Certificates for Properties of Chromatic Polynomials of Graphs

by

Zoe Bukovac

Thesis

This thesis is presented in partial fulfilment
of the requirements for the degree of

Bachelor of Computer Science with Honours

at Monash University

Supervisors: Graham Farr
Kerri Morgan

Contents

List of Tables

List of Figures

Abstract

The chromatic polynomial gives the number of possible colourings of a graph for some number of available colours. In general, calculating the chromatic polynomials of graphs is hard, so any means of reducing the difficulty of finding out information about them are of interest. Certificates of equivalence can provide proofs about the relationships between the chromatic polynomials of graphs. The lengths of these proofs may provide insight into the computational complexity of determining if two non-isomorphic graphs share the same chromatic polynomial. In this thesis a new linear bound on the lengths of certificates of equivalence for some specific types of tree graphs is given. An algorithm for finding short certificates of equivalence using a minimal set of certification steps is designed and implemented. This algorithm is used to find the shortest certificates of equivalence for all graphs of order $n \leq 7$. A linear upper bound of $2(n-3)$ on the length of shortest certificates for an infinite class of tree graphs is also introduced. We conjecture that this bound holds for all trees.

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Zoe Elizabeth Bukovac
November 26, 2012

Acknowledgements

I am grateful to my supervisors Prof. Graham Farr and Dr. Kerri Morgan for their time and advice. They are both outstanding supervisors; their attention to the little details, generosity with meetings, and unwavering support have been invaluable. Graham is a superb source of guidance regarding academic writing and academia in general. Kerri is always full of enthusiasm and new ideas. She encouraged me to undertake honours and I will always be grateful to her for her faith in me.

Thanks are due to the members of the Discrete Mathematics research group at Monash University for their advice and for broadening my understanding of the field.

I am thankful for the helpful feedback provided over the course of this project by my examiners David Albrect and John Betts. I also thank the staff of the Clayton School of Information Technology for all of their assistance and encouragement.

I thank my parents Diana and Branko Bukovac for their love and support.

I would also like to thank Chris, whom, without this project, I might never have met, and without whom this project might never have been completed.

Zoe Bukovac

Monash University
November 2012

1 Introduction

A graph is a collection of vertices and a collection of edges that link pairs of vertices. Due to their nature as representations of relationships between objects, graphs lend themselves to many purposes and they are regularly used to model many problems in a diverse range of fields. Applications exist in various engineering disciplines, including the analysis of electrical circuits and their design, the design of traffic networks and the layout of facilities within buildings [19]. They have been used in physics to represent the atomic scale structure of crystals [7], in chemistry to model the structure of molecules [9] and in biology to describe evolutionary trees and genetic structure [35]. There are applications in the social sciences, economics, logistics, game theory and computer science [19].

The first known application of graph theory is attributed to Euler in 1736 [5]. The town of Königsberg had seven bridges and it was a common game to attempt to walk through the town while crossing each of the bridges exactly once. No-one could find a successful route, so it was believed to be impossible. By expressing the problem as a graph where the edges represented the bridges, Euler was able to show that such attempts were indeed in vain, as there was no possible path that allowed exactly one crossing of each bridge [5].

Graphs are useful representations of problems in many situations, but they are associated with many computationally difficult problems. Some of these problems involve colourings of graphs. A *colouring* of a graph assigns a colour to each vertex of a graph such that adjacent vertices receive different colours. Many aspects of graph theory grew out of one of the oldest graph colouring problems, known as the Four Colour Conjecture, now known as the Four Colour Theorem, which was posed in 1852. It asked if every map could be coloured with at most four colours in such a way that no countries that shared a border received the same colour. Although easy to state, it ultimately proved very difficult to solve. Since a map can be represented as a graph, where the vertices represent countries and the edges represent borders, it is clearly a graph colouring problem. Despite its simple appearance, its proof remained elusive for many years, and all existing proofs for the problem are computer generated and are too complex to verify by hand [1, 2, 21, 40].

Even though much of the work on proving the Four Colour Theorem did not result in a formal proof or resulted in proposed proofs that were later shown to be wrong, a great deal of insight into the nature of graph colouring problems in general was gained. The chromatic polynomial $P(G;\lambda)$, first introduced by Birkhoff in [6], gives the number of possible colourings of a graph G using at most λ colours. Ultimately, it did not yield a proof for of the Four Colour Theorem [50], but the chromatic polynomial has since been the subject of a great deal of investigation which has revealed a number of interesting properties [17].

Calculating the chromatic polynomial of a graph is at least as computationally hard as determining the smallest number of colours needed to colour a graph [39], which is known to be NP-complete [25]. As a result, any method that can reveal information about the chromatic polynomial of a graph without needing to calculate it is of interest. The certificates for the chromatic polynomial introduced by Morgan and Farr [31, 32, 33, 34] provide such a method by demonstrating how the chromatic polynomials of graphs are related.

In this project we consider certificates that can be used to show that two different graphs have the same chromatic polynomial. In order to produce computationally feasible software, we restrict these certificates to a more compact subset of possible operations. An algorithms for finding short certificates was designed and implemented, along with some additional software tools for investigating certificate structures. We give a linear bound on the lengths of certificates of equivalence for some specific types of tree graphs. An algorithm for finding short certificates of equivalence using a minimal set of certification steps is designed and implemented. This algorithm is used to find the shortest certificates for all chromatically equivalent graphs of order $n \leq 7$.

1.1 Thesis Structure

This thesis is structured as follows.

Chapter 2 discusses the wider context of this work. A brief overview of graph colouring is presented. The chromatic polynomial's origins and various expressions are considered, along with a discussion of its coefficients and roots. The concepts of chromatic equivalence and factorisation are introduced. Certificates are discussed from both a general computational complexity perspective as well as within the context of the chromatic polynomial.

Chapter 3 introduces some new terminology for describing certificates for the chromatic polynomial. A proof for the length of certificates of equivalence for a particular infinite family of graphs is given. Some important corollaries of this proof is outlined.

Chapter 4 explains the reasoning behind selection of a minimal set of certification steps for the certificates in this project. A description of the algorithms used to find certificates in this research is given.

A discussion of the experimental procedure is given in Chapter 5. The certificates of equivalence found during the experimental tests are considered along with their schemas. We present a conjecture regarding the lengths of certificates for pairs of chromatically equivalent trees.

Conclusions and a discussion of possible further work are detailed in Chapter 6.

1.2 Main Contributions

The following were achieved in this project:

- Found an linear upper bound of $2(n-3)$ on the length of shortest certificates for an infinite family of tree graphs.
- Designed an algorithm for finding shortest certificates of equivalence using a minimal set of certification steps.
- Produced the `certsearch` software for generating and searching for certificates.
- Found the shortest certificates for all chromatically equivalent pairs of graphs of order ≤ 7.

- Found the schemas for the above certificates for all chromatically equivalent pairs of graphs of order ≤ 6.

Experimental results suggest the following conjecture:

- Conjecture: For all pairs of chromatically equivalent trees of order n there exists certificate of equivalence of length $\leq 2(n-3)$.

1.3 Definitions

In this section some formal definitions are given that will be used in this document.

A *graph* $G = (V,E)$ is a set of *vertices* V, often denoted with $V(G)$, together with a set of *edges* $E \subseteq \{\{u,v\} \mid u \in V, v \in V, \text{ and } u \neq v\}$, often denoted with $E(G)$. Two vertices u and v are said to be *adjacent in* G if the edge $\{u,v\}$ (usually written as uv) belongs to E; we also say, in such a case, that the edge uv is *incident to* the vertices u and v. On the other hand, if $uv \notin E$, then we often call uv a *non-edge of* G. The *degree* of a vertex is the number of edges incident to the vertex. Furthermore, The number $|V(G)|$ of vertices in G is called the *order of* G, whereas the number $|E(G)|$ of edges in G is called the *size of* G. We shall often write n and m for $|V(G)|$ and $|E(G)|$ respectively.

The following graph operations are fundamental to our work. Let u and v be vertices in G. If uv is an edge of G, then we can form a new graph, denoted $G \setminus uv$, by removing the edge uv from G. We call this process *edge deletion*. In contrast, if uv is a non-edge of G, then we can insert an edge between u and v to obtain a new graph $G + uv$. We call this process *edge addition*. Finally, and regardless of whether uv is an edge in G, we can derive a new graph G/uv by merging u with v into a single vertex x, whose adjacencies are inherited from u and v to give the following edge set:

$$\{xy \mid uy \in E(G) \text{ or } vy \in E(G), \text{ but } u \neq y \neq v\}.$$

We call this process *vertex identification*; however, when uv is an edge in G, we shall often call it *edge contraction*.

A *tree* is a graph which contains no cycles. A graph $G' = (V',E')$ is said to be a *subgraph* of $G = (V,E)$ if $V' \subseteq V$ and $E' \subseteq E$. A subgraph H of a graph G is said to be *induced* if, for any pair of vertices $u,v \in H$, uv is an edge in H if and only if uv is an edge in G.

The graphs $G = (V,E)$ and $G' = (V',E')$ are *isomorphic* if there exists a bijection $\varphi : V \to V'$ with $xy \in E \Leftrightarrow \varphi(x)\varphi(y) \in E'$. In this research two isomorphic graphs G and G' are considered to be identical and we write $G \simeq G'$.

A *colouring* of a graph G is a function $f : V(G) \to \Lambda$ such that if two vertices u and v are adjacent then $f(u) \neq f(v)$. Λ is the set of available colours and a λ-colouring is a colouring of a graph which uses at most $|\Lambda|$ colours. A graph for which there exists a λ-colouring is said to be λ-colourable. The smallest number of colours needed to colour a graph G is known as the *chromatic number* and is written as $\chi(G)$.

A *clique* is subset of the vertices of a graph where each pair of vertices in the subset is connected by an edge. A clique with r vertices is an *r-clique* or K_r, where K_r is the *complete* graph on r vertices.

A graph G is said to have a *chordless cycle* if G has an induced subgraph with vertex set $V' \subseteq V$ and $E' = \{uv \in E \mid u,v \in V'\}$ that is a cycle. A *chordal graph* is a graph that contains no chordless cycle of size > 3

If two disjoint graphs H_1 and H_2 both contain a clique of at least size r then the graph G formed by identifying an r-clique in H_1 with an r-clique in H_2 is an *r-gluing*. A graph obtained by an r-gluing of the two graphs is said to be a *clique-separable* graph.

We conclude with some miscellaneous definitions. If k is a natural number, then we often write the set $\{1,\ldots,k\}$ as $[k]$. When working with two sequences ρ and σ, we shall use $\rho\sigma$ to denote the concatenation of the two such that ρ is followed by σ.

Furthermore, if x is some object, then the sequence comprising x alone will often be written as x, when the context is clear. The sequence comprising no entries is called the *empty sequence*, and shall be denoted with ϵ.

2 Literature Review

2.1 Graph Colouring

Graph colouring has important practical applications to scheduling [14] and register allocation problems [10]. Graph colouring problems are in some sense concerned with partitioning a set of objects into classes according to a set of rules. Vertices represent these objects and two vertices with an edge between them are restricted from belonging to the same class. Vertices assigned the same colour are considered to belong to the same class. In many graphs however, it is possible to the partition vertices in more than one way, so that for some graph G with two non-adjacent vertices $u,v \in V(G)$, u and v may belong to the same class in one partitioning, but in some other partitioning they may not. All of these possible partitionings contribute to the total sum of conditions needed to determine if a graph is λ-colourable and what its chromatic number is.

With the exception of a few cases, graph colouring is computationally hard. Both determining a graph's chromatic number, and determining for all $\lambda > 2$ if a graph is λ-colourable are NP-complete problems [24].

2.1.1 The Four Colour Theorem

Many areas of graph theory can trace their origins to the Four Colour Conjecture. It was first posed by Francis Guthrie and presented by his brother, Frederick Guthrie, to DeMorgan in 1852 [50] and asked if every map can be coloured with just four colours so that no adjacent countries receive the same colour. A map can be represented as a *planar* graph, which is a graph that can be drawn in the plane so that no two edges cross. By representing the countries with vertices and the borders between them with edges, the problem become one of determining if every planar graph is 4-colourable.

There were numerous attempts to find a proof for what is now known as the Four Colour Theorem. In the years between its first proposition and the proof finally produced in 1977 there was a wealth of research into the problem and several attempted proofs, including one by Kempe [26] and another by Tait [42], which were accepted for a short time before being shown to be incorrect [50]. Much of this work did however contribute other results in graph theory, including Birkhoff's introduction of the chromatic polynomial [6].

Finally proven by Appel and Haken [1, 2] and subsequently given further (but related) proofs by Robertson et al. [40] and Gonthier [21], the proof has been largely accepted by the mathematical community. However, due to the computationally intensive method which Appel and Haken used to produce their proof it is impossible to check it by hand in all of its detail, leaving some to remain unsatisfied with what they consider to be a mathematically inelegant proof [50].

2.2 The Chromatic Polynomial

2.2.1 Origins and Applications

The *chromatic polynomial* $P(G;\lambda)$ of a graph G gives the number of λ-colourings of G. As mentioned earlier, it was first introduced by Birkhoff as a possible algebraic approach for a proof for the Four Colour Theorem [6]. While it did not yield the desired proof, it has proved to have other applications.

The chromatic polynomial is studied extensively in graph theory. Dong et al. [17] provide a summary of much of this research. The chromatic polynomial is also of interest in statistical mechanics, as the Potts model partition function generalises the chromatic polynomial [4]. Potts model partition function was introduced by Potts in [36] and is a function of graphs that has multiple variables. The chromatic polynomial of a graph can be found by assigning certain values to some of these variables.

2.2.2 Expressing and Evaluating the Chromatic Polynomial

Early research into the chromatic polynomial focused on the chromatic polynomials of planar graphs. Whitney [48, 49] was the first to explore the chromatic polynomials of general graphs, using the principle of inclusion-exclusion to express the chromatic polynomial as

$$P(G,\lambda) = \sum_{c,s} (-1)^s N(c,s) \lambda^c$$

where $N(c,s)$ is the number of subgraphs of G with s edges and c components [49]. Read [37] provided another expression for the chromatic polynomial

$$\begin{aligned} P(G,\lambda) &= \sum_{r=1}^{n} \binom{\lambda}{r} r! P_G(r) \\ &= \lambda^{(r)} P_G(r) \end{aligned}$$

where $P_G(r)$ is the number of ways of partitioning the vertex set into exactly r (non-empty) independent sets and $\lambda^{(r)}$ is the *falling factorial* $\lambda(\lambda-1)...(\lambda-r+1)$.

Calculating the chromatic polynomial is #P-hard [45], which is a class of hard counting problems [46]. Even when restricted the family of subgraphs of square lattices it is #P-hard [18]. The following two relations can be used to calculate the chromatic polynomial. By applying them recursively they will eventually give the chromatic polynomial of G, but for most graphs it will take exponential time to compute. The deletion-contraction relation states that for any edge $e \in E$

$$P(G,\lambda) = P(G \backslash e,\lambda) - P(G/e,\lambda).$$

The addition-identification relation states that for any vertices $u,v \in V, uv \notin E$

$$P(G,\lambda) = P(G+uv,\lambda) + P(G/uv,\lambda).$$

Zykov [51] gives another method of evaluating the chromatic polynomial if G is a r-gluing of some graphs H_1 and H_2.

$$P(G,\lambda) = \frac{P(H_1,\lambda)P(H_2,\lambda)}{P(K_r,\lambda)}$$

This method for evaluation can only be applied when G is clique-separable, which does restrict it, but its divide and conquer nature means that under such circumstances it can offer a significant reduction to the difficulty of calculating a graph's chromatic polynomial.

2.2.3 Coefficients of the Chromatic Polynomial

The chromatic polynomial is a *monic* polynomial, meaning that the coefficient of the highest order term is 1. It is also a polynomial of degree n with integer coefficients that alternate in sign and no constant term [37]. Not all polynomials that satisfy these conditions are the chromatic polynomial of some graph. For example, $\lambda^4 - 3\lambda^3 + 3\lambda^2$ satisfies the conditions, but is not the chromatic polynomial of a graph [37].

