2018

Self-Assembly of DNA Graphs and Postman Tours

Katie Bakewell
University of North Florida, n00184316@ospreys.unf.edu

Self-Assembly of DNA Graphs and Postman Tours

by

Katie Bakewell

A thesis submitted in partial fulfillment
of the requirements for the degree of
Masters of Science
Department of Mathematics and Statistics
College of Arts and Sciences
University of North Florida

Date of Approval:
August 10, 2018

Keywords: Thickened graphs, Boundary components, Windy Postman Problem,
Reporter strands, DNA computing

Table of Contents

Abstract

DNA graph structures can self-assemble from branched junction molecules to yield solutions to computational problems. Self-assembly of graphs have previously been shown to give polynomial time solutions to hard computational problems such as 3-SAT and k-colorability problems. Jonoska et al. [13] have proposed studying self-assembly of graphs topologically, considering the boundary components of their thickened graphs, which allows for reading the solutions to computational problems through reporter strands. We discuss weighting algorithms and consider applications of self-assembly of graphs and the boundary components of their thickened graphs to problems involving minimal weight Eulerian walks such as the Chinese Postman Problem and the Windy Postman Problem.

Chapter 1

Introduction

In 1994, Adleman showed in [1] that a polynomial time algorithm can identify solutions to the Hamiltonian Path Problem using solely DNA strands and their Watson-Crick self-assembly properties, and thus, he initiated DNA Computing. Since then, DNA computing and nano-technology have been of interest in solving many, computationally hard, mathematical problems. One such method is the self-assembly of DNA graphs. The 3-colorability and 3-SAT problems have been shown, e.g. in [16, 10, 11, 26], to give solutions in polynomial number of steps using DNA self-assembly algorithms, thus, showing that any other nondeterminstic polynomial problem could be solved in polynomial time using encoded DNA molecules. Since a self-assembled DNA graph represents each edge of the graph by a double-stranded DNA molecule, which in essence, traverses an edge twice in opposite directions, Jonoska et al. [13] proposed to study such self-assembled graphs topologically, as thickened graphs, where each of the two strands representing an edge can be viewed as part of a boundary component of the underlying topological space. In [13], the authors showed that for every connected multi-graph, there exists a boundary component which traverses every edge of the graph at least once and not more than twice.

Postman tour problems relate to the problem of identifying a minimal Eulerian walk over a graph given certain conditions. The most general case, the Chinese Postman Problem, is solvable in polynomial time. More specific cases, such as the

Windy Postman Problem, are nondeterministic polynomial. The Windy Postman problem, discussed here, is NP- Complete, with current optimal solutions using cutting-plane algorithms.

In this work, the relationship between such boundary components and postman tours is considered. Additionally, the possibility of self-assembled solutions to the family of postman tour problems through DNA self-assembly is discussed.

This paper is organized as follows. Chapter 2 is a review of the graph theoretic and topological notions used in this work. An overview of self-assembly of DNA graphs, along with two computational algorithms using polynomial number of steps to find solutions to NP-complete problems in laboratory testing, is provided in Chapter 3. Additionally, the concept of embedding weights within a DNA-based graph is discussed there, as well.

Chapter 4 introduces thickened graphs which are manifolds that contain a graph as a deformation retract. The properties of thickened graphs and some topological analysis of the surfaces is provided, followed by the construction of additional thickened graphs through elementary operations on the boundary components. The relationship between thickened graphs and DNA self-assembly is explored.

An algorithm reducing any connected graph to a 3-valent graph is provided in Chapter 5, along with an example of the algorithm for K_5 given in detail. The topological "lift" of this 3-valent graph is also discussed. Subsequently, in Chapter 6, the proof for the existence of reporter strands for 3-valent connected graphs is discussed in detail.

Finally, Chapter 7 considers the relationship between postman tours and reporter strands. A theorem discussing the existance of a reporter strand, which contains a given postman tour is proven, followed by an algorithm to identify the embedded postman tour as a subgraph. This chapter also presents a language theoretic approach to removing edges from reporter strands to obtain postman tours. Concluding remarks and a discussion of future work are presented in Chapter 8.

Chapter 2

Graph Theoretic and Topological Background

In this chapter, we recall the definitions and basic properties from graph theory and topology that are relevant to the notions we discuss. We define and review the graph theoretic terminology used, including paths and cycles, digraphs, and graph homomorphisms. In the second section, postman tour problems are reviewed. The topology section recalls manifolds and their properties, invariants such as the Betti Numbers, lifting and covering maps, and finally, homologies and deformation retracts.

2.1 Graph Theoretic Notions

We begin this section reviewing the notions of undirected graphs and paths and cycles within undirected graphs and conclude with a review of directed graphs. We follow the definitions as stated in [5].

A *graph* G is composed of a *set of vertices*, V, and a *set of edges*, E. An *edge* $\{u, v\}$ is an unordered pair of vertices $u, v \in V$. An edge that is joined to a vertex is said to be *incident* with that vertex. Two vertices that are joined by an edge are called *adjacent*. The *degree of a vertex* $v \in V$, denoted by $deg(v)$ is the number of edges incident with that vertex. A vertex is called *odd* (resp. *even*), if its degree is an odd (resp. even) number. We sometimes use *valency* to mean degree. A corollary to the Fundamental Theorem of Graph Theory states that the number of odd vertices in a graph must be even, see [5].

For an integer k, a *k-valent* or a *k-regular* graph has only vertices of degree k. The *order* of a graph is the number of its vertices, $|V|$. The *size* of a graph is the number of edges, denoted by $|E|$. For a k-valent graph of order n, the number of edges is $|E| = \frac{kn}{2}$.

Within a graph, a sequence beginning with a vertex, followed by alternating edges and vertices and ending with a vertex is called a *walk*. If the initial and final vertex of a walk are the same vertex, the walk is called *closed*. A walk with no repeated edges is called a *trail*. Note that vertices in a trail may repeat. A trail with no repeated vertices is called a *path*. A closed path (with the same initial and terminal vertex) is called a *cycle*.

A graph is said to be *Eulerian* if there exists a closed trail, called an *Eulerian cycle*, which contains every edge of the graph. It can be shown (see [5]) that having only vertices of even degree is a necessary and sufficient condition for a graph to be Eulerian. Furthermore, a graph can have an *Eulerian trail*, also called an *Eulerian walk* or an *Eulerian path*, that is a trail containing each edge of the graph, but not a cycle, if and only if the graph has exactly two vertices of odd degree. To relate this to the previous result, one can create an edge between the two vertices of odd degree, which will result in a closed trail and all vertices would be of even degree.

Since a graph that is not Eulerian has an even (positive) number of odd vertices, one can construct an Eulerian graph from it by adding edges that join two odd vertices until all odd vertices have become even. Thus, the minimum number of edges that need to be added is half the number of odd vertices. Another way to construct Eulerian graphs from non-Eulerian is to take a closed walk which contains every edge and create a new parallel edge anytime an edge is repeated. This way, an additional edge represents traversing the edge twice. An example of this method

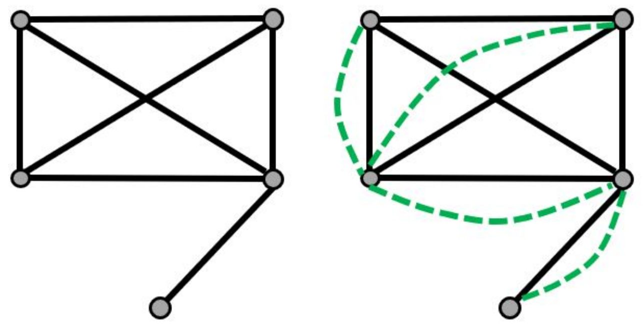

Figure 2.1: An Eulerian graph is constructed by adding the green dashed edges.

is shown in Figure 2.1 with the green dashed edges representing the added edges.

Weights can be assigned to the edges of a graph. These weights allow for the calculation of an aggregate weight for cycles. Weighted graphs allow for many interesting optimization problems such as the Traveling Salesman Problem and the family of postman tours problems discussed in this paper.

For graphs $G = (V, E)$ and $G' = (V', E')$, a *graph homomorphism* $\phi : G \to G'$ is a mapping from V to V' such that for every adjacent pair of vertices in G there is an adjacent pair of vertices in G'. That is, if $\{u, v\} \in E$, then $\{f(u), f(v)\} \in E'$. If there exists a graph homomorphism between G and G', then G' is said to be a *homomorphic image of* G.

Throughout this paper, we consider multiple successive graph homomorphisms on a single graph. Such composition of graph homomorphisms is itself a graph homomorphism.

Edges in graphs can be undirected or directed. When edges are undirected we denote them as unordered pairs of vertices, e.g. $\{u, v\}$, while directed edges (arcs) are denoted as ordered pairs of vertices, e.g. (u, v). So far, we have discussed graphs containing undirected edges only. Such graphs are called *undirected graphs*, whereas a graph containing only directed edges is a called a *directed graph* or a *digraph.* A graph can contain both directed and undirected edges, in which case it is called a *mixed graph.* Undirected graphs which have at most one edge joining each pair of vertices are called *simple.* When parallel edges between the same pair of vertcies are allowed, the graph is called a *multigraph*, and when, in addition, loops are allowed, the graph is called a *general graph.* The notion of a multigraph can be extended to directed graphs and a directed multigraph can contain self-loops. The graphs considered throughout this paper are directed graphs, with at most two arcs joining any two vertices. If two directed edges join the same pair of vertices, then they have opposite orientation. One way to transform an undirected graph into directed is to replace each undirected edge $\{u, v\}$ by two directed edges in opposite direction (u, v) and (v, u). A digraph is Eulerian if the *indegree* (the number of edges going to a vertex) and the *outdegree* (the number of edges going out of a vertex) are equal. Thus, a digraph obtained from an undirected graph by replacing each edge with two oppositely oriented arcs is necessarily Eulerian.

Graphs in this paper are denoted by $G = (V, E, t)$ where V is the set of vertices in G, E is the set of edges of G, and $t : E \to W$, where W is a set of one and two element subsets of V defining the endpoints of the edges. If $v \in V$ and $e \in E$, we say that $v \in t(e)$ if and only if e is incident with v. Note that multiedges and loops are allowed in G. If $t(e) = t(e') = \{u, v\}$ (respectively, if $t(e) = t(e') = (u, v)$), we say that e and e' are undirected (resp. directed) *parallel* edges. For digraphs, the set W is called *the set of adjacent vertex ordered pairs in* G. An adjacent vertex ordered pair defines a pair of vertices (u, v) which has a directed edge going out of u and into v. In the case of a loop at vertex v, the ordered pair (v, v) is included in $t(G) = t(E) = W$. Since we are interested in representing undirected edges by directed edges, in the case of an undirected edge $\{u, v\}$, two vertex pairs (u, v) and (v, u) are included in $t(G)$, representing the traversal of $\{u, v\}$ in either direction.

2.2 Postman Tours and Associated Problems

A *postman tour* τ is a closed walk that contains every edge of a graph, or else, that traverses each edge at least once. This tour is named due to the imagery of a postman having to walk down every street at least once without regard to whether he must occasionally duplicate a street. Thus, there is an inherent connection between postman tours and Eulerian graphs, and also, between postman tours and constructions that transform a graph into an Eulerian one.

Two optimization problems arise from postman tours: identifying the minimal length closed walk that contains each edge and identifying the minimal weight closed walk that traverses each edge. When weights are included in the identification of postman tours, then the family of minimal weight Eulerian walk problems, often called *Postman* or *Route Inspection Problems*, arises. The most basic problem involves the identification of a minimal weight postman tour in a connected graph. This problem is often referred to as the *Chinese Postman Problem (CPP)*, named in honor of Chinese mathematician Guan Meigu (also known as Mei-Ko Kwan through phonetic spelling), who proposed it in 1962, see [22].

Postman problems have a wide variety of real world applications in mathematical programming and operations research. Obviously, route inspections and postal routes can be planned this way as can other city services such as street sweeping and garbage collection. In addition, applications in flow and network management increase the importance of solving this class of problems.

Note that every Eulerian cycle is a postman tour. If a graph is Eulerian, then a minimal length postman tour is one of the Eulerian cycles contained within G. Thus, the interest in solving postman tour problems typically focuses on graphs whose vertices are not all of even degree. Also note that every graph can be made Eulerian by duplicating each of its edges. Thus, we consider only postman tours which visit each edge at least once and no more than twice.

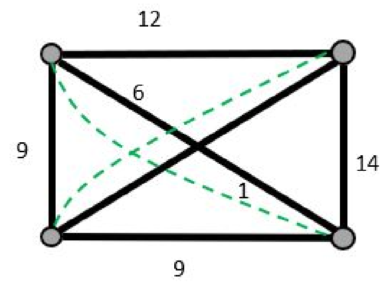

Figure 2.2: A solution to the CPP for G using additional edges in dashed green lines.

Many polynomial time algorithms exist for the CPP. One such algorithm is based on constructing an Eulerian graph from a non-Eulerian graph by joining pairs of odd vertices. All possible pairings of odd vertices are created and their respective weights are calculated. The weight of edges between sets of adjacent vertex pairs are considered, providing the pairing creates an Eulerian graph. The minimal weight combination(s) of pairing edges is(are) identified. Those edges are then added to the graph as parallel edges, and then an Eulearian path is identified. An example of such a solution is given in Figure 2.2, where the two edges traversed twice (dotted in green), provide the minimal edge weight duplication such that all vertices are even.

Further refinements of this process can improve computational time. Given l as the length of a postman tour, we can define $p(G) = l - |E(G)|$ to be the number of edges that are traversed more than once in order to construct a minimal length postman tour. Bounds on $p(G)$ for certain type of graphs were found due to postamn tours constructed by Kostochka and Tulai [14].

Definition 2.2.1 The *robustness* of a graph G relates to its ability to withstand damage. An *m-robust* graph will remain connected with any m edges removed from the graph.

If G is a $(2r+1)$-regular $2t$-edge-connected m-robust multigraph, where m is odd, then

$$p(G) \leq \frac{n}{2} + \max(0, \frac{n-2m}{(r+t)/(r-t)(m+1)}).$$

We consider 3-regular graphs, where $r = 1$ which gives that

$$p(G) \leq \frac{n}{2} + \max(0, \frac{n-2m}{(1+t)/(1-t)(m+1)}.$$

Suil O and Douglas West, see [20], further refined sharp bounds for $p(G)$ in 3-regular multigraphs. Additionally, they conjecture that similar construction could be applied to $(2r+1)$-regular graphs.

There are many variants of postman problems that are non-polynomial due to the computational complexity of calculating all possible postman tours. The *Windy Postman Problem (WPP)* requires identification of the minimal weight postman tour on a directed weighted graph. After introducing the Chinese Postman Problem, Meigo Guan proposed this WPP generalization in 1984 and proved that the WPP is NP-complete, see [6].

It has been proven that the *Mixed Postman Problem (MPP)*, requiring finding a minimal weight postman tour on a mixed graph, is NP-hard. The theorem below shows that the WPP can be reduced to the MPP, and vice versa. This shows that the Windy Postman Problem is both NP-complete and NP-hard, see [6].

Theorem 2.2.2 *WPP is NP-hard.*

Proof. In order to show that the Mixed Postman Problem can be reduced to the Windy Postman Problem, let G be a mixed graph, with some edges being directed and some edges being undirected. Construct G' by adding a parallel edge (v_j, v_i) for each edge $e = (v_i, v_j)$, oriented in the opposite direction. Assign an arbitrarily high weight for these edges, M. Consider the undirected edges as two parallel equal weight edges, one in each direction. For each edge in G, this newly constructed G' is a symmetric graph.

Any postman tour in G' that uses only the original edges will also exist in G, with the same weight. Identifying postman tours in G' including newly added edges will not optimize the solution, as long as M is sufficiently high, and thus the solution of the Windy Postman Problem will solve the Mixed Postman Problem.

As it has been shown that the Mixed Postman Problem is NP-hard, the Windy Postman Problem, in turn, must also be NP-Hard.

$\blacksquare$

As the Windy Postman Problem is NP-hard and can be reduced efficiently to another NP-hard problem, we can conclude the Windy Postman Problem (and thus the MPP) is NP-Complete.

Additional variants of these problems include the Rural Postman Problem and the Street Sweeper Problem. The *Rural Postman Problem (RPP)* requires identification of the minimal weight postman tour on a graph where some edges may not be required, though may be necessary to optimize the solution. The *Street Sweeper Problem (SSP)* requires identification of the minimal weight postman tour on a graph where some edges may only exist in one direction. Combined problems are also considered, such as the Windy Rural Postman Problem.

