Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Minimisation of the signed four colour theorem's counterexample

Master Thesis

2023

Matúš Matok

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Minimisation of the signed four colour theorem's counterexample

Master Thesis

Study Programme:	Computer Science
Field of Study:	Computer Science
Department:	Department of Computer Science
Supervisor:	RNDr. Ing. František Kardoš, PhD.

Bratislava, 2023
Matúš Matok

97878874

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Matúš Matok
Študijný program:	informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	diplomová
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Minimalizácia protipríkladu vety o štyroch farbách pre signované planárne grafy
Minimisation of the signed four colour theorem's counterexample

Anotácia: Je známe, že Veta o štyroch farbách sa nedá rozšíriť na signované planárne grafy - existujú konštrukcie takých signovaných planárnych grafov, ktoré nie sú zafarbiteľné štyrmi farbami. Nie je doteraz potvrdené, či najmenší doteraz nájdený kontrapríklad je aj najmenší možný.

Cieľ: Cieľom práce je zlepšiť doteraz známy dolný i horný odhad veľkosti najmenšieho kontrapríkladu, a to pomocou systematického prehľadávania s pomocou počítača, a tiež s využitím kombinatorických invariantov.

Kľúčové slová: Signovaný grqf, planárny graf, farbenie grafov

Vedúci:	RNDr. Ing. František Kardoš, PhD.
Katedra:	FMFI.KI - Katedra informatiky
Vedúci katedry:	prof. RNDr. Martin Škoviera, PhD.
Dátum zadania:	14.12.2021

Dátum schválenia: 27.04.2023

prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

....
študent

....
vedúci práce

97878874

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Matúš Matok
Study programme:	Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Diploma Thesis
Language of Thesis:	Slovak
Secondary language:	English

Title: Minimalizácia protipríkladu vety o štyroch farbách pre signované planárne grafy
Minimisation of the signed four colour theorem's counterexample

Annotation: Je známe, že Veta o štyroch farbách sa nedá rozšíriť na signované planárne grafy - existujú konštrukcie takých signovaných planárnych grafov, ktoré nie sú zafarbiteľné štyrmi farbami. Nie je doteraz potvrdené, či najmenší doteraz nájdený kontrapríklad je aj najmenší možný.

Aim: Cieľom práce je zlepšiť doteraz známy dolný i horný odhad veľkosti najmenšieho kontrapríkladu, a to pomocou systematického prehľadávania s pomocou počítača, a tiež s využitím kombinatorických invariantov.

Keywords: Signovaný grqf, planárny graf, farbenie grafov

Supervisor:	RNDr. Ing. František Kardoš, PhD.
Department:	FMFI.KI - Department of Computer Science
Head of department:	prof. RNDr. Martin Škoviera, PhD.
Assigned:	14.12.2021

Approved: 27.04.2023 prof. RNDr. Rastislav Kráľovič, PhD.
Guarantor of Study Programme

....
Student

....
Supervisor

Acknowledgments: I would like to thank my supervisor Mr. RNDr. Ing. František Kardoš, PhD. for his enormous contribution and dedication to this work. His friendly and supportive approach cannot be appreciated enough.

Abstrakt

Pojem signovaného grafu je zovšeobecnením tradičného pojmu graf. Máčajová, Raspaud a Škoviera vyslovili hypotézu, že veta o 4 farbách platí aj pre signované planárne grafy. Táto hypotéza bola zakrátko vyvrátená Kardošom a Narbonim, ktorí škonštruovali protipríklad. Našim cieľom je nájsť menší protipríklad tým, že sa budeme zaoberať štruktúrou duálneho grafu. Navrhli a implementovali sme počítačové prehľadávanie, ktoré overuje vlastnosti indukovaných podgrafov duálneho grafu.

Kľúčové slová: protipríklad, kubický graf, signovaný planárny graf, duálny graf

Abstract

A signed graph is a generalisation of the traditional concept of a graph. A conjecture that the four colour theorem for planar graphs holds for signed planar graphs as well was brought up Mačajova, Raspaud and Škoviera. It got soon disputed by Kardoš and Narboni who constructed a counterexample. Our goal is to find a smaller counterexample by studying the properties of the dual. We implement an exhaustive computer-assisted search that verifies the properties on induced subgraphs of the dual.

Keywords: counterexample, cubic graph, signed planar graph, dual graph

Contents

Introduction

From ancient ages, humanity has been competition-driven. In the past, being among the best was a matter of survival. Fortunately nowadays, it's mostly matter of bragging rights and other related benefits. Who can throw the javelin the furthest? Who can run a track in the shortest period of time? Those are great areas to be competitive in. But why stop there? Why not extend this concept to the realm of intellect? Fortunately again, it is not a novel idea. Novel or poem, who can write the most intriguing one? Highly subjective, but we are getting closer. Who can solve the most mathematical equations in an hour? Impressive, but those equations have been solved before. How about answering a question that no one ever answered before? That must be truly rewarding. It resembles of what science is about. Literature, chemistry, physics. Computer science? Is that even a science? Inventing new gadgets, crafting new theories just to research them? Sounds like a thing a lunatic would do. Fortunately (again), I can call myself a lunatic to a degree. To improve this degree, I present you, dear reader, my master thesis. A concept of signed graph was introduced by Thomas Zaslavsky in 1982. It is a generalisation of the usual concept of a graph. In 2016 Máčajová, Raspaud and Škoviera brought up a conjecture that the four colour theorem holds for signed planar graphs as well. This got disputed in 2019 by Kardoš and Narboni who constructed a counterexample. The topic of this work will be a computer-assisted search that verifies if the counterexample is the smallest possible or there exists a smaller one. Firstly, we are going to introduce necessary notions and notation, then present the known results, followed by an introduction of our concepts and theoretical observations and finished by the implementation and results of the work.

Chapter 1

Notions, notations & known results

1.1 Basic notations

In this work we will use commonly used terminology in the field of graph theory. For that purpose we will adopt notions and notation used by Diestel [2].

A *graph* is a pair $G = (V, E)$ of sets such that $E \subseteq [V]^2$; thus, the elements of E are 2-element subsets of V. To avoid notational ambiguities, we shall always assume tacitly that $V \cap E = \emptyset$. The elements of V are the *vertices* (or *nodes*, or *points*) of the graph G, the elements of E are its *edges* (or *lines*). A graph with vertex set V is said to be a *graph on* V. The vertex set of a graph G is referred to as $V(G)$, its edge set as $E(G)$. These conventions are independent of any actual names of these two sets: the vertex set W of a graph $H = (W, F)$ is still referred to as $V(H)$, not as $W(H)$. We shall not always distinguish strictly between a graph and its vertex or edge set. For example, we may speak of a vertex $v \in G$ (rather than $v \in V(G)$), an edge $e \in G$, and so on.

A vertex v is *incident* with an edge e if $v \in e$; then e is an *edge at* v. The two vertices incident with an edge are its *endvertices* or *ends*, and an edge *joins* its ends.

Two vertices x, y of G are *adjacent*, or *neighbours*, if xy is an edge of G. Two edges $e \neq f$ are *adjacent* if they have an end in common vertex. If all the vertices of G are pairwise adjacent, then G is complete. A complete graph on n vertices is denoted as K_n.

The *degree* (or *valency*) $d_G(v) = d(v)$ of a vertex v is the number of edges at v; by our definition of a graph, this is equal to the number of neighbours of v. A vertex of degree 0 is *isolated.* The number $\delta(G) := min\{d(v)|v \in V\}$ is the *minimum degree* of G, the number $\Delta(G) := max\{d(v)|v \in V\}$ its *maximum degree.* If all the vertices of G have the same degree k, then G is *k-regular*, or simply *regular.* A 3-regular graph is called *cubic.*

The hand-shaking lemma (which is considered to be folklore) states that the sum of the degrees of all vertices of a graph equals twice its number of edges, and hence it is always even. As a consequence, every graph has an even number of vertices of odd degree. In particular, a cubic graph always has an even number of vertices.

As each edge is incident to precisely two vertices, the sum off all vertex degrees in a graph G is double the number of its edges. If we subtract the degrees of all even-degree vertices, we are left with an even number, which is a sum of all degrees of odd-degree vertices in G. To produce an even number through addition of odd numbers, we must sum an even number of them.

A *path* is a non-empty graph $P = (V, E)$ of the form

$$V = \{x_0, x_1, \dots, x_k\} \quad E = \{x_0x_1, x_1x_2, \dots, x_{k-1}x_k\},$$

where the x_i are all distinct.

If $P = x_0, \dots, x_{k-1}$ is a path and $k \geq 3$, then the graph $C := P + x_{k-1}x_0$ is called a *cycle*. As with paths, we often denote a cycle by its (cyclic) sequence of vertices; the above cycle C might be written as $x_0 \dots x_{k-1}x_0$. The length of a cycle is its number of edges (or vertices); the cycle of length k is called a k-cycle and denoted by C^k.

If a cycle C contains all vertices of a graph G, we say that C is a *Hamilton* cycle.

Let (X, Y) be a partition of $V(G)$, i.e., $X \cap Y = \emptyset$ and $X \cup Y = V(G)$. Then $E(X, Y) := \{xy \in E(G) | x \in X, y \in Y\}$ is an *edge-cut*. If $|X| = 1$ or $|Y| = 1$ then $E(X, Y)$ is *trivial*.

For a vertex set X, we denote by $\partial(X)$ the *boundary edges* of X, i.e., the set $E(X, V(G) \setminus X)$ of edges having one endvertex in X and the other one outside X.

A non-empty graph G is called *connected* if any two of its vertices are linked by a path in G. Instead of 'not connected' we usually say *'disconnected'*.

G is called k-*connected* (for $k \in \mathbb{N}$) if $|G| > k$ and $G - X$ is connected for every set $X \subseteq V$ with $|X| < k$. In other words, no two vertices of G are separated by fewer than k other vertices. The greatest integer k such that G is k-connected is the *connectivity* $\kappa(G)$ of G.

If $|G| > 1$ and $G - F$ is connected for every set $F \subseteq E$ of fewer than l edges, then G is called l-*edge-connected*. The greatest such that G is l-edge-connected is the *edge-connectivity* $\lambda(G)$ of G. In particular, we have $\lambda(G) = 0$ if G is disconnected.

An *acyclic* graph, one not containing any cycles, is called a *forest*. A connected forest is called a *tree*. Thus, a forest is a graph whose components are trees.

Let $G = (V, E)$ and $G^* = (V^*, E^*)$ be two graphs. We call G and G^* *isomorphic*, and write $G \simeq G^*$, if there exists a bijection $\phi : V \to V^*$ with $xy \in E \iff \phi(x)\phi(y) \in E^*$ for all $x, y \in V$. Such a map ϕ is called an *isomorphism*.

1.2 Planar graphs

Informally, a graph drawn on a piece of paper so that no two edges intersect in a point other than a vertex is called a *plane* graph. An abstract graph that can be drawn in this way is called *planar*. To define them properly, we first need some basic topological definitions and facts. We will once again use those by Diestel.

A *straight line segment* in the Euclidean plane is a subset of $\mathbb{R}^2$ that has the form $\{p + \lambda(q - p) | 0 \leq \lambda \leq 1\}$ for distinct points $p, q \in \mathbb{R}^2$. A *polygon* is a subset of $\mathbb{R}^2$ which is the union of finitely many straight line segments and is homeomorphic to the unit circle S^1, the set of points in $\mathbb{R}^2$ at distance 1 from the origin. A *polygonal arc* is a subset of $\mathbb{R}^2$ which is the union of finitely many straight line segments and is homeomorphic to the closed unit interval $[0, 1]$. The images of 0 and of 1 under such a homeomorphism are the *endpoints* of this *polygonal arc*, which links them and runs between them. Instead of 'polygonal arc' we shall simply say *arc*. If P is an arc between x and y, we denote the point set $P \setminus \{x, y\}$, the *interior* of P, by $\bar{P}$.

Let $O \subseteq \mathbb{R}^2$ be an open set. Being linked by an arc in O defines an equivalence relation on O. The corresponding equivalence classes are again open; they are the *regions* of O. A closed set $X \subseteq \mathbb{R}^2$ is said to *separate* O if $O \setminus X$ has more than one region. The *frontier* of a set $X \subseteq \mathbb{R}^2$ is the set Y of all points $y \in \mathbb{R}^2$ such that every neighbourhood of y meets both X and $\mathbb{R}^2 \setminus X$.

A *plane* graph is a pair (V, E) of finite sets with the following properties (the elements of V are again called vertices, those of E edges):

- $V \subseteq \mathbb{R}^2$;
- every edge is an arc between two vertices;
- different edges have different sets of endpoints;
- the interior of an edge contains no vertex and no point of any other edge.

A plane graph (V, E) defines a graph G on V in a natural way. As long as no confusion can arise, we shall use the name G of this abstract graph also for the plane graph (V, E), or for the point set $V \cup \bigcup E$.

For every plane graph G, the set $\mathbb{R}^2 \setminus G$ is open; its regions are the *faces* of G. Since G is bounded — i.e., lies inside some sufficiently large disc D — exactly one of its faces is unbounded: the face that contains $\mathbb{R}^2 \setminus D$. This face is the *outer* face of G; the other faces are its *inner* faces. We denote the set of faces of G by $F(G)$. A face is *incident* to an edge e if e is a subset of its frontier. *Size* of a face f is the length of f's frontier, where the doubly-incident vertices are counted twice.

A *triangulation* is a graph where each face is of size 3. Formally:

$$\forall f \in F(G) : |f| = 3.$$

An *embedding* in the plane, or *planar embedding*, of an (abstract) graph G is an isomorphism between G and a plane graph H. The latter will be called a *drawing* of G. It is considered folklore that a planar graph can be embedded into a plane if and only if it can be embedded into a sphere. The latter has a neat property that we do not need to deal with the concept of outer face. We say that two planar embeddings are equivalent if they have a common unique embedding into sphere. If that is the case then cyclical sequence of neighbours for a face f is the same in any embedding of G.

Theorem 1.1 (Whitney 1932) *Any two planar embeddings of a 3-connected graph are equivalent.*

A *plane multigraph* is a pair $G = (V, E)$ of finite sets (of vertices and edges, respectively) satisfying the following conditions:

- $V \subseteq \mathbb{R}^2$;
- every edge is either an arc between two vertices or a polygon containing exactly one vertex (its endpoint);
- apart from its own endpoint(s), an edge contains no vertex and no point of any other edge.

Let $G = (V, E)$ and (V^*, E^*) be any two plane multigraphs, and put $F(G) =: F$ and $F((V^*, E^*)) =: F^*$. We call (V^*, E^*) a *plane dual* of G, and write $(V^*, E^*) =: G^*$, if there are bijections

$$\begin{array}{ccc} F \to V^* & E \to E^* & V \to F^* \\ f \mapsto v^*(f) & e \mapsto e^* & v \mapsto f^*(v) \end{array}$$

satisfying the following conditions:

- $v^*(f) \in f$ for all $f \in F$;
- $|e^* \cap G| = |e^* \cap e| = |e \cap G^*| = 1$ for all $e \in E$, and in each of e and e^* this point is an inner point of a straight line segment
- $v \in f^*(v)$ for all $v \in V$.

From the point of view of G^*, G is its *plane primal.* See Figure 1.1 for a visual example.

Dual graph G^* of a graph G that is not 3-connected may be a multigraph, because it may contain loops or parallel edges. Moreover, there may be more embeddings of G resulting in non-isomorphic dual graphs.

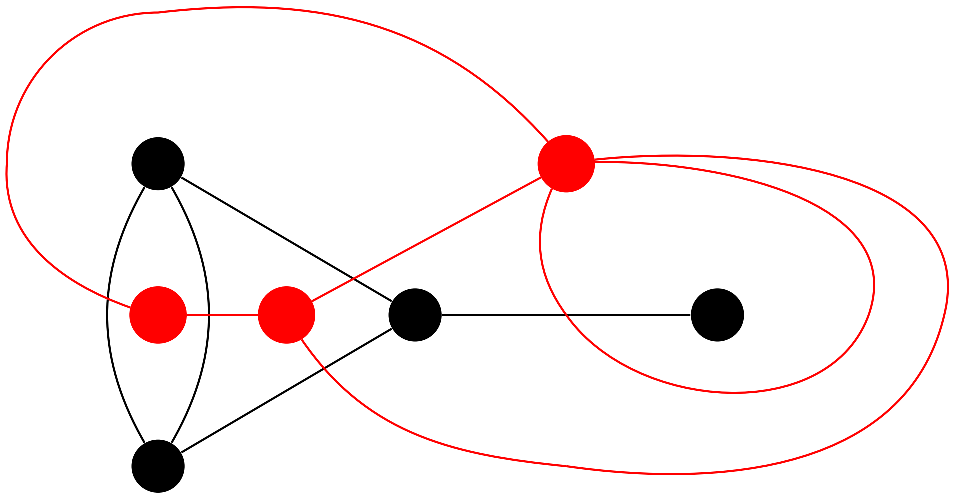


Figure 1.1: An example of a dual graph (red) of a multigraph (black).

