Enumeration of Polyhedral Graphs

Samuel George Kamperis

Doctor of Philosophy
Oxford Brookes University
2019

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*Samuel George Kamperis*)

I owe so many, so much...

To my family and friends for their love and support throughout this process.

I thank the staff at Oxford Brookes who have become such a big part of my life.

Dr Rachel Long for her invaluable guidance and supervision, Professor Khaled Hayatleh and the rest of my supervisory team for their help over the years.

To the PRS and the memory of Dr Alex (AJ) Boorman.

My brother Ollie and the brotherhood of Josh and Louis Catlett. My sister Sophie and the "brotherhood" of Katy Stroud.

My parents George and Michele, whose infinite patience and unconditional love have taught me the true meaning of infinity.

My beloved grandmothers, the heads of my amazing family, Maureen Heathfield and the sorely missed Katrina Kamperis.

The one who I met through this PhD and has been my dearest companion ever since, my darling, Aya ♡.

Kampo X

Abstract

This thesis is concerned with the design of a polyhedron enumeration algorithm. The approach taken focuses on specific classes of polyhedra and their graph theoretic properties. This is then compared more broadly to other graph enumeration algorithms that are concerned with the same or a superset which includes these properties.

An original and novel algorithm is contributed to this area. The approach taken divides the problem into prescribed vertex and face degree sequences for the graphs. Using a range of existence, ordered enumeration and isomorphism techniques, it finds all unique 4-regular, 3-connected planar graphs. The algorithm is a vertex addition algorithm which means that each result output at a given stage has a new vertex added. Other results from different stages are never required for further computation and comparison, hence the process is embarrassingly parallel. Therefore, the enumeration can be distributed optimally across a cluster of computers.

This work has led to a successfully implemented algorithm which takes a different approach to its treatment of the class of 4-regular, 3-connected planar graphs. As such this has led to observations and theory about other classes of graphs and graph embeddings which relate to this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

This work attempts to answer some open questions about patterns and properties of certain groups of polyhedra. It is motivated by the limitation of current data available due to the computational complexity of calculating it and has been driven by optimisations of these algorithms.

1.2 Thesis Outline

The focus of this thesis is the development of algorithms which construct topologies of polyhedra up to a given number of vertices. The approach taken in the leading algorithms in this area have all evolved from the principle operations set out in Tutte's Wheel Theorem [Tutte, 1961] (described in Subsection 2.3.5). Rather than manipulating existing results to find others, the work described here uses vertex and edge addition methods, providing high degrees of problem subdivision, increasing potential for parallelisation.

The thesis first introduces the geometric and graph theoretic concepts used throughout the research. A review of relevant algorithms and their significance is then given. We then discuss the implementation and analysis of results. The research methodology broadly falls under one of two categories:

- Enumeration, the process of counting all distinct results in a given search

space.

- Analysis, an in-depth discussion of patterns and relationships between subsets of the results.

Both themes present their own unique challenges and combinatorial properties. Enumeration requires completion without duplication and as such at any stage both uniqueness must be guaranteed and existence is desirable. Analysis of existence and consequently quantity is considered throughout this research, providing termination to computation that yields no results and upper bounds on expected results. This is clear as if the enumeration search space is reduced, computation is optimised.

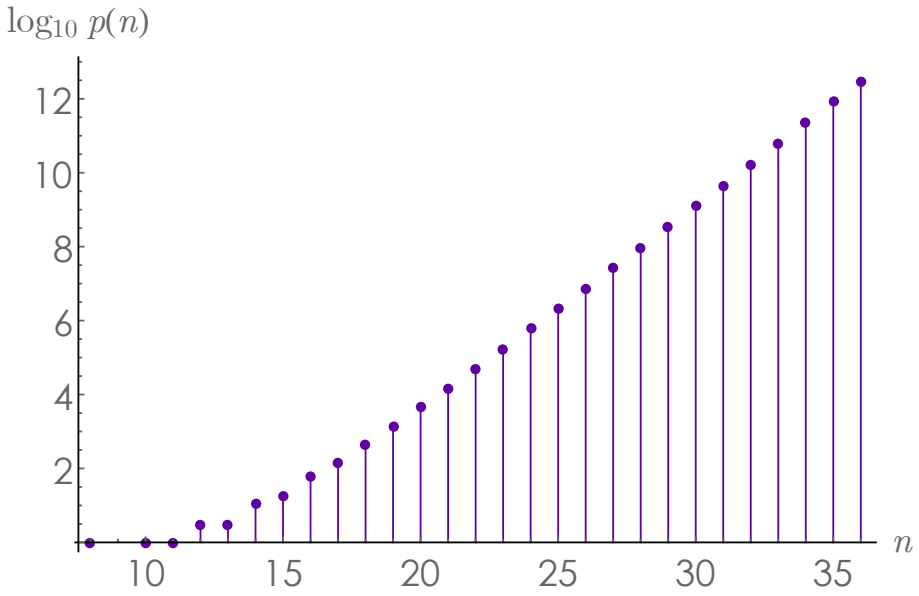


Figure 1.1: Log plot of numbers of polyhedra $p(n)$ with only quadrilateral faces and n vertices

The polyhedra of interest are the duals of 4-regular polyhedra. That is, polyhedra with only quadrilateral faces. The leading algorithm in this area is plantri [Brinkmann et al., 2005]. The up to date list of total numbers of 4-regular polyhedra is kept on the Online Encyclopedia of Integer Sequences [Sloane, 1994]. Table 1.1 shows the first few numbers known for polyhedra with only quadrilateral faces. The current limit of data calculated is 36 vertices which has 3,000,183,106,119 polyhedra. Figure 1.1 shows a log plot of all currently known results. We see that after the initial cases where $n < 15$ there is a consistent exponential growth

however, the initial cases do not exhibit such behaviour as there are extremely few 4-regular polyhedra that exist with so few vertices (the numbers of 4-regular $p(n)$ is Table 1.1 for $8 \leq n \leq 14$ are $1, 0, 1, 1, 3, 3, 11$).

n	$p(n)$
8	1
9	0
10	1
11	1
12	3
13	3
14	11
15	18
16	58
17	139
18	451
19	1,326
20	4,461

Table 1.1: Number of polyhedra $p(n)$ with only quadrilateral faces and n vertices

After a discussion of the methods and results used in enumeration, this work then discusses other properties that are more pertinent in this research and how they may be applied to other problems. Further avenues of investigation are then proposed which could yield not only more results but other applications.

Chapter 2 first introduces the definitions of polyhedra that motivate the algorithm designed in this thesis. We then review fundamental theorems in both polyhedral geometry and graph theory. This is then framed in the context of different algorithms which have been applied to problems related to these areas.

Chapter 3 then introduces the definitions and theory on which this research is based. Through a deeper investigation of different properties of specific classes of polyhedral graphs, we prove theorems on which the algorithm relies to generate these results.

Chapter 4 details the main body of work in this thesis. We provide detail of the algorithm design and comment on optimisations made to improve the performance and reduce the computational cost of enumerating the 4-regular

polyhedral graphs. This is then followed by a comparative review of different code implementations of the algorithm with and without specific optimisations to highlight gains that have been made. Results generated by the algorithm are then investigated in the context of the theory developed in Chapter 3.

Chapter 5 then proposes some new formulae for determining the realisability of generating results from the algorithm using only the initial parameters provided. These are then proved, their impact discussed and further avenues of investigation proposed.

Chapter 6 discusses related open problems where enumeration algorithms and their design could be applicable. A summary of the work and future research to consider is then provided. The thesis concludes with a summary of the contributions made and commentary of the novelty of this algorithm.

Chapter 2

Background

2.1 Introduction

This chapter provides a review of the mathematics relevant to the research in this thesis. Initially in Section 2.2 we introduce polyhedra and their geometry, this informs the initial motivation behind the research. Section 2.3 then presents the graph theoretic principles which translate the topologies of polyhedra to planar graphs, providing the objects and operations on which the enumeration process is designed. Finally in Section 2.4 we highlight algorithms where graph theoretic data structures and processes are applied to solve similar or related problems.

2.2 Polyhedra

2.2.1 Definitions

Convexity

Convex polygons and convex polyhedra are convex hulls of a finite set of points in 2-dimensional and 3-dimensional space respectively. A **convex hull** of a set of points X is the smallest convex set that contains X. A convex set is defined as:

"A set K in d-dimensional space is called a **convex set** or **convex body** if the line segment joining any pair of points of K lies entirely in K." [Croft et al., 1991,

p. 6]

In this sense convex polyhedra are three-dimensional solids constructed from polygons which have no spikes, caves or holes.

Polyhedra

A **polyhedron** is a three-dimensional solid constructed from two-dimensional polygons. When referring to polyhedra, unless otherwise stated we mean convex polyhedra, which are defined as follows:

"We call a 3-dimensional convex hull of a finite set of points a **convex polyhedron**. A boundary point **x** is a **vertex** if there is some plane that intersects the polyhedron P in the single point **x**. A line segment L in the boundary is an **edge** of P if there is a plane that intersects P in the segment L, and a region in the boundary of P is a **face** if it is the intersection of a plane with P and has a positive area." [Croft et al., 1991, p. 48]

Consequently if a polyhedron is convex, so too are the polygonal faces. [Cromwell, 1999]

We note the following properties:

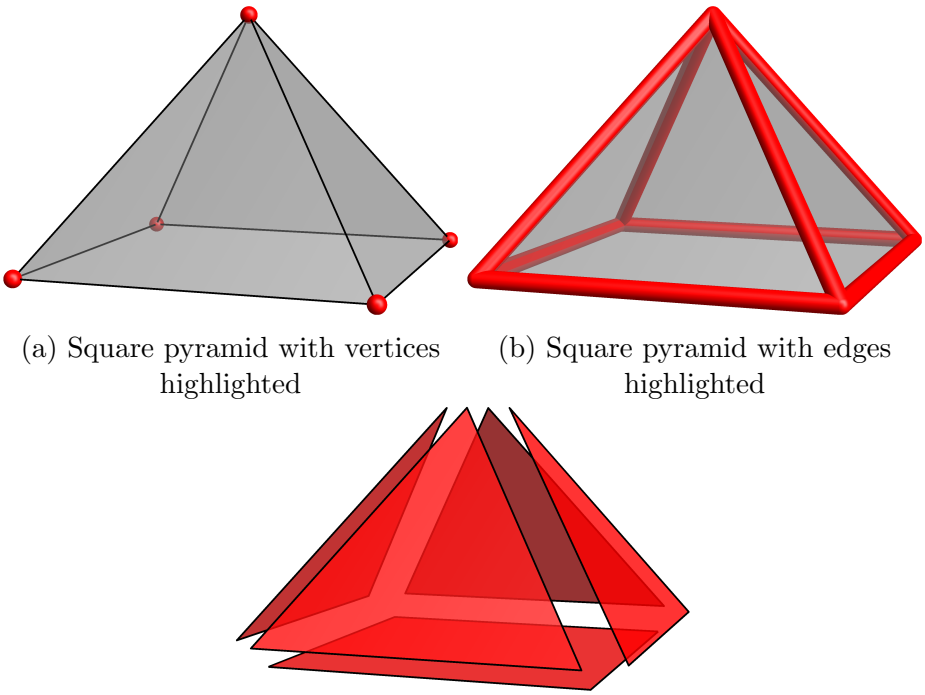
- The lines between two polygons are ***edges***.
- The points at which three or more polygons meet are ***vertices***.
- The polygons themselves are ***faces***.

Distinctiveness

By **distinct** polyhedra, we mean polyhedra which are topologically different. Later we define these in terms of planar graphs relating to the polyhedra, in which case an equivalent statement about the isomorphism of the graphs can be stated.

2.2.2 Euler's Formula

The classical result for convex polyhedra, Euler's Formula [Euler, 1758] states:



(a) Square pyramid with vertices highlighted

(b) Square pyramid with edges highlighted

(c) Square pyramid with faces highlighted

Figure 2.1: Square pyramids with vertices, edges & faces highlighted

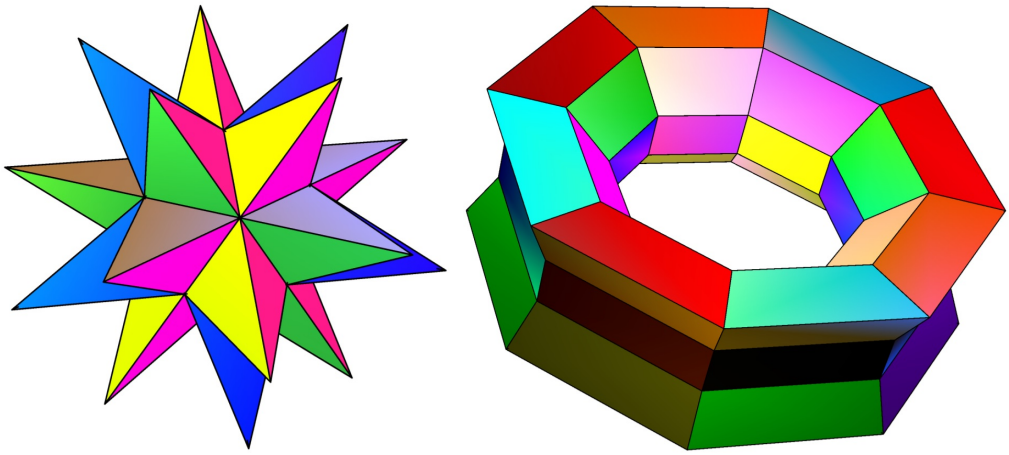
Theorem 2.2.1 (Euler's Polyhedron Formula)**.** *Given the number of vertices* $|V|$*, edges* $|E|$ *and faces* $|F|$ *of a convex polyhedron,*

$$|V| - |E| + |F| = 2.$$

This formula still holds for polyhedra which can be mapped to the surface of a sphere. That is, polyhedra which do not have holes. Figure 2.2 (a), the great stellated dodecahedron, although not convex still satisfies Euler's Formula. However, the toroidal polyhedron in Figure 2.2 (b) which clearly has a hole in it and would not map to the surface of a sphere but instead a torus, would not satisfy the formula. The value of its Euler Characteristic ($|V| - |E| + |F|$) is still deterministic based on the number of holes in the polyhedron.

2.3 Planar Graphs

If we only need consider the topology of a polyhedron then it suffices to know the **graph** of the polyhedron from which we can infer its **embedding** in the plane.



(a) Great stellated dodecahedron (b) Toroidal polyhedron

Figure 2.2: Non-convex polyhedra

2.3.1 Definitions

A **graph** $G(V, E)$ is a set of **vertices** $V = \{v_1, v_2, \ldots, v_n\}$ and **edges** $E \subseteq V \times V$ that represent connections between them. A **subgraph** $G(V', E')$ is a graph that consists of a subset of vertices and edges of another graph. Two vertices v_i and v_j in a graph are **adjacent** iff there exists the edge $\{v_i, v_j\}$ in the edge set.

A graph is **simple** if it has at most one edge between any two vertices and no 'self looping' edges that join a vertex to itself. Unless otherwise stated all graphs are simple. A graph is **undirected** if the edges are not directed from one vertex to another.

There are other equivalent objects for representing the edges in a graph. An **adjacency matrix** of a graph is a $|V| \times |V|$ matrix where for a simple undirected graph if an edge $\{v_i, v_j\}$ exists then its adjacency matrix A will have the value 1 for positions $A_{i,j}$ and $A_{j,i}$. For any entry which does not represent an edge the value is 0. An **adjacency list** is a list of lists where if there exists an edge $\{v_i, v_j\}$ the i^{th} list will contain j and the j^{th} list will contain i.

An **embedding** is a set of relative positions of vertices with respect to their adjacent (edge-connected) vertices in the graph. A fixed geometric layout is not

required, it will suffice that an ordered list of adjacencies (**adjacency list**) about each vertex is given.

If an embedding of the vertices and edges joining them can be found in the plane such that no edges intersect other than at a vertex they both share, then the graph is **planar** and this is a **planar embedding**.

Example: Graph Embedding

We provide an example of vertices, edges and a planar embedding of the graph relating to the octahedron. Let V be the set of vertices:

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\},$$

and E be the set of edges:

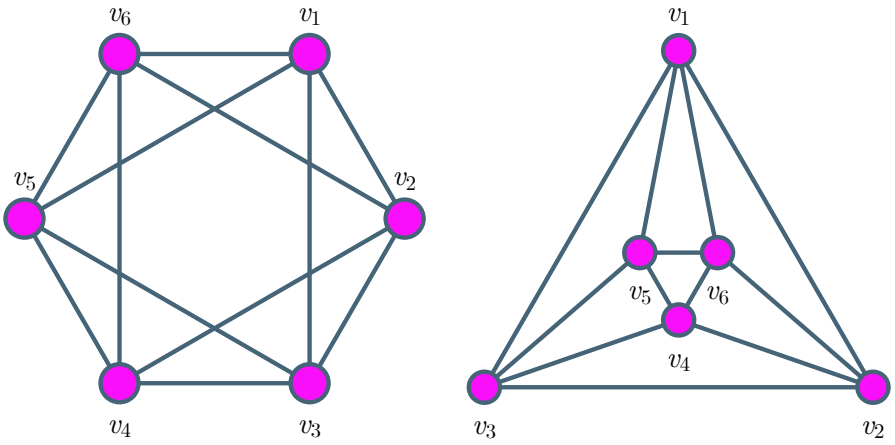
$$\begin{aligned} E = \{&\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_5\}, \{v_1, v_6\}, \{v_2,\ v_3\}, \{v_2, v_4\}, \\ &\{v_2, v_6\}, \{v_3, v_4\}, \{v_3, v_5\}, \{v_4,\ v_5\}, \{v_4, v_6\}, \{v_5, v_6\}\}, \end{aligned}$$

An ordered adjacency list which gives a planar embedding to this graph is given in Table 2.1 and illustrated in Figure 2.3 (b).

v_1	v_2	v_6	v_5	v_3
v_2	v_1	v_3	v_4	v_6
v_3	v_1	v_5	v_4	v_2
v_4	v_2	v_3	v_5	v_6
v_5	v_1	v_6	v_4	v_3
v_6	v_1	v_2	v_4	v_5

Table 2.1: Ordered adjacency list for the graph of an octahedron

The **dual** of a planar graph embedding is the graph formed by placing a vertex in every face of the embedding and an edge connecting any two vertices representing faces which shared an edge in the embedding. Figure 2.4 gives an example of this



(a) Non-planar embedding of the graph of an octahedron

(b) Planar embedding of the graph of an octahedron

Figure 2.3: Graphs representing an octahedron

using the embedding of the octahedron graph. The dual of the octahedron graph is the cube graph.

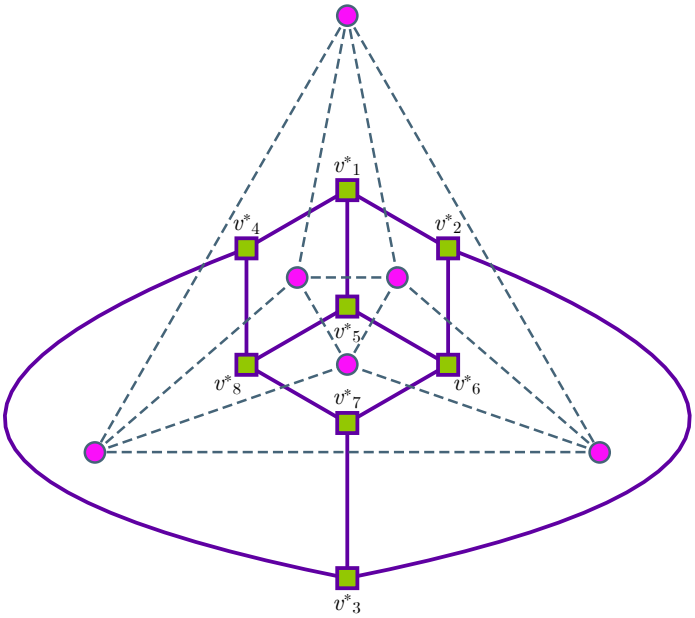


Figure 2.4: Creating the dual of a planar embedding

k **connected** graphs require the removal of at least k vertices from the graph in order to disconnect the graph.

k **edge connected** graphs require the removal of at least k edges from the graph in order to disconnect the graph.

The **degree** of a vertex is the number of edges incident with that vertex. For example all vertices in the graph of an Octahedron (Figure 2.3) have degree 4. For a planar embedding of a graph the degree of a face is the number of vertices (equal to the number of edges) around the face.

The **vertex degree sequence** of a graph is the ordered list of the vertex degrees of all vertices in its vertex set. Similarly the **face degree sequence** of a planar embedding is the ordered list of face degrees in the embedding, this includes the outline face. The face degree sequence of a planar embedding is the vertex degree sequence of its dual graph.

A graph is **bipartite** if there exists a bipartition of the vertex set V into two independent sets V_1 and V_2 where no vertices in V_1 share an edge and no vertices in V_2 share an edge. Consequently, every edge in the graph connects a vertex in V_1 with a vertex in V_2. For example the non-planar $K_{3,3}$ graph in Figure 2.5 (b) is bipartite. The vertices in the set V_1 ($\{v_1, v_2, v_3\}$) are shown as pink circles and the vertices in the set V_2 ($\{v_4, v_5, v_6\}$) are shown as lime squares.

Any planar embedding of a bipartite graph will have a face degree sequence with only even degrees. Consequently the dual of a planar embedding of a bipartite graph will have a vertex degree sequence with only even degrees.

This can be shown by considering the edges of a bipartite graph $G(V, E)$. Suppose we have the vertex sets V_x and V_y which represent a bipartition of a planar graph such that $V = V_x \cup V_y$ and $E \subseteq V_x \times V_y$. An embedding of this graph will have faces that can be drawn by a cyclic list of a union of subsets of V_x and V_y.

Let the degree of a face in this embedding be of size f. This face has f edges and f vertices. Each vertex will have two edges emanating from it to draw the face and similarly each edge will be connected to two vertices that draw the face.

Consider a subgraph $G' = G(V', E')$ consisting only of the edges and vertices for that face. Every vertex in this subgraph will have degree 2. It will still be bipartite and as such every edge must connect from a vertex in V_x to a vertex in V_y.

If all vertices have two edges, they must appear in the edge set exactly twice. Let there be m vertices from V_x in the subgraph. Since every edge in the edge set must have exactly one vertex from V_x and this vertex appears exactly twice, there must be $2m$ edges in E'. Hence $f = 2m$ for some m and is even.

Graph isomorphism is an equivalence between two graphs where the graphs can be non-simple having multiple edges between two vertices and 'self looping' edges, it is defined as:

"Two graphs G_1 and G_2 are **isomorphic** if there is a one-one correspondence between the vertices of G_1 and those of G_2 such that the number of edges joining any two vertices of G_1 is equal to the number of edges joining the corresponding vertices of G_2." [Wilson, 2010, pg. 9]

When stating that a subgraph exists in two graphs it is implicit that the labelling of the vertices (and consequently the edges) might be different but that the subgraphs are isomorphic.

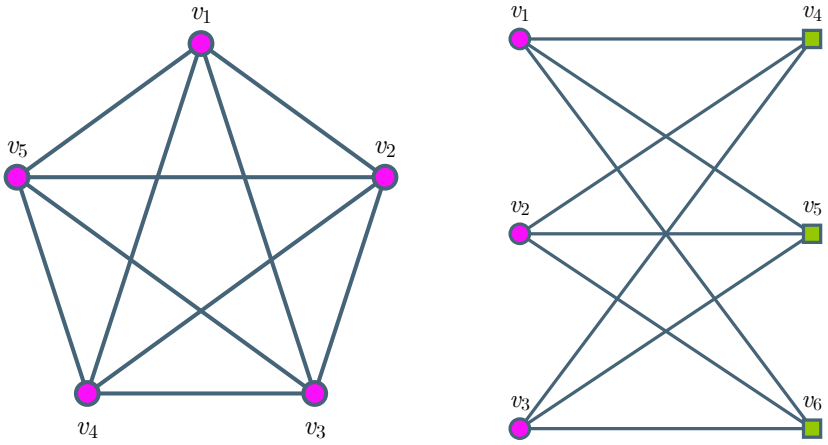
Theorem 2.3.1. *If $\delta(G)$ is the minimum vertex degree of a graph G, $\kappa(G)$ is the vertex connectivity of a graph G and $\lambda(G)$ is the edge connectivity of a graph G. Then,*

$$\kappa(G) \leq \lambda(G) \leq \delta(G).$$

2.3.2 Kuratowski's Theorem

Kuratowski's Theorem [Kuratowski, 1930] proved that all non-planar graphs can be reduced to one of two non-planar graphs through joining vertices. Hence if a graph cannot be reduced in this manner it is planar.

Theorem 2.3.2 (Kuratowski's Theorem)**.** *A graph is planar if and only if it does not contain a subdivision of K_5 or $K_{3,3}$. [Nishizeki and Chiba, 1988]*



(a) K_5 (b) $K_{3,3}$

Figure 2.5: The non-planar graphs K_5 and $K_{3,3}$

The two graphs K_5 and $K_{3,3}$ are shown in Figure 2.5. K_5 is the simple complete graph with five vertices, each of which has degree four and therefore is adjacent to all other vertices in the graph. For K_5 the vertex set can be defined as:

$$V = \{v_1, v_2, v_3, v_4, v_5\},$$

with the edge set:

$$E = \{\{v_i, v_j\}, \text{ for } i = 1, 2, 3, 4 \text{ and } j = i + 1, \ldots 5\}.$$

$K_{3,3}$ is the complete bipartite graph with three vertices in each bipartition. Therefore every vertex has degree three. We can define $K_{3,3}$ by the vertex partitions:

$$V_1 = \{v_1, v_2, v_3\} \text{ and } V_2 = \{v_4, v_5, v_6\}.$$

Where the vertex set $V = V_1 \cup V_2$ and the edge set is:

$$E = V_1 \times V_2.$$

2.3.3 Steinitz' Theorem

Steinitz proved that each topologically distinct convex polyhedron can be projected onto a plane creating a unique 3-connected planar graph, and as such there

is only one embedding for this graph. Similarly the dual of any polyhedron has the dual graph for its projection [Steinitz and Rademacher, 1934]. In his book *Convex Polytopes*, Grunbaum comments that Steinitz' Theorem is "the most important and deepest known result on 3-polytopes" [Grünbaum et al., 1967, pg. 235].