Here we focus specifically on the Windy Postman Problem. For Eulerian graphs, the Windy Postman Problem has been shown to be solvable in polynomial time, see [25]. In that paper, Win showed an algorithm, generalizing to all graphs, which requires at most twice the optimal computational time.

Many algorithms have been explored to improve the time needed to solve these complex problems. Branch and Cut algorithms have shown success in reducing the computational complexity of nondeterministic polynomial postman problems including the WPP. These approaches rely on the construction of subsets of edges, with vertices falling within subsets of $V(G)$, and the use of linear programming to optimize solutions within subsets of the graph. Corberan et al., in [3], developed a polynomial time solution to identify violated subsets and a heuristic for the combination of cut pieces to develop a solution to the problem. In 2012, Corberan tested his WPP solution against published methods on 36 graphs with orders ranging from 52 to 264 and size ranging from 78 to 489, with the cut and branch algorithm being the most efficient in 31 of the 36 instances.

2.3 Topological Background

The construction and application of thickened graphs in this paper depends on the topological properties of these graphs. This section includes a brief review of the notions and properties necessary for the discussion of thickened graphs. First, we recall some definitions and properties of manifolds (specifically surfaces), then we introduce Betti numbers, homotopy, and homeomorphisms. Finally, we discuss retractions and lifting. Topological definitions and notations here are from Munkres' *Topology*, see [18].

Definition 2.3.1 A topology on a set X is a collection $\mathcal{T}$ of subsets of X having the following properties:

1. The empty set $\varnothing$ and X are in $\mathcal{T}$.
2. The union of the elements of any subcollection of $\mathcal{T}$ is in $\mathcal{T}$.
3. The intersection of the elements of any finite subcollection of $\mathcal{T}$ is in $\mathcal{T}$.

A set X for which a topology $\mathcal{T}$ has been specified is called a *topological space.* If X is a topological space with topology $\mathcal{T}$, a subset U of X is called an *open set* *of* X, if U belongs to the collection $\mathcal{T}$. A subset A of a topological space X is said to be *closed* if $X - A$ is open. The following theorem outlines basic properties of topological spaces.

Theorem 2.3.2 *Let X be a topological space. Then the following conditions hold:*

1. *The empty set $\varnothing$ and X are closed.*
2. *Arbitrary intersections of closed sets are closed.*
3. *Finite unions of closed sets are closed.*

Proof.

1. Consider X. The compliment of X is the empty set $\varnothing$, which is vacuously closed. Similarly, $\varnothing$ has the open set X as a complement, and thus, $\varnothing$ must also be closed. Hence, since the open sets X and $\varnothing$ are complements of each other, they are also closed.

2. Let $\{A_\alpha\}_{\alpha \in J}$ be a collection of closed sets A_α. Then

$$X - \bigcap_{\alpha \in J} (A_a) = \bigcup_{\alpha \in J} (X - A_\alpha)$$

by DeMorgan's Law. Note that

$$X - A_\alpha = X \cap A_\alpha^c$$

and by definition A_α^c is open, as it is the complement of a closed set. Thus, $X - A_\alpha$ is open for all $\alpha \in J$. Since an arbitrary union of open sets is open, this gives that $\bigcap_{\alpha \in J} A_\alpha$ is closed, and thus arbitrary intersections of closed sets are closed.

3. Let A_i be a closed set for $i \in [1, n]$. Then again, using DeMorgan's Law

$$X - \bigcup_{i=1}^{n}(A_i) = \bigcap_{i=1}^{n}(X - A_i).$$

   In a similar reasoning as above, all $X - A_i$ sets are open. Since finite intersections of open sets are open, the intersection of the n sets $X - A_i$ must also be open which gives that $\bigcup A_i$ is a closed set.

■

Due to the above definition, there exist topologies where one-point sets are not closed. However, these topologies are rarely studied, as they behave strangely and present issues in proving generalized theorems. Felix Hausdorff constructed a definition that removes these examples to ensure that the spaces considered in a proof do not have such examples.

Definition 2.3.3 A topological space X is called a *Hausdorff space* if for each pair x_1, x_2 of distinct points of X, there exist neighborhoods U_1 and U_2 of x_1 and x_2, respectively, that are disjoint.

Any subset of a Hausdorff space also meets the criteria of a Hausdorff space. Some notions of topological (in this case Hausdorff) spaces are helpful in discussing the structures considered in this paper.

Definition 2.3.4 A collection $\mathcal{A}$ of subsets of a space X is said to *cover* X, or to be a *covering of* X, if the union of elements of $\mathcal{A}$ is equal to X.

If all of the elements of a cover of X are open subsets of X, then this is called an *open covering.* Based on the properties of an open covering of X, the space may be considered compact. An example of a cover and open covering of a compact space X is shown in Figure 2.3.

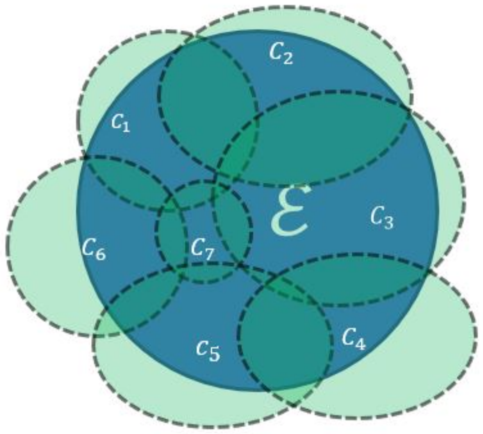

Figure 2.3: A cover and an open covering of a space $\mathcal{E}$.

Definition 2.3.5 A space X is said to be *compact* if every open covering $\mathcal{A}$ of X contains a finite subcollection that also covers X.

One useful property of reducing the set of topological spaces considered to only those meeting the Hausdorff condition is that all compact Hausdorff spaces are closed. This allows the properties of closed spaces to apply to all the spaces considered.

In this paper, we consider a subset of Hausdorff spaces, called manifolds. We first review the concept of homeomorphism and then introduce the notion of a manifold.

Definition 2.3.6 Let X and Y be topological spaces and let $f : X \to Y$ be a bijection. If both the function f and the inverse function

$$f^{-1} : Y \to X$$

are continuous, then f is called a *homeomorphism*.

Manifolds are a useful type of topological spaces. They are extensively studied in differential geometry and algebraic topology. For each point in the space, the local neighborhood is homeomorphic to an Euclidean space. Manifolds can be beneficial as the properties of Euclidean spaces can be used locally, as opposed to a more complex structure. The dimensionality of that space can be identified using the more specific $m-$manifold.

Definition 2.3.7 Given a set X, a *basis* for a topology on X is a collection $\mathcal{B}$ of subsets of X (called *basis elements*) such that

1. For each $x \in X$, there is at least one basis element B containing x.
2. If x belongs to the intersection of two basis elements B_1 and B_2, then there is a basis element B_3 containig x such that $B_3 \subset B_1 \cap B_2$.

Definition 2.3.8 An *m-manifold* is a Hausdorff space X with a countable basis such that each point x of X has a neighborhood that is homeomorphic with an open subset of $\mathbb{R}^m$. A $1-manifold$ is called a curve, while a $2-manifold$ is called a surface.

Theorem 2.3.9 *If X is a compact m-manifold, then X can be embedded in $\mathbb{R}^N$ for some positive integer N.*

That is, for a compact surface S, S can be embedded in a finite dimensional Euclidean space. In this paper, we will discuss graphs as surfaces, and focus on analyzing the points at the "boundaries" of the surface.

Let E be a subspace of a space X. Consider a point $x \in X$. If every open ball centered at x has elements from the set E and from the set $X - E$, then x is called a *boundary point* of E. A continuous set of boundary points on a space is a *boundary component* of E, see [23]. The set of all boundary components of E is denoted $\delta(E)$.

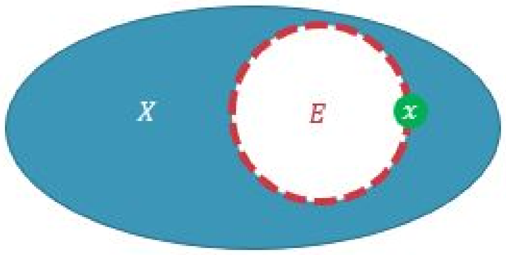

Figure 2.4: A boundary point of E is represented by the green point x and the red dashed curve is a boundary component of E, an element in $\delta(E)$.

Topological invariants are essential to the study of topological spaces. *Invariants* are properties of the space that are preserved under homeomorphisms. One such set of invariants, the Betti Numbers, are particularly useful in the application discussed here. The maximum number of cuts that can be made to a surface without dividing it into two components is called the *Betti Number* of a surface. In other words, the k^{th} Betti Number, β_k, of a surface represents the number of k-dimensional holes or voids in the surface. Thus, when considering the topology of graphs, the zeroth Betti number is the number of connected components in the graph.

Lemma 2.3.10 *Given G has n vertices, m edges, and k connected components, we can calculate the first Betti number of G as*

$$\beta_1(G) = m - n + k$$

Proof.

We proceed by mathematical induction on the number of edges m.

Consider a graph G of n vertices, with no edges. G has n connected components, each consisting of one isolated vertex connected to itself by a trivial path. G should, therefore, have a Betti number of 0, as no cuts are required to separate the the components. Thus, for this graph,

$$\beta_1(G) = 0 - n + n = 0$$

.

Assume that for $m = j$, G has k components, $\beta_1(G) = j - n + k$. By adding an additional edge e to G, one of two possible cases occurs.

Case 1. Let C be a single component within the graph G. Then $G = C$. Adding an additional edge to G will join two vertices within C, but the number of connected components in G will remain k.

Case 2. If G contains two or more components consider C_1 and C_2 to be two compenents. Adding the edge e connecting C_1 and C_2, decreases the number of connected components contained in G by one. Although G has one fewer connected component, the maximum number of cuts to separate the graph into two pieces will remain the same, that is $\beta_1(G + e) = \beta_1(G)$. This follows from $\beta_1(G + e) = j + 1 - n + k - 1 = j - n + k = \beta_1(G)$ as anticipated.

■

We can generalize this lemma to state that for a graph consisting of a single connected component, the first Betti number is the difference between the number of edges and the number of vertices plus one.

Another topological invariant is the genus. The *genus* of a surface is the number of "handles", such as the the one formed by the hole in the center of a torus. A sphere has genus 0, while a torus has genus 1, see [12].

Further, if a surface is divided into polyhedral regions, the *Euler Characteristic* is defined by

$$\chi(S) = |V| - |E| + |F|$$

where $|V|, |E|$, and $|F|$ are the number of vertices, edges, and, faces respectively, see [12].

In addition to the properties of topological spaces and invariants, the construction of a thickened graph, discussed in Chapter 4 depends on the notion of a deformation retract, and thus, we discuss homotopy. For the following, let I be defined as the closed interval $[0, 1]$.

Definition 2.3.11 If f and f' are continuous maps of the space X into the space Y, we say that f is *homotopic* to f' if there is a continuous map $F : X \times I \to Y$ such that

$$F(x, 0) = f(x)$$

and

$$F(x, 1) = f'(x)$$

for each $x \in X$. The map F is called a *homotopy* between f and f'. If f is homotopic to f' then we write $f \simeq f'$.

If $f \simeq f'$ and f' is a constant map, f is *nulhomotopic.*

A homotopy is essentially a parametric equation for the parameter $t \in I$. Given the homotopy f, we can say f continuously deforms the map f to the map f' as t increases throughout the interval I ($[0, 1]$ as defined above). These maps are incredibly useful in identifying the existence, or lack thereof, of a homeomorphic relationship between two spaces.

Definition 2.3.12 Let $p : E \to B$ be a continuous surjective map. The open set U of B is said to be *evenly covered* by p if the inverse image $p^{-1}(U)$ can be written as the union of disjoint open sets V_α in E such that for each α, the restriction of p to

V_α is a homeomorphism of V_α onto U. The collection V_α will be called a partition of $p^{-1}(U)$ into *slices*.

A simple approach for considering even coverings is the notion of a map p evenly covering an open set U thought of as considering the inverse images $p^{-1}(U)$ as a "stack of pancakes". Each "pancake" is an evenly sized slice of U. The map p can then "squash" these slices onto U.

Definition 2.3.13 Let $p : E \to B$ be continuous and surjective. If every point b of B has a neighborhood U that is evenly covered by p then p is called a *covering map* and E is said to be a *covering space* of B.

Let X be a topological space. Then the covering map for the stack of pancakes example can be given by defining a space E as $X \times \{1, \ldots, n\}$. Then E consists of n copies of the space X "stacked" on top of each other. A covering map for E can be defined as

$$p : E \to X$$

$$\text{where} \quad p(x, i) = x \qquad \text{for all} \quad i \in \{1, \ldots, n\}.$$

Restricting a covering map to a subspace does not necessarily result in a covering map of that subspace. However, conditions can be employed such that this is the case. When considering covering maps, the existence of a *lift* is a useful notion.

Definition 2.3.14 Let $p : E \to B$ be a map. If f is a continuous mapping of some space X into B, a *lifting* of f is a map $\tilde{f} : X \to E$ such that $p \circ \tilde{f} = f$.

Consider Figure 2.5. By the the existence of a continuous mapping $f : X \to B$ and the mapping $p : E \to B$, we say that B can be "lifted" to E by the mapping $\tilde{f}$.

$$\begin{tikzcd}
 & E \arrow[d, "p"] \\
X \arrow[ur, "\tilde{f}"] \arrow[r, "f"'] & B
\end{tikzcd}$$

Figure 2.5: $\tilde{f}$ is a lifting of f.

Some special maps exist from a space to a subspace of itself. These maps provide the generalized topology related to graphs and thickened graphs discussed throughout this paper.

Definition 2.3.15 If $A \subset$ X, a *retraction* of X onto A is a continuous map $r : X \rightarrow A$ such that $r|_A$ is the identity map of A. That is, $r(a) = a$ for each $a \in A$.

Definition 2.3.16 Let $f : X \rightarrow Y$ and $g : Y \rightarrow X$ be continuous maps. Suppose that the map $g \circ f : X \rightarrow X$ is homotopic to the identity map of X and the map $f \circ g : Y \rightarrow Y$ is homotopic to the identity map on Y. Then the maps f and g are called *homotopy equivalences* and each is said to be a *homotopy inverse* of the other.

In addition to the general notion of a retract, we consider a specific case of a retract: the deformation retract. A deformation retract is a special case of homotopy equivalence representing a map that continuously shrinks a space X into $A : A \subset X$.

Definition 2.3.17 Let A be a subspace of X. We say that A is a *deformation retract* of X if the identity map of X is homotopic to a map that carries all of X into A such that each point of A remains fixed during the homotopy. This means

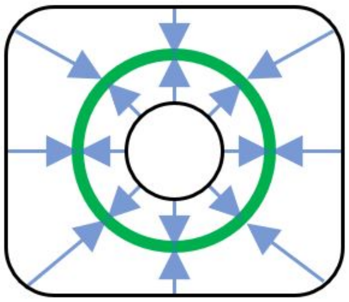

Figure 2.6: The space X is bounded by black boundary components and its deformation retract A is the green circle.

that there is a continuous map $H : X \times I$ such that $H(x, 0) = x$ and $H(x, 1) \in A$ for all $x \in X$ and $H(a, t) = a$ for all $a \in A$.

If A is a deformation retract of X, then X is homotopic to A, and X and A share the same homotopy type.

An example of a deformation retract is given in Figure 2.6. A *skeleton* is another term for a deformation retract of the space that it defines. The skeleton is equidistant to its boundaries at each point and preserves the topology. Studying a space through its skeleton can be useful as the properties of the shape are emphasized.

Chapter 3

Self-Assembly of DNA Graphs

DNA-based computing has been a topic of study in mathematics and theoretical computer science since the Adleman experiment from 1994, see [1]. Mathematical properties of DNA were studied prior to Adleman, e.g. in [7], Tom Head defined the splicing operation on DNA strands spliced by enzymes as a formal operation on strings and sets of strings (languages). However, after Adleman showed that molecules can, in fact, compute the answer to a mathematical problem, the multi-disciplinary field of natural computing, a term coined by G. Rozenberg in the 70s, found its true identity. In addition to mathematics and computer science, natural computing also brings together scientists from biochemistry, molecular biology, bioengineering, and medicine to name a few. When considering DNA for computational purposes, see [1, 21], there are significant benefits to DNA computation versus traditional computing. By allowing millions of molecules to interact simultaneously, DNA computing is capable of achieving a massive parallelism that is not feasible with traditional computing. Additionally, the storage potential and energy efficiency of DNA far outweighs classical computing technologies, see [1].

