Recursive Algorithms for Generation of Planar Graphs

Mohammadreza Jooyandeh

A thesis submitted for the degree of
PhD of Computer Science at
The Australian National University

Supervised by Prof. Brendan D. McKay

September 2014

Except where otherwise indicated, this thesis is my own original work.

Mohammadreza Jooyandeh
6 September 2014

to Fateme

Acknowledgments

This thesis would certainly not have been possible without the help and support of many people. Firstly, I want to express my special appreciation and thanks to my supervisor Brendan McKay. There are too many reasons for this which I am incapable of expressing them all. You guided me through my research while letting me be myself and grow. You had been there always for me and whenever I needed some wisdom, I knew I could count on you. Also, I want to thank you for the belief you had in me which was the reason I could trust my path towards my passion and follow it. Your advice on both research as well as on my career have been priceless. I am very thankful for all opportunities I gained because of you recommending me.

I would like to thank the members of my jury for reading my thesis and for their suggestions to further improve. I am thankful to Prof. Weifa Liang and Dr. Pascal Schweitzer for serving as my panel members. Weifa, I appreciate the opportunity you gave me to be the tutor of your course from which I learnt so much.

Also, I am more than grateful to my beloved wife Fateme, who has supported me in ups and downs. She encouraged me whenever I felt too weak, wanted to give up and stop challenging difficulties. She has offered me a listening ear whenever I needed and I can never thank her enough. A special thanks also goes to my family and my in-laws for their tremendous support and motivation they gave to me during this journey. Your prayer for me was what sustained me this far. Also, I want to thank Miriam who has been like a mother to me and helped me a lot specially in my early times in Australia while I needed the most. I am also thankful to my special friend Ehsan who has helped me a lot in the difficulties and been so much more than a good friend. Ehsan, being your friend has been exhilarating.

During my PhD I had the honour of collaborating with Prof. Gunnar Brinkmann. My sincere thanks goes to him as I have learnt so much from him. He had been a great support and helped me a lot while he was extremely busy in my visit period. I also want to thank Annelies, Gunnar, Hossein, Jan and Maryam for the great time I had in Ghent.

I would like to thank Mrs. Chelsea Holton and Mrs. Julie Arnold who were great with my administration tasks. Without you, I would have been in so much trouble.

Last but not the least, I want to thank my friends and all the people who have been beside me during this marathon in Australia. I would like to thank Alireza K., Alireza M., Anita, Amir, Ashkan, Beáta, Behrooz, Behzad, Ben, Brendan, Cathrine, Duncan, Ehsan A., Ehsan N., Fahimeh, Hamid, Ivan, Jeanette, Josh, Keane, Michelle, Miriam, Mohammad D., MohammadAli, Monique, Morteza, Muhammad I., Pascal, Qiufen, Quentin, Rohan, Sahba, Shahriar, Tom, Vicki, Zahra M., Zahra Z., Zan and Zichuan; and my team mates in ANU Phoenix, CECS United and CompSci Chumps.

Abstract

In this thesis we introduce recursive algorithms for generation of two families of plane graphs. These algorithms start with small graphs and iteratively convert them to larger graphs. The families studied in this thesis are k-angulations (plane graphs whose faces are of size k) and plane graphs with a given face size sequence.

We also design a very fast method for canonical embedding and isomorphism rejection of plane graphs. Most graph generators like *plantri* generate graphs up to isomorphism of the embedding, however our method does the isomorphism checking up to the underlying graph while taking advantage of the planarity and embeddings to speed up the computation.

The next subject discussed in this thesis is a type of graph called *hypohamiltonian* in which after removing each vertex from the graph, there is a Hamiltonian cycle through all remaining vertices while the original graph does not have any such cycle. One of the problems in the literature since 1976 is to find the smallest planar hypohamiltonian graphs. The previous record by Weiner and Araya was a planar graph with 42 vertices. We improve this record by finding 25 planar hypohamiltonian graphs on 40 vertices while discovering many larger ones on 42 and 43 vertices.

The final subject in the thesis is a family of molecules called *fullerenes* which are entirely composed of carbon atoms. The structure of fullerenes are 3-connected plane graphs with exactly 12 faces of size 5 and the rest of size 6. A famous conjecture regarding fullerenes, called face-spiral conjecture claims that the drawing of their graph can be unwound in a spiral manner starting from one face and circulating around that face until all faces are traversed. This conjecture is known to be incorrect and the smallest counterexample is made of 380 carbon atoms. We have extended this conjecture to families of 3-connected plane graphs with f_3, f_4 and f_5 faces of size 3, 4, and 5 while the remaining faces have size 6 and found counterexample for all possible values of $\langle f_3, f_4, f_5 \rangle$. We also found the smallest counterexamples for 11 of these families out of 19 possible cases.

Contents

List of Figures

List of Tables

List of Algorithms

Preface

In this thesis we introduce recursive algorithms for generation of structures which can be used in mathematics and chemistry. Each structure is presented as a mathematical model called a *graph* which incorporates relations between pairs of objects. Each graph contains a set of vertices and a set of edges. The vertices can represent some objects and edges connect related objects to each other. For example a graph can model a map in which edges represent roads and vertices are end-points of the roads. Another example could be molecules in which atoms are vertices and edges are their bindings. In this chapter we mainly deal with a family of graphs called *planar graphs* which can be drawn on the plane with vertices drawn as points and edges as arcs joining their end-points such that the edges can only cross each other at their end-points. Each drawing divides the plane into regions which are called *faces*.

The recursive algorithms which we present in this thesis are methods that repetitively follow a procedure. These algorithms start with small structures (graphs) and iteratively convert them to larger ones to construct desired structures for some specific problems.

The second subject tackled in this thesis is towards one of the traditional problems in mathematics and computer science i.e., *isomorphism*. The isomorphism problem discuss whether two structure are “equivalent” under a some particular definition for equivalence.

The third subject discussed in this thesis is a type of graph called *hypohamiltonian* in which after removing each vertex from the graph, there is a cycle going through all remaining vertices using edges without passing vertices more than once, but the original graph does not have this property. This graph class has been studied since 1963 [103] and is interesting both in mathematics and in computer science.

The final subject in the thesis is towards extending the boundary in the literature of a family of molecules called *fullerenes* which are entirely composed of carbon atoms. The structure of fullerenes are planar graphs and each drawings of them has exactly 12 faces made of five edge while the other faces have six edges. The construction of these structures has started in 1985 [72]. Also there is a famous conjecture regarding fullerenes, called face-spiral conjecture which claims that the drawing can be unwound in a spiral manner [81] starting from one face and circulating around that face until all faces are traversed. This conjecture is known to be incorrect [80] and the smallest counterexample has is made of 380 carbon atoms [17]. We have extended this conjecture to some other families of graphs and found counterexamples for all of them.

Chapter 1

Introduction

1.1 Definitions

graph
incidence
end-point
$V(G)$
$E(G)$
$I(G)$
order

A *graph* G is an incidence structure triple (V, E, I) where the elements of V and E are called vertices and edges of the graph. Also, $I \subseteq V \times E$ is a relation which defines the *incidence* of vertices and edges with the condition that every edge is incident to one or two vertices, called its *end-point(s)*. The sets of vertices, edges and the incidence relation of G are denoted by $V(G)$, $E(G)$ and $I(G)$. The *order* of G, written as $o(G)$, is the number of vertices of G i.e., $o(G) = |V(G)|$. In some cases for sake of convenience we may define a graph as a pair (V, E) in which $E \subseteq \{\{u, v\} : u, v \in V\}$. In these cases a vertex v and an edge e are incident if $v \in e$. For simplicity, by abuse of language we may say $v \in G$ and $e \in G$ instead of $v \in V(G)$ and $e \in E(G)$.

loop
loopless graph
adjacent
neighbours
simple graph
multigraph

A *loop* is an edge which has only one end-point and a graph with no loops is called a *loopless graph*. Two vertices of a graph are called *adjacent* or *neighbours*, if they have an edge incident to both of them. A graph is called *simple* if its loopless and for every pair of adjacent vertices, there is exactly one edge incident to both of them; otherwise it is called a *multigraph*.