Lemma 1.1 *The dual planar graph G^* of a 3-connected planar graph G is a planar graph.*

Proof. According to Whitney's theorem 1.1, all embeddings of G are equivalent which means its dual is well defined. Parallel edges in G^* would correspond to a 2-edge-cut in G and loops in G^* would correspond to bridges in G, i.e., edges belonging to an 1-edge-cut. □

Note that in this work, we are only interested in graphs that contain no parallel edges, and no *loops*, i.e., edges incident to a single vertex on both ends. When assuming multigraphs as graphs, we call such graphs *simple* graphs. As mentioned, we are strictly interested in simple graphs, so if we will need to speak of multigraphs, we will explicitly state so.

Theorem 1.2 (Euler's formula) *Let G be a connected planar graph, where n, m and f the numbers of vertices, edges and faces in G respectively, then*

$$n - m + f = 2.$$

This formula is considered folklore in the field of graph theory, but for the sake of completeness, we will provide a simple proof.

If G contains only one vertex, then $1 - 0 + 1 = 2$, therefore it holds. Let's now assume that it holds for all connected plane graphs on n vertices. Consider a plane graph G on $n + 1$ vertices. Pick an edge e of G. If e is incident to two distinct faces, then the graph G' obtained from G by removing e is connected, and it has n vertices, $m - 1$ edges and $f - 1$ faces. By induction, we have $n - (m - 1) + (f - 1) = 2$, and thus also $n - m + f = 2$, as desired. If e is incident to the same face twice, then by removing e we obtain two connected components, say G_1 and G_2, having n_1 and n_2 vertices, m_1

and m_2 edges, and f_1 and f_2 faces, respectively, where $n_1 + n_2 = n$, $m_1 + m_2 = m - 1$, and $f_1 + f_2 = f - 1$. By induction, $n_1 - m_1 + f_1 = 2$ and $n_2 - m_2 + f_2 = 2$. Therefore,

$$n-m+f = (n_1+n_2)-(m_1+m_2+1)+(f_1+f_2-1) = (n_1-m_1+f_1)+(n_2-m_2+f_2)-2 = 2.$$

As a simple consequence of this theorem, the following Lemma will come in handy.

Lemma 1.2 *Let G be a cubic planar graph, then G contains at least one face of size 5 or less.*

Proof. Let G be a cubic planar graph that does not contain a face of size 5 or smaller. Let $n = |V|$, $m = |E|$ and $f = |F|$. Since G is cubic, each vertex is incident to 3 vertices. Each edge is naturally incident to 2 vertices, therefore $n = \frac{2}{3}m$. Now, we can substitute than into Euler's formula.

$$\frac{2}{3}m - m + f = 2$$
$$\frac{1}{3}m + 2 = f$$

Additionally, we know that each face is incident to at least 6 edges. Each edge is incident to 2 faces, therefore $f \leq \frac{1}{3}m$. As a result, we get the following equation

$$\frac{1}{3}m + 2 \leq \frac{1}{3}m$$
$$2 \leq 0$$

which is obviously a contradiction, therefore Lemma 1.2 holds. □

1.3 Colouring

When speaking of colouring we usually want to assign colours to vertices (or edges) of a graph so that no two adjacent vertices or edges are of the same colour. The most obvious question to ask is how many colours are sufficient to colour a given graph. We will once again follow Diestel's way to present these notions and concepts formally.

A *vertex colouring* of a graph $G = (V, E)$ is a map $c : V \to S$ such that $c(v) \neq c(w)$ whenever v and w are adjacent. The elements of the set S are called the available *colours*. All that interests us about S is its size: typically, we shall be asking for the smallest integer k such that G has a *k-colouring*, a vertex colouring $c : V \to \{1, \ldots, k\}$. This k is the *(vertex-) chromatic number* of G, it is denoted by $\chi(G)$. A graph G with $\chi(G) = k$ is called *k-chromatic*; if $\chi(G) \leq k$, we call G *k-colourable*. Note that a k-colouring is nothing but a vertex partition into k independent sets, now called *colour classes*.

Theorem 1.3 (Brooks 1941) *Let G be a connected graph. If G is neither complete nor an odd cycle, then*

$$\chi(G) \leq \Delta(G).$$

Theorem 1.4 (Four Colour Theorem) *Every planar graph is 4-colourable.*

We decided to quote some remarks of Diestel to this theorem as it is the subject that this work is revolving around. The proof technique mentioned in these remarks is still polarising and it will to some extend resemble the proof techniques used in the latter parts of this work.

The four colour problem, whether every map can be coloured with four colours so that adjacent countries are shown in different colours, was raised by a certain Francis Guthrie in 1852. He put the question to his brother Frederick, who was then a mathematics undergraduate in Cambridge. The problem was first brought to the attention of a wider public when Cayley presented it to the London Mathematical Society in 1878. A year later, Kempe published an incorrect proof, which was in 1890 modified by Heawood into a proof of the five colour theorem. In 1880, Tait announced 'further proofs' of the four colour conjecture, which never materialised. The first generally accepted proof of the four colour theorem was published by Appel and Haken in 1977. The proof builds on ideas that can be traced back as far as Kempe's paper, and were developed largely by Birkhoff and Heesch. Very roughly, the proof sets out first to show that every plane triangulation must contain at least one of 1482 certain 'unavoidable configurations'. In a second step, a computer is used to show that each of those configurations is 'reducible', i.e., that any plane triangulation containing such a configuration can be 4-coloured by piecing together 4-colourings of smaller plane triangulations. Taken together, these two steps amount to an inductive proof that all plane triangulations, and hence all planar graphs, can be 4- coloured.

Appel & Haken's proof has not been immune to criticism, not only because of their use of a computer. The authors responded with a 741 page long algorithmic version of their proof, which addresses the various criticisms and corrects a number of errors (e.g. by adding more configurations to the 'unavoidable' list): K. Appel & W. Haken[1]. A much shorter proof, which is based on the same ideas (and, in particular, uses a computer in the same way) but can be more readily verified both in its verbal and its computer part, has been given by N. Robertson, D. Sanders, P.D. Seymour & R. Thomas[5].

1.4 Signed graphs

In 1982, Thomas Zaslavsky [6] introduced a concept of colouring on signed graphs. This concept was further explored by Máčajová, Raspaud and Škoviera[4]. To understand these and further results we need to introduce additional notation. For that purpose, we will paraphrase Zaslavsky's definitions.

A *signed graph* G_σ consist of an unsigned graph $H = (V, E)$ and a mapping $\sigma : E(H) \to \{\pm 1\}$, the *signature*. We denote such graph as $G = (H, \sigma)$. In this case, both $V(G)$ and $V(H)$ address the same vertex set. Similarly for edges and eventually faces if H is planar.

Switching is and operation on signed graphs defined as follows. Suppose $G = (H, \sigma)$ is a signed graph and $\omega : V(G) \to \{\pm 1\}$ is any sign function. Switching G by ω means forming the switched graph $G^\omega = (H, \sigma^\omega)$, whose underlying graph is the same but whose sign function is defined on an edge $e = uv$ by

$$\sigma^\omega(e) = \omega(u)\sigma(e)\omega(v).$$

In other words, the sign function ω partitions the vertex set into two parts. The sign of an edge is switched if and only if its endvertices belong to different parts.

Balance. Any path $P = e_1 e_2 \dots e_k$ has a *value*, obtained by multiplying the signs of its edges:

$$\sigma(P) = \sigma(e_1)\sigma(e_2)\dots\sigma(e_k).$$

A cycle whose value is positive is called *balanced.* An edge set is called balanced when every cycle in it is balanced.

Assume (G, σ) is a signed graph and k is a positive integer. Let

$$N_k = \begin{cases} \{\pm q, \pm(q-1), \dots, \pm 1\}, & \text{if } k = 2q \text{ is even,} \\ \{\pm q, \pm(q-1), \dots, \pm 1, 0\}, & \text{if } k = 2q+1 \text{ is odd.} \end{cases}$$

A *proper k-colouring* of (G, σ) is a mapping $f : V(G) \to N_k$ such that for any edge $e = xy$ of G $f(x) \neq \sigma(e) \cdot f(y)$.

In other words, two vertices joined by a positive edge are not allowed to have identical colours, whereas two vertices joined by a negative edge are not allowed to have opposite colours.

Máčajová, Raspaud and Škoviera [4] have proven an array of theorems related to signed graphs and their colouring. We would like to highlight that Brooks theorem also holds for a signed planar graph G, if we give put stronger restriction on G.

Theorem 1.5 (Brooks theorem for signed graphs) *[4] Let G be a simple connected signed graph. If G is not a balanced complete graph, a balanced odd cycle, or an unbalanced even cycle, then*

$$\chi(G) \leq \Delta(G).$$

At the end of their paper they stated a conjecture that is the origin for this work. The conjecture is as follows.

Conjecture 1.1 *[4] Every simple signed planar graph G has $\chi(G) \leq 4$.*

This conjecture got disproved by Kardoš and Narboni [3], when they found a counterexample. Its structure will be essential to our work and we will analyse it in a latter section of this work. Before that we need to look at signed non-4-colourable graphs in general.

1.5 Non-4-colourable signed graphs

When constructing their counter-example, Kardoš and Narboni [3] translated the problem of 4-colouring of a signed graph to a different problem on its dual. In this section we want to describe the translation in detail.

First and foremost we need to define what the dual of a signed planar graph is. Let $G_\sigma = (G, \sigma)$. Let H be the dual graph of G, and let σ_v be a labelling function on $V(H)$ defined as

$$\sigma_v(v^*) = \prod_{e \in N_{v^*}} \sigma(e) \qquad \text{for } v^* \in V(H),$$

where N_{v^*} is the set of all edges incident to the face corresponding to v^* in the primal graph G_σ. $H_{\sigma_v} = (H, \sigma_v)$ is then called the *signed dual plane graph* of G_σ. Note that H_{σ_v} is no longer a signed planar graph, but a *vertex-signed planar graph.*

In other words, the underlying graph H of H_{σ_v} is the dual of G and the labelling works as follows. The label of a vertex v^* in $V(H_{\sigma_v})$ is negative if its corresponding face f in $F(G_\sigma)$ is incident to an odd number of negative edges. Otherwise, v^* is positive.

Lemma 1.3 *Vertex-signed planar graph H_{σ_v} obtained as a dual of a signed planar graph always contains an even number of negative edges.*

Proof. Let's observe the relationship between σ and σ_v. If G_σ contains no negative edges, i.e. $\forall e \in E(G) : \sigma(e) = 1$, then each face is incident to 0 negative edges. 0 is an even number, therefore each vertex in H_{σ_v} is positive. Assume we have two edge-labelling functions σ_1 and σ_2, where $\sigma_1(e) = \sigma_2(e)$ for all edges from $E(G)$ but one, denoted as $e_{\neq}$. Without loss of generality, let's say that $\sigma_2(e_{\neq}) = -1$. If $e_{\neq}$ is negative, it effects polarity of precisely two vertices v_1^*, v_2^* in H_{σ_v}. If $\sigma_1(v_1^*) = \sigma_1(v_2^*)$ that means that $\sigma_2(v_1^*) = \sigma_2(v_2^*)$ as well so the parity of negative vertices is not going to change, as it either increases or decreases by 2. If $\sigma_1(v_1^*) \neq \sigma_1(v_2^*)$, then $\sigma_2(v_1^*) \neq \sigma_2(v_2^*)$, which means that one vertex is going to become positive and the other negative which will not affect the parity as well. As the parity does not change in neither case we can clearly see that H_{σ_v} will always contain an even number of negative edges. □

A *valid* vertex-signed plane graph is a graph that contains an even number of negative vertices. When speaking of a vertex-signed cubic planar graph, we will always assume it is valid.

Lemma 1.4 *For a fixed signed plane graph G, the vertex-signature of it vertex-signed dual plane graph G^* is invariant to the switching operation on G.*

Proof. Let f^* be a vertex in G^* corresponding to a face f in G. If we switch a vertex v incident to f in G, polarities of precisely two edges incident to f are going to flip, which does not change the parity of negative edges incident to f. For that reason, G^* is invariant to switching operation on G. □

Having described the setting, we will continue with the problem itself. Edge labelling is a mapping $E(H) \to L$, where L is a set of labels. In this context, most often we will use $L = \{0, a, b\}$. Let $d_x(v)$ denote the number of edges labelled x adjacent to v and $d_H(v)$ the degree of v in H.

Definition 1.1 *[3] Let H be a 3-connected planar graph with an even number of negative vertices and let c be a $\{0, a, b\}$-edge-labelling of H. The labelling c is a* weak signed edge-labelling *of H if*

- *$d_0(v) \equiv d_H(v)$ (mod 2), and*
- *$d_a(v) \equiv d_b(v) \equiv d_H(v)$ (mod 2) if v is a positive vertex, or*
- *$d_a(v) \equiv d_b(v) \equiv d_H(v) + 1$ (mod 2) if v is a negative vertex*

Definition 1.2 *[3] Let H be a 3-connected planar graph with an even number of negative vertices. A $\{0, a, b\}$-edge-labelling c of H is a strong signed edge-labelling if*

- *c is a weak signed edge-labelling of H, and*
- *$d_0(v) < d_H(v)$ for every odd-degree vertex v of H.*

Theorem 1.6 (Kardoš and Narboni [3]) *A signed planar graph G is signed 4-colourable if and only if its dual H does not admit a weak edge-labelling.*

In their work, they used the strong edge-labelling to enforce some further restrictions so they could construct their counter-example. They showed that the strong edge-labelling is essentially equivalent to constructability of a *consistent 2-factor*, which is a 2-factor, where each cycle must contain an even number of positive edges. As we will not be looking for a graph that does not admit a strong edge-labelling, we will introduce a *semi-2-factor* which is equivalent to the weak edge-labelling. In the next chapter we will discuss and analyse these definitions in more detail.

Chapter 2

Semi-2-factor in the dual

In this chapter we will discuss the process of searching for the smallest non-4-colourable signed planar graph. In the first section, we will analyse the definitions by Kardoš and Narboni [3].

2.1 Problem translation

Lemma 2.1 *If there is a non-4-colourable signed plane graph on n vertices, then there is a non-4-colourable signed plane triangulation on n vertices.*

Proof. Consider there is a signed non-4-colourable planar graph G which is not a triangulation. Then it contains a face f which is greater in size than 3. In such face, there are 2 vertices v, u that are not connected by an edge, implying that those two vertices are non-adjacent in G (otherwise there would be an edge crossing in G). As each edge is an extra restriction on colouring, then $G \cup uv$ is signed non-4-colourable as well. Through iteration of this process, until G is a triangulation, we see that the Lemma holds. $\square$

We will not look for the non-4-colourable graph itself, but a dual which does not admit a weak signed edge-labelling. When speaking of the dual of a triangulation, we are dealing with a cubic plane graph. It is easy to see as all faces in a triangulation are of size 3, hence all vertices in the dual must be of degree 3 as well. A graph with all degrees of size 3 is by definition 3-regular, i.e., cubic. We will discuss the impacts of definition 1.1 on 3-connected cubic planar graphs below.

Lemma 2.2 *Let G be a signed cubic planar graph. Then a $\{0, a, b\}$-edge-labelling of G is a weak signed edge labelling if and only if*

- *each positive vertex is incident to edges labelled with three different labels,*
- *each negative vertex is either incident to three edges labelled 0 or one edge labelled 0 and two edges labelled with the same label a or b.*

Proof. We prove both implications at the same time.

The first condition of definition 1.1 claims that the number of edges labelled 0 incident to a fixed vertex is always odd, in our case 1 or 3.

The second condition deals with positive vertices and claims that each positive vertex must be incident to an odd number of edges labelled 0, a or b. In our case the only option is that each positive vertex is incident to exactly one edge of each label.

The last condition deals with negative vertices and claims that each vertex must be incident to an even number of edges labelled a or b, in our case 0 or 2. □

Lemma 2.3 *Let G be a vertex-signed plane triangulation with an even number of negative vertices. Let φ be a weak edge labelling of G. Then the set of edges labelled a or b induces a collection of disjoint cycles, each containing an even number of positive vertices.*

Proof. It follows directly from the characterisation of weak edge labellings in definition 1.1 that at each vertex, (positive or negative) the number of edges labelled a or b is either 2 or 0, and so the set of edges labelled a or b induces a collection of disjoint cycles.

Moreover, alongside each cycle the label of the traversed edge remains invariant if we traverse an negative vertex and switches to the opposite (from a to b or vice versa) if we traverse a positive vertex. If we traverse the entire cycle so that we stop at the edge we started at, the number of switches from a to b and back must have been an even number (we must switch back to the original starting label). Since a switch is equivalent to a traversal of a positive vertex, the number of positive vertices of a cycle must be even. □

Kardoš and Narboni [3] defined a *consistent* 2*-factor* of a vertex-signed planar graph H as a set of cycles covering all positive vertices, where each cycle consists of an even number of positive vertices. This concept is an alternation of the usual interpretation of a 2*-factor* as it adds a restriction on each cycle to contain an even number of positive vertices. This however does not allow negative vertices to be of degree 0, i.e., to have all 3 of their adjacent edges labelled 0. As explained above, we need to allow some negative vertices to be of degree 0.