Dioxyribonucleic Acid (DNA) is a molecule constructed of nucleotides. Each nucleotide consists of a phosphate group, a base, and a sugar and there are four different types of nucleotides based on their bases: Adenine (A), Guanine (G), Thymine (T), and Cytosine (C). The nucleotides used in constructing DNA are also classified as purines (Adenine and Guanine) and pyrimidines (Cytosine and

Thymine). Nucleotides can attach to each other by covalent (strong) bonds in any order to form a single strand in the direction of $5'$ to the $3'$ end denoting the phosphate end and the hydroxyl group end respectively. Hence, these singe DNA strands are oriented. In addition, nucleotides form hydrogen bonds (weaker) between two strands, if the strands are the Watson-Crick complements of each other, i.e. A and T bind together (are complements of each other) and also C and G. When two strands bind together to form a double-stranded molecule, as found in nature, the strands are oppositely oriented.

The structure of the nucleotide bonds is integral to the weighting methods we consider later. As shown in Figure 3.1, the A/T bonds require two hydrogen bonds to coalesce, while the C/G bonds require three hydrogen bonds to coalesce. This creates a stronger bond between the C/G base pairs versus the A/T base pairs.

Short, single stranded DNA molecules called oligonucleotides can be constructed with user defined sequencing for research purposes, including DNA Computing. When single strands of DNA are placed in a solution, strands with complementary bases attach to each other on their own due to the Watson-Crick complementarity and form hydrogen bonds to create the traditional double stranded molecule found in nature. In this process of hybridization, open nicks may form between two neighboring nucleotides in a strand. These can be naturally repaired by an enzyme called ligase when added to the solution and the covalent bonds between such nucleotides can be formed to create the double stranded DNA molecule. Thus, single stands are unstable and if they meet their complements in a solution, they will attach to them when there is a significant portion of complementary pairs. That is why it is imperative that this property is considered when the encoding of a problem into DNA sequences is determined. This gave rise, see [9], to the study of unwanted hybridization through involution mappings and DNA codes, which led to a generalization of classical coding theory concepts.

In short, a mathematical problem can be encoded on DNA single strands, or single stranded overhangs of otherwise double stranded molecules and left in a solution to self-assemble, i.e. to form double strands by themselves due to the Watson-Crick complementarity. The encoded building blocks may also be branched junction molecules with "sticky" single stranded ends resembling a vertex and its incident edges attached to it as "arms". Different problems are studied using a variety of algorithms in which different building blocks are used. Once a DNA computer produces a desired structure, i.e., solves a problem, the resulting solution is encoded in a DNA molecule or it is the existence of the molecule itself. Strands that form in the ligase solution can be isolated and read through gel electrophoresis. To aid in the extraction of solutions, enzymes can be employed to destroy bonds or strands that do not meet the structural requirements of a solution.

The limitations of current DNA computing capacity are consistently being challenged in the lab. In 2009, Wu et al. [26], showed that the reporter strand solution to the 3-colorability problem hypothesized by Jonoska et al. [10] could be completed in an experimental lab. The DNA solution to this problem was a strand of 804 nucleotides. This experiment additionally identified the presumed bound on the size of graphs as at most 13 vertices and 28 edges for this method based on current technology.

Given a graph, its vertices can be encoded on branched junction molecules and edges can be encoded on duplex (double stranded) molecules. These molecular *building blocks* have single-stranded (sticky) piece at each end. This sticky end carries appropriate encoding and leaves the end of the block ready to bond only with allowed complimentary molecules. For graph problems, edges are represented by DNA strands and vertices are represented by k-armed junction molecules (k-armed double helixes joined at a single point, each with a sticky overhang), see [10]. When put into a ligase solution, these molecular building blocks will autonomously assemble into graph structures. This process of DNA building blocks assembling

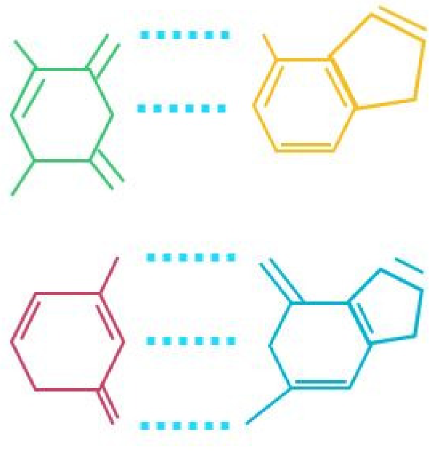

Figure 3.1: Hydrogen bonds between various nucleotides in two DNA strands.

into complete structures is called *self-assembly* and if it represents graphs, then it is referred to as *self-assembly of DNA graphs*.

Following self-assembly, resulting molecules can be read using a method such as gel electrophoresis. This allows the solution to problems solved though self-assembly to be decoded. DNA computing has been shown to give polynomial time solutions (polynomial number of steps in the algorithm) to nondeterministic polynomial problems, beginning with Adleman's Hamiltonian Path Problem solution, see [1]. Self-assembly has shown to give polynomial time solutions to the 3-SAT and 3-colorability problems (both examples of NP-complete problems), see [10, 11, 26]. There are, however, limits to the computational capabilities of DNA with the present technology. Wu et al. [26] showed that the 3-colorbility DNA-based computing solution is limited to graphs with at most seven vertices and 23 edges. In 2003, Jonoska et al. [10] showed the application of self-assembly in solving the 3-SAT and k-colorability problems with a theoretical method and laboratory testing. As with prior experiments, these algorithms, though requiring exponentially increasing biological materials, do not increase in complexity as the magnitude of the problem increases and thus represent a polynomial time solution. These processes are highlighted below.

3.1 Solving the 3-SAT Problem through Self-Assembly

The SAT problem asks whether there exists a set of truth value assignments for the boolean variables in a logical formula, such that the logical formula takes value "true" or 1. In 1971 Cook proved in the now famous Cook's Theorem, see [8], that the SAT problem is NP-complete and it became the classic example of an NP-complete problem that other problems are often reduced to in order to establish their NP-completeness. Since any logical formula can be converted into Conjunctive Normal Form (CNF) in polynomial number of steps, we only need to investigate the computational complexity of CNF formulas. A formula is in CNF if it is a conjunction of clauses, such that every clause is a disjunction of literals (variables or their negation). If all clauses have three literals, the form is called 3-Conjunctive Normal Form or 3-CNF. Since even solving the 3-SAT problem, i.e., the instance when the formula is in 3-CNF is NP-complete, we can only consider that special case.

Consider, the following formula in 3-CNF which consists of three clasues and has three boolean variables:

$$\alpha = (\bar{x} + y + \bar{z})(x + \bar{y} + z)(\bar{x} + \bar{y} + z).$$

Here each clause is in paretheses. The literals in a clause are separated by a "+" denoting disjunction and the sign for conjunction is omitted. The bar of a variable, e.g., $\bar{x}$ denotes the negation, or else, complement of x and the set of boolean variables is $\{x, y, z\}$. Since the formula α is small, it is easy to verify that it is satifiable, since we can check all eight truth value assignments to the variables and can see that some, e.g. $\{1, 1, 1\}$, make the formula true. In general, there are no known fast algorithms and when the number of variables grows, the possibilities of truth value assignments grow exponentially.

Two methods have been proposed for self-assembly solutions of the 3-SAT problem by Jonoska et al. [10, 11]. One algorithm uses contact networks, where a self-assembled structure traverses through junctions beginning with a source and ending with a target and representing whether or not each clause has been satisfied. The second method constructs a graph, $G(\alpha)$, and then constructs molecular "building blocks" each representing one clause or one variable. This method is described below.

As each clause has three literals, each clause molecule is 3-armed and each arm has a sticky end unique to the clause and to the truthvalue of the literal it represents. Each variable molecule has as many arms as the number of clauses the variable is included in, plus it is connected to its negation and the negation itself has as many arms as the number of clauses it appears in. For example, the junction molecule for x and $\bar{x}$ has one arm on the side of x and two arms on the side of $\bar{x}$. The following algorithm is proposed.

1. Combine all variable building blocks with all clause junctions in a single test tube and allow the complementary ends to hybridize and be ligated.
2. Determine whether the DNA graph structure corresponding to $G(\alpha)$ has formed by:
   (a) Removing partially formed 3D DNA structures with open ends that have not been matched.
   (b) From the graphs formed in the above steps, remove the graphs that are larger than the original graph.
   (c) If there are graph structures remaining in the tube, then we conclude that the formula is satisfiable, i.e. that there are variable assignments that give the formula value 1. Otherwise, the formula is not satisfiable.

This algorithm remains in constant number of steps, regardless of the number of variables or clauses in the formula. The amount of DNA based material used in the experiment grows at a non-polynomial rate as this number increases.

3.2 Solving the k-colorability Problem through Self-Assembly

Also NP-complete, the k-colorability problem asks whether for a given graph, it is possible to color its vertices using k colors in such a way that no two adjacent vertices have the same color. This problem can be addressed using self-assembly in a manner similar to that used in the 3-SAT problem.

For each edge in a graph, a double stranded molecule is constructed with two sticky ends, each of which represents a different "color" and is encoded to be the complement of the vertex it is incident with. Many-junctioned molecules are constructed to represent the vertices of differing degree. All sticky ends of all arms of one vertex are encoded with one "color" and each arm is encoded relative to a specific edge incident with the vertex. The algorithm used to generate the solution (or lack thereof) for the k-colorability problem follows a similar path as that of the 3-SAT problem [11].

1. Combine all vertex building blocks with all edge building blocks in a single test tube and allow the complementary ends to hybridize and be ligated.
2. Determine whether the DNA graph structure has formed by:
   (a) Removing partially formed 3D DNA structures with open ends that have not been matched.
   (b) From the graphs formed in the above steps, remove the graphs that are larger than the original graph (possible dimers and trimers).
   (c) If there are graph structures remaining in the tube, then we conclude that the graph can be colored with k colors.

For both of these problems, laboratory tests were conducted, see [10, 11]. The pitfalls of the experimentation, such as yield are discussed in [10, 11], but for each circumstance, the expected solution molecule was obtained which shows that these

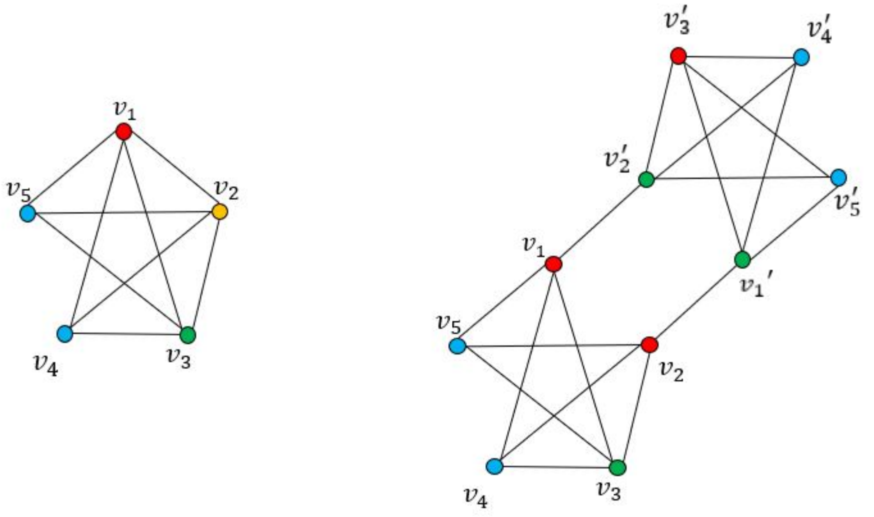

Figure 3.2: A Graph and its covering graph.

algorithms provide feasible biological solutions to these NP-complete problems.

3.3 Covering Graphs

The graph structures created through self-assembly can be regarded as 1-complexes, and thus topological properties can be applied to the structure. One possible complication of self-assembly is the accidental creation of a covering graph. Physically, this case occurs where there are cases where local ligations that occur correctly, however they create globally larger graphs that represent multiple copies of the original graph. Mathematically, this follows the definition of a covering map from Section 2.3. The exposed end has locally homeomorphic images in the solution that are globally dissimilar.

For each problem, the implications of covering graphs must be considered. In the case of the contact network solution to the 3-SAT problem, the construction of a covering graph is not indicitive of a false positive. However, when considering the 3-colorability solution using self-assembled vertex and edge building block molecules, it is possible for a covering graph to be created representing a false positive for 3-colorability. An example of such a covering graph is provided in Figure 3.2, where the graph is not 3-colorable but the covering graph formed is 3-colorable. Since such false positives must be excluded from the test tube with possible solutions, Step 2(b) is needed in the algorithm.

3.4 Weighting of Self-Assembled Graphs

Using the Watson-Crick complementarity and the hydrogen bonds shown in figure 3.1 to form pairs, the choice of nucleotides in an oligoneucleotide can be used to simulate the graph theory concept of weight in self-assembled DNA molecules. By assigning the number of Cytosine and Guanine nucleotides in an oligoneucleotide proportionately to the weight of an edge, the graph theoretic concept of weight can be translated to DNA graphs, see [24].

In the case of constructing a weighting schema for a self-assembled directed graph, three representative strands for each edge are necessary. For the edge, one weight represents the lowest weighted path, one weight represents the higher weight edge, and then the final weight represents the combined weights of the arcs, traversing the edge in both directions. If only strands representing the two individual directed edges were introduced to the ligase solution, the strand incorporating both elements could not be represented as each strand can only self-assemble with a matching complement.

Li et al., see [15], describe a variety of prior weighting schema for DNA based graphs. One of the first described weighting methods for self-assembly was created by Narayanan and Zorbalas in [24]. This algorithm creates weighted molecules with the number of nucleotide pairs constructing the edge proportionate to the weight of the edge [19]. For graphs with a relatively small range of edge weights, such encoding seems reasonable. The method was tested on travelling salesman problems with positive results. However, this method fails to account for the circumstances where edge weights may differ by orders of magnitude. The likelihood that longer oligonucleotides will hybridize (join together) is much higher than for shorter oligonucleotides.

The next weighting method considered was proposed by Yamamura et al. in [24], where the actual concentration of oligonucleotides present in the ligase solution is representative of the weights (edges with higher weights would have less representation in the solution), see [27]. However, this method can't guarantee that the optimal solutions are recognized and requires a high degree of difficulty in practical circumstances.

Li et al. described in [15] a method for encoding of numerical data on DNA used for computation through a temperature gradient. This approach relies on the "melting temperature" required to denature the DNA such that the strands with a lower melting temperature will separate more quickly and the solution with the highest melting temperature will remain. This method was applied to the traveling salesman problem as a test case and showed a satisfactory solution. In the laboratory, this could represent a reduction in the computation time for the solution discussed here.

Chapter 4

Thickened Graphs

Throughout this paper graphs and their associated postman tours are studied by embedding graphs within a topological surface. These surfaces, called thickened graphs, are introduced in this chapter and the construction of homeomorphic surfaces is identified. In studying thickened graphs, we can widen the amount of information generated about a graph by analyzing the surface based on its topological invariants. The relationship between thickened graphs and DNA self-assembly is also explored.

4.1 Properties of Thickened Graphs

A thickened graph is a graph where each undirected edge is expanded to two directed edges oriented in opposite direction. The face created between the directed edges along with the edges themselves make up the *edge ribbon*. We proceed with the definitions and terminology defined in [12].

Definition 4.1.1 Let G be a finite graph. A *thickened graph* $F(G)$ (of G) is a compact orientable surface (2-dimensional manifold) such that G is topologically embedded (as a 1-complex) in $F(G)$ as a deformation retract.

As G is embedded in $F(G)$ as a deformation retract, G is the skeleton of $F(G)$.

The most intuitive construction of $F(G)$ is constructed by simply "thickening" each edge in the graph and converting the line segment representing an edge to an *edge ribbon* as shown in the middle image of 4.1. However, $F(G)$ is not uniquely defined by G. For a graph G, there may be many thickened graphs. The thickened graphs of G need not be homeomorphic. We denote the family of all thickened graphs of G by $T(G)$. An example of a thickened graph is shown in Figure 4.1. Given the definition of a thickened graph, G is a skeleton of any $F(G)$ that is created.

Though the concept of thickened graphs is most easily seen intuitively within planar graphs, non planar graphs can also be thickened with each point on an edge ribbon being locally two-dimensional despite crossing over another edge.