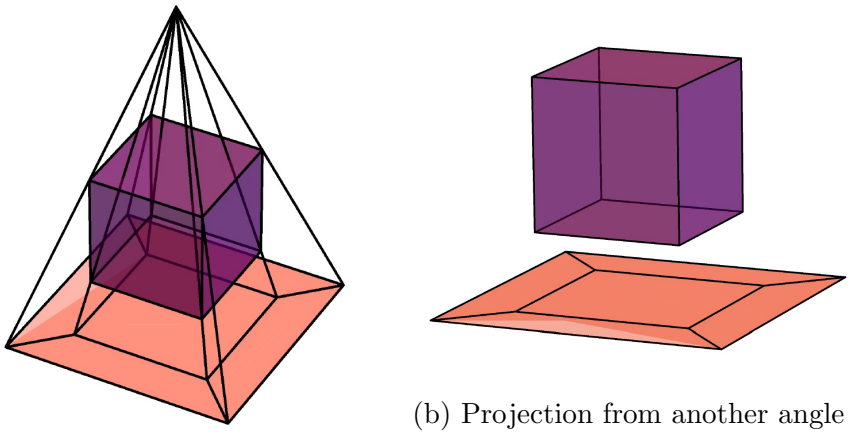
Theorem 2.3.3 (Steinitz' Theorem)**.** *Every convex polyhedron can be represented as a 3-connected planar graph and every 3-connected planar graph forms a convex polyhedron.*

2.3.4 Cauchy's Proof of Euler's Formula

A proof of Euler's formula was published by Cauchy in 1813 where a planar graph can be drawn representing the vertices, edges and faces of any convex polyhedron and it can subsequently be reduced to a single simple case. When mapped to the plane the polyhedron loses one face which is the outline of the graph (Figure 2.6). It can then be shown that every polygonal face with p sides can be cut into triangular faces by adding $p-3$ edges to existing vertices of that face (Figure 2.7 (b)). This increases the number of faces by $p-3$ for each p sided polygon and the equivalent number of edges. It therefore cancels in the formula. Two operations to remove triangular faces from the graph are then given. One would remove a face which has two edges on the outline and consequently a vertex shared by no other faces (Figure 2.7 (c)). The other would remove an edge and face from the outline, but no vertices. Neither would change the formula. Then by repeated edge and vertex removal the graph reduces to a simple case which shows the formula holds [Cauchy, 1813a, Cauchy, 1813b].

2.3.5 Tutte's Wheel Theorem

Tutte's Wheel Theorem [Tutte, 1961] states that any 3-connected planar graph can be found from repeated vertex contraction and splitting operations on a wheel graph. A description of wheel graphs is given in Subsection 3.2. Figure 2.8 shows the operations to create all 3-connected graphs that are derived from a wheel graph with six vertices.



(a) Projection with lines indicating point of origin

(b) Projection from another angle without lines

Figure 2.6: Projections of a cube onto a plane

Vertex contraction is performed by merging two vertices which are edge connected but have no adjacent vertices in common. That is, given vertices v_i and v_j in a graph $G(V, E)$, it is possible to contract v_i and v_j if there exists the edge $\{v_i, v_j\} \in E$ and for all other edges connected to v_i, ($\{v_i, v_x\}$, where $v_x \neq v_j$) the edge $\{v_j, v_x\}$ does not exist. If this is the case the vertices can be contracted, the graph loses one vertex (v_j) and one edge ($\{v_i, v_j\}$) and all other edges connected to v_j ($\{v_j, v_y\}$, $v_y \in V$ and $v_y \neq v_i$) are relabelled $\{v_i, v_y\}$. In Figure 2.8 the direction of the arrows indicates where vertex contraction has taken place. The arrow always points from a graph with $|V|$ vertices to a graph with $|V|-1$ vertices where one vertex has been contracted from the former graph.

Vertex splitting is the other operation performed, this can be performed on a vertex v_i with degree at least four. The edges connected to the vertex are divided into two groups, each containing at least two edges, the edges in each group must occur consecutively in the ordered adjacency list about the vertex. For instance if the ordered adjacency list about v_i is (v_1, v_2, v_3, v_4) then the groups could be either:

$$\{v_1, v_2\} \text{ and } \{v_3, v_4\},$$

or

$$\{v_4, v_1\} \text{ and } \{v_2, v_3\}.$$

A new vertex v_j is added to V and if taking the first case, the edges $\{v_i, v_1\}$ and $\{v_i, v_2\}$ are kept in E. The edges $\{v_i, v_3\}$ and $\{v_i, v_4\}$ are relabelled $\{v_j, v_3\}$ and $\{v_j, v_4\}$ and a new edge $\{v_i, v_j\}$ is added to E. This operation has the inverse effect to vertex contraction, adding one new vertex and one new edge to the graph.

Since both operations add or remove the same number of vertices and edges at each stage, every graph will have the same number of faces as the initial wheel graph as a consequence of Euler's Formula. In Figure 2.8 all graphs have six faces.

2.3.6 Tutte Embedding

Tutte also proved that if an outer face of a 3-connected planar graph is fixed such that it is convex, a unique solution to the coordinates of the internal vertices can be found. These coordinates will give a planar embedding where all edges can be drawn with straight lines and all faces are also convex [Tutte, 1963]. The coordinates of the internal vertices will be barycentric with respect to their adjacent vertices connected by edges. The process works as follows. For a graph with V vertices, select a p-sided face. Embed the p sided face in the plane such that it is convex. Then form a system of linear equations for the $V - p$ vertices where the coordinates in $\mathbb{R}^2$ are unknown. Hence there are two sets of $V - p$ equations to solve.

For example, suppose we wish to find the Tutte embedding of the octahedron graph in Figure 2.3 (b). Let $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$ be the Cartesian coordinates of the vertices v_1, v_2, v_3, v_4, v_5 and v_6 respectively. We then take the outer triangle consisting of vertices v_1, v_2 and v_3 and give them coordinates such that they form an equilateral triangle, where

$$c_1 = \begin{pmatrix} 0 \\ \sqrt{3} \end{pmatrix} \ , \ c_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \ \text{and} \ c_3 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}.$$

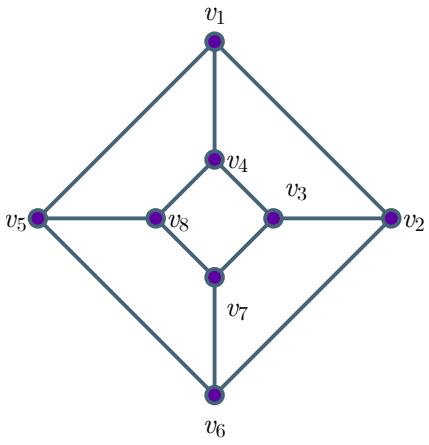
The three remaining coordinates c_4, c_5 and c_6 are the average of the coordi-

nates of their adjacent vertices and can therefore be expressed in following form:

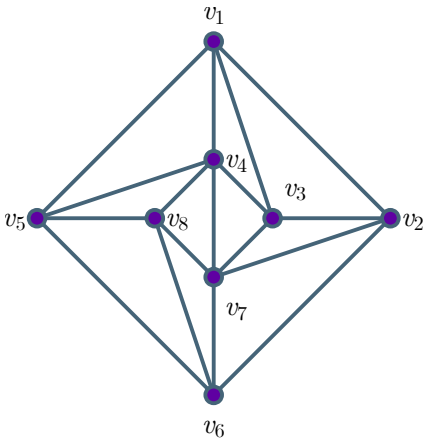
$$\begin{pmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} c_4 \\ c_5 \\ c_6 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

Consequently we find the remaining coordinates for a Tutte embedding of this graph:

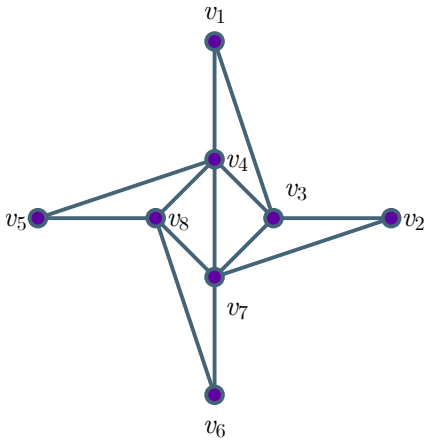
$$c_4 = \begin{pmatrix} 0 \\ \frac{\sqrt{3}}{5} \end{pmatrix} , \; c_5 = \begin{pmatrix} -\frac{1}{5} \\ \frac{2\sqrt{3}}{5} \end{pmatrix} \text{ and } c_6 = \begin{pmatrix} \frac{1}{5} \\ \frac{2\sqrt{3}}{5} \end{pmatrix} .$$



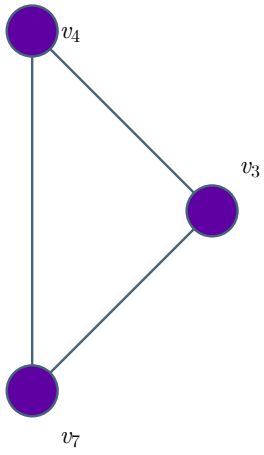
(a) Initial graph before edge addition



(b) Graph after edge addition to contain only triangular faces. Faces on the outline contain one independent edge.



(c) Faces with two indepedent edges and an independent vertex on the outline



(d) The final face

Figure 2.7: Planar graph of a cube at different stages of Cauchy's proof of Euler's formula

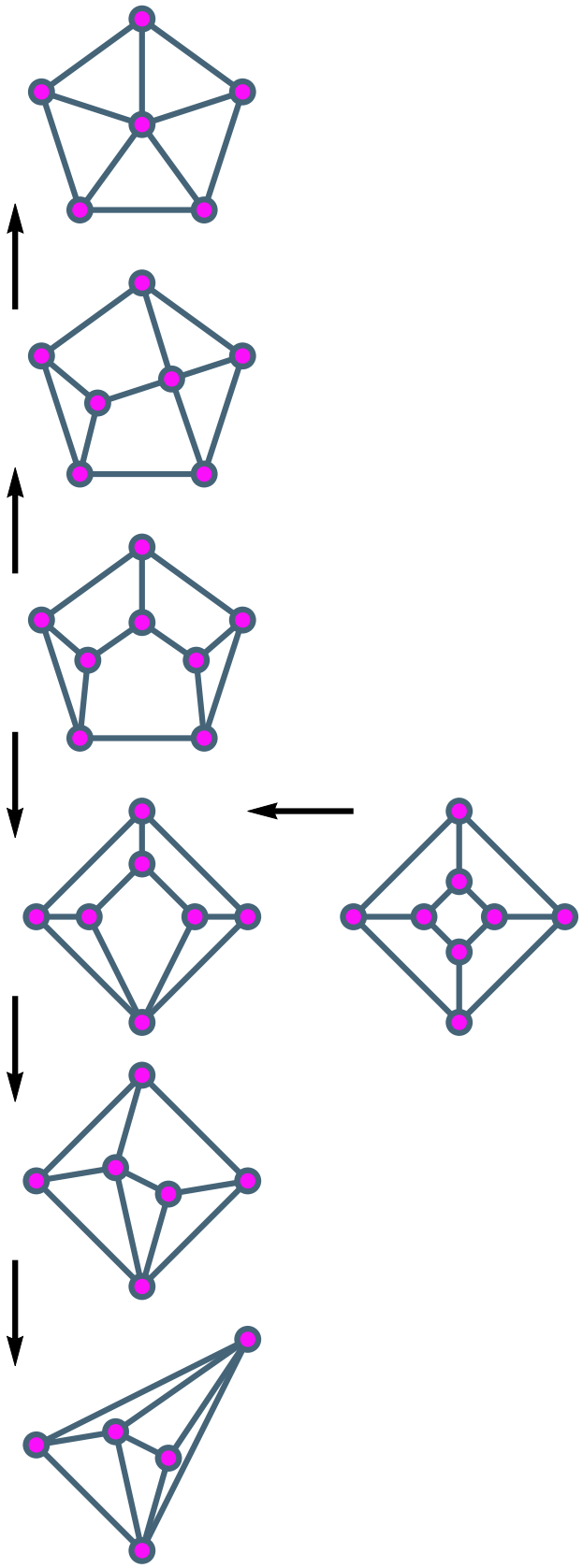


Figure 2.8: The different graphs resulting from vertex contraction and splitting on a 5 spoke wheel, where an arrow represents the vertex contraction operation

2.4 Algorithms

2.4.1 Havel-Hakimi and Erdős-Gallai

This section describes the Havel-Hakimi algorithm and the Erdős-Gallai formula. They give equivalent conditions either of which are necessary and sufficient for the determination of whether a vertex degree sequence can be realised as a simple connected graph.

Havel-Hakimi Algorithm

The Havel-Hakimi algorithm was discovered independently by Havel [Havel, 1955] and Hakimi [Hakimi, 1962]. The algorithm's procedure uses the following recursive property:

Theorem 2.4.1. *Let $D = \{d_1, \ldots, d_n\}$ be a list of n positive integers where $d_i \geq d_j$, for all $i < j$. Then D is a vertex degree sequence which can be realised as a simple connected graph if and only if the vertex degree sequence of $n-1$ integers:*

$$D' = \{d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n\}$$

can be realised as a simple connected graph.

Note that there may be zero values in D', after removing these and sorting the list into descending order, the process can be repeated until one of the following termination conditions is met:

- The reduced D' is a list of zeros. In this case it is true that the original degree sequence is realisable as a simple connected graph.
- The first value in the sorted list is greater than the number of remaining entries, i.e. $d_1 > n - 1$. This implies that the original degree sequence is not realisable as a simple connected graph.

Examples of Havel-Hakimi Algorithm

Example 1 Realisable Degree Sequence

To determine if the degree sequence $D = \{4, 4, 3, 3, 3, 3\}$ is realisable as a simple connected graph we take the highest vertex degree, in this case 4 and decrement 1 from the 4 highest vertex degrees remaining. This gives us the following:

$$\begin{aligned} D =& \{\mathbf{4}, 4, 3, 3, 3, 3\}, \\ \text{decrement first } \mathbf{4}\ & \{\overbrace{4, 3, 3, 3}, 3\}, \\ D' =& \{4-1, 3-1, 3-1, 3-1, 3\}, \\ =& \{3, 2, 2, 2, 3\}, \text{ (unsorted)} \\ =& \{3, 3, 2, 2, 2\}. \end{aligned}$$

We repeat this process with D' and its highest vertex degree 3:

$$\begin{aligned} D' =& \{\mathbf{3}, 3, 2, 2, 2\}, \\ \text{decrement first } \mathbf{3}\ & \{\overbrace{3, 2, 2}, 2\}, \\ D'' =& \{3-1, 2-1, 2-1, 2\}, \\ =& \{2, 1, 1, 2\}, \text{ (unsorted)} \\ =& \{2, 2, 1, 1\}. \end{aligned}$$

Similarly with D'' we take the vertex degree 2:

$$\begin{aligned} D'' =& \{\mathbf{2}, 2, 1, 1\}, \\ \text{decrement first } \mathbf{2}\ & \{\overbrace{2, 1}, 1\}, \\ D''' =& \{2-1, 1-1, 1\}, \\ =& \{1, 0, 1\}, \text{ (unsorted)} \\ =& \{1, 1\}. \end{aligned}$$

Finally we reduce D''':

$$\begin{aligned} D''' =& \{\mathbf{1}, 1\}, \\ \text{decrement first } \mathbf{1} & \overbrace{\{1\}}, \\ D'''' =& \{1-1\}, \\ =& \{0\}. \end{aligned}$$

Which is a list of one zero. Hence, $D = \{4, 4, 3, 3, 3, 3\}$ is realisable as a simple connected planar graph since through recursion we find that D''' is realisable as a simple planar graph.

Example 2 Non Realisable Degree Sequence

To determine if the degree sequence $D = \{5, 4, 4, 4, 2, 1\}$ is realisable as a simple connected graph we take the highest vertex degree, in this case 5 and decrement 1 from the 5 highest vertex degrees remaining. This gives us the following:

$$\begin{aligned} D =& \{\mathbf{5}, 4, 4, 4, 2, 1\}, \\ \text{decrement first } \mathbf{5} & \overbrace{\{4, 4, 4, 2, 1\}}, \\ D' =& \{4-1, 4-1, 4-1, 2-1, 1-1\}, \\ =& \{3, 3, 3, 1, 0\}, \\ =& \{3, 3, 3, 1\}. \end{aligned}$$

We repeat this process with D' and its highest vertex degree 3:

$$\begin{aligned} D' =& \{\mathbf{3}, 3, 3, 1\}, \\ \text{decrement first } \mathbf{3} & \overbrace{\{3, 3, 1\}}, \\ D'' =& \{3-1, 3-1, 1-1\}, \\ =& \{2, 2, 0\}, \text{ (unsorted)} \\ =& \{2, 2\}. \end{aligned}$$

Which is a list of two vertex degrees where the highest vertex degree is greater

than one. Hence, $D = \{5, 4, 4, 4, 2, 1\}$ is not realisable as a simple connected planar graph since through recursion we find that D'' is not realisable as a simple planar graph.

Erdős-Gallai Formula

The Erdős-Gallai formula [Erdös and Gallai, 1960] gives an iterative inequality of summations, which if satisfied by a degree sequence implies that a simple connected graph can be realised by it.

Theorem 2.4.2. *Let $D = \{d_1, \ldots, d_n\}$, be a list of n positive integers where $d_i \leq d_j$, for all $i < j$. Then D is a vertex degree sequence which can be realised as a simple connected graph if and only if:*

$$\sum_{i=1}^{n} d_i \text{ is even}$$

and

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k)$$

holds for $k = 1, \ldots, n$.

Examples of the Erdős-Gallai Formula

We test the realisability of the same two degree sequences ($\{4, 4, 3, 3, 3, 3\}$ and $\{5, 4, 4, 4, 2, 1\}$) given in the examples for the Havel-Hakimi algorithm.

Table 2.2 shows the calculations of different parts of the formula for the degree sequence $\{4, 4, 3, 3, 3, 3\}$. We see in this example that the last column *'inequality satisfied?'* has True in every row. This indicates that the iterative inequality holds for all $k = 1, \ldots, n$, hence the degree sequence is realisable as a simple connected graph.

Table 2.3 shows the same calculations for the degree sequence $\{5, 4, 4, 4, 2, 1\}$ in this case we see that the inequality is not satisfied for $k = 3$ or $k = 4$. The formula requires that the inequality holds for every $k = 1, \ldots, n$, since it does not, the degree sequence is not realisable as a simple connected graph.

k	$\sum_{i=1}^{k} d_i$	$k(k-1)$	$\sum_{i=k+1}^{n} \min(d_i, k)$	inequality satisfied?
1	4	0	5	True
2	8	2	8	True
3	11	6	9	True
4	14	12	6	True
5	17	20	3	True
6	20	30	0	True

Table 2.2: Calculations for the Erdős-Gallai Formula when $D = \{4, 4, 3, 3, 3, 3\}$

k	$\sum_{i=1}^{k} d_i$	$k(k-1)$	$\sum_{i=k+1}^{n} \min(d_i, k)$	inequality satisfied?
1	5	0	5	True
2	9	2	7	True
3	13	6	6	**False**
4	17	12	3	**False**
5	19	20	1	True
6	20	30	0	True

Table 2.3: Calculations for the Erdős-Gallai Formula when $D = \{5, 4, 4, 4, 2, 1\}$

2.4.2 Duijvestijn and Federico

The first enumeration algorithm which computed significantly large numbers of polyhedra (greater than that which could be processed by hand), was implemented by Duijvestijn and Federico [Duijvestijn and Federico, 1981]. This algorithm used Tutte's Wheel Theorem as the fundamental process by which it calculated results. They published counts of polyhedra with up to 12 vertices. Their results are given in Table 2.4. The * entries in Table 2.4 are approximate values. The table is symmetric as the number of polyhedra with x vertices and y faces is the same as the number of polyhedra with y vertices and x faces, this is a consequence of duality from Steinitz' Theorem (2.3.3).

This approach was the first exhaustive computer program for enumerating polyhedral graphs and the numbers generated have since been verified independently by the program plantri by Brinkmann and McKay reviewed in Subsection 2.4.5 and by the results generated from the research in this thesis.

Vertices \Faces	4	5	6	7	8	9	10	11	12
4	1								
5		1	1						
6		1	2	2	2				
7			2	8	11	8	5		
8			2	11	42	74	76	38	14
9				8	74	295	633	768	558
10				5	76	633	2635	6134	8822
11					38	768	6134	25626	64439
12					14	558	8822	64439	268394
13						219	7916	104213	706770*
14						50	4442	112082	1259093*
15							1404	79773	1552824*
16							233	36528	1334330*
17								9714	786625*
18								1249	304087*
19									69564*
20									7595
Total	1	2	7	34	257	2605	32300	440564	6363115*

Table 2.4: Numbers of topologically distinct polyhedra with given numbers of vertices and faces (* entries are approximate values).

2.4.3 Hopcroft and Tarjan

Simple planar graphs are a subset of all simple graphs. If generating a set containing all simple planar graphs which includes some non-planar graphs, then the process of efficiently removing the non-planar graphs from the set is required. Planarity testing of a graph has been shown to be possible in linear ($O(n)$) time. The first linear testing algorithm was given by Hopcroft and Tarjan [Hopcroft and Tarjan, 1974].

The core principle behind the testing is if all biconnected (2-vertex connected) subgraphs of a graph are planar, then the graph is planar. Consequently, any planar subgraphs that are connected solely by two vertices can also be embedded in the plane.

2.4.4 Hopcroft and Wong

If a generating function which guarantees completeness of results, but not unique (isomorphism free) or planar results, then not only is the speed of planarity testing important, but also the speed at which isomorphism testing between graphs can be computed. Subsequently with Wong, Hopcroft gave a linear time algorithm for isomorphism testing [Hopcroft and Wong, 1974]. This allows two planar graphs to be input and in linear time (with respect to the number of edges $|E|$) return a binary result of the existence of an isomorphism.

2.4.5 Brinkmann and McKay

The state of the art in enumeration of polyhedral graphs is a program plantri by Brinkmann and McKay [Brinkmann et al., 2005]. Plantri is an enumeration algorithm implemented in C which can generate certain types of planar graphs. They also have a specific focus on *quadrangulations of the sphere*, this is 2-connected planar graphs with solely 4-sided faces. This has given the total numbers of 4-regular 3-connected planar graphs up to 34 vertices, making significant improvements on the previously known results of up to 24 vertices generated by Dillencourt [Dillencourt, 1996].

This work has been cited in many recent articles which have developed theory on 4-regular planar graphs. A recent proof by Bau [Bau, 2016] shows how a 4-regular planar graph with minimum vertex degree three is 3-connected if there is no separating quadrilateral. That is, a cycle of four edges that disconnects the graph. This is not a logical biconditional since 3-connected 4-regular planar graphs can have a separating quadrilateral. An example of this is a cuboid graph where two cubes are joined at one face. However, it does show that if there is no separating quadrilateral then 3-connectivity is implied.

A recent PhD thesis by Kapolnai [Kápolnai, 2014] applies the plantri algorithm to the enumeration of primary and secondary equilibrium classes of natural shapes. An equilibrium class of a natural shape is method of classifying a polyhedron by its stable and unstable points. The relative 3-dimensional coordinates determined by a force directed system where vertices are repulsed and

attracted to each other based on whether there are edges connecting them. A stable point is a vertex which would return to its coordinate if recalculated after a small perturbation to its position. An unstable point is a vertex which would change its coordinate if recalculated after a small perturbation to its position. A primary equilibrium class is the number of stable and unstable points the polyhedron contains. A secondary equilibrium class is a refinement of the primary equilibrium classes into related quadrangulations of the sphere. These quadrangulations are planar graphs with quadrilateral faces. This is an encoding that compares topologies of shapes and the number of different colourings that planar graphs can have.

Lehel [Lehel, 2006] uses similar theory to that used in the plantri algorithm to correct a different 4-regular graph enumeration algorithm first proposed by Manca [Manca, 1979]. This corrected algorithm can generate all 4-regular planar graphs by a sequence of operations on the octahedron graph. The required correction added an additional possible operation.

As recently as August 2017 work by Noy [Noy et al., 2017] has been published citing the plantri algorithm applying the theory to the first recursive technique for counting the number of 4-regular labelled planar graphs. This counts all permutations of labelling 4-regular planar graphs such that no two adjacency matrices are the same.

The following chapters 3 and 4 will present the theory and describe the process by which the algorithm contributed by this thesis enumerates 4-regular polyhedral graphs. At this point however, it is useful highlight that the Brinkmann and McKay program plantri enumerates results by starting with a base set of graphs and performing multiple mutation operations in order to exhaustively generate all graphs. The main distinction between the plantri program and algorithm presented in this thesis is that all graphs generated at any stage by this algorithm have immutable vertices and edges, as such no edge can be removed and no vertex degree can change. Therefore, the problem subdivision that can be utilised due to the permanence of these graph invariants is distinct from that in either the Duijvestijn and Federico or the Brinkmann and McKay programs.

2.5 Summary

The theory presented in this chapter is provided to highlight the research used to inform the approach taken in this thesis.

Section 2.2 (Polyhedra) provides the background on polyhedra and it's geometric properties. This gives insight into where the motivation for this work stems and how the enumeration directly corresponds to three dimensional polytopes.

Section 2.3 (Planar Graphs) introduces the graph theoretic definitions and notation which are used throughout the rest of the thesis. We introduce also various work at the interface between polyhedral geometry and graph theory to show other avenues of investigation in related areas and help provide context for the theory developed in Chapter 3. Cauchy's proof of Euler's Formula in Subsection 2.3.4 shows an application of planar embeddings to prove this fundamental formula. Tutte's Wheel Theorem (Subsection 2.3.5) is reviewed to give background theory of other enumeration algorithms, although this theorem is not applied directly to the algorithm in this thesis, it is of great importance in understanding the current body of knowledge in planar graph enumeration. The Tutte Embedding (Subsection 2.3.6) provides a method for realising a fixed geometric planar embedding of a 3-connected planar graph knowing only the topological embedding of a single face.