$N_G(v)$
degree
valency
$d_G(v)$
$\Delta(G)$
$\delta(G)$
k-regular graph
regular graph
cubic graphs

The set of neighbours of a vertex v is denoted by $N_G(v)$ or $N(v)$. The *degree-* or *valency* of a vertex v in G, denoted by $d_G(v)$ or $d(v)$ is the number of edges incident to v. A vertex with degree t is called a t-valent vertex. The maximum and minimum degree of a graph G are denoted by $\Delta(G)$ and $\delta(G)$ or for simplicity Δ and δ, respectively. A graph whose vertices have the same degree k is called a *k-regular graph*, or *regular graph* if we do not wish to identify k. The family of 3-regular graphs sometimes are referred to as *cubic graphs*.

walk
length of a walk
closed walk
open walk
path
cycle
k-path
k-cycle

A *walk* in a graph G is an alternating sequence of vertices and edges, begining and ending with a vertex, $v_1 e_1 v_2 e_2 \cdots v_n$ such that for all $1 \leqslant i < n$, v_i and v_{i+1} are the end-points of e_i, if it is not a loop; otherwise $v_i = v_{i+1}$ is the end-point of e_i. The *length of a walk* is the number of edges (with multiplicity) of that walk. Note that the length of a walk could be zero. A walk is *closed* if its first and last vertices are the same and *open* otherwise. A walk is called a *path* if its vertices are distinct and a *cycle*, if its vertices are all distinct except the first and last vertices which are the same. A path and a cycle with k edges are called a *k-path* or a *k-cycle*, respectively.

complete graph
$\mathbf{K}_T$
$\mathbf{K}_n$
path graph
$\mathbf{P}_n$

The *complete graph* $\mathbf{K}_T$ is the graph (T, E) whose vertices are adjacent pairwise. For simplicity $\mathbf{K}_{\{1,2,\cdots,n\}}$ is written as $\mathbf{K}_n$. The *path graph* $\mathbf{P}_n$ is the graph whose vertices are $1, 2, \cdots, n$ with two vertices v and w are adjacent if $|v - w| = 1$. The *cycle graph* $\mathbf{C}_n$ has the same vertex set as the path graph on n vertices but two vertices v and w are adjacent, if $|v - w| = 1 \mod n$.

cycle graph
$\mathbf{C}_n$

A *subgraph* H of a graph G, written as $H \subseteq G$ is a graph with $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$ and $I(H) = I(G) \cap (V(H) \times E(H))$ such that $V(H)$ includes end-points of edges in $E(H)$. Moreover, H is called an *induced subgraph* of G, if H is a subgraph of G and for every $e \in E(G)$ with end-points u and v ($u = v$ if it is a loop), $e \in E(H)$ if $u, v \in V(H)$. The induced subgraph of G with vertices $V' \subseteq V(G)$ or $E' \subseteq E(G)$ is written as $G[V']$ or $G[E']$, respectively.

subgraph
induced subgraph

Let $G_1 = (V_1, E_1, I_1)$ and $G_2 = (V_2, E_2, I_2)$ be two graphs. The *union*, denoted by $G_1 \cup G_2$ is the graph $(V_1 \cup V_2, E_1 \cup E_2, I_1 \cup I_2)$. The *disjoint union*, written as $G_1 \dot{\cup} G_2$ is the graph $(V_1 \times \{1\} \cup V_2 \times \{2\}, E_1 \times \{1\} \cup E_2 \times \{2\}, I)$ in which $I = \bigcup_{i=1}^{2} \{((v, i), (e, i)) : (v, e) \in I_i\}$. If G is a graph, $G' \subset G$ and $V' \subset V(G)$ and $E' \subset E(G)$, then $G \backslash G'$ and $G \backslash V'$ are the induced subgraph $G[V(G) \backslash V(G')]$ and $G[V(G) \backslash V']$, respectively, but for $E \subset E(G)$, $G \backslash E'$ is the graph $(V(G), E(G) \backslash E', I(G) \backslash V(G) \times E')$ i.e., the graph obtained from G by removing edges of E'. When $V' = \{v\}$ or $E' = \{e\}$, we might simply write $G \backslash \{v\}$ and $G \backslash \{e\}$ as $G \backslash v$ and $G \backslash e$, respectively.

union of graphs
disjoint union of graphs
$G \backslash G'$
$G \backslash V'$
$G \backslash E'$

A graph G is said to be *connected*, if there is a path between every pair of vertices in G, otherwise it is *disconnected*. A graph G is *k-connected*, if $|V(G)| > k$ and for every $S \subset V(G)$ with $|S| < k$, $G \backslash S$ is connected. A subset C of vertices of a graph G whose removal makes G disconnected is called a *cut-set*. The *connectivity* of a G, denoted by $\kappa(G)$ is defined as the size of one of its minimal cut-sets, if G is not a complete graph. The connectivity of $\mathbf{K}_n$ is defined to be $n - 1$ for convenience. Graphs with connectivity at least two and three are called *biconnected* and *triconnected*, respectively. The maximal connected subgraphs of G are called its *components*.

connected
disconnected
k-connected
cut-set
connectivity
$\kappa(G)$
biconnected
triconnected
component

An *abstract isomorphism* between two graphs $G_1 = (V_1, E_1, I_1)$ and $G_2 = (V_2, E_2, I_2)$ is a bijection $\pi : V_1 \cup E_1 \to V_2 \cup E_2$ such that:

abstract isomorphism

- $\pi(V_1) = V_2$.
- $\pi(E_1) = E_2$.
- $\forall v \in V_1\ \forall e \in E_1 : (v, e) \in I_1 \Leftrightarrow (\pi(v), \pi(e)) \in I_2$.

1.2 Graph Embedding and Plane Graphs

In this section the notion of embedding used in this thesis is explained. The definition used for embedding is based on [74] and is known as the rotation system which is based upon putting an ordering on the edges incident with each vertex but before starting the combinatorial definition we introduce a topological definition for drawing. Then, after the combinatorial definition of rotation system, we show that these two definitions are equivalent.

Definition 1.1. *A* map *M for a graph G is a drawing into an orientable surface X such that:*

map

- *Vertices of G are represented as distinct points in X.*

- *Edges are represented as simple curves in the surface whose end-points are their incident vertices with the condition that edges can intersect only at their end-points.*

face of map

With such a drawing we can consider G as a subset of X. So $X \backslash G$ is a disjoint union of maximal connected components, these components are called faces *of G in that map.*

D_E
dart relation
dart
inverse dart
$\text{inv}(x)$
$\text{head}(x)$
$\text{tail}(x)$

Let $G = (V, E, I)$, and let D_E be a relation, called the *dart relation*, which maps each edge $e \in E$ to two directed edges e^1 and e^2 called *darts* of e with the same end-points as e, but in opposite directions. These two darts are said to be the *inverses* of each other and the inverse of a dart x is denoted by $\text{inv}(x)$. The end-points of a dart x are called its *head* and *tail*, denoted by $\text{head}(x)$ and $\text{tail}(x)$, note that:

$$\begin{aligned}
&\text{inv}(\text{inv}(x)) = x, \\
&\text{head}(x) = \text{tail}(\text{inv}(x)), \\
&\text{tail}(x) = \text{head}(\text{inv}(x)).
\end{aligned}$$

Every dart is said to be incident to its tail and the set of all darts incident to a vertex are called the darts of that vertex. We use the term $\vec{uv}$ to indicate a dart from vertex u to v. The set of all darts of G is denoted by $D_E(G)$.

rotation
rotation system
embedding

Definition 1.2. *Let $G = (V, E, I)$ be a graph and D_E be a dart relation for G. A* rotation *for a vertex v, written as* $\text{rot}(v)$*, is a permutation made of only one cycle on all the darts directed out of v. A* rotation system *or an* embedding *of G is a pair of permutations (α, σ) on $D_E(G)$. The permutation α is made of cycles of length two that map each dart of G to its inverse. The permutation σ is obtained from the product of rotations of all vertices i.e.,* $\sigma = \prod_{v \in V} \text{rot}(v)$.

next dart
previous dart
$\text{next}(e)$
$\text{prev}(e)$
$\text{next}_S(e)$
$\text{prev}_S(e)$

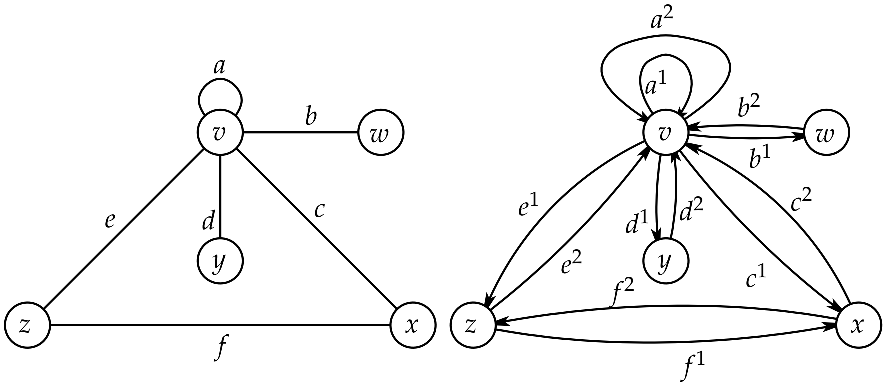
Let (α, σ) be a rotation system of a graph G. The *next dart* and *previous dart* of a dart e are defined as $\text{next}(e) = \sigma(e)$ and $\text{prev}(e) = \sigma^{-1}(e)$, respectively. Furthermore, $\text{next}_S(e)$ and $\text{prev}_S(e)$ are the first dart in the sequence $\text{next}(e), \text{next}^{(2)}(e), \text{next}^{(3)}(e), \ldots$ and $\text{prev}(e), \text{prev}^{(2)}(e), \text{prev}^{(3)}(e), \ldots$ whose head is in the set S, respectively.

