A Comparison of Vertex-Splitting and Spider-Splitting for the Study of Three-Dimensional Rigidity

by

Jason N. d'Eon

A Thesis Submitted to
Saint Mary's University, Halifax, Nova Scotia
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science (Honours)

April 2018, Halifax, Nova Scotia

Approved: Dr. Wendy Finbow-Singh
Supervisor

Approved: Dr. Bert Hartnell
Reader

Date: March 29, 2018

A Comparison of Vertex-Splitting and Spider-Splitting for the Study of Three-Dimensional Rigidity

Jason N. d'Eon

April, 2018

Abstract

In two dimensions, generic rigidity is a combinatorial property of a framework, but extensions into three dimensions fail to completely characterize generic rigidity. It is therefore interesting to investigate two graph operations introduced by Walter Whiteley, vertex-splitting and spider-splitting, which are known to take a minimally rigid framework in three dimensions to a new minimally rigid framework with an additional vertex. We present algorithms for generating all possible graphs obtained by vertex-splitting, spider-splitting, and combinations of vertex-splitting and spider-splitting. For graphs with up to and including 8 vertices, the set of graphs obtained by spider-splitting is a subset of the set obtained by vertex-splitting. Additionally, the set produced by combinations of vertex-splitting and spider-splitting is equal to the set obtained by vertex-splitting. This suggests that as a method for generating rigid graphs, spider-splitting is inferior to vertex-splitting at all steps of iteration.

Acknowledgements

First, I would like to thank my supervisor, Dr. Wendy Finbow-Singh, for her indispensable guidance. Without her, this thesis would not have been possible. I also want to acknowledge the Mathematics and Computing Science faculty at Saint Mary's for furthering my mathematical maturity.

I would like to mention Jack Graver for his excellent book, "Counting on Frameworks", that served me well as an introduction into rigidity theory, as well as Walter Whiteley at York University for his work on vertex-splitting and spider-splitting for which this thesis revolves around.

Finally, I want to thank my colleagues, Justin and Will, for keeping me enthusiastic about math when things were stressful; my brother Greg, who always pushes me to work hard; and my parents who have continuously supported me in everything I do.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Rigidity theory is an area of discrete geometry concerned with understanding what makes objects sturdy or flexible. It has significant applications, for example, in understanding how a building will behave when subject to forces such as wind and earthquakes. A common approach to determining if an object is rigid is to represent it as a graph that captures the connections between vertices of the object, but without any sense of the position of vertices or distance between them. Part of this theory is to come up with theorems that allow an object's rigidity to be predicted from its structure-graph. One would hope that rigidity depends only on the way vertices are connected and is independent of the positioning of vertices and edge-length. In reality, there are shapes whose rigidity changes based on how they are embedded in two dimensions or three dimensions. Figure 1.1 shows an example of two embeddings of the same structure-graph. The chain of vertices along the top are held taut in the left graph, but are able to move in the right graph. A randomly chosen embedding of this graph is not likely to have these vertices lined up and will most likely not be rigid. Later we will see that the embedding on the left is considered to be a special embedding. If all embeddings of a structure-graph are rigid except for some special ones, the graph is said to be generically rigid. The majority of this thesis will focus

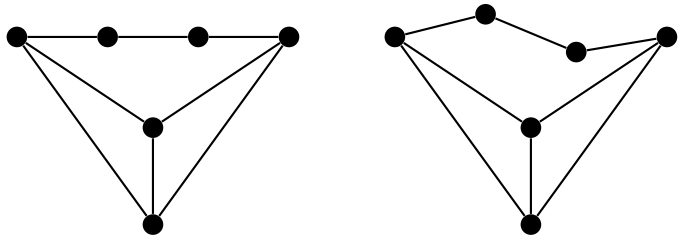

Figure 1.1: Example of two embeddings of the same graph. The graph on the left is rigid; the graph on the right is flexible.

on this definition of rigidity.

Two-dimensional rigidity is well characterized. Laman, [9], described necessary and sufficient conditions on structure-graphs for rigidity, and Henneberg, [7], described a method by which one could obtain exactly the set of rigid graphs by inductively applying certain graph operations starting from a single edge.

On the other hand, studying objects in three dimensions is of interest since results are highly applicable. However, the rigidity of these objects is not as easily characterized. Laman's conditions for three dimensions are only necessary, since there are counter-examples of non-rigid graphs that satisfy the Laman conditions. One such example is the double banana graph shown in Figure 1.2. It satisfies the Laman conditions, but is clearly not rigid. The left and right sides hinge on two points and can be independently revolved around a vertical axis. An extension of Henneberg's construction method to three dimensions has been proven to produce a set that contains all rigid graphs, but additionally contains graphs that are not rigid.

A nice result regarding rigidity for the class of convex polyhedra comes from combining the theorems of Cauchy, Dehn, and Steinitz. Cauchy, [1], and Dehn, [3], showed that strictly convex polyhedra are rigid if, and only if, they are triangulated. Then, Steinitz, [10], showed that convexity of a polyhedron is equivalent to planarity of its structure-graph, meaning that planar graphs are fully characterized in three dimensions. It is therefore desirable to be able to describe rigidity of non-convex

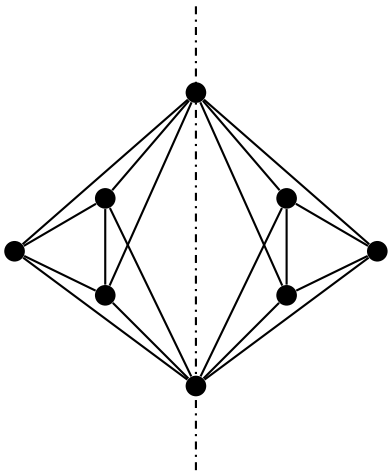

Figure 1.2: The double banana and its axis of revolution

polyhedra.

Some recent attempts at generating rigid polyhedra involve graph operations that preserve rigidity, such as vertex-splitting and spider-splitting ([11], [12]). Studying these operations may give insight into conditions for rigidity in three dimensions. For this work, we implemented these graph operations and focused on comparing the sets of graphs generated.

The format of this thesis is as follows. Chapter 2 outlines background information that is relevant to the work done in this thesis. Chapter 3 describes the vertex-splitting and spider-splitting algorithms we implemented, and Chapter 4 presents our observations of the graphs generated by vertex-splitting and spider-splitting.

Chapter 2

Background

2.1 Frameworks

A *graph* $G = (V, E)$ is a set of vertices, V, together with a set of edges, E, consisting of unordered pairs of vertices from V. A *framework* $\mathcal{F} = (G, p)$ in $\mathbb{R}^n$ is a graph $G = (V, E)$ with an embedding function $p : V \to \mathbb{R}^n$. A framework is also referred to as a realization or configuration of G. In this Chapter, we will see how the rigidity of a graph depends on its embedding, and to what extent we can consider rigidity a combinatorial property.

2.2 Planar Graphs

A graph $G = (V, E)$ is said to be *planar* if G can be drawn in the plane such that no edges are crossing. A subgraph, $H = (V_H, E_H)$ of G is a graph given by a subset of vertices $V_H \subseteq V$ and a subset of edges $E_H \subseteq E$ using the vertices in V_H. A *subdivision of an edge*, (x, y) in G is the addition of a new vertex z, and the replacement of the edge (x, y) with edges (x, z) and (z, y). A *subdivision of* G is a graph, $G' = (V', E')$ obtained by subdividing edges of G.

Kuratowski's Theorem gives a characterization of non-planar graphs.

Theorem 2.1 (Kuratowski's Theorem [8]). *A graph, $G = (V, E)$, is non-planar if and only if it contains a subgraph H that is a subdivision of K_5 or $K_{3,3}$.*

Subdivisions of K_5 and $K_{3,3}$ are sometimes referred to as *Kuratowski subgraphs*.

A framework $\mathcal{F} = (G, p)$ in $\mathbb{R}^3$ is said to be *convex* if for all $x, y \in V$, the line segment between $p(x)$ and $p(y)$ remains inside the framework. A graph is said to be *connected* if for every pair of vertices, there is a path between them. A graph is *n-connected* if at least n vertices need to be removed to disconnect the graph. The following result by Steinitz is an important theorem for rigidity theory regarding frameworks whose underlying graph is planar.