Section 2.4 reviews algorithms in historical order that either enumerate planar graphs or can be applied as part of the enumeration process. The Havel-Hakimi Algorithm (Erdős-Gallai Formula) is applied during the enumeration algorithm designed in this thesis and is of great importance given the problem subdivision approach taken. Duijvestijn and Federico presented the first algorithm programmed for a computer giving directly relevant results. The planarity testing algorithm by Hopcroft and Tarjan in Subsection 2.4.3 and subsequent planar graph isomorphism testing algorithm by Hopcroft and Wong in Subsection 2.4.4 give insight into the most computationally efficient methods known for these problems. This is of interest since if proposing an enumeration algorithm which is exhaustive but does not guarantee results that are unique or planar these methods would have the lowest computational costs for post-processing to achieve the

desired data. Finally in Subsection 2.4.5 we present the current leading contributor to results of 4-regular polyhedral graphs the plantri program by Brinkmann and McKay. This work is reviewed not only to highlight the novelty of the contribution in this thesis but also through discussing other papers which have cited it, show the relevance of current research on planar graph enumeration algorithms.

Chapter 3

4-regular 3-connected Planar Graphs

3.1 Introduction

This chapter introduces the theory used to describe and enumerate 4-regular, 3-connected planar graphs. Specifically we describe the process of decomposing a 4-regular, 3-connected planar graph into a pair of 2-vertex connected, 3-edge connected planar graph embeddings. The motivation behind the study of these particular graphs is that they can be partially connected while efficiently maintaining planarity via a pair of labelled adjacency matrices. This is a result of the duality between the pair of graphs and the guarantee that it can reconstruct a 4-regular, 3-connected planar graph. Either of these graph embeddings can determine completely the unique 4-regular, 3-connected planar graph to which this pair of embeddings corresponds. We also introduce the notation and definitions used to describe these graphs and their properties throughout the remainder of the thesis. We prove certain properties regarding the graph decomposition and show how these can take advantage of the enumeration process to filter results early, especially if polyhedral graphs are the only desired output.

From Theorem 2.3.1 we know the vertex-connectivity of a graph $\kappa(G)$ is always less than or equal to the edge-connectivity of a graph $\lambda(G)$. We simplify our descriptions of graphs by the following convention: If the graphs are 3-vertex

connected and consequently 3-edge connected, it is sufficient to define these as 3-connected. If the graphs are 2-vertex connected, 3-edge connected they are stated as such. If the graphs are 2-vertex connected but not 3-vertex connected we say they are **exactly** 2-vertex connected, the implication being that we can always find two vertices whose removal would disconnect the graph. Recall k-vertex connectivity requires that no $k-1$ set of vertices can disconnect a graph, similarly with k-edge connectivity for edges. Therefore a k-connected graph is $k-1$ connected and so on.

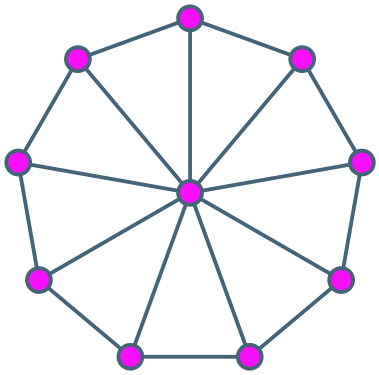
This research is solely concerned with planar embeddings, therefore unless otherwise stated an embedding is always assumed to be planar. For a graph G we denote an embedding of this graph as $\Gamma(G)$.

Given a planar graph G, we define a dual graph embedding of G with respect to an embedding Γ as $\Gamma(G)^*$. Similarly the dual embedding of $\Gamma(G)^*$, $(\Gamma(G)^*)^* = \Gamma(G)$. We define the graph of an embedding Γ as $(\Gamma)'$, and note that for such a graph G with an embedding Γ, $(\Gamma(G))' = G$.

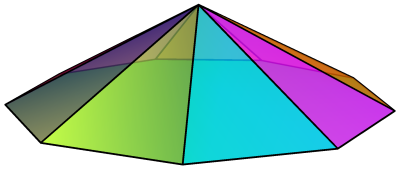
3.2 3-connected Graph Embeddings

If a planar graph is 3-connected then from Steinitz' theorem (Subsection 2.3.3) it has a single embedding, the dual graph will also be 3-connected and hence has a single embedding. This graph can be self dual. One set of examples of self dual 3-connected planar graphs are the wheel graphs. A wheel graph with $|V|$ vertices has one vertex with degree $|V|-1$ and $|V|-1$ vertices with degree 3. This can be shown graphically as a regular $|V|-1$ sided polygon with all vertices joined at a central vertex, creating a shape similar to a wheel with 'spokes'. Figure 3.1 shows an example of a wheel graph with 10 vertices.

The algorithm designed in this thesis enumerates all 4-regular, 3-connected planar graphs. The algorithm achieves this by computing the duals of these graphs, that is 3-connected planar graphs where every face is quadrilateral. All planar embeddings with solely even faces such that the number of sides about each face are an even number are bipartite. Since simple planar graphs cannot have



(a) A self dual planar graph



(b) A nonagonal pyramid

Figure 3.1: A wheel graph with 10 vertices and its polyhedral representation

faces of degree 2, we can constrain the degree sequences of the bipartitions such that if a 3-connected bipartite planar graph is found, all faces in any embedding will be quadrilateral. These constraints are defined in Chapter 4 Subsection 4.2.

3.3 Exactly 2-vertex, 3-edge connected Graph Embeddings

Exactly 2-vertex, 3-edge connected planar graphs can have multiple distinct embeddings. The duals of these embeddings will also be exactly 2-vertex, 3-edge connected. This is Theorem 3.5.1 and is proved later in this chapter when introducing a test for 3-connectedness. Each dual of an exactly 2-vertex, 3-edge connected graph can also have multiple distinct embeddings. Figure 3.2 shows two distinct embeddings, Embedding 1 is shown twice with two layouts. When stating this we take care to make sure that the embeddings of the graphs are not equivalent. In these figures it is clear, as there is an outer face of degree 6 in Embedding 1 shown in Figure 3.2 (a) and no internal face of degree 5. Similarly for Embedding 2 in Figure 3.2 (c) there is an outer face of degree 5 and no internal face of degree 6. Therefore the graph of the dual for Embedding 1 will have a vertex of degree 6 and no vertex of degree 5. Similarly the graph of the dual for Embedding 2 will have a vertex of degree 5 and no vertex of degree 6, hence the

embeddings cannot be equivalent.

However, with Figures 3.2 (a) and 3.2 (b) although there is a different outer face for each, the embeddings are the same. This can be seen by comparing the ordered adjacency lists about each vertex or the face cycles. The face cycles are the cyclic lists of vertices that make up the faces. For example the outer face cycle in Figure 3.2 (a) is:

$$(v_1, v_2, v_4, v_6, v_7, v_5)\,.$$

The outer face cycle for Figure 3.2 (b) is:

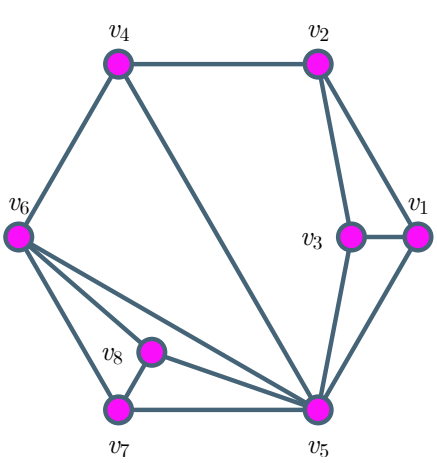
$$(v_4, v_6, v_5)$$

However, the outer face cycle from Figure 3.2 (a) exists in Figure 3.2 (b) and the outer face cycle from Figure 3.2 (b) exists in Figure 3.2 (a). Since all the face cycles in both embeddings are the same, the embeddings are equivalent.

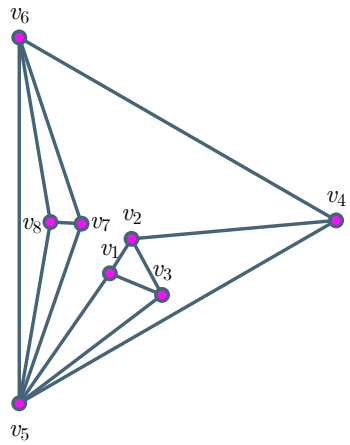
3.4 Decomposition

The key concept behind this algorithm is the decomposition of duals of 4-regular, 3-connected planar graphs. The bipartitions are used to deconstruct a 4-regular, 3-connected planar graph into two 2-vertex, 3-edge connected graph embeddings. The embeddings are represented by two adjacency matrices, one for each bipartition.

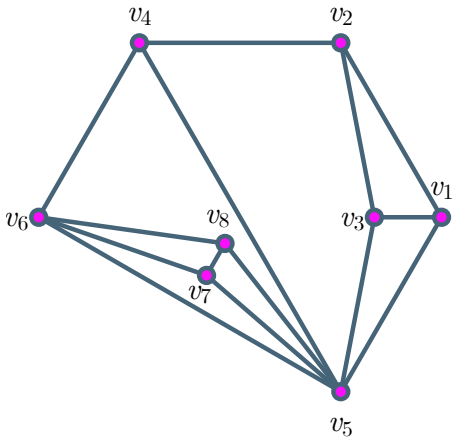
The graph in Figure 3.3 has 12 vertices, 6 in each bipartition. The degree sequences of both partitions is $\{4, 4, 3, 3, 3, 3\}$. We label the vertices such that the first bipartition V_1 has the vertices $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ (shown as pink circles) and the second bipartition V_2 has the vertices $\{v_7, v_8, v_9, v_{10}, v_{11}, v_{12}\}$ (shown as lime squares).



(a) Embedding 1



(b) Alternative layout of Embedding 1



(c) Embedding 2

Figure 3.2: Two distinct embeddings of an exactly 2-vertex, 3-edge connected planar graph

The edge set for this graph is:

$$\begin{aligned} E = \{&\{v_1, v_9\}, \{v_1, v_7\}, \{v_1, v_{11}\}, \{v_1, v_8\}, \\ &\{v_2, v_7\}, \{v_2, v_{10}\}, \{v_2, v_{12}\}, \{v_2, v_8\}, \\ &\{v_3, v_7\}, \{v_3, v_9\}, \{v_3, v_{10}\}, \\ &\{v_4, v_7\}, \{v_4, v_{11}\}, \{v_4, v_{12}\}, \\ &\{v_5, v_8\}, \{v_5, v_9\}, \{v_5, v_{10}\}, \\ &\{v_6, v_8\}, \{v_6, v_{11}\}, \{v_6, v_{12}\}\}. \end{aligned}$$

We have ten quadrilateral faces in this graph, each has two vertices from V_1

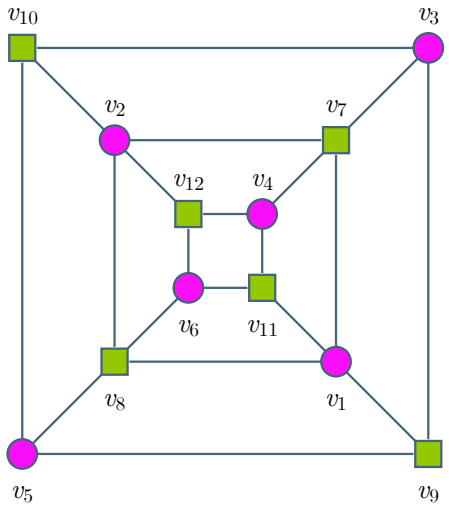


Figure 3.3: Bipartite graph with only quadrilateral faces

and two vertices from V_2. The faces must be bipartite, therefore the four edges which share this face must all join one vertex from V_1 with one vertex from V_2. Consequently we have a pair of vertices in V_1 which are adjacent to the same pair of vertices in V_2. Furthermore, each vertex must exist in the same number of faces as its vertex degree. Hence a vertex v_i in V_1 with degree d_i must have d_i pairs of vertices in V_2, each with a vertex in V_1 with which it shares a face.

As the graph is 3-connected, no two faces can have the same pair of opposing vertices (vertices from the same bipartition). If a pair of opposing vertices existed in two faces, one of these faces would be internal and one external. This would imply that the pair of vertices would disconnect the graph, hence the graph would not be 3-connected. Therefore the pairs of vertices in V_1 sharing a face are unique and likewise for the pairs of vertices in V_2.

Therefore we can represent all of this information in two matrices of sizes $|V_1| \times |V_1|$ and $|V_2| \times |V_2|$. In equations 3.1 and 3.2 the 6×6 matrices A and B show this for the example graph in Figure 3.3, if a face exists with the vertices v_w, v_x, v_y, v_z, where $v_w, v_x \in V_1$ and $v_y, v_z \in V_2$. We assign to the matrix entries $A_{w,x}$ and $A_{x,w}$ the values y and z. Taking into account the indexing of the matrix B, let $i = y - |V_1|$ and $j = z - |V_1|$. We assign to the matrix entries $B_{i,j}$ and $B_{j,i}$ the values w and x.

For the example graph we have a face with vertices v_1, v_7, v_3, v_9 (in cyclic

order). Therefore the matrix A has $A_{1,3} = 9$, $A_{3,1} = 7$, $B_{1,3} = 3$ and $B_{3,1} = 1$. There is no requirement that the symmetric entries in a matrix be assigned in order, for example if $A_{1,3} = 7$ and $A_{3,1} = 9$ this would be equivalent.

$$A = \begin{pmatrix} 0 & 0 & 9 & 11 & 9 & 11 \\ 0 & 0 & 7 & 7 & 8 & 8 \\ 7 & 10 & 0 & 0 & 9 & 0 \\ 7 & 12 & 0 & 0 & 0 & 11 \\ 8 & 10 & 10 & 0 & 0 & 0 \\ 8 & 12 & 0 & 12 & 0 & 0 \end{pmatrix} \tag{3.1}$$

$$B = \begin{pmatrix} 0 & 0 & 3 & 2 & 4 & 2 \\ 0 & 0 & 5 & 2 & 6 & 2 \\ 1 & 1 & 0 & 3 & 0 & 0 \\ 3 & 5 & 5 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 4 \\ 4 & 6 & 0 & 0 & 6 & 0 \end{pmatrix} \tag{3.2}$$

If taken as binary matrices such that any non-zero values are 1 and zero values are 0, the matrices are symmetric. These would be the adjacency matrices of the graphs of the decomposed embeddings. There is also a cyclic pattern about the row and column entries for any given vertex. These indicate not only their adjacencies, but also the order in which they are embedded about the vertex.

Distinct embeddings Γ of 2-vertex, 3-edge connected graphs can be defined by their ordered adjacency lists. Two embeddings are isomorphic if there exists a one to one mapping between the vertices. The ordered adjacency lists are cyclic and as such the cycle in an adjacency list can be shifted in order to achieve an exact matching. Similarly since orientation reversal is permitted, an entire embedding can have its ordered adjacency list reversed and is still isomorphic.

We cannot use a simple isomorphism test of the dual graphs of a set of embeddings since the dual of an exactly 2-vertex, 3-edge connected graph is a 2-vertex, 3-edge connected graph. As such there exist cases where multiple distinct em-

beddings

$$\Gamma_1, \Gamma_2 \dots$$

of a single graph G have dual graphs

$$(\Gamma_1(G))^*, (\Gamma_2(G))^*, \dots$$

which are isomorphic, however, since the embeddings are distinct, the 3-connected planar graphs with solely quadrilateral faces (denoted by $Q(\Gamma)$) that can be reconstructed from either an embedding or its dual would not be isomorphic.

$$Q(\Gamma_1) \equiv Q((\Gamma_1)'), Q(\Gamma_2) \equiv Q((\Gamma_2)'), \dots$$

There is an exception to this in the case where the embedding is self dual and as such the graph from the dual of the embedding is isomorphic to the original graph. Where this is the case for two distinct embeddings on G, Γ_i and Γ_j where:

$$(\Gamma_i)' \equiv \Gamma_j \text{ and consequently we find } G \equiv (\Gamma_i(G))^* \equiv (\Gamma_j(G))^*,$$

however since, $\Gamma_i \not\equiv \Gamma_j$, this results in two isomorphic 4-regular 3-connected graphs reconstructed from two distinct embeddings on the same exactly 2-vertex, 3-edge connected planar graph.

When considering this decomposition graphically, the graph representing matrix A is illustrated in Figure 3.4. The original graph is shown with dashed purple edges, the decomposed graph for V_1 is overlaid with solid grey edges. Any pair of non-zero entries $A_{i,j}$ and $A_{j,i}$ are represented by a solid edge. We can see every quadrilateral face in the initial graph has an edge across it, joining the two opposing vertices in V_1. The outer face has a curved edge to respect the ordered adjacencies of the graph embedding. In Figure 3.5 we show the two decomposed graph embeddings that are equivalent to matrices A and B.

In this example, not only are the two decomposed graphs isomorphic, but so too are their embeddings. These graphs are exactly 2-vertex, 3-edge connected graphs. As for the graph in Figure 3.5 (a) we can disconnect the graph with the

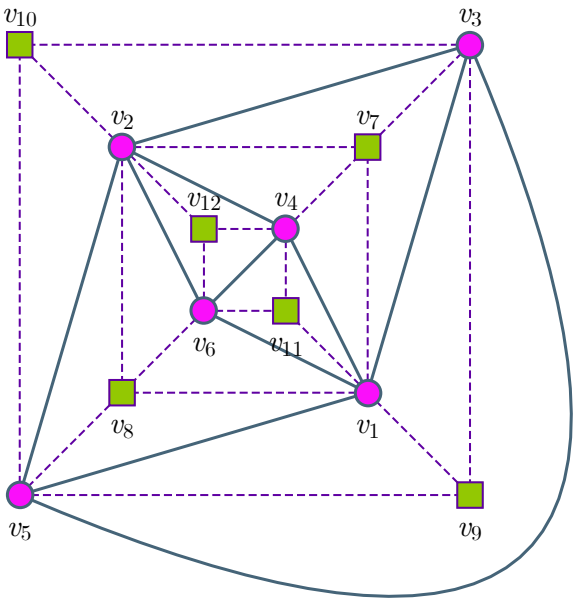
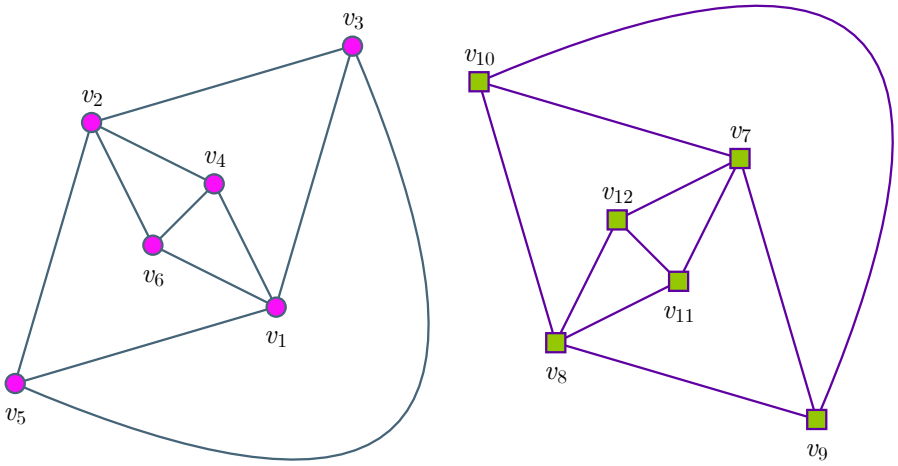


Figure 3.4: Example graph with the face sharing vertices in V_1 overlaid

pair of vertices v_1 and v_2 and similarly for Figure 3.5 (b) we can disconnect the graph with the pair of vertices v_6 and v_7.



(a) Decomposed graph embedding for V_1

(b) Decomposed graph embedding for V_2

Figure 3.5: Decomposed graph embeddings

3.5 3-vertex Connectivity Test

This test requires additional values to be added to the adjacency matrices for the decomposed 2-vertex, connected 3-edge connected graph embeddings. We intro-

duce this procedure by first proving that exactly 2-vertex, 3-edge connectivity is constant with respect to duality.

Theorem 3.5.1. *If a simple planar graph is exactly 2-vertex, 3-edge connected, then the graphs of the duals of all embeddings of that graph are also exactly 2-vertex, 3-edge connected simple planar graphs.*

This theorem is fundamental to the connectivity test and its proof requires the following four theorems:

Theorem 3.5.2. *A dual G^* of an exactly 2-vertex, 3-edge connected simple planar graph G is at most 2-vertex connected ($\kappa(G^*) \leq 2$).*

Proof

If a dual graph G^* is 3-connected, then by Steinitz' Theorem, G is 3-connected (Theorem 2.3.3). □

Theorem 3.5.3. *A simple 3-edge connected planar graph has minimum face degree three.*

Recall from Theorem 2.3.1, since the edge-connectivity $\lambda(G) \geq 3$, then the minimum vertex degree $\delta(G)$ is three.

Proof

If a connected planar graph with minimum vertex degree three has a face of degree one, it would have a single self-edge with one vertex (Figure 3.6), which would mean the graph is not simple. If a connected planar graph with minimum vertex degree three has a face of degree two, it is either drawn by two self-edges off of the same vertex (Figure 3.7 (a)) or by two vertices sharing two edges (Figure 3.7 (b)). Either of these would imply the graph is not simple. □

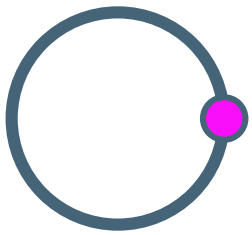
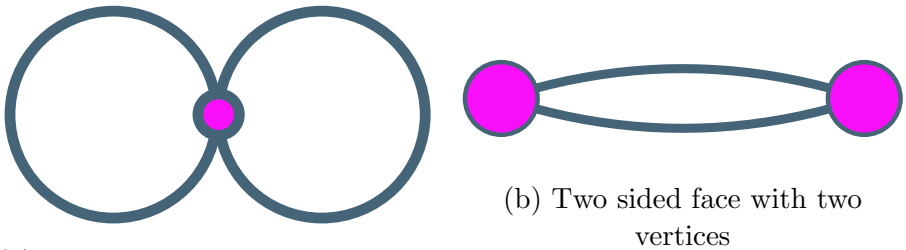


Figure 3.6: Planar graph with one sided face



(a) Two sided face with one vertex

(b) Two sided face with two vertices

Figure 3.7: Planar graphs with two sided faces

Theorem 3.5.4. *A simple planar graph with minimum vertex degree three is 3-edge connected if and only if no two faces share two edges.*

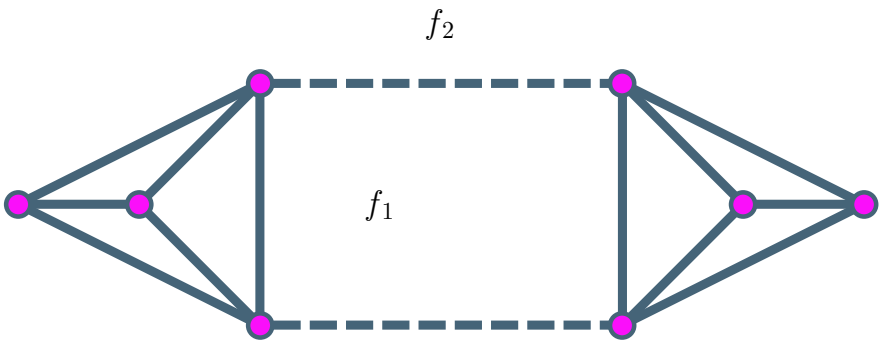


Figure 3.8: 2-edge connected planar graph

Proof

If two faces f_1 and f_2 share two edges, these would form an edge cut set, hence the graph would be 2-edge connected. This is shown in Figure 3.8 where the two edge cut set is dashed. If no two faces share more than one edge, then the removal of two edges from the graph causes one of the following three cases to occur:

Case 1: Removal of an edge shared by faces f_1 and f_2 and removal of an edge shared by two faces f_3 and f_4 which are not adjacent to f_1 or f_2 (an example of which is shown in Figure 3.9).

Case 2: Removal of an edge shared by faces f_1 and f_2 and removal of an edge shared by two faces f_3 and f_4 where f_1 and f_2 are both adjacent to f_3 (an example of which is shown in Figure 3.10).

Case 3: Removal of an edge shared by faces f_1 and f_2 and removal of an edge shared by faces f_1 and f_3 (an example of this is shown in Figure 3.11).

For case 1 consider the removal of the edge shared by faces f_1 and f_2. We find that there is one less face in the graph, the new face having degree $d(f_1) + d(f_2) - 2$. We can still determine the cycle of vertices about this face and there is no subgraph disconnected. If we repeat this operation for f_3 and f_4 we find that there is again one less face and a new face of degree $d(f_3) + d(f_4) - 2$ and similarly to the first operation, there is no disconnected subgraph.

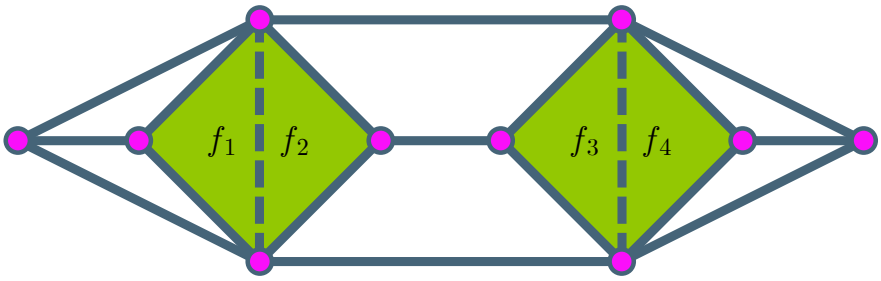


Figure 3.9: Planar graph highlighting Case 1

For case 2 as with case 1 if we first remove the edge shared by f_1 and f_2 we find one less face in the graph and a new face with degree $d(f_1) + d(f_2) - 2$. If the edge between faces f_3 and f_4 is also removed, although f_3 is adjacent to both f_1 and f_2, we can still determine the cycle of vertices about the new face of degree $d(f_3) + d(f_4) - 2$. Hence, there is no disconnected subgraph.