Definition 2.1 (Semi-2-factor) *Semi-2-factor is a collection of disjoint cycles covering all positive vertices of G, such that every cycle in the collection contains an even number of positive vertices.*

Given the explanation above, it should be obvious that a vertex-signed planar graph G admits a weak edge-labelling if and only if it admits a semi-2-factor. Therefore, we will not be looking for a graph that does not admit any concrete weak edge-labelling, but a graph that does not admit a semi-2-factor.

2.2 Reducible Configurations

Let $E(X,Y)$ be an edge cut of G. There are two types of edges uv in G:

1. both u and v are in X or both of them are in Y,
2. u is in X, but v is not (or vice versa).

We will call the second type of edges *open edges* or *semi-edges*, when considering X only, disregarding Y. Simply put, those are the edges connecting X with the rest of G. These are exactly the edges of $\partial(X)$.

Let $\mathcal{G}$ be a graph class and $\mathcal{P} \subseteq \mathcal{G}$ be a graph property. H^* is a *reductor* of H if for all $G \in \mathcal{G}$ such that H is a subgraph of G, the graph G^* obtained by replacing H by H^* is in $\mathcal{G}$, moreover, whenever $G^* \in \mathcal{P}$, then also $G \in \mathcal{P}$. A subgraph H of $G \in \mathcal{G}$ is a *reducible configuration* if H has a reductor H^*.

Obviously, we want to introduce reducible configurations with respect to existence of a semi-2-factor. The importance of these reducible configurations is simple. Suppose, we have a vertex-signed planar graph G that does not admit a semi-2-factor, but contains a reducible configuration. On one hand G is a valid counter-example, but by the definition of the reducible configuration, there exists a smaller graph $G_<$ which is a counter-example as well. That is why we will be strictly interested in graphs containing no reducible configurations. Below, we list all reducible configurations known prior to this work. This list will be expanded in future parts of this work.

Lemma 2.4 *A triangle is reducible into a single vertex.*

Proof. The idea of the proof is very simple. We want to prove that for every possible signature of H, for each way a semi-2-factor can cover the semi-edges of a the corresponding reductor H^* (a single vertex) there is a way the semi-2-factor can be extended to a semi-2-factor in H. See Figures 2.1 and 2.2 for illustration. □

Lemma 2.5 *Two adjacent faces of size 4 are reducible into a single edge.*

Proof. Similarly as for Lemma 2.4, we want to prove that for every possible signature of the configuration, there is a choice of a signature of the reductor such that for every possible way how a semi-2-factor can traverse the reductor it is possible to extend it into a semi-2-factor in the configuration. In this case there are more cases to consider, therefore we will only illustrate one of them (see Figure 2.4); we leave the rest for the reader. □

As mentioned before, this list will grow in the latter parts of the work.

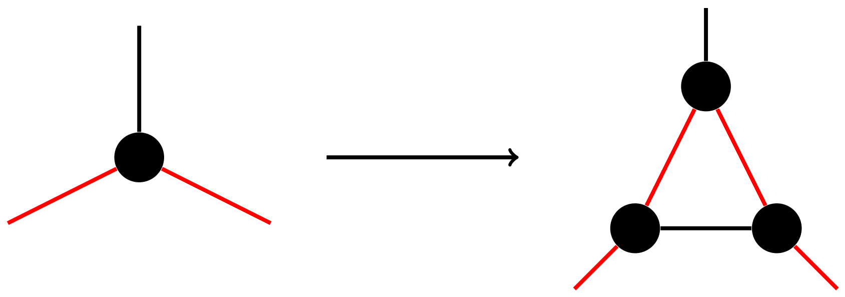


Figure 2.1: If H is a positive triangle (i.e., it contains an odd number of positive vertices), then H^* is a positive vertex. All positive vertices of H can be covered by a semi-2-factor, hence H^* is a valid reductor of H.

2.3 Tripoles

Let (X, Y) be an edge-cut in G. If $|E(X, Y)| = k$, then X (or Y) is a k-pole. In this section we are interested in properties of 3-poles, referred to as *tripoles*. Note that we also assume the edges in $E(X, Y)$ to be part of the k-pole, but not containing edges from the other set. Hence, those edges are only considered as semi-edges. Moreover, $\partial(T)$ will denote the set of semi-edges of a tripole T, whose size is obviously always equal to 3.

We will research tripoles contained in the potential smallest counter-example. We want to know their structure and properties. When building their counter-example, Kardoš and Narboni[3] used concrete *gadgets*, i.e., building block that are tripoles.

Lemma 2.6 *Any tripole in a 3-connected cubic planar graph is of odd degree.*

Proof. Let T be a non-trivial tripole in G. Let's now remove its semi-edges s_1, s_2, s_3. What we are left with is a graph with 3 vertices of degree 2. The other vertices are of degree 3 as we have not removed any edge incident to them. We know there must be an even number k of vertices of degree 3. Hence, $k+3$ gives an odd number, which proves that T must contain an odd number of vertices. Note that the removal of semi-edges will always yield 3 vertices of degree 2. If that were not the case, then there would be a vertex v incident to 2 semi-edges s_1 and s_2. Then G would contain a 2-edge-cut $\{s_3, e_v\}$, where e_v is the edge incident to v other than s_1, s_2. □

Let's now define the polarity or a signature of a tripole T, denoted as $\sigma(T)$. If T contains an odd number of positive vertices, then $\sigma(T)$ is positive. Otherwise $\sigma(T)$ is negative. This should correspond to how the tripole affects the graph containing it. If the tripole is positive, then any semi-2-factor covering T will contain an odd number of positive vertices of T.

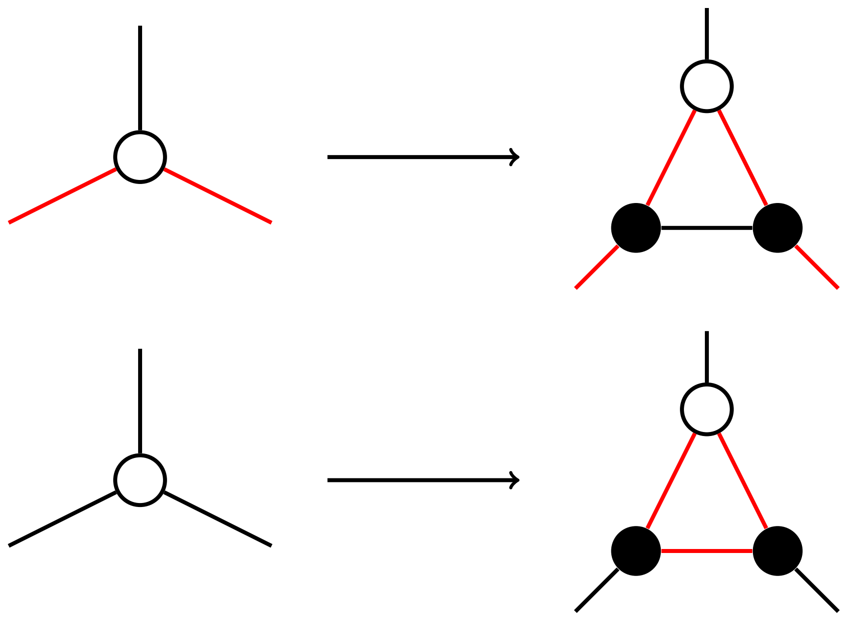


Figure 2.2: If H is a negative triangle (i.e., it contains an even number of positive vertices), then H^* is negative vertex, say x. If x is covered by a cycle of a semi-2-factor, then all (positive) vertices of H can be covered by the corresponding cycle in a semi-2-factor. If x is not covered by any cycle of a semi-2-factor, then a new cycle of length 3, covering an even number of positive vertices, can be introduced. Hence H^* is a valid reductor of H.

We are now going to analyse how precisely a semi-2-factor can traverse a tripole. The following observation is straightforward.

Lemma 2.7 *Let X be a positive tripole in a vertex-signed cubic plane graph G and let C be a semi-2-factor. Then $|C \cap \partial(X)|$ is even.*

Therefore T behaves as a positive vertex when it comes to a semi-2-factor composition, which means precisely 2 of T's semi-edges must be included in any semi-2-factor. On the other hand, if T is negative then any semi-2-factor must contain an even number of positive vertices. Therefore, any semi-2-factor containing any two semi-edges of T must contain an even number of positive vertices in T. Since T contains an even number of positive vertices that opens up a possibility for a semi-2-factor to not use any of the semi-edges. Therefore, such tripole may act as a negative vertex.

Let T be a tripole and $p = \{s_i, s_j\}$, $i \neq j$ be a pair of its semi-edges. We say that p is *coverable* if there exists a semi-2-factor in T containing both semi-edges of p.

Let's now classify non-trivial tripoles based on how many pairs of its semi-edges are coverable. Let T be a tripole and s_1, s_2, s_3 be its semi-edges. Based on the structure

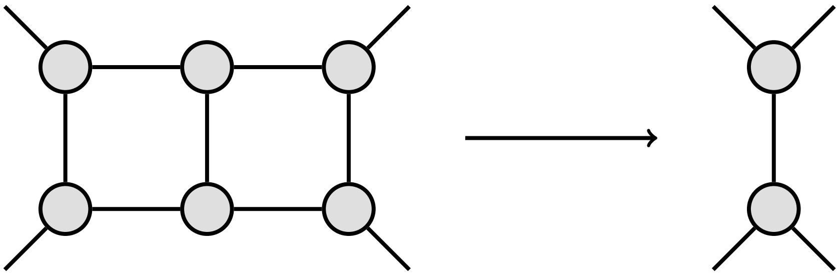


Figure 2.3: A non-signed example of the reducible configuration (on the left) and its reductor (on the right).

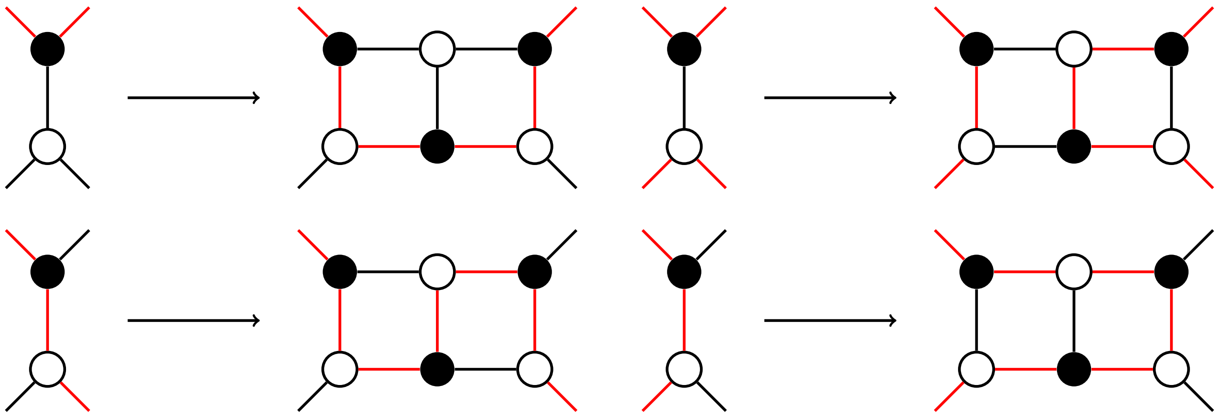
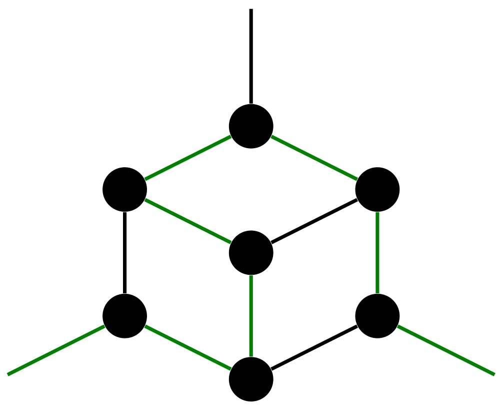
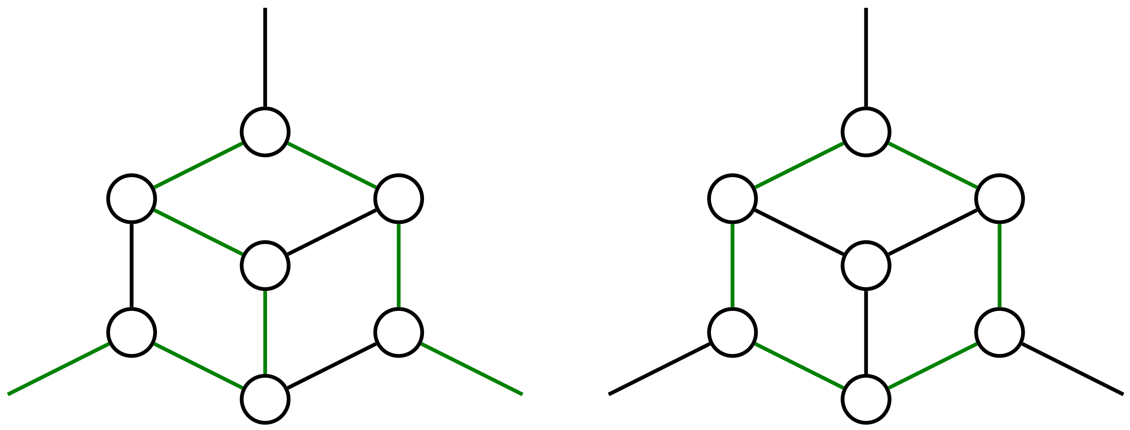


Figure 2.4: An example of a signature for the reducible configuration an its corresponding reductor signature, together with all possibilities that a semi-2-factor can cover the reductor, extended into the reducible configuration.

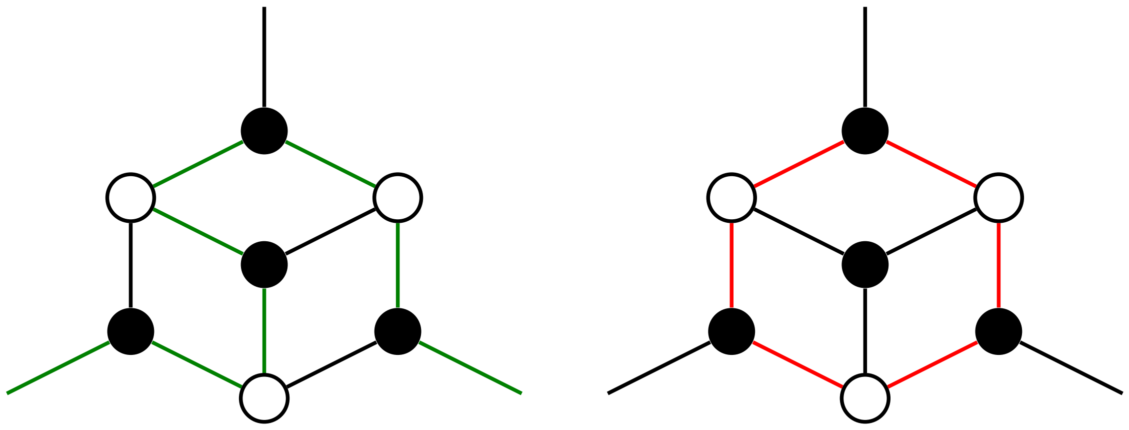
of T, T may not admit a semi-2-factor containing a concrete pair of its semi-edges. In total, there are 3 pairs of semi-edges in T. Therefore, there are 4 ways as to how many pairs are coverable. Additionally, we are interested in the polarity of T. For positive tripoles, these are the only 4 options. However, for negative tripoles, there are 8 options. For each option of semi-edge pair count, there is an additional option of whether T admits a semi-2-factor containing no semi-edge of T. We will determine a type for each tripole T. Firstly, we will refer to it as P if T is positive or N if T is negative. Additionally, we will add a lower index containing an integer from the range of 0 to 3. Moreover, if T is negative, we may add an optional upper index $*$ if T admits a semi-2-factor containing no semi-edge of T. To make it a little more clear, let's look at a few examples. We will always refer to the tripole in the picture as a tripole T.



As seen in the picture above, T is positive, because all of its 7 vertices are positive. All 3 semi-edge pairs in T are coverable due to symmetry. Therefore, T is of type P_3.



As seen in this picture, T is negative, because none of its 7 vertices is positive. All 3 semi-edge pairs are coverable due to symmetry. However, as seen in the right picture, T admits a semi-2-factor covering no pair of its semi-edges. Therefore, T is of type N_3^*



The final example, seen in this picture, is an example of the type N_3 tripole. It is a negative tripole because it contains an even number of positive vertices. Moreover, all of its semi-edge pairs are coverable in similar fashion as in the previous examples. However, it does not admit a semi-2-factor not containing any of the semi-edges. It is easy to prove. If we do not want the semi-2-factor to contain any of the semi-edges, then all three vertices adjacent to the semi-edges must have the other 2 edges included in the semi-2-factor, because they are positive and we must cover them. Therefore, as seen on the right picture, we inevitably create a cycle that not only contains 3 positive vertices, but there is no way to cover the central vertex as well. That proves that T is of type N_3.

This tripole was essential for Kardoš and Narboni in their construction of the counter-example. It is not difficult to verify that this, indeed, is the smallest tripole of its type. We leave that as an exercise for an enthusiastic reader. We may refer to this tripole as a *cube* in this work.