A sequence $a_0, a_1, \ldots, a_{n-1}$ is *unimodal* if there exists some j such that $a_i \leq a_{i+1}$ whenever $0 \leq i < j$ and $a_i \geq a_{i+1}$ whenever $j \leq i \leq n-2$. Read [37] conjectured that the sequence of absolute values of the coefficients of the chromatic polynomial was unimodal and computational results have shown that this is true for all chromatic polynomials of graphs of order $n \leq 11$ [27]. Some recent work by Huh [22] gives a proof of the unimodality of the coefficients of the chromatic polynomial.

2.2.4 Chromatic Roots

The *roots* of a polynomial $P(\lambda)$ are the values of λ for which the polynomial evaluates to zero. The roots of the chromatic polynomial are often referred to as *chromatic roots*. Research on chromatic roots has focused on three areas: integer roots, real roots and complex roots.

The chromatic polynomial of any graph has integer roots $\{0, 1, \ldots, \chi(G) - 1\}$ and the integer roots of the chromatic polynomial cannot exceed the maximum degree of the graph [15]. A *chordal graph* is a graph that contains no chordless cycle of size > 3. The chromatic polynomials of chordal graphs have only integer roots. However, there exist non-chordal graphs that have the same chromatic polynomial as chordal graphs and so they too have only integer chromatic roots [38].

Real roots are excluded from some very specific intervals on the real number line. There are no non-integer real chromatic roots in the interval $(-\infty, 32/27]$ [23] and no real roots in the interval $[5.664\Delta, \infty)$ where Δ is the maximum degree of the graph [16]. It has been shown that the Beraha numbers, $B_i = 2 + 2\cos(2\pi/i), i \geq 5$, [43] (excluding possibly B_{10}) are not chromatic roots [44].

The complex roots of the chromatic polynomial are dense in the complex plane [41], meaning that, informally, for every point in the complex plane, the point is either in the set of chromatic roots or arbitrarily "close" to a chromatic root. Even when restricted to the chromatic polynomials of planar graphs which have no real roots in the interval $(5, \infty)$, there exists a family of planar graphs with chromatic roots dense in the entire complex plane (possibly excluding the region $|\lambda - 1| < 1$) [41]. There also exist complex chromatic roots with a negative real part [8], even though no negative real number can be a chromatic root.

2.3 Chromatic Equivalence

If two graphs G and G' have the same chromatic polynomial, then they can be said to be *chromatically equivalent*, written $G \sim G'$ [17]. It is interesting to note that this is possible when the graphs are not isomorphic. Two chromatically equivalent graphs may differ in their fundamental structure and yet they still share all of the same information that is encoded in their chromatic polynomial. However, there is currently no way of easily showing that two graphs are in fact chromatically equivalent. Calculating their chromatic polynomials and comparing the results will determine if they are equivalent, but these calculations are intractable in all but a small number of cases.

The general characterisation of chromatically equivalent graphs is unknown, but there is a wealth of research on the subject of chromatic equivalence, much of which is summarised in [13] and [17]. Research in this area focuses on either small sets of graphs that have been found to be chromatically equivalent or infinite families of chromatically equivalent graphs. A result by Morgan [32] uses the certificates discussed later in this review to construct infinitely many pairs of chromatically equivalent graphs where one graph in the pair is clique-separable and the other is not.

The idea of a chromatically unique graphs was introduced by Chao and Whitehead Jr. [12]. A graph G is *chromatically unique* if the only graphs which have the same chromatic polynomial are also isomorphic to G.

2.4 Chromatic Factorisation

The factorisation of a polynomial is the first step to finding its roots. The chromatic polynomial is said to have a *chromatic factorisation* if there exist graphs H_1 and H_2 such that

$$P(G,\lambda) = \frac{P(H_1,\lambda)P(H_2,\lambda)}{P(K_r,\lambda)}$$

where $\chi(H_i) \geq r \geq 0$ and $H_i \neq K_r$ for all i [34]. A graph G is said to have a chromatic factorisation if $P(G,\lambda)$ has a chromatic factorisation. In a chromatic factorisation each of the factors therein is itself the chromatic polynomial of some smaller graph.

Such a factorisation of the chromatic polynomial always exists when G is clique-separable. Most graphs, however, are not clique-separable. If a graph is chromatically equivalent to a clique-separable graph then it is said to be *quasi-clique-separable*, so all graphs of this type will also have a chromatic factorisation. A *strongly non-clique-separable* graph is graph that is not quasi-clique-separable and it is not as immediately clear that such graphs can have chromatic factorisations. Morgan and Farr [34] showed that there do exist chromatic factorisations for the chromatic polynomials of some strongly non-clique-separable graphs.

Certainly, not all strongly non-clique-separable graphs have a chromatic factorisation. For example, a complete graph has no chromatic factorisation. The smallest graph of this type which is not a complete graph is the cycle C_4 [31]. At this stage there is no known general characterisation of strongly non-clique-separable graphs with no chromatic factorisation.

2.5 Certificates

2.5.1 Certificates in Computational Complexity Theory

A *decision problem* is a set of instances D with a subset of instances $Y \subset D$ [20]. Informally, a decision problem poses a question that has only yes and no answers, but generally the yes answers are of more interest. Chromatic equivalence and chromatic factorisation are both examples of decision problems.

Chromatic Equivalence :
INSTANCE: Graphs $G = (V, E), G' = (V', E')$
QUESTION: Does $P(G, \lambda) = P(G', \lambda)$?

Chromatic Factorisation :
INSTANCE: Graph $G = (V, E)$
QUESTION: Does G have a chromatic factorisation?

A decision problem (or *language*) is in the complexity class **NP** if given an input x, we can easily verify that x is an affirmative instance of the problem (or equivalently, x is in the language) if we are given the polynomial-size solution for x that verifies this fact. Such a solution is a called a *certificate*. A certificate, also known as a witness, certifies that the answer to some instance of a decision problem is yes.

A more formal definition of **NP** given by Arora and Barak [3] states that a language $L \subseteq \{0,1\}^*$ is in **NP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time Turing Machine M such that for every $x \in \{0,1\}^*$,

$$x \in L \leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \text{ such that } M(x,u) = 1.$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$ then u is a certificate for x (with respect to the language L and machine M). This definition emphasises that a decision problem is in **NP** if, for all affirmative instances x of the problem, the length of the certificate u must be polynomially bound in the size of the input x so the machine M is able to verify it in polynomial time.

2.5.2 Certificates for the Chromatic Polynomial

Certificates to verify instances of chromatic equivalence and chromatic factorisation were first introduced by Morgan and Farr [34] in 2009. A certificate of this type is a sequence of algebraic transformations based on identities for the chromatic polynomial and algebraic properties. A *certificate of factorisation* explains a factorisation of a graph's chromatic polynomial. A *certificate of equivalence* demonstrates the chromatic equivalence of two graphs. Each of the individual algebraic transformations in a certificate of either kind is the result of performing an operation, called a *certification step*. By starting with a graph G and performing a sequence of these certification steps it may be possible to to express G as factorised expression involving several smaller graphs in a certificate of factorisation, or as some other non-isomorphic graph G' in a certificate of equivalence.

Each application of a certification step adds a single expression to the certificate for G, increasing the length of the certificate. Each expression in a certificate for G can be evaluated to a polynomial, and all these polynomials will be equal to the chromatic polynomial of G. Importantly, the actual chromatic polynomial of G is not calculated or required at any point in the process of creating a certificate for G.

The following is a description of each of the certification steps used to produce a certificate. Each new expression in a certificate is obtained by applying one of the following certification steps to the previous expression in the certificate:

CS1 $G \longrightarrow (G\backslash e) - (G/e)$ for some edge $e \in G$.

CS2 $(G\backslash e) - (G/e) \longrightarrow G$ for some edge $e \in G$.

CS3 $G \longrightarrow (G + uv) + (G/uv)$ where the vertices $u, v \in G$ and u, v are not adjacent.

CS4 $(G + uv) + (G/uv) \longrightarrow G$ where the vertices $u, v \in G$ and u, v are not adjacent.

CS5 $G - (G\backslash e) \longrightarrow (G/e)$ for some edge $e \in G$.

CS6 $G \longrightarrow G_1G_2/K_r$ where G is isomorphic to the graph obtained by an r-gluing of G_1 and G_2.

CS7 $G_1G_2/K_r \longrightarrow G$ where G is isomorphic to the graph obtained by an r-gluing of G_1 and G_2.

CS8 Applying field operations to the terms in an expression a finite number of times to produce a different expression.

Note that certification steps **(CS1)** and **(CS3)** are based upon the deletion-contraction and addition-identification relations seen earlier. Also note that **(CS2)** and **(CS4)** are the inverses of these two steps respectively. Throughout this thesis, we shall refer to these four steps using both the numbering scheme listed above, and the following naming scheme: step **(CS1)** is called *deletion-contraction*, whereas the inverse step **(CS3)** is called *inverse deletion-contraction*; similarly, step **(CS4)** is called *addition-identification*, and the inverse step **(CS4)** is called *inverse addition-identification*.

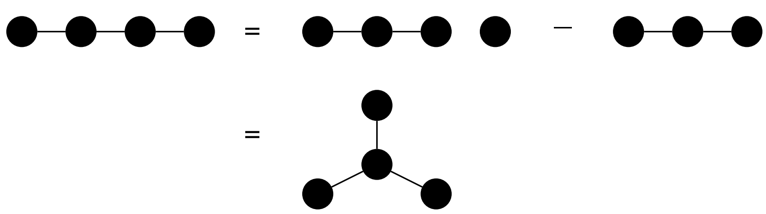


Figure 2.1: A certificate of equivalence of length 2

Figure 2.1 shows a very short example of certificate of equivalence. The certification steps performed in the certificate in Figure 2.1 are as follows:

$$
\begin{aligned}
G &\longrightarrow (G\backslash e) - (G/e) && \textbf{(CS1)} \\
&\longrightarrow (G\backslash e + f) && \textbf{(CS2)}
\end{aligned}
$$

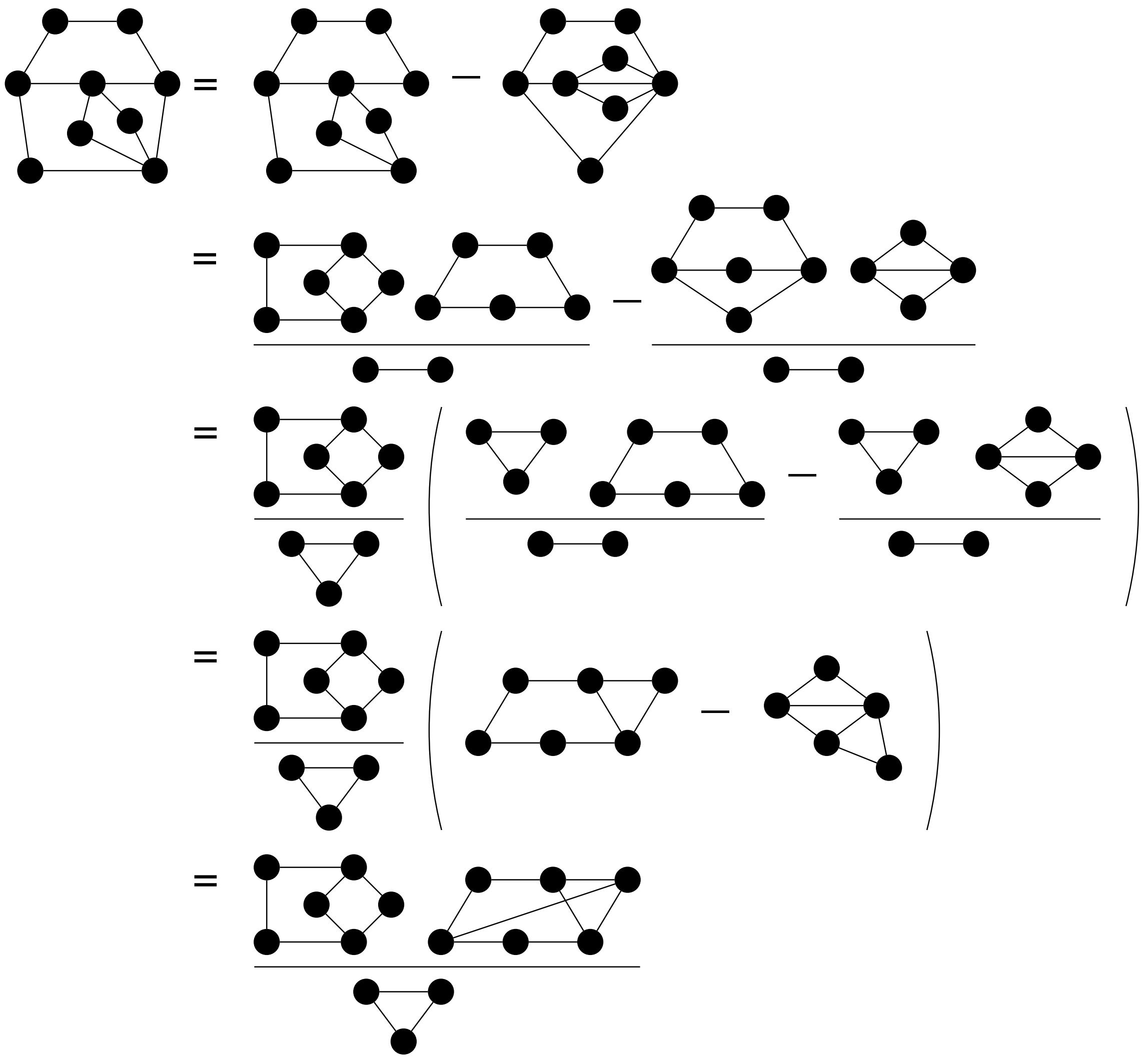


Figure 2.2: A certificate of factorisation for a strongly non-clique-separable graph from [31]

Figure 2.2 gives an example of a certificate of factorisation. The certification steps performed in the certificate in Figure 2.2 are as follows:

$$
\begin{aligned}
G &\longrightarrow H_3 - H_4 && \textbf{(CS1)}\\
&\longrightarrow \frac{H_1H_5}{K_2} - \frac{H_1H_6}{K_2} && 2\times\textbf{(CS6)}\\
&\longrightarrow \frac{H_1}{K_3}\left(\frac{K_3H_5}{K_2} - \frac{K_3H_6}{K_2}\right) && \textbf{(CS8)}\\
&\longrightarrow \frac{H_1}{K_3}(H_7 - H_8) && 2\times\textbf{(CS7)}\\
&\longrightarrow \frac{H_1H_2}{K_3} && \textbf{(CS2)}
\end{aligned}
$$

2.5.3 Finding Certificates

At present, there is no existing software for finding certificates in general, and no known algorithm to find the shortest possible certificate of either factorisation or equivalence. Finding a shortest certificate of either equivalence or factorisation for a graph appears to be a difficult problem, but exactly how difficult is unknown. For a graph with n vertices there exist upper bounds on the lengths of certificates of equivalence (length $< 2^{n^2/2}$) and factorisation (length $\leq n^2 2^{n^2/2}$). These are very long, however in practice the shortest certificates of both equivalence and factorisation that have been found so far are much shorter than these bounds [34].

2.5.4 Schemas

A *schema* is a template for a certificate which represents a set of certificates that all share certain subsequences of steps. A schema can include some actual certification steps as well as gaps, which must be replaced by a sequence of certification steps to form an actual certificate. Certificates which conform to a schema in this manner can be said to belong to the schema. Morgan and Farr found certificates of factorisation that explained the factorisations of all chromatic polynomials of strongly non-clique-separable graphs with at most 9 vertices. They were able to show that vast majority of these certificates belonged to a small number of schemas and that the lengths of these certificates were much shorter than the upper bound given above [34].

3 Certificate Theory

3.1 Fundamental Concepts and Terminology

In this research we view the expressions in certificates as formal mathematical entities in their own right, rather than as just expressions of chromatic polynomials. If we are to generate of certificates of equivalence computationally, we must shift from viewing the expressions therein as polynomials to viewing them as This allows us to systematically store and manipulate them, which is key to the aims of this research. While the possible manipulations of these symbolic expressions arise out of the way in which they are used to express the chromatic polynomial, it is more convenient from a computational perspective to see certificates of equivalence as sequences that begin with a single graph, end with a single graph and transform the former into the latter by means of the certification steps available to us.

To be able to do this more easily, it is convenient to define some terms for discussing the mechanics of creating and manipulating certificates and the expressions they contain. Firstly, if G and G' are chromatically equivalent graphs and T is a certificate for this equivalence that begins with G, then we shall often refer to T as being a *certificate from G to G'*. This puts an emphasis on the fact that T transforms the graph G, as the sole symbol in an expression, into the graph G', as the sole symbol in an expression.