Example 1.3. Consider the graph presented in Figure 1.1(a) and its directed version (Figure 1.1(b)). The rotation system of this graph can be presented as:
$\sigma = (a^1 a^2 b^1 c^1 d^1 e^1)(b^2)(c^2 f^2)(d^2)(e^2 f^1)$ and $\alpha = \prod_{e \in E}(e^1 e^2)$.

face of rotation system

Definition 1.4. *Let (α, σ) be a rotation system for a graph G and $\phi = \alpha^{-1}\sigma^{-1}$. Then the cycles of ϕ are called* faces *of G for this rotation system and the edges of each cycle are called the edges of that face.*

Maps and rotation systems are equivalent i.e., each map can be converted to a rotation system and vice versa. Building rotation systems from maps is quite straightforward. We explain this approach with an example. Consider the graph G which is drawn in Figure 1.1(a), to build a rotation system for this map, first we define the relation D_E which maps every edge e to darts e^1 and e^2. Then we can reconstruct the embedding as follows. The permutation α can be defined as $\alpha = \prod_{e \in E}(e^1 e^2)$



(a) (b)

Figure 1.1: Rotation System

which maps each dart to its inverse. To define the permutation σ we use the fact that $\sigma = \prod_{v \in V(G)} \mathrm{rot}(v)$ (by Definition 1.2) and define $\mathrm{rot}(v)$ for each vertex. The rotation of a vertex v can be defined as a cycle containing every dart going out of v in clockwise order. So for the graph of Figure 1.1(b), $\sigma = (a^1a^2b^1c^1d^1e^1)(b^2)(c^2f^2)(d^2)(e^2f^1)$ which gives us a rotation system based on the map and the dart mapping relation.

Inversely, we can convert an embedding to a drawing on a surface. Let G be a graph and (α, σ) be a rotation system for it. Also assume $\phi = \alpha^{-1}\sigma^{-1}$ which is made of cycles of faces. Now we can form a disk for each rotation face whose boundary is the edges of that face in the order of its cycle. Then, we can glue all disks based on the inverse relation of darts. Assume e^1 and e^2 are two darts which are inverses of each other. Then we attach the disks of faces that include e^1 and e^2 along the part of the boundary which corresponds to these darts. This gives us a surface which the graph is mapped to. Finally, we choose the orientation of the surface such that the darts in the rotations of vertices are in clockwise ordering.

Theorem 1.5 ([74, chap. 1.3]). *Let (α, σ) be a rotation system for a graph G which is constructed from a map and let $\phi = \alpha^{-1}\sigma^{-1}$. Then each cycle of ϕ (faces of the rotation system) is the cycle of darts of a face of the corresponding map in clockwise order.*

Theorem 1.5 allows us to use the term *face* both for rotation system faces and map faces. Also we may refer a face as a *k-face* or say its size is k if the length of its corresponding cycle in the rotation system is k.

face

k-face

In simple graphs a pair of adjacent vertices can uniquely determine the edge between them. We may use this property for the convenience of argument in some part of this thesis and define faces as cycles of vertices.

A careful look to the previous Theorem shows that if e is a dart such that the face on its right hand side is f, then $\phi(e)$ is the next dart in the clockwise traversal of f after edge e. The reason is that the next clockwise dart on the face after the edge e is $\mathrm{prev}(\mathrm{inv}(e))$.

Example 1.6. In the rotation system of Example 1.3 there are three faces namely:

$$\phi = (a^1)(a^2e^1f^1c^2b^1b^2)(c^1f^2e^2d^1d^2)$$

It should be noted that for the outer face, the order of the edges might give the impression that they are in counter-clockwise order but actually this is not true because the embedding is on the sphere so we should view the outer face from the other side of the sphere. Also one can easily check that in the face $(c^1f^2e^2d^1d^2)$ each dart e is mapped by ϕ to $\text{prev}(\text{inv}(e))$ for example $\text{prev}(\text{inv}(c^1)) = \text{prev}(c^2) = f^2$ and $\text{prev}(\text{inv}(f^2)) = \text{prev}(f^1) = e^2$.

planar graph
plane graph
underlying graph of a plane graph

A graph is called *planar*, if it has an embedding on the plane. A *plane graph G* is a tuple $(V, E, I, D_E, \alpha, \sigma)$ in which (V, E, I), called the *underlying graph*, is a planar graph, D_E is a dart set and (α, σ) is a rotation system for that graph. For the sake of convenience we may refer to a plane graph as just the underlying graph, a triple (V, E, I), if we do not intend to use the actual embedding.

Theorem 1.7 (Euler Formula). *Let $G = (V, E, I, D_E, \alpha, \sigma)$ and F be the set of faces of G. Then $|V| - |E| + |F| = 2$ if and only if G is a plane graph.*

mirror image
Mir(G)

Definition 1.8. *The* mirror image *of a rotation system (α, σ) of a graph G is the rotation system (α, σ^{-1}) and is denoted by* Mir(G).

order-preserving isomorphism

An *order-preserving isomorphism* between plane graphs $G_1 = (V_1, E_1, I_1, D_{E_1}, \alpha_1, \sigma_1)$ and $G_2 = (V_2, E_2, I_2, D_{E_2}, \alpha_2, \sigma_2)$ is a bijection $\pi : V_1 \cup E_1 \cup D_{E_1}(G_1) \to V_2 \cup E_2 \cup D_{E_2}(G_2)$ such that:

1. $\pi_{|V_1 \cup E_1}$ is an abstract isomorphism between G_1 and G_2.
2. $\forall e \in E_1\ \forall e' \in D_{E_1}(e) : \pi(e') \in D_{E_2}(\pi(e))$.
3. $\forall e \in D_{E_1}(G_1)\ \forall v \in V_1 : e \in \text{rot}(v) \Rightarrow \pi(e) \in \text{rot}(\pi(v))$.
4. $\forall e \in D_{E_1}(G_1) : \pi(\alpha_1(e)) = \alpha_2(\pi(e))$.
5. $\forall e \in D_{E_1}(G_1) : \pi(\sigma_1(e)) = \sigma_2(\pi(e))$.

order-reversing isomorphism
plane isomorphism

Similarly, an *order-reversing isomorphism* between them is a bijection π with the same properties except the last one and instead as its name suggests it reverses the orders: $\forall e \in D_{E_1} : \pi(\sigma_1(e)) = \sigma_2^{-1}(\pi(e))$. Finally, a *plane isomorphism* between two plane graphs is a bijection that is an order-preserving or/and order-reversing isomorphism.

Theorem 1.9 (Whitney's Theorem [122]). *For two 3-connected plane graphs, abstract isomorphism and plane isomorphism are equivalent.*

Let G be a plane graph with rotation system (α_G, σ_G) and $\phi_G = \alpha_G^{-1}\sigma_G^{-1}$. Also assume F_G is the set of faces of G (cycles of ϕ_G). Now, the dual of G, denoted by G^* is the plane graph defined as follows. The vertex set of G^* is F_G, the set of darts of G^* has a bijection $\perp$ to D_E which maps a dart $e \in D_E$ to $e_\perp$ such that if $e \in f \in F_G$ and inv$(e) = e' \in f' \in F_G$, then in G^*, $e_\perp \in \text{rot}(f)$ and $e'_\perp \in \text{rot}(f')$. The rotation system of G^* is defined as $(\alpha_{G^*}, \sigma_{G^*})$ in which $\alpha_{G^*}(e_\perp) = \alpha_G(e)_\perp$ and $\sigma_{G^*}(e_\perp) = \phi(e)_\perp$.

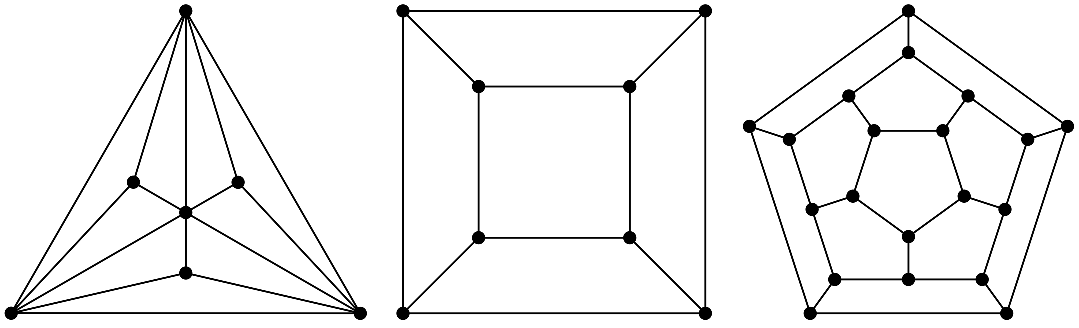
mple closed curve

A *simple closed curve* is the image of circle under an injective continuous $\mathbb{R}^2 \to \mathbb{R}^2$ map. The *Jordan curve theorem*, shorten as *JCT*, indicates that if C is a simple closed curve, $\mathbb{R}^2 \backslash C$ is made of exactly two connected components, the interior and the exterior of the curve which are bounded by C [65]. We may refer to a closed curve made of edges of a k-cycle which is a not the boundary of a face as a *separating k-cycle*.