Theorem 2.2 ([10]). *$G = (V, E)$ is a 3-connected planar graph with at least 4 vertices if and only if there exists an embedding function, $p : V \to R^3$, such that $\mathcal{F} = (G, p)$ is convex.*

2.3 Infinitesimal Rigidity

We think of a motion as a set of velocity vectors associated with the vertices of the framework, and we want to define the concept of a rigid motion and a non-rigid motion, or flex, in this context. Since we are only considering frameworks that have edges of fixed length (non-contractible and non-expandable), our definitions of rigid and non-rigid motions should enforce that the distances between connected vertices stay the same throughout the motion. Intuitively, a rigid motion is a motion where the distances between all vertices, whether they are connected or not, stay the same. For example, translations and rotations of the framework would be rigid motions. A flex, or non-rigid motion, would have unconnected vertices moving closer together or further apart. Finally, we consider an object to be rigid if it is impossible to flex the object without deforming its edges.

We represent our intuitive idea of rigid motions and flexes as a condition on the

velocity vectors of a motion. Let $\mathcal{F} = (G, p)$ be a framework in $\mathbb{R}^2$, and let $i \in V$ and $P_i(t) = (x_i(t), y_i(t))$ be a function of time that describes the movement of vertex i. Then $P_i'(0) = (x_i'(0), y_i'(0))$ represents the instantaneous velocity vector on vertex i at the beginning of the motion. Consider the equation $(x_i(t)-x_j(t))^2+(y_i(t)-y_j(t))^2 = h$ where $\sqrt{h}$ is the length of the edge between vertices i and j. Differentiating both sides and evaluating at 0, we get

$$2(x_i(0) - x_j(0))(x_i'(0) - x_j'(0)) + 2(y_i(0) - y_j(0))(y_i'(0) - y_j'(0)) = 0$$
$$(x_i(0) - x_j(0), y_i(0) - y_j(0)) \cdot (x_i'(0) - x_j'(0), y_i'(0) - y_j'(0)) = 0$$

This allows us to define a condition on the instantaneous velocity vectors that will guarantee that the motion preserves edge lengths. Having seen the motivation in $\mathbb{R}^2$, we are now ready to define an infinitesimal motion.

An *infinitesimal motion* on a framework $\mathcal{F} = (G, p)$ is a function $v : V \to \mathbb{R}^n$ such that for every $(x, y) \in E$, the equation $(p(x) - p(y)) \cdot (v(x) - v(y)) = 0$ is satisfied. In this definition, v represents the set of velocity vectors on vertices in $\mathcal{F}$. This condition is equivalent to preserving the lengths of edges in $\mathcal{F}$.

An *infinitesimal rigid motion* on a framework $\mathcal{F} = (G, p)$ is a function $v^* : V \to \mathbb{R}^n$ such that for every pair of vertices $x, y \in V$, the equation $(p(x)-p(y))\cdot(v^*(x)-v^*(y)) = 0$ is satisfied. This condition says that the distances between any pair of vertices is maintained by the motion. In other words, it is as if there are invisible edges between every pair of vertices that prevent them from getting closer together or further apart. $\mathcal{F}$ is said to be *infinitesimally rigid* if every infinitesimal motion is a rigid infinitesimal motion.

Intuitively, we think of an object as being rigid if we are not able to deform it by applying a force to it. It is not immediately clear whether the concept of infinitesimal rigidity is equivalent to our intuitive understanding of what rigidity should mean. We should ask, are there examples of frameworks that are considered infinitesimally rigid, but not rigid? The short answer is no. The following theorem is due to Gluck.

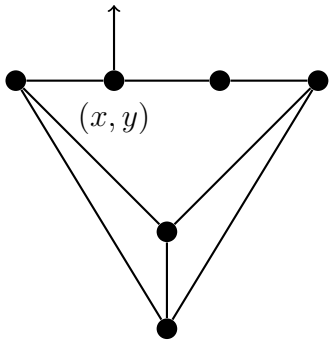

Figure 2.1: Example of a rigid framework that is not infinitesimally rigid

Theorem 2.3 ([5])**.** *If a framework is infinitesimally rigid, then it is rigid.*

On the other hand, infinitesimal rigidity turns out to be a stronger notion than rigidity, as there are there examples of frameworks that are rigid, but not infinitesimally rigid. Figure 2.1 shows a rigid framework. Consider the motion v given by the vector pictured on vertex (x, y) and zero vectors on every other vertex. That is, v can be thought of as pulling on the string of vertices along the top that are held taut. The motion pictured is attempting to rotate (x, y) around the vertex to its left and right at the same time. Since v does not distort the lengths of any edges in the framework, it is an infinitesimal motion. However, it would distort the distances between (x, y) and the two lower points, so it is not a rigid infinitesimal motion, and the framework is not infinitesimally rigid.

Earlier we observed that some realizations of the graph in Figure 2.1 are not rigid by our intuitive idea of rigidity. For this graph, all realizations will not be infinitesimally rigid. By this, we may be misled to believe that infinitesimal rigidity on a framework is independent of the framework's embedding, but there are also examples of graphs that have both infinitesimally rigid and non-infinitesimally rigid embeddings. Figure 2.2 shows an example of this. The framework on the left is not rigid; if the vertical edges were all rotated in the same direction, the framework would "collapse" without deforming any edge lengths. For the framework on the

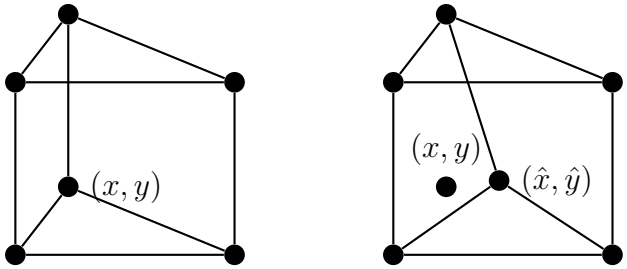

Figure 2.2: Example of two embeddings of the same graph. The graph on the left is not infinitesimally rigid; the graph on the right is infinitesimally rigid.

right, consider the fact that any infinitesimal motion, v, must be an infinitesimal rigid motion when restricted to the vertices on the lower triangle. Let r be the restriction of the motion v to the bottom triangle. Then for all v, $\hat{v} = v - r$ is also an infinitesimal motion, one that assigns zero vectors to the bottom triangle. Therefore, to avoid deforming edges, any vectors of $\hat{v}$ on the vertices of the upper triangle must be perpendicular to the vertical edge connected to them. Since the vertical edges are not all parallel, the only way this is possible is if zero vectors are assigned to every vertex, otherwise the upper triangle will be deformed. We conclude that for every infinitesimal motion v, there is a rigid infinitesimal motion r such that $v = r$, and therefore, the framework on the right is infinitesimally rigid. Even though there are many examples of embeddings of a graph that have different rigidity, we are able to characterize these special embeddings that are in disagreement with most other embeddings.

2.4 Special Embeddings

The term *special embedding* refers collectively to classes of embeddings where rigidity may not be combinatorial. Understanding special embeddings is the key to understanding when the rigidity of different embeddings of the same graph will agree. Let $\mathcal{F} = (G, p)$ be a framework. $\mathcal{F}$ is said to be *constricted* if the vertices of $\mathcal{F}$ are embed-

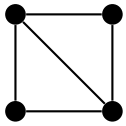

Figure 2.3: Example of a constricted framework. It will be rigid in $\mathbb{R}^2$, but not in $\mathbb{R}^3$.

ded in a space of lower dimension than the embedding function allows. For example, a square lying on the xy-plane in $\mathbb{R}^3$ is a constricted framework. If $\mathcal{F}$ is not constricted, it is said to be *normal*. It is important to separate these kinds of embeddings since rigidity of a framework in a lower dimension does not guarantee rigidity in a higher dimension. Figure 2.3 shows an example of a graph that is rigid in $\mathbb{R}^2$, but in $\mathbb{R}^3$ it is not rigid as it could be folded along the diagonal edge.

For the remainder of this thesis, we will assume that we are dealing with normal frameworks.

A *general embedding* of $\mathcal{F}$ is best explained inductively:

$\mathbb{R}^1$: an embedding where no two vertices are mapped to the same point.

$\mathbb{R}^2$: an embedding where no two vertices are mapped to the same point and no three vertices lie on a line.

$\mathbb{R}^3$: an embedding where no two vertices are mapped to the same point, no three vertices lie on a line, and no four vertices lie on a plane.

If $\mathcal{F}$ is associated with a general embedding, it is called a *general framework*. Previously, we have seen that frameworks with collinear vertices can have rigidity that disagrees with other embeddings, so it is clear why non-general embeddings are considered to be special.