It is helpful to think of the certificates discussed herein as being sequences of lines. A *line* in a certificate is a single expression. The first line in a certificate between two graphs G and G' is just the graph G. Subsequent lines are created by applying a certification step to the preceding line. The final line in such a certificate would be the graph G'. Note that the length of certificate is not the number of lines in the certificate but the number of certification steps applied to transform G into G'.

Also the concept of a *partial certificate*, a sequence of lines which is not yet a complete certificate is important. A sequence of lines which, should some additional lines be added to it, forms a certificate of equivalence for a particular pair of graphs is a *partial certificate*.

3.2 Certificates for Trees

Tree graphs are one of the most important kinds of graph. They find applications in many areas of science, and their structural simplicity has lead to a great deal of research into their mathematical properties. From the perspective of chromatic equivalence alone, the situation is particularly simple: all trees of a given order are chromatically equivalent to each other [17]. However, when it comes to the structure of the certificates that confirm instances of this equivalence, very little is known at all. This section shall perform a theoretical exploration of such certificates by focussing on a particular kind of tree.

We begin our explorations with a well-known fact from the theory of chromatic polynomials–that the smallest (as in smallest order) example of distinct chromatically equivalent graphs is a particular pair of trees. In fact, we have already met these trees in Figure 2.1, which illustrated one of the certificates between the two.

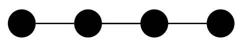


Figure 3.3: The path P_4.

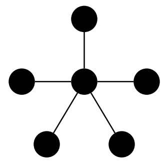


Figure 3.4: The star S_5.

It turns out that these two trees respectively belong to two general types of tree, both of which we shall now define.

Let $k \geq 1$. A *path* is a sequence $v_1 \dots v_k$ of vertices such that each pair of consecutive vertices is joined by an edge. We call the number of vertices k the *length of the path*. Furthermore, there is only one path for such a k, and we shall write it as P_k. Figure 3.3 provides us with an example of such a tree: the path P_4. This is also the path that features in the certificate found in Figure 2.1.

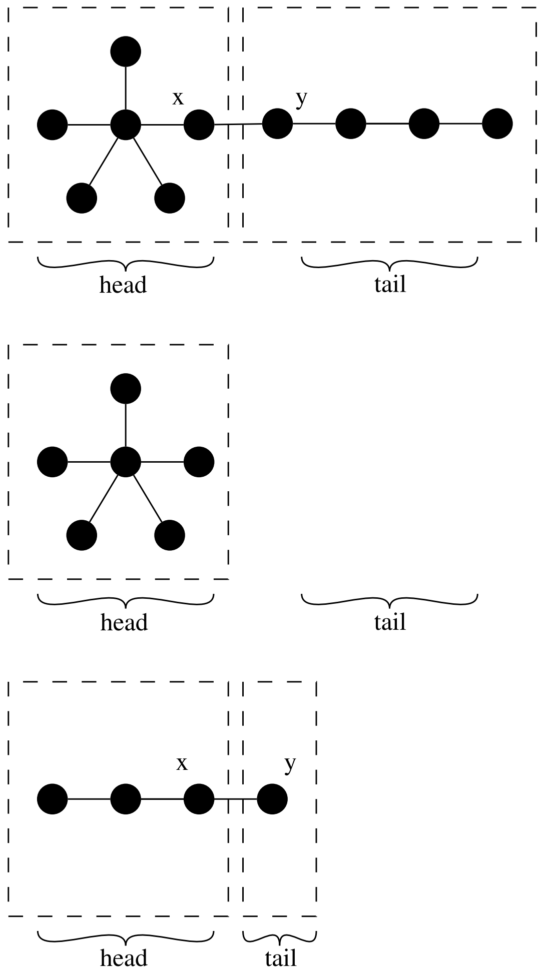


Figure 3.5: The comets $C_{5,4}$, $C_{5,0}$ and $C_{2,1}$

Now let $k \geq 2$. We define a *star* to be a tree comprising k vertices of degree 1 adjacent to and surrounding a single vertex u of degree k. The number k is called the *size* of the star. Like the path P_k, there is only one star for such a k, and we denote it with S_k. We shall refer to the k vertices of degree 1 as the *outer vertices of* S_k, and refer to u as the *central vertex of* S_k. The star S_5 can be seen in Figure 3.4, whereas the star that appears in Figure 2.1 is S_3.

These two types of tree provide us with a starting point for studying certificates between trees in general: not only do they fulfil the fundamental role discussed above, they are particularly structurally simple. Consequently, we shall focus our investigatation on the structure of certificates between stars and paths; however, we shall do this by introducing and working with a more general type of tree that unifies the preceding two.

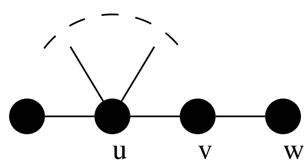
Let h and t be in $\mathbb{N}$ such that $h \geq 2$ and $t \geq 1$. Consider the graph constructed by inserting an edge between an end vertex y of the path P_t and an outer vertex x of the star S_h. We call this graph a *comet* and denote it with $C_{h,t}$. The subgraphs S_h and P_t are respectively called the comet's *head* and *tail*, whereas the numbers h and t are the comet's *head size* and *tail size*. Furthermore, we regard the star graph S_h as a comet with head size h and tail size 0; that is, $C_{h,0} = S_h$. For example, the graphs in Figure 3.5 are the comets $C_{5,4}$, $C_{5,0}$, and $C_{2,1}$. Note, in particular, that $P_4 = C_{2,1}$; it is not hard to extrapolate that $P_n = C_{2,n-3}$ whenever $n \geq 4$. Also note that the order of a comet is $h+t+1$, which implies that the smallest comets are trees of order 4.

Since comets are a type of tree, all comets of a given order are chromatically equivalent to each other. Therefore, all pairs of such comets will have associated certificates. The following result provides some insight into the precise structure of these certificates by showing that each pair of chromatically equivalent comets possesses a certificate of a particular length.

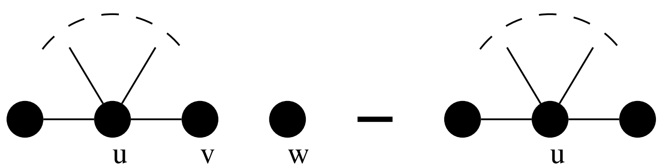
Theorem 3.1. *For each $t \geq 1$, each $h \geq 2$, and each $s \in \mathbb{N}$ such that $1 \leq s \leq t$, there is a certificate of chromatic equivalence of length $2s$ from the comet $C_{h,t}$ to the comet $C_{h+s,t-s}$.*

Proof. We proceed by induction on the tail length t.

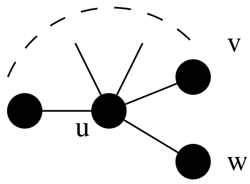
Base case: Let $t = 1$, let $h \geq 2$, and let $1 \leq s \leq t$. We build our certificate as follows. The comet $C_{h,t}$ has the form



and shall constitute our initial line. Perform a deletion-contraction **(CS1)** on the edge vw to obtain the line



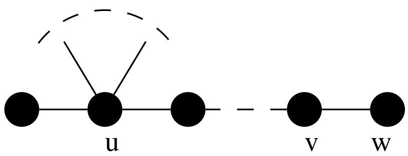
Now we perform an inverse deletion-contraction **(CS2)** on the non-edge uw to obtain the graph



,

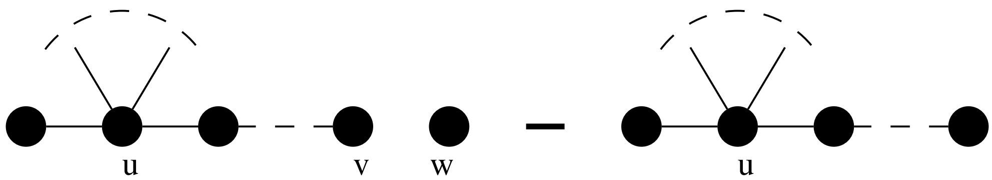
which is the star S_{h+1} and the final line in our sequence. Since $s = 1 = t$ and $S_{h+1} = C_{h+1,0} = C_{h+s,t-s}$, this sequence of lines constitutes a certificate of length $2 = 2s$ from $C_{h,t}$ to $C_{h+s,t-s}$. Consequently, a suitable certificate can be found for each possible selection of h and s, which confirms our base case.

Inductive step: Let $c \in \mathbb{N}$ such that $c \geq 1$. Assume that if $t = c$, then there is a certificate of length $2s$ from $C_{h,t}$ to $C_{h+s,t-s}$ for each h and s satisfying $h \geq 2$ and $1 \leq s \leq t$. Now consider the case of $t = c+1$. Let h and s be in $\mathbb{N}$ such that $h \geq 2$ and $1 \leq s \leq t$. We shall now construct a certificate. The comet $C_{h,t}$ has the form



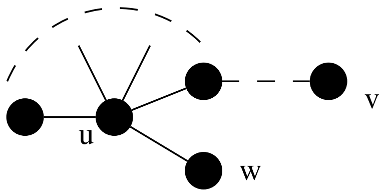
,

since $t = c+1 \geq 2$, and constitutes our initial line. Now perform a deletion-contraction **(CS1)** on the edge vw to obtain the line



.

We then perform an inverse deletion-contraction **(CS2)** on the non-edge uw to obtain the graph



,

which is the comet $C_{h+1,t-1}$. We now have two cases to consider.

If $s = 1$, then $C_{h+1,t-1} = C_{h+s,t-s}$, and so our sequence of three lines constitutes a certificate of length $2 = 2s$ from $C_{h,t}$ to $C_{h+s,t-s}$. Seeing as this is a certificate of the required length, we need not proceed further. Suppose then that $s \neq 1$.

Firstly, $2 \le s \le t$, so $1 \le s-1 \le t-1$; letting $s' = s-1$, it follows that $1 \le s' \le c$. Furthermore, the most recent line in our sequence, $C_{h+1,t-1}$, can be expressed as $C_{h+1,c}$. We can use these two facts to extend our sequence of lines further. Applying our initial assumption about the presence of certificates when $t = c$, we obtain a length $2s'$ certificate T from $C_{h+1,c}$ to $C_{(h+1)+s',c-s'}$. Since

$$(h+1)+s' = (h+1)+(s-1) = h+s$$

and

$$c-s' = c-(s-1) = (c+1)-s = t-s,$$

we can conclude that $C_{(h+1)+s',c-s'} = C_{h+s,t-s}$. Therefore, the certificate T has length $2(s-1)$ and concludes with the graph $C_{h+s,t-s}$. Identifying the first line of T with the final line of our initial sequence of three lines, we obtain a length $2+2(s-1) = 2s$ certificate from $C_{h,t}$ to $C_{h+s,t-s}$; this concludes our second case. Consequently, regardless of what values s and h take, our initial assumption allows us to construct a suitable certificate from $C_{h,t}$ to $C_{h+s,t-s}$.

□

It is not difficult to see that the preceding result can be applied to any pair of distinct chromatically equivalent comets. Let $C_{h,t}$ and $C_{h',t'}$ be two such comets of order $n \ge 4$. Both must have head size at least 2 and tail length at least 0. Furthermore, either comet distributes its n vertices over its head and tail, so we must have

$$h+t+1 = n = h'+t'+1. \tag{1}$$

Since the comets are distinct, (1) implies that they must have different tail lengths; we may assume that $C_{h,t}$ has the longer tail. We can then view the theorem's variable s as specifying the number of vertices that need to be shifted from the tail of $C_{h,t}$ to its head in order to obtain $C_{h',t'}$. Indeed, if we let $s = t-t'$, then we can use (1) to deduce that $s = t-(h+t-h') = h'-h$. Consequently, $h+s = h+(h'-h) = h'$ and $t-s = t-(t-t') = t'$, which implies that $C_{h+s,t-s} = C_{h',t'}$. Furthermore, $1 \le t-t' \le t$, and so the theorem provides a certificate from $C_{h,t}$ to $C_{h',t'}$ of length $2(t-t')$. Note that the length of this certificate is double the difference between the comets' tail lengths. By applying this reasoning to the specific instance of the star and path of order n, we can obtain the following important consequence of the theorem.

Corollary 3.2. *There is a certificate of length* $2(n-3)$ *from the path* P_n *to the star* S_{n-1}*, for each* $n \ge 4$.

Proof. Let $n \ge 4$. As touched upon earlier, $P_n = C_{2,n-3}$ and $S_{n-1} = C_{n-1,0}$. Let $h = 2$, let $t = n-3$, and let $s = n-3$. Since $h \ge 2$ and $1 \le s \le t$, we can apply Theorem 3.1 to h, t, and s to deduce that there is a certificate of length $2s$ from $C_{h,t}$ to $C_{h+s,t-s}$. Noting that $2s = 2(n-3)$, that $C_{h,t} = C_{2,n-3} = P_n$, and that $C_{h+s,t-s} = C_{n-1,0} = S_{n-1}$, we can conclude that our claim is true. □

The preceding corollary provides us with the information that we initially set out to find: insight into the structure of certificates between stars and paths. However, the certificate length that it determines has repercussions for certificates between comets in general. As we recently discussed, Theorem 3.1 provides certificates for comets that have length determined by the difference in comet tail length.

Intuitively speaking, since the path P_n and the star S_{n-1} are respectively the order n comets with the longest and shortest tails, the difference in their tail length provides an upper bound for the difference in tail length of any pair of order n comets. This entails an upper bound for the length of shortest certificates between comets; the following corollary of Theorem 3.1 constitutes an exact specification of this bound.

Corollary 3.3. *Let $n \geq 4$, and let $C_{h,t}$ and $C_{h',t'}$ be distinct comets of order n. If l is the length of the shortest certificates from $C_{h,t}$ to $C_{h',t'}$, then $l \leq 2(n-3)$.*

Proof. Recall our discussion that immediately followed Theorem 3.1. We may assume that $t < t'$; furthermore, we know that the theorem guarantees us a certificate of length $2(t-t')$ from $C_{h,t}$ to $C_{h',t'}$. Letting l be the length of the smallest certificates from $C_{h,t}$ to $C_{h',t'}$, we then know that $l \leq 2(t-t')$. By definition, a comet's head size is at least 2 and its tail size is at least 0, so $h+1 \geq 3$ and $t' \geq 0$. It then follows from (1) that $t = n-(h+1) \leq n-3$, which implies that $t-t' \leq n-3$. We can conclude that $l \leq 2(t-t') \leq 2(n-3)$. □

Our later experimental results shall provide further information about certificates between trees. In particular, we shall explore data that pertains to the two preceding corollaries, which shall then allow us to make some conjectures about how the corollaries might extend to all trees. Before that point, however, we must outline our algorithmic methods and software design, which we do in the next chapter.

4 Algorithms and Software

4.1 Design Decisions

This work is largely concerned with *shortest certificates*, that is, the shortest certificates that can be found with the set of certification steps available to us. The primary goal of the software designed in this project was to create a research tool to aid the process of finding shortest certificates, particularly shortest certificates of equivalence. There is no known fast algorithm for finding a certificate of equivalence for a pair of chromatically equivalent graphs. During the design of the algorithms developed in this research some important decisions were made, which are detailed in this section.

Consider a single line in a partial certificate that starts with a graph G of order n. It is a non–trivial problem to determine exactly which certification steps can or should be performed on this line to produce a new line, let alone one that is part of the shortest certificate possible. How many applications of the certification steps listed in Chapter 2 are even possible? How does one decide where they can be applied within the line? Is there any way of knowing which of the many possible applications of these certification steps is more likely to produce a short certificate for G? While we may not be able to answer these questions entirely, attempting to do so gives us an excellent starting point for designing an algorithm to find shortest certificates.

Firstly, we have no exact or heuristic methods to inform our decisions about the order in which certification steps should be performed. This suggests suggests that an exhaustive search might be appropriate; however, given that graphs are combinatorial objects, any sort of exhaustive approach is likely to only be possible for a small number of graphs before exponential explosion makes computation infeasible. We can then conclude that an exhaustive approach that considers only a small number of graph orders is a suitable point at which to start. However, an exhaustive search algorithm must be able to take a single line of a partial certificate and determine precisely the entire set of certification steps that could be performed on that line. Furthermore, this process must be performed for each line that arises during the exhaustive search.