Let $s_1 \in T_1$ and $s_2 \in T_2$ be two semi-edges. Let v_1 and v_2 be their respective endvertices. Then, if we *connect* s_1 and s_2, we end up having a graph G, where $V(G) = V(T_1) \cup V(T_1)$, $E(G) = E(T_1) \cup E(T_2) \cup \{v_1v_2\}$ and $S(G) = S(T_1) \cup S(T_2) \setminus \{s_1, s_2\}$. Observe that G no longer is a tripole, as $|S(G)| = 4$. However, from the point of view of a tripole T, it remains a tripole as the set of its semi-edges $\partial(T)$ has not changed, even though we know that it is an actual edge in the bigger picture. For better imagination, one might think of a tripole, as a building block of the graph.

Let T_a and T_b be tripoles with semi-edges $s_{1,a}, s_{2,a}, s_{3,a} \in T_a$ and $s_{1,b}, s_{2,b}, s_{3,b} \in T_b$. If we connect all 3 pairs $(s_{i,a}, s_{i,b})$ to each other, we say that T_a and T_b are now *fully connected*. Note that the indexation of semi-edges belonging to a tripole is not fixed, therefore this definition admits any combination of semi-edge pairs. The only condition is that each semi-edge in one tripole has a paired semi-edge in the other one.

Lemma 2.8 *Let G be a 3-connected vertex-signed cubic planar graph. Additionally let T_1 and T_2 be tripoles in G, i.e., $V(T_1) \cup V(T_2) \subseteq V(G)$. Then T_1 and T_2 are not* *connected or connected by a single edge. If T_1 or T_2 is non-trivial then they can be fully connected as well.*

Proof. To prove the first two option, i.e., when they are not connected or connected by a single edge, let's assume T_1 and T_2 to be just vertices. Clearly, they can be not connected, or connected by a single edge, if those vertices are incident. To prove the last point, let's assume G to be any vertex-signed cubic planar graph and T_1 be a tripole consisting of a single vertex v. Then clearly, T and $G \setminus v$ is an example of two fully connected tripoles. Now, let's discuss the option where we T_1 and T_2 were connected by 2 edges. If they are connected by a single edge, then there are 4 semi-edges in $T_1 \cup T_2$. If we connect on more pair of semi-edges, the resulting graph will have 2 semi-edges. If it has two semi-edges and they are not connected to each other that means that $|V(G \setminus (T_1 \cup T_2))| > 0$. But that also implies that $T_1 \cup T_2$ is connected to the rest of the graph with precisely 2 edges, which means that G would not be 3-connected. □

We say that two tripoles are *incompatible* if there is a way to fully connect them, so that the resulting graph does not admit a semi-2-factor. Let's take a look at the combination of tripoles that Kardoš and Narboni used.

One of the two tripoles T_1 was of type P_2. If a tripole is of type P_2, it means that 2 pairs of its semi-edges can be covered by a semi-2-factor. In other words, T_1 admits no semi-2-factor containing the third pair of semi-edges. Since there are only 3 semi-edges in a tripole, that implies that there exists a semi-edge $s_{unavoidable}$ that is in any semi-2-factor that T_1 admits. We call such edge or semi-edge *unavoidable.* The second tripole T_2 was of type P_1. If T_2 admits only semi-2-factors that cover only on pair of semi-edges, it means that there is a semi-edge $s_{untouchable}$ in T_2 that is not contained in any semi-2-factor that T_2 admits. Similarly, we call such edge or semi-edge *untouchable.* Now, if we fully connect T_1 and T_2 so that $s_{unavoidable}$ is connected to $s_{untouchable}$, we get a graph that does not admit a semi-2-factor. This can be generalised into following lemma.

Lemma 2.9 *Let G be a counterexample (a 3-connected cubic planar graph that does not admit a semi-2-factor). Let (X, Y) be a 3-edge-cut. Then the types of the tripoles X and Y are incompatible.*

Proof. Assume that X and Y are compatible. That means there is a way to fully connect them so that G admits a semi-2-factor. That is a contradiction with G being a counterexample. □

Note that the polarity of X and Y must be the same, because otherwise G would not be a valid dual vertex-signed planar graph. Without further analysis which would be mostly equivalent to the one prior to this lemma, let's list the possible combinations of X and Y's types. For the positive tripoles, we already mentioned the combination of P_2 and P_1, however P_3 and P_0 works as well. For the negative tripoles, let's first do a following observation. If X and Y were both tripoles that admit a semi-2-factor not containing any of its semi-edges (types marked with asterisk) then G would admit a semi-2-factor. Therefore there are 6 combinations of tripole types found in the counterexample G: N_3^* and N_0; N_3 and N_0^*; N_3 and N_0; N_2^* and N_1; N_2 and N_1^*; and finally N_2 and N_1.

The goal of this work is to find the smallest tripole of each type. Prior to this work, we already knew of 3 that are the smallest. Moreover, we knew of the existence of 4 other types.

The smallest tripole of type P_3 is a single positive vertex v. A positive vertex v allows any pair of its edges to be in a semi-2-factor and it is a positive tripole.

Similarly, the smallest tripole of type N_3^* is a single negative vertex v. It admits any pair of its edges to be in a semi-2-factor and it is a negative tripole. Additionally, v admits a semi-2-factor that uses no edges adjacent to v, i.e., a semi-2-factor avoiding v.

The last known smallest tripole is a tripole of type N_3. It is the aforementioned cube, which we proved to be of type N_3.

Since we already mentioned what tripole types Kardoš and Narboni used, we know of the existence of tripoles of types P_2 and P_1. However, we do not know if those used are the smallest possible. Lastly, we know of tripoles of type P_0 and N_0, as those are tripoles gathered by a removal of a positive or negative vertex, respectively, from the known counter-example.

Let's discuss the tripole of type P_2 used by Kardoš and Narboni. When William Thomas Tutte published the first counterexample to Tait's conjecture that every planar cubic graph has a Hamilton cycle, he used a tripole now called *Tutte's fragment.* It has a special property, that each Hamilton cycle must contain one of its semi-edges e. The tripole consist of 15 vertices and is the smallest tripole with this property. Kardoš and Narboni used this tripole as a base for their tripole of type P_2. They found a signature of this Tutte's fragment and substituted two of its vertices by a negative irreducible cube (a tripole of type N_3). Such tripole has 27 vertices and they proved that it is of type P_2 where e is unavoidable.

Similarly for the tripole of type N_1 they used a tripole from the counterexample of Tait's conjecture. The very first counterexample to that conjecture was a K_4 where three of its vertices were substituted by a Tutte's fragment so that the only non-substituted vertex is incident to three edges that must be in every Hamilton cycle,

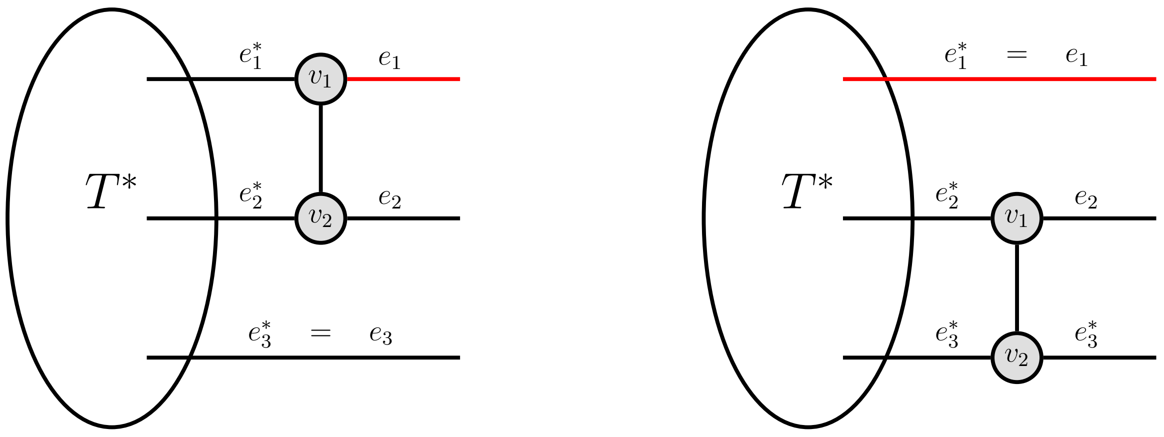


Figure 2.5: Two cases how a tripole T without triangles can be obtained from a graph G with one triangle. In both cases, the unavoidable edge is marked with red colour. In the first picture, the unavoidable edge $e1$ is incident to the triangle in G. In the second picture, the unavoidable edge e_1 is not incident to the triangle in G but one of its endvertices is.

which is a contradiction with the graph being Hamiltonian. Kardoš and Narboni used the same structure, however they substituted the modified signed version of Tutte's fragment into K_4 resulting in a graph not admitting a consistent 2-factor. The tripole of type P_1 is then a tripole obtained from an edge cut (X, Y) in the counterexample where X is the modified signed Tutte's fragment and Y the rest. The size of Y is 55. Kardoš and Narboni found a smaller counterexample where the tripole of type P_1 is of size 47. That is the smallest know tripole of type P_1.

2.4 Derivation of tripoles

Let G be an counter-example containing an edge cut splitting G into tripoles T and T^* of types N_2 (or N_1) and N_1 respectively. We know that G contained no triangles, however if we complete the tripoles to a proper graphs (by an addition of a single vertex incident to all the semi-edges), such graphs may contain a triangle.

Let's focus on a tripole T of type N_2 or N_2^* obtained through removal of a single vertex from a graph G. There are precisely two situations how the triangle could be positioned in G in relation to the unavoidable edge so that T contains no triangles. For illustration, see Figure 2.5.

Let's first analyse the first case in the Figure 2.5, where the unavoidable edge is incident to the triangle of G. We want to prove the following lemma.

Lemma 2.10 *Let T be tripole of type N_2^* or N_2, with the structure as seen in Figure 2.5 on the left, then T^* must be of type P_2, N_2^*, N_2 or something even more restrictive.*

Proof. Let T^* admit a semi-2-factor containing edge e_1^* and e_3^*. Then such semi-2-factor can be extended to a semi-2-factor in T containing edges e_3, e_1^*, v_1, v_2 and e_2, which is in contradiction with T being a tripole of type N_2^*. □

Lemma 2.11 *Let T be a tripole of type N_2^*, N_2, with the structure as seen in Figure 2.5 on the left. Then v_1 must be positive.*

Proof. Let v_1 be negative. According to the lemma 2.10, T^* must not admit a semi-2-factor containing edges (semi-edges) e_1^* and e_3^*. However, a combination e_2^*, e_3^* is perfectly viable. Such semi-2-factor however, can be extended to a semi-2-factor containing edges e_3, e_2^* and e_2 in T, which is in contradiction with T being of type N_2^*.
□

Now we have a choice at v_2. Let's cover both cases with lemmas.

Lemma 2.12 *Let T be a tripole with a structure as seen in Figure 2.5 on the left; moreover, conforming to lemmas 2.10 and 2.11. Then if v_2 is negative, T is of type N_2^*.*

Proof. Note that T^* must be a positive tripole, so that T is negative. Let's now assume there exists a semi-2-factor in T containing semi-edges e_2 and $_3$. As we know from lemma 2.10, T^* does not admit a semi-2-factor containing edges e_1^* and e_3^*. As e_3^* is already covered, then e_2^* must be covered as well. Then however there are two edges incident to v_2 contained in a semi-2-factor, which implies that the vertex v_1, which is positive according to lemma 2.11, remained uncovered. That proves that T does not admit a semi-2-factor containing edges e_2 and e_3, rendering e_1 an unavoidable edge. Moreover, T^* admits a semi-2-factor containing vertices e_1^* and e_2^*. Through an extension of the semi-2-factor with the edge v_1v_2, we see that T admits a semi-2-factor not using any of its semi-edges, hence T is of type N_2^*. □

Lemma 2.13 *Let T be a tripole with a structure as seen in Figure 2.5 on the left; moreover, conforming to lemmas 2.10 and 2.11. Then if v_2 is positive, T is of type N_2^*.*

Proof. The reasoning is equivalent to the previous lemma. □

Let's now discuss the consequences of lemmas 2.12 and 2.13. The second one says that a tripole of type N_2^* or N_2 can be transformed to N_2^*. When speaking of the tripoles in the smallest counter example, a transformation from N_2^* to N_2^* is reducible, as the nested tripoles has the same properties and is smaller. When it comes to the transformation from N_2 to N_2^*, that is highly redundant as N_2 is more restrictive than N_2^* and the tripole of type N_2 is even smaller in this case. However, the first lemma might be very useful as we transformed a positive tripole into a negative one. Note that the transition from a negative one to a positive one works too, but we do not find it relevant enough.

Let's now analyse the second case in the Figure 2.5.

Lemma 2.14 *Let T be tripole of type N_2^* or N_2, with the structure as seen in Figure 2.5 on the right, then T^* must be of type P_2, N_2^*, N_2 or something even more restrictive.*

Proof. Assume T^* admits a semi-2-factor containing e_2^* and e_3^*. Then such two factor could be extended to a two factor in T containing edges e_2^*, e_2, e_3^* and e_3. That is however a contradiction with T being of type N_2 or N_2^*. □

Lemma 2.15 *Let T be a tripole with the structure as seen in Figure 2.5 on the left. If T^* is positive and one of the two vertices v_1 and v_2 is positive, then T is of type N_2.*

Proof. According to the lemma 2.14, we know that there is no semi-2-factor in T containing e_2 and e_3, rendering edge e_1 unavoidable. Moreover, all vertices of T must be covered therefore the semi-2-factor covers an even number of vertices in T as T^* was a negative tripole. Furthermore, T does not admit a semi-2-factor containing none of its semi-edges. Assume it does, then T^* would have to admit a semi-2-factor containing edges v_1v_2, e_2^* and e_3^* which we know is not possible. That proves that T is of type N_2. □

In other words, lemma 2.15 claims that if we have a tripole of type P_2 and we add two vertices in the special way, we produce a tripole of type N_2.

Chapter 3

Search for the smallest tripoles

As the first step in our research, we wanted to find the smallest tripoles of types that Kardoš and Narboni used. At that point in time, we did not have a formal classification of tripoles as described in the previous chapter. We were interested in the existence of tripoles that would have an unavoidable or untouchable edge that is smaller in size that the one used by Kardoš and Narboni. The tripole of type P_2 has 27 vertices and the tripole of type P_1 has 47 vertices. For that reason, we focused on tripoles of type P_2.

Let G be a valid vertex-signed planar graph. If we remove a vertex v from G and retain its edges as semi-edges, the remaining graph is a tripole T. If v was a positive vertex, then T is a positive tripole, otherwise T is negative. As G contains an even number of positive vertices, removal of a single positive vertex yields an odd number of positive vertices in T. Similarly for the other case.

For that reason, we do not need to go through all tripoles and for each one verify if it contains an unavoidable semi-edge. We need to go through all 3-connected cubic planar graphs and verify whether one of them has an unavoidable edge. If one does, by removal of one of its endvertices, we get a tripole with an unavoidable edge. If the removed vertex was positive, we found a tripole of type P_2. If it was negative then we found a tripole of type N_2 or N_2^*, but more on those later.

As a result, we need to somehow obtain all vertex-signed cubic planar graphs and then evaluate each obtained graph. For reasons that will become apparent later, we chose to generate such graphs.

3.1 Canonical code of 3-connected cubic planar graphs

The first step in generating all vertex-signed cubic planar graphs is to generate all 3-connected cubic planar graphs. When doing so, we want to avoid generating a graph multiple times. For that reason we need a way to differentiate between graphs and establish a hierarchy amongst them. The concept of a *canonical code* will serve that purpose.

Let G be a 3-connected planar cubic graph. Let f be a face in G of size k and $NCS(f) = \{f_0, \dots, f_{k-1}\}$ be a *cyclical sequence of adjacent faces*, which contains faces adjacent to f, where f_i is adjacent to $f_{i-1(mod\ k)}$ and $f_{i+1(mod\ k)}$. We say such sequence is cyclical, as a rotation of the elements defines the same sequence. For each f there are precisely two sequences, where one is the inverse of the other. In other words, if f_i was followed by f_{i+1} in one sequence, then f_{i+1} is going to be followed by f_i in the other. These represents the sequences of faces in a plane embedding of G. According to 1.1 all embeddings of a 3-connected graph are equivalent, therefore these two sequences are, in fact, the only two such sequences. Based on the direction of the enumeration, we get the respective sequence. We will refer to this sequences as *clockwise* and *counter-clockwise*.

A *spanning tree* ST in G is a subgraph of G that is a tree and contains all vertices of G.

Let G be a 3-connected cubic planar graph and G^* be its dual. Let ST be a spanning tree in G^*. We say that ST is a *face-spanning tree* of G, if we can reference its vertices by corresponding faces in G.

A *Breadth First Search* algorithm (BFS) is a graph covering algorithm. It covers a graph layer by layer. In such algorithm, there are two important attributes of each vertex v. The first attribute is *visited* which is a Boolean attribute indicating whether the algorithm already visited v. The other attribute, called *covered*, is again a Boolean attribute indicating whether all adjacent vertices of v are visited. We construct a *traversal sequence* of vertices in the order they become visited. Beginning with the starting vertex v_0, we add it to the sequence and mark it visited. Now, until there are any non-covered vertices in the traversal sequence, we do the following. We take the first non-covered vertex v_i from the traversal sequence, add all non-visited adjacent vertices of v_i to the traversal sequence and mark v_i covered.