Now we develop the notion of a set of embeddings of a graph for which the infinitesimal rigidity of the corresponding frameworks always agree. Thus, we are able to study rigidity as a property on graphs, while keeping in the back of our mind that the graphs will need to avoid certain embeddings. This is the purpose of generic embeddings. We say a set of n points in $\mathbb{R}^m$ are *algebraically independent* over a field F

if they do not satisfy any non-trivial polynomial $f(x_1, x_2, ..., x_n) = 0$ with coefficients in F. An embedding, p, is *generic* if the coordinates of the vertices are algebraically independent over the rationals, $\mathbb{Q}$. A framework that is associated with a generic embedding is called a *generic framework*.

2.5 Linear Algebra of Infinitesimal Motions

We now define $\mathbb{R}^{mn}$, the vector space that describes all functions mapping a vertex set V to $\mathbb{R}^m$. This provides a platform for understanding both embeddings of graphs and motions on frameworks as they are both functions that map V to $\mathbb{R}^m$. Consider the case of $m = 3$. Let $\mathcal{F} = (G, p)$ be a framework with n vertices in $\mathbb{R}^3$. Let $v : V \to \mathbb{R}^3$ be a function that represents a motion on $\mathcal{F}$. Then a vector $q \in \mathbb{R}^{3n}$ is defined by $q = (x_1, y_1, z_1, x_2, y_2, z_2, ..., x_n, y_n, z_n)$ where x_i, y_i, and z_i correspond to the components of the vector on vertex i. It is clear to see that for motions, addition of vectors in $\mathbb{R}^{mn}$ is equivalent to composing two motions together, and scalar multiplication is an amplification of a motion.

Let $\mathcal{M}(\mathcal{F}) \subseteq \mathbb{R}^{mn}$ represent the set of infinitesimal motions on the framework $\mathcal{F} = (G, p)$. Then $\mathcal{M}(\mathcal{F})$ is a vector subspace. If $\mathcal{F}$ is normal, we will let $\mathcal{R}(\mathcal{F}) \subseteq \mathcal{M}(\mathcal{F})$ represent the set of rigid infinitesimal motions on the framework $\mathcal{F}$. Then $\mathcal{R}(\mathcal{F})$ is also a vector subspace. So an equivalent definition of infinitesimal rigidity of $\mathcal{F}$ is that $\mathcal{R}(\mathcal{F}) = \mathcal{M}(\mathcal{F})$.

Theorem 2.4 ([6])**.** *Let $\mathcal{F} = (G, p)$ be a framework. Then $\mathcal{F}$ is infinitesimally rigid if and only if $dim[\mathcal{M}(\mathcal{F})] = dim[\mathcal{R}(\mathcal{F})]$.*

Proof. Since $\mathcal{R}(\mathcal{F}) \subseteq \mathcal{M}(\mathcal{F})$ and both are vector spaces, then $\mathcal{R}(\mathcal{F}) = \mathcal{M}(\mathcal{F})$ if and only if $dim[\mathcal{M}(\mathcal{F})] = dim[\mathcal{R}(\mathcal{F})]$. □

It may not be immediately obvious, but it turns out that rigid infinitesimal motions of a general framework, $\mathcal{F} = (G, p)$, in any dimension are exactly the motions given

by linear combinations of translations and rotations. Because of this, we are able to explicitly determine $dim[\mathcal{R}(\mathcal{F})]$, by finding the number of basis vectors required to describe all possible linear combinations of translations and rotations. In $\mathbb{R}^2$, two translation vectors are needed: one for the x-direction and one for the y-direction. One rotation vector is needed to describe all rotations in the xy-plane, since any counter-clockwise rotation is also a clockwise rotation. Therefore, in $\mathbb{R}^2$, $dim[\mathcal{R}(\mathcal{F})] = 3$. In $\mathbb{R}^2$, there are three translation vectors for the x, y, and z directions, and three rotation vectors for the xy, xz, and yz planes. Therefore, in $\mathbb{R}^3$, $dim[\mathcal{R}(\mathcal{F})] = 6$.

From the previous observations, we can derive a useful theorem.

Theorem 2.5 ([6])**.** *Let $\mathcal{F} = (G, p)$ be a general framework in $\mathbb{R}^2$ with $|V| \geq 2$. If $|E| < 2|V| - 3$, then $\mathcal{F}$ is not infinitesimally rigid.*

Proof. Let $\mathcal{F} = (G, p)$ be a general framework in $\mathbb{R}^2$. Let $E = \{e_1, e_2, ..., e_{|E|}\}$ and let $E_i \subseteq E$, be given by $E_i = \{e_1, e_2, ..., e_i\}$. Since every motion on a framework with no edges is an infinitesimal motion, we know that:

$$\mathcal{M}((V, E_0), p) = \mathbb{R}^{2|V|}, \text{ and}$$
$$dim[\mathcal{M}((V, E_0), p)] = 2|V|.$$

Now, if we add one edge from E, the dimension of $\mathcal{M}(\mathcal{F})$ will reduce by at most 1, since each edge concerns only two velocity vectors. Therefore, for each $i \in \{1, 2, ..., |E|\}$:

$$dim[\mathcal{M}((V, E_i), p)] \geq 2|V| - i, \text{ and}$$
$$dim[\mathcal{M}(\mathcal{F})] \geq 2|V| - |E|.$$

If $\mathcal{F}$ is infinitesimally rigid, then $3 \geq 2|V| - |E|$. So if $|E| < 2|V| - 3$, then $\mathcal{F}$ cannot be infinitesimally rigid. □

It is straightforward to prove the analogue of this theorem for $\mathbb{R}^3$.

Theorem 2.6 ([6])**.** *Let $\mathcal{F} = (G, p)$ be a general framework in $\mathbb{R}^3$ with $|V| \geq 3$. If $|E| < 3|V| - 6$, then the framework is not infinitesimally rigid.*

Proof. The proof of this theorem is an adaptation of the proof for $\mathbb{R}^2$, where the only difference is that $dim[\mathbb{R}^{3|V|}] = 3|V|$ and $dim[\mathcal{M}(\mathcal{F})] = 6$ when $\mathcal{F}$ is infinitesimally rigid. □

Since the structure graph of a convex polyhedron is a planar graph, we can strengthen Theorem 2.6 for convex frameworks by applying Euler's Formula.

Theorem 2.7 ([6]). *Let $\mathcal{F} = (G, p)$ be a general, convex framework in $\mathbb{R}^3$ with $|V| \geq 3$. If $\mathcal{F}$ is infinitesimally rigid, then $|E| = 3|V| - 6$.*

Proof. Let $\mathcal{F} = (G, p)$ be a general, convex, infinitesimally rigid framework in $\mathbb{R}^3$. By Theorem 2.6, $|E| \geq 3|V| - 6$. Let F denote the faces of $\mathcal{F}$. The sum of the number of sides of each face is equal to twice the number of edges. However, each face has at least 3 sides. So $2|E| \geq 3|F|$, or $\frac{2}{3}|E| \geq |F|$. Then by Euler's Formula:

$$|V| - |E| + |F| = 2$$
$$|F| = |E| - |V| + 2$$
$$\tfrac{2}{3}|E| \geq |E| - |V| + 2$$
$$|V| - 2 \geq \tfrac{1}{3}|E|$$
$$3|V| - 6 \geq |E|$$

Therefore, $|E| = 3|V| - 6$. □

From this result, we begin to see that rigidity is easier to characterize combinatorially for convex polyhedra.

2.6 Generic Rigidity

In Section 2.4, we defined the notion of a generic framework. This concept allows us to treat rigidity and infinitesimal rigidity purely as a combinatorial property of graphs. By our definition of generic, one could prove that the set of all non-generic

embeddings lie on a finite collection of subspaces of $\mathbb{R}^{mn}$ of lower dimension. So we are able to conclude the following:

Theorem 2.8 ([6]). *Almost all embeddings (in the measure theory sense) of a framework in $\mathbb{R}^2$ or $\mathbb{R}^3$ are generic.*

This is worth noting, since we will soon be making conclusions about the rigidity of graphs that will only hold for generic embeddings of that graph.

The next two theorems from [6] capture the purpose of generic embeddings. The first one tells us that for generic embeddings, infinitesimal rigidity follows our intuitive idea of what rigidity should mean.

Theorem 2.9 ([6]). *If $\mathcal{F} = (G, p)$ is a generic framework, it is either both rigid and infinitesimally rigid, or neither rigid nor infinitesimally rigid.*

The second theorem tells us that when we are only considering generic embeddings, rigidity of a framework is only dependent on its graph.