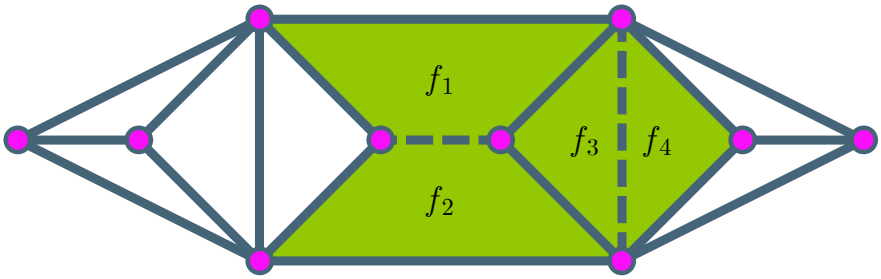


Figure 3.10: Planar graph highlighting Case 2

For case 3 first remove the edge shared by f_1 and f_2. This is the initial operation performed in cases 1 and 2, consequently we find the resulting graph has one less face with a new face of degree $d(f_1) + d(f_2) - 2$. If we now remove the edge shared by f_1 and f_3 we find that there is one less face and now a face of degree $d(f_1) + d(f_2) + d(f_3) - 4$. There is no vertex with all edges removed and no disconnected subgraph. All vertices affected by these operations form one

cycle about the new face. Therefore, we cannot disconnect a simple planar 3-edge connected graph with minimum vertex degree three if every pair of faces share at most one edge. □

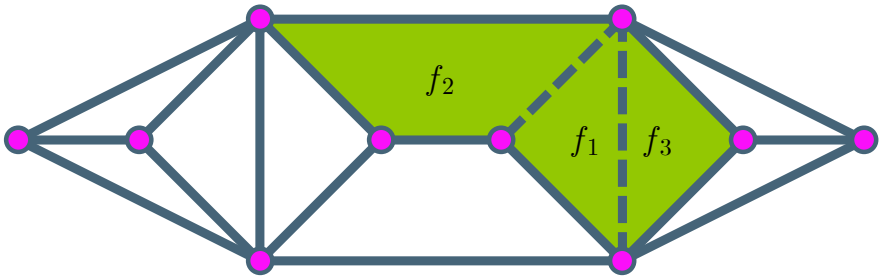


Figure 3.11: Planar graph highlighting Case 3

Theorem 3.5.5. *A dual graph G^* of any 2-vertex, 3-edge connected simple planar graph G is a simple planar graph with no two faces sharing more than one edge.*

Proof

Since G is a 3-edge connected planar graph, then from Theorem 3.5.4 we have that no two faces share more than one edge. Since G is 2-vertex connected, we have that no face can have a vertex in its cycle more than once. Therefore G^* has no multi-edges (two vertices sharing more than one edge) and no self-looping edges (a vertex with an edge to itself). Hence G^* is simple.

Since G is simple no two vertices share more than one edge, consequently G^* has no two faces sharing more than one edge. □

Lemma 3.5.6. *A dual graph G^* of any 2-vertex, 3-edge connected simple planar graph G is a 3-edge connected simple planar graph*

Proof

From Theorem 3.5.3 we know that the minimum face degree of G is three and therefore G^* has minimum vertex degree three. Consequently we can apply Theorems 3.5.4 and 3.5.5 to give us $\lambda(G^*) = 3$. □

Finally with proof of Lemma 3.5.6 we can now proceed to prove Theorem 3.5.1.

Since G^* is simple, if there exists a vertex that disconnects the graph into multiple subgraphs there must exist a face that contains that vertex in the face

cycle twice. This would imply that four edges from this vertex draw the face. If this is the case then G would also have a vertex with four edges drawn existing in a single face. Consequently G would be 1-vertex connected, which is a contradiction. Hence G^* must also be 2-vertex connected.

Therefore we have that any dual of an exactly 2-vertex, 3-edge connected simple planar graph is an exactly 2-vertex, 3-edge connected simple planar graph, thus proving Theorem 3.5.1. □

As a consequence of Theorem 3.5.1, we can propose a 3-connectivity test on subgraphs in the enumeration algorithm. When considering a cycle about a vertex in V_1, if any pair of vertices v_i, v_j in V_2 are shared by more than one vertex in V_1 and they do not draw a face with these vertices in V_1, then any resultant decomposed graph will be exactly 2-vertex connected. This can be checked in the enumeration process by having a placeholder in matrix B for any vertices v_i, v_j which are not cyclically adjacent about a vertex but share edges with it. If at any stage vertices with these placeholders also share a face, or a placeholder is required for a pair which already share a face, then any resultant graphs must be exactly 2-vertex connected. This information could be stored and further such checks in later iterations would not be required.

3.6 Summary

The theory in this chapter highlights the relationship between 2-vertex, 3-edge connected graph embeddings and the duals of 4-regular 3-connected graphs, and provides the perspective from which the enumeration algorithm is designed. The matrices in the graph decomposition give some insight into the form in which data is calculated and used during computation.

The additional vertex connectivity was introduced as a possible filter to distinguish between results which are 3-connected or exactly 2-vertex, 3-edge connected. Since this test guarantees exactly 2-vertex, 3-edge connectivity without connecting the entire graph it permits filtering earlier in the enumeration search tree (introduced in Chapter 4). This could be used to only enumerate the 3-

connected planar (polyhedral) graphs.

With these concepts introduced, Chapter 4 will show how the problem of enumerating all duals of 4-regular, 3-connected planar graphs with a given number of vertices $|V|$ is divided into subproblems of enumerating all duals with distinct pairs of degree sequences. This, with an overview of the parallelisation in the algorithm, explains how other data structures further constrain computation.

Chapter 4

Algorithm Design

4.1 Introduction

This chapter will describe the various steps of the main enumeration algorithm. We begin by describing the subdivision of the problem for enumerating all 4-regular, 3-connected planar graphs with $|V|$ faces. In the previous chapter it was explained that we actually take the dual of 4-regular, 3-connected planar graphs (which consequently would have $|V|$ vertices).

We first define the number of vertices $|V|$ of the graphs which we are enumerating. From this all permissable combinations of vertex degrees that can exist for the bipartitions of graphs with $|V|$ vertices are found, these are the pairs of degree sequences. For any initial starting run of the algorithm, a pair of degree sequences are the parameters provided to enumerate the graphs. Degree sequences are graph invariants, hence results generated from distinct pairs can never have overlapping isomorphisms. Following these definitions and an overview of the computation process, further properties are introduced which can be inferred in the subgraphs as they are calculated. These properties restrict the number of options considered during recursion and consequently reduce the size of the search space for results, this is discussed in Section 4.7. All restrictions presented maintain exhaustive enumeration of all required data.

Algorithm Overview

At each iteration a subgraph is provided with edges yet to be connected. The steps at each iteration of algorithm are processed in the following order:

- Take a vertex from the first bipartition V_1 and consider possible adjacencies to vertices in the second bipartition V_2 (combinations of which are restricted by the enumeration process).
- For each possible set of adjacencies, consider all possible embeddings in the subgraph.
- For each embedding, verify that the remaining unconnected vertices and edges still have potential adjacencies that could lead to a complete graph.
- Across the newly generated set of subgraphs generated, verify uniqueness and return for further iterations.

The most computationally expensive step in the enumeration algorithm is actually the initial process of considering all possible adjacencies of a vertex in V_1 to vertices in V_2. This is due to there being large amounts of combinations which are considered but then immediately rejected in an iterative process. By removing these combinations initially the computational cost overall is reduced. The complexity of these possible adjacencies is discussed further in Section 4.5 (Combinations and Banning).

Throughout the chapter each step will be given a detailed explanation highlighting where optimisations have been made.

4.2 Definitions

The following is a definition of these graphs and the notation used to describe computation.

For the purpose of this algorithm we define a resultant graph as a simple undirected bipartite planar 3-connected graph whose dual is 4-regular. We denote

this graph as $G(V, E)$, a set of vertices $V = \{v_1, v_2, \ldots, v_{|V|}\}$ and edges $E \subset V \times V$. Since G is bipartite, we can find the vertex sets V_1 and V_2. Such that

$$V = V_1 \cup V_2,$$

where

$$V_1 \cap V_2 = \emptyset,$$

which leads to

$$|V_1| + |V_2| = |V|.$$

Although the graph is undirected, as it is simple and has no repeated edges, it is useful to order the pairs of vertices in the edges as:

$$E \subseteq V_1 \times V_2.$$

Hence for an edge $e = \{v_a, v_b\} \in E$, we have that the first vertex in the edge $v_a \in V_1$ and the second vertex in the edge $v_b \in V_2$.

We define the vertex degree of a vertex v in the graph as $d(v)$. The vertices are ordered

$$V_1 = \{v_1, v_2, \ldots, v_{|V_1|}\}$$

and

$$V_2 = \{v_{|V_1|+1}, v_{|V_1|+2}, \ldots, v_{|V|}\}.$$

The input to the algorithm is a pair of ordered degree sequences D_1 and D_2. These are listed in descending order and correspond to the vertices in V_1 and V_2. Therefore the degree sequences are:

$$D_1 = \{d(v) : v \in V_1 \text{ where } d(v_i) \geq d(v_j), i < j\},$$

and

$$D_2 = \{d(v) : v \in V_2 \text{ where } d(v_i) \geq d(v_j), i < j\}.$$

Handshaking Lemma

The Handshaking Lemma states that the sum of the vertex degrees is equal to twice the number of edges. Therefore for a graph $G(V, E)$

$$\sum_{i=1}^{|V|} d(v_i) = 2|E|.$$

With the Handshaking Lemma we have that:

$$\sum_{d \in D_1} D_1 = \sum_{d \in D_2} D_2 = |E|.$$

The minimum vertex degree of a graph is denoted as $\delta(G)$. Since the graph is 3-connected:

$$\delta(G) \geq 3.$$

As the dual graphs are 4-regular, we have that all faces are quadrilateral. Therefore since every edge is contained in exactly two quadrilateral faces and each quadrilateral face contains exactly four edges, twice the number of edges is four times the number of faces:

$$2|E| = 4|F|,$$

hence from Euler's formula we have

$$|V| - |E| + \frac{|E|}{2} = 2,$$

$$|E| = 2|V| - 4.$$

To list all possible pairs of degree sequences for a given number of vertices $|V|$, we consider the different sizes of V_1 and V_2. The maximum size of a partition is

$$\left\lfloor \frac{2|V| - 4}{3} \right\rfloor,$$

with the minimum partition size

$$|V| - \left\lfloor \frac{2|V| - 4}{3} \right\rfloor.$$

These sizes are the maxima and minima of the partitions V_1 and V_2, however they're assignment as either V_1 and V_2 or the reverse is not based solely on this factor, but rather which has the maximum number of vertices of degree three. If both partitions have the same number of vertices of degree three the larger partition is chosen, if both partitions are the same size with the same number of vertices of degree three then D_1 will precede D_2 in canonical order of degrees with greater value.

The maximum vertex degree of a graph is denoted as $\Delta(G)$. Again from handshaking we can see on a bipartite simple graph that the maximum vertex degree must satisfy the following theorem

Theorem 4.2.1.

$$\Delta(G) < \min(|V_1|, |V_2|).$$

This is due to the maximum vertex degree in a degree sequence D being less than the length of D as otherwise the graph cannot be simple.

$$\max(D) < |D|.$$

This can be proved by considering the faces of the planar graph. If every face has four sides, it will have two vertices from each partition. A vertex v with vertex degree $d(v)$ must have $d(v)$ faces surrounding it. For the graph to be 3-connected no two faces can share the same pair of vertices unless these vertices are adjacent. In a bipartite graph no two vertices in the same partition share an edge. Therefore if vertex v has $d(v)$ faces surrounding it. There must be $d(v)$ vertices distinct from v in the same partition such that it is possible for v to be contained in faces with them. Hence,

$$\forall d(v) \in D, d(v) < |D|.$$

Corollary 4.2.1.1. *It also follows from this proof that both degree sequences must independently satisfy the Erdős-Gallai Theorem (2.4.1).*

All possible pairs of degree sequences D_1 and D_2 for a given size $|V|$ can then be found by looking at the range of permissible sizes s of $|V_1|$ and $|V_2|$ which will be

contained in the interval

$$s \in \left[|V| - \left\lfloor \frac{2|V|-4}{3} \right\rfloor, \left\lfloor \frac{2|V|-4}{3} \right\rfloor \right], |V| \geq 10.$$

Note that we can calculate the range of sizes for s when $|V| = 8$ ($[4, 4]$) and there are no possible values for s when $|V| = 9$ as shown from Figure 4.1 where the minimum is greater than the maximum for $|V| = 9$.

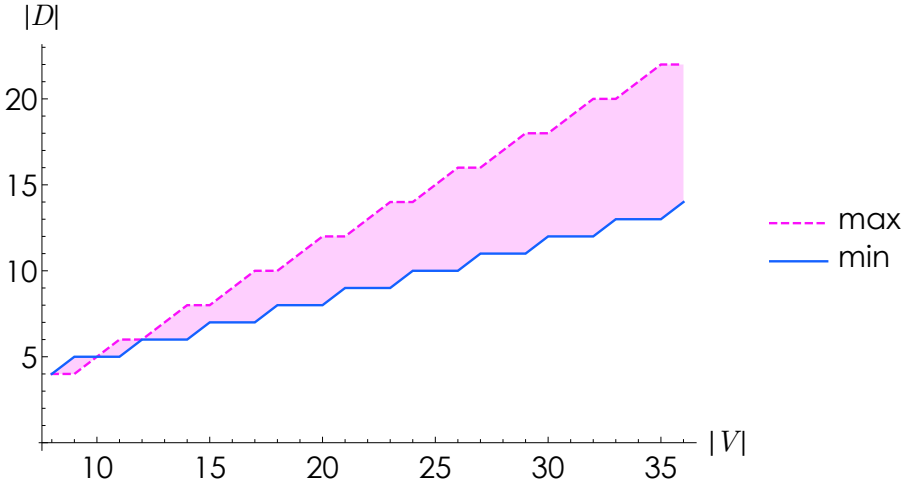


Figure 4.1: Range of permissible sizes for degree sequences by $|V|$

With $\delta(G) = 3$, we then find all integer partitions P of $|V| - 8$ (the additional edges that emanate from some vertices above the necessary three) such that

$$|P| \leq s,$$

$$\max(P) + 3 < s \text{ and } \max(P) \leq |V| - s.$$

If we let P be sorted in descending order such that $p_i \geq p_j$, where $i < j$,

$$D = \left\{ d_i = \begin{matrix} p_i + 3, & \text{if } i \leq |P| \\ 3, & \text{otherwise} \end{matrix}, \text{ where } i = 1, 2, \ldots, s \right\}.$$

We then consider all pairs of degree sequences where $|V_1| + |V_2| = |V|$.

Table 4.1 lists all possible pairs of degree sequences for graphs with 14 vertices. As shown in Figure 4.1, $|V| = 14$ is the lowest number of vertices where there are multiple permissible sizes of V_1 and V_2. We see that there are pairs of degree

D_1	D_2
$\{3,3,3,3,3,3,3,3,3,3,3,3,3,3\}$	$\{5,5,5,3,3,3\}$
$\{3,3,3,3,3,3,3,3,3,3,3,3,3,3\}$	$\{5,5,4,4,3,3\}$
$\{3,3,3,3,3,3,3,3,3,3,3,3,3,3\}$	$\{5,4,4,4,4,3\}$
$\{3,3,3,3,3,3,3,3,3,3,3,3,3,3\}$	$\{4,4,4,4,4,4\}$
$\{6,3,3,3,3,3,3\}$	$\{6,3,3,3,3,3,3\}$
$\{6,3,3,3,3,3,3\}$	$\{5,4,3,3,3,3,3\}$
$\{6,3,3,3,3,3,3\}$	$\{4,4,4,3,3,3,3\}$
$\{5,4,3,3,3,3,3\}$	$\{5,4,3,3,3,3,3\}$
$\{5,4,3,3,3,3,3\}$	$\{4,4,4,3,3,3,3\}$
$\{4,4,4,3,3,3,3\}$	$\{4,4,4,3,3,3,3\}$

Table 4.1: The possible pairs of degree sequences for $|V| = 14$.

sequences where both $|V_1| = 7$ and $|V_2| = 7$ and there are also pairs of degree sequences where $|V_1| = 8$ and $|V_2| = 6$ (since the sum $|V_1| + |V_2| = |V|$). The pairs are ordered based on the sorting procedure detailed previously.

4.3 Outline

The enumeration algorithm is a vertex addition algorithm. At each step a vertex from V_1 is completely connected to vertices in V_2. All possible options for connecting this vertex are considered at this point. They are then tested and successful connections create new subgraphs, each subgraph is processed recursively until all vertices in V_1 are completely connected.

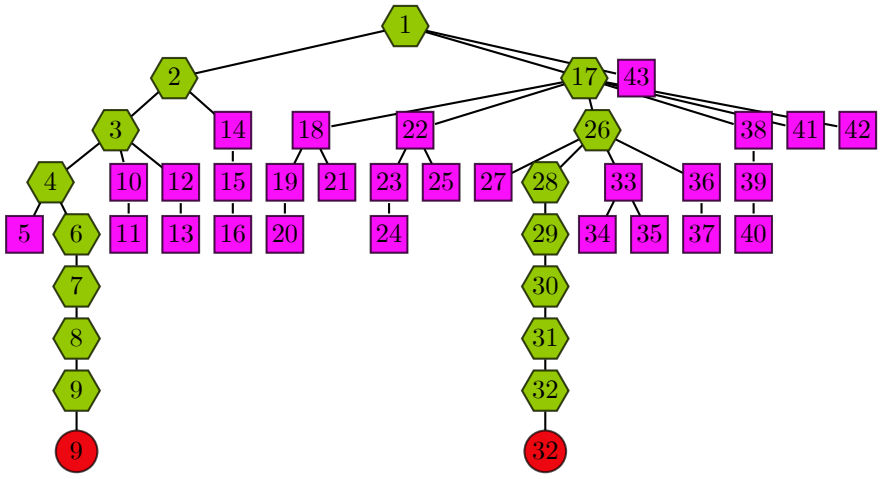


Figure 4.2: Search tree of graph generation with degree sequences: $\{5,4,3,3,3,3,3\}$ and $\{4,4,4,3,3,3,3\}$

Figure 4.2 shows the tree graph representing of all stages of the algorithm when finding graphs with $D_1 = \{5, 4, 3, 3, 3, 3, 3\}$ and $D_2 = \{4, 4, 4, 3, 3, 3, 3\}$. For this pair of degree sequences there are two distinct graphs. We use the following legend to denote the properties of nodes on the search trees:

- Green hexagonal nodes represent subgraphs which lead to solutions of the algorithm.
- Red circular nodes represent a graph which has successfully connected all vertices (completed graphs).
- Pink quadrilateral nodes represent subgraphs which do not lead to a solution of the algorithm.
- Two red circular nodes which are not connected to the search tree but share an edge are graphs which are isomorphic, these nodes can only exist when the degree sequences are equal ($D_1 = D_2$).

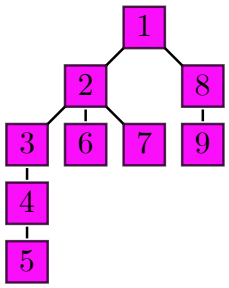


Figure 4.3: Search tree of graph generation with degree sequences: $\{5, 3, 3, 3, 3, 3\}$ and $\{4, 4, 3, 3, 3, 3\}$

Figure 4.3 shows a search tree of nodes generated by the algorithm where no completed graphs are returned. It therefore follows that there exist no 2-vertex 3-edge connected planar graphs with the pair of degree sequences $\{5, 3, 3, 3, 3, 3\}$ and $\{4, 4, 3, 3, 3, 3\}$. We see from the figure that nine nodes are generated before all branches of the search tree are terminated. It would be preferable to have some formula for determining that this pair of degree sequences is not realisable and prove there is no requirement to pass these parameters to the algorithm. In Chapter 5 we present the current work achieved in rejecting pairs of degree sequences which cannot be realised as 3-connected planar graphs.

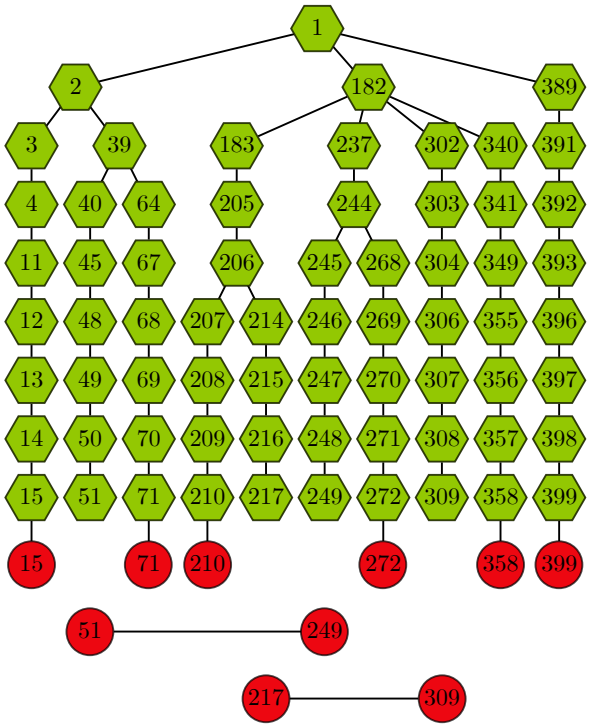


Figure 4.4: Search tree of successful nodes in graph generation with degree sequences: $\{4, 4, 4, 4, 3, 3, 3, 3\}$ and $\{4, 4, 4, 4, 3, 3, 3, 3\}$

Figure 4.4 shows the search tree of solely successful nodes in the graph generation of degree sequences $\{4, 4, 4, 4, 3, 3, 3, 3\}$ and $\{4, 4, 4, 4, 3, 3, 3, 3\}$. These degree sequences generate 403 subgraphs during computation and the search tree is large and less useful for making observations. However, the numbering of the nodes, a depth first numbering of tree in the order that computation actually occurs, alludes to the areas at which unsuccessful nodes occur, but not their exact location or structure. For example, we find that there are a large number of unsuccessful nodes grouped together after the third result (node 71) is generated. We can infer from the search tree that these unsuccessful nodes are either descendants of node 2 or are descendants of unsuccessful child nodes of the root node 1. This can be inferred from the gap in the numbering between the completed graph at node 71 and the next highest node 182. If there is a large number of unsuccessful nodes grouped together then there is more likely to be a common ancestor which yields no results. This would be a desirable candidate for further investigation, as a method for rejecting this common ancestor would have the greatest impact on the enumeration of this particular pair of degree sequences.

In this search tree there are two connected pairs of red circular nodes, which indicate completed graphs that occur in the algorithm twice. This can only occur when $D_1 = D_2$, there can only ever be two completed graphs that are isomorphic in these outputs.

Further work is needed to calculate the inequality which would these identify cases. Currently it is known additional computation on the vertices in V_2 during enumeration would be required and that the pair of decomposed 2-vertex, 3-edge connected planar graphs are not isomorphic to each other. However, they are isomorphic to the pair of decomposed 2-vertex, 3-edge connected planar graphs from the other output where the graph related to V_1 in one output is isomorphic the graph related to V_2 in the other output.

4.4 Invariants

The enumeration algorithm works by utilising invariants that occur in the graph at each stage of processing to reduce unnecessary computation. The first is clearly vertex degree. With vertex degrees of both partitions declared we have an immediate way of distinguishing between vertices. In the algorithm all vertices are initially grouped first by partition and then by vertex degree.

Table 4.2 shows the vertex labelling for the degrees sequences:

$$\{5, 4, 3, 3, 3, 3, 3\} \text{ and } \{4, 4, 4, 3, 3, 3, 3\}.$$

There is also a fixed order for the connection of the vertices in V_1. It is always the vertex with smallest degree number followed by the lowest vertex label. In the case shown in Table 4.2 the connection order of the vertex labels is : $(v_3 \to v_4 \to v_5 \to v_6 \to v_7 \to v_2 \to v_1)$. This means all vertices in V_1 of the same degree are connected consecutively.

There is then an order of precedence on the adjacencies of the vertex in V_1 being connected, with respect to vertices of the same degree in V_1 that have already been connected. This is based initially on the vertex degrees of vertices in partition V_2, followed by the vertex labelling and is introduced in Section 4.5,

$DS_1 = \{5,4,3,3,3,3,3\}$			$DS_2 = \{4,4,4,3,3,3,3\}$		
Vertex Labels	Degree	Freq	Vertex Labels	Degree	Freq
v_1	5	1	v_8, v_9, v_{10}	4	3
v_2	4	1	$v_{11}, \ldots, v_{14}$	3	4
$v_3, \ldots, v_7$	3	5			

Table 4.2: The vertex labelling for degree sequences: $\{5,4,3,3,3,3,3\}$ and $\{4,4,4,3,3,3,3\}$

with further reductions to the search space in Chapter 5.

4.5 Combinations and Banning

4.5.1 Introduction

If a vertex v with degree $d(v)$ is being connected to vertices in a separate partition of size $|V_2|$, where $|V_2'|$ is the number of vertices in the partition which have at least one edge left to connect, the number of different sets of adjacencies that v could have is $\binom{|V_2'|}{d(v)}$. Following these combinations, we would then need to consider the circular permutations of these adjacency sets in order to find all possible cyclic orders of the edges. Therefore if no reductions are made to the possible choices of adjacencies to the vertex, the number of choices would be:

$$\binom{|V_2'|}{d(v)} \times (d(v)-1)!$$

The growth of the number of these combinations has factorial complexity and is of order $|V_2'|!$, this will cause issues with the scalability of the algorithm. As $|V|$ grows so does $|V_2'|$ and necessarily the values $|V_2'|$, in the following therefore present strategies implemented in the algorithm to counter this and limit the number of choices required.