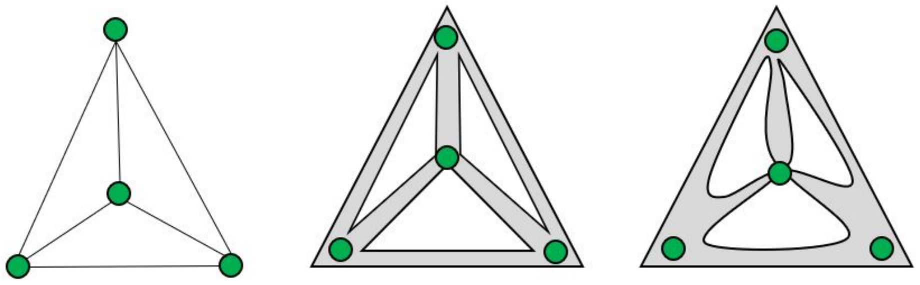
Figure 4.1: A graph G, a thickened graph $F(G)$, and a thickened graph $F'(G)$ following an elementary boundary operation.

For each thickened graph $F(G)$, the boundary components of the surface $\delta(F(G))$ can be identified. We denote the number of boundary components in $F(G)$ as $|\delta(F(G))|$. The maximal number of boundary components, also called the *maximal double strand number*, of any thickened graph in $T(G)$ is denoted by $DS(G)$ while the minimal double strand number is denoted by $ds(G)$.

For a graph G that is the union of two disjoint graphs G_1 and G_2 the graph G

will always have two distinct sets of boundary components: those belonging to G_1 and those belonging to G_2. This gives that

$$DS(G) = DS(G_1) + DS(G_2)$$

and the similar result for the minimal double strand number of G.

The concept of Betti numbers relates well to topological graph theory. The first two Betti numbers β_0 and β_1 have intuitive definitions. The zeroth Betti number, β_0 represents the number of components in a graph while the first Betti number, β_1 is equal to the number of loops within a graph. Subsequent Betti numbers are not considered in this paper.

A graph G can be embedded in a surface S. The minimal genus of a surface in the set of the surfaces in which G could be embedded is defined as the minimal genus of G denoted $\gamma(G)$. Similarly, the maximal genus of G is denoted $\gamma_m(G)$, following [12].

Additionally, if the empty surface between boundary components of a thickened graph $F(G)$ were filled with a disk to create a solid surface without boundary, the Euler characteristic $\chi(G)$ can be calculated.

Jonoska and Saito showed in [12] that

$$DS(G) = 2 - \chi(G) - 2\gamma(G) = (\beta_1(G) + 1) - 2\gamma(G).$$

Lemma 4.1.2 *The bounds of $DS(G)$ can be defined as follows:*

$$ds(G) \leq DS(G) \leq \beta_1(G) + 1$$

where $\beta_1(G)$ is the first Betti number of the graph.

Proof. By the definition, the minimal number of boundary components of a thickened graph is at most equal to the maximal number of boundary components of a thickened graph. Thus, we focus on the second half of the inequality $DS(G) \leq \beta_1(G) + 1$.

From the prior statement, $DS(G) = (\beta_1(G) + 1) - 2\gamma(G)$. If a thickened connected 3-valent graphs can be embedded in a sphere, then $2\gamma(G) = 0$ and then equality holds, i.e., $DS(G) = \beta_1(G) + 1$. Otherwise, $2\gamma(G) \geq 1$, which results in a strict inequality. Thus, $ds(G) \leq DS(G) \leq \beta_1(G) + 1$.

■

Beyond constructing an upper bound for the maximal double strand number, the first Betti number, $\beta_1(G)$, is also of opposite parity to a potential $|\delta(F(G))|$ for any $F(G)$ that can be constructed.

Lemma 4.1.3 *The number of boundary components of a thickened graph $F(G) \in T(G)$ must be of the same parity as $\beta_1(G) + 1$. That is:*

$$|\delta(F(G))| = (\beta_1(G) + 1)(mod 2)$$

for all $F(G) \in T(G)$.

Recalling the equation from Section 2.3, the Betti Number of a topologically embedded graph G can be calculated as $\beta_1(G) = m - n + k$ where $m = |E(G)|$, $n = |V(G)|$, and k is the number of components in G. Considering a 3-valent connected graph of order n, the number of edges must be $\frac{3n}{2}$ and the number of components must be one. Thus, $DS(G) \leq \frac{n+2}{2}$.

4.2 Constructing the Family of Thickened Graphs $\mathcal{F}(G)$

From a thickened graph $F(G)$, multiple thickened graphs can be created through elementary boundary operations at one or more of the vertices of $F(G)$, as shown on the right of Figure 4.1. An *elementary boundary operation (EBO)* at a vertex is a permutation of the edges incident with a vertex. Considering a vertex of degree 3, there are two possible permutations: (123) and (132) that create 3-cycles. A representation of these two permutations is shown in Figure 4.2.

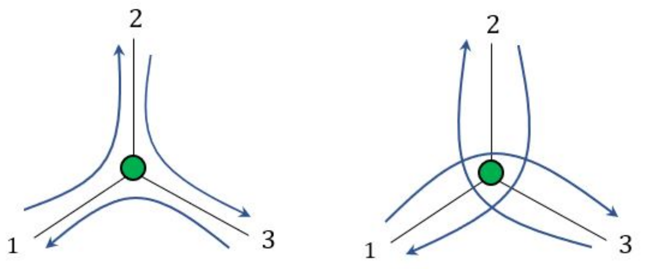

Figure 4.2: A vertex v before and after an EBO.

By performing an elementary boundary operation at one or more of the vertices of a graph G additional surfaces are constructed. As stated above, these surfaces need not be topologically homeomorphic, however they must maintain the same parity of boundary components in $T(G)$.

The effect of elementary boundary operations on the number of boundary components *involved* with a vertex of a thickened graph has interesting properties. For a thickened graph $F(G)$ an elementary boundary operation can be performed at a vertex v, creating another thickened graph $F(G')$. The number of boundary components involved with v in G' depends on the number of boundary components involved with v in G. Given a 3-valent graph, the number of boundary components involved with v can be either 1, 2, or 3.

Lemma 4.2.1 *Performing an elementary boundary operation from* (123) *to* (132) *at vertex v changes the number of components involved with v. These changes to the number of involved components can be represented as* $1 \leftrightarrow 1$, $2 \leftrightarrow 2$ *and* $1 \leftrightarrow 3$. *That is, if the number of boundary components involved with v is 1, then the EBO will result in 1 or 3 involved components. If the number is 2, then the number*

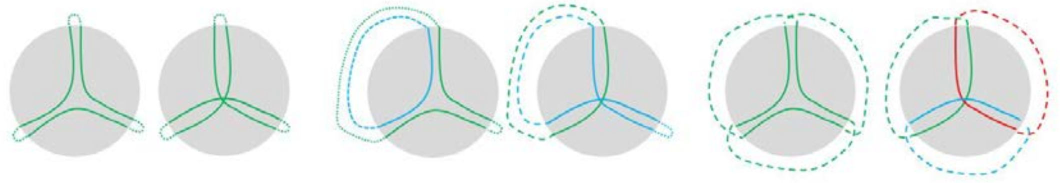

Figure 4.3: Resulting changes tothe number of boundary components.

remains unchanged. If the number is 3, then there the resulting graph will have 1 component involved with v.

Proof. This property can be shown by performing the operations as shown in Figure 4.2. Figure 4.3 shows the boundary components within the local neighborhood of a vertex v as solid lines and the extension to the rest of the graph in dashed lines. Each local vertex area is shown before and after an elementary boundary operation for each possible case.

As there are only two permutations of (123), each relationship is bidirectional. That is, if 1 component can become 3 following an elementary boundary operation then 3 components can become 1. The three possible cases are considered below.

There are two possible scenarios where the vertex has one involved component.

Case 1. The first case, shown in the left two images of Figure 4.3, considers a component which traces an edge from v, visits one or more other vertices in $V(G)$, and then returns to v using the same edge. In this case, the EBO will maintain the same structure locally with one involved component at v, though the path of the boundary component will change.

Case 2. If v has only one involved component and does not fit case one, it must have the structure shown in the right of Figure 4.3, where each traversal of v returns to the vertex using an in-edge different than the out-edge. Following a boundary operation, this unifies three prior involved components into one component.

Otherwise, there are two components involved at v, resulting in the final case.

Case 3. If v is involved with two components, it must have the structure of the center two graphs in Figure 4.3. Given $\{e_1, e_2, e_3\} = E(G)$, one component visits v using e_1 and leaves using e_2. The second component must then traverse e_1 and e_2 in the opposite orientation and traverse e_3 in both directions. An elementary boundary operation changes which pair of edges is traversed by the first boundary component with the complement and retains the remaining structure.

As these properties are bidirectional, the only possibility for a vertex involved with three boundary components is addressed in case two, and thus these three cases define all possible conditions.

■

An additional property of boundary components of thickened graphs is how a boundary component interacts with specific edge types within a graph. Each edge ribbon may be traversed by two oppositely oriented boundary components or by a single boundary component in each direction. In a graph that is constructed as two components connected through a single edge, e, that edge is called a *bridge*.

Lemma 4.2.2 *Given a graph G with a bridge e, a boundary component σ in the thickened graph $T(G)$ either does not traverse e or traverses e twice.*

Proof. Assume there exists a σ in the boundary components of $T(G)$ such that σ traverses e exactly once. Let e separate components C_1 and C_2, such that $C_1 \bigcup e \bigcup C_2 = G$. Then $p(\sigma) = p_{\sigma_1} e p_{\sigma_2}$ where σ_1 traverses edges exclusively in C_1 and σ_2 traverses edges exclusively in C_2. By definition, $p(\sigma)$ must begin and terminate at the same vertex v. This vertex must reside in C_1 or in C_2. Without loss of generality, assume v is in C_1. As defined, $p(\sigma)$ can not be complete and must return to C_1. However, the only edge connecting C_1 and C_2 is e, which σ has already traversed. This is a contradiction. Either $p(\sigma)$ traverses only edges in C_1 and thus doesn't traverse e or $p(\sigma)$ traverses e twice.

■

4.3 Thickened Graphs and DNA Graphs

Thickened graphs can be directly correlated to the structure created when graphs self-assemble from DNA structures.

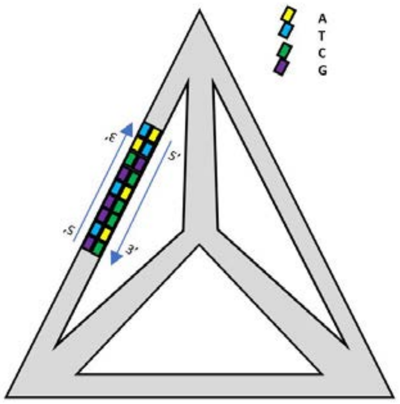

Figure 4.4: DNA Structure representing a thickened graph of K_4.

Each edge ribbon in a thickened graph is represented by double stranded DNA molecule. The orientability of the manifold is necessary in the DNA representation, as the strands paired by the Watson-Crick complementarity must run in opposite directions (a strand from $5' \to 3'$ must pair with a strand oriented from $3' \to 5'$).

When an elementary boundary operation is performed, the now permuted in and out edges can change the orientation of the boundary component it is joining. Additionally, the manner in which this join is oriented can change the number of boundary components within the graph.

A boundary component of a thickened graph is respresented by a single strand of DNA. If this boundary component is extracted from the self-assembled structure, it can be read through gel electrophoresis or electron microscopy. Reading this structure will give the details of the sequence and the orientation of the path that the boundary component represents.

Chapter 5

3-Valent Graphs

Regular graphs are graphs in which all vertices have the same degree. This leads to some simplification in investigating graph preperties. An r-regular graph is a graph, each of whose vertices has degree r. We investigate graphs for which $r = 3$, since it was shown in [13] that complex graphs can be reduced to 3-valency, thus providing for more uniformity and for additional beneficial properties that can be added to the analysis.

In Chapter 4, it was shown that for $G = G_1 \cup G_2$ with G_1 and G_2 being distinct components,

$$|\delta(F(G))| = |\delta(F(G_1 \cup G_2))| = |\delta(F(G_1))| + |\delta(F(G_2))|.$$

Thus, for the analysis of reporter strands, only connected graphs need to be considered. To further reduce the complexity, we consider only 3-valent graphs. For graphs with vertices of more than degree three, a 3-degree perturbation can be employed to reduce the graph to a 3-regular graph. This chapter details the process of reducing an arbitrary graph to 3-valency and how the content of this paper can be extended to all graphs. It provides a construction of a 3-regular graph from K_5.

5.1 Reduction to 3-Valency

The reduction of any multi-graph to a 3-valent graph allows properties of 3-regular graphs to be extended to the graph in question. For the graphs discussed here, the initial considerations made are vertices of degree one (pendants) or degree two.

In order to construct an Eulerian cycle on a graph with a vertex of degree one (a pendant), p, the edge connecting p to the remainder of the graph must be traversed twice sequentially. When considering a DNA based structure, this is considered a *hairpin* formation. These need not be considered as the topology of the thickened graph remains unchanged.

Similarly, the edges incident to a vertex of degree two must be traversed in order according to orientation. These two edges e_i, e_{i+1} and the associated vertex v_i, forming the path $v_{i-1}e_iv_ie_{i+1}v_{i+1}$ can be collapsed to form another edge $e\prime$ with a resulting path $v_{i-1}e\prime v_{i+1}$ representing the path $v_{i-1}e_iv_ie_{i+1}v_{i+1}$ connecting v_{i-1} and v_{i+1}. Because of this construction to replace a vertex of degree two, there is no topological change to the graph by replacing the vertex, so these vertices are not considered. Going forward, only graphs with vertices of degree 3 or higher are considered.

In Figure 5.1, the left image shows a graph G which is similar to the graph K_4 with two additional vertices, one of degree 2 and one of degree 1. The center image shows a thickened G, and the right image shows a topologically equivalent surface. The vertices of degree one and two can be removed from the graph without changing the topology of the surface. Thus, we only consider vertices of degree 3 or higher.

In order to reduce a vertex of degree four or higher, a *3-degree perturbation* is performed. The 3-degree perturbation is defined by Jonoska et al. in [13] and shown in Figure 5.2. Given a graph $G = (V, E, t)$ with a vertex $v \in V$ of degree

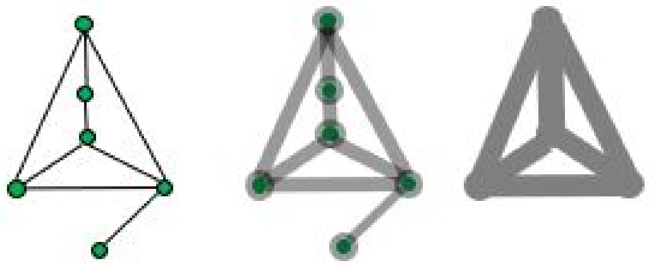

Figure 5.1: A graph G, it's thickened graph $F(G)$, and the thickened graph $F(G')$, where G' is obtained from G by removing the pendant.

k, with $k \geq 4$, add $k-3$ vertices to the graph to form $G' = (V', E', t')$ with $V' = V \cup \{u_1, u_2, \ldots, u_{(k-3)}\}$. Similarly, we add corresponding edges to the set of edges to create

$$E' = (E \setminus \{e_3, \ldots, e_k\}) \cup \{f_1, \ldots, f_{k-3}, e'_3, \ldots, e'_k\}$$

satisfying:

$t'(f_i) = \{(u_{i-1}, u_i) \mid i = 1, \ldots, k-3\}$ with $u_0 = v$, the vertex being perturbed.
$t'(e'_j) = \{(v_j, u_{j-2}) \mid j = 3, \ldots, k\}$ with $u_{k-2} = u_{k-3}$
and
$t'(e) = t(e)$ for $e \in E$.

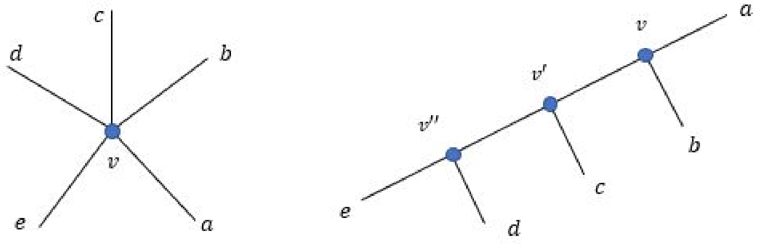

Figure 5.2: A 5-degree vertex that has been perturbed.

