) is acknowledged. The contents of this research report are solely the responsibility of the author and do not necessarily represent the official views of NCRR or NIH. Support for this research was also made possible through the National Security Agency.

REFERENCES

1. Pevzner, P.A.; Tang, H.; Waterman, M.S. “An Eulerian Path Approach to DNA Fragment Assembly,” *Proceedings of the National Academy of Sciences*. 98 (2001), no. 17, 9748-9753.
2. Pevzner, P.A.; Tang, H.; Waterman, M.S. “A New Approach to Fragment Assembly in DNA Sequencing,” *RECOMB 01*. (2001), 256-267.
3. Bokhari, S.H.; Sauer, J.R. “A Parallel Graph Decomposition Algorithm for DNA Sequencing with Nanopores,” *Bioinformatics*. 21 (2005), no. 7, 889-896.
4. Tucker, A. *Applied Combinatorics*. 5th Edition. (John Wiley & Sons, Inc., Danvers, Massachusetts, 2007).
5. West, D. *Introduction to Graph Theory*. 2nd Edition. (Prentice Hall, Upper Saddle River, New Jersey, 2001).
6. de Bruijn, N.G. “A Combinatorial Problem,” *Koninklijke Nederlandse Akademie V. Wetenschappen*. 49 (1946), 758-764.
7. Jones, N.C.; Pevzner, P.A. *An Introduction to Bioinformatics Algorithms*. (The MIT Press, Cambridge, Massachusetts, 2004).
8. Watson, J.D.; Crick, F.H.C. “Molecular Structure of Nucleic Acids,” *American Journal of Psychiatry*. 160 (2003), 623-624
9. Pevzner, P.A. *Computational Molecular Biology: An Algorithmic Approach*. (MIT Press, Cambridge, Massachusetts, 2000).
10. Idury, R.M.; Waterman, M.S. “A New Algorithm for DNA Sequence Assembly,” *Journal of Computation Biology*. 2 (1995), no. 2, 291-306.
11. Pevzner, P.A. “DNA Physical Mapping and Alternating Eulerian Cycles in Colored Graphs,” *Algorithmica*. 13 (1995), no. 1-2, 77-105.
12. van Aardenne Ehrenfest, T.; de Bruijn, N.G. "Circuits and Trees in Oriented Linear Graphs,” *Simon Stevin*. 28 (1951). 203-217.

13. Tutte, W.T.; Smith, C.A.B. “On Unicursal Paths in a Network of Degree 4,” *Amer. Math. Monthly*. 48 (1941). 233-237.
14. Bollobás, B. *Graduate Texts in Mathematics: Modern Graph Theory*. (Springer-Verlag, New York, NY, 1998).
15. Fleischner, H. “Eulerian Graphs and Related Topics,” Part 1, Volume 1. *Annals of Discrete Mathematcs*. 45 (1990).
16. Fleischner, H. “Eulerian Graphs and Related Topics” Part 1, Volume 2. *Annals of Discrete Mathematcs*. 50 (1990).
17. Pevzner, P.A.; Tang, H. "Fragment Assembly with Double-barreled Data," *Bioinformatics*. 17 (2001), S225-S233.
18. Zheng, W.; Shi W.; Zhou W. “A Parallel DNA Fragment Assembly Algorithm Based on Eulerian Superpath,” *Online Journal of Bioinformatics*. 5 (2004), 91-101.
19. Zheng, J.; Leong, H.W.; Tang, H. “An Improved Algorithm for Error Correction of Reads in DNA Fragment Assembly,” *RECOMB 07*. 2007.
20. Alekseyev, M.A; Pevzner, P.A. “Colored de Bruijn graphs and the Genome Halving Problem,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. 4 (2007), no. 1, 98-107.
21. Myers, E.W. “The Fragment Assembly String Graph,” *Bioinformatics 2005*. 21 (2005), Supplement 2, ii79-ii85.
22. Raphael, B.; Zhi, D.; Tang, H.; Pevzner, P.A. “A Novel Method for Multiple Alignment of Sequences with Repeated and Shuffled Elements,” *Genome Research*. 14 (2004), 2336-2346.