Jordan curve theorem
JCT
separating k-cycle

If G is a connected plane graph with $f = \sum_{i=1}^{t} f_i$ faces such that f_i of them are of size k_i, then G is called a *$\{(k_1, f_1), \ldots, (k_t, f_t)\}$-angulation*. If the counts of the faces are not important it may be refered as a *$\{k_1, \ldots, k_t\}$-angulation*. In the case that all the faces have the same size it is written as a *(k, f)-angulation* or simply a *k-angulation*. A $\{k_1, \ldots, k_t\}$-angulation is *trivial* if it is a cycle graph $\mathbf{C}_k$, i.e. a $(k, 2)$-angulation. The set of all simple $(3, f)$-angulations (triangulations) and $(4, f)$-angulations (quadrangulations) are written as $\mathcal{T}_f$ and $\mathcal{Q}_f$, respectively. Figure 1.2 presents a $(3, 10)$-angulation, a $(4, 6)$-angulation and a $(5, 12)$-angulation.

$\{(k_1, f_1), \ldots, (k_t, f_t)\}$-angulation
$\{k_1, \ldots, k_t\}$-angulation
(k, f)-angulation
k-angulation
trivial k-angulation
$\mathcal{T}_f$
$\mathcal{Q}_f$



(a) $k = 3$ and $f = 10$ (b) $k = 4$ and $f = 6$ (c) $k = 5$ and $f = 12$

Figure 1.2: Sample (k, f)-angulations

1.3 Recursive Generation of Combinatorial Objects

Assume $\mathcal{F}$ is a family of graphs and $\mathcal{U}$ is a superset of $\mathcal{F}$, $\mathcal{I} \subset \mathcal{U}$ and $\mathcal{E}$ a set of functions from $\mathcal{U}$ to $2^{\mathcal{U}-\mathcal{I}}$. Then $(\mathcal{I}, \mathcal{E}, \mathcal{U})$ is said to *recursively generate* $\mathcal{F}$, if for every graph $G \in \mathcal{F}$ there is a finite sequence $G_0, \ldots, G_n = G$ (n could be 0) such that $G_0 \in \mathcal{I}$ and for all $i < n$, $G_{i+1} \in X(G_i)$ for some $X \in \mathcal{E}$. The members of $\mathcal{E}$ are called *expansions*. Moreover, for each expansion X the function R_X maps every graph G to all graphs G' that can be expanded to G (the set can be empty) using X; mathematically speaking $R_X(G) = \{G' \in \mathcal{U} : G \in X(G')\}$. These functions are called *reductions*. Members of $\mathcal{I}$ and $\mathcal{U} - \mathcal{I}$ are called *irreducible graphs* and *reducible graphs*, respectively. Also for a graph $G \in \mathcal{U}$, all $G' \in R_X(G)$ for some $X \in \mathcal{E}$ are called *parents* of G.

recursive generation
expansions
reductions
irreducible graphs
parents

In most studies like [21, 24, 22, 18, 23, 51, 52] the sets $\mathcal{U}$ and $\mathcal{F}$ in the definition above are the same, which means the generator starts with some irreducible graphs from the desired family and expands them to larger ones, while remaining inside the family. Having $\mathcal{U} = \mathcal{F}$ has the advantage that it usually needs fewer intermediate graphs which helps the performance of generators. The new definition might be slower, but allows more options for finding generators. We employ this new definition to make recursive generators for k-angulations (Chapter 2), $\{k_1, \cdots, k_t\}$-angulations (Chapter 3) and a family of planar graphs called 4-face deflatable hypohamiltonian graphs (Chapter 5).

1.4 Isomorph-Free Generation

isomorph-free

A generation is called *isomorph-free* if no pair of objects generated by the generator are isomorphic. The act of removing isomorphic copies in the process of generation is called *isomorph rejection*. There are some different isomorph rejection methods used in the literature which we discuss later in this section.

isomorph rejection

Let S be a set of combinatorial objects with a designated isomorphism. We can define a group Γ whose action on an object $s \in S$ gives us all isomorphic copies of S. We call a function I whose domain is S an *invariant*, if $I(s) = I(s^g)$ for every $g \in \Gamma$. More specifically, an invariant c which maps members of S to vectors of elements in a totally ordered set with the criteria that $c(s_1) = c(s_2) \Leftrightarrow \exists g \in \Gamma : c_1^g = c_2$ is called a *canonical code*. In this thesis, we use the lexicographic ordering for comparing canonical codes.

invariant

canonical code

The way that invariants and canonical codes are defined allows us to use them for isomorphic checking of combinatorial objects. Let I and c be an invariant and a canonical code for a set S. Then $s_1, s_2 \in S$ are isomorphic if and only if $c(s_1) = c(s_2)$. Also if $I(s_1) \neq I(s_2)$, then s_1 and s_2 are not isomorphic. So in practice for isomorphism testing, we can define some easily computable invariants and whenever we wish to check whether s_1 and s_2 are isomorphic, check those invariants first and use canonical codes only when all invariants had the same values for both of them. A *canonical labelling* is an invariant which maps each graph to a specific labeling of it i.e., maps graphs to their isomorphic classes. A graph is called *canonical* if the canonical labelling maps it to itself.

canonical labelling

canonical graph

The most trivial approach for isomorph rejection is to keep all generated objects in a memory, preferably RAM, and then compare them pairwise and output only one copy from each isomorphism class. This approach is of course only useful in limited cases when the set of generating objects is not too big i.e., isomorphic testing and keeping them all in memory is plausible for example it is used in [47] for generation of small sets of irreducible objects. We also use this approach in some parts of Chapter 5. Most graph generators that we deal with generate every labelling of each graph. Therefore, if we only output the canonical ones, then the output will contain one copy from each isomorphism class.

orderly generation

The second method called *orderly generation* is a recursive generation with an extra criteria on the definition of canonical code which is inherited to a sub-object which is considered as its canonical parent. This method was introduced independently by Faradzev [37, 36] and Read [98]. Some of many examples of using this method are [12] for generation of cubic graphs, [86] for generation of regular graphs and [95] for generation of extended binary trees i.e., rooted binary trees in which every vertex has either two or no children.

The next method called canonical construction path will be discussed in more details in Section 1.4.1.

1.4.1 Canonical Construction Path

In 1998 McKay introduced a new method called *canonical construction path* (*CCP*), which does not involve explicit isomorphism testing [83]. We will use this method in Chapters 2 and 3 to recursively generate some families of k-angulations and $\{k_1, \cdots, k_t\}$-angulations.

canonical construction path

CCP

We explain this approach parallel to an actual example of generating simple graphs and use labels "General" and "Graphs" to indicate the context.

General: We have a set of objects $\mathcal{L}$, called *labelled objects* with a group Γ acting on $\mathcal{L}$ whose orbits are called *unlabelled objects*, $\mathcal{U}$. Each labelled object X has a *order* $o(X) \in \mathbb{N}$ which is constant for objects in an orbit of $\mathcal{L}^{\Gamma}$. This allows us to define the *order* of unlabelled objects to be the common order of its comprising labelled objects.

labelled objects

unlabelled objects

order of labelled objects

order of unlabelled objects

Graphs: The set of labelled graphs $\mathcal{L}$ is the set of all simple graphs, $\Gamma = \prod_{i,j \in \mathbb{N}}^{\infty}(S_i \times S_j)$ such that for a graph $G = (V, E, I)$, action of $g \in \Gamma$ on G is defined by the factor $S_{|V|} \times S_{|E|}$ of g which permutes the labels of V, E and I in correspondence to them which means each orbit of $\mathcal{L}^{\Gamma}$ is made of all isomorphic labellings of a $G \in \mathcal{L}$. The order of G is also defined as $o(G) = |V|$.