An example of a 5-degree vertex that has been perturbed to a set of 3-valent vertices is given in Figure 5.2.

Theorem 5.1.1 *If G' is a 3-degree perturbation at a vertex v of G, then G is a homomorphic image of G'.*

Proof. Let $\phi : G' \to G$ be a graph homomorphism defined by the following: for each $v \in V$, $\phi(v) = v$ and for each $e \in E$, $\phi(e) = e$. For each new vertex in G', $u_j \in V' \setminus V$, $\phi(u_j) = v$ for $j = 1, \ldots, k-3$. For each new edge in G', $\phi(f_j) = v$ and $\phi(e'_j) = e_j$. By the construction of G', ϕ is a graph homomorphism and is surjective.

■

Definition 5.1.2 A *perturbed 3-degree graph* denoted T_G for a graph $G = (V, E, t)$ is a graph obtained from G by successive 3-degree perturbations at every vertex of degree 4 or higher.

The value of these perturbed graphs is two-fold: the properties of 3-regular graphs can be applied to the perturbed graph, while the perturbation maintains the underlying structure of G as a graph homomorphism.

It is valuable to note that a perturbation of series of perturbations of a the vertices of a graph G result in a homomorphic image G'. That is, the perturbation preserves the structure of G. If a vertex in v in G is adjacent to u in G then this adjacency is preserved in the perturbation of G.

Corollary 5.1.3 *Let $T_G = (V_T, E_T, t_T)$ be a perturbed 3-degree graph of $G = (V, E, t)$. Then G is a homomorphic image of T_G.*

Proof. Let $G = G_0, G_1, \ldots, T_G = G_k$ be a sequence such that G_i is a 3-degree perturbation at a vertex of G_{i-1} for all $i = 1, \ldots, k$. Then, by the theorem above, there are graph homomorphisms $\phi_i : G_i \to G_{i-1}$. The composition $\phi_1 \circ \ldots \circ \phi_k = \phi_T$ of graph homomorphisms is a graph homomorphism from T_G onto G.

■

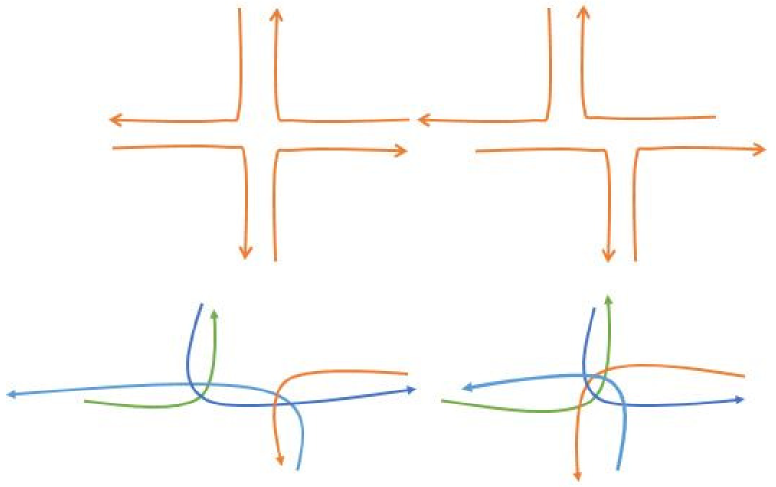

Figure 5.3: A 3-degree perturbation of a 4-degree vertex, a permutation of the edges, and the lift.

Given that T_G is a perturbed 3-valent graph of G, we can consider the homomorphism between T_G and G. The *perturbation map* defines this relationship.

Definition 5.1.4 Let T_G be a perturbed 3-degree graph of G and let the sequence $G = G_0, G_1, \ldots, G_k = T_G$ be such that G_i is the 3-degree perturbation at a vertex of G_{i-1} for all $i = 1, \ldots, k$. Let $\phi_i : G_i \to G_{i-1}$ be the corresponding vertex perturbations. The homomorphism $\phi_T : T_G \to G$ obtained as the composition $\phi_T = \phi_1 \circ \ldots \circ \phi_k$ is called a *perturbation map.*

Generally speaking, the perturbation map is the map that maps the perturbed graph T_G to the original graph G. The perturbation map is a composition of each individual map $\phi_i : T_{G_i} \to G$ where T_{G_i} is the graph G with the ith vertex perturbed to 3-regularity.

5.2 A 3-Degree Perturbation of K_5

In order to display the construction of a 3-valent pertubed graph, we show the process over K_5, the 4-regular graph of order 5. As stated in the definition of the construction, for each vertex v of degree higher than 3, $deg(v)-3$ additional vertices are added. For K_5, there are five vertices of degree 4, resulting in five perturbations, each of which creates one new vertex. These five perturbations of vertices of degree 4 result in a 3-regular graph of order 10. The first perturbation is shown in the left two graphs of Figure 5.4. A perturbed 3-valent graph of K_5 is shown on the right.

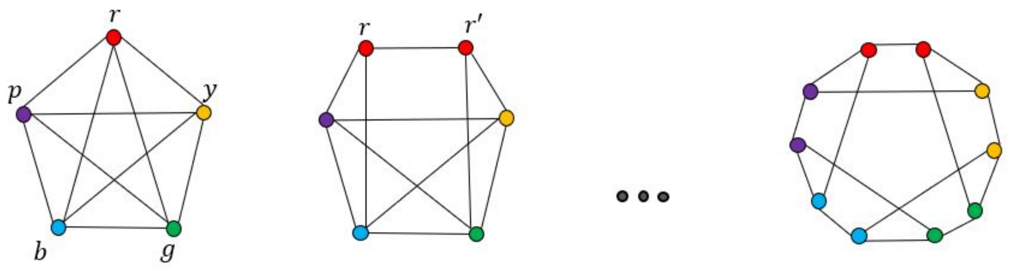

Figure 5.4: The stages of creating a perturbed 3-valent graph of K_5.

For the example above, the graph G initially begins defined as follows:

$$G = (V, E, t)$$

$$V = \{r, y, g, b, p\}$$

$$E = \{\{r, y\}, \{r, g\}, \{r, b\}, \{r, p\}, \{y, g\}, \{y, b\}, \{y, p\}, \{g, b\}, \{g, p\}, \{b, p\}\}$$

After the first perturbation on the red vertex, r, the edge set, vertex set, and ordered pairs of G are changed to represent G' as follows:

$$G = (V, E, t)$$

$$V = \{r, r'y, g, b, p\}$$

$$E = \{\{r', y\}, \{r, r'\}, \{r, g\}, \{r, b\}, \{r, p\}, \{y, g\}, \{y, b\}, \{y, p\}, \{g, b\}, \{g, p\}, \{b, p\}\}$$

After the final perturbation, T_G is defined as follows:

$$G = (V, E, t)$$

$$V = \{r, r', y, y'g, g'b, b'p, p'\}$$

$$E = \{\{r, r'\}, \{y, y'\}, \{g, g'\}, \{b, b'\}, \{p, p'\}, \{p', r\}, \{r', y\}, \{y', g\}, \{g', b\}, \{b', p\},$$

$$\{r, b'\}, \{r', g\}, \{y, p'\}, \{y', b\}, \{g, r'\}, \{g', p\}, \{b, y'\}, \{b', r\}, \{p, g'\}, \{p', y\}\}$$

In most cases, a 3-degree perturbation could be performed at a vertex in multiple ways. In the case of the example given, the outer cycle is maintained and the internal edges adjacent edges are perturbed. However, this example represents only one possible perturbation map.

5.3 Lifting the 3-Valent Graph

The ability to return to the original construction following the perturbation is necessary. In order to map to the original graph G following the perturbation, the thickened graph is *lifted.* The general concept of lifting a surface is described in the topological background section (Section 2.3). As thickened graphs are topological surfaces, the concept of lifting can be applied to thickened graphs as shown by Jonoska et al. in [13]. This process is defined in the following section.

Within a thickened graph, each boundary component σ is oriented with opposing orientations on the boundaries of each edge ribbon. For a graph G, define $F(G)$ as a thickened graph of G. Within $F(G)$, each edge must be traversed twice, once in each direction. The edges of this edge ribbon may be traversed by one boundary component or by two boundary components.

Define an orientation of $F(G)$ as $\mathcal{O}$. A boundary component σ traverses some number of edges $e_i \in G$ separated by vertices w_i in $F(G)$. A path $p_\sigma = (w_0 e_1 w_1 \ldots e_{n-1} e_n w_0)_\sigma$ can be assigned as the path is traversed by σ. The order of the path p_σ is uniquely determined by the orientation. If p_σ traverses the path $e_i w_i e_{i+1}$ then $V^{\mathcal{O}}(e_i) = e_{i+1}$.

A permutation is called a *cyclic permutation* if and only if the non-fixed elements of the permutation are contained in exactly one cycle. If S is the subset of non-fixed elements of a cyclic permutation and $|S| = k$, then the permutation is a k-cycle.

Lemma 5.3.1 *If v is a vertex in G of degree k, then for every oriented thickened graph $F(G)$, the permutation $v^{\mathcal{O}}$ is a k-cycle.*

Proof. Let $S \subset \{e_1, \ldots, e_k\}$ exists such that $v^{\mathcal{O}}(S) = S$. This gives $|S| < k$. Let σ be a boundary component visiting vertex v. Then, either σ traverses a pair (pairs) of edges $e_i, e_j \in S$ or traverses two edges not in S.

No edges in $\{e_1, \ldots, e_k\} \backslash S$ are traversed by σ, otherwise the number of edge ribbons at v is $|S| < k$, but v is of degree k by the hypothesis.

In this case, when a deformation retract reduces the thickened graph to its skeleton, G, the ribbons representing each edge would retract to one point for the edges in S and each edge not in S would map to at least one separate point.

Thus, as each edge in S is permuted in exactly one cycle and each edge not in S remains fixed, S forms a cyclic permutation. Given that the complete edge set is of cardinality k, we can conclude that S is a k-cycle.

■

Given a graph G, the process contained here considers perturbing G to create T_G and then thickening G to create $F(T_G)$. It is important to show that the properties identified for $F(T_G)$ can be identified in a thickened graph of G. That is, we must show that $F(T_G)$, the thickened graph of T_G, can be lifted to a thickened graph $F_T(G)$ of G. The following lemma shows that for a boundary component in a perturbed thickened graph, the same boundary component exists in the thickened original graph.

Theorem 5.3.2 *Let G be a connected multigraph and $F(T_G)$ be a thickened graph of a perturbed 3-degree graph T_G of G. Let $F_T(G)$ be the lift of $F(T_G)$. Then for every boundary component σ' in $\partial(F(T_G))$ there is a boundary component σ in $\partial(F_T(G))$ such that $\phi_T(p_{\sigma'}) = p_\sigma$ where $\phi_T : T_G \to G$ is the perturbation map. Moreover, this correspondence is one-to-one, i.e., if $\sigma' \neq \sigma''$ in $\partial(F(T_G))$ then $\phi_T(p'_\sigma) \neq \phi_T(p''_\sigma)$.*

Proof. Let $G = (V, E, t)$ be a graph and T_G be a 3-degree perturbation of G with $F(T_G)$ as a thickened graph of T_G. Then $F_T(G)$ is the lift of the thickened graph $F(T_G)$ containing G as a deformation retract.

Let σ be a boundary component in $F_T(G)$, and define the path traversed by σ as

$$p_\sigma = (w_0 e_2 w_1 \ldots w_{n-1} e_n w_0)_\sigma$$

with orientation $\mathcal{O}$. That is, σ traverses n edges and terminates in its starting vertex. Because σ is a k-cycle, $(w_1 \ldots w_{n-1} w_0)^{\mathcal{O}}(e_1) = e_1$ and $(w_i \ldots w_{i+r-1})^{\mathcal{O}}(e_i) = e_{i+r}$.

Similarly, let σ' be a boundary component in the perturbed graph $F(T_G)$ and define

$$p_{\sigma'} = (x_0 f_1 \ldots x_{m-1} f_m x_0)_{\sigma'}$$

with orientation $\mathcal{O}$. With this construction, as the path traverses and edge f_i to f_{i+1} for $i \in (1, m)$, $X_i^{\mathcal{O}}(f_1) = f_{i+1}$.

The set of edges E' is then the inverse image of E over the map ϕ_T. Restricting the map to edges not in E' then gives $\phi_{T|E'} : E' \to E$ is a one-to-one map.

A subpath U of $p_{\sigma'}$ can then be identified such that $U = (x_i f_{i+1} \dots x_{i+r-1})$ for some positive integer r where the edges are edges in T_G and are not inverse images of E over ϕ_T and thus $f_i, f_{i+r} \in E'$. Within this subpath, the vertices $x_i, \dots, x_{i+r-1}$ are vertices obtained in the perturbation of $v_i \in V$. Thus each of these r vertices, when lifted, will map to v. That is $\phi_T(x_i) = \phi_T(x_{i+r-1)} = v$. If v was a vertex of degree 3 in G then $U = x_i$ and $\phi_T(x_i) = v$.

The path $p_{\sigma'}$ is then equivalent to the concatenation of subpaths U_i which contain no edges in E', separated by edges in E'. Given that $U_j = (x_i f_{i+1} \dots x_{i+r-1})$, $f_i = f'_j$ and $f_{i+r} = f'_{j+1}$. Thus, $(x_i \dots x_{i+r-1})^{\mathcal{O}}(f'_j) = f'_{j+1}$ This gives that the perturbation map ϕ_T maps each element in the sequence to v. Given that ϕ_T maps f'_j and f_i to e_j and f'_{j+1} to e_{j+1}, the lift mapping gives that v maps e_j to e_{j+1} with respect to the orientation $\mathcal{O}$.

Because the lift results in this map, a boundary components $\sigma_j \in \delta(F_T(G))$ exists which traverses an edge f'_j, the vertex defined by the perturbation map $\phi_T(U_j)$ and then traverses f'_{j+1}. Recall if the vertex v is not perturbed then U_j is a single element. That is, the boundary component traverses p_{σ_j} containing the substring $(f'_j \phi_T(U_j) f'_{j+1})$ for each j. Because the boundary component travels with respect to the orientation $\mathcal{O}$, the boundary component σ_j and the boundary component which maps to $e_{j+1} v_{j+1} e_{j+2}$ must then connect, as they share a path traversing the edge e_{j+1}. Thus, all of the boundary components σ_j representing these subpaths must be pieces of the boundary components $\sigma \in \delta(F_T(G))$.

Applying the perturbation map ϕ_T to the elements of $\sigma' \in \delta(F(T_G))$, we see that the map is the equivalent of applying the map at each U and each separating edge, which map to the vertices and edges in the boundary component of $\sigma \in \delta(F_T(G))$

$$\phi_T(p_{\sigma'}) = (\phi_T(x_0)\phi_T(f_i)\dots\phi_T(x_0)$$

$$= (\phi_T(U_0)\phi_T(f_1')\dots\phi_T(U_{s-1})\phi_T(f_s')\phi_T(U_0))$$

$$= (v_0e_1\dots v_{s-1}e_sv_0) = p_\sigma.$$

Because the segments are distinct oriented paths with no repeating edges or vertices, the boundary component σ' uniquely determines σ. Further, the perturbation map for the edges f_j' is a bijection.

■

The lifting property that shows that an arbitrary boundary component in a thickened perturbed graph will exist in the thickened original graph. Jonoska et al. in [13] give the following corollary showing that a boundary component of $F(T_G)$ that traverses every edge of T_G at least once could then be lifted to a boundary component in $F_T(G)$ that traverses every edge of G at least once.

Corollary 5.3.3 *Let G be a connected multigraph, $F(T_G)$ be a thickened graph of a perturbed 3-degree graph T_G of G. If there is a boundary component $\sigma' \in \partial F(T_G)$ that traverses every edge of T_G, then there is a boundary component $\sigma \in \partial F_T(G)$ in the lift of $F(T_G)$ that traverses every edge of G.*

Proof. Consider the statement provided in Lemma 5.3.2. Let $G = (V, E, t)$ be a graph and T_G be it's 3-degree perturbation. The thickened graph $F(T_G)$ can be lifted to the thickened graph of G, $F_T(G)$. If a boundary component $\sigma' \in \delta(F(T_G)$ of the perturbed graph traverses every edge of the graph, then we can define the path of σ' as $p_{\sigma'} = (x_0f_1x_1\dots f_nx_0)$ such that $E_T = f_1,\dots,f_n$. That is, the edges traversed by $p_{\sigma'}$ $f_i, i = (1,n)$ are equal to the edge set for the perturbed graph T_G, thus the edges f_i map to E via the perturbation map ϕ_T.