In some cases, it appears that the process of determining exactly when some types of certification step can be performed appears to be a computationally difficult problem. Consider, for example, the certification step **(CS6)**, also known as clique separation. A linear time algorithm for determining if a graph is clique separable is given by Whitesides in [47]. However, the algorithm does not return all of the possible separating cliques, and there is no known efficient algorithm that does so. If the search is to consider *all* possible steps that could be taken for a single line in a partial certificate, then all of the separating cliques in each of the clique separable graphs would need to be determined for an exhaustive search. So we can conclude that for at least some types of certification step, determining where they may be applied is not always straightforward.

We also need to consider just how many applications of each of the certification steps might be possible at each line in a partial certificate. Let us restrict ourselves for a moment to just two types of certification step: deletion-contraction and addition-identification. For each pair of vertices (u,v) in a graph G of order n, exactly one of these two certification steps may be applied to produce a new line that includes some new graphs. If there is an edge between u and v in G, then deletion-contraction may be applied. If there is no edge between u and v in G, the addition-identification may be applied. So the number of possible steps that can be taken to produce a new expression is

$$\binom{n}{2} = \frac{n(n-1)}{2}.$$

In a line that contains multiple graphs $G_1,\ldots,G_l$ the above is true for each graph G_i of order n_i. So the number of possible steps that can be taken is then

$$\sum_{i=1}^{l} \binom{n_i}{2} = \sum_{i=1}^{l} \frac{n_i(n_i-1)}{2}.$$

However these two steps cannot alone produce a certificate as they only extend the length of a line. Let us consider just how such a line can be obtained during an exhaustive search for certificates. Consider the inverses of the deletion-contraction and addition-identification steps, **(CS2)** and **(CS4)** respectively. We know from Morgan's PhD Thesis [31] that **(CS1)**, **(CS2)**, **(CS3)**, and **(CS4)** are sufficient for providing certificates between chromatically equivalent graphs. Therefore, an exhaustive search can restrict itself to these four steps. We can make the following intuitive deductions about a line obtained from an order n graph in such a search. Firstly, all four steps maintain the maximum graph order of a line when constructing a new one. This means, if we begin our search with an order n graph, that every encountered line must contain a graph of order n. Therefore, if we regard our preceding hypothetical line $G_1,\ldots,G_l$ as having been constructed by such a search–one that began with an order n graph–then we can be certain that one of the terms in the line must be of order $\binom{n}{2}$, and so there are at least $\binom{n}{2}$ lines to be derived from $G_1,\ldots,G_l$.

Applying this reasoning to every line encountered in such a search, we can conclude that if we are searching up to some maximum length d for a certificate, then there are at least

$$\binom{n}{2}^d$$

partial certificates to consider in the search. The actual number of possible partial certificates is greater, as there tend to be other graphs to consider in each line, as well as the possible inverse steps on each line. Nevertheless, this lower bound is not small, being considerably larger than the well-known binary tree exponential of 2^d.

Despite these considerable challenges, the algebraic structure of the resulting lines mentioned above is purely additive–owing to the nature of the four certification steps that are considered. This would make for lines that are not only easy to represent as mathematical structures, but also easy to manipulate computationally. In the following section, we shall explore the possibility of an exhaustive search that takes advantage of these observations.

4.1.1 The Minimal Set of Certification Steps

As we have just seen, when considering the task of searching for shortest certificates, one of the first things that becomes apparent is the inherent computational difficulty involved. There are many possible paths one could take to produce a certificate, and which among those may lead to the shortest one possible is not easy to discern. For these reasons, this research considers a smaller set of the certification steps than the one that is usually dealt with in the literature.

The *minimal set of certification steps* is the set comprising steps **(CS1)**, **(CS2)**, **(CS3)**, and **(CS4)**. In the computational work of this research, we only consider the certification steps in this minimal set, for reasons discussed in the previous section. Morgan and Farr [30,31,32,33] use an extended set of possible certification steps to allow for the creation of certificates of factorisation, which are not considered in this research. This extended set does, under some circumstances, allow for the creation of certificates of equivalence that are shorter than those that only arise from the minimal set. However, since little information is available regarding the general structure of shortest certificates of equivalence, short certificates that use the minimal set are a relatively simple, yet non-trivial, starting point for gaining more knowledge into the matter.

Constraining the set of certification steps to just these few is a natural choice. Deletion-contraction is a fundamental property of the chromatic polynomial, as is the very closely related addition-identification. By choosing to use these two steps, along with their inverses, we are able to make real progress with understanding how certificates can be found for chromatically equivalent graphs, and what the shortest of those certificates are like, while simultaneously making the task of generating them more reasonable computationally.

Importantly, the choice to investigate certificates created using this smaller portion of the various certification steps available does not affect our ability to draw conclusions about their relationship to computational complexity. If, for example, we could show that shortest certificates of this type were always polynomially bounded in length, then, regardless of the excluded certification steps, we would still have shown that chromatic equivalence is in NP. So the lengths of the certificates found using this smaller subset of certification steps retain their relevance to the computational complexity of chromatic equivalence.

4.2 Implemented Algorithms

The `certsearch` software developed to support this research consists of a program written in C, tested under the Linux operating system. It was designed with extensibility in mind, supporting this quality through modularisation. It was also designed to have a small memory footprint, facilitating the possible increases in memory usage required for searches involving increasingly larger orders of graphs. At the time of writing, there is no other existing software is available for searching for shortest certificates of equivalence. The program will now be summarised and the core algorithms presented. We take a close look at the exhaustive search algorithm employed by `certsearch`. Consult Appendix A for more information regarding the use of this software.

4.2.1 The Exhaustive Search Algorithm

In this section we explore the fundamental algorithms that the software is built around. To do this, we need a way to structurally represent the notions of a line and the partial certificates that they constitute.

A *line representation* is a sequence of signed graphs; that is, a sequence

$$(G_1, s_1), \dots, (G_l, s_l)$$

of ordered pairs such that, for each $1 \le i \le l$, G_i is a graph, and s_i is an integer from the set $\{-1, 1\}$. Each such pair represents the position and sign that a graph possesses in a partial certificate line. A partial certificate can then be represented as a sequence of line representations.

We now examine four algorithms that correspond to the four certification steps that our exhaustive search uses to generate lines–those from the minimal set of certification steps. Our first two respectively stem from the deletion-contraction step **(CS1)** and its inverse **(CS2)**. Our first algorithm is very straightforward, owing to the simple nature of the step **(CS1)**.

Algorithm 4.1. Performs a deletion-contraction **(CS1)** at a specified location in a line representation.

Input: A line representation $(G_1, s_1), \dots, (G_l, s_l)$, an index $1 \le j \le l$, and two distinct vertices u and v from G_j. The only requirement is that uv be an edge in G_j.

Output: A new line representation.

1. Perform the edge deletion by letting $D := G_j \setminus uv$. The sign for D is inherited from G_j by letting $d := s_j$.
2. Perform the edge contraction by letting $C := G_j/uv$. The sign for C is the opposite to that of G_j, so let $c := -s_j$.
3. Construct the new line representation by letting

$$\rho := (G_1, s_1), \dots, (G_{j-1}, s_{j-1}), (D, d), (C, c), (G_{j+1}, s_{j+1}), \dots, (G_l, s_l).$$

4. Terminate with the line representation ρ as output.

Our second algorithm is more involved. This is because the step **(CS2)** requires various pre-conditions to be satisfied before it can be used on a line. In particular, these conditions include an isomorphism check. We shall discuss some of the algorithm's subtleties shortly.

Algorithm 4.2. Performs an inverse deletion-contraction **(CS2)** using two specified locations in a line representation.

Input: A line representation $(G_1, s_1), \dots, (G_l, s_l)$, two distinct indices $1 \le j \le l$ and $1 \le k \le l$, and two distinct vertices u and v from G_j. The only requirements are that uv not be an edge in G_j, that $s_j \ne s_k$, and that $G_j/uv \simeq G_k$.

Output: A new line representation.

1. Reverse the deletion using edge addition: let $A := G_j + uv$. The sign for A is inherited from G_j by letting $a := s_j$.

2. Determine the order of the indices by letting $x := \min(j,k)$ and $y := \max(j,k)$.

3. Construct the new line representation by letting

$$\rho := (G_1, s_1), \ldots, (G_{x-1}, s_{x-1}), (A, a), (G_{x+1}, s_{x+1}), \ldots, \\ (G_{y-1}, s_{y-1}), (G_{y+1}, s_{y+1}), \ldots, (G_l, s_l).$$

4. Terminate with the line representation ρ as output.

Both of the these algorithms are based upon the deletion-contraction relation

$$G = G \setminus uv - G/uv.$$

This equation can be viewed as a general rule for how subexpressions of lines are transformed by the steps **(CS1)** and **(CS2)**: the former step replaces an instance of the left-hand side with an instance of the right; the latter does the reverse substitution. Both directions allow for the inversion of graph signs, and both of the preceding algorithms handle this fact by checking and maintaining the relationship between the signs being worked with, rather than dealing with literal values.

The reverse substitution poses some particular challenges, because we need to take into account some of the basic rules of additive algebra: firstly, the two graphs that we use for the reversal may not be adjacent in the line representation; secondly, they may not appear in the order given in the above expression. Both of these problems are dealt with by working with two indices for the line representation, and making no assumptions on the order in which the graphs appear. The position in which the resulting graph is inserted is purely arbitrary; by convention we replace the earlier of the two graphs and remove the subsequent one.

Our next two algorithms are analogous the preceding two, and are based upon the addition-identification step **(CS3)** and its inverse step **(CS4)**. The preceding comments also apply to them in a clear analogous manner; therefore, we shall not follow these two algorithms with a detailed commentary.

Algorithm 4.3. Performs an addition-identification **(CS3)** at a specified location in a line representation.

Input: A line representation $(G_1, s_1), \ldots, (G_l, s_l)$, an index $1 \le j \le l$, and two distinct vertices u and v from G_j. The only requirement is that uv not be an edge in G_j.

Output: A new line representation.

1. Perform the edge addition by letting $A := G_j + uv$. The sign for A is inherited from G_j by letting $a := s_j$.

2. Perform the vertex identification by letting $I := G_j/uv$. The sign for I is also inherited from G_j, so let $i := s_j$.

3. Construct the line representation by letting

$$\rho := (G_1, s_1), \ldots, (G_{j-1}, s_{j-1}), (A, a), (I, i), (G_{j+1}, s_{j+1}), \ldots, (G_l, s_l).$$

4. Terminate with the line representation ρ as output.

Algorithm 4.4. Performs an inverse addition-identification **(CS4)** using two specified locations in a line representation.

Input: A line representation $(G_1,s_1),\ldots,(G_l,s_l)$, two distinct indices $1 \le j \le l$ and $1 \le k \le l$, and two distinct vertices u and v from G_j. The only requirements are that uv be an edge in G_j, that $s_j = s_k$, and that $G_j/uv \simeq G_k$.

Output: A new line representation.

1. Reverse the addition using edge deletion: let $D := G_j \setminus uv$. The sign for D is inherited from G_j by letting $d := s_j$.
2. Determine the order of the indices by letting $x := \min(j,k)$ and $y := \max(j,k)$.
3. Construct the new line representation by letting

$$\rho := (G_1,s_1),\ldots,(G_{x-1},s_{x-1}),(D,d),(G_{x+1},s_{x+1}),\ldots,\\ (G_{y-1},s_{y-1}),(G_{y+1},s_{y+1}),\ldots,(G_l,s_l).$$

4. Terminate with the line representation ρ as output.

All four of the preceding algorithms require that their inputs meet some very specific restrictions. If we are to use them in an exhaustive search, then we must be certain that we can discover all the possible ways in which they may be applied to a given line representation. Only then can we be sure that the certificate produced is in fact of minimal length for the certification steps available. On the other hand, we want to reduce unnecessary work as much as possible, so potential applications of the preceding algorithms that do not meet the necessary requirements must be excluded as quickly as possible. The following recursive algorithm attempts to balance these requirements: it performs an exhaustive search for shortest certificates of a certain kind, and uses some basic heuristics to expediate the process.

Algorithm 4.5. Finds a shortest certificate that begins with a specified line sequence, ends in a specified graph, and does not exceed a specified number of lines.

Input: A sequence ρ of non-empty line representations that constitutes a valid partial certificate, a target graph T, and an integer $M \ge 3$ specifying the maximum number of lines.

Output: A shortest certificate that begins with the sequence ρ, ends with T, and has no more than M lines. If no certificate could be found, then the algorithm returns the empty sequence.

1. Perform the following checks and initialization:

   1.1. Check whether M has been exceeded: if $|\rho| > M$, then return the empty sequence ϵ.

   1.2. Let $\phi := (G_1,s_1),\ldots,(G_l,s_l)$ be the final line representation in ρ. Check whether ρ is a certificate that ends with our target graph: if $l = 1$, $G_1 = T$, and $s_1 = 1$, then terminate with ρ as output.

1.3. Define a best certificate so far, initially the empty sequence: let $\beta := \epsilon$.

2. We shall iterate over the line representation ϕ, so let $j := 1$.

3. If $j > l$, then terminate with β as output. Otherwise let $v_1, \ldots, v_n$ be the vertices of G_j. We shall iterate over this sequence to form vertex pairs. For each x such that $1 \le x \le n$ and each y such that $1 \le y < x$, perform the following:

   3.1. If $v_x v_y$ is an edge of G_j, then perform a deletion-contraction: apply Algorithm 4.1 to ϕ, index j, and vertices v_x and v_y. Otherwise, perform an addition-identification: apply Algorithm 4.3 instead. In either case, let μ be the resulting line representation.

   3.2. Use recursion by applying Algorithm 4.5 to the partial certificate $\rho\mu$, target graph T, and maximum length M. Let γ be the resulting sequence of lines.

   3.3. If $\gamma \neq \epsilon$, then a certificate was found, so update our best certificate and maximum length: let $\beta := \gamma$, and let $M := |\beta| - 1$. Otherwise, no certificate was found. In either case, we move on to the next values of x and y.

4. We shall now iterate over the sequence $(G_1, s_1), \ldots, (G_j, s_j)$, so let $k := 1$.

5. If $k = j$, then let $j := j + 1$ and go to Step 3.

6. If we cannot choose indices a and b from $\{j, k\}$ such that $|V(G_b)| = |V(G_a)| + 1$, then proceed to Step 7. Otherwise, using such a choice, let $v_1, \ldots, v_n$ be the vertices of G_b. For each x such that $1 \le x \le n$ and each y such that $1 \le y < x$, perform the following:

   6.1. If $v_x v_y$ is an edge of G_b, then attempt an inverse addition-identification:

      6.1.1. Check the signs of G_a and G_b: if $s_a \neq s_b$, then go to Step 7.

      6.1.2. Check for isomorphism: if $G_b / v_x v_y \not\cong G_a$, then go to Step 7.

      6.1.3. Perform the operation: apply Algorithm 4.4 to ϕ, b and a (in that order), and v_x and v_y. Let μ be the result.

   6.2. If $v_x v_y$ is not an edge of G_b, then attempt an inverse deletion-contraction:

      6.2.1. Check the signs of G_a and G_b: if $s_a = s_b$, then go to Step 7.

      6.2.2. Check for isomorphism: if $G_b / v_x v_y \not\cong G_a$, then go to Step 7.

      6.2.3. Perform the operation: apply Algorithm 4.2 to ϕ, b and a (in that order), and v_x and v_y. Let μ be the result.

   6.3. Use recursion by applying Algorithm 4.5 to the partial certificate $\rho\mu$, target graph T, and maximum length M. Let γ be the resulting sequence of lines.

   6.4. If $\gamma \neq \epsilon$, then a certificate was found, so update our best certificate and maximum length: let $\beta := \gamma$, and let $M := |\beta| - 1$. Otherwise, no certificate was found. In either case, we move on to the next values of x and y.

7. Let $k := k + 1$ and proceed to Step 5.

Although the body of the preceding algorithm provides a good deal of explanation, a few particular points regarding its heuristics are worth mentioning. Firstly, whenever a new certificate is found in the algorithm, we can be certain that it will be the shortest one yet: the variables β and M are maintained in a way that ensures, once β is a valid certificate, that $M = |\beta| - 1$. Since M is passed to recursive calls of the algorithm, all discovered certificates must be shorter than the shortest currently known. This also means that the number of partial certificates to examine decreases as certificates are found.