4.5.2 Reverse Circular Permutations

The first optimisation that can be made is reverse circular permutations. The cyclic order of edges about vertices is fixed when connecting a vertex. So different cyclic orders must be considered as different subgraphs. However, because the

graph is bipartite and its dual is 4-regular there are more constraints on the embedding when connecting a vertex. If $v \in V_1$ is connected to the vertices $\{v_{a_1}, v_{a_2}, \ldots, v_{a_{d(v)}}\}$, with a cyclic order $\left(v_{a_1}, v_{a_2}, \ldots, v_{a_{d(v)}}\right)$, this means that $d(v)$ of the faces in the graph have the pairs of the vertices:

$$\begin{gathered}\left\{v_{a_1}, v_{a_2}\right\}, \\ \left\{v_{a_2}, v_{a_3}\right\}, \\ \vdots \\ \left\{v_{a_{d(v)}}, v_{a_1}\right\}.\end{gathered}$$

Providing that the information about the pairs of vertices that are contained in the faces is defined, we have no need to attempt a direction about the vertex (clockwise or anticlockwise). It is sufficient to just declare the faces, therefore it is unnecessary to attempt a cyclic permutation that is the reverse of another.

$$\left(v_{a_1}, v_{a_2}, \ldots, v_{a_{d(v)}}\right) \equiv \left(v_{a_1}, v_{a_{d(v)}}, \ldots, v_{a_2}\right)$$

With this reduction the number of possible ordered adjacencies is now:

$$\frac{\binom{|V_2'|}{d(v)} \times (d(v)-1)!}{2}.$$

4.5.3 Cycles

At every step in the enumeration algorithm a vertex from V_1 is completely connected to vertices in V_2. This vertex is not only connected, but also endowed with the cyclic order of edges about itself. When the vertex from V_1 has degree three, there can only be one possible order as the number of circular permutations of a set of three elements (orientation ignored) is one. When the vertex from V_1 has degree greater than three, the number of circular permutations to consider increases. This number of circular permutations of edges about a vertex v with degree d is:

$$\frac{(d(v)-1)!}{2}.$$

As this number of permutations increases with factorial order, there is a great desire to restrict the possibilities further. Consider the subgraph in Figure 4.5. The degree sequence pair being connected is:

$$\{6,3,3,3,3,3,3\} \text{ and } \{6,3,3,3,3,3,3\}.$$

Suppose that the subgraph at a certain stage of the algorithm has the 6 vertices of degree 3 from V_1 $(v_2, v_3, \ldots, v_7)$ completely connected. The final vertex v_1 has degree six and is yet to be connected to the subgraph and consequently is not present in Figure 4.5. There are only six vertices from V_2 with an edge left to connect, hence this is the only combination of edges. Naively, we now have 60 circular permutations of these edges to consider. However, this is not the case and there is actually only one possible circular permutation of these edges that will permit the vertex to be connected in a planar embedding. Figure 4.5 illustrates this example. We can infer this from the open faces found in the $|V_2| \times |V_2|$ matrix B (4.2). That for any non-zero entry $b_{i,j}$ in B of a completed graph $b_{j,i}$ must also be non-zero. Hence if the matrix B was converted to a binary matrix H where:

$$h_{i,j} = \begin{cases} 1, & \text{if } b_{i,j} \neq 0, \\ 0, & \text{otherwise.} \end{cases},$$

then H would be symmetric for a completed graph. We therefore find the set of pairs:

$$\{\{i,j\}, \text{ where } b_{i,j} \neq 0 \text{ and } b_{j,i} = 0\}.$$

We refer to these pairs as open, in the sense that they are part of a face that has not been assigned its second vertex from V_1. We can also make similar inferences for vertices in V_1. The counts of these pairs of open and closed faces for both V_1 in A and V_2 in B are given in tables 4.3 and 4.4. We see that since v_1 is yet to be connected and is not in the subgraph shown in Figure 4.5, all of its face counts are zero in Table 4.3.

If connecting a vertex v_x from V_1 to $d(v_x)$ vertices in V_2, the cyclic order of the edges emanating from v_x provide $d(v_x)$ pairs of vertices in V_2 which share faces.

If two vertices in V_2 are a pair of vertices which share a face, the algorithm must query whether they already share a face or if this is a new face to be declared. This can be ascertained by checking the corresponding two entries in the adjacency matrix for V_2 (the example being matrix B in Equation 4.2) since the non-zero entries of this matrix correspond to all pairs of face sharing vertices. The non-zero entries in this matrix are the vertices in V_1 with which the pair share a face.

If there does not currently exist an open face between a pair vertices in V_2 then both entries will be 0. Since it is now an open face, one entry will be updated with the vertex number for v_x. If the pair of vertices are already contained in a face, one entry will have the vertex number of a vertex in V_1 and the other will be 0. Consequently the entry that is 0 will be updated with the vertex number for v_x and this is now a closed face.

When a closed face is determined for a pair of vertices in V_2 the corresponding vertices in V_1 have their matrix entries updated in the adjacency matrix for V_1 with the vertex numbers from pair of vertices in V_2. Since the faces are closed for any update to the adjacency matrix for V_1 (the example being matrix A in Equation 4.1), the zero entries are always symmetric in this matrix. Therefore the table for open face counts for all vertices in V_1 are always zero as in Table 4.3, however for the open and closed face counts for vertices in V_2 the faces will first be open and then closed at later stage in the enumeration algorithm. The number of pairs to determine is $d(v_x)$ and the face sharing check in the adjacency matrices is of fixed order. Hence the computational cost of updating these tables is of order $d(v_x)$.

The reduction in circular permutations occurs when a vertex in V_2 has exactly two open face counts and its total face count equal to its vertex degree. We can see this occuring with six of the seven vertices in V_2 for the example shown in Table 4.4 where $v_9, v_{10}, \ldots, v_{14}$ are all vertices of degree three, with total face counts of three and open face counts of two. Whenever a vertex in V_2 is meets this criteria we 'fix' it to the only two possible vertices with which it can share a face in V_2. If we have k vertices in V_2 which meet this criteria then we find the number of circular permutations $p(d(v), k)$ to consider is now:

$$p(d(v),k) = \begin{cases} \frac{(d(v)-1-k)!}{2}, & \text{if } d(v)-1-k > 0, \\ 1, & \text{otherwise.} \end{cases}.$$

In the example given we are connecting v_1 from V_1 which has a vertex degree $d(v_1) = 6$ and all six vertices in V_2 meeting the criteria. Hence we have the number of permutations $p(6,6) = 1$. This optimisation highlights the motivation for connecting the vertices in V_1 by ascending vertex degree order, since this maximises the reduction in computation as the numbers of vertices in V_2 which meet the criteria (k) is greatest for later stages of the enumeration and have greater impact on larger vertex degrees.

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 10 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 & 11 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 & 12 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & 13 \\ 0 & 0 & 0 & 8 & 0 & 0 & 14 \\ 0 & 0 & 0 & 0 & 8 & 8 & 0 \end{pmatrix} \tag{4.1}$$

$$B = \begin{pmatrix} 0 & 2 & 4 & 5 & 6 & 7 & 7 \\ 3 & 0 & 2 & 3 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 4 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 5 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 6 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 7 & 0 \end{pmatrix} \tag{4.2}$$

4.5.4 Vertex Degree Sets

The next restriction of the possible adjacencies is given by a constraint on permissible lists of vertex degrees from vertices in V_2. We define a **vertex degree set** as a subset of a vertex degree sequence. Each vertex degree set is a different

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Open Face Count	0	0	0	0	0	0	0
Closed Face Count	0	2	2	2	2	2	2
Total Face Count	0	2	2	2	2	2	2

Table 4.3: Open and closed face counts for matrix A

	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
Open Face Count	0	2	2	2	2	2	2
Closed Face Count	6	1	1	1	1	1	1
Total Face Count	6	3	3	3	3	3	3

Table 4.4: Open and closed face counts for matrix B

combination of vertex degrees in V_2. Consider the vertices in Table 4.2. If enumerating the possible adjacencies for v_3, all sets of three arising from the seven vertices are first considered (since there is only one circular permutation of three elements if reverses are ignored) which gives 35 possible adjacency sets. There is an order of precedence when connecting a vertex in V_1 which requires that the canonical ordering of the vertex degrees corresponding to its adjacency list cannot precede that of a vertex of the same degree that has already been connected.

So if v_3 is connected with adjacency list $\{v_8, v_{11}, v_{12}\}$, with vertex degrees $\{4, 3, 3\}$ respectively, no vertices in V_1 of degree three could then have an adjacency list with corresponding vertex degrees $\{4, 4, 3\}$, as this would contradict the order. The ordering of these vertex degrees is descending and is not determined by the cyclic order of the adjacencies. The possible vertex degree sets in canonical order of length three for $D_2 = \{4, 4, 4, 3, 3, 3, 3\}$ are shown in Table 4.5.

Vertex Degree Set
$\{4, 4, 4\}$
$\{4, 4, 3\}$
$\{4, 3, 3\}$
$\{3, 3, 3\}$

Table 4.5: Possible vertex degree sets of length three from degree sequence: $\{4, 4, 4, 3, 3, 3, 3\}$.

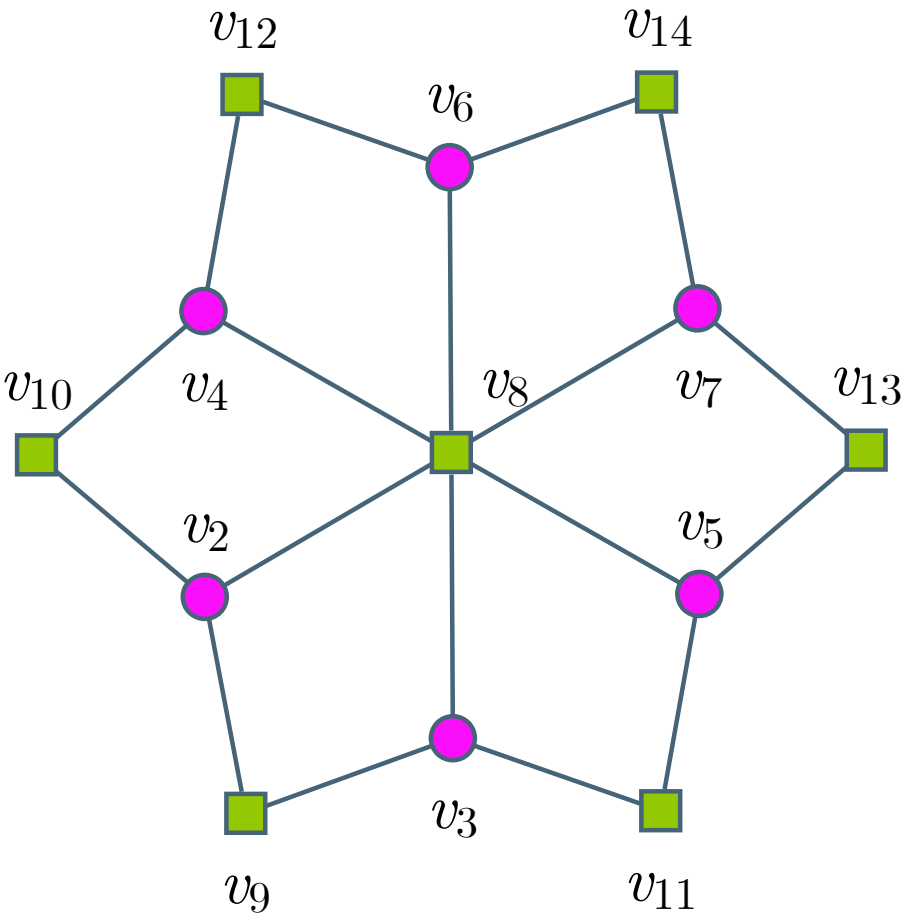


Figure 4.5: Graph example for optimised circular permutations

4.5.5 Bans from Vertex Labels

When finding different adjacencies for a vertex v in V_1 being connected, there is another restriction added to search. If vertex v has the same vertex degree as the previous vertex connected, not only must the vertex degree set be the same or procede the previous set in canonical order, but if the vertex degree set of the adjacencies is the same, the ordered vertex labels must be greater than or equal to previous ordered vertex labels. This again disregards actual cyclic order about the vertex.

For example, taking v_3, v_4 and v_5 from Table 4.2, if vertex v_3 connects to $\{v_8, v_9, v_{11}\}$ and then vertex v_4 connects to $\{v_8, v_{10}, v_{12}\}$, we can see that both vertex degree sets are $\{4, 4, 3\}$. When looking to connect v_5 not only must the vertex degree set be the same or proceed $\{4, 4, 3\}$ but so must the vertex labels.

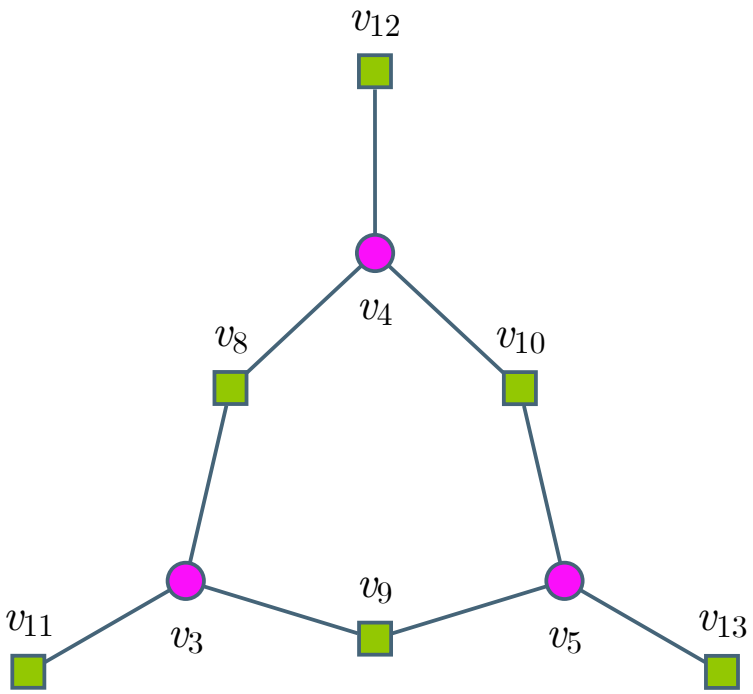


Figure 4.6: Subgraph with adjacencies from Table 4.6
(node 27 on the search tree in Figure 4.2)

For instance, you could not consider the adjacency set $\{v_8, v_9, v_{13}\}$, as the ordered adjacency set for v_4 has v_{10} at position two, which supersedes the v_9 at position two in $\{v_8, v_9, v_{13}\}$. However, an adjacency set for v_5 could be $\{v_9, v_{10}, v_{13}\}$. This example is actually enumerated in the algorithm, the adjacency sets are listed in Table 4.6. This does not lead to a complete result but actually terminates after the connection of v_5. This is node 27 in Figure 4.2. The subgraph after connecting these three vertices is shown in Figure 4.6.

$v_i \in V_1$	Adjacency set of vertices in V_2
v_3	$\{v_8, v_9, v_{11}\}$
v_4	$\{v_8, v_{10}, v_{12}\}$
v_5	$\{v_8, v_{10}, v_{13}\}$

Table 4.6: An example of permissible adjacency sets for the first three vertices connected when $D_1 = \{5, 4, 3, 3, 3, 3, 3\}$ and $D_2 = \{4, 4, 4, 3, 3, 3, 3\}$

4.6 Subgraph Mapping

The key goal of combinatorial enumeration is to achieve complete results, without isomorphisms at the least computational expense. This algorithm has at all costs attempted to reduce the need for graph isomorphism checks during graph enumeration. Isomorphism checks are one to one tests that are not currently known to be run in polynomial time. However, if the graphs are both planar we can test for isomorphism in linear time [Hopcroft and Wong, 1974]. Other strategies have been proposed to reduce the number of tests required or to eliminate the need altogether. One particularly notable method was Read [Read, 1978] who created a test where generated graphs are scored based on a canonical ordering if the graph has not already been found then its score will be greater than any already seen by the algorithm. The implication being you must have an up to date score for the comparison.

This graph algorithm uses a specific subgraph mapping procedure designed with the following points in mind:

- Isomorphisms should be found early in the enumeration (before graphs are complete).
- Any information required to identify an isomorphism needs to be held with subgraph data in a given node, to allow parallelisation.
- All graph invariants that are useful should be known and require little computation to infer information in comparisons.

The gains made by the subgraph mapping procedure are reviewed in Section 4.7 where implementations of the algorithm with and without this process are compared.

Space Complexity of Isomorphism Data

The extra data held in each node to identify isomorphisms in the enumeration while still permitting parallelisation is of fixed size per isomorphism held. The potential number of isomorphisms held is proportional to $|V|!$, however, this num-

ber is dependant on connected graphs generated after the optimisation detailed in Subsection 4.5. Hence, in practice the number is heavily reduced by heuristics.

Graph Invariants Held

The graph invariants held to guarantee repeated computation is kept to a minimum are labelled adjacency matrices, current number of edges incident to each vertex and open and closed face counts.

The labelled adjacency matrices that describe the faces of the graph (examples of which are given in Chapter 3, equations 3.1 and 3.2) eliminate any requirement to retrace the embedding to find which vertices share faces. These matrices have a memory size of order V^2 and a linear computational cost of $O(V)$ to update.

The current number of edges incident to each vertex could be calculated by counting the number of adjacent vertices in the adjacency list. However, since it is used in restricting which graphs have potential isomorphisms, has a fixed memory size of order V for the data structure and is of $O(1)$ to update, it is also held in each node.

The open and closed face counts are discussed in Subsection 4.5.3 where examples are given in Table 4.3 and Table 4.4. These again are of $O(1)$ to update and have a fixed memory size of order V. The tradeoff between holding this extra information and the cost of repeated calculation has then been made in favour of holding more information in memory.

4.6.1 Subgraph Data Structure

When enumerating graphs, we have already declared certain graph invariants that are computed in order. The degree of the vertex in V_1 and the vertex degree set its adjacencies in V_2. Therefore, any overlaps (possible graph isomorphisms) must occur when graphs with identical vertex degree sets are being enumerated. The need is then for an algorithm that identifies the subgraph that has just changed, can compare against any others that have also been made and exclude graphs which can only give results isomorphic to others found earlier in the search tree. Subgraphs can be held through several iterations for future vertices connected at

later stages, even if they are not currently isomorphic to a subgraph in the node. This is required until the vertex degree set being connected by the vertex in V_1 is changed. The number of iterations where subgraphs are held is at most $|V_1| - 1$ since the root node holds no subgraphs.

This is effectively another order of precedence. When iterating a subgraph a new vertex from V_1 is connected. This new vertex will be part of a connected subgraph in the subgraph. This is not necessarily the whole subgraph, as vertex degree precedence is being used. This is shown in Figure 4.9, where there are two disconnected subgraphs at this node. If there are other successful adjacencies with the vertex degree sets for this same node, they will hold a small piece of extra data. This extra data is the vertex number currently being connected and its cyclically ordered adjacencies. From this information the connected subgraph we wish to exclude from other iterations can be inferred, since all other information will be inherited from the parent node.

4.6.2 Subgraph Isomorphism Test

We can now define our subgraph mapping, when a new vertex is connected with a cyclic order. Providing that the degree of this vertex and the vertex degree set are the same as its predescent vertex, we now attempt to map all excluded subgraphs against any vertex number that has a greater or equal connection order to that of the subgraph exclusion.

The mapping works by attempting to map the adjacency list of a vertex v to that held for the excluded subgraph. As they are both held cyclically there are $2 \times d(v)$ ways of mapping the cycle. These are first checked for the graph invariants (the computational cost of checking a particular cycle of an adjacency list is of the order $O(d(v))$), if they match then that particular orientation and fixed cyclic order of adjacencies is checked for permission of a subgraph meeting the criteria. The next step is to verify that the same faces exist in the graph. So the face matrices are checked for the same face relationships between the mapping. If both of these properties exist the algorithm performs a depth first search, mapping all vertices in the original subgraph to the new graph. If a

complete mapping is found then the graph will not be worked on further in the algorithm.

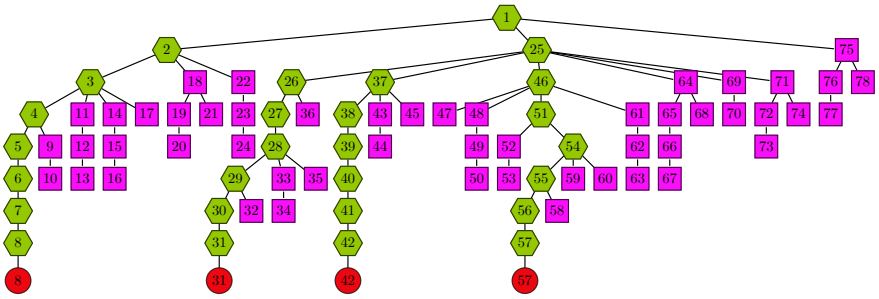


Figure 4.7: Search tree for $D_1 = \{4, 4, 4, 3, 3, 3, 3\}$ and $D_2 = \{4, 4, 4, 3, 3, 3, 3\}$

The search tree in Figure 4.7 shows all the nodes enumerated for the pair of degree sequences $\{4, 4, 4, 3, 3, 3, 3\}$ and $\{4, 4, 4, 3, 3, 3, 3\}$. The subgraphs at nodes 76 and 78 are shown in Figures 4.8 and 4.9 respectively. Since node 78 is created from the same parent as 76 (node 75) we exclude any subgraph of the children of node 78 from being isomorphic to node 76. Since the nodes have the same parent, there is only one vertex from V_1 with different adjacencies, v_5. Therefore we endow node 78 with an additional list of adjacencies that is $v_5 \leftrightarrow \{v_9, v_{11}, v_{12}\}$. When enumerating node 78 and finding child nodes, we attempt to map any vertices in V_1 connected vertices with the same vertex degree set as v_5 with the excluded v_5. Any vertices connected prior to v_5 that are connected in the excluded subgraph (in this example v_4) are also then mapped. If a one - one mapping can be found, with all vertex degrees matching then a subgraph isomorphic to one excluded in the enumeration algorithm has been found and consequently will be disregarded. A subgraph which was found but not logged as a node for further computation is shown in Figure 4.11 where a potential child node of 78 is found to be isomorphic to node 76. We can disregard these nodes while maintaining exhaustive enumeration. This is a consequence of the node with which the isomorphism was found (in this case node 76) continuing its enumeration without having any subgraph exclusions from nodes with which it shared a parent that have a higher node label. Hence we see node 77 in Figure 4.10 having several isomorphisms with node 78. An equivalent definition would be to state that nodes will only ever hold excluded subgraphs of other nodes with

a label number less than their own. Furthermore, this information is only held while vertices in V_1 being connected are attempting combinations of adjacencies in V_2 with the same vertex degree set. In the example given here, the V_1 vertices v_4, v_5 and v_6 all have adjacencies with vertex degree set $\{4, 3, 3\}$. If the algorithm was to exhaust all possibilities for this set and move to adjacencies with vertex degree set $\{3, 3, 3\}$ all the excluded subgraphs could be disregarded as no possible edges could be added that would make a vertex in v_1 isomorphic to an excluded subgraph without breaking order in which these graph invariants are considered.

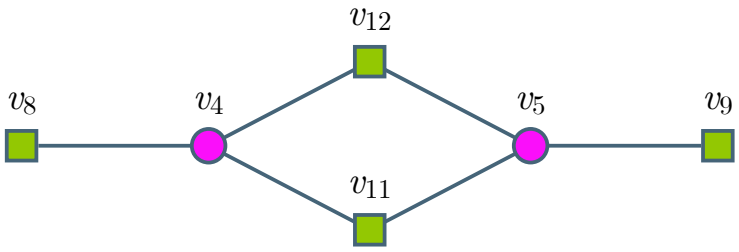


Figure 4.8: Subgraph at node 76 in $D_1 = \{4, 4, 4, 3, 3, 3, 3\}$ and $D_2 = \{4, 4, 4, 3, 3, 3, 3\}$

When considering which vertices in a new graph to compare for isomorphisms to an excluded subgraph, all vertices with adjacencies that have the current vertex degree set must be checked at every step. In the example subgraphs A1 and B1 shown in Figures 4.12 and 4.13 we have two graphs which are both direct descendants of the same parent node. These graphs come from the pair of degree sequences $D_1 = \{5, 4, 4, 4, 4, 3, 3, 3, 3, 3\}$ and $D_2 = \{4, 4, 4, 4, 4, 4, 3, 3, 3, 3\}$. Since the parent node is the same, vertex v_8 (the vertex in V_1 connected to create the parent node) has the same adjacencies in both A1 and B1 $\{v_{11}, v_{12}, v_{17}\}$, likewise for v_6 and v_7. The vertex in V_1 connected to create A1 and B1 is v_9 with v_9 in A1 having adjacencies $\{v_{11}, v_{14}, v_{17}\}$ and v_9 in B1 having adjacencies $\{v_{13}, v_{14}, v_{18}\}$. A1 was created first and therefore has a value which precedes the label of B1 in the search tree. In these examples v_8 and v_9 have adjacencies with the same vertex degree sets $\{4, 4, 3\}$. Since v_9 has the same vertex degree set, B1 is endowed with an exclusion of any subgraphs isomorphic to A1 in further enumeration.