Because every edge in T_G is traversed by σ', it follows that the map of $p_{\sigma'}$ is equal to p_σ. This gives that σ traverses $(v_0 e_1 v_1 \dots e_n v_0)$ where the elements $e_i, i = (1, n)$ make up the edge set of G, and thus σ traverses every edge in G.

$\blacksquare$

Given that a boundary component traversing every edge of the perturbed graph can be lifted to a boundary component traversing every edge in the original non-perturbed graph, the question of identifying an Eulerian walk in a graph is raised.

Chapter 6

The Existence of Reporter Strands

A reporter strand is as a boundary component which traverses each edge of a thickened graph at least once and no more than twice. Identifying a reporter strand within a thickened graph would allow for study of the structure of the graph through its topological properties. The first consideration is the case of identifying a reporter strand within a thickened graph with exactly two boundary components.

6.1 Thickened Graphs with Only Two Boundary Components

Lemma 6.1.1 *Let G be a 3-valent graph and $T(G)$ a thickened graph of G. If $T(G)$ has only two boundary components σ_1 and σ_2, then there is a thickened graph $\hat{T}(G)$ obtained from $T(G)$ by elementary boundary operations which has two boundary components $\hat{\sigma_1}$ and $\hat{\sigma_2}$ such that $\hat{\sigma_1}$ traverses every edge of G.*

Proof. We label the two boundary components red (σ_1) and blue (σ_2). For each vertex, there are three edges, each traversed twice by one or both of the boundary components. For each vertex, v, one of the following circumstances occurs with respect to the edges incident to the vertex (shown in Figure 6.1):

1. v is a red vertex. All edges are traversed by σ_1.

2. v is a blue vertex. All edges are traversed by σ_2.

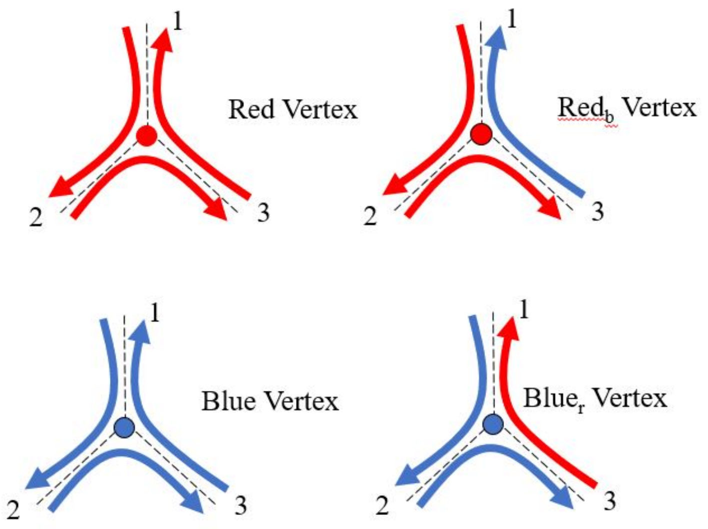

Figure 6.1: The four possible boundary component configurations at v given that only two boundary components exist.

3. v is a Red_b vertex. One edge incident to v is traversed exclusively by σ_1. The other two edges are traversed once by σ_1 and once by σ_2.

4. v is a $Blue_r$ vertex. One edge incident to v is traversed exclusively by σ_2. The other two edges are traversed once by σ_1 and once by σ_2.

Without loss of generality, if there are no vertices labeled as blue (blue or $blue_r$), then every edge in $T(G)$ is traversed at least once by the red boundary component σ_1. This implies that σ_1 is a strand traversing every edge and the lemma holds.

Otherwise, there must be at least one vertex that is $blue_r$. Let v be the $blue_r$ vertex, such that σ_1, the red boundary component, traverses two edges incident to v one time. Then σ_2, the blue boundary component traverses two edges incident to v once and traverses the third edge twice. Call this doubly traversed edge e. By the orientation of T_G, σ_2 must visit v, then exit v via e, visiting one or more additional vertices, then return to v via the second traversal of e. Let w be the second vertex incident to e. As e is incident to w, w is a blue or $blue_r$ vertex.

Performing a boundary operation at v will construct a new thickened graph

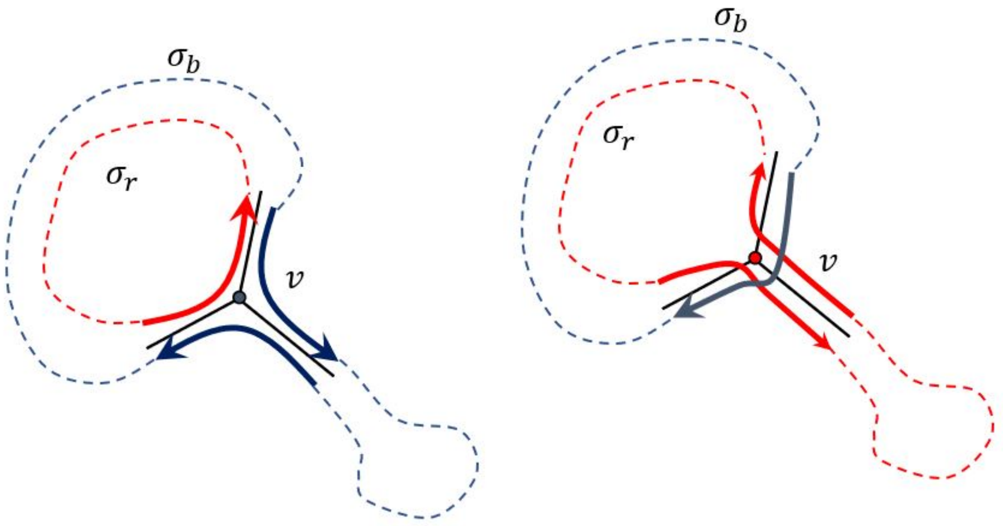

Figure 6.2: The vertex $v \in F(T_G)$ and the vertices $v \in F(T'_G)$.

$F(T'_G)$, which will have two new boundary components from σ_1 and σ_2 to $\hat{\sigma_1}$ and $\hat{\sigma_2}$ respectively. The connections at v are changed by this permutation, as shown in Figure 6.2.

This gives that if w were $blue_r$, then w is now red. Otherwise, w was blue and is now either red or red_b. The path between the first and second visit to w was previously traversed by σ_2 and will now be traversed by $\hat{\sigma_1}$. This operation creates no new boundary component and preserves the two prior boundary components, and thus $F(T'_G)$ has two boundary components. As all graphs considered are finite, a finite number of elementary boundary operations will ensure that each vertex is either *red* or red_b. Once each vertex is Red or Red_b, the graph is in the state of the initial case, and thus has a boundary component that traverses every edge.

$\blacksquare$

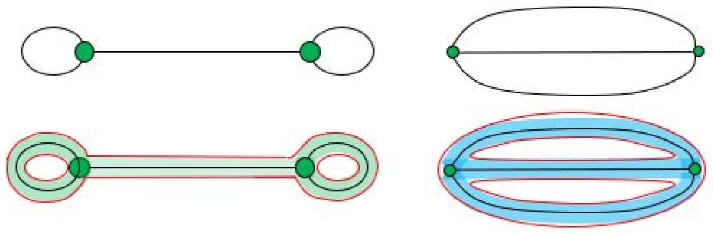

Figure 6.3: The two graphs of order 2.

6.2 Identifying a Reporter Strand in a Graph G

This result is further generalized in the following theorem given by Jonoska et al. in *On the existence of reporter strands in DNA-based graph structures*[13].

Theorem 6.2.1 *Given a connected 3-valent graph $G = (V, E, t)$, there is a thickened graph $F(G)$ and $\sigma \in \delta(F(G))$ such that $p_\sigma = (v_0 e_1 v_2 \dots v_n e_n v_0)$ and $\{e_1, \dots, e_n\} = E$*

Proof. We proceed by mathematical induction on the number of vertices in G.

Consider the basis case. There are two 3-valent graphs of order two, shown in Figure 6.3. The graph of three parallel edges has a single reporter strand following an elementary boundary operation at one of the two vertices. This reporter strand traverses every edge twice. The lollipop graph with two loops has three boundary components: two components traverse the interior of each loop and the third traverses the loops once and the single edge twice.

Assume that the theorem holds for all connected 3-valent graphs with $2n$ vertices. We show that the theorem holds for graphs with $2n + 2 = 2(n + 1)$ vertices. (Recall: a graph can not have an odd number of odd degreed vertices, thus the next case for a three-valent graph requires the addition of a pair of vertices). Let G be a graph of order $2n + 2$ and let v and w denote the two vertices. There are two general cases for the incorporation of v and w into the graph, given below:

Case 1. The graph G has two edges incident to both v and w.

Assume that there are two parallel edges between the v and w. There are then two possible cases of this scenario depending on how v and w interact with the remaining graph. In the first case, case 1a, v and w are adjacent to a single third vertex, x. This case is shown in Figure 6.4. In the second case, case 1b, v is adjacent to x and w is adjacent to y. The two subsets of this case are shown in Figures 6.5 and 6.6.

In both cases, it is necessary to ensure that the graph can be reduced to the graph of order $2n$, G', through the removal of v and w, with each vertex adjacent to v and w of degree three. In the case of 1a, v and w can be replaced with a loop at x. The non-loop edge incident to x would remain, and the including the loop, x would have degree 3, and thus G' meets the criteria of the hypothesis. In the case of 1b, v and w could be replaced by a single edge joining x and y. Without loss of generality, the two edges incident to x not incident to v remain, and the new edge adjacent to y creates the third incident edge, and thus x (and therefore y) is of degree 3.

In each case, we begin with G' and its reporter strand σ and show that the addition of the two vertices creates G and the extension to σ creates $\hat{\sigma}$.

Case 1a. G has a parallel edge and v, w are adjacent to a single vertex.

Consider the thickened graph of order $2n$, $T(G')$. By the inductive hypothesis, $T(G')$ has a reporter strand σ traversing every edge at least once and no more than twice. So each edge in $T(G')$ is traversed twice, either by a single boundary component or by two boundary components. The edge e must be traversed by σ at least one time. As boundary components must by cyclic and e is a bridge, e must be traversed by σ in both directions.

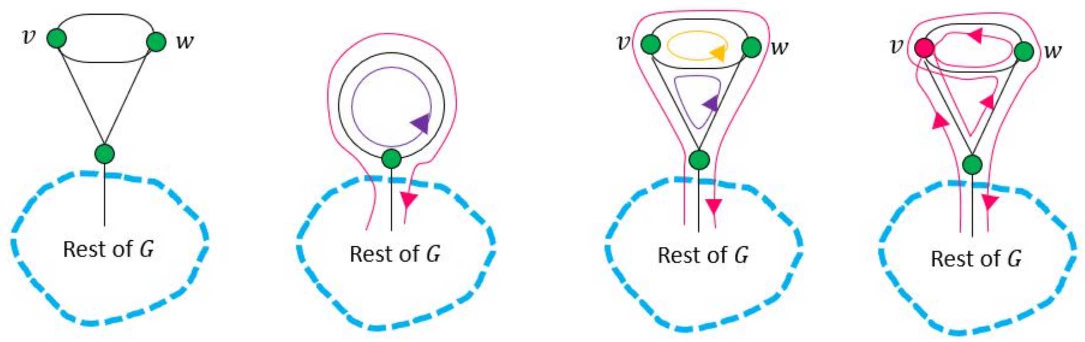

Figure 6.4: The graph G, the graph G' with its reporter strand σ in red, the extension of σ in G, and the creation of $\hat{\sigma}$ in red.

Because the orientation of a thickened graph must be preserved, the interior boundary of the loop can not be traversed by the same boundary component as the exterior of the loop. This gives that the boundary component that traverses e twice and the exterior of the loop must be σ.

In order to construct G, we expand the loop and add two vertices v and w, then add an additional edge between v and w. The resulting graph, shown in the third image in Figure 6.4, creates the boundary component $\hat{\sigma}$ traversing the exterior and the boundary component from the inside of the loop and additionally adds a boundary component between the two parallel edges.

Without loss of generalization, consider v. As three boundary components visit v, an elementary boundary operation at v will result in the three boundary components becoming a single boundary component due to Lemma 4.2.1. This new boundary component is shown on the right of Figure 6.4.

By the inductive hypothesis, $\hat{\sigma}$ also traverses the remaining edges in the rest of

G, and thus $\hat{\sigma}$ is a reporter strand traversing every edge at least once.

Case 1b. G has a parallel edge and v, w are each adjacent to a separate vertex

Consider the thickened graph, $T(G')$. Let f be the edge incident to x and y. Removing the edge f reduces x and y to degree two. This allows adding an edge to x incident to v and an edge to y incident to w. Adding parallel edges between v and w creates a 3-valent graph.

By the inductive hypothesis, there is a reporter strand σ in G' that traverses every edge at least once and no more than twice. There are two possible cases for the placement of $\hat{\sigma}$, shown in Figures 6.5 and 6.6.

Consider the case shown in Figure 6.5, where σ traverses f once. When x and y are added to G', $\hat{\sigma}$ traverses the exterior of the path from x to y via the new vertices. Two additional boundary components are present, one traversing the interior portion and some of the rest of G and one inside the two parallel edges between v and w. Without loss of generality, consider v. As with case $1a$, three boundary components visit v. Thus, a boundary operation at v will result in a single boundary component. As one of the boundary components visiting v traversed the remaining edges of the graph at least once, there exists a $\hat{\sigma}$ traversing every edge at least once.

Otherwise, σ must traverse f twice. Going from G' to G, the boundary component traversing f will then traverse each new edge twice, as shown in Figure 6.6. As the boundary component in G' traversed every edge at least once, $\hat{\sigma}$ traverses every edge in $T(G)$ at least once.

Case 2. The graph G has no parallel edges

Consider the graph G' of order $2n$. A graph G of order $2n + 2$ could be con-

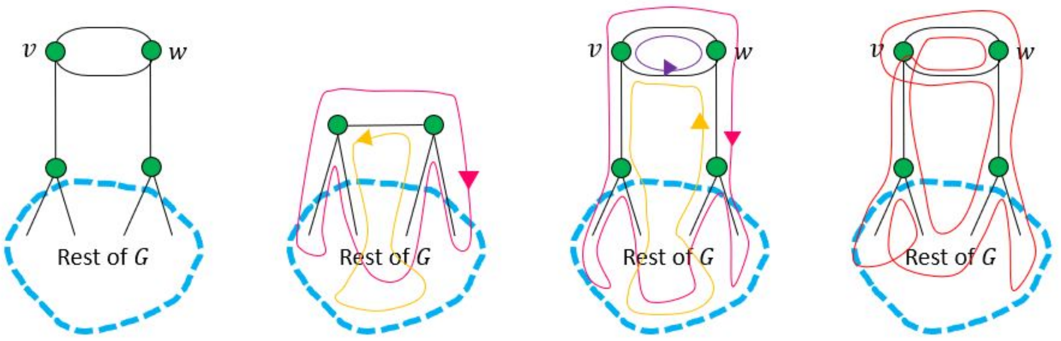

Figure 6.5: The graph G with parallel edges, the graph G' with its reporter strand σ in red, the extension of σ in G, and the reporter strand $\hat{\sigma}$ of G in red.

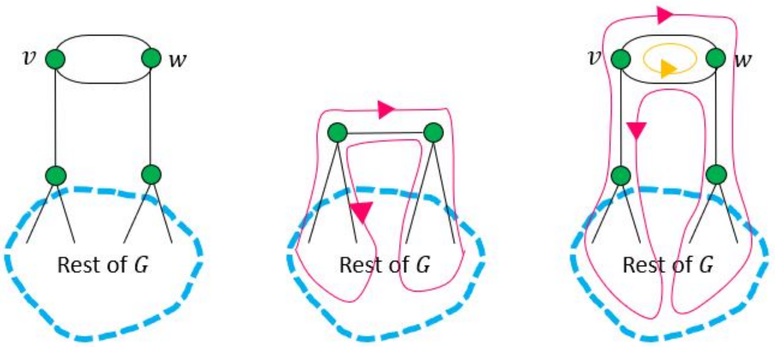

Figure 6.6: The graph G with parallel edges, the graph G' with its reporter strand σ in red, and the extension of σ to $\hat{\sigma}$ in red.

structed by adding the vertex v on the edge $f_v \in G'$ creating e_v and $e_{v'}$. Similarly, the vertex w on the edge $f_w \in G'$, creating e_w and $e_{w'}$. To construct G as a 3-valent graph, we add an edge e between v and w.