General: We define the sets of *lower objects* and *upper objects* for each $X \in \mathcal{L}$ denoted by $L(X)$ and $U(X)$ such that for $X_1 \neq X_2 \in \mathcal{L}$, the sets $\{X_1\}$, $L(X_1)$, $U(X_1)$, $\{X_2\}$, $L(X_2)$ and $U(X_2)$ should be pairwise disjoint. Then we define the sets of all lower and upper objects as $\check{\mathcal{L}} = \bigcup_{X \in \mathcal{L}} L(X)$ and $\hat{\mathcal{L}} = \bigcup_{X \in \mathcal{L}} U(X)$. The *order* of lower and upper objects of X are defined to be the same as order of X.

lower objects

upper objects

order of lower and upper objects

Graphs: We create a graph that has one more vertex by adding a new vertex to a smaller graph and joining to its neighbours. Lower objects include the information to reduce the graph back to its parent so for $G \neq \mathbf{K}_1$ we can define $L(G) = \{\langle G, w\rangle : w \in V(G)\}$ that means to remove w from G and as $\mathbf{K}_1$ does not have any parents, we define $L(\mathbf{K}_1) = \varnothing$. Conversely, the upper objects demonstrate, how one graph can be extended to its children. In our case, we can define it as $U(G) = \{\langle G, W\rangle : W \subseteq V(G)\}$ which means to add a vertex and connect it to all vertices in W.

General: We need a group Γ acting on $\mathcal{L} \cup \check{\mathcal{L}} \cup \hat{\mathcal{L}}$ in addition to a relation $R \subseteq \check{\mathcal{L}} \times \hat{\mathcal{L}}$ satisfying Axioms C1-C7 to relate lower and upper objects to each other which is used for defining parents of objects. The functions f and f' used in the list of axioms are defined as

$$f(\check{Y}) = \left\{\hat{X} : \left\langle \check{Y}, \hat{X} \right\rangle \in R\right\} \tag{1.1}$$

$$f'(\hat{X}) = \left\{\check{Y} : \left\langle \check{Y}, \hat{X} \right\rangle \in R\right\} \tag{1.2}$$

C1. Γ fixes each of $\mathcal{L}$, $\check{\mathcal{L}}$ and $\hat{\mathcal{L}}$ setwise.

C2. $\forall X \in \mathcal{L}\ \forall g \in \Gamma : L(X^g) = L(X)^g \wedge U(X^g) = U(X)^g$.

C3. $\forall \check{Y} \in \check{\mathcal{L}}\ \forall \hat{X} \in \hat{\mathcal{L}} : f(\check{Y}) \neq \varnothing \wedge f'(\check{X}) \neq \varnothing$.

C4. $\forall \check{Y} \in \check{\mathcal{L}}\ \forall g \in \Gamma\ \forall \hat{X}_1 \in f(\check{Y})\ \forall \hat{X}_2 \in f(\check{Y}^g)\ \exists h \in \Gamma : \hat{X}_1^h = \hat{X}_2$.

C5. $\forall \hat{X} \in \hat{\mathcal{L}}\ \forall g \in \Gamma\ \forall \check{Y}_1 \in f'(\hat{X})\ \forall \check{Y}_2 \in f'(\hat{X}^g)\ \exists h \in \Gamma : \check{Y}_1^h = \check{Y}_2$.

C6. $\forall X \in \mathcal{L}\ \forall g \in \Gamma : o(X^g) = o(X)$.

C7. $\forall \check{Y} \in \check{\mathcal{L}}\ \forall \hat{X} \in f(\check{Y}) : o(\hat{X}) < o(\check{Y})$.

Graphs: We can easily extend the action of $g \in \Gamma$ on $\langle G, w \rangle \in \check{\mathcal{L}}$ and $\langle G, W \rangle \in \hat{\mathcal{L}}$ as $\langle G^g, w^g \rangle$ and $\langle G^g, W^g \rangle$, respectively which guarantees axiom C1. To define R we use the construction path from lower to upper objects: $f : \langle G, w \rangle \mapsto \{\langle G \backslash w, N_G(w) \rangle^g : g \in \Gamma\}$. With this definition one can easily verify that C2-C7 hold too.

isomorphism

automorphism group

Aut

irreducible objects

reducible

General: Two labelled objects X_1 and X_2 are *isomorphic*, if there is $g \in \Gamma$ such that $X_1^g = X_2$. Also the *automorphism group* of a labelled object X, written as $\mathrm{Aut}(X)$, is the stabiliser of X in Γ i.e., $\mathrm{Aut}(X) = \{g \in \Gamma : X^g = X\}$. Unlabelled objects which do not have any lower objects are called *irreducible objects* and the other labelled objects are *reducible*. By Axiom C2, being reducible or not is invariant under Γ. Therfore, we can extend the definition to unlabelled objects as well and partition $\mathcal{U}$ into $\mathcal{U}_I$ and $\mathcal{U}_R$, to be the set of irreducible unlabelled and labelled objects, respectively.

Graphs: By the choice of Γ to be $\prod_{i,j \in \mathbb{N}}^{\infty}(S_i \times S_j)$, and the way that we defined action of $g \in \Gamma$ on graphs, this definition of isomorphism is the same as abstract isomorphism. By our definition of lower objects, only $L(\mathbf{K}_1)$ is the empty set so it is the only irreducible object.

General: The final step is to define a function $m : \mathcal{L} \to 2^{\check{\mathcal{L}}}$ such that Axioms M1-M3 holds. This function allows us to define a unique construction path (Lemma 1.10) for each object in $\mathcal{U}_R$.

M1. If $L(X) = \varnothing$, then $m(X) = \varnothing$.

M2. If $L(X) \neq \varnothing$, then $m(X)$ is an orbit of the $L(X)^{\mathrm{Aut}(X)}$.

M3. $\forall X \in \mathcal{L}\ \forall g \in \Gamma : m(X^g) = m(X)^g$.

canonical reduction

We refer to $m(X)$ as the *canonical reduction* of X. The Axiom M3 guarantees that two isomorphic labelled objects have equivalent canonical reduction. Therefore, we can define the canonical reduction of an unlabelled object S as

Lemma 1.10 ([83, Lemma 1]). *There is a unique function $p : \mathcal{U}_R \to \mathcal{U}$ such that*

$$\forall S \in \mathcal{U}_R \; \forall X \in S \; \forall \check{X} \in m(X) \; \exists Y \in p(S) : f(\check{X}) \subseteq U(Y)$$

Graphs: A very simple but not efficient way to define $m(G)$ for $G \neq \mathbf{K}_1$ is to concatenate the rows of adjacency matrix of G and find the lexicographically minimal such vector amongst all different labellings of G. Now let G^g be the labelling with the minimal code and w be the vertex with label "1" in G^g, then we can define

$$m(G) = \begin{cases} \varnothing, & G = \mathbf{K}_1 \\ \{\langle G, v\rangle : \exists t \in \mathrm{Aut}(G) : v^t = w\}, & \text{otherwise} \end{cases}$$

General: Lemma 1.10 shows that each unlabelled object can be mapped to a unique parent based on the function m. Also, by Axiom C7, $o(Y) < o(X)$ which means by finitely many iteration each $\forall S \in \mathcal{U}_R$ can be constructed from an unlabelled irreducible object. We refer to $p(S)$ as the *canonical parent* of S and the set of *canonical ancestors* of S is $\{S, p(S), p(p(S)), \cdots, p(\cdots p(S)) \in \mathcal{U}_I\}$. Inversely, we can define *child* and *descendents* of an unlabelled object. The fact that each unlabelled object has a unique canonical parent gives us a directed forest whose roots are irreducible objects. Using a preorder traversal of this tree McKay designed Algorithm 1.1 (the algorithm presented here is slightly different) to generate all. We use this algorithm in Chapters 2 and 3.

canonical parent
canonical ancestors
child
descendents

Algorithm 1.1 Canonical Construction Path [83]

```
function SCAN($G$: Labelled object, $max\_order$: int)
    if $o(G) = max\_order$ and $G \in \mathcal{F}$ then
        output $G$
        return
    end if

    for each orbit $A$ of the action of $Aut(G)$ on $U(G)$ do
        select a $\hat{G} \in A$ and a $\check{G}' \in f'(\hat{G})$
        let $\check{G} \in L(G')$
        if $o(G') \leqslant max\_order$ and $\check{G} \in m(G')$ then
            SCAN($G'$, $max\_order$)                    ▷ $\hat{G}$ is accepted
        end if
    end for
end function
```

Theorem 1.11 ([83, Theorem 1]). *For each $S \in \mathcal{U}$ and $X \in S$ having $o(X) \leqslant n$,* SCAN *function outputs exactly one labelled object belonging to each unlabelled object of order n that is a descendent of S.*

Corollary 1.12. *Calling* Scan *on one labelled object of every unlabelled irreducible object generates one labelled object of every unlabelled object in $\mathcal{F}$ i.e., the set $\mathcal{F}$ is generated exhaustively with no isomorphic copies.*

Proof. By Theorem 1.11, Scan produce each descendent of any irreducible object once. Adding the fact that each object has a unique set of ancestors which start with exactly once irreducible object, the proof is obtained. □

1.5 Hamiltonian and Hypohamiltonian Graphs

Hamiltonian graph
Hamiltonian cycle
traceable graph
Hamiltonian path

A graph G is called *Hamiltonian* if there is a cycle C in G which passes through every vertex exactly once, and the cycle C is called a *Hamiltonian cycle* of G (Figure 1.3(a)). Similarly, G is called *traceable* if there is a path in G that passes through every vertex exactly once, and such a path is called a *Hamiltonian path* (Figure 1.3(b)). Determining if a graph is Hamiltonian or traceable is NP-complete [44].