Theorem 2.10 ([6]). *If $\mathcal{F} = (G, p)$ and $\mathcal{G} = (G, \hat{p})$ are generic frameworks with the same underlying graph, then either they are both rigid and infinitesimally rigid, or both are neither rigid nor infinitesimally rigid.*

These theorems are not at all obvious, however, the proofs will be omitted as they involve techniques beyond the scope of this thesis. Based on these theorems, we will define a new type of rigidity. A graph G is said to be *generically rigid in* $\mathbb{R}^m$ if there exists a generic embedding function, $p : V \to \mathbb{R}^m$, such that the framework $\mathcal{F} = (G, p)$ is rigid and infinitesimally rigid. If one generic embedding function produces a rigid framework, then all other generic embeddings will also yield rigid frameworks. So, in terms of generic frameworks, rigidity is a combinatorial property.

2.7 The Laman Conditions

We say a graph, G, is *isostatic* in $\mathbb{R}^m$ if it is generically rigid in $\mathbb{R}^m$ and for all $e \in E$, the graph $G' = (V, E \setminus \{e\})$ is not generically rigid in R^m. In other words, it is minimally rigid in terms of its edges and the removal of any edge will destroy its rigidity.

Theorem 2.11 ([6]). *In $\mathbb{R}^2$ and $\mathbb{R}^3$, a graph, $G = (V, E)$ is generically rigid if and only if there exists a generically rigid subgraph, $G' = (V, E')$ where $E' \subseteq E$.*

Proof. Let $G = (V, E)$ be generically rigid. If G is isostatic, then we are done. If G is not isostatic, then there is an edge $e \in E$ that can be removed such that the graph $G' = (V, E \setminus \{e\})$ is still generically rigid. If G' is not isostatic, repeat this process again, until the graph is isostatic. In $\mathbb{R}^2$ and $\mathbb{R}^3$, this process must halt since infinitesimal rigidity (and hence, generic rigidity) requires that $|E| \geq 2|V| - 3$ for two dimensions and $|E| \geq 3|V| - 6$ for three dimensions.

On the other hand, if we start with an isostatic graph, the addition of any edges cannot possibly destroy its rigidity. So a graph G having an isostatic subgraph $G' = (V, E')$ where $E' \subseteq E$ implies G is generically rigid. □

If a graph is generically rigid in $\mathbb{R}^2$ and $|E| = 2|V| - 3$, then it is isostatic since the removal of any edge will give $|E| < 2|V| - 3$. A similar observation can be made for generically rigid graphs in $\mathbb{R}^3$ where $|E| = 3|V| - 6$. By these observations, we are hopeful that we can easily characterize isostatic graphs in $\mathbb{R}^2$ and $\mathbb{R}^3$.

We know that a generically rigid graph in $\mathbb{R}^2$ satisfies $|E| \geq 2|V| - 3$ and edges can be removed until it becomes an isostatic graph, which must also satisfy $|E| \geq 2|V| - 3$. It may be slightly surprising that it is always possible to remove edges until the inequality becomes equal. That is, if a graph $G = (V, E)$ is isostatic, then $E = 2|V| - 3$. However, a graph satisfying this condition turns out not to be sufficient for generic rigidity. For example, Figure 2.4 shows a graph that satisfies $|E| = 2|V| - 3$, but

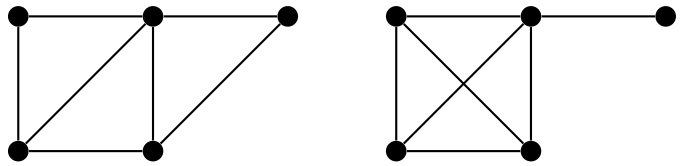

Figure 2.4: The graph on the left is isostatic, but by moving one of its edges, we find a graph that satisfies $|E| = 2|V| - 3$ that is not isostatic.

is not generically rigid. From this example, it seems that in order to be isostatic, a graph must have exactly the correct number of edges, but in addition, the edges must be well-distributed across the graph in some sense. This brings us to the Laman conditions for isostatic graphs in $\mathbb{R}^2$.

Theorem 2.12 (Laman Conditions in $\mathbb{R}^2$ ([9]))**.** *In $\mathbb{R}^2$, a graph, $G = (V, E)$, is isostatic if and only if*

1. $|E| = 2|V| - 3$.
2. *For all $U \subseteq V$ where $|U| \geq 2$, $|E(U)| \leq 2|U| - 3$.*

It is tempting to try writing down an analogue of this theorem in three dimensions, but it is not as straightforward. The Laman conditions are necessary in $\mathbb{R}^3$, but not sufficient.

Theorem 2.13 (Laman Conditions in $\mathbb{R}^3$ ([6], [9]))**.** *In $\mathbb{R}^3$, if a graph, $G = (V, E)$, is isostatic, then*

1) $|E| = 3|V| - 6$.

2) For all $U \subseteq V$ where $|U| \geq 3$, $|E(U)| \leq 3|U| - 6$.

A famous counterexample of the converse of this statement is the double banana, pictured in Figure 2.5. Recall from the introduction that this graph is not rigid; see Figure 1.2. The following is an argument verifying that the double banana satisfies the Laman conditions in R^3. First note that $|E| = 18$ and $|V| = 8$, and since $18 = 3(8) - 6$, 1) is satisfied. To check 2), let $U \subset V$, where $|U| \geq 3$. If $|U| = 3$, then

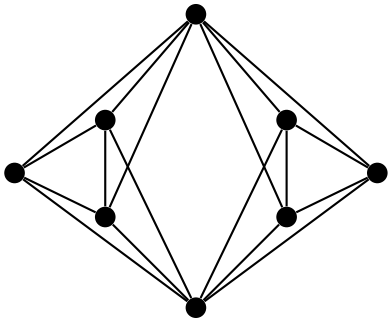

Figure 2.5: The double banana satisfies the Laman Conditions in $\mathbb{R}^3$, but it is not generically rigid.

the maximum number of edges in U is three, when U is complete, and $3 \leq 3(3) - 6$. If $|U| = 4$, then the maximum number of edges is six, and $6 \leq 3(4) - 6$. If $|U| = 5$, the condition will only fail if the subgraph is a K_5, but the double banana contains no K_5. When $|U| = 6$, then U is equal to V minus two vertices. In this case, the maximum number of edges is obtained by deleting two adjacent vertices of degree 4, removing seven edges. Then $11 \leq 3(6) - 6$. When $|U| = 7$, then U is equal to V minus one vertex. The maximum number of edges is obtained when a vertex of minimum degree is deleted, which will remove four edges. Then $14 \leq 3(7) - 6$. This concludes our verification.

2.8 Vertex-Splitting

Definition 2.1. *Let $G = (V, E)$ be a graph, $x \in V$, and let $E_1 = \{(x, 1), (x, 2)\}$ and $E_2 = \{(x, 3), (x, 4),...,(x, k)\}$ where $k \leq deg(x)$. Then a vertex-split on x is the graph $G' = (V', E')$ obtained by adding a new vertex x' so that $V' = V \cup \{x'\}$ and $E' = (E \setminus E_2) \cup \{(x', x), (x', 1), (x', 2),...,(x', k)\}$.*

Figure 2.6 shows an example of a vertex-split. The red edges represent E_1 and the blue edges are the three new edges of the modified graph. Vertex-splitting is a useful operation for studying rigidity theory as it behaves well with rigid frameworks.

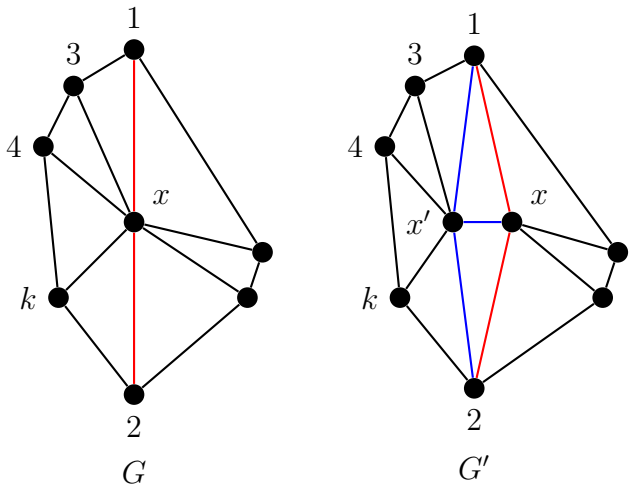

Figure 2.6: A vertex-split on vertex x

This brings us to a theorem due to Whiteley.