The thickened graph $T(G')$ has up to four boundary components traversing f_v and f_w. Denote these boundary components $\sigma_1, \sigma_2, \sigma_3$, and σ_4, with σ_1 and σ_2 traversing f_w in opposite orientations and σ_3 and σ_4 traversing f_v in opposite directions. Constructing $T(G)$ with the new vertices v and w will change these boundary components to $\hat{\sigma_1}, \hat{\sigma_2}, \hat{\sigma_3}$ and $\hat{\sigma_4}$, with $\hat{\sigma_1}$ visiting w and traversing e_w and e_w' and similarly $\hat{\sigma_4}$ visiting v and traversing e_v and e_v'. $\hat{\sigma_2}$ will now traverse e_w, visit w, then

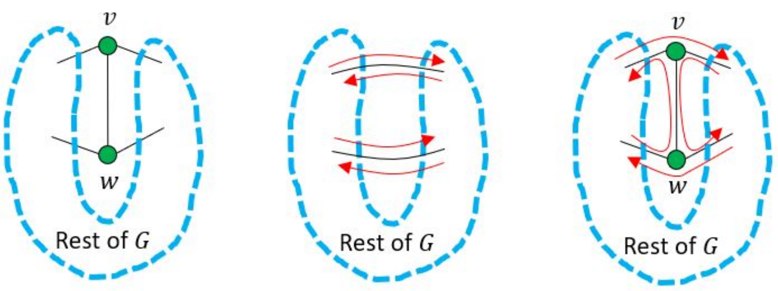

Figure 6.7: The graph G, G' with its reporter strand σ in red, and the extension of σ to $\hat{\sigma}$ in G.

traverse e, visit v, and traverse e_v. $\hat{\sigma_3}$ will now traverse e'_v, visit v, then traverse e, visit w, and traverse e'_w. Note that $\hat{\sigma_2}$ and $\hat{\sigma_3}$ each traverse e in opposite orientation along the edge ribbon which deforms to e.

By the inductive hypothesis, $T(G')$ has a boundary component σ that traverses f_v and f_w at least once and no more than twice. Thus, at least two of $\hat{\sigma_1}, \hat{\sigma_2}, \hat{\sigma_3}$ and $\hat{\sigma_4}$ are the same boundary component. Consider the following subcases:

(a) σ traverses both edges f_v and f_w once.

(b) σ traverses one of f_w, f_v twice and the other once.

(c) σ traverses both f_w and f_v twice.

We consider each of these three subcases below.

Case 2a. σ traverses f_v and f_w each once.

This case is illustrated in Figure 6.8.

In the case that σ traverses each edge f_w and f_v exactly once, then two of $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are both a part of σ. Without loss of generality, assume that σ_1 and σ_4 are the part of σ. The two remaining boundary components could either be a singular boundary component $\delta \neq \sigma$ or part of two distinct boundary components

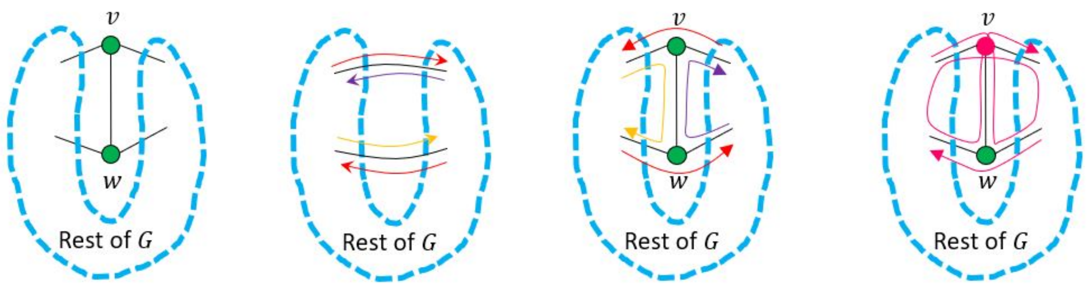

Figure 6.8: If σ traverses the edges of G' exactly once, then the extension to G constructed by including v and w and the edge incident to both will have 3 boundary components. A boundary operation at v will result in the reporter strand $\hat{\sigma}$ shown in red.

δ and ξ, both of which are distinct from σ.

Assume that δ is the boundary component traversing both of the opposite orientation boundaries of the edge ribbons of f_v and f_w in $T(G')$, as shown on the top left of Figure 6.8. The inclusion of v and w to create $T(G)$ will maintain σ's traversal of the outer edges of the ribbons created by e_v, e'_v, e_w and e'_w creating $\hat{\sigma}$. The boundary component δ will be split by e in $T(G)$, creating $\hat{\delta}$ and $\hat{\xi}$. The boundary component δ will traverse e'_w, visit w, traverse e, visit v and then traverse e'_v, then traverse some path in G and return to e'_w. The boundary component ξ will then traverse the non-prime edges and e in a similar manner. This structure results in three boundary components visiting the vertices v and w, $\hat{\sigma}, \hat{\delta}$ and $\hat{\xi}$. Without loss of generality, consider an elementary boundary operation at v. As three boundary components visit v, this boundary operation will result in a unification of the three components. Because one of the components that is unified is $\hat{\sigma}$, this resulting boundary component is a reporter strand, traversing every edge at least once.

Otherwise, $F(T_{G'})$ is traversed locally by three boundary components: σ, which is a reporter strand traversing each edge of G at least once, δ, which traverses f_v in the opposite orientation of σ, and ξ which traverses f_w in the opposite orientation of σ. In this case, an elementary boundary operation at v will produce the edge ribbon associated with e in $F(T_G)$ such that δ remains, but xi and σ are unified to form $\hat{\sigma}$. Thus $\hat{\sigma}$ traverses every edge in G at least once.

Case 2b. σ traverses one of f_v and f_w once and the other twice.

Without loss of generality, assume σ traverses f_V once and f_w twice. Then in $T(G')$, σ_1, σ_3, and σ_4 all belong to the same boundary component σ and σ_2 belongs to another component.

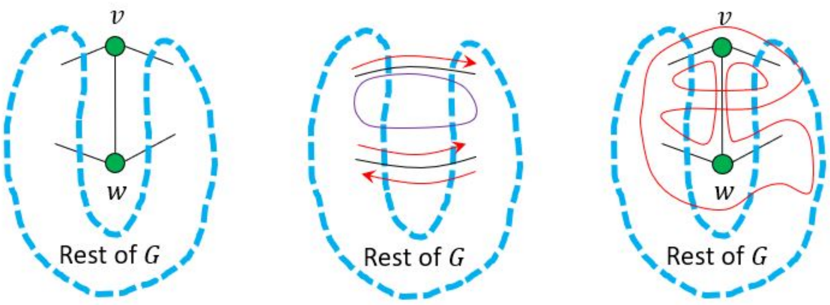

Figure 6.9: If σ traverses one edge in G' twice and the other once, then the resulting extension to G will have a boundary component that traverses every edge incident to v or w twice.

Adding vertices v and w and the edge e to construct G results in one unified $\hat{\sigma}$ as shown on the left in Figure 6.9. Thus, $\hat{\sigma}$ traverses every edge in $T(G)$ at least once.

Case 2c. σ traverses both f_v and f_w twice.

If both of the edges f_v and f_w in $T(G')$ are traversed by the same boundary component σ, then $\sigma_1, \sigma_2, \sigma_3$, and σ_4 all belong to σ.

In order to construct $T(G)$ from $T(G')$, we add the vertices v and w along the edges f_v and f_w and the edge e joining v and w. The boundary component $\hat{\sigma}$ now traverses each exterior edge ribbon, and then doubly traverses, without loss of generality, e'_w, e, e'_v. A new boundary component, δ is formed traversing the remaining local edges.

Two scenarios must be considered. If every edge that δ traverses is traversed exactly once, then $\hat{\sigma}$ is a reporter strand.

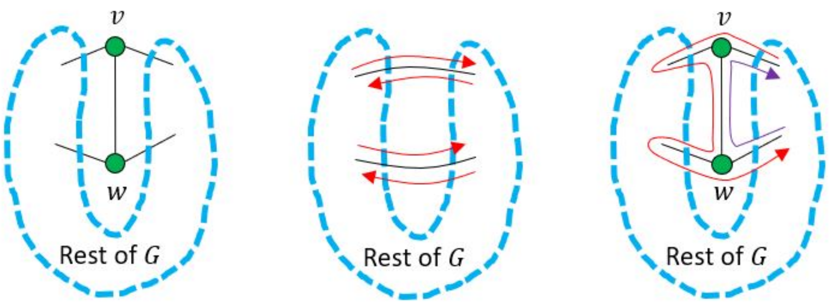

Figure 6.10: The graph G' with σ traversing each edge twice and the resulting graph G with σ extended to $\hat{\sigma}$.

If $T(G)$ contains two boundary components, then by Lemma 6.1.1, $\hat{\sigma}$ is a boundary component traversing every edge at least once.

Otherwise, suppose $T(G)$ has at least 3 boundary components, two of which area $\hat{\sigma}$ and δ. By the inductive hypothesis, every edge is traversed at least once by one of $\hat{\sigma}$ or δ.

In the case that a vertex v in $T(G)$ is visited by each of the three boundary components, then an elementary boundary operation at v will unify the three boundary components. This gives a boundary component that traverses every edge at least once within $T(G)$.

Otherwise, δ traverses some number of edges twice. Using the same process as in Theorem 6.1.1, an elementary boundary operation can reduce these occurrences. Each edge that δ traverses is either traversed twice by δ or once by δ and once by $\hat{\sigma}$. Traversing δ and performing an EBO every time a doubly traversed edge is encountered ensures that when no doubly traversed edges exist, $\hat{\sigma}$ traverses every edge of G at least one time.

Thus, in any case of two additional vertices, $\exists \sigma \in \delta(F(G))$ such that σ traverses every edge in G.

$\blacksquare$

In combination with the prior chapters, this result gives that for each connected multigraph, elementary boundary operations can produce a thickened graph with a reporter strand traversing each edge at least once.

Chapter 7

Relating Postman Tours to Reporter Strands

By definition every reporter strand is also a postman tour, since a reporter strand traverses every edge at least once and no more than twice. Several other questions, however, are not as obvious and are worth investigating. One of them is the converse of this statement. It can be shown through a counterexample that not every postman tour is a reporter strand. This leads to investigating which types of postman tours are and which types are not reporter strands. A second question is that of "containment" and our main result addresses that. We show that every postman tour is "contained" in a reporter strand. Third, it will be interesting to know, whether given a reporter strand, which we know is a postman tour, it is possible to "reduce" it to obtain a minimal postman tour.

7.1 Classifying Postman Tours by Reporter Strands

Since every reporter strand is a postman tour, obviously, at least some postman tours are reporter strands. It turns out that not all postman tours are reporter strands. The following counterexample shows that there exist postman tours that are not a single boundary component, i.e. a reporter strand.

The minimal order 3-valent graph is the basis case in the proof for Theorem 6.2.1 presents such a counterexample. For these two graphs of order two, it is easily observed that in each case, there are postman tours that cannot be represented as a reporter strand. In the case of the lollipops graph, the maximal postman tour traversing every edge twice is not a reporter strand. In the case of the three parallel edges joining the two vertices, the only reporter strand is the maximal reporter strand. Thus, the postman tour which traverses exactly one edge twice is not a reporter strand.

Next, we investigate what prevents a postman tour to be a reporter strand. We proceed by considering simple graphs, i.e. graphs which do not contain multiedges or loops. The minimal order 3-valent simple graph is K_4. For K_4, there are three possible types of postman tours: 1) a tour that traverses exactly two edges twice, 2) a tour that traverses exactly three edges twice, and 3) the tour that traverses every edge twice. The parity of $|\delta(K_4)|$ is even, and thus a singular boundary component traversing every edge twice can not exist. However, by the sharp bound defined in 2.2, the maximal postman tour is never the tour of minimal length, and thus, the maximal postman tour cannot not be an optimal path.

As shown in Figure 7.1, performing an elementary boundary operation on one vertex or on three vertices of K_4 results in a reporter strand traversing exactly two edges twice. Performing an elementary boundary operation on two vertices gives a reporter strand traversing exactly two edges twice. The thickened graph constructed by performing a boundary operation on every vertex of K_4 results in a thickened graph with no reporter strand.

For the case of K_4, every non-maximal postman tour exists as a reporter strand. We conjecture that for every postman tour τ of a graph G, there exists a thickened graph $F(G)$ with a reporter strand $\sigma \in \delta(F(G))$ that exactly represents τ.

Consider the case in Figure **??**. Following the boundary for the parity of the graph described in 2.2, there exists a postman tour τ of minimal length constructed by adding $\frac{n}{2}$ edges, specifically between pairs of odd vertices. The magenta lines

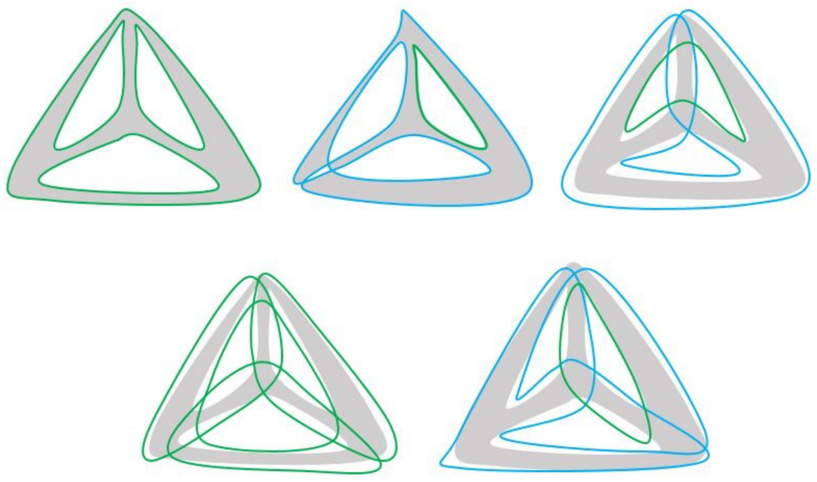

Figure 7.1: (Clockwise from top left) K_4, K_4 after 1 boundary operation, K_4 after 2 boundary operations, K_4 after 3 boundary operations, and K_4 after 4 boundary operations.

in Figure 7.2 creates a semi-Eulerian graph with one such minimal length tour. However, if this were to be represented as a thickened graph, there are 8 remaining edges that need to be incorporated as part of a boundary component. The middle subcycle of 4 edges can not be connected to the outer subcycle of four edges as each of the vertices connecting the two are traversed twice in τ. These two cycles must then each be part of separate boundary components. However, the thickened graph T_G has six boundary components, and thus any thickened graph constructed through boundary operations on T_G has an even number of boundary components. Because the length of each remaining cycle is 4, there are no subcycles, and thus there must be exactly two additional boundary components to create this structure. This thickened graph has 3 boundary components and thus could not be constructed through self assembly.

This issue in a thickened graph $F(T_G)$ can be directly related to the behavior of paths in the original graph G. In order to construct the paths traversing only the

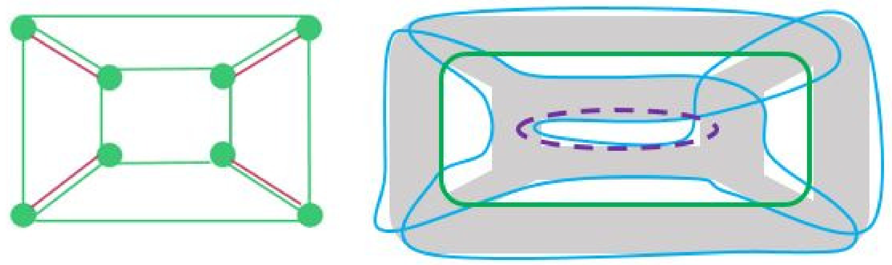

Figure 7.2: The graph G has a postman tour τ traversing the 4 maroon edges twice. The thickened graph $F(G)$ contains a reporter strand σ in blue containing the postman tour τ and the additional loop in purple.

maroon edges of the graph in Figure 7.2, the path has to traverse an edge only to visit a vertex and immediately traverse the same edge again in the opposite direction.

Definition 7.1.1 The edge (v, u) is called the reverse of (u, v). If a path includes two consecutive reverse edges (u, v) and (v, u), i.e. the path includes u, v, u then v is called *a point of reverse*. Two reverse arcs are called separated if there is a loop l between them (which starts at the head of one of the edges and ends at the tail of the other).