Graphs A2 and B2 in Figures 4.14 and 4.15 are potential child nodes of A1 and B1 respectively. In both A2 and B2 v_{10} also has adjacencies with vertex

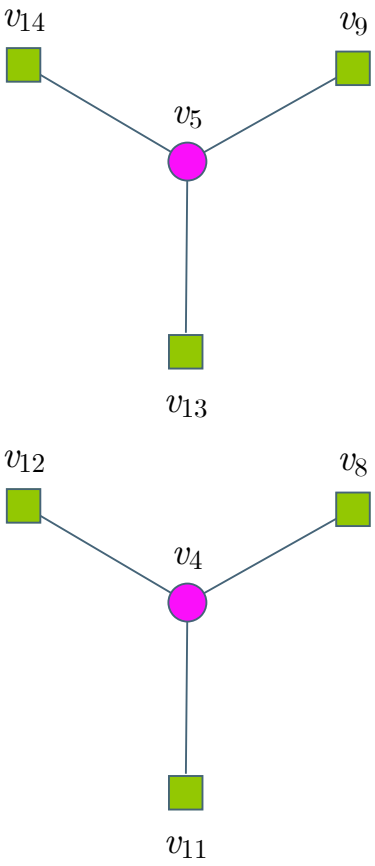


Figure 4.9: Subgraph at node 78 in $D_1 = \{4, 4, 4, 3, 3, 3, 3\}$ and $D_2 = \{4, 4, 4, 3, 3, 3, 3\}$

degree set $\{4, 4, 3\}$ and hence when considering B2 the excluded subgraph A1 must be checked against B2 for an isomorphism. Recall that to check this we attempt to map v_9 from A1 and all previously connected vertices from V_1 (v_6, v_7 and v_8) to the graph in B2. Let the vertices in the graph for B2 be denoted v_i. We see that there is a mapping from v_9 to a vertex in B2 which would lead to a complete one-one relationship, implying isomorphism. However, it does not occur when considering the newly connected vertex v'_{10} but rather v'_9. This is significant because this isomorphism did not exist between two vertices v_9 in A1 and B1, but since v_8, v_9, v_{10} and v'_8, v'_9, v'_{10} all have adjacencies with the same vertex degree sets they can be mapped to each other. We find the isomorphism by $v_9 \to v'_9$ and $v_8 \to v'_{10}$, $v_7 \to v'_6$ and $v_6 \to v'_7$. This example is given to highlight why the algorithm attempts to map both v'_9 and v'_{10} to v_9 at this stage. The vertex v'_8 would not, however, be considered for mapping against v_9 since it was generated prior to the subgraph exclusion. It can be included in the mapping, just not for

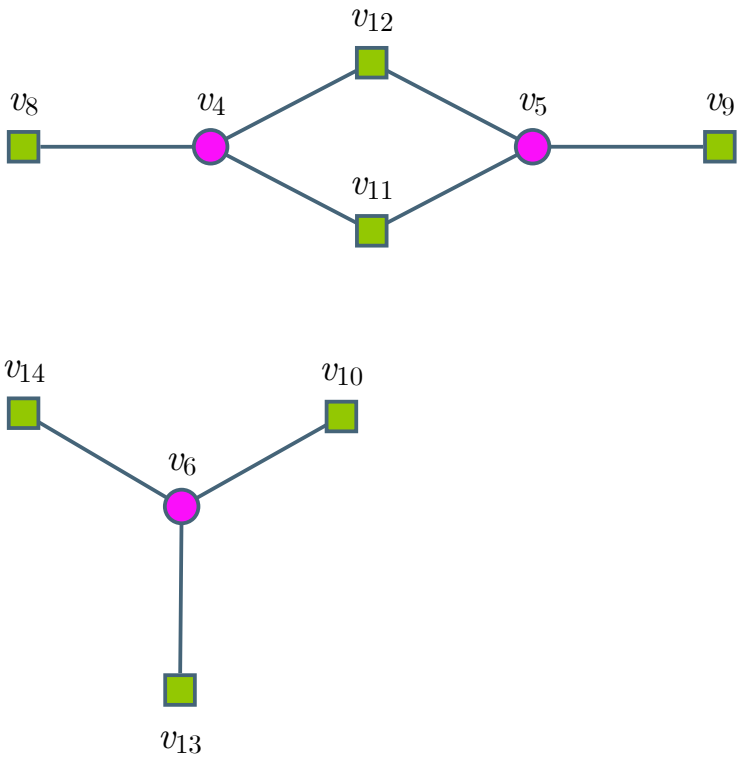


Figure 4.10: Subgraph at node 77 in $D_1 = \{4, 4, 4, 3, 3, 3, 3\}$ and $D_2 = \{4, 4, 4, 3, 3, 3, 3\}$

the first vertex. These subgraph exclusions only relate to connected graphs at the time the exclusion is declared, for example if Node 77 from Figure 4.10 was an excluded subgraph for another node. The only mapping required is v_6 since the vertices v_4 and v_5 are not mapped as they are disconnected.

4.7 Computational Complexity

This section details the reduction in computational cost made by optimisations in the algorithm. Results have been generated for low numbers of $|V|$ without the optimisations added to give a comparitive analysis and show the actual savings made.

The code implementation for the algorithm has been adjusted to run without the following two optimisations:

- The banning method (Section 4.5) which removes combinations of vertices that have equivalent adjacency lists.

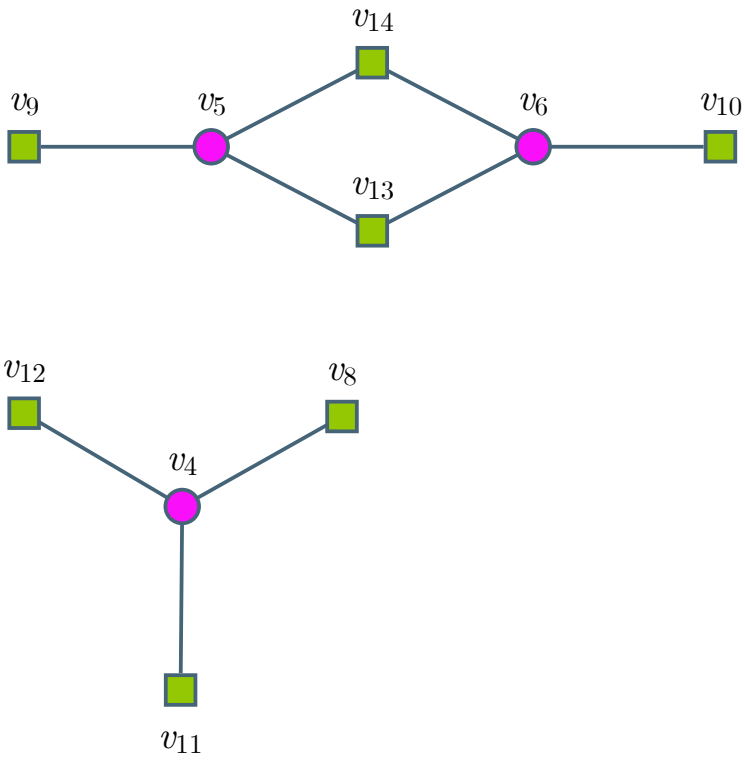


Figure 4.11: A subgraph isomorphic to subgraph at node 77 in $D_1 = \{4, 4, 4, 3, 3, 3, 3\}$ and $D_2 = \{4, 4, 4, 3, 3, 3, 3\}$

- The subgraph mapping procedure (Section 4.6) which eliminates isomorphic results (with the exception of the potential duplicates when $D_1 = D_2$).

4.7.1 Reduction in Computation from Bans

The banning method introduced in Section 4.5 restricts the number of combinations of vertices in V_2 which are considered for adjacency lists when connecting a vertex from V_1. If these restrictions are not imposed but the subgraph mapping procedure is still implemented we find that the number of nodes generated is the same as final code. This is due to the subgraph mapping procedure identifying a greater number of isomorphic graphs, however the overhead on CPU time is increased since reducing combinations at the earlier stage is more efficient than having them identified by subgraph mapping.

Figure 4.16 shows the CPU time taken for running the different pairs of degree sequences of all pairs considered for $|V| \leq 18$. The red line is $y = x$ and is shown to highlight that the time taken without the optimisation is always greater. Figure

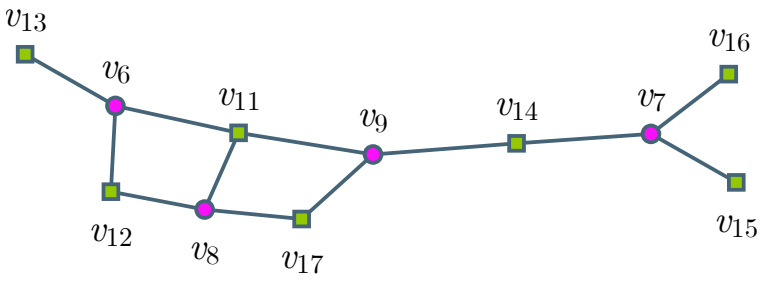


Figure 4.12: Example Graph A1

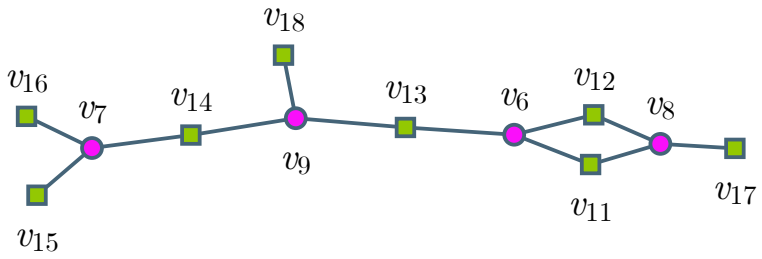


Figure 4.13: Example Graph B1

4.17 shows the number of nodes generated between the code with and without the banning method compared to the CPU time difference, this is plotted for all pairs of degree sequences for $|V| \leq 18$. The total CPU time taken by the algorithm with the optimisation is 1117.8 seconds, without the banning method the CPU time is 1700.6 seconds giving a reduction in CPU time over the enumeration of all graphs with up to 18 vertices of 34%.

4.7.2 Reduction in Computation from Subgraph Mapping

When a version of algorithm is run without the subgraph mapping procedure (as described in Section 4.6) we not only have a larger number of nodes generated but also there are no guarantees on the uniqueness of results. This is because the subgraph mapping procedure identifies the isomorphic graphs and discards them. Therefore not only do we see an overhead on computation during the algorithm run, there is an additional post processing step required on the results from each pair of degree sequences to remove overlapping results. The computational cost of the post processing is $O(|V|R^2)$ where R is the number of results output by a pair of degree sequences. This is due to isomorphism checks being one to one tests

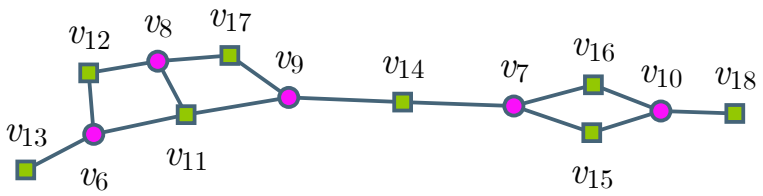


Figure 4.14: Example Graph A2

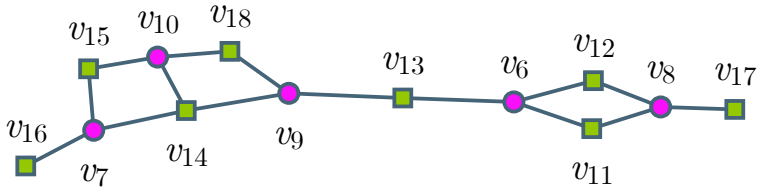


Figure 4.15: Example Graph B2

and therefore the upper bound on tests required is $\frac{R(R-1)}{2}$. Figure 4.18 compares the numbers of nodes generated for different pairs of degree sequences with and without the subgraph mapping procedure. The number of nodes generated for $|V| \leq 18$ is 54821 for the algorithm with the optimisation, without the subgraph mapping procedure it is 98847, which is a reduction of 45%.

4.7.3 Overall Reduction in Computational Cost

In this section we present the computational cost when neither the combinations banning or the subgraph mapping procedure are implemented in the code and compare it to the output from the algorithm. Time taken with optimisations is 283.1 seconds, without is 1677.1 seconds which is an 83.1% reduction. Total number of nodes generated with optimisation is 17313 compared to 122852 without which is an 86% reduction. As $|V|$ increases the proportion of nodes generated that is reduced the optimisations is also increasing. Table 4.7 show this for $|V| \leq 17$. This reflects the subgraph mapping procedure identifying isomorphic subgraphs early in the enumeration.

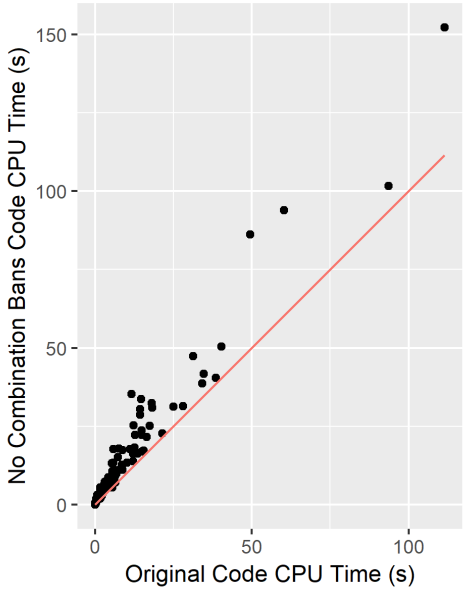


Figure 4.16: Graph Comparing CPU Time for Different Versions of the Code

$\|V\|$	With Optimisations	Without Optimisations	Proportion
10	7	13	0.462
11	19	171	0.889
12	34	86	0.605
13	81	257	0.685
14	441	1921	0.770
15	828	3279	0.747
16	3238	17693	0.817
17	12665	99432	0.873

Table 4.7: Numbers of nodes generated with and without optimisations of the code for $|V|$

4.8 Results Generated by the Algorithm

4.8.1 Introduction

In this section we summarise the results of the enumeration algorithm. The analysis of the results will follow in a similar fashion to that of the problem subdivision in the algorithm itself. First the total number of graphs for a given $|V|$ is broken down into the numbers for $|V_1|$ and $|V_2|$. From these we take the results which are exactly 2-vertex, 3-edge connected graphs whose vertex degrees are the sequence D_1 and face degrees are D_2 and as such their dual graphs, which are also exactly 2-vertex, 3-edge connected. Since for an exactly 2-vertex, 3-edge

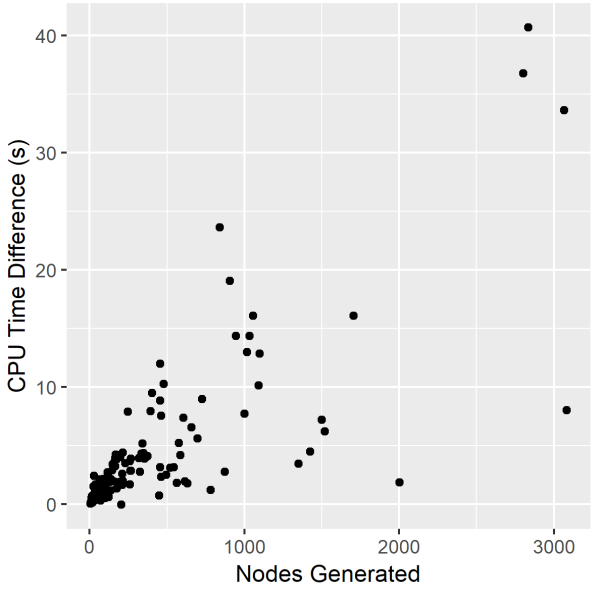


Figure 4.17: Graph of Number of Nodes Generated vs CPU Time Difference for Code Versions

connected planar graph, there can exist multiple embeddings $\Gamma_{x_1}, \Gamma_{x_2}, \ldots, \Gamma_{x_n}$ of a graph G_x where the duals $\Gamma_{y_1}, \Gamma_{y_2}, \ldots, \Gamma_{y_n}$ are not equivalent but the graphs $(\Gamma y_i)', (\Gamma y_j')$ are isomorphic, we find that Table 4.12 is asymmetric as it lists numbers of graphs as opposed to embeddings.

4.8.2 Comparison with Existing Algorithms

When comparing to other algorithms a key metric is the speed for returning the same results. However, the nature in which this algorithm enumerates graphs gives advantages when considering subsets of graphs with larger vertices. For instance, if only concerned with duals of 4-regular polyhedra with certain degree sequences, the algorithm from this thesis benefits from generating graphs by their degree sequences as opposed to generating all the duals of 4-regular polyhedra and then filtering the results to return only those which fit the criteria. Gains made by the optimisations in this algorithm are compared in Section 4.7 and the parallelisation possible by this approach is reviewed in Chapter 6.

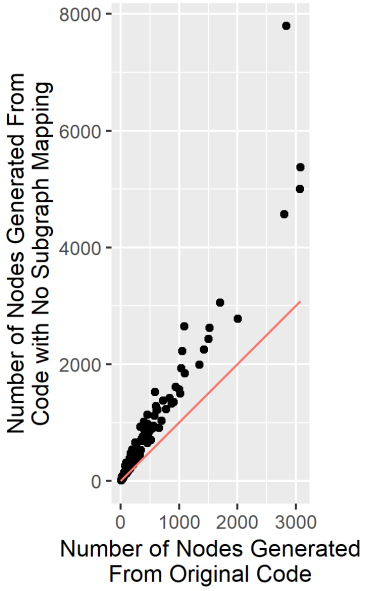


Figure 4.18: Graph Comparing Number of Nodes Generated for Different Versions of the Code

4.8.3 Total Numbers of Duals of 4-Regular 3-Connected Planar Graphs

This section provides the numbers of duals of 4-regular, 3-connected planar graphs with up to 22 vertices. This is shown in Table 4.8 where the results have been divided into the bipartition numbers $|V_1|$ and $|V_2|$ where $|V| = |V_1|+|V_2|$. This table is different from the results of Duijvestijn and Federico [Duijvestijn and Federico, 1981] (Table 2.4), as these include all embeddings of all 2-vertex, 3-edge connected planar graphs. To calculate the total number of duals of 4-regular, 3-connected graphs for a given $|V|$ we sum the table entries ($|V_1|$, $|V_2|$) where $|V_1| \geq |V_2|$. Hence for $|V| = 14$ we take the entries (8,6) and (7,7) which are 2 and 9 respectively. Therefore the total number of unique 3-connected graphs with solely quadrilateral faces and 14 vertices is 11.

Table 4.9 lists the numbers of all duals of 4-regular, 3-connected planar graphs which decompose along their bipartitions to 3-connected planar graphs. Where as Table 4.8 will always include entries greater than or equal to Table 2.4, Table 4.9 will always have entries less than or equal. Where $|V_1| \neq |V_2|$ the entries are the same, however for $|V_1| = |V_2|$ the numbers are less for $|V_1| = |V_2| \geq 7$. The distinction in the tables being that we are counting the number 4-regular,

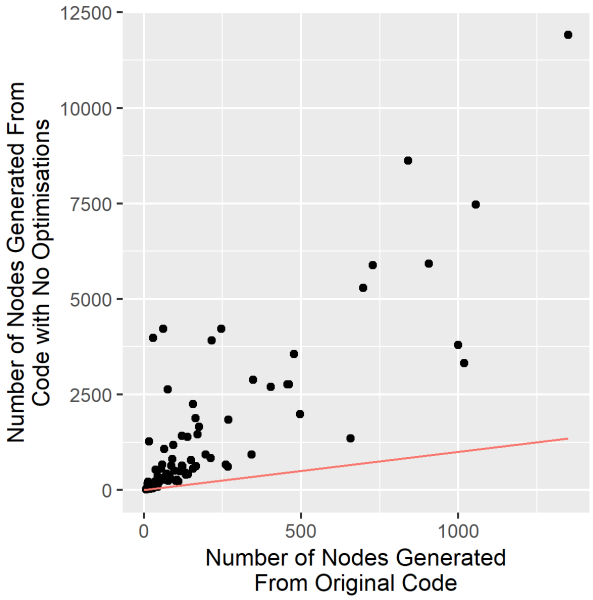


Figure 4.19: Graph Comparing Number of Nodes Generated with and without Optimisations of the Code

3-connected graphs being constructed from the polyhedral graphs, not the polyhedral graphs themselves. Where we have a polyhedral graph with the same number of vertices and faces which is not self dual, this graph and its dual together construct one 4-regular, 3-connected planar graph. Hence if we denote entries in the Table 2.4 as $T^X_{|V_1|,|V_2|}$ and entries in Table 4.9 as $T^Y_{|V_1|,|V_2|}$, the number of polyhedra with the same number vertices $|V|$ as faces $|F|$ which are not self dual, is twice the difference between these two tables and is given by

$$2(T^X_{|V|,|V|} - T^Y_{|V|,|V|}).$$

Consequently the number of self dual polyhedra with $|V|$ vertices is given by

$$2T^Y_{|V|,|V|} - T^X_{|V|,|V|}.$$

Table 4.10 lists the numbers of all duals of 4-regular, 3-connected planar graphs which decompose along their bipartitions to exactly 2-vertex connected 3-edge connected planar graphs. The sum of this table and Table 4.9 is Table 4.8. The table is symmetric in $|V_1|$ and $|V_2|$, a key distinction between this table and Table 4.9 is that when equating these entries to the decompositions, we are counting

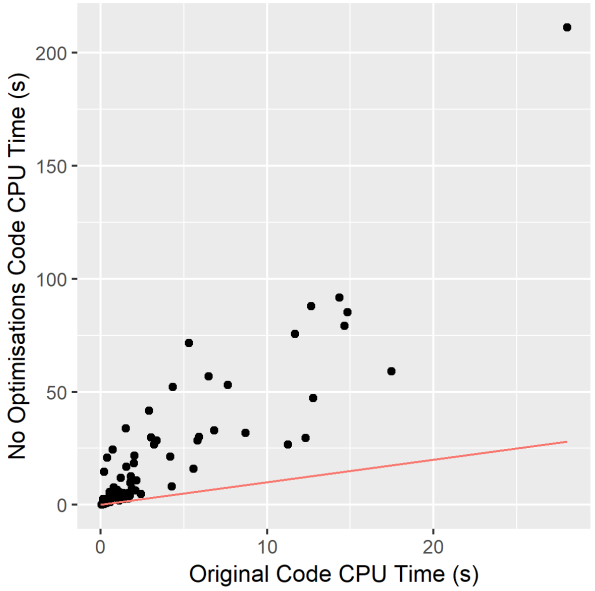


Figure 4.20: Graph Comparing CPU Time with and without Optimisations of the Code

embeddings as opposed to graphs. Hence, the entries which are not on the leading diagonal are the number of 2-vertex, 3-edge connected embeddings with $|V_1|$ vertices and $|V_2|$ faces (or $|V_1|$ faces and $|V_2|$ vertices). When considering the entries on the leading diagonal where $|V_1| = |V_2|$ a similar relationship occurs to that in Table 4.9 a self dual embedding is counted once and all pairs of embeddings Γ_x, Γ_y where $\Gamma_x = \Gamma'_y$ are also counted once since together Γ_x and its dual Γ_y construct one 4-regular, 3-connected planar graph.

Table 4.11 gives the numbers of all exactly 2-vertex, 3-edge connected planar graphs with a given number of vertices $|V|$ and faces $|F|$. The number of vertices are listed by row and the number of faces by column. The asymmetry in this table is due to the variance in number of distinct embeddings formed by a single graph. The number of unique dual graphs resulting from the embeddings with $|V|$ vertices and $|F|$ faces are not necessarily the same as the number of unique dual graphs resulting from the embeddings with $|F|$ vertices and $|V|$ faces. From the table we have that the number of graphs with vertices $|V|$ greater than faces $|F|$ increase faster than their duals. This also implies for the numbers given, that each of those graphs on average must have less distinct embeddings. This is noted in Chapter 7 for future work as the trend appears to be that exactly 2-vertex,

V_1 / V_2	4	5	6	7	8	9	10	11	12
4	1								
5		1	1						
6		1	3	3	2				
7			3	9	18	10	5		
8			2	18	48	134	123	50	14
9				10	134	328	1276	1420	
10				5	123	1276	3027		
11					50	1420			
12					14				

Table 4.8: Numbers of duals of 4-regular, 3-connected planar graphs

V_1 / V_2	4	5	6	7	8	9	10	11	12
4	1								
5		1	1						
6		1	2	2	2				
7			2	7	11	8	5		
8			2	11	29	74	76	38	14
9				8	74	173	633	768	
10				5	76	633	1400		
11					38	768			
12					14				

Table 4.9: Numbers of duals of 4-regular 3-connected planar graphs that decompose to 3-connected graphs

3-edge connected planar graphs with fewer vertices than faces have more distinct planar embeddings.

We conclude this section by giving Table 4.12 which is the number of all 2-vertex, 3-edge connected planar graphs with a given number of vertices $|V|$ and faces $|F|$. As with Table 4.11 the number of vertices are listed by row and faces by column. The difference between this table and Table 4.11 is Table 2.4. These are the complete numbers of all graphs resulting from the decomposition of the duals of 4-regular, 3-connected planar graphs. We note that the algorithm designed in this thesis generates the distinct embeddings of 2-vertex, 3-edge connected planar

V_1 \ V_2	4	5	6	7	8	9	10	11	12
4									
5									
6			1	1					
7			1	2	7	2			
8				7	19	60	47	12	
9				2	60	155	643	652	
10					47	643	1627		
11					12	652			
12									

Table 4.10: Numbers of duals of 4-regular, 3-connected planar graphs that decompose to exactly 2-vertex, 3-edge connected graphs

graphs, not the set of unique graphs. Hence the resulting graph data gives us the numbers of distinct embeddings directly. However, to find the unique 2-vertex, 3-edge connected planar graphs isomorphism checks were performed.

4.8.4 Summary of Results

This information gives a new integer sequence for the total number of 2-vertex, 3-edge connected planar graphs with a given number of vertices. There are also interesting properties pertaining to the proportions of exactly 2-vertex, 3-edge connected planar graphs that exist compared to the number 3-connected planar graphs (for a given $|V|$ and $|F|$). These results are further broken down in Appendix A with Tables A.1 - A.7, listing the numbers of graphs that decompose to exactly 2-vertex, 3-edge connected graphs and the numbers that decompose to 3-connected graphs. They are listed by degree sequence pairs and also include the number of nodes generated in the search tree when running the algorithm.