Theorem 2.14 ([11]). *Let G be an isostatic graph in $\mathbb{R}^2$ or $\mathbb{R}^3$ and let $G' = (V', E)$ be a vertex-split on a vertex in G. Then G' is isostatic.*

2.9 Spider-Splitting

Definition 2.2. *Let $G = (V, E)$ be a graph, $x \in V$, and let $E_1 = \{(x, 1), (x, 2), (x, 3)\}$ and $E_2 = \{(x, 4), (x, 5), ..., (x, k)\}$ where $k \leq deg(x)$. Then a spider-split on x is the graph $G' = (V', E')$ obtained by adding a new vertex x' so that $V' = V \cup \{x'\}$ and $E' = (E \setminus E_2) \cup \{(x', 1), (x', 2), ..., (x', k)\}$.*

Figure 2.7 shows an example of a spider-split. It is similar to vertex-splitting, but 3 adjacent vertices to x are selected to connect to x' instead of 2 adjacent vertices and x itself. The red edges represent E_1 and the blue edges are the three new edges added as a result of the spider-split.

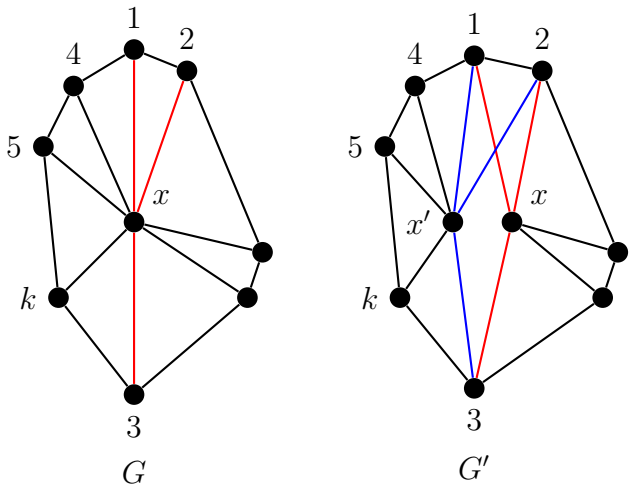

Figure 2.7: A spider-split on vertex x

We now arrive at another theorem by Whiteley.

Theorem 2.15 ([12])**.** *Let G be an isostatic graph in $\mathbb{R}^2$ or $\mathbb{R}^3$ and let $G' = (V', E')$ be a spider-split on a vertex in G. Then G' is isostatic.*

Chapter 3

Methodology

In order to compare the methods of vertex-splitting and spider-splitting in generating rigid graphs, we develop algorithms that generate the sets of graph produced by iteratively applying these operations, starting from a given graph. Algorithm 3.1 and Algorithm 3.2 generate all splits on a single graph using vertex-splitting and spider-splitting respectively. Algorithm 3.3 generates graphs of any size using vertex-splitting or spider-splitting as a building block, starting from a base tetrahedron. Algorithm 3.4 generates graphs up to any size by using any combination of vertex-splitting and spider-splitting on a tetrahedron. The generated graphs can be found at `http://cs.smu.ca/~j_deon/` in a three-dimensional viewable format.

3.1 Vertex-Splitting Algorithm

An implementation of vertex-splitting was written in Python. It takes a graph, G, as input and returns the set, $\mathcal{K}$, of all possible graphs obtainable by applying vertex-splitting to G.

Algorithm 3.1 Vertex-Splitting Algorithm

1: Let $G = (V, E)$ be the given graph
2: Let $\mathcal{K} = \{\}$ be the set that will contain vertex-splits of G
3: **for** each $x \in V$ **do**
4: **for** each pair, x_1, x_2, of neighbours of x **do**
5: Let $G' = (V', E')$ be a copy of G
6: Add a new vertex, x', to V'
7: Add $(x', x), (x', x_1), (x', x_2)$ to E'
8: **for** each C, a subset of neighbours of x other than x_1 and x_2 **do**
9: For all $c \in C$, remove (x, c) from E'
10: For all $c \in C$, add (x', c) to E'
11: **if** G' is not isomorphic to any members of $\mathcal{K}$ **then**
12: Add G' to $\mathcal{K}$
13: **end if**
14: **end for**
15: **end for**
16: **end for**

We verify that this algorithm will generate the set of graphs obtained by vertex-splitting on a given graph G.

Theorem 3.1. *Let $G = (V, E)$ be a graph and let $\mathcal{K}$ be the set of graphs returned by Algorithm 3.1. Let $\mathcal{K}'$ be the set of graphs obtainable by applying vertex-splitting to G. Then $\mathcal{K} = \mathcal{K}'$.*

Proof. Let $H \in \mathcal{K}$. Then $H = (V', E')$ is produced by applying Algorithm 3.1 to G. We show that H is obtainable by performing a vertex-split on G, and conclude that $H \in \mathcal{K}'$. Let $x \in V$ be the vertex of G chosen from Line 3 of Algorithm 3.1, let x_1 and x_2 be the neighbours of x chosen from Line 4, and define $E_1 = \{(x, x_1), (x, x_2)\}$. By

Lines 5 and 6, $V' = V \cup \{x'\}$. From Line 8, $C = \{x_3, x_4,...,x_k\}$, and note that every element of C is also a neighbour of x. Let $E_2 = \{(x, i) | i \in C\}$. Then by Lines 7, 9, and 10, $E' = (E \setminus E_2) \cup \{(x', x), (x', x_1), (x', x_2), ..., (x', k)\}$. Therefore, by Definition 2.1, H is obtained by vertex-splitting on G, so $H \in \mathcal{K}'$ and $\mathcal{K} \subseteq \mathcal{K}'$.

Let $H = (V', E') \in \mathcal{K}'$, with $V' = V \cup \{x'\}$, $E_1 = \{(x, x_1), (x, x_2)\}$, $E_2 = \{(x, x_3), (x, x_4),...,(x, x_k)\}$, and $E' = (E \setminus E_2) \cup \{(x', x), (x', x_1), (x', x_2),...,(x', x_k)\}$. Since Algorithm 3.1 iterates over every vertex of G with Line 3, it would select x at some point during execution. From Line 4, it would also select x_1 and x_2 at some point. Since x_3, x_4,...,x_k are all neighbours of x, the set C from Line 8 would contain exactly these elements at some point. From Lines 7, 9, and 10, the graph generated by the algorithm would contain the same edges as G, except with the edges in E_2 removed, and the edges in $\{(x', x), (x'x_1), (x', x_2),...,(x', x_k)\}$ added. Therefore, H would be produced by Algorithm 3.1, and $\mathcal{K}' \subseteq \mathcal{K}$.

So $\mathcal{K} = \mathcal{K}'$. □

3.2 Spider-Splitting Algorithm

An implementation similar to Algorithm 3.1 was used to generate the set, $\mathcal{L}$, of all possible graphs obtainable by applying spider-splitting to a given graph, G.

Algorithm 3.2 Spider-Splitting Algorithm

1: Let $G = (V, E)$ be the given graph
2: Let $\mathcal{L} = \{\}$ be the set that will contain spider-splits of G
3: **for** each $x \in V$ **do**
4: **for** each triplet, x_1, x_2, x_3, of neighbours of x **do**
5: Let $G' = (V', E')$ be a copy of G
6: Add a new vertex, x', to V'
7: Add $(x', x_1), (x', x_2), (x', x_3)$ to E'
8: **for** each C, a subset of neighbours of x other than x_1, x_2, and x_3 **do**
9: For all $c \in C$, remove (x, c) from E'
10: For all $c \in C$, add (x', c) to E'
11: **if** G' is not isomorphic to any members of $\mathcal{L}$ **then**
12: Add G' to $\mathcal{L}$
13: **end if**
14: **end for**
15: **end for**
16: **end for**

We verify that this algorithm generates the set of graphs obtainable by performing spider-splitting on a given graph G.