If the loop l which separates two reverse edges, (u, v) and (v, u), is removed, then the result is a point of reverse at v.

When considering the examples for the graphs of order 6 and order 8, superfluous sub-cycles occur when the traversed path contains a point of reverse. This structure can not be constructed in DNA as this would form a hairpin in the thickened graph, which would be topologically equivalent to the vertex and it's incident edges being removed.

7.2 Containment in Reporter Strands

While we identified postman tours that can and can not exist as reporter strands, here we consider whether some of the postman tours that could not exist as a boundary component within a surface. For each 3-valent graph, a thickened graph can be identified such that for every non-maximal postman tour τ', a boundary component of the thickened graph contains all the edges of τ and possibly additional edges. As shown in Figure 7.2, the blue boundary component is a reporter strand, traversing every edge in τ and additionally traversing one of the subcycles, resolving the parity issue discussed previously. This additional traversal of the subcycle, circled in purple allows for σ to be a reporter strand while containing each edge in τ.

Definition 7.2.1 A reporter strand σ is said to *contain* a postman tour τ if $\sigma = \tau$ or if every edge in τ is contained in $p(\sigma)$.

For instance, in the case of Figure 7.2, the graph on the right has a blue boundary component which *contains* the postman tour τ which traverses the four edges twice. However, this boundary component is not strictly equal to τ. The boundary component includes edges not traversed by τ. We use this definition of contains to state the following theorem:

Theorem 7.2.2 *For every non-maximal postman tour τ in a connected 3-valent graph G, there exists a $\sigma \in \delta(T_G)$ with the edges of τ embedded as a subgraph G'.*

Proof.

We proceed by induction on the number of vertices in G.

The minimal order for a 3-valent graph is 2. There are two non-isomorphic 3-valent graphs of order 2: the "lollipop" graph with one edge connecting the two vertices and a loop at each vertex and the graph with three parallel edges connecting the two vertices, shown in Figure 6.3.

First, consider the case of the lollipop graph. Because two boundary components visit a vertex in each case, any elementary boundary operation will result in the same number of boundary components. Thus the maximal postman tour can not be constructed. However, reporter strands exist traversing the edge connecting the two vertices twice, and traversing either the interior or the exterior of each loop. Performing no elementary boundary operation, the path will traverse the outside of both loop. Performing one boundary operation, the path will traverse the outside of one loop and the inside of the second loop. Finally, two boundary operations will result in the path traversing the inside of each loop. No other possible combinations can exist, and thus every non-maximal τ has an equivalent σ.

In the case of the order two graph with three parallel edges, the only reporter strand that can be constructed is the maximal boundary component, traversing every edge. Thus, the edges of every τ are contained in σ.

Assume that for every non-maximal postman tour of a graph G' of order $2n$, there exists a thickened graph such that $\exists \sigma \in \delta(T_{G'})$ that contains the edges of τ. Consider G a 3-valent graph of order $2n+2$ constructed by adding two vertices to G'. The possible cases for the addition of two new vertices, v and w, to construct G are the same as the cases in Theorem 6.2.1. From Theorem 6.2.1, it is shown that in each case, a $\hat{\sigma}$ can be constructed such that $\hat{\sigma}$ extends to the rest of the graph beyond the local area around v and w. We proceed by showing that the edges included in any traversal of the neighborhood of v and w by a postman tour τ are included in some $\hat{\sigma}$.

Case 1. The graph G has parallel edges.

Within this case, there are two subcases based on how v and w related to the vertices of G'.

Subcase 1. v and w are adjacent to a single vertex in G'.

As the edge $e \in G'$ is a bridge the component containing v and w, any postman tour that traverses e must traverse e in each direction. Then τ' will traverse the loop in one orientation or the other. Adding v and w to the loop and creating a parallel edge between them will result in the structure shown in Figure 6.4. As shown in the proof of Theorem 6.2.1, the reporter strand traversing G will traverse each edge local to w and v twice, and thus edges contained in τ are contained in $p(\hat{\sigma})$.

Subcase 2. v and w are adjacent to two distinct vertices x and y in G'

Consider a postman tour τ that traverses the the edges of the structure created when adding two parallel edges to G'. The postman tour τ must follow one of the following paths:

(a) $p(\tau)$ traverses every edge twice.

(b) $p(\tau)$ traverses the edge incident to v and x and the edge incident to w and y once and one of the parallel edges twice.

(c) $p(\tau)$ traverses the edge incident to v and x and the edge incident to w and y each twice and traverses each parallel edge once.

In the cases of (a) and (c), the resulting reporter strands from the proof of Theorem 6.2.1 directly match τ and all of the edges of (b) are contained within the strand duplicating each edge twice. Thus for the case of two parallel edges, all possible postman tours are contained within some reporter strand $\hat{\sigma}$.

Case 2. The graph G does not have parallel edges.

As with case one, this case contains a significant number of possible τ. Consider the graph structure on the left of Figure 6.7.

(a) $p(\tau)$ traverses every edge twice.

(b) Without loss of generality, $p(\tau)$ traverses e_v and e'_w twice and all other edges once.

(c) $e_v vewe_w$ and $e'_w weve'_v$ are subpaths contained in $p(\tau)$.

Cases (a) and (c) are contained in the path traversing every edge twice created in the second subcase (where σ traverses one of f_w, f_v once and the other twice in G'). Otherwise, the postman tour τ is contained in the boundary component $\hat{\sigma}$ created from σ traversing each edge twice in the G'.

As each possible postman tour τ of G in the cases and subcases described herein can be contained in a one of the $\hat{\sigma}$ that exist, we conclude that for every τ of a graph G, $\exists \sigma \in \delta(F(T_G))$ such that $p(\sigma)$ contains all the edges of τ.

■

The proof above shows that for every 3-valent (multi)graph G, every postman tour is contained within a reporter strand of G. However, in order to provide a solution to postman tour problems, it is necessary to identify the exact postman tours, and thus the superfluous traversals must be removed.

7.3 A Language Theoretic Approach to Removing Edges

In order to remove edges, we consider two major possibilities: the removal of edges through chemical processing (enzymatic cutting once criteria is met, etc.) and the removal of edges through a language theoretic algorithm. Here, we focus on the language theoretic approach. A complete solution to removing edges from the reporter strands created is part of our future work. Provided here is an initial approach to algorithmically remove unnecessary edges using subword patterns. This approach focuses on identifying patterns within the sequence generated by reporter strand whose path has been labeled properly following assembly. Additionally, we consider heuristics that allow for fewer final paths to be analyzed.

The labeled results of reporter strands that are removed from the solution and read. This initial step of removal and reading through electron microscopy or gel electrophoresis poses delay in obtaining a solution. Once a strand has been removed, the path can be defined as a word w. Subwords contained within w can be identified as subsequences beginning and terminating with the same vertex v. Loops that are possibly unnecessary can be isolated by considering the number of times the vertex is repeated in w.

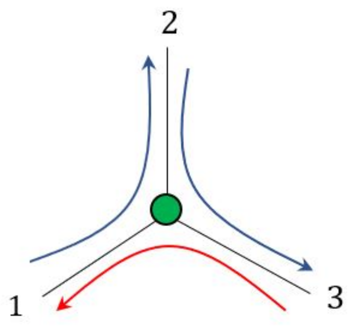

Figure 7.3: A reporter strand σ traversing every edge in the neighborhood of v twice with a superfluous loop (colored in red).

Consider Figure 7.3. If v is traversed by a boundary component with the red superfluous loop traveling $3 \to 1$, then the boundary component must travel $1 \to 2$ and $2 \to 3$ or the boundary component will no longer be a reporter strand. Thus we can conclude that any superfluous loop can only occur at a vertex which appears in w three times.

For a vertex v that appears in w three times, call the first visit v_1, the second visit v_2 and the third visit v_3. There are then three possible loops that can be removed: the subword between v_1 and v_2, the subword between v_2 and v_3 and the subword consisting of the portion of the word before v_1 and the portion of the word after v_3. Each removed subloop can be no longer than $\frac{n}{2}$, as this would not meet the criteria to create a semi-Eulerian graph.

Once each possible superfluous loop is identified, the subwords can be iteratively removed to create w' and the following conditions are checked for w':

1. The path is at least $\frac{3n}{2}$ long.
2. Each edge is represented at least once in the sequence.

If these conditions are met, then w' represents a postman tour. If these conditions are not met, then the removed subword is necessary to the integrity of the postman tour.

We believe that the further refinement of this language theoretic approach could create a solution to postman tour style problems.

Chapter 8

Conclusion

In this work we propose to study the connection between postman tours and reporter strands. Postman tour problems model a wide range of phenomena, and hence are very applicable, but many variants of them have a high computational complexity. Finding a postman tour in a graph is solvable in polynomial time, but the weighted variants of this problem, including the Windy Postman Problem, have been proven to be NP-hard.

We investigated known results about thickened graphs and boundary components. Each graph G can be perturbed to create a 3-valent graph T_G. Once these perturbed graphs are thickened, the resulting structure can be lifted to represent the thickened original graph. The thickened graphs of T_G, denoted $F(T_G)$ are not distinct. Through elementary boundary operations at one or more vertices of the thickened graph various thickened graphs with different number of boundary components can be constructed. One of the main results in this line of research is that for a given graph G, there exists a reporter stand, a single boundary component that traverses each edge, in at least one of the thickened graphs $F(T_G)$. We discuss these results in detail and show how we use them to prove our main result.

The main result of this paper states that for every postman tour there exists a single molecule, i.e. a reporter strand (a single boundary component), containing all edges of the tour, which can self-assemble in a DNA graph. Thus, the existence of a reporter strand that contains each postman tour makes it possible to study and analyze postman tours through reporter strands. Since the Windy Postman Problem is NP-complete, there are no known efficient algorithms for finding a minimal weight postman tour. To that effect, we propose to study more efficient ways to finding postman tours through considering all reporter strands and "extracting" from them minimal postman tours.

Our study of postman tours through reporter strands points to two new directions of future research. One direction is to develop polynomial time algorithms to solve this set of related problems using DNA self-assembly and the other direction is to use language theoretic tools to obtain postman tour solutions from reporter strands.

As with other DNA computing models, the first direction is to show how to solve different variants of the NP-complete postman tour problems efficiently by employing an enzymatic approach and making use of the massive parallelism property of DNA strands. The goal in this approach will be to design and study the optimal (not only in terms of polynomial number of steps, but also in terms of wet computation and lab protocols) algorithms using DNA self-assembly.

The second direction of our current and future research is to study graph theoretic problems, such as postman tour problems, by transforming them to language theoretic ones. The existence of reporter strands, which are strands of DNA nucleotides that can ultimately be represented as words over the DNA alphabet makes it possible to abstract the postman tour problems to languages.

We are currently studying language theoretic approaches which produce solutions to postman tour type problems. A reporter strand can be viewed as a word and a postman tour consists of some or all of its subwords. Thus, the problem of finding an optimal postman tour reduces to finding certain subwords of words.

References

[1] L. Adleman, *Molecular computation of solutions to combinatorial problems*, Science 226 (1994) 1021-1024.

[2] B. Bollobas, *Graph theory*, Springer-Verlag, New York, 1979.

[3] A. Corberan, G. Mejia, J.M. Sanchis, *New Results on the Mixed General Routing Problem*, Mathematical Programming 26 (2005) 363-376.

[4] A. Corberan, M. Oswald, I. Plana, G. Reinelt, J.Sanchis, *New results on the Windy Postman Problem*, Mathematical Programming Ser. A 132 (2012) 309-332.

[5] J.L. Gross, J. Yellen, *Handbook of Graph Theory*, CRC Press, Boca Raton, 2004.

[6] M. Guan, *On The Windy Postman Problem*, Discrete Applied Mathematics 9 (1984) 41-46.

[7] T. Head, *Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors*, Bulletin of Mathematical Biology 49-6 (1987) 737-759.

[8] J.E. Hopcroft, R. Motwani, J. Ullman, *Automata Theory, Languages, and Computation*, 3rd edition, Pearson Addison Wesley, 2007.

[9] S. Hussini, L. Kari, S. Konstantinidis, *Coding properties of DNA languages*, Theoretical Computer Science 290-3 (2003) 1557-1579.

[10] N. Jonoska, S.A. Karl, M. Saito, *Three dimensional DNA structures in computing*, BioSystems 51 (1999) 143-153.

[11] N. Jonoska, P. Sa-Ardyen, N. C. Seeman, *Computation by Self-assembly of DNA graphs*, Genetic Programming and Evolvable Machines 4 (2003) 123-137.

[12] N. Jonoska, M. Saito, *Boundary components of thickened graphs*, Lecture Notes in Computer Science 2340 (2002) 70-81.

[13] N. Jonoska, N.C. Seeman, G. Wu, *On the existence of reporter strands in DNA-based graph structures*, Theoretical Computer Science 410 (2009) 1448-1460.

[14] A.V. Kostochka, N. Tulai, *On the length of the path of a Chinese postman in homogenous graphs*, Sibirsk Zh. Issled. Oper. 1-3 (1994) 20-37.

[15] J. S. Lee, S. Shin, T. Park, B. Zhang, *Temperature gradient-based DNA computing for graph problems with weighted edges*, Lecture Notes in Computer Science 2568 (2003) 73-84.

[16] R. Lipton, *DNA solution of hard computational problems*, Science 268 (1995) 542-545.

[17] E. Minieka, *The Chinese potman problem for mixed networks*, Management Science 25 (1979) 643-648.

[18] J.R. Munkres, *Topology, A First Course*, Prentice-Hall, 1975.

[19] A. Naryanana, S. Zorbalas, *DNA algorithms for computing shortest paths*, Proceedings of Genetic Programming (1998) 718-723.

[20] S. O, D.B. West, *Sharp bounds for the Chinese Postman Problem in 3-regular graphs and multigraphs*, Discrete Applied Mathematics 190-191 (2015) 163-168.

[21] G. Paun, G. Rozenberg, A. Salomaa, *DNA Computing, New Computing Paradigms*, Springer-Verlag Berlin Heidelberg, 1998.

[22] K.H. Rosen, *Discrete Mathematics and Its Applications*, 7th edition, McGraw-Hill 2012.

[23] H. Royden, P. Fitzpatrick, *Real Analysis*, Pearson, 2010.

[24] S. Shin, B. Zhang, S. S. Jun, *Solving traveling salesman problems using molecular programming*, Proceedings of Congress on Evolutionary Computation (1999) 994-1000.

[25] Z. Win, *On the Windy Postman Problem on Eulerian Graphs*, Mathematical Programming 44 (1989) 97-112.

[26] G. Wu, N. Jonoska, N.C. Seeman, *Construction of DNA nano-object directly demonstrates computation*, Biosystems 98 (2009) 80-84.

[27] M. Yamamura, Y. Hiroto, T. Matoba, *Solutions of shortest path problems by concentration control*, Lecture Notes in Computer Science 2340 (2001) 231-240.

About the Author

Katie Bakewell graduated with dual bachelor's degrees in Mathematics and Statistics from the University of North Florida in 2013. During her undergraduate studies, she worked with Dr. Daniela Genova on an analysis of Hamiltonian Paths via the bridges of Jacksonville. This work was presented at the 2012 Florida Mathematical Association of America meeting.

During her graduate studies, Katie focused on three areas of research. Her main area of research was the topic of this thesis which she studied with Dr. Genova and another former graduate student Sudam Surasinghe. The preliminary draft of this work was presented at the 45th Southeastern International Conference on Combinatorics, Graph Theory, and Computing at Florida Atlantic University in February 2016 and at the 21st International Conference on DNA Computing and Molecular Programming at Harvard University's Wyss Institute in August 2016. Dr. Genova and Katie are also working with another graduate student Benjamin Webster on constructing a multivariate geometric distribution using automata and language theory. This work was presented at the 2016 Florida Mathematical Association of America meeting.

Additionally, Katie presented research in forecasting commodities prices using exogenous regressors at the Conference on Statistical Practice in San Diego. This presentation was based on research completed with Dr. Pali Sen. Katie is extremely grateful to have been able to attend these conferences due to travel grants from the UNF Graduate School, College of Arts and Sciences, and Department of Mathematics and Statistics.

Katie is a lead statistician at NLP Logix, a local predictive modeling company.

She has been with NLP Logix for five years. With NLP Logix, she co-presented statistical research with the Florida Poison Information Center Network at the 2014 and 2017 North American Congress of Clincal Toxicologist conferences and co-organizes Big Data Jax.

Katie lives in Jacksonville, FL with her husband, Sagara Lebunu-Hewage.