4.9 Summary

The amalgamation of the properties and methods in this chapter results in an algorithm which will enumerate all bipartite planar graphs whose duals are 4-regular and 3-connected given a pair of degree sequences. The graphs are calcu-

Faces \Vertices	4	5	6	7	8	9	10	11	12
4									
5									
6			1	1					
7			1	3	5	2			
8				7	27	42	32	10	
9				2	51	202	377	360	
10					43	473	1909		
11					11	525			
12									

Table 4.11: Numbers of exactly 2-vertex, 3-edge connected planar graphs with minimum vertex degree three

Faces \Vertices	4	5	6	7	8	9	10	11	12
4	1								
5		1	1						
6		1	3	3	2				
7			3	11	16	10	5		
8			2	18	69	116	108	48	14
9				10	125	498	1010	1128	
10				5	119	1106	4544		
11					49	1293			
12					14				

Table 4.12: Numbers of 2-vertex, 3-edge connected planar graphs

lated without isomorphisms, with the current exception being at most pairs of isomorphic results if the degree sequences are the same ($D_1 = D_2$). Some methods and data captured during computation require a deeper explanation than that given in the overview in order to appreciate the reductions in cost for the algorithm.

Chapter 5

Observations from the Data

5.1 Introduction

In this chapter we prove properties of integer partitions which when considered as vertex and face degree sequences of 3-connected planar graphs are non-realisable.

These properties are calculated using arithmetic operations on ordered integer partitions. As such, no data is held other than one or two one-dimensional arrays of integers and no graphs are constructed. The space complexity of this data is of order $|E|$ (which is equivalent to order $|V| + |F|$). They are presented as two tests which if failed imply that the input degree sequence(s) are non-realisable.

Initially integer partitions relating to entire degree sequences are considered, either as a pair or individually. Clearly if a degree sequence is proved unrealisable independent of a pair, there is no need to consider the properties of any pairing with that degree sequence.

The following tests are described:

1. A test on a pair of degree sequences verifying whether the largest face/vertex of one can exist with the face/vertex degrees of the other.

2. A test on individual degree sequences comparing the size of a sequence against the largest degrees.

5.2 Realisability of Prescribed Duals on a Vertex Degree Sequence

5.2.1 Overview

A 3-connected planar (polyhedral) graph has a unique planar embedding. This is because no two faces can share more than two vertices and if two faces share a pair of vertices they must be edge connected and adjacent in the cycles of both faces. Consequently by duality, three faces cannot share a pair of vertices in a 3-connected planar graph. Since a pair of vertices that are shared by two faces must be edge connected and adjacent in the faces, any pair of vertices contained in a face that are not edge connected are unique to that face. As such we can construct an augmented graph containing complementary edges between vertices that share a face but are not adjacent in the face cycle. These edges are unique for each face since any non-adjacent pair of vertices exists only in one face.

It is possible to count these complementary edges from face degree sequence since they are not dependent on the embedding.

5.2.2 Definitions

A **complementary edge** of a 3-connected planar graph is an edge between any two vertices which share a face but are not edge-connected.

A **planar complement** of a 3-connected planar graph is the set of all complementary edges.

Theorem 5.2.1 (Planar Complement Size of a 3-connected graph)**.** *Given a set of faces $F = \{f_1, f_2, \ldots, f_{|F|}\}$ of a 3-connected planar graph, where the degree of each face f is defined as $d(f)$. The number of edges in the planar complement is:*

$$\sum_{f \in F} \frac{d(f)^2}{2} - 3|E|.$$

Where $|E|$ is the number of edges in the planar graph.

Proof

We first determine the number of complementary edges in a single face. In Figure 5.1 the complementary edges in each of the faces are shown by red dashed lines. We see that each vertex shares a complementary edge with any vertex that is not cyclically adjacent to it in the face and is not itself. Therefore for Figure 5.1(b) where we have four vertices in a quadrilateral face, each vertex can share a complementary edge with one other.

The number of complementary edges emanating from each vertex increases at the same rate as the face degree. Hence, for a pentagonal face there are two complementary edges emanating from each vertex, a hexagonal face has three complementary edges emanating from each vertex and so on. Therefore, a face f with $d(f)$ vertices has $d(f)-3$ complementary edges emanating from each vertex. Since each complementary edge is shared by two vertices to calculate the number of complementary edges in the face we can sum the number of complementary edges emanating from each vertex and divide by two. Hence, for a face f with $d(f)$ vertices, the number of complementary edges is:

$$\frac{d(f)(d(f)-3)}{2}.$$

To find the number of edges in the planar complement of the entire graph we sum this for all faces:

$$\sum_{f\in F}\frac{d(f)(d(f)-3)}{2}=\sum_{f\in F}\frac{d(f)^2}{2}-3\sum_{f\in F}\frac{d(f)}{2}.$$

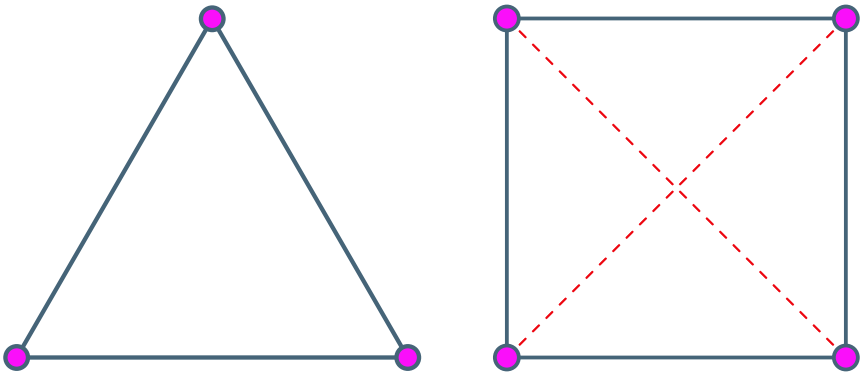
Recall the sum of the faces degrees is equal to the sum of the vertex degrees and is twice the number of edges in a planar graph. Hence

$$\sum_{f\in F}\frac{d(f)}{2}=|E|$$

.

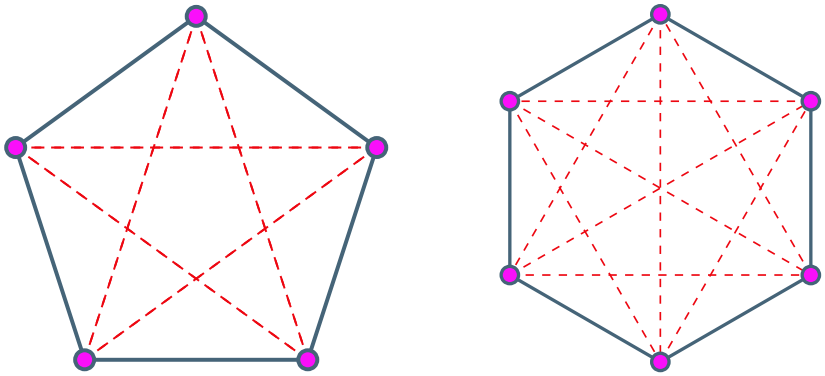
$$\sum_{f\in F}\frac{d(f)^2}{2}-3\sum_{f\in F}\frac{d(f)}{2}=\sum_{f\in F}\frac{d(f)^2}{2}-3|E|.$$

Thus proving Theorem 5.2.1. □



(a) Triangular face with no complementary edges

(b) Quadrilateral face with two complementary edges



(c) Pentagonal face with five complementary edges

(d) Hexagonal face with nine complementary edges

Figure 5.1: Polygonal faces with complementary edges

Consequently the size of the planar complement of a 2-vertex 3-edge connected planar graph is a strict inequality.

Lemma 5.2.2 (Planar Complement Size of an exactly 2-vertex, 3-edge connected planar graph)**.** *Given a face set F of an exactly 2-vertex, 3-edge connected planar graph with face degree sequence:*

$$D_F = \{d(f_1), d(f_2), \ldots, d(f_{|F|})\}, \text{ where } f_i \in F.$$

The size of the planar complement is strictly less than:

$$\sum_{f \in F} \frac{d(f)^2}{2} - 3|E|.$$

Proof

If a planar graph is exactly 2-vertex, 3-edge connected, then there exists a pair

of vertices that disconnect the graph. There are two cases to consider:

Case 1: There exists a pair of vertices that are edge connected and are a vertex-cut set that disconnect the graph.

Case 2: There exists a pair of vertices that are not edge connected and are a vertex-cut set that disconnect the graph.

An example of Case 1 can be seen in Figure 5.2 where vertices v_2 and v_3 are edge connected and disconnect the graph. When this is the case a pair of vertices exist in three faces, they are cyclically adjacent in two internal faces and hence are edge connected. However, they also exist in the outer face of the planar embedding where they are not cyclically adjacent. Since the formula in Theorem 5.2.1 would count this pair as a complementary edge it will always be greater than the actual number of complementary edges in the graph.

An example of Case 2 can be seen in Figure 5.4 where the pair of vertices v_1 and v_2 are not edge connected but are a vertex-cut set (the pair of vertices v_1 and v_3 would also disconnect the graph). In this case we see that the pair of vertices share two faces, but are not cyclically adjacent in either of them. In this case if attempting to calculate the number of complementary edges using the formula in Theorem 5.2.1 a complementary edge would be counted twice as it exists in two faces. Therefore the formula will always calculate a number greater than the actual number of complementary edges in the graph.

Hence for both cases the actual number of complementary edges in an exactly 2-vertex, 3-edge connected planar graph is strictly less than a 3-connected planar graph with the same face degree sequence, we prove Lemma 5.2.2. □

5.2.3 Theory

Given a pair of degree sequences D_V and D_F which prescribe the vertex and face degrees of a 3-connected planar graph, we can infer from the planar complement some restrictions on the degrees of vertices which can exist about the faces. For a face f in F with a face degree $d(f)$, the maximum degree of any vertex about the face is $|V| - d(f_i) + 2$.

We can also make a constraint on the sum of the maximum face degrees about any vertex. For a vertex v in V with vertex degree $d(v)$ the sum of the degrees of the faces about the vertex must be strictly less than $|V| + 2d(v)$.

The test is therefore defined as follows take the maximum face degree $d(f)$ in D_F, verify that there are $d(f)$ vertices with a maximum degree of $|V| - d(f_i) + 2$. Repeat this test for the maximum vertex degree $d(v)$ in D_V and verifying there are $d(v)$ faces with a maximum face degree of $|F| - d(v_i) + 2$. This test requires two linear computations of $O(|V|)$ and $O(|F|)$. Since the graphs are planar we have that computational complexity is $O(|E|)$ from Euler's Formula [Euler, 1758].

5.2.4 2-vertex, 3-edge connectivity Counterexample

The criteria for vertex degrees about faces in embeddings of exactly 2-vertex, 3-edge connected planar graphs is less restrictive. This is because it is possible for a pair of vertices which disconnect the graph to be edge connected. When this is the case a complementary edge of the external face is an edge in the graph and consequently increases the maximum vertex degree permitted about the face.

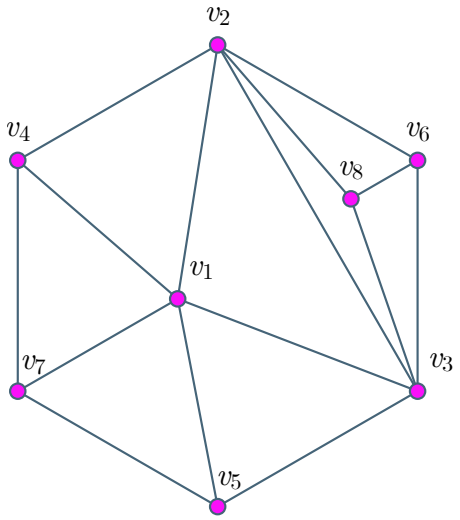


Figure 5.2: Counter example for 2-vertex 3-edge connected graphs

The graph in Figure 5.2 is a 2-vertex 3-edge connected planar graph with the vertex degree sequence $\{5, 5, 5, 3, 3, 3, 3, 3\}$ and the face degree sequence $\{6, 3, 3, 3, 3, 3, 3, 3, 3\}$. A 3-connected planar graph with this pair of degree sequences is non-realisable since there is a face f_1 of degree $d(f_1) = 6$ and $|V| = 8$. This gives the maximum

vertex degree about that face of 4. Therefore the vertex degree sequence would require six vertices with a maximum degree of 4 and there are only five.
However, this degree sequence pair can be realised for an exactly 2-vertex, 3-edge connected planar graph. From the example in Figure 5.2 we can see that the disconnecting pair of vertices v_2 and v_3 are edge connected and share three faces.

- (v_1, v_2, v_3),
- (v_2, v_3, v_8),
- $(v_2, v_4, v_7, v_5, v_3, v_6)$.

In the cyclic order about the face of degree six v_2 and v_3 are not adjacent. Therefore if the graph was 3-connected the vertices would share no other faces and would not be edge connected since the edge would be in the planar complement. As this is not the case for exactly 2-vertex, 3-edge connected graphs we can permit vertices of a higher degree (in this case $d(v_2) = d(v_3) = 5$).

5.2.5 Exclusions Found

Table 5.1 gives a list of pairs of degree sequences which cannot be realised as 3-connected planar graphs due to insufficient vertices with small enough degree. These are the pairs which can be found non-realisable by checking only the single largest degree of both sequences against the smallest degrees of the other. This does not exclude all pairs, since a degree sequence may have multiple large degrees all of which must have their complement edges assignable in the partner degree sequence.
An example of a degree sequence pair which is non-realisable due to the planar complements is:

$$D_V = \{4, 4, 3, 3, 3, 3, 3, 3\} \text{ and } D_F = \{6, 5, 3, 3, 3, 3, 3\}.$$

In this case, since $|V| = 8$ the open degree sequence $D_V^o = \{4, 4, 4, 4, 4, 4, 3, 3\}$. The largest face degree is 6 which requires six vertices with open degrees of at least 3. All vertices have an open degree of at least 3 however, no matter which

are assigned to the face of degree 6, we find that the remaining open degrees are not distributed such that the complementary edges of the degree 5 face can be assigned to five vertices.

For the labelled faces $\{f_1, f_2, f_3, \ldots, f_7\}$ with corresponding face degrees $\{6, 5, 3, \ldots, 3\}$ and labelled vertices $\{v_1, v_2, v_3, \ldots, v_8\}$ with corresponding vertex degrees $\{4, 4, 3, \ldots, 3\}$, we find that there are three distinct combinations of vertices that can be assigned to f_1. These will have the following vertex degrees:

$$\{4, 4, 3, 3, 3, 3\},$$

$$\{4, 3, 3, 3, 3, 3\},$$

$$\{3, 3, 3, 3, 3, 3\}.$$

Since $d(f_1) = 6$ we require six open degrees of at least 3. If we consider the open degree sequences of the three combinations after taking complementary edges of f_1 we have:

$$\{0, 0, 1, 1, 1, 1, 3, 3\},$$

$$\{4, 0, 1, 1, 1, 1, 1, 3\},$$

$$\{4, 4, 1, 1, 1, 1, 1, 1\}.$$

The face f_2 has degree $d(f_2) = 5$. This requires five vertices to each have two complementary edges for this face. As we can see, after considering any of the options for assigning vertices to f_1 there are are no five vertices with two remaining open degrees.

Table 5.2 shows the number of degree sequence pairs initially considered, the number non-realisable as a 3-connected planar graph and the number of those which are caught by the Test 1. In the Figure 5.3 the catchrate $\left(\frac{\text{number caught}}{\text{number non-realisable}}\right)$ is plotted for $|V| + |F|$, the total number of degrees across both sequences.

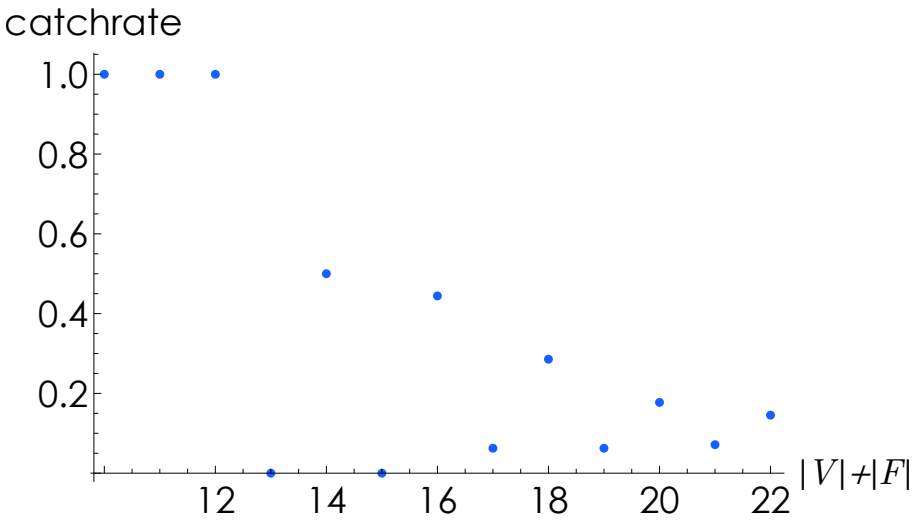


Figure 5.3: Catchrate of degree sequence pairs for Test 1

$\lvert V\rvert + \lvert F\rvert$	D_V	D_F	2-vertex, 3-edge realisable?
12	{5, 3, 3, 3, 3, 3}	{4, 4, 3, 3, 3, 3}	
14	{6, 3, 3, 3, 3, 3, 3}	{5, 4, 3, 3, 3, 3, 3}	
14	{6, 3, 3, 3, 3, 3, 3}	{4, 4, 4, 3, 3, 3, 3}	
16	{7, 3, 3, 3, 3, 3, 3, 3}	{6, 4, 3, 3, 3, 3, 3, 3}	
16	{7, 3, 3, 3, 3, 3, 3, 3}	{5, 5, 3, 3, 3, 3, 3, 3}	
16	{7, 3, 3, 3, 3, 3, 3, 3}	{5, 4, 4, 3, 3, 3, 3, 3}	
16	{7, 3, 3, 3, 3, 3, 3, 3}	{4, 4, 4, 4, 3, 3, 3, 3}	
17	{6, 3, 3, 3, 3, 3, 3, 3, 3}	{5, 5, 5, 3, 3, 3, 3, 3}	yes
18	{8, 3, 3, 3, 3, 3, 3, 3, 3}	{7, 4, 3, 3, 3, 3, 3, 3, 3}	
18	{8, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 5, 3, 3, 3, 3, 3, 3, 3}	
18	{8, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 4, 4, 3, 3, 3, 3, 3, 3}	
18	{8, 3, 3, 3, 3, 3, 3, 3, 3}	{5, 5, 4, 3, 3, 3, 3, 3, 3}	
18	{8, 3, 3, 3, 3, 3, 3, 3, 3}	{5, 4, 4, 4, 3, 3, 3, 3, 3}	
18	{8, 3, 3, 3, 3, 3, 3, 3, 3}	{4, 4, 4, 4, 4, 3, 3, 3, 3}	
19	{7, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 5, 5, 3, 3, 3, 3, 3, 3}	yes
19	{7, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{5, 5, 5, 4, 3, 3, 3, 3, 3}	yes
20	{9, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{8, 4, 3, 3, 3, 3, 3, 3, 3, 3}	
20	{9, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{7, 5, 3, 3, 3, 3, 3, 3, 3, 3}	
20	{9, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{7, 4, 4, 3, 3, 3, 3, 3, 3, 3}	
20	{9, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 6, 3, 3, 3, 3, 3, 3, 3, 3}	
20	{9, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 5, 4, 3, 3, 3, 3, 3, 3, 3}	
20	{9, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 4, 4, 4, 3, 3, 3, 3, 3, 3}	
20	{9, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{5, 5, 5, 3, 3, 3, 3, 3, 3, 3}	
20	{9, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{5, 5, 4, 4, 3, 3, 3, 3, 3, 3}	
20	{9, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{5, 4, 4, 4, 4, 3, 3, 3, 3, 3}	
20	{9, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{4, 4, 4, 4, 4, 4, 3, 3, 3, 3}	
20	{8, 4, 3, 3, 3, 3, 3, 3, 3, 3}	{5, 5, 5, 3, 3, 3, 3, 3, 3, 3}	

Table 5.1: List of degree sequence pairs which fail Test 1

Vertices	Prospective Pairs	Realisable 3-connected	Caught by Test 1	Catch Rate
10	1	0	0	-
11	1	0	0	-
12	3	1	1	1
13	3	1	0	0
14	10	4	2	0.5
15	10	4	0	0
16	23	9	4	0.444444
17	37	16	1	0.0625
18	58	21	6	0.285714
19	86	32	2	0.0625
20	162	62	11	0.177419
21	210	70	5	0.071429
22	345	110	16	0.145455

Table 5.2: Degree sequence pairs caught by Test 1

5.3 Realisability of Large Vertex Degrees with respect to Sequence Size

5.3.1 Overview

An equivalent definition of a 3-connected planar graph is that the every face of degree $d(f)$ shares exactly $d(f)$ edges with $d(f)$ different faces. Consequently we can state that for every consecutive pair of vertices about a face in its face cycle we can find this pair of vertices consecutively (all be it with reversed orientation) in exactly one other face cycle. Through considering how many possible pairings must exist about faces of high degree we can find a lower bound on the number of vertices that must be in the graph.

5.3.2 Theory

Given a vertex degree sequence D_V and a face degree sequence D_F, any two faces can share at most two vertices. Therefore the number of vertices $|V|$ must satisfy the inequality:

$$|V| \geq \sum_{i=1}^{d(f_i)>2(i-1)} d(f_i) - 2(i-1).$$

This lower bound for the number of vertices assumes that all faces of high degree share an edge with each other. Therefore when summing the first two face degrees we can allow two vertices to exist in both. When the first three face degrees, we account for the maximum possible connectivity which would exist when all three faces share three pairs of vertices and hence would discount six from the sum of the face degrees. The order of computation for this summation is $O(|F|)$ since it is a linear operation proportional to the number of faces in the degree sequence.

5.3.3 2-vertex, 3-edge connectivity Counterexample

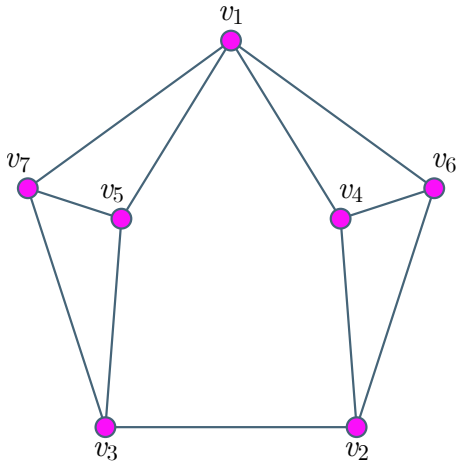


Figure 5.4: Counter example for 2-vertex 3-edge connected graphs

Similarly to Test 1, we find that Test 2 does not hold for 2-vertex, 3-edge connected graphs. In Figure 5.4 we have a graph with face degree sequence $D_F = \{5, 5, 3, 3, 3, 3\}$ and vertex degree sequence $D_V = \{4, 3, 3, 3, 3, 3, 3\}$. We find two faces of degree five share three vertices v_1, v_2, and v_3. The test gives us a lower bound of eight vertices for this face degree sequence if it were 3-connected. This counter example shows the test does not hold for the realisability 2-vertex, 3-edge connected graphs.

5.3.4 Exclusions Found

In Table 5.3 we list the first few cases where Test 2 establishes non-realisability of a pair of degree sequences. In this instance there is actually only one degree sequence used as input. In all cases it is the shorter sequence. This allows multiple pairs of degree sequences to be excluded while only testing one sequence and in any case the test itself requires only each individual sequence to be tested (as opposed to all combinations of viable pairs with the other). Table 5.4 gives the first few numbers of excluded degree sequences along with the catchrate. This catchrate is also shown in Figure 5.5.