The remainder of our heuristics pertain to deciding when pairs of graphs in a line representation are valid candidates for an inverse certification step. Since the algorithm ultimately needs to perform an isomorphism check when processing such graphs–not an easy computational feat–we perform an initial graph order check in Step 6 to discount a potentially large number of possible graph pairings. We also perform the necessary edge and sign checks in steps 6.1, 6.2, 6.1.1, and 6.2.1, before performing the final check for isomorphism in steps 6.1.2 and 6.2.2.

Although Algorithm 4.5 provides a general method for performing exhaustive searches for shortest certificates of a certain kind, we need to use it in a specific way if we wish to find shortest certificates for pairs of graphs. The following algorithm provides a means of initializing Algorithm 4.5 in the way that we desire.

Algorithm 4.6. Finds a shortest certificate between two distinct specified graphs that has no more than a specified number of lines.

Input: Two distinct graphs G and G', and an integer $M \geq 3$ representing the maximum number of lines.

Output: A shortest certificate from G to G' in the form of a sequence of line representations. If no such certificate is found, then the output is the empty sequence ϵ.

1. Construct our initial line representation: let $\rho := (G, 1)$.
2. Apply Algorithm 4.5 to the partial certificate ρ, the target graph G', and the maximum number of lines M. Let γ be the resulting sequence of line representations.
3. Terminate with γ as output.

Algorithm 4.6 is able to take two graphs G and G' and find a shortest certificate of equivalence between them, provided that one exists with at most M lines. Should there be no such certificate, then the algorithm will return the empty sequence ϵ. Note that this failure will certainly happen when G and G' are not chromatically equivalent; bear in mind, however, that it can also happen when they are chromatically equivalent. This means that the algorithm has no general use in deciding chromatic equivalence between graphs. However, the algorithm is used in all of the experiments described in Chapter 5.

5 Experiments

5.1 Experimental Procedure

The `certsearch` program was written in the C programming language and compiled with gcc using the Linux operating system. The exhaustive search algorithm described in Chapter 4 was used for all of the computational experiments.

The experiments were carried out as follows. For each chromatic equivalence class, pairs of graphs were exhaustively formed such that each graph from the class appeared with another precisely once. Each of these pairs were given as an input to the exhaustive search algorithm, which then found a shortest certificate for the pair and wrote this certificate out to the file of results for the corresponding graph order. This procedure was performed for graph orders 4, 5, 6, and 7. Table 5.1 provides details about the computer on which all of the experiments were performed.

Computer:	Lenovo ThinkPad X1
Processor:	Intel(R) Core(TM) i5–2520M CPU @ 2.50GHz
Speed:	800.00 MHz
Memory (RAM):	3.8 GB
Operating System:	Linux openSUSE 12.2

Table 5.1: Details of the Computer Used for Computational Experiments

The search algorithm requires lists of the chromatic equivalence classes for graphs of order $4 \leq n \leq 7$. This data was provided by Kerri Morgan. These lists themselves contain lists of the graphs, indexed by certain integers and arranged by equivalence class. The indices correspond to graph data provided by Brendan McKay, which is made available at [30]. The program also uses `nauty` [29], also developed by McKay [28] to perform isomorphism checking during the running of the search algorithm.

5.2 Discussion of Results

The experiments that we described in the preceding section examined a total of 3821 pairs of chromatically equivalent graphs from 157 equivalence classes, and obtained a shortest certificate for each of these pairs. In this section we discuss the lengths of the certificates that were found. We also consider the schemas to which some of these certificates belong. We conclude the section by examining the portion of results that pertains to trees.

5.2.1 Certificate Length

One of the main reasons for conducting the experiments was to find information about the length of shortest certificates of chromatic equivalence. Table 5.2 lists the certificate length data from the experiments. For each graph order, it lists the number of shortest certificates found of each length. Although the experiments considered only a small number of small graph orders, the certificate lengths that they found are, relative to corresponding graph order, very short. In particular,

Graph Order	Length 2	Length 4	Length 6	Length 8
4	1			
5	8	1		
6	113	48	2	
7	1610	1759	272	7

Table 5.2: The lengths of shortest certificates found for chromatically equivalent pairs of graphs of order ≤ 7

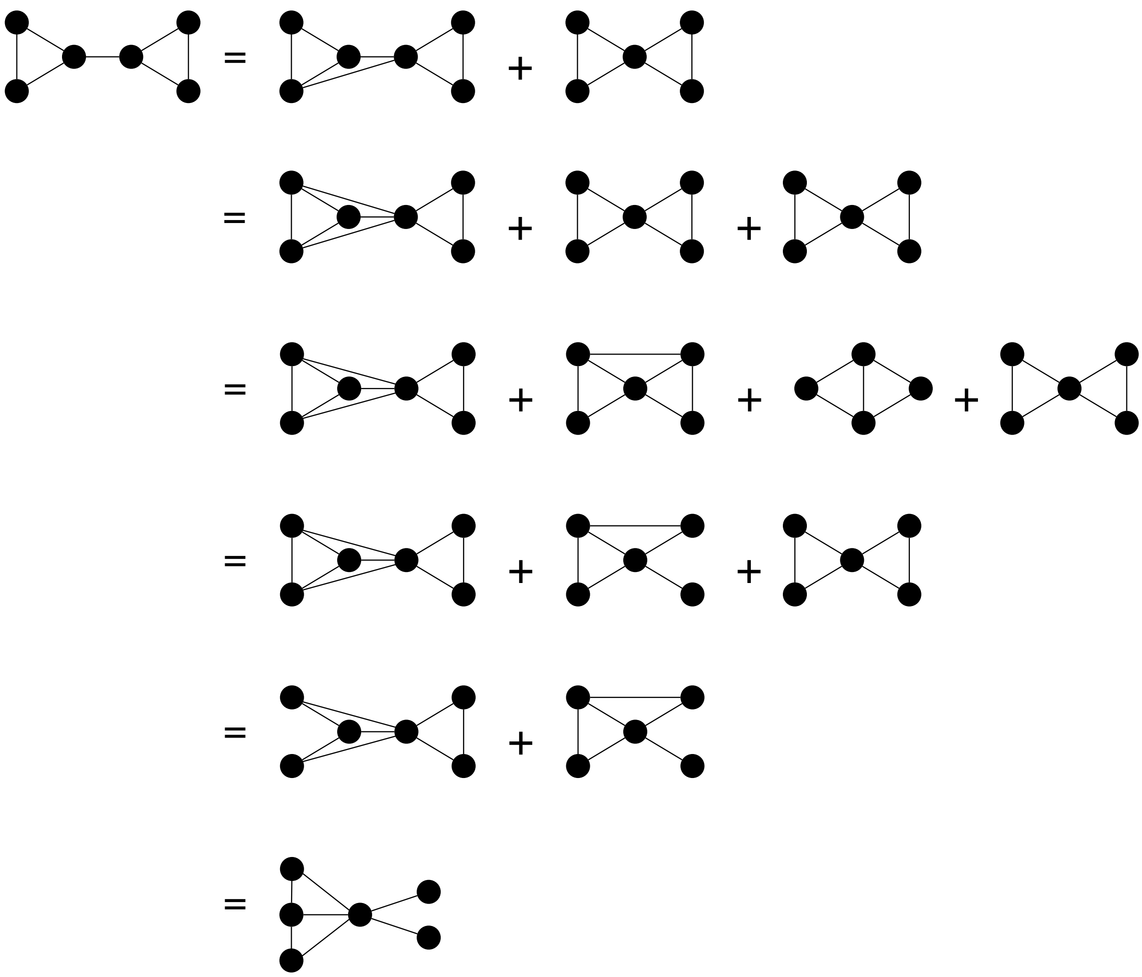


Figure 5.6: A certificate of equivalence for two graphs of order 6, belonging to Schema 14

only seven of the certificates have length greater than the order of their associated graphs, and these certificates are of length 8. In general, certificates that are shorter relative to graph order are more numerous. While these experiments only consider graphs of very small order, it is encouraging that so far the shortest certificates produced have been very short indeed, especially since the best known upper bound on the length of certificates is $< 2^{n^2/2}$, which is exponential in the order of the pair of graphs.

The certificate in Figure 5.6 is an example of one of the many certificates found using `certsearch`. All certificates found during the experiments for orders $4 \le n \le 6$ can be found in Appendix B. These certificates, as well as those for the order 7 graphs, can also be found on the accompanying CD.

5.2.2 Schemas

Recall from Chapter 2 that a schema provides a template for a certificate which represents a set of certificates that all share common subsequences of steps. A schema may include some actual certification steps, or gaps that need to be replaced by a sequence of certification steps to form an actual certificate. A certificate which follows the pattern of certification steps given in a schema is said to *belong* to the schema In this section we will only be considering schemas which have *all* of the certification steps required to form a certificate.

Appendix B lists the schemas to which each certificate found via the computational experiments belongs, for all certificates for graphs of order $4 \le n \le 6$. These schemas were obtained by analysing the certificate data produced from the experiments. Lacking a computational method, it was not feasible to also find the schemas for certificates for graphs of order 7, as they number in the thousands.

Schema	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14
Length	2	2	4	4	6	4	4	4	4	4	4	4	4	6

Table 5.3: The length of each schema.

Order	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14
4	1													
5	4	4	1											
6	62	51	4	4	1	10	2	2	9	5	6	1	3	1

Table 5.4: The distribution of encountered shortest certificates amongst the schemas, for each graph order considered.

Table 5.3 details the lengths of these schemas. For each such schema, Table 5.4 provides the number of discovered shortest certificates that belong to it. Schemas given in Appendix B are not all of the *possible* schemas for certificates up to length 6, they are only those to whom at least one certificate in the results belongs. There exist many other possible schemas of the same lengths.

The exhaustive algorithm finds just one of potentially many shortest certificates for each input graph pair. The set of schemas to which the resulting certificates belong to are in part artefacts of how the graphs are stored, as the labelling of edges affects the order of edge selection during the running of the algorithm. The order in which possible certification steps are attempted will also affect the schema to which a shortest certificate conforms. Consequently, the schemas to which the certificates from our results belong are not necessarily the only ones for which certificates of the same length for each pair could be produced, although they are the shortest. There may exist other certificates of the same length for a given pair of graphs that belong to some other schema; either one of the others listed in Appendix B, or another schema altogether.

Nevertheless, we are still able to draw some important conclusions from the information we do have. Since shortest certificates that were found conform to a small set of only 14 schemas, and there certainly exist other possible schemas of these lengths, we can say that the entire set of possible schemas may not need to be considered when searching for shortest certificates.

The vast majority of the certificates found conform to Schemas 1 and 2. This is not unexpected, as these two schemas describe the only two sequences of certification steps that can produce a certificate of length 2. Schemas 5 and 14 both describe length 6 certificates, and the remainder describe all of those found with length 4.

Two schemas differ somewhat to the others that were found. Schemas 13 and 14 are a little unusual. If one considers the final line of any of the schemas, the final graph in the schema is given in terms of the first graph in the schema, with some set of edges removed, and some other set added. We will call this sum the for the pair of graphs the their *edge difference*. For example, Schema 3 has the final graph $(G+e\backslash f+g\backslash h)$, so the edge difference of a pair of graphs that conform to this schema is 4. In the first 12 schemas, the edge difference is always the length of the schema.

However, in Schemas 13 and 14, the edge difference for them both is two less than their respective lengths. This suggests that the process of transforming the pairs of graphs whose certificates belong to these schemas is perhaps a slightly more complicated task than those that belong to the other schemas. It is possible that the certification steps in the extended set, which were not used in this research, may be able to produce certificates for these pairs of graphs that have a length less than or equal to their edge difference.

5.2.3 Certificates for Trees Revisited

As we saw in Chapter 3, all trees of a given order are chromatically equivalent, and some interesting theoretical results about the certificates between them can be found. We shall now examine how our experimental results complement the theoretical results that we obtained in that chapter.

Tree Order	Length 2	Length 4	Length 6	Length 8
4	1			
5	2	1		
6	9	5	1	
7	27	20	7	1

Table 5.5: The lengths of shortest certificates found for chromatically equivalent trees of order ≤ 7

Firstly, for each $4 \leq n \leq 7$, the star S_{n-1} and path P_n of order n have the longest certificate of equivalence amongst the shortest certificates for pairs of trees. This also happens to be the case for the corresponding shortest certificates between pairs of graphs, for the graph orders given; however, it seems less likely that this may be a trend that continues for higher graph orders. Table 5.5 lists the certificate length data from the experiments for pairs of trees.

Interestingly, there is exactly one certificate of precisely length $2(n-3)$ for each order n in the results. Each of these certificates is, in fact, a certificate between a star and a path. The edge difference, as described in the previous section regarding schemas, for star and path pairs appears to be $2(n-3)$. This is also the upper bound on the length of shortest certificates between comets that Corollary 3.3 established. In a sense, the star and path appear to provide a worst case scenario for the lengths of certificates of equivalence for pairs of trees: it appears that the edge difference of such pairs may be smaller, or at least no worse, than it is for the star and path. This corollary, together with our computational results, suggests the following conjecture.

Conjecture 5.1. *For each pair of trees of order n, there exists a certificate of equivalence for that pair that has length no greater than* $2(n-3)$.

If this conjecture were shown to be true, then a linear bound for shortest certificate length on an even larger infinite subset of chromatically equivalent graphs would be confirmed. Our intention in this work has been to explore just how short shortest certificates can get, if only for a small set of graph orders. Since certificates find their origins in computational complexity, their length is a crucial thing to consider, particularly upper bounds on their length. Having such a bound for the length of shortest certificates between trees would be significant.

6 Conclusion

In this research a new method for finding certificates of equivalence was designed and implemented and some new research tools for finding certificates were introduced. The decisions made during the course of this research have allowed for the design of a software tool which uses a minimal set of certification steps in order to make the exhaustive search feasible while producing interesting new data about the general form of certificates of equivalence. The certificates found are all quite short, relative to the order of their graph pairs. They also conform to only a small number of schemas.

This research has also proven a new bound of $2(n-3)$ on the lengths of certificates of equivalence for comet graphs. This class of graph includes the star and path graphs, which were subsequently shown for orders $4 \leq n \leq 7$ to have the longest certificates amongst all trees of the same order in the experimental results. This theoretical result, in conjunction with the certificates produced by the exhaustive search algorithm, lead us to conjecture that this bound applies to all shortest certificates for trees.

It is clear that there are many things still unknown about the properties of the chromatic polynomial, as well many things still unknown about certificates for these properties. Finding short certificates is of particular interest due to their implications for the computational complexity of chromatic equivalence and chromatic factorisation. The certificates employed in this research are quite a recent development and there remain many questions about the types of certificates that can be found, as well as their computational complexity implications. In general, the certificates that have been found so far are significantly shorter than the upper bounds on their length known at this time, so it is possible that further research could uncover tighter upper bounds.

Although the chromatic polynomial has been investigated in considerable depth, there has been little research into its algebraic theory [31]. Chromatic equivalence has been the topic of much research, but knowledge about the characterisation of chromatically equivalent graphs in general is far from complete. The certificates of equivalence that have been found so far provide some tantalising hints as to how they may behave generally, but there remain a great number of things about them that are unknown. Consequently, there is a wealth of potential directions for further research into certificates for properties of the chromatic polynomial.

6.1 Achievements

An algorithm was designed and implemented to aid research into the lengths and general structure of certificates of equivalence. This algorithm allows for the automated generation of shortest certificates. The exhaustive search algorithm was used to find the shortest certificates for all 3821 pairs of chromatically equivalent graphs of order ≤ 7 These certificates are all short relative to the orders of the graphs for which they certify equivalence. The longest found were at most length $n+1$ for graphs of order n.

The schemas for all of the certificates for pairs of graphs of order $<= 6$ were also found. The vast majority of the certificates found conform to Schemas 1 and 2, which describe certificates of length 2. Each of the 163 certificates found for graphs of order $n \leq 6$ belongs to one of only 14 schemas. This suggests that perhaps not all possible schemas need be considered when searching for certificates of equivalence, as some are substantially more numerous and others are apparently not necessary.

The research also uncovered a new linear upper bound of $2(n-3)$ on the length of shortest certificates for an infinite family of tree graphs. This family of graphs, comets, includes the star and path graphs, both of which seem to play an important role in the theory of chromatic equivalence.