Theorem 3.2. *Let $G = (V, E)$ be a graph and let $\mathcal{L}$ be the set of graphs returned by Algorithm 3.2 on input G. Let $\mathcal{L}'$ be the set of graphs obtainable by applying spider-splitting to G. Then $\mathcal{L} = \mathcal{L}'$.*

Proof. Let $H \in \mathcal{L}$. Then $H = (V', E')$ is produced by applying Algorithm 3.2 to G. We show that H is obtainable by performing a spider-split on G, and conclude that $H \in \mathcal{L}'$. Let $x \in V$ be the vertex of G chosen from Line 3 of Algorithm 3.2, let x_1, x_2, and x_3 be the neighbours of x chosen from Line 4, and define $E_1 = \{(x, x_1),$

(x, x_2), $(x, x_3)\}$. By Lines 5 and 6, $V' = V \cup \{x'\}$. From Line 8, $C = \{x_4, x_5,...,x_k\}$, and note that every element of C is also a neighbour of x. Let $E_2 = \{(x, i)|i \in C\}$. Then by Lines 7, 9, and 10, $E' = (E \setminus E_2) \cup \{(x', x_1), (x', x_2), ..., (x', k)\}$. Therefore, by Definition 2.2, H is obtained by spider-splitting on G, so $H \in \mathcal{L}'$ and $\mathcal{L} \subseteq \mathcal{L}'$.

Let $H = (V', E') \in \mathcal{L}'$, with $V' = V \cup \{x'\}$, $E_1 = \{(x, x_1), (x, x_2), (x, x_3)\}$, $E_2 = \{(x, x_4), (x, x_5),...,(x, x_k)\}$, and $E' = (E \setminus E_2) \cup \{(x', x_1), (x', x_2),...,(x', x_k)\}$. Since Algorithm 3.2 iterates over every vertex of G with Line 3, it would select x at some point during execution. From Line 4, it would also select x_1, x_2, and x_3 at some point. Since x_4, x_5,...,x_k are all neighbours of x, the set C from Line 8 would contain exactly these elements at some point. From Lines 7, 9, and 10, the graph generated by the algorithm would contain the same edges as G, except with the edges in E_2 removed, and the edges in $\{(x'x_1), (x', x_2),...,(x', x_k)\}$ added. Therefore, H would be produced by Algorithm 3.2, and $\mathcal{L}' \subseteq \mathcal{L}$.

So $\mathcal{L} = \mathcal{L}'$. □

3.3 Generating Graphs Iteratively Using a Single Operation

An algorithm to find all graphs obtainable by repeatedly applying vertex-splitting or spider-splitting starting from a tetrahedron was implemented in Python. The algorithm below contains a place holder that can be replaced by either vertex-splitting or spider-splitting. It works by first applying the operation to a tetrahedron, and then applying it to each of the newly generated graphs, and so on.

Algorithm 3.3 Generate Graphs Iteratively Using a Single Operation

Let $T = (V, E)$ be the graph representing a tetrahedron
Let $\mathcal{M} = \{T\}$ be the set that will contain iterative splits of T
for each $G \in \mathcal{M}$ **do**
    Let $\mathcal{K}$ be the set returned by either Algorithm 3.1 or Algorithm 3.2
    for each H in $\mathcal{K}$ **do**
        if H is isomorphic to any members of $\mathcal{M}$ **then**
            Remove H from $\mathcal{K}$
        end if
    end for
    Let $\mathcal{M} = \mathcal{M} \cup \mathcal{K}$
end for

We verify that this algorithm generates the set of graphs obtainable from a tetrahedron either using vertex-splitting or spider-splitting.

Theorem 3.3. *Let $\mathcal{M}_{\mathcal{K}}$ be the set of graphs generated by Algorithm 3.3 using Algorithm 3.1 as the splitting technique. Let $\mathcal{M}'_{\mathcal{K}}$ be the set of graphs obtainable by iteratively applying vertex-splitting zero or more times starting from a tetrahedron. Then $\mathcal{M}_{\mathcal{K}} = \mathcal{M}'_{\mathcal{K}}$.*

Proof. The proof follows from Theorem 3.1. □

Theorem 3.4. *Let $\mathcal{M}_{\mathcal{L}}$ be the set of graphs generated by Algorithm 3.3 using Algorithm 3.2 as the splitting technique. Let $\mathcal{M}'_{\mathcal{L}}$ be the set of graphs obtainable by iteratively applying spider-splitting zero or more times starting from a tetrahedron. Then $\mathcal{M}_{\mathcal{L}} = \mathcal{M}'_{\mathcal{L}}$.*

Proof. The proof follows from Theorem 3.2. □

It is important to note that without any other conditions, this algorithm will run

forever. In practice, you can force the algorithm to stop running, for example, once all graphs with a certain number of vertices are generated.

3.4 Generating Graphs Iteratively Using Both Operations

An algorithm to find all graphs obtainable by repeatedly applying combinations of vertex-splitting and spider-splitting starting from a tetrahedron was implemented in Python. It applies both operations to every graph at each step of the iteration.

Algorithm 3.4 Generate Graphs Iteratively Using Both Operations

Let $T = (V, E)$ be the graph representing a tetrahedron
Let $\mathcal{M} = \{T\}$ be the set that will contain iterative splits of T
for each $G \in \mathcal{M}$ **do**
    Let $\mathcal{K}$ be the set returned by Algorithm 3.1
    for each H in $\mathcal{K}$ **do**
        if H is isomorphic to any members of $\mathcal{M}$ **then**
            Remove H from $\mathcal{K}$
        end if
    end for
    Let $\mathcal{L}$ be the set returned by Algorithm 3.2
    for each H in $\mathcal{L}$ **do**
        if H is isomorphic to any members of $\mathcal{M}$ or $\mathcal{K}$ **then**
            Remove H from $\mathcal{L}$
        end if
    end for
    Let $\mathcal{M} = \mathcal{M} \cup \mathcal{K} \cup \mathcal{L}$
end for

We verify that this algorithm generates the set of graphs obtained from a tetrahedron by using combinations of vertex-splitting and spider-splitting.

Theorem 3.5. *Let $\mathcal{M}$ be the set of graphs generated by Algorithm 3.4. Let $\mathcal{M}'$ be the set of graphs obtainable by applying a combination of vertex-splits and spider-splits starting from a tetrahedron. Then $\mathcal{M}_{\mathcal{K}} = \mathcal{M}'_{\mathcal{K}}$.*

Proof. The proof follows from Theorems 3.1 and 3.2. □

3.5 Software

The algorithms from Sections 3.1 to 3.4 were implemented in Python 2.7 and run on an Asus X456UV laptop with a 2.59 GHz Intel Core i7-6500U processor and 12GB of RAM. The NetworkX 2.0 library was used to manipulate the graphs and check for isomorphisms. The isomorphism procedure in NetworkX is an interface to a C implementation of the VF2 graph-matching algorithm [2]. The Planarity 0.4.1 library was used to check graphs for planarity. Plotly 2.1.0 was used to produce interactive, 3-dimensional visualizations of the graphs. Memory was not an issue when running these algorithms, but long computation times only allowed us to generate graphs with as many as 8 vertices.

Chapter 4

Results

In this chapter, we compare the sets of graphs produced by Algorithms 3.3 and 3.4 using the two splitting techniques.

4.1 Vertex-Splitting

Vertex-splitting was observed to have interesting properties regarding the planarity of graphs. It was found that vertex-splitting on a planar graph produced both planar and non-planar graphs, while vertex-splitting on non-planar graphs appeared only to produce non-planar graphs. We verify this result in Theorem 4.1. For the proof of this result, we define an edge contraction, which is the inverse operation of vertex-splitting ([11]).

The following definition is given in [4]. Let $G = (V, E)$ be a graph and let $C = \{(x, y), (x, a), (x, b), (x, u_1),...,(x, u_m), (y, a), (y, b), (y, v_1),...,(y, v_n)\} \subset E$. Then an *edge contraction* on the edge (x, y) is the graph $G' = (V', E')$ where $V' = V \setminus \{x, y\} \cup \{z\}$ and $E' = (E \setminus C) \cup \{(z, a), (z, b), (z, u_1),...,(z, u_m), (z, v_1),...,(z, v_n)\}$. An example of an edge contraction is pictured in Figure 4.1.