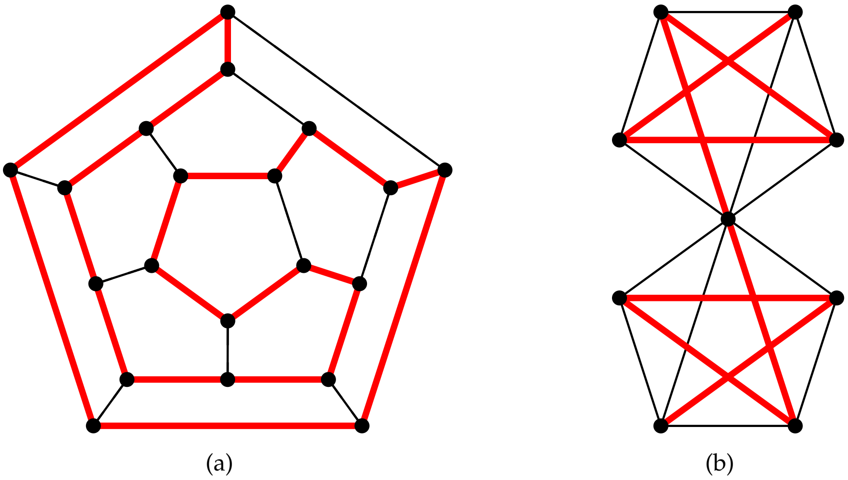
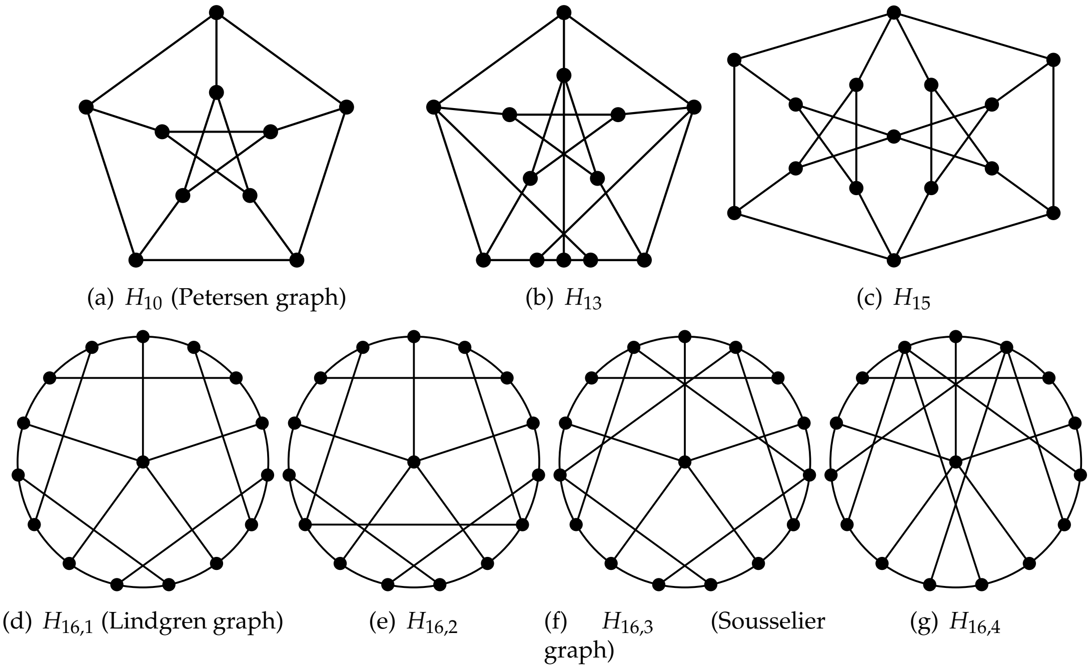


Figure 1.3: Sample Hamiltonian and traceable graphs

hypohamiltonian
hypotraceable

A graph G is called *hypohamiltonian/hypotraceable*, if it is not Hamiltonian/traceable, but the deletion of any single vertex $v \in V(G)$ gives a Hamiltonian/traceable graph. Hypohamiltonian graphs first appeared in the literature with a question by Sousselier [103] and its solution by Gaudin, Herz and Rossi [45] that the Petersen graph is the smallest hypohamiltonian graph.

In 1967, Herz, Duby and Vigué [54] using an exhaustive computer search showed that there is no hypohamiltonian graph on 11 or 12 vertices. Later Collier and Schmeichel [27] proved the same result for graphs on 14 vertices. Aldred, McKay and Wormald [1] classified all hypohamiltonian graphs with fewer than 18 vertices: there is one such graph for each of the orders 10, 13, and 15, four of order 16, and none of order 17 (Figure 1.4). Moreover, hypohamiltonian graphs exist for all orders greater than or equal to 18.



(a) H_{10} (Petersen graph)

(b) H_{13}

(c) H_{15}

(d) $H_{16,1}$ (Lindgren graph)

(e) $H_{16,2}$

(f) $H_{16,3}$ (Sousselier graph)

(g) $H_{16,4}$

Figure 1.4: All hypohamiltonian graphs up to 17 vertices

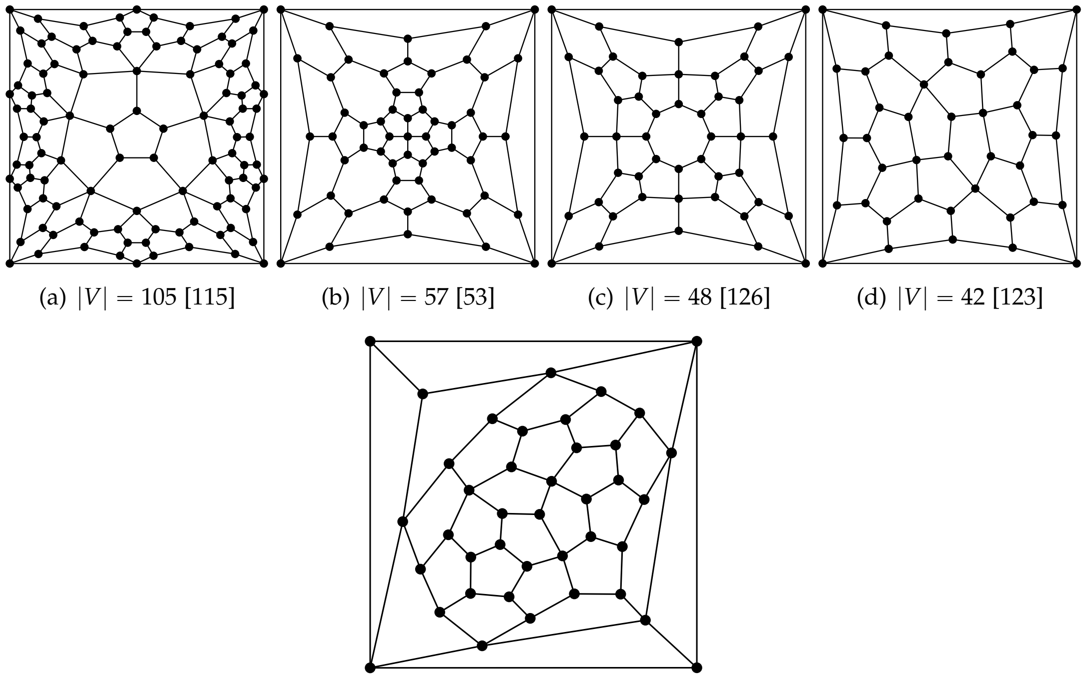
Until 1976, it was unknown if *planar* hypohamiltonian graphs exist. Thomassen [115] in 1976 showed there are infinitely many of them. The smallest amongst them has order 105. In 1979, Hatzel found a planar hypohamiltonian graph on 57 vertices [53]. This result was improved to order 48 (C. Zamfirescu and T. Zamfirescu [126] in 2007), and 42 (Wiener and Araya [123] in 2011). These four graphs are depicted in Figures 1.5(a), 1.5(b), 1.5(c) and 1.5(d), respectively. In Chapter 5 we break this record by finding 25 graphs on 40 vertices one of which is presented in Figure 1.5(e) and the complete set of them can be found in Figure 5.6.