Computational results suggest an upper bound on the lengths of shortest certificates of equivalence for trees. When viewed in light of the new bound on the length of shortest certificates between comet graphs, this data suggests that the comet graph bound may extend to all trees. In fact, we conjecture that between each pair of distinct trees of order n there is a certificate of equivalence of length no greater than $2(n-3)$.

6.2 Further Work

There are a number of ways in which the work in this project could be extended. Some of these avenues, and their potential challenges, are outlined in this section.

It is possible to conduct a search for certificates on graphs with order $n > 7$ using the software tools developed in this research. However, it is likely that such an undertaking would require the use of a high performance computing environment, due the incredibly large number of partial certificates that are created during the search. With increasingly large orders of graphs comes both a larger number of ways to apply certification steps to lines, as well as a larger number of chromatic equivalence classes.

The schemas found in this research could be used to inform new heuristics for generating certificates. By attempting to find certificates between pairs of graphs using the schemas produced during this research as templates, it may be possible to find certificates for larger orders of graph. Some schemas seem to be more common than others, so attempting to find certificates that conform to them is a possible avenue for improving the time taken to find certificates.

This research developed a means of finding a shortest certificate for a given pair of graphs. However, there may be other certificates for such a pair that have the same length, but conform to schemas that differ to those found. A search for *all* of the certificates of shortest length for a pair of chromatically equivalent graphs could be devised. This could give a more complete understanding of just what the full range of possible schemas for shortest certificates is. This information could then be used to inform a heuristic search.

The search algorithm could be expanded to include the extended set of certification steps studied by Morgan and Farr [34]. Were this to be done, it is quite possible that shorter certificates of equivalence could be found for some of the certificates found in this project. It is also possible that such a method would find shorter certificates, in general.

Certificates of factorisation use the same certification steps as certificates of equivalence, with an extended set of steps to allow for factorisation. Extending the search capabilities of our algorithms to include searching for certificates of factorisation is an avenue for further work. There are some issues that such an extension would need to address, such as the implementation of the extended set of certification steps, as well as major changes to the certificate data structures.

7 References

[1] K. Appel and W. Haken, Every planar map is four colorable. I. Discharging, *Illinois J. Math.* 21:429–490, 1977.

[2] K. Appel, W. Haken and J. Koch, Every planar map is four colorable. II. Reducibility, *Illinois J. Math.* 21:491–567, 1977.

[3] S. Arora, B. Barak, *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[4] L. Beaudin, J. Ellis-Monaghan, G. Pangborn, and R. Shrock, *A little statistical mechanics for the graph theorist.* 2008. Available from `http://arxiv.org/abs/0804.2468`

[5] N.L. Biggs, E.K. Lloyd, R.J. Wilson, *Graph Theory 1736-1936.* Oxford University Press, Oxford, 1986.

[6] G.D. Birkhoff, A determinant formula for the number of ways of colouring a map. *Ann. of Math. (2)* 14:42–46, 1912–1913.

[7] V.A. Blatov, Search for isotypism in crystal structures by means of the graph theory, *Acta Crystallogr A.* 56:178–88, 2000.

[8] J.I. Brown, C.A. Hickman, On chromatic roots with negative real part, *Ars Combin.* 63:211–221, 2002.

[9] A. Caley, On the mathematical theory of isomers, *Philos Mag*. 67:444–4446, 1857.

[10] G. Chaitin, M. Auslander, A. Chandra, J. Cocke, M. Hopkins, P. Markstein, Register allocation via coloring, *Comp. Lang.* 6:47–57, 1981.

[11] C. Chao, E.G. Whitehead Jr, Chromaticity of self-complementary graphs, *Arch. Math (Basel).* 32:295–340, 1979.

[12] C. Chao, E.G. Whitehead Jr, Chromatically unique graphs, *Discrete Math.* 27:171–177, 1979.

[13] G.L. Chia, A bibliography on chromatic polynomials, *Discrete Math.* 172:175–191, 1997.

[14] D. de Werra, Restricted coloring models for timetabling, *Discrete Math.* 165/166:161–170, 1997.

[15] R. Diestel, *Graph Theory.* Springer-Verlag, New York, 2000.

[16] F.M. Dong, K.M. Koh, Bounds for the real zeros of chromatic polynomials, *Combin. Probab. Comput.* 17:749–759, 2008.

[17] F.M. Dong, K. M Koh, K. L Teo, *Chromatic Polynomials and Chromaticity of Graphs.* World Scientific, Singapore, 2005.

[18] G. Farr, The complexity of counting colourings of subgraphs of the grid, *Combin. Probab. Comput.* 15:377–383, 2006.

[19] L.R. Foulds, *Graph Theory Applications.* Springer-Verlag, New York, 1992

[20] M.R. Garey, D.S. Johnson, *Computers and Intractability.* W.H. Freeman and Company, New York, 1979.

[21] G. Gonthier, Formal proof–The four-color theorem. *Notices of the American Mathematical Society* 11:1382–1393, 2008

[22] J. Huh, Milnor numbers of projective hypersurfaces and the chromatic polynomial of graphs, Preprint, 2011.

[23] B. Jackson, A zero-free interval for chromatic polynomials of graphs, *Combin. Probab. Comput.* 2:325–336, 1993.

[24] T. Jensen, B. Toft. *Graph Colouring Problems.* John Wiley and Sons, New York, 1995.

[25] R.M. Karp, Reducibility among combinatorial problems. In R.E. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum, Boston, 1972.

[26] A.B. Kempe, On the geographical problem of the four colors, *Amer. J. Math.* 2:193–200, 1879.

[27] P.H. Lundow, K. Markström, Broken-cycle-free subgraphs and the log-concavity conjecture for chromatic polynomials, *Experiment. Math.* 15:343–353, 2006.

[28] B. McKay, Practical Graph Isomorphism. *Congr. Numer.* 30:45–87, 1981.

[29] B. McKay, The Nauty Page. `http://cs.anu.edu.au/~bdm/nauty/`.

[30] B. McKay, Graph Data. `http://cs.anu.edu.au/~bdm/data/graphs.html`.

[31] K. Morgan, *Algebraic Aspects of the Chromatic Polynomial*. PhD Thesis, Monash University, Clayton School of Information Technology, 2010.

[32] K. Morgan, Pairs of chromatically equivalent graphs, *Graph Combinator.* 27:547–556, 2011.

[33] K. Morgan, G. Farr, Certificates of factorisation for a class of triangle-free graphs, *Electron. J. Combin.* 16:#R75, 2009.

[34] K. Morgan, G. Farr, Certificates of factorisation for chromatic polynomials *Electron. J. Combin.* 16:#R74, 2009.

[35] D. Penny, L.R. Foulds, M.D Hendy, Testing the theory of evolution by comparing phylogenetic trees constructed from five different protein sequences, *Nature*. 297:197–200, 1982.

[36] R.B. Potts, Some generalized order-disorder transformations, *PCPS-P. Camb. Philol. S.* 48:106–109, 1952.

[37] R.C. Read, An introduction to chromatic polynomials, *J. Combin. Theory.* 4:52–71, 1968.

[38] R.C. Read, Review, *Mathematical Reviews.* 50: Review 6906, 1975.

[39] R.C. Read, W.T. Tutte, Chromatic polynomials. In L.W. Beineke and R.J. Wilson, editors, *Selected Topics in Graph Theory*, volume 3, pages 15–42. Academic Press, London, 1988.

[40] N. Robertson, D. Sanders, P. Seymour, and R. Thomas, The four-colour theorem. *J. Combin. Theory Ser. B* 70:2–44, 1997.

[41] A.D. Sokal, Chromatic roots are dense in the whole complex plane, *Combin. Probab. Comput.* 13:221–261, 2004.

[42] P.G. Tait, Note on a theorem in geometry of position, *Trans. Roy. Soc. Edinburgh.* 29:657–660, 1880.

[43] W.T. Tutte, Chromials. In C. Berge and D Ray-Chaudhuri, editors, Hypergraph Seminar, volume 411 of Lecture Notes in Mathematics, pages 243–266. Springer-Verlag, Berlin, 1972.

[44] W.T. Tutte, Chromatic sums for rooted planar triangulations: The cases $\lambda = 1$ and $\lambda = 1$, *Canad. J. Math.* 25:426–447, 1974.

[45] L.G. Valiant, The complexity of computing the permanent, *Theoret. Comput. Sci.* 8:189–201, 1979.

[46] L.G. Valiant, The complexity of enumeration and reliability problems, *SIAM J. Comput.* 8:410–421, 1979.

[47] S.H. Whitesides, An algorithm for finding clique cut-sets. *Inform. Process. Lett.* 12:31–32, 1981.

[48] H. Whitney, The colouring of graphs. *Ann. of Math. (2nd Series).* 33:688–718, 1932.

[49] H. Whitney, A logical expansion in mathematics. *Bull. Amer. Math. Soc.* 38:572–579, 1932.

[50] R.J. Wilson, *Four Colours Suffice: How the Map Problem was Solved.* Allen Lane, London 2002.

[51] A.A. Zykov, On some properties of linear complexes. *Amer. Math. Soc. Transl.* 79, 1952. Translated from original article in Math. Sbornik, 24:163–188, 1949.

8 Appendix A Software Information

This appendix provides some information about the `certsearch` software produced in this project. For more detailed information about the finer points of the program implementation, we refer the reader to the source files provided in the accompanying CD.

8.1 Environment

As was discussed in Chapter 4, all of the experimental runs using `certsearch` were completed on a computer with the following specifications.

Computer:	Lenovo ThinkPad X1
Processor:	Intel(R) Core(TM) i5–2520M CPU @ 2.50GHz
Speed:	800.00 MHz
Memory (RAM):	3.8 GB
Operating System:	Linux openSUSE 12.2

8.2 Building the Software

The `certsearch` software was written in the C programming language and was compiled with gcc. A makefile is included with the source code on the accompanying CD.

The program `certsearch` uses `nauty`, developed by Brendan McKay and available at [29]. In the software, `nauty` is used for the graph isomorphism checks performed during the search. The program also uses a function from some work by Kerri Morgan [31], which is used as an interface to `nauty`. This function was modified during this research to make it compatible with the graph data structures used by `certsearch`. The source code files for `nauty` and the modified code from Morgan are included along with the other source files required to build the `certsearch` program.

Also provided are the `n_polys` files, which contain lists of all of the chromatic equivalence classes for all non chromatically unique graphs of order 4, 5, 6 and 7. These files were provided by Kerri Morgan. The `graphs*` files are also included. They give the adjacency matrices of all graphs of orders 4, 5, 6 and 7. These files are provided by Brendan McKay and are made available at [30]. Both the `n_polys` and the `graphs*` files are required by the automated exhaustive search functions.

8.3 Using the Software

When running `certsearch` the user is presented with a number of options. Most important amongst these is option 2 which runs the batch experiments for all of the pairs of graphs of order $4 \le n \le 7$ and was used to find all of the computational results in this thesis. This search option will likely take many hours to complete on the average computer, but the certificates for graphs of order $4 \le n \le 6$ will complete in a number of minutes. The certificates found during this search option are written out to the `order_*_certificates` files in the `graphs` directory.

8.4 Interpreting Output Certificates

This section contains information about interpreting the data output to file by `certsearch`. Please note that in the certificates in the `order_*_certificates` files and Appendix C, the graphs G and $G0$ are in fact the same graph.

In the software, order n graphs are always defined over the set of vertices $\{0,\ldots,n-1\}$. The only graph operations that the software performs are edge deletion, edge addition, and vertex identification (recall that this is also called contraction when the vertices involved are adjacent). Out of these three, identification is only one that alters the order of the resulting graph, and thus the vertex set being worked with. Furthermore this change has an effect on the edges of the graph; it is important to have some knowledge of how this change occurs, if the output of the software is to be interpreted correctly.

When we perform a identification on u and v, because of the way the graphs are represented, one of these vertices will have a smaller vertex label than the other, so let the smaller of the two be u. We proceed in the identification by essentially absorbing v into u, and then removing v. When v is removed from the graph, all of the vertices adjacent to v, excluding u, become adjacent to u if they are not already adjacent to u. Now, for each w from $v+1$ up to n, we do the following: replace w with $w-1$ such that every edge adjacent to w becomes adjacent to $w-1$.

In short, v is removed, u inherits the adjacencies of v, and the vertices and edges of the resulting graph are transformed according to the map

$$\phi(w)=\begin{cases} w & \text{if } w<v \\ w-1 & \text{if } w>v. \end{cases}$$

With this information, together with the adjacency matrices in the `graphs*` files, it is possible to interpret the certificates in Appendix C. Note that the certificates listed in the `order_*_certificates` files, which can be found on the accompanying CD, are preceded by the edge lists of the graphs involved in the certificate. The first of the two graphs listed is G in the certificate. The second is the graph found in the final line of the certificate. The certificates in the `order_*_certificates` files are also interpreted in the manner described above.

9 Appendix B Schemas

This appendix contains a number of schemas. All of the certificates found for pairs of graphs of order $4 \leq n \leq 6$ during the experimental runs of the exhaustive search algorithm belong to one of the following schemas.

Schema 1:

$$\begin{aligned} G &= (G+e) \;+\; (G/e) && \textbf{(CS3)} \\ &= (G+e\backslash f) && \textbf{(CS4)} \end{aligned}$$

Schema 2:

$$\begin{aligned} G &= (G\backslash e) \;-\; (G/e) && \textbf{(CS1)} \\ &= (G\backslash e+f) && \textbf{(CS2)} \end{aligned}$$

Schema 3:

$$\begin{aligned} G &= (G+e) \;+\; (G/e) && \textbf{(CS3)} \\ &= (G+e\backslash f) \;-\; (G+e/f) \;+\; (G/e) && \textbf{(CS1)} \\ &= (G+e\backslash f+g) \;+\; (G/e) && \textbf{(CS2)} \\ &= (G+e\backslash f+g\backslash h) && \textbf{(CS4)} \end{aligned}$$

Schema 4:

$$\begin{aligned} G &= (G\backslash e) \;-\; (G/e) && \textbf{(CS1)} \\ &= (G\backslash e+f) && \textbf{(CS2)} \\ &= (G\backslash e+f\backslash g) \;-\; (G\backslash e+f/g) && \textbf{(CS1)} \\ &= (G\backslash e+f\backslash g+h) && \textbf{(CS2)} \end{aligned}$$

Schema 5:

$$
\begin{aligned}
G &= (G+e) \;+\; (G/e) && \textbf{(CS3)}\\
&= (G+e+f) \;+\; (G+e/f) \;+\; (G/e) && \textbf{(CS3)}\\
&= (G+e+f\backslash g) \;+\; (G/e) && \textbf{(CS4)}\\
&= (G+e+f\backslash g\backslash h) && \textbf{(CS4)}\\
&= (G+e+f\backslash g\backslash h+i) \;+\; (G+e+f\backslash g\backslash h/i) && \textbf{(CS3)}\\
&= (G+e+f\backslash g\backslash h+i\backslash j) && \textbf{(CS4)}
\end{aligned}
$$

Schema 6:

$$
\begin{aligned}
G &= (G+e) \;+\; (G/e) && \textbf{(CS3)}\\
&= (G+e+f) \;+\; (G+e/f) \;+\; (G/e) && \textbf{(CS3)}\\
&= (G+e+f\backslash g) \;+\; (G/e) && \textbf{(CS4)}\\
&= (G+e+f\backslash g\backslash h) && \textbf{(CS4)}
\end{aligned}
$$

Schema 7:

$$
\begin{aligned}
G &= (G+e) \;+\; (G/e) && \textbf{(CS3)}\\
&= (G+e+f) \;+\; (G+e/f) \;+\; (G/e) && \textbf{(CS3)}\\
&= (G+e+f\backslash h) \;+\; (G+e/f) && \textbf{(CS4)}\\
&= (G+e+f\backslash g\backslash h) && \textbf{(CS4)}
\end{aligned}
$$

Schema 8:

$$
\begin{aligned}
G &= (G\backslash e) \;-\; (G/e) && \textbf{(CS1)}\\
&= (G\backslash e+f) && \textbf{(CS2)}\\
&= (G\backslash e+f+g) \;+\; (G\backslash e+f/g) && \textbf{(CS3)}\\
&= (G\backslash e+f+g\backslash h) && \textbf{(CS4)}
\end{aligned}
$$

Schema 9:

$$\begin{aligned} G &= (G+e) \quad + \quad (G/e) && \textbf{(CS3)} \\ &= (G+e\backslash f) && \textbf{(CS4)} \\ &= (G+e\backslash f+g) \quad + \quad (G+e\backslash f/g) && \textbf{(CS3)} \\ &= (G+e\backslash f+g\backslash h) && \textbf{(CS4)} \end{aligned}$$