As the graph is finite, this terminates at some point and we are left with a fully traversed graph. The traversal sequence may be interpreted a spanning tree of G. If we keep only those edges v_iv_j of G corresponding to the addition of an adjacent edge v_j of v_i to the traversal sequence, then such graph is indeed a spanning tree.

We would like to run a BFS algorithm on the faces of a 3-connected cubic planar graph G. We know, there exists a dual G^* of G which allows us to run BFS on G^*. For convenience purposes, we would like to reference the vertices of G^* by the corresponding faces of G.

The definition above is general, i.e., non-specific to a special class of graphs. We would like to specify it for the class of 3-connected cubic planar graphs. The only aspect that needs a specification is in what order will we add adjacent non-visited faces of a face to the traversal sequence. Let G be a plane cubic graph. Each face has a clockwise and a counter-clockwise cyclical sequence of adjacent faces. When running the algorithm, we will fix the direction to one of the two. In other words, if the adjacent faces of the starting face were added in a clockwise order, all faces will add their non-visited adjacent faces to the traversal sequence in clockwise order. As the sequence is cyclical, we must specify the leading element. The leading element of f's sequence will always be the face that added the f to the traversal sequence, with an exception of the starting face as it was not added by a face from traversal sequence.

Let S be a traversal sequence produced by a BFS algorithm in G. Then sequence of integers obtained from S by mapping faces onto their sizes is called *BFS code.*

We say, that a BFS code S is lexicographic-ally smaller than a BFS code T, if there exists a non-negative integer i such that $s_i < t_i$ and $\forall j, 0 \leq j < i \; s_j = t_j$. For example $\{4, 5, 5, \dots\}$ is lexicographic-ally smaller than $\{4, 5, 6, \dots\}$.

Let G be a 3-connected cubic planar graph G. Let $\mathcal{S}$ be a set of all BFS codes in G. Let $S \in \mathcal{S}$ be a BFS code that such it is lexicographic-ally the smallest in $\mathcal{S}$, then S is called the *canonical* code of G.

Let G be a 3-connected cubic planar graph G and f be a face in G. Let $\mathcal{S}_f$ be a set of all BFS codes obtained from traversal sequences of a BFS starting with face f. Then the lexicographic-ally amongst those S is called *lexicographical* code of f. Moreover, if S is the canonical code of G as well, then f is referred to as the *origin* face of G.

To get the canonical code of a 3-connected cubic planar graph G, we must choose from $6|V(G)|$ BFS codes. This is not an arbitrary number as each BFS code starts with 3 faces that have a one mutually incident vertex. From the point of view of a fixed vertex v, there are precisely 3 faces in which the BFS code can start if the first 3 faces of the code are meant to be incident to v. For each face there are 2 direction, which gives a product of 6 distinct possibilities for each fixed vertex; $6|V(G)|$ for all vertices. That gives us a motivation to perform some optimisations.

To find the lexicographic code of a face f, the cyclical sequence of adjacent faces of f which adds faces to the traversal sequence must be lead by a face of smallest size adjacent to f. Secondly the direction of the sequence can be chosen based on which direction continues with a smaller face. To find the canonical code of a graph G, it is sufficient to choose amongst lexicographic codes of faces of the smallest degree in G.

As mentioned, the construction of a canonical code c in a 3-connected cubic planar graph G traverses a face-spanning tree G. As we cover each face exactly once, which is equivalent to one edge for each non-origin face in G, such structure must, indeed, be a tree. To better understand this concept, see Figure 3.1.

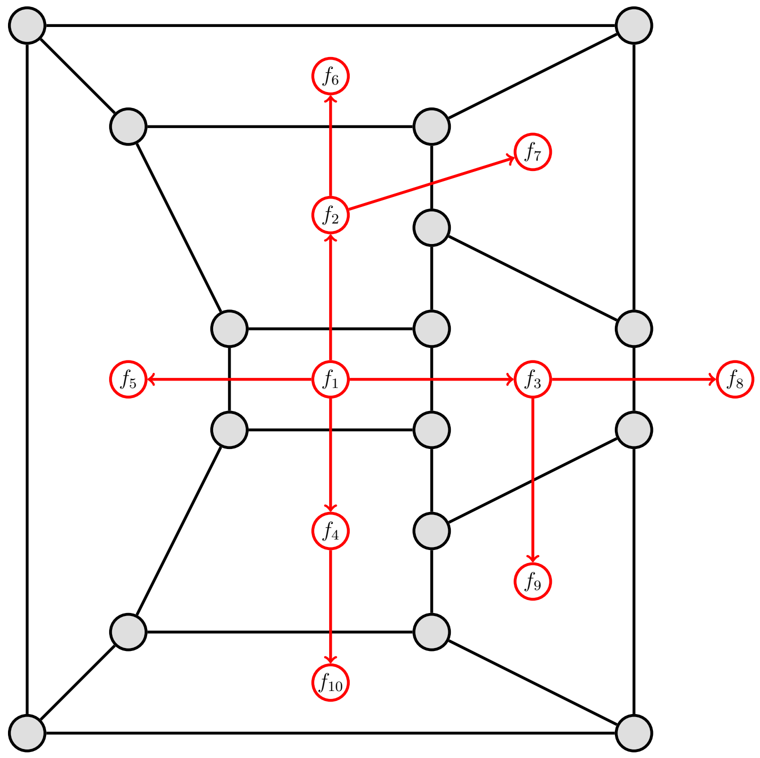


Figure 3.1: In this example, faces of the graph G are denoted according to their index in the traversal sequence S sequence. We chose f_1 as the origin face. Therefore, at the moment $S = \{f_0\}$. As we can see, it is adjacent to two faces of degree 5 and two faces of degree 6. Lets take the two cyclical sequences of adjacent faces and shift them so that they are lead by a face of the smallest size. We end up having 4 different sequences, two clockwise ($\{f_2, f_3, f_4, f_5\}$ and $\{f_4, f_5, f_2, f_3\}$) and two counter-clockwise ($\{f_2, f_5, f_4, f_2\}$ and $\{f_4, f_3, f_2, f_5\}$). For demonstration purposes, we chose the first clockwise sequence. Following the first step, we have all faces from the chosen sequence into S which is now equal to $\{f_1|f_2, f_3, f_4, f_5\}$ where $|$ splits the covered and open faces. Let's now take to second part of the algorithm, where we want to cover all faces in S. First up, we need to cover f_2. Therefore, we take the clockwise cyclical sequence of adjacent faces of f_2, shifted so that it is lead by f_1. Let's now add all faces that are not already in S into S. The second face in this sequence is f_5, so we do not add it to S. Similarly, the last face is f_3, so we do not add that one either. However, we need to add f_6 and f_7 as those have not yet been added. At this point we have covered f_2 and S looks as follows $\{f_1, f_2|f_3, f_4, f_5, f_6, f_7\}$. The next face to cover is f_3. Again we take the clockwise sequence lead by f_1. For the same reasons, we only add f_8 and f_9 to S. Now, $S = \{f_1, f_2, f_3|f_4, \dots, f_9\}$. After covering f_4, $S = \{f_1, f_2, f_3, f_4|f_5, \dots, f_{10}\}$. At this point, all faces of G are in S, so for the rest we can just check that they are covered and finish the process. Therefore, our pre-lexicographic sequence S looks as follows $\{f_0, \dots, f_{10}\}$, which makes the lexicographic code l look as $\{4, 5, 6, 5, 6, 4, 4, 5, 4, 4\}$. Observer that $l \neq \gamma(G)$, because l starts with $\{4, 5, \dots\}$ and G contains two adjacent faces of size 4 (for example f_6, f_7), so $\gamma(G)$ will start as $\{4, 4, \dots\}$.

Lemma 3.1 *Let G and H be two 3-connected cubic planar graphs. Then $G \simeq H$ if and only if $\gamma(G) = \gamma(H)$.*

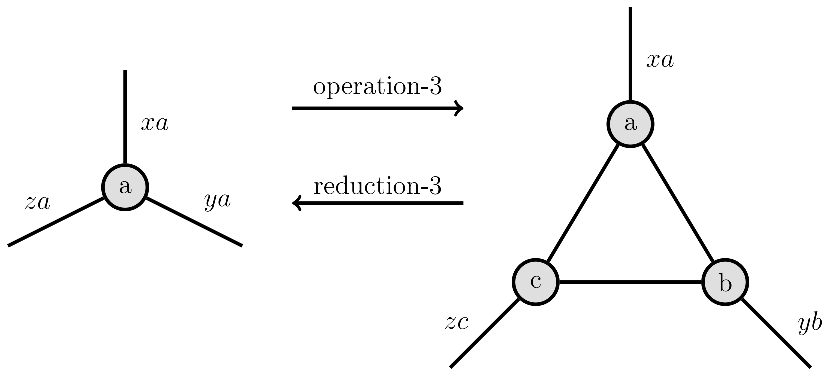
Proof. For the implication from left to right, we know that G and H are isomorphic. Non-formally said, G and H are the same graph, therefore the canonical code construction will yield the same result.

For the opposite implication, let $\gamma(G) = \{c_1, \dots, c_n\}$ be a canonical code. Additionally $\gamma(G) = \gamma(H)$. Let H be a graph non-isomorphic to G. Let's try to rebuild G and H from $\gamma(G)$. As we know, c_1 is the size of the origin face in both G and H, which means it is surrounded by c_1 faces $f_2, \dots, f_{c_1+1}$. Therefore both G and F must contain a subgraph consisting of these $c_1 + 1$ faces. At this point we have covered c_1, as all its adjacent faces have been rebuilt. Let's now take an open element from $\gamma(G)$ which is an integer c_i corresponding to the size of an open face f_i. At this we have j $(i < j)$ faces rebuilt in G and H. Through silent induction assumption, we know that all the faces placed into G so far create the same subgraph as in H. In other words, the rebuilt subgraphs are isomorphic. According to the placement of f_i, there are $k \geq 0$ faces adjacent to f_i that we have not yet rebuilt. After rebuilding those in G and H the known subgraphs are still isomorphic as we did precisely the same steps in both graphs. We repeat the process until we cover all the elements in $\gamma(G)$. In the end the induction assumption still holds as in it did not break in none of the iterations. Now we have rebuilt both graphs and they are still isomorphic, we can clearly see a contradiction with the assumption that G and H are not isomorphic, therefore lemma 3.1 holds. $\square$

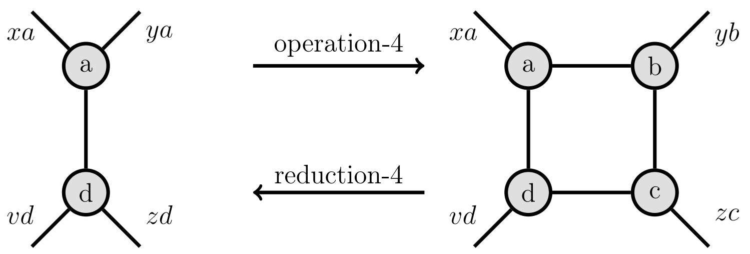
3.2 Hierarchy of 3-connected cubic planar graphs

Let $\mathcal{C}$ be a class of 3-connected cubic planar graphs. Moreover, $\mathcal{C}_{\leq n}$ is a subclass of $\mathcal{C}$ containing only graphs of size n and smaller. Lastly, $\mathcal{C}_n$ will denote a subclass of $\mathcal{C}_{\leq n}$ containing only graphs of size n. Our goal will be to generate all graphs of $\mathcal{C}_{\leq n}$ for some fixed n.

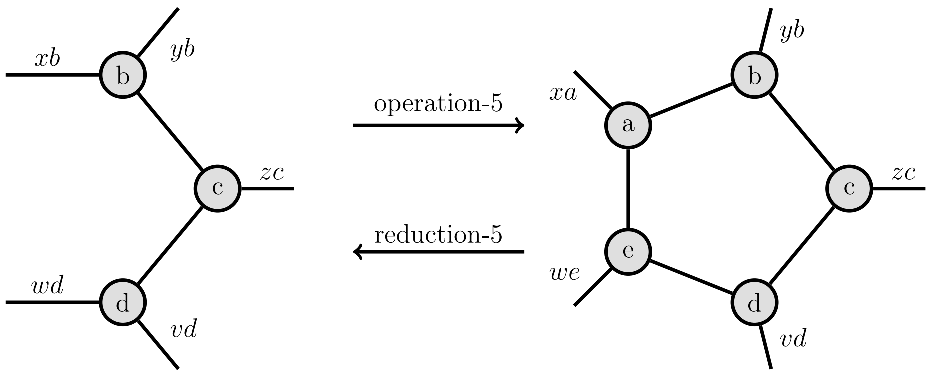
Let f be a face of size 3 in a 3-connected planar cubic graph $G = (V, E)$. Let f be incident to vertices a, b and c, which implies the incidence of f to edges ab, bc and ca. Let xa, yb, zc be edges not incident to f but incident to a, b and c respectively. Let $G^*(V^*, E^*)$ be a 3-connected cubic planar graph, where $V^* = V \setminus \{b, c\}$ and $E^* = E \cup \{ya, za\}$. Note that edges incident to removed vertices are not in E^* as well. An operation $G \rightarrow G^*$ is called *reduction-3*, whereas an operation $G^* \rightarrow G$ is called *operation-3*.



Let f be a face of size 4 in a 3-connected planar cubic graph $G = (V, E)$. Let f be incident to vertices a, b, c and d, which implies that f is incident to edges ab, bc, cd and da as well. Let xa, yb, zc and vd be edges not incident to f, but incident to a, b, c and d respectively. Let $G^*(V^*, E^*)$ be a 3-connected cubic planar graph, where $V^* = V \setminus \{b, c\}$ and $E^* = E \cup \{ya, zd\}$. Edges incident to b or c are not present in G^* either. An operation $G \to G^*$ is called *reduction-4* and its inverse operation $G^* \to G$ is called *operation-4*.



Lastly, let f be a face of size 5 in a 3-connected planar cubic graph $G = (V, E)$. Let f be incident to vertices a, b, c, d and e, which implies that f is incident to edges ab, bc, cd, de and ea as well. Let xa, yb, zc, vd and we be edges not incident to f, but incident to a, b, c, d and e respectively. Let $G^*(V^*, E^*)$ be a 3-connected cubic planar graph, where $V^* = V \setminus \{a, e\}$ and $E^* = E \cup \{xb, wd\}$. Edges incident to a or e are not present in G^* either. An operation $G \to G^*$ is called *reduction-5* and its inverse operation $G^* \to G$ is called *operation-5*.



We say that G is *reducible* into G^* if there exist a reduction-3,4 or 5 such that G^* is the result of such reduction.

As a consequence of Euler's formula, each 3-connected cubic planar graph G contains a face f_0 of size smaller then 5 which is also the origin face of G. If we apply a corresponding reduction on f_0 the resulting graph G^* is the *parent* of G. Additionally, if G^* is the parent of G then, then G is the *child* of G^*.

Let $Q = (V, E)$ be an oriented graph. If Q has the following properties:

- Each vertex represents a 3-connected cubic planar graph.
- If v_1, v_2 are vertices, then Q contains an arc from v_1 to v_2 if and only if the graph represented by v_2 is a child of graph represented by v_1.
- Q is rooted at a vertex representing the smallest planar cubic graph K_4.

Then Q is a *search tree.*

A *subtree* of a vertex v in a tree Q usually refers to a subgraph of Q where all the vertices are connected to V, i.e., there is a directed path from v to any vertex in the subtree. Since Q is a search tree, we will refer to a subtree as a *search branch.*

An eagle-eyed reader may have noticed that the parent-child relation is defined rather vaguely. For example, there are two ways to perform the reduction-4 for G's origin face of size 4. That would mean that G has two parents which is not desired.

We do not need to bother with a proper definition of the parent-child relationship for reduction-3 as there is only one way to perform it. What we however need to specify is the relationship when it comes to the reduction-4 and 5. The reduction from child to parent will be called *canonical.*

Let f_0 be the origin face of G of size n (= 4 or 5). Let $\gamma(G) = \{n, \dots\}$ be the canonical code of G corresponding to a sequence of faces $\{f_0, f_1, \dots\}$. Then the reduction which removes the edge between faces f_0 and f_1 is the canonical reduction.

Lemma 3.2 *Let R be the canonical reduction $G^* \to G$, where the origin face f of G^* is a square.*

Proof. Assume G is not 3-connected. That implies that the edge between f and f_1 (the second face in G^*'s canonical code) has been removed by the canonical reduction. If f is a square then f_1 must be at least of size 6 so that G^* is 3-connected. We know that the canonical code starts as $\{4, 6, \dots\}$ which implies that if G^* contains squares other than f, they must not be incident to a face of size smaller than 6. Due to high amount of cases, we omit the technical details, but it can be shown that with these restrictions, G^* cannot be completed, i.e., such G^* does not exist. $\square$

Lemma 3.3 *Let R be the canonical reduction $G^* \to G$, where the origin face f of G^* is of size 5.*

Proof. Similarly as before, for G to be non-3-connected, the edge between f and f_0 must have been removed. For G^* to be 3-connected, f_0 must be at least of size 5. Naturally, G^* must not contain any squares or triangles. Once again, due to high amount of cases, we omit the technical detail as to why G^* cannot be completed, i.e., why G^* with these restrictions does not exist.