$\lvert V\rvert + \lvert F\rvert$	D_V	D_F	2-vertex, 3-edge realisable?
13	{4, 3, 3, 3, 3, 3, 3}	{5, 5, 3, 3, 3, 3}	yes
14	{3, 3, 3, 3, 3, 3, 3, 3}	{5, 5, 5, 3, 3, 3}	
15	{5, 3, 3, 3, 3, 3, 3, 3}	{6, 5, 3, 3, 3, 3, 3}	
15	{4, 4, 3, 3, 3, 3, 3, 3}	{6, 5, 3, 3, 3, 3, 3}	yes
16	{4, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 6, 4, 3, 3, 3, 3}	yes
16	{4, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 5, 5, 3, 3, 3, 3}	
17	{3, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 6, 6, 3, 3, 3, 3}	
17	{3, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 6, 5, 4, 3, 3, 3}	
17	{6, 3, 3, 3, 3, 3, 3, 3, 3}	{7, 5, 3, 3, 3, 3, 3, 3}	
17	{5, 4, 3, 3, 3, 3, 3, 3, 3}	{7, 5, 3, 3, 3, 3, 3, 3}	yes
17	{4, 4, 4, 3, 3, 3, 3, 3, 3}	{7, 5, 3, 3, 3, 3, 3, 3}	yes
17	{6, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 6, 3, 3, 3, 3, 3, 3}	
17	{5, 4, 3, 3, 3, 3, 3, 3, 3}	{6, 6, 3, 3, 3, 3, 3, 3}	
17	{4, 4, 4, 3, 3, 3, 3, 3, 3}	{6, 6, 3, 3, 3, 3, 3, 3}	yes
18	{5, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{7, 7, 3, 3, 3, 3, 3, 3}	
18	{4, 4, 3, 3, 3, 3, 3, 3, 3, 3}	{7, 7, 3, 3, 3, 3, 3, 3}	yes
18	{5, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{7, 6, 4, 3, 3, 3, 3, 3}	
18	{4, 4, 3, 3, 3, 3, 3, 3, 3, 3}	{7, 6, 4, 3, 3, 3, 3, 3}	yes
18	{5, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{7, 5, 5, 3, 3, 3, 3, 3}	yes
18	{4, 4, 3, 3, 3, 3, 3, 3, 3, 3}	{7, 5, 5, 3, 3, 3, 3, 3}	yes
18	{5, 3, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 6, 5, 3, 3, 3, 3, 3}	yes
18	{4, 4, 3, 3, 3, 3, 3, 3, 3, 3}	{6, 6, 5, 3, 3, 3, 3, 3}	yes

Table 5.3: List of degree sequence pairs which fail Test 2

Vertices	Prospective Pairs	Realisable 3-connected	Caught by Test 2	Catch Rate
10	1	0	0	-
11	1	0	0	-
12	3	1	0	0
13	3	1	1	1
14	10	4	1	0.25
15	10	4	2	0.5
16	23	9	2	0.222222
17	37	16	8	0.5
18	58	21	8	0.380952
19	86	32	15	0.46875
20	162	62	24	0.387097
21	210	70	37	0.528571
22	345	110	52	0.472727

Table 5.4: Degree sequence pairs caught by Test 2

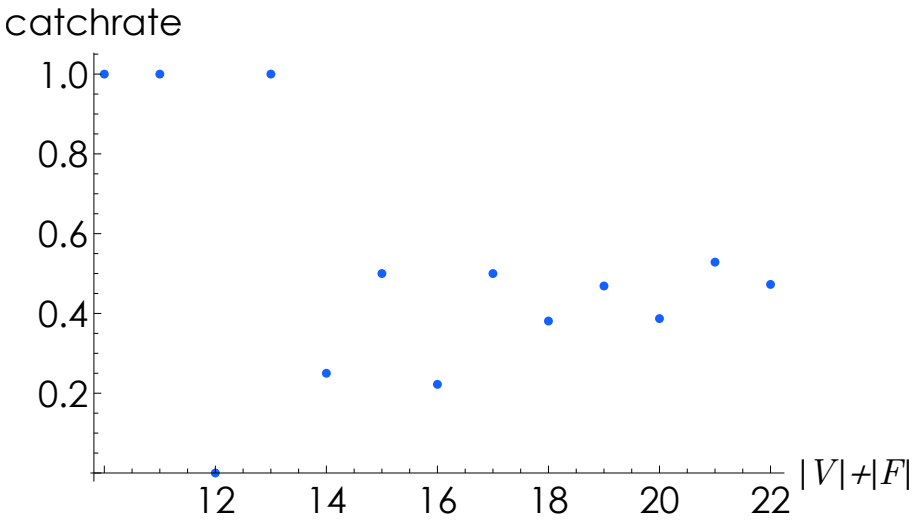


Figure 5.5: Catchrate of degree sequence pairs for Test 2

5.4 Summary

When considering both of these tests together for numbers of $|V| + |F| \leq 22$ excluding at least 50% of the non-realisable pairs of degree sequences for 3-connected planar graphs. Up to these numbers the tests have no overlap in their exclusions meaning that they complement each other well when considering which degree sequences to attempt to enumerate for purely 3-connected (polyhedral) graphs.

When comparing the performance of the two tests we see from Figure 5.6 that the catchrate on Test 2 is significantly greater than that of Test 1 outperforming it for all $|V| > 16$. It also worth noting that the two tests complement performance on odd and even numbers of vertices with Test 1 having a higher catchrate on even numbers of vertices compared to its adjacent odd numbers (e.g. Test 1 has a higher catchrate for $|V| = 20$ than $|V| = 19$ or $|V| = 21$). Similarly Test 2 has higher catchrate on odd numbers of vertices compared to its adjacent even numbers (e.g. Test 2 has a higher catchrate for $|V| = 21$ than $|V| = 20$ or $|V| = 22$).

Further work in this area would be to attempt a proof of the independence of the properties in Tests 1 and 2 for these pairs and consequently identify subsets that are now completely removed. Another interesting property which has held for small numbers is all face degree sequences D_F where $|V| + |F| = 3k$ for $k \in \mathbb{R}$ are found non-realisable by Test 2.

This work not only achieves some promising early results for exclusions of pairs of degree sequences but also proposes further research questions to be pursued in the future.

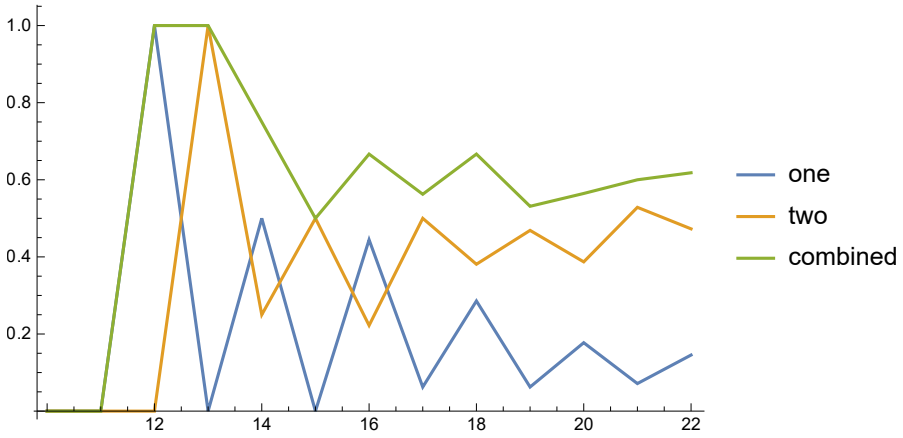


Figure 5.6: Catchrate of both Tests 1 and 2

Chapter 6

Future Work, Potential Applications and Conclusion

6.1 Future Work

This research has so far been concerned with the viability of an algorithm which is embarrassingly parallelisable. The implementation has thus far been to prove parallelisation rather than take advantage of it. The immediate progression of this research is to implement a version of the algorithm for a cluster. Fortunately the algorithm is already designed and written in a functional paradigm using no third party methods other than basic list operations, meaning that a direct translation to C or Fortran is possible from the current code. Since numerical accuracy is not a criteria and C compilers are more widely distributed, the current aspiration is to translate this code to C. As is common with the implementation of parallelisable algorithms, the MPI library will be employed to handle the process by which the program runs on a cluster since network communication and multi-thread processing are not of concern in this research. The integer sequence generated from the numbers of exactly 2-vertex, 3-edge connected planar graphs is yet to be contributed to the Online Encyclopedia of Integer Sequences (`www.oeis.org`) and once significant numbers have been generated from a cluster could be submitted.

Another immediate and crucial avenue of inquiry is the overlap of results for degree sequences that are identical. Currently the algorithm can admit up

to one isomorphic result, the hypothesis is that this could be omitted with an inequality that reflects across the two vertex groups. This would provide a further optimisation to the algorithm design and improve performance.

Further research into the realisability of 2-vertex, 3-edge connected and 3-connected planar graphs with prescribed vertex and face degree sequences will be pursued. The completeness of Test 2 in Chapter 6 for $|D_F|$ where $|V| + |F| = 3k$ for $k \in \mathbb{R}$ along with test independence would also be of great interest in this area.

6.2 Potential Applications

6.2.1 Existence Testing on Embedded Subgraphs

The realisability of vertex and face degree sets of planar graphs and their compatibility with subgraphs is an open question of great interest. There are some non-trivial partial results already known. Grunbaum offered a conjecture that states some conditions on certain types of planar graphs [Grünbaum et al., 1969]. This was then proved by Jendrol [Jendrol' and Jucovič, 1972]. Brinkmann has developed an algorithm for finding all cubic and quartic planar graphs with prescribed face degrees [Brinkmann et al., 2003]. An open question has been asked in [Mihail and Vishnoi, 2002] of whether an efficient algorithm can exist to find a planar graph with prescribed vertex and face degrees if certain restrictions are placed.

Clearly any results that are enumerated to exhaustion for small vertex and face numbers give a working data set to test hypotheses. With the properties focused on in this algorithm there is potential to test variations on this data yet to be considered.

6.2.2 Unfolding

There is a conjecture by Shephard [Shephard, 1975] that any convex polyhedron can be unfolded along its edges into a non overlapping planar net. Counterexamples exist for non-convex polyhedra and several algorithms have been proposed

that could potentially achieve this. However, this is still an open question where any additional properties of 3-connected planar graphs could contribute. The initial scope of any research into this area would be to consider polyhedra constructed solely from quadrilateral faces and look to finding subsets of this group which can be edge unfolded without overlaps. The potential application of this research being that the subdivision of these polyhedra used for the enumeration may have a relationship to how they could unfold.

One possible route of investigation could be to determine if there are any patterns that exist linking the nets of these polyhedra to their decomposed 2-vertex, 3-edge connected planar graphs. For example, are there properties specific to the nets of polyhedra which decompose to exactly 2-vertex, 3-edge connected planar graphs when compared to the nets of polyhedra which decompose to 3-connected planar graphs? There could also be other relationships investigated between the properties of nets and their pairs of vertex degree sequences.

6.2.3 Truncating

A hypothesis which has since been proved by the four colour theorem, proposed that any planar graph can be truncated about it's vertices to create a graph where all vertices are of a degree which is a multiple of three. It has been proven, however a simple proof is yet to be found. Similarly to work on Shephard's conjecture, any additional properties of planar graphs found or data which could lead to other hypothesis is desirable to further research towards such a proof.

6.2.4 Fullerenes

A fullerene is a polyhedron with twelve hexagonal faces and all other faces pentagonal. These polyhedra relate to potential chemical compounds and are of great interest in computational chemistry. Although great work has been done with regards to enumeration [Andova et al., 2016] any underlying properties that persist in these polyhedra, could be of great importance in other research areas.

6.3 Conclusion

This thesis has been concerned with the design of a new enumeration algorithm for the duals of 4-regular polyhedra. The algorithm contributed generates these in a fundamentally different way to state of the art in this area and as such is novel in its approach. An implementation of this code has been made available written in the Wolfram symbolic language and can be accessed at `https://github.com/skamper1/enumeration`.

In Chapter 3 the theory that leads to the decomposition of the 4-regular polyhedral graphs was introduced and invariance of connectivity across duality proved. This lead on to highlighting how exactly 2-vertex, 3-edge connected planar graphs can have their vertex connectivity bounded early in the enumeration due to their fixed embedding.

Chapter 4 gives a detailed treatment of the enumeration process. We first took the polyhedra by number of vertices, then explained the generation of the complete set of degree sequence pairs for the bipartitions. A description of the properties held for each subgraph was then used to show the restrictions in the combinations required for completeness. The subgraph mapping procedure was then demonstrated to highlight how isomorphic results are avoided early in the search tree. A review of the effects of the subgraph mapping procedure and the restrictions in combinations were then shown compared to the omission of these optimisations. The initial results found by the implementation of the algorithm programmed during this research were then presented. We highlighted where this has relationships to other data already generated. It also showed how there are open questions between numbers of graphs and embeddings in the exactly 2-vertex, 3-edge connected case.

Chapter 5 proved two tests for non-realisability of degree sequences for 3-connected planar graphs. This left several open questions on the tests proposed and there is yet to be any such work for 2-vertex, 3-edge connected planar graphs.

In this Chapter other such open problems relating to these graphs are then posed along with other applications. The breadth of potential applications that can take advantage of this work provides many interesting avenues for future

research. Moving forward, more research exploiting these properties of 4-regular 3-connected planar graphs could give rise to new results.

The primary distinctions of this algorithm are the problem subdivision by pairs of degree sequences and the parallelisation of the enumeration. When a new vertex is connected from the first bipartition, all possible unique subgraphs are generated, these are endowed with all the data they require to work independently of each other without ever needing to feedback for verification. As such the algorithm not only only works in parallel but can scale without requiring data access. This provides flexible task scheduling since all jobs or nodes that need to be processed do not communicate other than to output new nodes of work. This parallelisation means that the algorithm can take full advantage of a cluster and is not limited by the amount of separate machines that can be used concurrently.

References

[Andova et al., 2016] Andova, V., Kardoš, F., and Škrekovski, R. (2016). Mathematical aspects of fullerenes. *ARS MATHEMATICA CONTEMPORANEA*, 11(2):353–379.

[Bau, 2016] Bau, S. (2016). Reductions of 3-Connected Quadrangulations of the Sphere. In Adiprasito, K., Bárány, I., and Vilcu, C., editors, *Convexity and Discrete Geometry Including Graph Theory: Mulhouse, France, September 2014*, pages 231–238. Springer International Publishing, Cham.

[Brinkmann et al., 2005] Brinkmann, G., Greenberg, S., Greenhill, C., McKay, B. D., Thomas, R., and Wollan, P. (2005). Generation of simple quadrangulations of the sphere. *Discrete Mathematics*, 305(1-3):33–54.

[Brinkmann et al., 2003] Brinkmann, G., Harmuth, T., and Heidemeier, O. (2003). The construction of cubic and quartic planar maps with prescribed face degrees. *Discrete Applied Mathematics*, 128(2-3):541–554.

[Cauchy, 1813a] Cauchy, A. L. (1813a). Recherches sur les polyedres. *Journal de l'Ecole Polytechnique*, 9(a):68–86.

[Cauchy, 1813b] Cauchy, A. L. (1813b). Recherches sur les polygones et les polyedres. *Journal de l'Ecole Polytechnique*, 9(b):87–98.

[Croft et al., 1991] Croft, H. T., Falconer, K. J., and Guy, R. K. (1991). *Unsolved Problems in Geometry*, volume 2 of *Problem Books in Mathematics*. Springer New York, New York, NY.

[Cromwell, 1999] Cromwell, P. R. (1999). *Polyhedra*. Cambridge University Press.

[Dillencourt, 1996] Dillencourt, M. B. (1996). Polyhedra of small order and their hamiltonian properties. *journal of combinatorial theory, Series B*, 66(1):87–122.

[Duijvestijn and Federico, 1981] Duijvestijn, A. J. W. and Federico, P. J. (1981). The Number of Polyhedral (3-Connected Planar) Graphs. *Mathematics of Computation*, 37(156):523.

[Erdös and Gallai, 1960] Erdös, P. and Gallai, T. (1960). Graphs with prescribed degrees of vertices (Hungarian). *Mat. Lapok*, 11:264–274.

[Euler, 1758] Euler, L. (1758). Elementa doctrinae solidorum. *Novi Commentarii Academiae Scientarum Petropolitanae*, 4(1758):109–140.

[Grünbaum et al., 1967] Grünbaum, B., Klee, V., Perles, M. A., and Shephard, G. C. (1967). *Convex polytopes*, volume 16. Springer.

[Grünbaum et al., 1969] Grünbaum, B., Rademacher, H., Steinitz, E., and Rademacher, H. (1969). Planar maps with prescribed types of vertices and faces. *Mathematika*, 16(01):28.

[Hakimi, 1962] Hakimi, S. L. (1962). On Realizability of a Set of Integers as Degrees of the Vertices of a Linear Graph. I. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506.

[Havel, 1955] Havel, V. (1955). A remark on the existence of finite graphs (Czech). *Časopis Pěst. Mat.*, 080(4):477–480.

[Hopcroft and Tarjan, 1974] Hopcroft, J. and Tarjan, R. (1974). Efficient planarity testing. *Journal of the ACM (JACM)*, 21(4):549–568.

[Hopcroft and Wong, 1974] Hopcroft, J. E. and Wong, J. K. (1974). Linear Time Algorithm for Isomorphism of Planar Graphs (Preliminary Report). In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, pages 172–184, New York, NY, USA. ACM.

[Jendrol' and Jucovič, 1972] Jendrol', S. and Jucovič, E. (1972). On a conjecture by B. Grünbaum. *Discrete Mathematics*, 2(1):35–49.

[Kápolnai, 2014] Kápolnai, R. (2014). *Restricted generation of quadrangulations and scheduling parameter sweep applications*. PhD thesis, Budapesti Műszaki és Gazdaságtudományi Egyetem.

[Kuratowski, 1930] Kuratowski, C. (1930). Sur le probleme des courbes gauches en topologie. *Fundamenta mathematicae*, 15(1):271–283.

[Lehel, 2006] Lehel, J. (2006). Generating all 4-regular planar graphs from the graph of the Octahedron. *Journal of Graph Theory*, 5:423 – 426.

[Manca, 1979] Manca, P. (1979). Generating all planer graphs regular of degree four. *Journal of graph theory*, 3(4):357–364.

[Mihail and Vishnoi, 2002] Mihail, M. and Vishnoi, N. K. (2002). On generating graphs with prescribed vertex degrees for complex network modeling. *Position Paper, Approx. and Randomized Algorithms for Communication Networks (ARACNE)*, 142.

[Nishizeki and Chiba, 1988] Nishizeki, T. T. and Chiba, N. N. (1988). *Planar graphs : theory and algorithms*. North-Holland.

[Noy et al., 2017] Noy, M., Requilé, C., and Rué, J. (2017). Enumeration of labeled 4-regular planar graphs. *Electronic Notes in Discrete Mathematics*, 61:933–939.

[Read, 1978] Read, R. C. (1978). Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. *Annals of Discrete Mathematics*, 2:107–120.

[Shephard, 1975] Shephard, G. C. (1975). Convex polytopes with convex nets. *Mathematical Proceedings of the Cambridge Philosophical Society*, 78(3):389–403.

[Sloane, 1994] Sloane, N. J. A. (1994). Number of 4-regular polyhedra with n nodes.

[Steinitz and Rademacher, 1934] Steinitz, E. and Rademacher, H. (1934). *Vorlesungen uber die theorie der polyeder: unter einschluss der elemente der topologie*. J. Springer.

[Tutte, 1961] Tutte, W. T. (1961). A theory of 3-connected graphs. In *Indagationes Mathematicae (Proceedings)*, volume 64, pages 441–455.

[Tutte, 1963] Tutte, W. T. (1963). How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1):743–767.

[Wilson, 2010] Wilson, R. J. (2010). *Introduction to graph theory*. Prentice Hall/Pearson.

Appendix A

Enumeration Algorithm Results

These tables were created with (`https://github.com/skamper1/enumeration`).

Each of the tables A.1 to A.7 give the numbers of duals of 4-regular, 3-connected planar graphs with $|V| = 10, 11, \ldots, 16$. They list all degree sequence pairs considered for a given $|V|$. The column $\kappa(G) < 3$ is the frequency of results that decompose to exactly 2-vertex, 3-edge connected graphs, $\kappa(G) = 3$ is the frequency of results that decompose to 3-connected graphs and the total column is the sum of the previous two. The Nodes column is the number of nodes computed by the algorithm in the search tree for each case. Degree sequence pairs which yield no results but are currently considered are given for completion since nodes are still computed in the search tree when the algorithm is run.

D_1	D_2	$\kappa(G) < 3$	$\kappa(G) = 3$	Total	Nodes
$\{4,3,3,3,3\}$	$\{4,3,3,3,3\}$	0	1	1	7
Total		**0**	**1**	**1**	**7**

Table A.1: Results computed for 10 vertices

D_1	D_2	$\kappa(G) < 3$	$\kappa(G) = 3$	Total	Nodes
$\{3,3,3,3,3,3\}$	$\{4,4,4,3,3\}$	0	1	1	19
Total		**0**	**1**	**1**	**19**

Table A.2: Results computed for 11 vertices

D_1	D_2	$\kappa(G) < 3$	$\kappa(G) = 3$	Total	Nodes
$\{5,3,3,3,3,3\}$	$\{5,3,3,3,3,3\}$	0	1	1	9
$\{5,3,3,3,3,3\}$	$\{4,4,3,3,3,3\}$	0	0	0	9
$\{4,4,3,3,3,3\}$	$\{4,4,3,3,3,3\}$	1	1	2	16
Total		**1**	**2**	**3**	**34**

Table A.3: Results computed for 12 vertices

D_1	D_2	$\kappa(G) < 3$	$\kappa(G) = 3$	Total	Nodes
$\{4,3,3,3,3,3,3\}$	$\{5,5,3,3,3,3\}$	1	0	1	16
$\{4,3,3,3,3,3,3\}$	$\{5,4,4,3,3,3\}$	0	1	1	34
$\{4,3,3,3,3,3,3\}$	$\{4,4,4,4,3,3\}$	0	1	1	31
Total		**1**	**2**	**3**	**81**

Table A.4: Results computed for 13 vertices

D_1	D_2	$\kappa(G) < 3$	$\kappa(G) = 3$	Total	Nodes
$\{3,3,3,3,3,3,3,3,3\}$	$\{5,5,5,3,3,3\}$	0	0	0	38
$\{3,3,3,3,3,3,3,3,3\}$	$\{5,5,4,4,3,3\}$	0	1	1	166
$\{3,3,3,3,3,3,3,3,3\}$	$\{5,4,4,4,4,3\}$	0	0	0	19
$\{3,3,3,3,3,3,3,3,3\}$	$\{4,4,4,4,4,4\}$	0	1	1	10
$\{6,3,3,3,3,3,3\}$	$\{6,3,3,3,3,3,3\}$	0	1	1	13
$\{6,3,3,3,3,3,3\}$	$\{5,4,3,3,3,3,3\}$	0	0	0	17
$\{6,3,3,3,3,3,3\}$	$\{4,4,4,3,3,3,3\}$	0	0	0	27
$\{5,4,3,3,3,3,3\}$	$\{5,4,3,3,3,3,3\}$	1	1	2	30
$\{5,4,3,3,3,3,3\}$	$\{4,4,4,3,3,3,3\}$	1	1	2	43
$\{4,4,4,3,3,3,3\}$	$\{4,4,4,3,3,3,3\}$	0	4	4	78
Total		**2**	**9**	**11**	**441**

Table A.5: Results computed for 14 vertices

D_1	D_2	$\kappa(G) < 3$	$\kappa(G) = 3$	Total	Nodes
$\{5,3,3,3,3,3,3,3,3\}$	$\{6,5,3,3,3,3,3,3\}$	0	0	0	25
$\{4,4,3,3,3,3,3,3,3\}$	$\{6,5,3,3,3,3,3,3\}$	1	0	1	33
$\{5,3,3,3,3,3,3,3,3\}$	$\{6,4,4,3,3,3,3\}$	1	1	2	69
$\{4,4,3,3,3,3,3,3,3\}$	$\{6,4,4,3,3,3,3\}$	1	1	2	96
$\{5,3,3,3,3,3,3,3,3\}$	$\{5,5,4,3,3,3,3\}$	1	0	1	104
$\{4,4,3,3,3,3,3,3,3\}$	$\{5,5,4,3,3,3,3\}$	2	2	4	140
$\{5,3,3,3,3,3,3,3,3\}$	$\{5,4,4,4,3,3,3\}$	0	1	1	66
$\{4,4,3,3,3,3,3,3,3\}$	$\{5,4,4,4,3,3,3\}$	1	4	5	156
$\{5,3,3,3,3,3,3,3,3\}$	$\{4,4,4,4,4,3,3\}$	0	0	0	67
$\{4,4,3,3,3,3,3,3,3\}$	$\{4,4,4,4,4,3,3\}$	0	2	2	72
Total		**7**	**11**	**18**	**828**

Table A.6: Results computed for 15 vertices

D_1	D_2	$\kappa(G) < 3$	$\kappa(G) = 3$	Total	Nodes
$\{4,3,3,3,3,3,3,3,3,3\}$	$\{6,6,4,3,3,3,3\}$	1	0	1	150
$\{4,3,3,3,3,3,3,3,3,3\}$	$\{6,5,5,3,3,3,3\}$	0	0	0	47
$\{4,3,3,3,3,3,3,3,3,3\}$	$\{6,5,4,4,3,3,3\}$	1	2	3	212
$\{4,3,3,3,3,3,3,3,3,3\}$	$\{6,4,4,4,4,3,3\}$	0	1	1	87
$\{4,3,3,3,3,3,3,3,3,3\}$	$\{5,5,5,4,3,3,3\}$	0	2	2	460
$\{4,3,3,3,3,3,3,3,3,3\}$	$\{5,5,4,4,4,3,3\}$	0	2	2	456
$\{4,3,3,3,3,3,3,3,3,3\}$	$\{5,4,4,4,4,4,3\}$	0	1	1	58
$\{4,3,3,3,3,3,3,3,3,3\}$	$\{4,4,4,4,4,4,4\}$	0	0	0	9
$\{7,3,3,3,3,3,3,3\}$	$\{7,3,3,3,3,3,3,3\}$	0	1	1	17
$\{7,3,3,3,3,3,3,3\}$	$\{6,4,3,3,3,3,3,3\}$	0	0	0	30
$\{7,3,3,3,3,3,3,3\}$	$\{5,5,3,3,3,3,3,3\}$	0	0	0	19
$\{7,3,3,3,3,3,3,3\}$	$\{5,4,4,3,3,3,3,3\}$	0	0	0	42
$\{7,3,3,3,3,3,3,3\}$	$\{4,4,4,4,3,3,3,3\}$	0	0	0	82
$\{6,4,3,3,3,3,3,3\}$	$\{6,4,3,3,3,3,3,3\}$	1	1	2	47
$\{5,5,3,3,3,3,3,3\}$	$\{6,4,3,3,3,3,3,3\}$	0	0	0	43
$\{6,4,3,3,3,3,3,3\}$	$\{5,4,4,3,3,3,3,3\}$	4	1	5	133
$\{6,4,3,3,3,3,3,3\}$	$\{4,4,4,4,3,3,3,3\}$	1	0	1	115
$\{5,5,3,3,3,3,3,3\}$	$\{5,5,3,3,3,3,3,3\}$	0	1	1	34
$\{5,5,3,3,3,3,3,3\}$	$\{5,4,4,3,3,3,3,3\}$	3	1	4	132
$\{5,5,3,3,3,3,3,3\}$	$\{4,4,4,4,3,3,3,3\}$	3	1	4	121
$\{5,4,4,3,3,3,3,3\}$	$\{5,4,4,3,3,3,3,3\}$	5	9	14	342
$\{5,4,4,3,3,3,3,3\}$	$\{4,4,4,4,3,3,3,3\}$	1	7	8	197
$\{4,4,4,4,3,3,3,3\}$	$\{4,4,4,4,3,3,3,3\}$	1	7	8	403
Total		**21**	**37**	**58**	**3236**

Table A.7: Results computed for 16 vertices