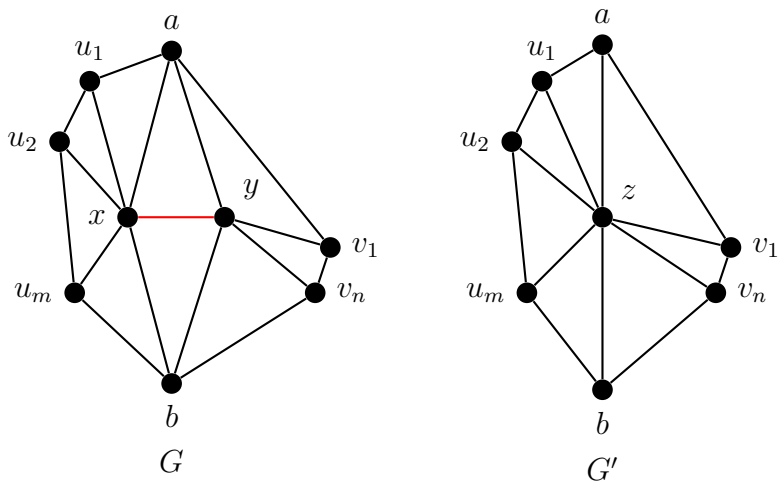

Figure 4.1: An edge contraction on edge (x, y)

Theorem 4.1. *A vertex-split on a non-planar graph, $G = (V, E)$, yields a non-planar graph $G' = (V', E')$.*

Proof. We prove the contrapositive: if G' is planar, an edge contraction on edge $(x, y) \in E$ will yield a planar graph G.

Let G' be planar. Then it can be drawn in the plane with no edges crossing. It is clear from the definition of an edge contraction (see Figure 4.1) that the process of contracting an edge does not create any crossing edges. Therefore, G is planar, and edge contraction preserves planarity. So, vertex-splitting preserves non-planarity. □

This result is not true for spider-splitting. Instead, it is possible to spider-split from a planar graph to a non-planar graph and vice versa. An example of a spider-split from a non-planar graph to a planar graph is shown in Figure 4.2. In the graph on the left, x is the vertex being split, and the three selected neighbours of x are 1, 2, and 3. On the right, x' is the newly added vertex, edges $\{(x', 1), (x', 2), (x', 3)\}$ are added, and $(x, 4)$ is replaced by $(x', 4)$. Note that since the graph on the right represents a convex polyhedron, the graph is planar.

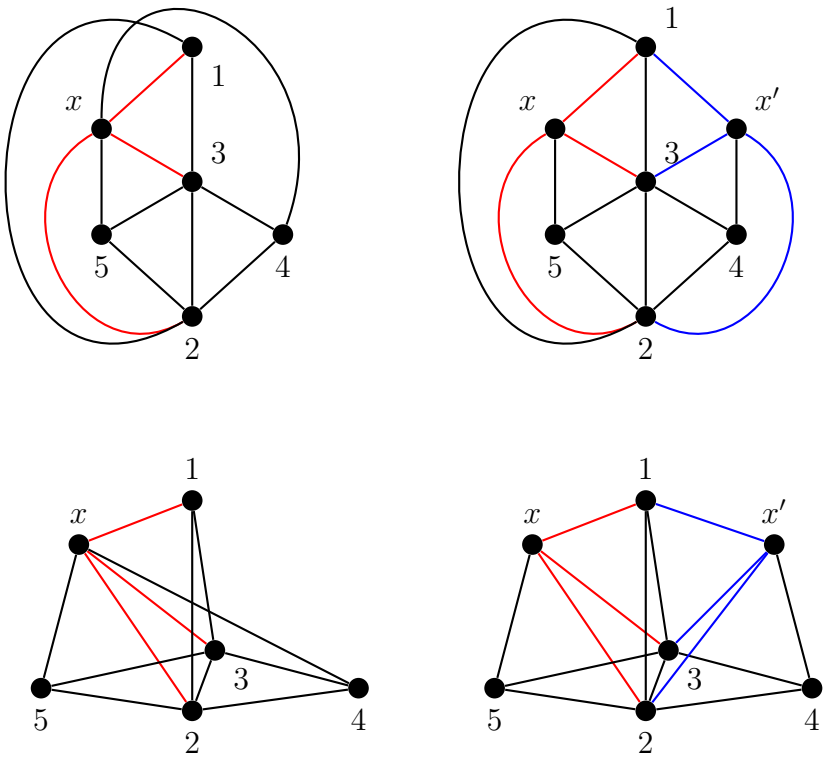

Figure 4.2: A spider-split from a non-planar graph (left) to a planar graph (right)

4.2 Spider-Splitting

Lemma 4.2. *The graph of a triangular bipyramid is the only graph with 5 vertices that can be obtained from a tetrahedron via spider-splitting.*

Proof. We check all possible graphs obtainable by spider-splitting on a tetrahedron. Let $G = (V, E)$, where $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2),\ (1, 3),\ (1, 4),\ (2, 3),\ (2, 4),\ (3, 4)\}$. By symmetry, it suffices to check only the spider-splits of a single vertex. Without loss of generality, let 1 be that vertex. Then the only triplet of neighbours of x we can select is $\{2, 3, 4\}$, in which case, $E_1 = \{(1, 2),\ (1, 3),\ (1, 4)\}$ (the red edges in 4.3), and $E_2 = \emptyset$. We obtain $G' = (V', E')$ with $V' = \{1,\ 2,\ 3,\ 4,\ x'\}$ and $E' = \{(1, 2),$

$(1,3),\ (1,4),\ (2,3),\ (2,4),\ (3,4),\ (2,x'),\ (3,x'),\ (4,x')\}$; see Figure 4.3. This is the graph of a triangular bipyramid. □

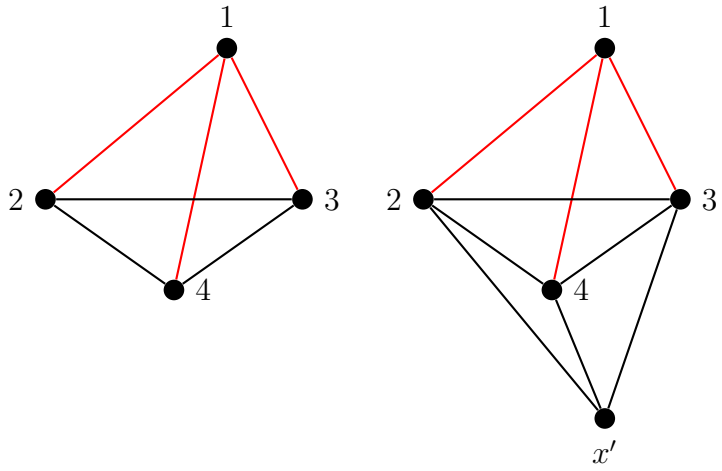

Figure 4.3: Spider-splitting on a tetrahedron with $E_2 = \emptyset$

The simplest example of a graph where vertex-splitting and spider-splitting differ is the octahedron.

Theorem 4.3. *An octahedron cannot be obtained from a tetrahedron by applying spider-splitting iteratively.*

Proof. By Lemma 4.2, it is sufficient to show an octahedron cannot be obtained from a triangular bipyramid. Let $G = (V, E)$ be the graph of a triangular bipyramid and let $x \in V$ be the vertex selected for spider-splitting. Let x' be the new vertex added as a result of spider-splitting. Note that x is either of degree 3 or 4 and that if x' receives no additional edges from x, it will be of degree 3. Since every vertex in an octahedron must be of degree 4, exactly one edge incident to x must be moved to x' in the vertex-split. Since x is of at most degree 4, its degree will be decreased to strictly less than 4. Thus, an octahedron cannot be obtained from a triangular bipyramid, and therefore, from a tetrahedron by spider-splitting. □

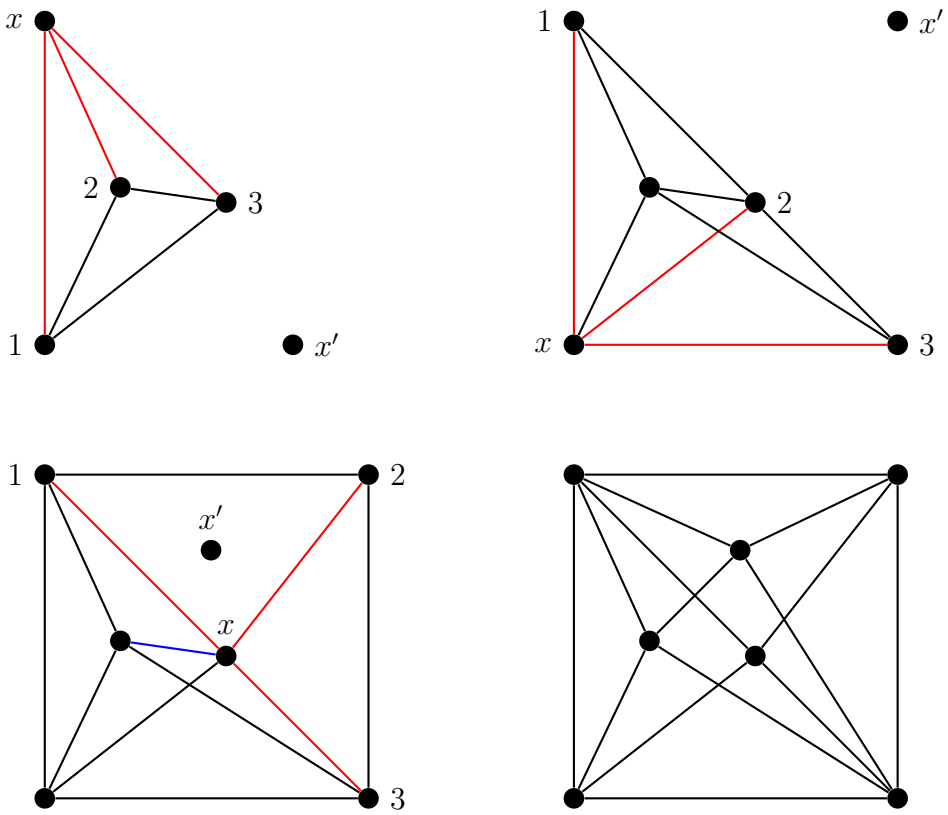

Figure 4.4: Constructing a pentagonal bipyramid from a tetrahderon using a sequence of spider-splits