It is next to impossible to determine all the children for a given 3-connected cubic planar graph G. We must do all admissible operations and later verify, if the newly discovered graph is a child of G. For that reason we need to traverse a larger acyclic graph that contains the entire search tree. Formally, let Q be a search tree and P be an acyclic graph such that $V(Q) = V(P)$ and $E(Q) \subseteq E(P)$. Then P is a search pseudo-tree.

Figures 3.2, 3.3 demonstrate all applications of operations-3 and 4. These are in fact all the possibilities as all the other operations-3 and 4 produce isomorphic graphs due to symmetry.

3.3 Scaling the search pseudo-tree

Let $P = (V, E)$ be a search pseudo-tree, then $|V|$ is called the *size* of Q. Moreover, $|E|$ is called the *density* of P.

In order to improve the complexity of our algorithm, we will want to make both size and density of the search pseudo-tree as small as possible.

To get all the 3-connected cubic planar graphs, we have to do do the operations 3,4 and 5 at each vertex v of the search pseudo-tree at all viable places in the graph represented by v. Each operation represent an edge in the search pseudo-tree. Our ultimate goal is to make an operation from G to G^* only if G is the parent of G^*.

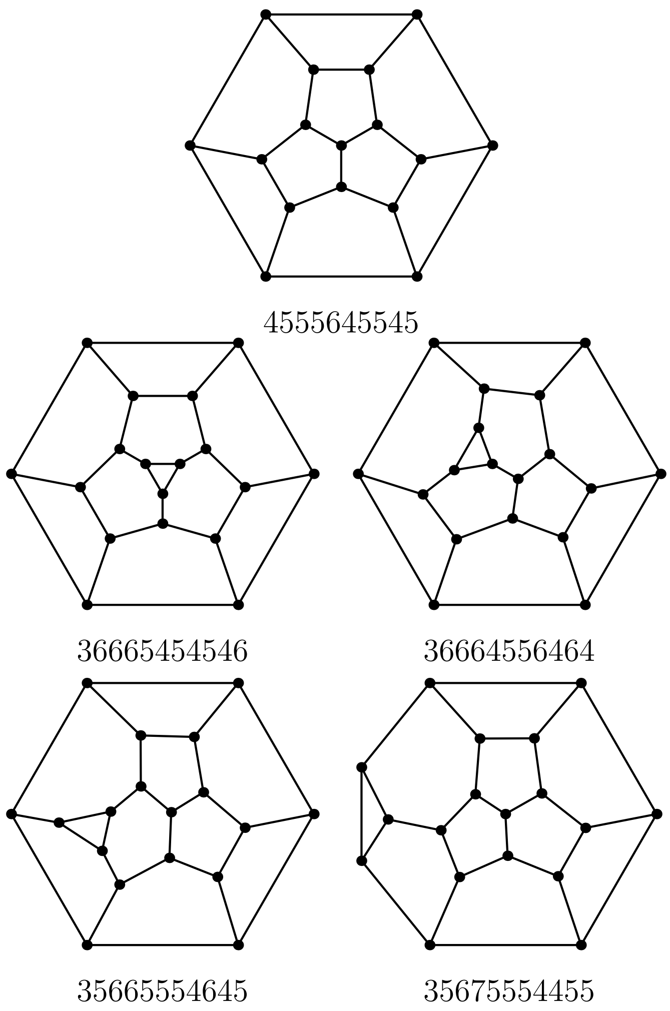


Figure 3.2: An example of a graph G_{16} (on the top) and all the graphs that can be obtained from it by applying the Operation 3, together with their canonical codes. Note that all these graphs are children of G_{16} as the newly added triangles are the only triangles meaning that the canonical code will start in them.

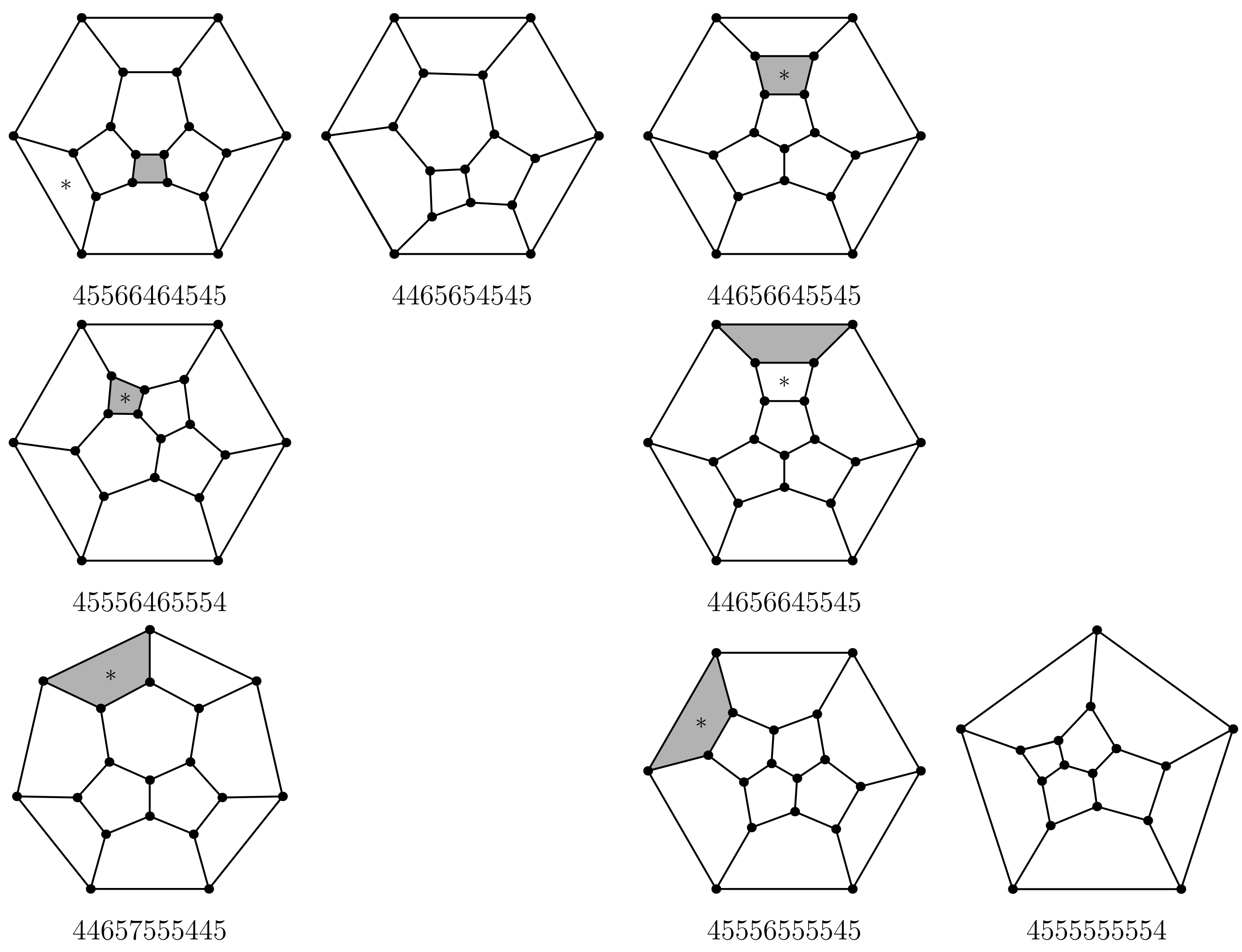


Figure 3.3: The graphs that can be obtained from the graph G_{16} depicted in the previous Figure, by applying Operation 4, together with their canonical codes (first and third column). A newly created face is shaded, whereas the face that defines the canonical code is marked with an asterisk. If a graph is not a child of G_{16}, then its parent is depicted to the right of it. Observe that there are always two ways to apply Reduction-4 to a 4-face, but only one of them is canonical. That's the reason why for the last graph, the asterisk coincides with the shades face, however, the graph has a different parent.

That however is not possible, because to determine if G is a parent of G^*, we have to know G^* to compute its canonical code which we cannot do before we do the actual operation. Fortunately, there are situations when we know for sure that G is not the parent, so we might avoid doing the operation, hence removing an edge in the search pseudo-tree. The following set of lemmas is going to describe the situations in more detail.

We say that an operation $\omega : G \to G^*$ is *futile* if it is guaranteed that G is not a parent of G. Otherwise, ω is *plausible*. In other words, an operation is futile if its corresponding edge in the search pseudo-tree is not present in the search tree.

Let f be a face in a 3-connected cubic planar graph G. Let $S_f = \{f_0, \ldots, f_{n-1}\}$ be a cyclical neighbour sequence of f. We say that the distance of faces f_i, f_j, $i < j$ at face f is $\min(j - i, i + j \mod n)$, denoted as $dist_{G,f}(f_i, f_j)$.

Let's see what happens when we perform and operation-4 ω in G. First we need to parameterise the operation. The first argument is the face in which we do the operation, lets denote it as f. Second argument is an edge incident to f, referred to as a *start edge*, denoted as e_s. We will use the notation of $\omega(face, edge)$. When we perform ω. Firstly, the start edge will be split in half by a vertex v_1. Therefore, the face f_s incident to e_s other than f will grow in size by one. Let's now take a face that is in distance of 2 at f from f_s, denoted as f_{s+2}. The edge shared by both and f_{s+2} will be denoted as e_{s+2}. The face and edge between these two faces at f will be denoted as f_{s+1} and e_{s+1} respectively. Similarly as for e_s, e_{s+2} will be split in half by v_2, rendering f_{s+2} bigger by one. Now, by connecting v_1 to v_2, f will be split into half by v_1v_2. In the end, there is a new face adjacent to f_s, f_{s+1}, f_{s+2}, f called a *new* face. Note that the size of face f and f_{s+1} did not change, whereas the size of f_s and f_{s+2} increased by one.

Lemma 3.4 *The parameter of an operation-4 ω must be a face of size 4 or greater.*

Proof. Assume we perform ω in a triangle t. As we know, t will not grow in size, therefore G^* will contain a triangle, rendering ω futile. □

Lemma 3.5 *Let $\omega : G \to G^*$ be an operation-4. ω is plausible only in one of these cases:*

- *G contains no triangles.*
- *G contains one triangle and G^* contains no triangle.*
- *G contains two triangles f_a, f_b. Moreover there exists a face f_m in G at which f_a and f_b are in distance of 2. Furthermore, G^* must contain no triangles.*

Otherwise, ω is futile.

Proof. Firstly, for all the cases. If G^* contained a triangle, than the triangle would be the origin face of G^* and not the new face added by ω. Now, let's go case by case.

- The first case is simple. By doing an operation-4, all faces in G remain the same or grow in size by one. If G contained no triangles, then G^* will not contain any as well. Therefore the origin face of G^* will be a face of size 4, potentially the new face.

- In this case, ω must add a new face next to the triangle t, so that it no longer is adjacent to just 3 faces, hence making it a square. That means that the parameters of ω are a face f adjacent to t and an edge incident to both f and t. Therefore, t will grow in size by one and G^* will contain no triangles.

- The third case is the trickiest. If we do not want any triangles in G^*, ω must eliminate both triangles. Therefore they must be in a precise position so that it is possible. That is precisely when the triangles are in a position as described in the lemma. Therefore, by performing ω, where the parameters are f_m and an edge f_s incident to both f_m and f_a (we assume that f_a precedes f_b in the given cyclical neighbour sequence). Therefore f_{s+2} is precisely the edge incident to both f_m and f_b. We know that performing ω with such parameters will increase size of faces f_a and f_b by one, therefore G^* will contain no triangles.

Suppose that G contains more than 2 triangles. A single operation-4 ω is only capable of removing 2 triangles, therefore G^* will contain a triangle which renders ω futile. □

Note that this lemma also proves lemma 3.4, but we decided to point out the content of lemma 3.4 separately.

Let $\omega_1 : G \to G_1^*$ and $\omega_2 : G \to G_2^*$ be operations-4. If $G_1^* \simeq G_2^*$ then we say, ω_1 and ω_2 are *duplicate.* Naturally, each operation is duplicate to itself. A pair of duplicate operations corresponds to parallel edges in the search pseudo-tree. We will want to avoid doing duplicate operations.

Lemma 3.6 *Let e be and edge in a 3-connected cubic planar graph G incident to f_l and f_r, where both faces are greater in size than 3. Let $S_l = \{f_{l,0}, f_r, f_{l,2}, \dots\}$ and $S_r = \{f_{r,0}, f_l, f_{r,2}, \dots\}$ be the cyclical neighbour sequences of f_l and f_r respectively. Furthermore, let $\omega_1 = \omega(f_l, e_{l,0}), \omega_2 = \omega(f_r, e_{r,0})$ be two operations-4, where $e_{l,0}$ is an edge incident to both f_l and $f_{l,0}$, and $e_{r,0}$ is an edge incident to both f_r and $f_{r,0}$. Then ω_1 is duplicate to ω_2.*

Proof. without loss of generality, lets assume that both S_l and S_r are clockwise sequences. Then it is easy to see, that $f_{l,0}$ is the same face as $f_{r,2}$, denoted as f_1, and $f_{l,2}$ is the same face as $f_{r,0}$, denoted as f_2. We know that ω_1 will increase the size of f_1 as it is the parameter and f_2 as it is in distance of 2 at f_l. The size of f_l and f_r will not change, but both will be adjacent to a new face f_n. Similarly for ω_2. The size of f_2 will increase because it is the parameter and the size of f_0 will increase as it is in distance of 2 from f_2 at f_r. Both f_l and f_r will have their size unchanged and both will be adjacent to the new face. □

To discuss operation-5 in more detail, we first need to parameterise it. Similarly as for operation-4, the parameters will be a face and an edge incident to it. We will once again use the notation $\omega(face, edge)$. There are a lot of similarities with operation-4, therefore we will focus on the differences. Let f be the face and e_s the edge in parameters. Let f_s is the face incident to the start edge other than f. This time, we will be interested in a face f_{s+3} in distance of 3 from f_s at f. Let s_{s+3} be the edge incident to both f and f_{s+3}. Both e_s and e_{s+3} will be split by a vertex and connect by and edge e. This edge e is adjacent to f and the new face f_n. As a result, the cyclical neighbour sequence of f_n looks as $\{f_s, \ldots, f_{s+3}, f\}$. Observe that the only face size that have increased are the size of faces f_s and f_{s+3}. However, the size of face f has decreased by one as it has been adjacent to f_{s+1} and f_{s+2} in G whereas in G^* it is no longer adjacent to those but the new face f_n.

Lemma 3.7 *The parameter of an operation-5 ω must be a face of size 6 or greater.*

Proof. Assume that the parameter face f is of size $n < 6$ or smaller. Then by performing ω, f will be of size $n - 1$ in G, i.e., f will be of size 4 or smaller. Therefore, the new face is of greater size than f, hence it will never be the origin face of G^*, rendering ω futile. □

Lemma 3.8 *Let $\omega : G \to G^*$ be an operation-5. ω is plausible only if G contains no triangles. Moreover, one of these cases must be true.*

- *G contains no cubes.*
- *G contains one cube and G^* contains none.*
- *G contains two cubes f_a, f_b. Moreover there exists a face f_m in G at which f_a and f_b are in distance of 3. Furthermore, G^* must contain no squares.*

Proof. If G contained triangles, then G^* would contain faces of size 4 or smaller, therefore the new face will not be the origin face. Now let's discuss the cases. To proofs of the first two cases are analogical to the cases first two cases in lemma 3.5.

In the third case, we must acknowledge that the face in which we do the operation is at least of size 6. Therefore ω will for sure not decrease its size below 5. Additionally, the condition in the third case puts the two cubes in exact position so that the first $_s$ is incident to the start edge e_s and the other one f_{s+3} is incident to e_{s+3}. Those are precisely the faces that are going to increase in size, therefore G^* will contain no cubes. □

Our goal is to find the smallest tripole of each type. That implies that those tripoles must not contain any reducible configurations. Therefore, we do not need to generate a set of all 3-connected cubic planar graphs, but only a subset of the set. Unfortunately, we cannot omit graphs that contain reducible configurations as such graphs may be unique parents of graphs that do. However we can omit those, where the reducible configuration cannot be gotten rid of.

The most simple of reducible configurations is a single face of size 3. Thanks to the lemmas 3.4 and 3.8, we know that an operation σ will produce a graph without triangles, if the conditions in the lemma 3.4 are met. We will show that a graph G not compliant to lemma 3.4, in a sense that there exists an operation-4 ω which is not futile, will never have a graph with no triangles in its respective search branch. We will call such graph *vain*.

Let G be a 3-connected cubic planar graph, then $\sigma_n(G)$ denotes the number of faces of size n in G.