Schema 10:

$$\begin{aligned} G &= (G+e) \quad + \quad (G/e) && \textbf{(CS3)} \\ &= (G+e\backslash f) \quad - \quad (G+e/f) \quad + \quad (G/e) && \textbf{(CS1)} \\ &= (G+e\backslash f\backslash g) \quad - \quad (G+e/f) && \textbf{(CS4)} \\ &= (G+e\backslash f\backslash g+h) && \textbf{(CS2)} \end{aligned}$$

Schema 11:

$$\begin{aligned} G &= (G+e) \quad + \quad (G/e) && \textbf{(CS3)} \\ &= (G+e\backslash f) && \textbf{(CS4)} \\ &= (G+e\backslash f\backslash g) \quad - \quad (G+e\backslash f/g) && \textbf{(CS1)} \\ &= (G+e\backslash f\backslash g+h) && \textbf{(CS2)} \end{aligned}$$

Schema 12:

$$\begin{aligned} G &= (G\backslash e) \quad - \quad (G/e) && \textbf{(CS1)} \\ &= (G\backslash e+f) \quad - \quad (G\backslash e/f) \quad + \quad (G/e) && \textbf{(CS3)} \\ &= (G\backslash e+f\backslash g) \quad - \quad (G/e) && \textbf{(CS4)} \\ &= (G+e\backslash f\backslash g+h) && \textbf{(CS2)} \end{aligned}$$

Schema 13:

$$
\begin{aligned}
G &= (G+e) \;+\; (G/e) && \textbf{(CS3)} \\
&= (G+e) \;+\; (G/e \backslash f) \;-\; (G/e/f) && \textbf{(CS1)} \\
&= (G+e) \;+\; (G/e \backslash f + g) && \textbf{(CS2)} \\
&= (G+e \backslash h) && \textbf{(CS4)}
\end{aligned}
$$

Schema 14:

$$
\begin{aligned}
G &= (G+e) \;+\; (G/e) && \textbf{(CS3)} \\
&= (G+e+f) \;+\; (G+e/f) \;+\; (G/e) && \textbf{(CS3)} \\
&= (G+e+f) \;+\; (G+e/f+g) \;+\; (G+e/f/g) \;+\; (G/e) && \textbf{(CS3)} \\
&= (G+e+f) \;+\; (G+e/f+g \backslash h) \;+\; (G/e) && \textbf{(CS4)} \\
&= (G+e+f \backslash i) \;+\; (G+e/f+g \backslash h) && \textbf{(CS4)} \\
&= (G+e+f \backslash i \backslash j) && \textbf{(CS4)}
\end{aligned}
$$

10 Appendix C Certificates

The following are all of the certificates found for pairs of chromaticaly equivlent pairs of graphs of order $4 \leq n \leq 6$. A somewhat more verbose version of these certificates, along with all of the certificates for the graphs of order 7, can be found in the files labeled `order_*_certificates` on the accompanying CD.

The numbers given to denote which graphs each certificate corresponds to are those listed in the `graphs*` files, also found on the CD. These files give the adjacency matrices of the graphs.

ORDER 4:

```
-----------------------------------------
GRAPH PAIR: 2 & 1 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}
```

ORDER 5:

```
-----------------------------------------
GRAPH PAIR: 2 & 1 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,3)}

-----------------------------------------
GRAPH PAIR: 9 & 1 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(2,0)} - G4{G1/(2,0)} + G2{G0/(1,0)}
    = G5{G3+(2,1)} + G2{G0/(1,0)}
    = G6{G5-(4,0)}

-----------------------------------------
GRAPH PAIR: 9 & 2 CERT LENGTH: 2

G   = G1{G0-(2,0)} - G2{G0/(2,0)}
    = G3{G1+(2,1)}

-----------------------------------------
GRAPH PAIR: 4 & 3 CERT LENGTH: 2

G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(4,1)}
```

```
------------------------------------------
GRAPH PAIR: 10 & 3 CERT LENGTH: 2

G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(4,3)}

------------------------------------------
GRAPH PAIR: 10 & 4 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(2,0)}

------------------------------------------
GRAPH PAIR: 11 & 6 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(2,0)}

------------------------------------------
GRAPH PAIR: 12 & 6 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,2)}

------------------------------------------
GRAPH PAIR: 12 & 11 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,2)}
```