We observe that many of the graphs with 8 or fewer vertices produced by vertex-splitting and that were not obtainable by spider-splitting contain an octahedron as a subgraph. We suspected that a pentagonal bipyramid may not be obtainable either, since it is also a bipyramid. Surprisingly, this is not true. Figure 4.4 shows one possible sequence of spider-splits that will yield a pentagonal bipyramid. Beginning with a tetrahedron on the top-left, we spider-split on x, add vertex x', and select $\{1, 2, 3\}$ as the triplet of neighbours of x. The second graph is pictured on the top-right. In each graph, the edges $\{(x, 1), (x, 2), (x, 3)\}$ are coloured red, and the edges $\{(x', 1), (x', 2), (x', 3)\}$ are the ones being added as a result of the spider-split. Any additional edges (x, k) that are replaced by (x', k) are coloured blue. The graph pictured in the bottom-right is the pentagonal bipyramid.

To even greater surprise, we found that the hexagonal bipyramid could not be constructed. Since the 3-gonal and 5-gonal bipyramids were achievable, and the 4-gonal and 6-gonal bipyramids were not, we suggest the following conjecture.

Conjecture 4.4. *An n-gonal bipyramid can be obtained by iteratively applying spider-splitting starting from a tetrahedron if and only if n is odd.*

4.3 Comparison of Vertex-Splitting and Spider-Splitting

The sets of all graphs obtained by vertex-splitting and spider-splitting with up to 8 vertices were generated by Algorithm 3.3. Table 4.1 describes the number of graphs on 5 vertices to 8 vertices obtained by vertex-splitting and spider-splitting, divided into planar and non-planar graphs.

Vertices	**Planarity**	**Vertex Splitting**	**Spider Splitting**
$n = 5$	Planar	1	1
	Non-Planar	0	0
$n = 6$	Planar	2	1
	Non-Planar	2	2
$n = 7$	Planar	5	4
	Non-Planar	20	19
$n = 8$	Planar	14	9
	Non-Planar	353	327

Table 4.1: Number of graphs obtained by vertex-splitting and spider-splitting

These sets were compared by checking for isomorphisms between graphs from each set. The following results were found.

Theorem 4.5. *Let $\mathcal{K}$ be the set of graphs with up and including to 8 vertices obtainable by iteratively applying vertex-splitting starting from a tetrahedron, and let $\mathcal{L}$ be the set of graphs with up to and including 8 vertices obtainable by iteratively applying spider-splitting starting from a tetrahedron. Then $\mathcal{L} \subseteq \mathcal{K}$.*

We conjecture that this is true in general, for graphs of any size.

Conjecture 4.6. *Let $\mathcal{K}$ be the set of graphs obtainable by iteratively applying vertex-splitting starting from a tetrahedron, and let $\mathcal{L}$ be the set of graphs obtainable by iteratively applying spider-splitting starting from a tetrahedron. Then $\mathcal{L} \subseteq \mathcal{K}$.*

Since spider-splitting was not able to generate graphs outside the set of graphs obtained by vertex-splitting, the next step was apply spider-splitting to the graphs it could not obtain on its own and see if that would result in the generation of new graphs. Algorithm 3.3 was used to generate the set of graphs with up to and including 8 vertices obtainable by performing all possible sequences of vertex-splits and spider-splits. Surprisingly, this set contained the same graphs as the set given by vertex-splitting alone. This result is summarized in the following theorem.

Theorem 4.7. *Let $\mathcal{K}$ be the set of graphs with up to and including 8 vertices obtainable by iteratively applying vertex-splitting starting from a tetrahedron, and let $\mathcal{M}$ be the set of graphs with up to and including 8 vertices obtainable by performing a sequence of vertex-splits and spider-splits starting from a tetrahedron. Then $\mathcal{K} = \mathcal{M}$.*

This suggests that spider-splitting is less powerful than vertex-splitting in general.

Conjecture 4.8. *Let $\mathcal{K}$ be the set of graphs obtainable by iteratively applying vertex-splitting starting from a tetrahedron, and let $\mathcal{M}$ be the set of graphs obtainable by performing a sequence of vertex-splits and spider-splits starting from a tetrahedron. Then $\mathcal{K} = \mathcal{M}$.*

4.4 Conclusions and Future Work

We have implemented algorithms that generate the sets of all possible graphs obtainable by vertex-splitting and spider-splitting up to and including 8 vertices. Comparing these sets, we found that the set generated via spider-splitting alone is a subset of the set of graphs generated via vertex-splitting. Some examples of graphs that are only produced by vertex-splitting are the octahedron and hexagonal bipyramid. Furthermore, we generated the set of all graphs with up to and including 8 vertices given by any arbitrary sequence of vertex-splits and spider-splits and found that this was equivalent to the set of graphs obtained by vertex-splitting alone. We conjecture that vertex-splitting is a more powerful operation than spider-splitting for any number of vertices at all steps of iteration.

One avenue for future work is to generate graphs with a larger number of vertices to look for counterexamples to our conjectures. This would require analysis of the algorithms to find ways to improve computation speed. The bottleneck of our algorithms is checking graph isomorphisms; it may be possible to use symmetry and generate fewer combinatorially equivalent graphs.

A more theoretical extension of this work would be to work on proving our conjectures. If these conjectures turn out to be false, it would imply that spider-splitting may be a useful tool for studying rigidity, in particular for non-convex polyhedra. On the other hand, if they are true, then vertex-splitting would prove to be a superior technique for studying rigidity. From there, we hope that we will be able to further characterize rigid frameworks in three dimensions, by identifying new classes of generically rigid graphs.

Bibliography

[1] A.L. Cauchy, *Recherche sur les polyèdres - premier mémoire*, Journal de l'École Polytechnique **9** (1813), 66–86.

[2] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, *An improved algorithm for matching large graphs*, In: 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, Cuen, 2001, pp. 149–159.

[3] M. Dehn, *Über die starrheit konvexer polyeder*, Math. Ann. **77** (1916), 466–473.

[4] W. Finbow-Singh and W. Whiteley, *Isostatic block and hole frameworks*, SIAM J. Discrete Math. **27** (2013), no. 2, 991–1020.

[5] H. Gluck, *Almost all simply connected closed surfaces are rigid*, Lecture Notes in Mathematics (T. Rushing (eds) Geometric Topology L. Glaser, ed.), vol. 438, Springer, Berlin, Heidelberg, 1975, pp. 225–239.

[6] J. Graver, *Counting on frameworks*, The Mathematical Association of America, Dolciani Mathematical Expositions, Number 25, 2001.

[7] L. Henneberg, *Die graphsche statik der starren systeme*, Leipzig, 1911.

[8] G. Kuratowski, *Sur le problème des courbes gauches en topologie*, Fund. Math. **15** (1930), no. 1, 271–283.

[9] G. Laman, *On graphs and rigid plane skeletal structures*, Journal of Engineering Mathematics **4** (1970), no. 4, 331–340.

[10] E. Steinitz, *Polyeder und raumeinteilungen*, Encyclopädie der mathematischen Wissenschaften, (Geometries) (IIIAB12) **3** (1922), 1–139.

[11] W. Whiteley, *Vertex splitting in isostatic frameworks*, Structural Topology **16** (1991), 23–30.

[12] ———, private communication, 2018.