Lemma 3.9 *Let G be a 3-connected cubic planar graph and G and $\omega : G \to G^*$ be an operation-3. Then $\sigma_3(G) \leq \sigma_3(G^*)$.*

Proof. Assume that $\sigma_3(G) > \sigma_3(G^*)$. Then G must contain a vertex that is incident to at least 2 faces of size 3, as a single operation-3 can increase the size of only the 3 faces incident to the vertex in parameter. That implies G would have to contain a 2-cut, which is in contradiction with G being 3-connected. □

Lemma 3.10 *Let G be a 3-connected cubic planar graph. If there exists no non-futile operation-4 ω, then G is vain.*

Proof. It should be apparent that lemma 3.8 is more restricting than lemma 3.5 in the sense, that if there exists a non-futile operation-5, then there must exist a non-futile operation-4.

If there exists no non-futile operation-4, then only operations-3 are viable, which means that G^* will always have at least as many triangles as G. Let's now analyse why the operation-4 might be non-futile in G and what consequence it will have on G^*.

- G contains more than $n > 2$ triangles. Then according to lemma 3.9, G^* will contain at least n triangles. Therefore, there will be no non-futile operation-4 in G^*.

- G contains 2 triangles that do not have a mutual adjacent face. By doing an operation-3, no faces will merge so that those triangles could have a mutual adjacent face. The rest of the reasoning is analogical to the previous case.

- G contains 2 triangles t_1, t_2 that have a mutual adjacent face f, but their distance at f is greater that 2. If ω adds the new face so that it is not adjacent to f, we can do the same reasoning as in the previous cases. If it does, then it either adds a third face adjacent to f, or it makes one of the triangles a square. It happens, if ω has one of the vertices incident to t_1 or t_2 as its argument. Let's look at the cyclical neighbour sequence S of f. Since t_1 and t_2 are in distance of at least 3 there is a minimum of 2 faces between t_1 and t_2 in S. Operation-3 is basically an insertion of a face of size 3 in to the sequence. As we know, it will next to t_1 or t_2. That however will not shrink the gap between t_1 and t_2. For that reason, G^* will not admit any non-futile operation-4.

In each case, the resulting graph does not admit a non-futile operation-4. Through induction, we can clearly see that G is vain. □

As all the graphs that do admit a non-futile operation-4 have at most 2-triangles, moreover those triangles are both adjacent to a face f, at which they are in distance of 2, we will denote a class of these graphs as $\mathcal{C}^{2\Delta}$.

Lemma 3.11 *Let G be a 3-connected cubic planar graph, where $\sigma_3(G) = 2$ and there exists a non-futile operation-4. Moreover, let $\omega : G \to G^*$ be a non-futile operation-4, then G is not the parent of G^*.*

Proof. As G admits an non-futile operation-4, there exist a face f that is adjacent to both triangles. Face f must be at least of size 5, otherwise there would be a 2-edge-cut in G. G^* as the result of ω contains a new face f_n of size 4 adjacent to two faces of size 4, i.e., the former triangles and two faces of size at least 5, one of which is f. If f_n is the origin face in G^* then the canonical code will for sure start as $\{4, 4, \dots\}$. By the definition of parent-child relationship, the canonical reduction is the one that removes an edge between the first two faces in the canonical code. As f is bigger than 4, the canonical reduction will never result in G. □

As a consequence of this lemma, and the fact that we only care for graphs containing two triangles because of their children that don't contain any, we do not need to bother with them at all. In other words, if we eliminate all graphs containing two triangles from the search tree, we will not loose any 3-connected cubic planar graphs that do not contain reducible configurations. The class of 3-connected cubic planar graphs containing only one triangle will be denoted as $\mathcal{C}^{\Delta}$.

3.4 Verifying semi-2-factors

Now that we have obtained a set of 3-connected cubic planar graphs, we want to verify if it contains a graph G for which exists a way to assign polarities to its vertices so that it behaves as one of the desired tripole types if we remove one of its vertices. A concrete assignment of polarities to vertices of a graph is called *polarity distribution* on G. The desired tripole types are those containing an unavoidable and untouchable edge. For each graph there are $2^{|V(G)-1|}$ possible polarity distributions. Note that the power is $|V(G) - 1|$ and not just $|V(G)|$, because we need G to be a valid dual of a a signed planar graph, i.e., contain an even number of vertices. Furthermore, for each polarity distribution, we need to check if G contains an unavoidable or untouchable edge. It is obvious that the complexity of such approach is inadmissible.

We say, that an edge e is ham-unavoidable, if there exists no Hamilton cycle not containing e.

Similarly, if there exists no Hamilton cycle containing an edge e, then e is ham-untouchable.

Lemma 3.12 *Let G be a vertex-signed planar graph G. Moreover, let e be an edge in G. Then if e is unavoidable then it must ham-unavoidable.*

Proof. Assume that e is unavoidable but not ham-unavoidable. That means there exists a Hamilton cycle C not containing e. Such Hamilton cycle must contain and even number of positive vertices, as it contains all vertices of G. That means that C is a semi-2-factor in G not containing e, which is in contradiction with e being unavoidable.
□

Lemma 3.13 *Let G be a vertex-signed planar graph G. Moreover, let e be an edge in G. Then if e is untouchable then it must be ham-untouchable.*

Proof. Similarly as before, assume that e is untouchable but not ham-untouchable. That means there exists a Hamilton cycle C containing e, which in itself is a semi-2-factor, which is in contradiction with e being untouchable. □

We said that for each polarity distribution we need to verify if there is an unavoidable or untouchable edge. Obviously, this statement is invertable. As Hamilton cycles do not deal with polarities it means that the concepts of ham-unavoidable and ham-untouchable edges are polarity-independent. For that reason we can first check if a 3-connected cubic planar graph contains an edge e of one of these edge types, and for each such edge deal with polarities later. That will allow us to avoid polarity distribution that would guarantee that e will not unavoidable or untouchable.

For each edge that is ham-unavoidable or ham-untouchable, we sort vertices of the graph into two groups. Firstly, it's the *forced positive* vertices, which is a set of vertices that we will for sure know about that must be positive, otherwise the edge would not be unavoidable or untouchable. The rest of the vertices are gonna be called *ambiguous*.

Lemma 3.14 *Let G be a 3-connected cubic planar graph and v a vertex in G. Moreover, let e be a ham-unavoidable edge in G. If $G \setminus \{v\}$ admits a Hamilton cycle not containing e then v must be forced positive otherwise e would not be unavoidable.*

Proof. Assume that e is unavoidable but v is negative. Note that v being negative allows a semi-2-factor to avoid v same as the removal of v from G allows Hamilton cycle to not cover v. If $G \setminus \{v\}$ admits a Hamilton cycle C not containing e then C is a semi-2-factor in G not containing e, which is in contradiction with e being unavoidable. □

Lemma 3.15 *Let G be a 3-connected cubic planar graph and v a vertex in G. Moreover, let e be a ham-untouchable edge in G. If $G \setminus \{v\}$ admits a Hamilton cycle containing e then v must be forced positive otherwise e would not be untouchable.*

Proof. Similarly as in the previous lemma, assume that e is untouchable but v is negative. If $G \setminus \{v\}$ admits a Hamilton cycle C containing e then C is a semi-2-factor in G containing e, which is in contradiction with e being untouchable. □

Chapter 4

Implementation and results

As previously mentioned, the goal of our work will be to find the smallest tripoles of each type. That however is impossible to do in hand, therefore we utilised computer assisted search. For this purpose we need to do all the steps described in the previous chapter. First we need to generate all the graphs that we will test for presence of semi-2-factors in some of their polarity distributions. First and foremost we need to interpret the concept of a graph so that a computer can do all the necessary operations with it efficiently.

4.1 Representation

The programming language of choice in this work was Java. Java is famous for its object oriented design. Therefore, All the concepts of a graph will be represented by their own class. Moreover, for reasons that will be explained later, we always keep the instances of these classes in such states so they represent a concrete plane embedding.

First lets describe the representation of a vertex. A vertex is represented by a class *Vertex*. All that an instance of *Vertex* needs to remember is a list of its incident edges in such order that it represents a fixed embedding.

Secondly, an edge is represented by a class *Edge*. An instance of *Edge* remembers the vertex it goes from and the vertex it goes to. Note that class *Edge* does not represented an oriented edge. Secondly, an instance remembers the face on the right and the face on the left from the point of view of the starting vertex. The point of this is to know the position of faces in the given embedding.

The last component of the graph is a face. A face is represented by a class *Face*, whose instance remembers a cyclical sequence of its incident edges.

Lastly, there is the class *Graph*, whose instance represents a graph. It remembers all of its vertices, edges and faces.

4.2 Generation

Our algorithm will traverse the search tree as described in the previous chapter. The only problem is that the search tree as defined is not finite. For that reason we will only traverse the tree to some predetermined depth that will be a parameter of the algorithm called *search tree depth*. For example, if the search tree depth is 26, the output of the generation will be all the graphs of degrees up to 26.

The operations corresponding to the edges in the search tree are performed by methods on the instance of class *Graph* itself. Each method has its reverse method same as for each operation there is a reduction. The necessity of the reverse operations is simple. We only want to maintain one instance of the graph.

Methods have exactly the same parameters as their corresponding operation.

- Method *op3*, corresponding to operation-3 takes as a parameter a vertex and performs operation-3. Its reverse method, called *undoOp3*, corresponding to reduction-3, takes as a parameter the face created by *op3*.
- Similarly, method *op4*, which corresponds to operation-4 takes as a parameter a face f of size at least 4 (lemma 3.4 and an edge incident to that face; and performs operation-4. Its reverse method, called *undoOp4* corresponds to reduction-4, and takes as a parameter the edge that is incident to both f and the new face.
- Lastly, method *op5*, which corresponds to operation-5 takes as a parameter a face f of size at least 6 (lemma 3.7) and an edge incident to f; and performs operation-5. Its reverse method, called *undoOp5* corresponds to reduction-5, and takes as a parameter the edge incident to both f and the new face.

Another necessary component is an utility constructing canonical codes. For that purpose, we have a class *BFSCoder*, which has a several neat methods. Note that a canonical code is represented as a list of integers. First of the methods is *getLexCodeForFace* that computes the lexicographic code of a face. The algorithm is pretty much copy of the lexicographic code's definition. The second important method is *isMinimal* which returns true if the lexicographic code provided as parameter is the canonical code of the graph. Otherwise it returns false.

The last component is a class which provides operands for operations. It is hosted in class called *OperandProvider*, which has 3 crucial methods. *getOp3Operands* returns a set of operands for operation-3 so that neither of the operations described in the results is futile or leads to a vain graph. *getOp4Operands* does the same as *getOp3Operands*, but for operation-4 and *getOp5Operands* for operation-5.

The algorithm traversing the search tree is implemented in class *TopVerifyingGenerator*. The process begins with an initial graph, which will be represented by the root node in the search tree. To traverse the edges of the search tree, we do operations-3,4 and 5 performed by the above mentioned methods and reductions-3,4 and 5 when we are coming back from a search tree branch. For each step in the recursion, the algorithm does the following.

- Checks if the newest face is the origin face in the current instance. If not it immediately returns, as the current instance is not the child of the graph in the previous step of the recursion.
- Checks if the graph with given canonical code was already discovered (necessary because of symmetries). If yes, it returns immediately, otherwise it adds its canonical code into the set of discovered codes and proceeds with the search branch.
- Checks if the graph size is equal to search tree depth. If yes, it proceeds with verification of semi-2-factor existence which we will discuss later. If not it proceeds as follows.
- Uses *getOp3Operands, getOp4Operands* and *getOp5Operands* to get operands for respective operations and proceeds to do all of them one by one.
- Finally, returns from the search branch.

Observe that we only verify semi-2-factor existence if the current depth of recursion is equal to the search tree depth. For example, if we set the search tree depth to 28, we will only verify semi-2-factors in graphs of size 28.

The effectivity of this algorithm is dependent on the amount of operands provided. We know that the number of 3-connected cubic planar graphs grow exponentially with n, where n is the size of graph. It is difficult to state the complexity of this algorithm, but the explicit measurements of the search tree size and density should be a good measure of how the improvements caused by several improvements effect the complexity.

As a naive approach, we wanted to generate all cubic planer graphs to some fixed size n, i.e., all graph classes $\mathcal{C}_4, \ldots, \mathcal{C}_n$ We knew that we will be using canonical codes to determine the isomorphism of generated graphs. From the very beginning, we were aware that operations-4 and 5 can result in G^* which is child of G only if G^* will not have triangles or squares respectively. We also took into account that operations-4 are duplicate if they are performed around the same edge, which corresponds to lemma 3.6. The last thing we did account for was that the operand of an operation-4 must be a face of size greater than 4 (lemma 3.4) and the operand of an operation-5 must be a face of size greater than 6 (lemma 3.7). The tables bellow demonstrate how the density and tree size changed when we incorporated the claims of lemmas 3.10 and 3.11 into the algorithm. For comparisons, see tables 4.1, 4.2 and 4.3.

Depth	8	10	12	14	16	18	20	22	24	26	28
Graphs	2	5	14	50	233	1249	7594	49543	339427	2403389	17452555
Density	9	38	107	319	1150	5281	29180	187382	1307208	9609985	73368003

Table 4.1: The number of graphs of size equal to the search tree depth n together with the density of the search tree. In this version, our algorithm was doing operation-3 at all vertices of a graph. Since we did not avoid any graphs, the number in the row graphs represents the total number of 3-connected cubic planar graphs of that size, i.e., graphs from class $\mathcal{C}_n$.

Depth	8	10	12	14	16	18	20	22	24	26	28
Graphs	2	3	6	17	61	241	1124	5601	29592	161963	910240
Density	9	32	69	153	406	1231	4254	18020	84546	435435	2776582

Table 4.2: The number of graphs of size equal to the search tree depth n together with the density of the search tree. The algorithm corresponding to this search tree was using claims of lemma 3.10, i.e., graphs from the class $\mathcal{C}_n^{2\Delta}$. We can see a dramatic decrease in density and the number of graphs for each n.

4.3 Semi-2-factor verification

Based on the results of lemmas 3.12 and 3.13, we must first check if the graph in question contains a ham-unavoidable or ham-untouchable edge. After we have done that, we will look for ambiguous vertices for the special edge. Both of these tasks revolve around Hamilton cycle construction, therefore we host an utility which constructs Hamilton cycles in class called *AbstractPreProcessor*. The key method in this class is called *cycleConstructible* which returns true if we can construct a Hamilton cycle in the provided graph. Otherwise it returns false. Then, for of the sub-tasks there is a class which extends this parent task. But more on that later, let's first take a closer look at how the *cycleConstructible* works.

We will remember a state of each edge and vertex. There are 3 states that an edge can be in.

- *UNDISCOVERED*. If an edge is *UNDISCOVERED* it means that it has not yet been decided, whether this edge will be in the Hamilton cycle or not. In the beginning.

Depth	8	10	12	14	16	18	20	22	24	26	28
Graphs	1	2	4	12	45	179	844	4261	22670	124829	704631
Density	0	20	51	123	340	1018	3420	14204	65502	335031	2227489

Table 4.3: The number of graphs of size equal to the search tree depth n together with the density of the search tree. The algorithm corresponding to this search tree was using claims of lemma 3.11, i.e., graphs belonging to a class $\mathcal{C}_n^\Delta$. Note that the density is offset, as we had to start with a graph on 8 vertices, which is the smallest graph containing at most 1 triangle.

- *INCLUDED*. If an edge is *INCLUDED*, it means that it is part of the Hamilton cycle being constructed.
- *EXCLUDED*. If an edge is *EXCLUDED*, it means that it is not part of the Hamilton cycle being constructed.

Secondly, lets describe the vertex states. There are 6 states that a vertex can be in.

- *UNDISCOVERED*. If a vertex is *UNDISCOVERED* it means that it is incident to only *UNDISCOVERED* edges.
- *CHOICE*. If a vertex is in *CHOICE* state, it means that it is incident to 2 *UNDISCOVERED* edges and one *INCLUDED* edge. Therefore, there is a choice as to which of the *UNDISCOVERED* edges will be *INCLUDED*.
- *DETERMINISTIC_INCLUDE_REST*. If a vertex is in this state, it means that it is incident to 2 *UNDISCOVERED* edges and one *EXCLUDED* edge. That means that the two *UNDISCOVERED* edges must become *INCLUDED*.
- *DETERMINISTIC_INCLUDE_REMAINING*. If a vertex is in this state, it means that it is incident to an *INCLUDED* edge, an *EXCLUDED* edge and an *UNDISCOVERED* edge which must be become *INCLUDED*.
- *DETERMINISTIC_EXCLUDE_REMAINING*. If a vertex is in this state, it means that it is incident to two *INCLUDED* edges and the third *UNDISCOVERED* one must become *EXCLUDED*.
- *COVERED*. If a vertex is *COVERED* it means that it is incident to two *INCLUDED* and one *EXCLUDED* edge.

For an illustration, see Figure 4.1.