ORDER 6:

```
------------------------------------------
GRAPH PAIR: 59 & 11 CERT LENGTH: 2

G   = G1{G0+(2,0)} + G2{G0/(2,0)}
    = G3{G1-(5,2)}

------------------------------------------
GRAPH PAIR: 9 & 7 CERT LENGTH: 2

G   = G1{G0-(4,2)} - G2{G0/(4,2)}
    = G3{G1+(5,2)}

------------------------------------------
GRAPH PAIR: 23 & 7 CERT LENGTH: 2
```

```
G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(5,3)}

------------------------------------------
GRAPH PAIR: 23 & 9 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,3)}

------------------------------------------
GRAPH PAIR: 31 & 7 CERT LENGTH: 2

G   = G1{G0+(2,0)} + G2{G0/(2,0)}
    = G3{G1-(5,2)}

------------------------------------------
GRAPH PAIR: 31 & 9 CERT LENGTH: 2

G   = G1{G0+(2,1)} + G2{G0/(2,1)}
    = G3{G1-(5,2)}

------------------------------------------
GRAPH PAIR: 31 & 23 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(3,2)}

------------------------------------------
GRAPH PAIR: 2 & 1 CERT LENGTH: 2

G   = G1{G0-(4,0)} - G2{G0/(4,0)}
    = G3{G1+(5,4)}

------------------------------------------
GRAPH PAIR: 4 & 1 CERT LENGTH: 4

G   = G1{G0-(4,0)} - G2{G0/(4,0)}
    = G3{G1+(5,0)}
    = G4{G3-(4,1)} - G5{G3/(4,1)}
    = G6{G4+(5,1)}

------------------------------------------
GRAPH PAIR: 4 & 2 CERT LENGTH: 2

G   = G1{G0-(4,0)} - G2{G0/(4,0)}
    = G3{G1+(5,0)}

------------------------------------------
```

```
GRAPH PAIR: 5 & 1 CERT LENGTH: 4

G   = G1{G0-(4,1)} - G2{G0/(4,1)}
    = G3{G1+(5,1)}
    = G4{G3-(4,0)} - G5{G3/(4,0)}
    = G6{G4+(5,4)}

------------------------------------------
GRAPH PAIR: 5 & 2 CERT LENGTH: 2

G   = G1{G0-(4,1)} - G2{G0/(4,1)}
    = G3{G1+(5,1)}

------------------------------------------
GRAPH PAIR: 5 & 4 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,1)}

------------------------------------------
GRAPH PAIR: 15 & 1 CERT LENGTH: 4

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(5,3)}
    = G4{G3-(4,1)} - G5{G3/(4,1)}
    = G6{G4+(5,4)}

------------------------------------------
GRAPH PAIR: 15 & 2 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(5,3)}

------------------------------------------
GRAPH PAIR: 15 & 4 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(3,1)}

------------------------------------------
GRAPH PAIR: 15 & 5 CERT LENGTH: 2

G   = G1{G0+(2,0)} + G2{G0/(2,0)}
    = G3{G1-(5,2)}

------------------------------------------
GRAPH PAIR: 19 & 1 CERT LENGTH: 6
```

```
G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(2,0)} + G4{G1/(2,0)} + G2{G0/(1,0)}
    = G5{G3-(4,1)} + G2{G0/(1,0)}
    = G6{G5-(5,1)}
    = G7{G6+(4,0)} + G8{G6/(4,0)}
    = G9{G7-(4,2)}

-----------------------------------------
GRAPH PAIR: 19 & 2 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(2,0)} + G4{G1/(2,0)} + G2{G0/(1,0)}
    = G5{G3-(4,1)} + G2{G0/(1,0)}
    = G6{G5-(5,1)}

-----------------------------------------
GRAPH PAIR: 19 & 4 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(2,0)} + G4{G1/(2,0)} + G2{G0/(1,0)}
    = G5{G3-(4,2)} + G2{G0/(1,0)}
    = G6{G5-(5,0)}

-----------------------------------------
GRAPH PAIR: 19 & 5 CERT LENGTH: 2

G   = G1{G0+(2,0)} + G2{G0/(2,0)}
    = G3{G1-(4,2)}

-----------------------------------------
GRAPH PAIR: 19 & 15 CERT LENGTH: 2

G   = G1{G0+(2,0)} + G2{G0/(2,0)}
    = G3{G1-(4,1)}

-----------------------------------------
GRAPH PAIR: 107 & 74 CERT LENGTH: 2

G   = G1{G0+(3,2)} + G2{G0/(3,2)}
    = G3{G1-(5,3)}

-----------------------------------------
GRAPH PAIR: 33 & 25 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(3,2)}

-----------------------------------------
```

```
GRAPH PAIR: 46 & 25 CERT LENGTH: 4

G   = G1{G0+(2,1)} + G2{G0/(2,1)}
    = G3{G1+(5,0)} + G4{G1/(5,0)} + G2{G0/(2,1)}
    = G4{G1/(5,0)} + G5{G3-(3,1)}
    = G6{G5-(4,0)}

------------------------------------------
GRAPH PAIR: 46 & 33 CERT LENGTH: 2

G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(4,1)}

------------------------------------------
GRAPH PAIR: 51 & 25 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(5,3)}

------------------------------------------
GRAPH PAIR: 51 & 33 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}

------------------------------------------
GRAPH PAIR: 51 & 46 CERT LENGTH: 2

G   = G1{G0+(2,1)} + G2{G0/(2,1)}
    = G3{G1-(3,1)}

------------------------------------------
GRAPH PAIR: 60 & 36 CERT LENGTH: 2

G   = G1{G0+(2,0)} + G2{G0/(2,0)}
    = G3{G1-(5,2)}

------------------------------------------
GRAPH PAIR: 65 & 36 CERT LENGTH: 2

G   = G1{G0+(2,0)} + G2{G0/(2,0)}
    = G3{G1-(4,2)}

------------------------------------------
GRAPH PAIR: 65 & 60 CERT LENGTH: 2

G   = G1{G0+(3,2)} + G2{G0/(3,2)}
    = G3{G1-(4,2)}
```

```
-----------------------------------------
GRAPH PAIR: 63 & 61 CERT LENGTH: 2

G   = G1{G0+(3,2)} + G2{G0/(3,2)}
    = G3{G1-(5,2)}

-----------------------------------------
GRAPH PAIR: 6 & 3 CERT LENGTH: 2

G   = G1{G0-(4,1)} - G2{G0/(4,1)}
    = G3{G1+(5,1)}

-----------------------------------------
GRAPH PAIR: 16 & 3 CERT LENGTH: 2

G   = G1{G0-(4,1)} - G2{G0/(4,1)}
    = G3{G1+(5,4)}

-----------------------------------------
GRAPH PAIR: 16 & 6 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,0)}

-----------------------------------------
GRAPH PAIR: 18 & 3 CERT LENGTH: 4

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,0)}
    = G4{G3+(4,3)} + G5{G3/(4,3)}
    = G6{G4-(5,0)}

-----------------------------------------
GRAPH PAIR: 18 & 6 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,0)}

-----------------------------------------
GRAPH PAIR: 18 & 16 CERT LENGTH: 2

G   = G1{G0-(4,1)} - G2{G0/(4,1)}
    = G3{G1+(5,1)}

-----------------------------------------
GRAPH PAIR: 20 & 3 CERT LENGTH: 4
```

```
G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,3)}
    = G4{G3+(4,0)} + G5{G3/(4,0)}
    = G6{G4-(5,0)}

-----------------------------------------
GRAPH PAIR: 20 & 6 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,3)}

-----------------------------------------
GRAPH PAIR: 20 & 16 CERT LENGTH: 2

G   = G1{G0-(4,2)} - G2{G0/(4,2)}
    = G3{G1+(5,2)}

-----------------------------------------
GRAPH PAIR: 20 & 18 CERT LENGTH: 2

G   = G1{G0+(2,0)} + G2{G0/(2,0)}
    = G3{G1-(4,2)}

-----------------------------------------
GRAPH PAIR: 21 & 3 CERT LENGTH: 4

G   = G1{G0-(4,2)} - G2{G0/(4,2)}
    = G3{G1+(5,2)}
    = G4{G3-(4,1)} - G5{G3/(4,1)}
    = G6{G4+(5,4)}

-----------------------------------------
GRAPH PAIR: 21 & 6 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,0)}
    = G4{G3+(2,1)} + G5{G3/(2,1)}
    = G6{G4-(4,2)}

-----------------------------------------
GRAPH PAIR: 21 & 16 CERT LENGTH: 2

G   = G1{G0-(4,2)} - G2{G0/(4,2)}
    = G3{G1+(5,2)}

-----------------------------------------
GRAPH PAIR: 21 & 18 CERT LENGTH: 2
```

```
G   = G1{G0+(2,1)} + G2{G0/(2,1)}
    = G3{G1-(3,0)}

------------------------------------------
GRAPH PAIR: 21 & 20 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,0)}

------------------------------------------
GRAPH PAIR: 30 & 3 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,1)}
    = G4{G3+(2,0)} + G5{G3/(2,0)}
    = G6{G4-(5,2)}

------------------------------------------
GRAPH PAIR: 30 & 6 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,1)}

------------------------------------------
GRAPH PAIR: 30 & 16 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(5,4)}

------------------------------------------
GRAPH PAIR: 30 & 18 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,0)} - G4{G1/(4,0)} + G2{G0/(1,0)}
    =  - G4{G1/(4,0)} + G5{G3-(4,1)}
    = G6{G5+(3,1)}

------------------------------------------
GRAPH PAIR: 30 & 20 CERT LENGTH: 2

G   = G1{G0+(2,1)} + G2{G0/(2,1)}
    = G3{G1-(4,1)}

------------------------------------------
GRAPH PAIR: 30 & 21 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(5,4)}
```

```
    = G4{G3-(3,0)} - G5{G3/(3,0)}
    = G6{G4+(3,2)}

------------------------------------------
GRAPH PAIR: 49 & 39 CERT LENGTH: 2

G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(4,1)}

------------------------------------------
GRAPH PAIR: 54 & 39 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}

------------------------------------------
GRAPH PAIR: 54 & 49 CERT LENGTH: 2

G   = G1{G0+(2,1)} + G2{G0/(2,1)}
    = G3{G1-(3,1)}

------------------------------------------
GRAPH PAIR: 82 & 39 CERT LENGTH: 4

G   = G1{G0-(2,0)} - G2{G0/(2,0)}
    = G3{G1+(2,1)} + G4{G1/(2,1)} - G2{G0/(2,0)}
    = G5{G3-(4,2)} - G2{G0/(2,0)}
    = G6{G5+(5,2)}

------------------------------------------
GRAPH PAIR: 82 & 49 CERT LENGTH: 2

G   = G1{G0-(2,0)} - G2{G0/(2,0)}
    = G3{G1+(2,1)}

------------------------------------------
GRAPH PAIR: 82 & 54 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G1{G0+(1,0)} + G3{G2-(1,0)} - G4{G2/(1,0)}
    = G1{G0+(1,0)} + G5{G3+(3,2)}
    = G6{G1-(4,0)}

------------------------------------------
GRAPH PAIR: 67 & 57 CERT LENGTH: 2

G   = G1{G0+(2,0)} + G2{G0/(2,0)}
    = G3{G1-(4,2)}
```

```
-----------------------------------------
GRAPH PAIR: 85 & 57 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(2,0)}

-----------------------------------------
GRAPH PAIR: 85 & 67 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(2,0)}
    = G4{G3+(3,0)} + G5{G3/(3,0)}
    = G6{G4-(5,3)}

-----------------------------------------
GRAPH PAIR: 100 & 69 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(5,1)}

-----------------------------------------
GRAPH PAIR: 87 & 71 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(2,0)}

-----------------------------------------
GRAPH PAIR: 10 & 8 CERT LENGTH: 2

G   = G1{G0-(4,2)} - G2{G0/(4,2)}
    = G3{G1+(5,2)}

-----------------------------------------
GRAPH PAIR: 17 & 8 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,0)}

-----------------------------------------
GRAPH PAIR: 17 & 10 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(2,0)} + G4{G1/(2,0)} + G2{G0/(1,0)}
    = G5{G3-(5,2)} + G2{G0/(1,0)}
    = G6{G5-(4,1)}

-----------------------------------------
```

```
GRAPH PAIR: 22 & 8 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,0)}
    = G4{G3-(4,2)} - G5{G3/(4,2)}
    = G6{G4+(5,2)}

-----------------------------------------
GRAPH PAIR: 22 & 10 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,0)}

-----------------------------------------
GRAPH PAIR: 22 & 17 CERT LENGTH: 2

G   = G1{G0-(4,2)} - G2{G0/(4,2)}
    = G3{G1+(5,2)}

-----------------------------------------
GRAPH PAIR: 24 & 8 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(5,3)}

-----------------------------------------
GRAPH PAIR: 24 & 10 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,3)}

-----------------------------------------
GRAPH PAIR: 24 & 17 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,1)}
    = G4{G3-(3,0)} - G5{G3/(3,0)}
    = G6{G4+(5,3)}

-----------------------------------------
GRAPH PAIR: 24 & 22 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,1)}

-----------------------------------------
GRAPH PAIR: 27 & 8 CERT LENGTH: 2
```

```
G   = G1{G0-(4,0)} - G2{G0/(4,0)}
    = G3{G1+(5,4)}

------------------------------------------
GRAPH PAIR: 27 & 10 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G1{G0+(1,0)} + G3{G2+(1,0)} + G4{G2/(1,0)}
    = G1{G0+(1,0)} + G5{G3-(3,2)}
    = G6{G1-(4,3)}

------------------------------------------
GRAPH PAIR: 27 & 17 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(5,1)}
    = G4{G3+(2,0)} + G5{G3/(2,0)}
    = G6{G4-(5,3)}

------------------------------------------
GRAPH PAIR: 27 & 22 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(2,1)} + G4{G1/(2,1)} + G2{G0/(1,0)}
    = G5{G3-(4,0)} + G2{G0/(1,0)}
    = G6{G5-(1,0)}

------------------------------------------
GRAPH PAIR: 27 & 24 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(2,1)} + G4{G1/(2,1)} + G2{G0/(1,0)}
    = G5{G3-(4,0)} + G2{G0/(1,0)}
    = G6{G5-(3,0)}

------------------------------------------
GRAPH PAIR: 32 & 8 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(5,3)}

------------------------------------------
GRAPH PAIR: 32 & 10 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,1)}

------------------------------------------
```

```
GRAPH PAIR: 32 & 17 CERT LENGTH: 2

G   = G1{G0+(5,3)} + G2{G0/(5,3)}
    = G3{G1-(4,0)}

-----------------------------------------
GRAPH PAIR: 32 & 22 CERT LENGTH: 2

G   = G1{G0+(2,1)} + G2{G0/(2,1)}
    = G3{G1-(4,1)}

-----------------------------------------
GRAPH PAIR: 32 & 24 CERT LENGTH: 2

G   = G1{G0+(2,1)} + G2{G0/(2,1)}
    = G3{G1-(4,0)}

-----------------------------------------
GRAPH PAIR: 32 & 27 CERT LENGTH: 2

G   = G1{G0+(2,0)} + G2{G0/(2,0)}
    = G3{G1-(5,2)}

-----------------------------------------
GRAPH PAIR: 50 & 8 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}
    = G4{G3+(2,0)} + G5{G3/(2,0)}
    = G6{G4-(4,2)}

-----------------------------------------
GRAPH PAIR: 50 & 10 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(4,3)} + G4{G1/(4,3)} + G2{G0/(1,0)}
    = G4{G1/(4,3)} + G5{G3-(5,0)}
    = G6{G5-(1,0)}

-----------------------------------------
GRAPH PAIR: 50 & 17 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}
    = G4{G3+(2,0)} + G5{G3/(2,0)}
    = G6{G4-(5,4)}

-----------------------------------------
```

```
GRAPH PAIR: 50 & 22 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(5,3)}

------------------------------------------
GRAPH PAIR: 50 & 24 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(5,4)}

------------------------------------------
GRAPH PAIR: 50 & 27 CERT LENGTH: 2

G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(4,1)}

------------------------------------------
GRAPH PAIR: 50 & 32 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}

------------------------------------------
GRAPH PAIR: 76 & 8 CERT LENGTH: 6

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(3,0)} + G4{G1/(3,0)} + G2{G0/(1,0)}
    = G3{G1+(3,0)} + G5{G4+(2,1)} + G6{G4/(2,1)} + G2{G0/(1,0)}
    = G3{G1+(3,0)} + G7{G5-(3,2)} + G2{G0/(1,0)}
    = G7{G5-(3,2)} + G8{G3-(3,1)}
    = G9{G8-(4,2)}

------------------------------------------
GRAPH PAIR: 76 & 10 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}
    = G4{G3-(4,2)} - G5{G3/(4,2)}
    = G6{G4+(5,2)}

------------------------------------------
GRAPH PAIR: 76 & 17 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(3,0)} + G4{G1/(3,0)} + G2{G0/(1,0)}
    = G5{G3-(3,1)} + G2{G0/(1,0)}
    = G6{G5-(5,1)}
```

```
-----------------------------------------
GRAPH PAIR: 76 & 22 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}

-----------------------------------------
GRAPH PAIR: 76 & 24 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(5,1)} - G4{G1/(5,1)} + G2{G0/(1,0)}
    =  - G4{G1/(5,1)} + G5{G3-(3,1)}
    = G6{G5+(2,1)}

-----------------------------------------
GRAPH PAIR: 76 & 27 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G1{G0+(1,0)} + G3{G2-(1,0)} - G4{G2/(1,0)}
    = G1{G0+(1,0)} + G5{G3+(3,2)}
    = G6{G1-(4,2)}

-----------------------------------------
GRAPH PAIR: 76 & 32 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}
    = G4{G3+(2,1)} + G5{G3/(2,1)}
    = G6{G4-(4,2)}

-----------------------------------------
GRAPH PAIR: 76 & 50 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}
    = G4{G3-(2,0)} - G5{G3/(2,0)}
    = G6{G4+(4,1)}

-----------------------------------------
GRAPH PAIR: 77 & 8 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(3,0)} + G4{G1/(3,0)} + G2{G0/(1,0)}
    = G5{G3-(3,1)} + G2{G0/(1,0)}
    = G6{G5-(4,2)}

-----------------------------------------
```

```
GRAPH PAIR: 77 & 10 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)} - G4{G1/(3,1)} + G2{G0/(1,0)}
    = G5{G3+(3,2)} + G2{G0/(1,0)}
    = G6{G5-(5,1)}

------------------------------------------
GRAPH PAIR: 77 & 17 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(3,0)} + G4{G1/(3,0)} + G2{G0/(1,0)}
    = G5{G3-(3,1)} + G2{G0/(1,0)}
    = G6{G5-(5,2)}

------------------------------------------
GRAPH PAIR: 77 & 22 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)} - G4{G1/(3,1)} + G2{G0/(1,0)}
    = G5{G3+(3,2)} + G2{G0/(1,0)}
    = G6{G5-(5,2)}

------------------------------------------
GRAPH PAIR: 77 & 24 CERT LENGTH: 2

G   = G1{G0+(3,0)} + G2{G0/(3,0)}
    = G3{G1-(5,1)}

------------------------------------------
GRAPH PAIR: 77 & 27 CERT LENGTH: 2

G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(5,3)}

------------------------------------------
GRAPH PAIR: 77 & 32 CERT LENGTH: 2

G   = G1{G0+(3,0)} + G2{G0/(3,0)}
    = G3{G1-(3,1)}

------------------------------------------
GRAPH PAIR: 77 & 50 CERT LENGTH: 2

G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(4,3)}

------------------------------------------
```

```
GRAPH PAIR: 77 & 76 CERT LENGTH: 2

G   = G1{G0-(5,0)} - G2{G0/(5,0)}
    = G3{G1+(5,3)}

------------------------------------------
GRAPH PAIR: 26 & 12 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,0)}

------------------------------------------
GRAPH PAIR: 34 & 12 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,1)}

------------------------------------------
GRAPH PAIR: 34 & 26 CERT LENGTH: 2

G   = G1{G0+(2,1)} + G2{G0/(2,1)}
    = G3{G1-(4,0)}

------------------------------------------
GRAPH PAIR: 40 & 12 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,3)}

------------------------------------------
GRAPH PAIR: 40 & 26 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,0)} - G4{G1/(4,0)} + G2{G0/(1,0)}
    =  - G4{G1/(4,0)} + G5{G3-(4,1)}
    = G6{G5+(5,2)}

------------------------------------------
GRAPH PAIR: 40 & 34 CERT LENGTH: 2

G   = G1{G0-(4,2)} - G2{G0/(4,2)}
    = G3{G1+(5,2)}

------------------------------------------
GRAPH PAIR: 41 & 12 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,3)}
```

```
----------------------------------------
GRAPH PAIR: 41 & 26 CERT LENGTH: 2

G   = G1{G0-(4,0)} - G2{G0/(4,0)}
    = G3{G1+(5,3)}

----------------------------------------
GRAPH PAIR: 41 & 34 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,0)} - G4{G1/(4,0)} + G2{G0/(1,0)}
    =  - G4{G1/(4,0)} + G5{G3-(4,2)}
    = G6{G5+(5,3)}

----------------------------------------
GRAPH PAIR: 41 & 40 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,1)}

----------------------------------------
GRAPH PAIR: 52 & 12 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}
    = G4{G3+(2,0)} + G5{G3/(2,0)}
    = G6{G4-(4,2)}

----------------------------------------
GRAPH PAIR: 52 & 26 CERT LENGTH: 2

G   = G1{G0+(5,2)} + G2{G0/(5,2)}
    = G3{G1-(3,0)}

----------------------------------------
GRAPH PAIR: 52 & 34 CERT LENGTH: 2

G   = G1{G0-(4,2)} - G2{G0/(4,2)}
    = G3{G1+(5,2)}

----------------------------------------
GRAPH PAIR: 52 & 40 CERT LENGTH: 2

G   = G1{G0+(2,0)} + G2{G0/(2,0)}
    = G3{G1-(3,1)}

----------------------------------------
```

```
GRAPH PAIR: 52 & 41 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}

------------------------------------------
GRAPH PAIR: 79 & 12 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}
    = G4{G3-(4,2)} - G5{G3/(4,2)}
    = G6{G4+(3,0)}

------------------------------------------
GRAPH PAIR: 79 & 26 CERT LENGTH: 2

G   = G1{G0-(4,0)} - G2{G0/(4,0)}
    = G3{G1+(5,4)}

------------------------------------------
GRAPH PAIR: 79 & 34 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(3,0)} + G4{G1/(3,0)} + G2{G0/(1,0)}
    = G5{G3-(3,1)} + G2{G0/(1,0)}
    = G6{G5-(5,1)}

------------------------------------------
GRAPH PAIR: 79 & 40 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}

------------------------------------------
GRAPH PAIR: 79 & 41 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(2,0)} - G4{G1/(2,0)} + G2{G0/(1,0)}
    =  - G4{G1/(2,0)} + G5{G3-(4,2)}
    = G6{G5+(2,1)}

------------------------------------------
GRAPH PAIR: 79 & 52 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,0)}

------------------------------------------
```

```
GRAPH PAIR: 43 & 14 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(4,3)}

------------------------------------------
GRAPH PAIR: 55 & 14 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}
    = G4{G3+(2,0)} + G5{G3/(2,0)}
    = G6{G4-(4,2)}

------------------------------------------
GRAPH PAIR: 55 & 43 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}

------------------------------------------
GRAPH PAIR: 56 & 14 CERT LENGTH: 4

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(5,0)}
    = G4{G3-(3,1)} - G5{G3/(3,1)}
    = G6{G4+(4,1)}

------------------------------------------
GRAPH PAIR: 56 & 43 CERT LENGTH: 2

G   = G1{G0-(3,0)} - G2{G0/(3,0)}
    = G3{G1+(5,0)}

------------------------------------------
GRAPH PAIR: 56 & 55 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,0)} - G4{G1/(3,0)} + G2{G0/(1,0)}
    = G5{G3+(5,0)} + G2{G0/(1,0)}
    = G6{G5-(4,0)}

------------------------------------------
GRAPH PAIR: 83 & 14 CERT LENGTH: 4

G   = G1{G0-(2,0)} - G2{G0/(2,0)}
    = G3{G1+(5,2)}
    = G4{G3-(3,1)} - G5{G3/(3,1)}
    = G6{G4+(4,3)}
```

```
------------------------------------------
GRAPH PAIR: 83 & 43 CERT LENGTH: 2

G   = G1{G0-(2,0)} - G2{G0/(2,0)}
    = G3{G1+(5,2)}

------------------------------------------
GRAPH PAIR: 83 & 55 CERT LENGTH: 2

G   = G1{G0-(4,2)} - G2{G0/(4,2)}
    = G3{G1+(5,2)}

------------------------------------------
GRAPH PAIR: 83 & 56 CERT LENGTH: 2

G   = G1{G0-(2,0)} - G2{G0/(2,0)}
    = G3{G1+(2,1)}

------------------------------------------
GRAPH PAIR: 98 & 92 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(5,2)}

------------------------------------------
GRAPH PAIR: 106 & 102 CERT LENGTH: 2

G   = G1{G0-(2,0)} - G2{G0/(2,0)}
    = G3{G1+(4,3)}

------------------------------------------
GRAPH PAIR: 35 & 28 CERT LENGTH: 2

G   = G1{G0-(4,1)} - G2{G0/(4,1)}
    = G3{G1+(5,1)}

------------------------------------------
GRAPH PAIR: 78 & 28 CERT LENGTH: 2

G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(5,3)}

------------------------------------------
GRAPH PAIR: 78 & 35 CERT LENGTH: 2

G   = G1{G0+(3,0)} + G2{G0/(3,0)}
    = G3{G1-(3,1)}
```

```
----------------------------------------
GRAPH PAIR: 62 & 37 CERT LENGTH: 2

G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(4,2)}

----------------------------------------
GRAPH PAIR: 80 & 37 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,1)}

----------------------------------------
GRAPH PAIR: 80 & 62 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(2,1)} + G4{G1/(2,1)} + G2{G0/(1,0)}
    = G5{G3-(4,0)} + G2{G0/(1,0)}
    = G6{G5-(5,3)}

----------------------------------------
GRAPH PAIR: 84 & 37 CERT LENGTH: 2

G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(4,3)}

----------------------------------------
GRAPH PAIR: 84 & 62 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(2,0)}

----------------------------------------
GRAPH PAIR: 84 & 80 CERT LENGTH: 2

G   = G1{G0-(4,1)} - G2{G0/(4,1)}
    = G3{G1+(5,3)}

----------------------------------------
GRAPH PAIR: 58 & 45 CERT LENGTH: 2

G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(4,1)}

----------------------------------------
GRAPH PAIR: 86 & 45 CERT LENGTH: 2
```

```
G   = G1{G0-(3,1)} - G2{G0/(3,1)}
    = G3{G1+(4,3)}

------------------------------------------
GRAPH PAIR: 86 & 58 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(2,0)}

------------------------------------------
GRAPH PAIR: 91 & 45 CERT LENGTH: 4

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1+(2,1)} + G4{G1/(2,1)} + G2{G0/(1,0)}
    = G5{G3-(4,1)} + G2{G0/(1,0)}
    = G6{G5-(5,1)}

------------------------------------------
GRAPH PAIR: 91 & 58 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,1)}

------------------------------------------
GRAPH PAIR: 91 & 86 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(4,2)}

------------------------------------------
GRAPH PAIR: 89 & 64 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(5,1)}

------------------------------------------
GRAPH PAIR: 88 & 70 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(2,0)}

------------------------------------------
GRAPH PAIR: 93 & 70 CERT LENGTH: 2

G   = G1{G0+(1,0)} + G2{G0/(1,0)}
    = G3{G1-(3,2)}

------------------------------------------
```

```
GRAPH PAIR: 93 & 88 CERT LENGTH: 2

G    = G1{G0+(1,0)} + G2{G0/(1,0)}
     = G3{G1-(4,2)}

------------------------------------------
GRAPH PAIR: 103 & 75 CERT LENGTH: 2

G    = G1{G0+(1,0)} + G2{G0/(1,0)}
     = G3{G1-(3,1)}
```