Grinberg in 1968 [48] proved the following necessary condition for a plane graph to be Hamiltonian which is a very useful criterion for showing a graph is not Hamiltonian.

Theorem 1.13 (Grinberg's Theorem [121, Theorem 7.3.5]). *Given a loopless plane graph with a Hamiltonian cycle C and f_i (f_i') i-faces inside (outside) of C, we have*

$$\sum_i (i-2)(f_i - f_i') = 0.$$

Proof. This theorem can be proved by induction on the number of edges. A Hamiltonian graph of order n has at least n edges (the cycle graph). So for the base case of induction we use cycle graphs. A cycle graph, $\mathbf{C}_n$ has two faces of size n which are on different sides of its Hamiltonian cycle. In this case $\sum_i (i-2)(f_i - f_i') = (n-2)(1-1) = 0$. For the induction hypothesis we assume that for every planar



(a) $|V| = 105$ [115]

(b) $|V| = 57$ [53]

(c) $|V| = 48$ [126]

(d) $|V| = 42$ [123]

(e) $|V| = 40$ [64]

Figure 1.5: Records for planar hypohamiltonian graphs

Hamiltonian graph G of order n with $n \leqslant |E(G)| < m$, $\sum_i (i-2)(f_i^G - f_i'^G) = 0$.

Now let $G' = (V, E)$ with $|V| = n$ and $|E| = m > n$ be a planar Hamiltonian graph and C be a Hamiltonian cycle of G'. As $m > n$, C has a chord e. Thus, $G = G' \backslash e$ is Hamiltonian as C is one of its Hamiltonian cycles. Assume F_1 and F_2 are the faces on opposite sides of e (in G') and F_3 be the result of merging these two faces after removal of e (in G). Also considering $f_1 = |F_1|$, $f_2 = |F_2|$ and $f_3 = |F_3|$, we have $f_1 + f_2 = f_3 + 2$.

Without loss of generality we can assume F_3 is on the side of C which is counted towards $f_{f_3}^G$. So as e is a chord of C, F_1 and F_2 fall on the side that is counted in $f_{f_1}^G$ and $f_{f_2}^G$ too. By the induction hypothesis $\sum_i (i-2)(f_i^G - f_i'^G) = 0$. Now for G' we have $\sum_i (i-2)(f_i^{G'} - f_i'^{G'}) = \sum_i (i-2)(f_i^G - f_i'^G) + (f_1 - 2) + (f_2 - 2) - (f_3 - 2) = 0 + (f_1 - 2) + (f_2 - 2) - (f_3 - 2) = f_1 + f_2 - f_3 - 2 = 0$. □

Although Grinberg's condition has a very simple proof it is very useful for showing a planar graph is not Hamiltonian. For example all four previous records for planar hypohamiltonian graphs (Figures 1.5(a), 1.5(b), 1.5(c) and 1.5(d)) have the property that in each of them every face except one (called the exception face) has size equal to 2 in modulo 3. So if they were Hamiltonian $\sum_i (i-2)(f_i - f_i') = 0$, but taking this equation modulo 3, all terms of the sum but the one for the exception face vanish because their multipliers are equal to zero modulo 3. Thus, having t equal to the size of the exception face ($t \neq 2 \mod 3$), we have $(t-2)(1-0) = 0 \mod 3$

which is a contradiction and shows that they are not Hamiltonian.

Several studies have shown that there infinitely many hypohamiltonian graphs: Lindgren 1967 in [77]; Bondy in 1972 [10]; Chvatál in 1973 [26]; Thomassen in 1974 [114],[115] and [117]; Doyen and Van Diest in 1975 [30]; Collier and Schmeichel 1978 [27]; etc. We explain two operations defined by Thomassen [117, 114] in more detail here as we use them later in Chapter 5.

Definition 1.14 ([117]). *Let G be a plane graph with a* 4*-cycle* x, y, z, t*, then* $\mathrm{Th}_{\mathrm{H}}(G; x, y, z, t)$ *is the graph obtained by removing the edges xy and zt then adding a* 4*-cycle* x', y', z', t' *and edges* xx'*,* yy'*,* zz' *and* tt' *(Figure 1.6). The set of all plane graphs obtained from G using* Th_{H} *is denoted by* $\mathrm{Th}_{\mathrm{H}}(G)$*.* Th_{H}

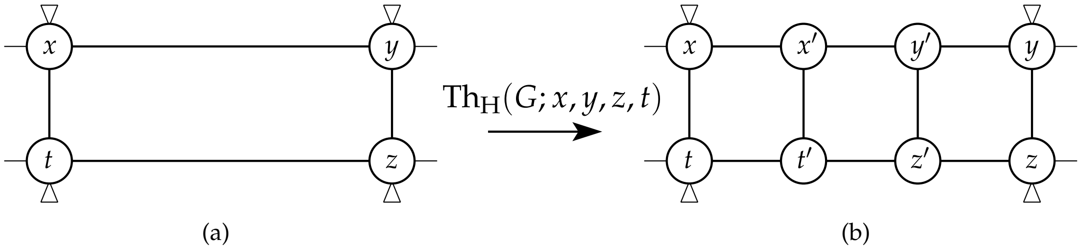


Figure 1.6: Thomassen's operation to expand plane hypohamitonian graphs [117]

Lemma 1.15. *If G is a planar graph with a* 4*-cycle* x, y, z, t*, then* $\mathrm{Th}_{\mathrm{H}}(G; x, y, z, t)$ *is a planar graph too.*

Proof. Let $G' = \mathrm{Th}_{\mathrm{H}}(G; x, y, z, t)$, we have $|V_{G'}| - |E_{G'}| + |F_{G'}| = (|V_G| + 4) - (|E_G| + 6) + (|F_G| + 2) = |V_G| - |E_G| + |F_G|$ which shows that the embedding of G' is planar too. □

Lemma 1.16 ([123, Lemma 4.3]). *If G is a planar non-Hamiltonian graph with a* 4*-cycle* x, y, z, t*, then* $G' = \mathrm{Th}_{\mathrm{H}}(G; x, y, z, t)$ *is a planar non-Hamiltonian graph too.*

Proof. Planarity of G' comes from Lemma 1.15. Now assume to the contrary that $\mathrm{Th}_{\mathrm{H}}(G; x, y, z, t)$ is Hamiltonian and $C_{G'}$ is one of its Hamiltonian cycles. We construct a Hamiltonian cycle C_G for G based on $C_{G'}$ as follows. To cover x', y', z' and t', we have one of these cases for $C_{G'}$:

1. $C_{G'}$ contains x, x', y', y and t, t', z', z: In this case C_G is obtained from $C_{G'}$ by replacing x, x', y', y and t, t', z', z with x, y and t, z, respectively.

2. $C_{G'}$ contains x, x', t', z', y', y: In this case C_G can be constructed from $C_{G'}$ by replacing x, x', t', z', y', y with x, y.

3. $C_{G'}$ contains t, t', x', y', z', z: In this case C_G is formed from $C_{G'}$ by replacing t, t', x', y', z', z with t, z.

Clearly C_G is a Hamiltonian cycle for G which is a contradiction. Thus G' could not have been Hamiltonian. □

Th$_{\mathrm{H}}$

Theorem 1.17 ([123, Lemma 4.4]). *If G is a planar hypohamiltonian graph with a 4-cycle x, y, z, t and all vertices of $\{x, y, z, t\}$ are 3-valent, then $G' = \mathrm{Th}_{\mathrm{H}}(G; x, y, z, t)$ is hypohamiltonian too.*

Proof. We show that for every $v \in V(G')$, $G' \backslash v$ has a Hamiltonian cycle using two cases:

- $v \in \{x', y', z', t'\}$: Without loss of generality assume $v = x'$. As G is hypohamiltonian $G \backslash t$ has a Hamiltonian cycle C_G and $d_{G \backslash t}(x) = 2$ so C_G contains the edge xy. Now replacing xy in C_G with x, t, t', z', y', y we can find a Hamiltonian cycle for $G' \backslash v$.

- $v \in V(G)$: Let C_G be Hamiltonian cycle of $G \backslash v$. After removal of v in G', in the remained graph, we have at least one of the 3-valent vertices of $\{x, y, z, t\}$ without loss of generality we can assume x is such a vertex. Now the fact that $d'_G(x) = 3$ enforces that at least one of edges xy and/or xt is in C_G. Depending on membership of xy or xt in C_G, we can replace that edge with x, x', t', z', y', y or x, x', y', z', t', t to make a Hamiltonian cycle for $G' \backslash v$.