Method *cycleConstructible* is recursive, therefore at each point it handles the graph in some state. State of a graph means, that all edges and vertices are in some state. Note that there may be many edges *INCLUDED*, yet not forming a single path or a

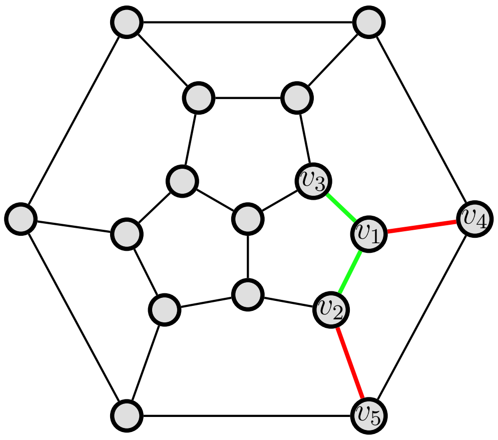


Figure 4.1: Included edges are coloured green, excluded red and undiscovered are black. In this example, v_1 is *COVERED* as none of its incident edges is undiscovered, v_2 is *DETERMINISTIC_INCLUDE_REMAINING* as it is incident to one edge of each state. v_3 is *CHOICE* as it is incident to only one included edge and the rest is undiscovered. Both v_4 and v_5 are *DETERMINISTIC_INCLUDE_REMAINING* as they are incident to one excluded edge and the rest is undiscovered. The remaining vertices are *UNDISCOVERED*.

cycle. That is why we need a way to tell in order not to close a cycle prematurely. For that purpose, we use a union-find algorithm enhanced by a *disunion* method hosted in *SimpleUnionFindSet* class. An union-find algorithm maintains elements in sets. Its *find* method can for any element tell what set it is in. Its *union* method can merge two sets into one. It is implemented as a tree, so the set of a vertex is the root of the tree it is in. As the algorithm is recursive, we need a way to disunion a set into two former sets when coming back from a recursive branch. Recursion has one neat feature, which is that a sequence of operations *union* and *disunion* always resembles a stack, therefore by using a tree implementation, we can easily disconnect a tree so that we are left with two former ones. A set in our case will represent a path in the graph.

The pseudo code in algorithm 4.3 uses simplified structure for explanatory purposes. In real implementation it is a little more tangled up. For the sake of completeness let's explain purpose of all the methods used in this pseudo code. First up, there is a method *cycleClosed*. It is a Boolean method which returns true, if there is Hamilton cycle constructed in the graph. Next, there is the *pollVertex* method which returns a vertex of the graph according to the following pattern. If there is a vertex in a *DETERMINISTIC* state, *pollVertex* returns such vertex. If not it returns a *CHOICE* vertex if there is one, otherwise it returns an *UNDISCOVERED* vertex. Vertex's *getUndiscoveredEdge* method returns any *UNDISCOVERED* edge incident to the vertex. The key method in this algorithm is *operationsFor*. For a given edge uv it returns a set of possible operations. For example, if u is *DETERMINISTIC_INCLUDE_REMAINING* and

v is *CHOICE*, then the returned set will only contain operation *INCLUDE*. If there is no mutual operation, it will return an empty set. Moreover, *INCLUDE* will be in the returned set only if u and v are from different paths or by including this edge, we close a valid Hamilton cycle. Finally, there are *include, undoInclude, exclude* and *undoExclude* methods, whose purpose is just to properly set the new states to the vertices and edges.

```
procedure cycleConstructible
    if cycleClosed = true then
        return true
    end if
    $vertex \leftarrow$ pollVertex
    $edge \leftarrow$ vertex.getUndiscoveredEdge
    $operations \leftarrow$ operationsFor($edge$)            ▷ $operations$ is a set
    $result \leftarrow$ false;
    if $operations$.contains(INCLUDE) then
        include($edge$)
        union($edge$)
        $result \leftarrow$ cycleConstructible
        disunion($edge$)
        undoInclude($edge$)
    end if
    if $operations$.contains(EXCLUDE) then
        exclude($edge$)
        $results \leftarrow$ cycleConstructible
        undoExclude($edge$)
    end if
    return $result$
end procedure
```

Having described the method, let's describe its use. According to lemma 3.12, if an edge is unavoidable then it must be ham-unavoidable. We want to use this method to determine if an edge of a graph is ham-unavoidable. This process is hosted by a class called *HamUnavoidableEdgeGetter*. It contains a method called *getResult* which returns all ham-unavoidable edges. Let's remind what it means that an edge is ham-unavoidable. It means that there exists no Hamilton cycle not containing it. That is precisely what we want to verify. If there exists a Hamilton cycle not containing the edge, we know for sure it is not ham-unavoidable. The implementation of the *getResult* is very simple. For a given edge, it excludes it from the Hamilton cycle, i.e., sets its state to *EXCLUDED* and calls the *cycleConstructible* method. If the method returns true, we know that the edge in question is not ham-unavoidable. Therefore, the result of the *getResult* method is a set of edges for which the *cycleConstructible* method returned false.

Similarly, according the lemma 3.13, if an edge is untouchable it must be untouchable. A class hosting *getResult* method for this verification is hosted in *HamUntouchableEdgeGetter*. In this case, we are looking for an edge that is not contained in any Hamilton cycle in the given graph. In other words, if there exists no Hamilton cycle containing the edge, then it is ham-untouchable. Therefore the implementation of the *getResult* method is again very simple. First, we include the edge in the Hamilton cycle and union the endvertices by union-find. The we call the *cycleConstructible* method. If it returns true, then the given edge is not ham-untouchable, otherwise it is.

There is a simple optimisation to be done. In the case of *HamUnavoidableEdgeGetter.getResult*, if we close a Hamilton cycle for a an edge, then all edges not contained in the Hamilton cycle, i.e., all *EXCLUDED* edges are not ham-unavoidable either as there exists a Hamilton cycle not containing them. For that reason, we do not need to verify the property for them again. Similarly for *HamUntouchableEdgeGetter.getResult*, if we close a Hamilton cycle, then all the edges contained in the cycle, i.e., all *INCLUDED* edges are not ham-untouchable either as there exists a Hamilton cycle containing them.

Let e be an edge of a graph G that is ham-unavoidable or ham-untouchable. To verify, whether e is unavoidable or untouchable, we need a signature of G. Starting with the case that e is ham-unavoidable, we want to establish which vertices must be positive. According to the lemma 3.14 a vertex v is forced positive, if there exists a Hamilton cycle in $G \setminus \{v\}$ not containing edge e. Otherwise, v is ambiguous by definition. An utility that returns a set of ambiguous vertices for a given ham-unavoidable edge is hosted in *getResult* method of*UnavoidableAmbiguousVertexGetter* class. The implementation of the method looks as follows. We do no physically remove v from G, but instead we exclude all its incident edges from the Hamilton cycle. Next we exclude e and call the *cycleConstructible* method. If the return value of the call is false, then v is ambiguous so we add it to the set that gets returned once we process all vertices of G by the *getResult* method.

Similarly for the case if e is ham-untouchable. According to lemma 3.15 a vertex v is forced positive if there exists a Hamilton cycle in $G \setminus \{v\}$ containing edge e. Otherwise, v is ambiguous by definition. An utility returning a set of ambiguous vertices for a given ham-untouchable edge is hosted in *getResult* method of *UntouchableAmbiguousVertexGetter* class. Firstly, we set all edges incident to v to *EXCLUDED* and then set e to *INCLUDED*. After that we call the *cycleConstructible* method. If it returns false, then v is ambiguous and we add it to the set of ambiguous edges that gets returned at the end of the *getResult* method.

The last step before semi-2-factor verification is to assign the underlying graph a signature. We have constraints on some vertices (forced positive vertices) and for the rest we have to verify all possible combinations (ambiguous vertices). This task is handled by a class named *VertexPolarityDistributor*. Its constructor method takes the underlying graph and the set of ambiguous vertices. The produced signatures of the underlying graph can be accessed through an iterator exposed by this class, where each iteration contains a different valid (the number of positive vertices is even) signature of the underlying graph.

Now that we described all utilities necessary to produce a vertex-signed cubic planar graph with a potentially unavoidable (ham-unavoidable) or untouchable (ham-untouchable) edge, let's proceed to the description of the semi-2-factor verification itself. Firstly, we need to describe the states of edges and vertices. When it comes to edges, there are precisely the states as there were during Hamilton cycle construction, however for vertices we need dedicated states for positive and negative vertices. For clarity, states of positive vertices will be prefixed with *P_* whereas states of negative vertices will be prefixed with *N_*.

- All the vertex states used in Hamilton cycle verification will be used by semi-2-factor builder as well. This will precisely be the states of positive vertices just prefixed by *P_*. For example, the formerly introduced state *CHOICE* will be called *P_CHOICE*.
- If a vertex is *N_UNDISCOVERED* all its incident edges are *UNDISCOVERED*.
- If a vertex is *N_END_PATH_CHOICE* it means that it is incident to two *UNDISCOVERED* edges and one *INCLUDED* edge.
- If a vertex is *N_IN_EX_CHOICE* it means that it is incident to two *UNDISCOVERED* edges and one *EXCLUDED* edge.
- If a vertex is *N_DETERMINISTIC_INCLUDE* it means that it is incident to one edge of each state.
- If a vertex is *N_DETERMINISTIC_COVER_EXCLUDE* it means that it is incident to one *UNDISCOVERED* edge and two *INCLUDED* ones.
- If a vertex is *N_DETERMINISTIC_EXCLUDE* it means that it is incident to one *UNDISCOVERED* edge and two *EXCLUDED* ones.
- If a vertex is *N_COVERED* it means that it is incident to one *EXCLUDED* edge and two *INCLUDED* ones.

- If a vertex is *N_EXCLUDED* it means that it is incident to three *EXCLUDED* edges.

When building two factors, we need to keep track of the paths and cycles built in the graph. For that purpose we use modified union-find set hosted in a class named *CountingUnionFindSet*. The main difference between the *CountingUnionFindSet* and *SimpleUnionFindSet* is that it it remembers the number of positive vertices in each path.

Finally, we can get to the component responsible for semi-2-factor verification. It all revolves around a method called *canBuildSemiTwoFactor* hosted in a class named *AbstractBuilder*. Its pseudo code is pretty much the same as the one in 4.3, however the behaviour of some used methods is altered. Firstly, the *cycleClosed* method returns true if there is a semi-2-factor built in the graph. Secondly, we need to address the *pollVertex* method as there are new vertex states. The pattern stays the same. If there is vertex in a state whose name contains the word *DETERMINISTIC*, then such vertex will get polled. If there is no such vertex a vertex whose state contains the word *CHOICE* will get polled, otherwise an *UNDISCOVERED* vertex would get polled. Lastly, we need to describe the *operationsFor* method. The principle stays the same as it returns non-conflicting actions to both endvertices, however there is a new constraint for edge inclusion, i.e., setting the edge state to *INCLUDED*. The edge can be included only if it will connect two distinct paths or if it closes a cycle containing an even number of positive edges.

We would like to show some statistics as to how effective the verification was. For more detail see Table 4.4. Compared to the Table 4.3, we can see how few graphs actually contain no reducible configurations. Moreover, the existence of an ham-unavoidable edge is very rare. Lastly, we can see that the number of ambiguous vertices is significantly lower than the size of the graph.

Depth	8	10	12	14	16	18	20	22	24	26	28
Graphs	1	0	0	2	3	5	20	60	242	1088	5180
Edges	0	0	0	0	3	2	1	12	45	168	713
Vertices	x	x	x	x	6	6	10	12.667	13.333	14.744	16.856

Table 4.4: The number of graphs at each depth that contain no reducible configuration, together with the total number of ham-unavoidable edges in all of them. Lastly, an average number of ambiguous vertices for an ham-unavoidable edge in graphs of given size.

Lastly, lets demonstrate, how long the verification takes compared to the generation. For more detail, see Table 4.5. We can clearly see that the time spent on verification is

Depth	12	14	16	18	20	22	24	26	28
Generation	31ms	47ms	73ms	132ms	355ms	1.03s	4.3s	23.2s	128.4s
Verification	31ms	51ms	105ms	165ms	428ms	1.16s	4.6s	25.2s	287.1s

Table 4.5: The length of generation compared to the length of generation and verification for each depth.

marginal compared to the time spent on generation up to the depth of 26. From there on, The verification seems to be the bigger expense.

4.4 Results

The main goal of this work was to find the smallest tripoles of given types. Prior to this work, we knew the smallest tripoles of type P_3, N_3^* and N_3. The minimal size of the rest was an open problem. Through verification of all 3-connected cubic planar graphs we came to the following results.

Lemma 4.1 *The smallest tripole of type P_2 has 27 vertices.*

Proof. Our algorithm did not find any positive tripoles of size smaller than 27 containing an unavoidable edge. □

In other words, we have verified that the tripole used by Kardoš and Narboni is the smallest tripole o type P_2. Together, we have found four distinct graphs for which there exists a signature such that they contain an unavoidable edge, i.e., by removal of a positive vertex incident to that edge we get a tripole of type P_2. Moreover, both vertices incident to that edge were in each case positive, therefore we get 8 non-isomorphic tripoles of type P_2. All of these tripoles have one thing in common. They are Tutte's fragment with a different placements of cubes (tripoles of type P_3). This was an expected result. The only possibility how there could be a smaller tripole of this type is, if there was a graph with greater number of vertices than Tutte's fragment that would need less cubes than 2 to be substituted into. However, we have not found such graph, hence the smallest tripole of type P_2 is of size 27.

Lemma 4.2 *The smallest tripole of type N_2^* has 27 vertices.*

Proof. Our algorithm did not find any negative tripoles of size smaller than 27 containing an unavoidable edge while admitting a semi-2-factor not containing any semi-edges. □

Similarly as for tripoles of type P_2 we exhaustively searched the class of 3-connected cubic planar graphs and found no tripoles smaller than 27. As we mentioned in the remark after the previous lemma, there are 4 distinct graphs that result in 8 distinct tripoles of type P_2. There is an interesting observation. For graphs for which exists a signature resulting in tripoles of type P_2 exists a signature such that the resulting tripoles are of type N_2^*. In other words, let G be a vertex-signed cubic planar graph of size 28 and type P_2. Then then there exists a different signature of G such that the tripoles produced from G are of type N_2^*. That means that both vertices incident to an unavoidable edge are negative which results in 8 distinct tripoles of type N_2^*. This is a new result as these tripoles have not been known before.

Lemma 4.3 *The smallest tripole of type N_2 has 29 vertices.*

Proof. Our algorithm has not found any tripoles of type N_2 with 27 or less vertices. By the claims of the lemma 2.15, we know that we can produce a tripole of type N_2 from P_2 by adding two vertices. As the smallest tripole of type P_2 has 27 vertices, a tripole of type N_2 obtained through this means has 29 vertices. □

We have not verified that these are the only tripoles of type N_2 of size 29, but we know that there are none smaller than that.

Chapter 5

Conclusion and remarks

The goal of the work was achieved as we found tripoles of types that were not discovered before, i.e., N_2^* of size 27 and N_2 of size 29. Moreover, the set of reducible configurations can be extended to contain all the tripoles that are not the smallest known tripoles of each type.

Unfortunately, we were not able to find a smaller tripole of type P_1, N_1^* or N_1; or verify if the one known is the smallest one. The reason for that is simple. As seen in the table 4.3, the number of graphs at each depth n is exponential with n even if we account for all the known optimisations. Moreover, due to the need to remember all the canonical codes because of symmetries, the program ran out of heap memory at depth 32.

We have observed, that if we restrained operation-3, we could generate all the 3-connected planar cubic graphs that contain no non-trivial tripoles. As there are very few non-trivial tripoles that are irreducible to a smaller tripole with the same properties, we propose the following algorithm.

Firstly, we need to account for orbits, i.e., classes of symmetry in a given graph. If we then perform the operations for only one element of each orbit and strictly follow the definitions of the parent-child relationship, we do not need to bother with remembering all the canonical codes.

Moreover, the aforementioned observation that each non-trivial tripole can be reduced into a smaller one of the same type can be abused in our favour as well. From the global point of view, a tripole has the same interface for the rest of the graph as a single vertex, i.e., its three semi-edges. Therefore, instead of building tripoles manually, encountering many that are reducible in the end, we can do the following. Each vertex will carry additional values, i.e., the type of the tripole it represents. If it represents a tripole of type P_2, N_2^* or N_2 it will also remember which of its edges is the unavoidable one.

This approach has 2 benefits. First one is obvious, as we only need to search a smaller class of graphs. Secondly, the verification will also be enhanced as we do not need to verify all the possibilities how the semi-2-factor can traverse the inside of a tripole.

Bibliography

[1] K.I. Appel and W. Haken. *Every Planar Map is Four Colorable.* Contemporary mathematics. American Mathematical Society, 1989.

[2] Reinhard Diestel. *Graph Theory.* Springer-Verlag, 2005.

[3] František Kardoš and Jonathan Narboni. On the 4-color theorem for signed graphs. *European Journal of Combinatorics*, 91:103215, 2021. Colorings and structural graph theory in context (a tribute to Xuding Zhu).

[4] Edita Máčajová, André Raspaud, and Martin Škoviera. The chromatic number of a signed graph. *The Electronic Journal of Combinatorics*, pages P1–14, 2016.

[5] Neil Robertson, Daniel Sanders, Paul Seymour, and Robin Thomas. The four-colour theorem. *Journal of Combinatorial Theory, Series B*, 70(1):2–44, 1997.

[6] Thomas Zaslavsky. Signed graphs. *Discrete Applied Mathematics*, 4(1):47–74, 1982.