Also by Lemma 1.16, G' is planar and non-Hamiltonian. □

Th$_{\mathrm{T}}$

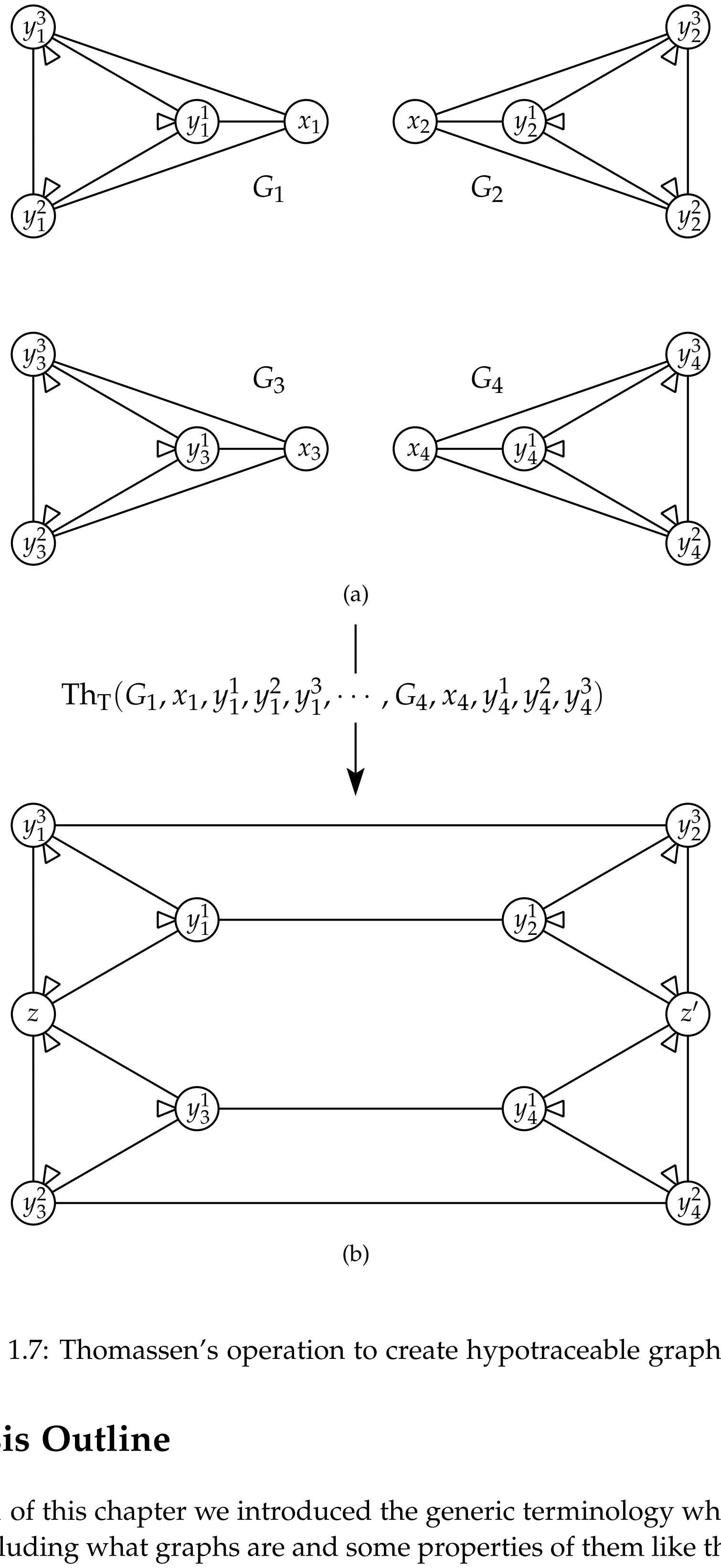
Definition 1.18 ([114]). *Assume for $i = 1, 2, 3, 4$, G_i is a graph which has a 3-valent vertex x_i. Also let y_i^1, y_i^2, y_i^3 be neighbours of x_i and $H_i = G_i \backslash x_i$. Then $\mathrm{Th}_{\mathrm{T}}(G_1, x_1, y_1^1, y_1^2, y_1^3, \cdots, G_4, x_4, y_4^1, y_4^2, y_4^3)$ is the graph obtained from identifying pairs (y_1^2, y_3^3) and (y_2^2, y_4^3) into z_1 and z_2, respectively and adding edges $y_1^1 y_2^1$, $y_1^3 y_2^3$, $y_3^1 y_3^1$ and $y_3^2 y_3^2$. The set of all graphs obtained from G_1, G_2, G_3 and G_4 using Th_{T} is denoted by $\mathrm{Th}_{\mathrm{T}}(G_1, G_2, G_3, G_4)$ (Figure 1.7).*

Theorem 1.19 (Extension of [114, Lemma 3.1]). *If G_1, G_2, G_3 and G_4 are hypohamiltonian each of which has at least a 3-valent vertex, then every $G' \in \mathrm{Th}_{\mathrm{T}}(G_1, G_2, G_3, G_4)$ is hypotraceable. Moreover, if all G_is are plane graph, G' is a plane graph too.*

Proof. We omit the proof of the first part which is exactly the same as [114, Lemma 3.1]. So for the second part assume the G_is are all plane graphs.

$$
\begin{aligned}
&|V_{G'}| - |E_{G'}| + |F_{G'}| \\
&\qquad = \left[\sum_{i=1}^{4} \left(|V_{G_i}| - 1\right) - 2\right] - \left[\sum_{i=1}^{4} \left(|E_{G_i} - 3|\right) + 4\right] + \left[\sum_{i=1}^{4} \left(|F_{G_i}| - 3\right) + 4\right] \\
&\qquad = \sum_{i=1}^{4} \left[|V_{G_i}| - |E_{G_i}| + |F_{G_i}| - 2\right] + 2 \\
&\qquad = 2
\end{aligned}
$$

Which shows G' is a plane graph. □



(a)

(b)

Figure 1.7: Thomassen's operation to create hypotraceable graphs [114]

1.6 Thesis Outline

In Section 1.1 of this chapter we introduced the generic terminology which is used in the thesis including what graphs are and some properties of them like the classic definition of isomorphism for graphs. Then, in Section 1.2 we introduced planar graphs which can be drawn on the plane and we referred to their drawings as plane graphs and later mentioned three different isomorphisms for plane graphs. In Section 1.3 we described recursive generation in mathematical form and then in Section 1.4 discussed how isomorphic copies can be discarded in the generation process and particularly Section 1.4.1 included *canonical construction path* which is the method used mainly in this thesis for the purpose of eliminating isomorphic copies.

In Chapter 2, we introduce our first generator for the class of k-angulations which are a family of plane graphs in which every face is surrounded by exactly k edges. This family has been targeted for small values of k by many researchers namely for $k = 3$ [11, 21, 20, 76, 88, 89, 90, 7, 5, 25, 6, 22], $k = 4$ [8, 91, 18, 87, 20] and $k = 5$ [52]. But for $k > 5$ no generator exists in the literature. We fill this gap by presenting a recursive generator for all $k > 4$. Section 2.2 contains the theoretical aspects of the generator, later in Section 2.3 we discuss how this generator is implemented, adapted to the canonical construction path method (Section 2.3.1) and optimized (Sections 2.3.2 and 2.3.3).

Chapter 2 discusses how we can generate every graph whose faces have k edges, we extend this family in Chapter 3 to the graphs whose faces edge count belong to a custom set $\{k_1, k_2, \cdots, k_t\}$. The software *plantri* is the only algorithm in the literature which is able to generate plane graphs with given maximum and minimum face size [20], but it does not allow gaps in the face sizes. Our approach, on the contrary, does support *every given set* of face sizes. Section 3.2 discusses the theoretical proofs to show the correctness of the generator. Then in Section 3.3 we discuss how this generator can be implemented, followed by adapting to the canonical construction path method (Section 3.3.1) and optimization of the generator (Sections 3.3.2 and 3.3.3).

Finding the smallest planar hypohamiltonian graph has been a challenge in the literature since 1976 [115] after it was conjectured that there is no such graph in 1973 [26]. The smallest one found in 1976 has 105 vertices, this result has been improved since to 57 [53], 48 [126] and to 42 [123] in 2011. We improve upon these records by showing that there are 25 planar hypohamiltonian graphs on 40 vertices. One of the families of planar graphs which have been targeted in the literature for finding smallest planar hypohamiltonian graphs is the family of *Grinbergian graphs* (Section 5.3) which includes all previous found records. We prove that no smaller hypohamiltonian graph exists in this family by showing that the smallest hypohamiltonian graphs in this family have 42 vertices and there are exactly 7 such graphs. We also introduce another family of graphs called *4-face deflatable graphs* (Section 5.4) which includes all Grinbergian graphs and show that the 25 graphs on 40 vertices which we have found have the minimum order in this family as well.

Fullerenes are a family of molecules which is entirely composed of carbon atoms in which each carbon atom is bound to three others. Therefore, fullerenes can be modelled as cubic graphs and in particular their graphs are all 3-connected and planar as well as having the property that all faces are of size 6 except for exactly twelve 5-faces. The generation of these structures has started in 1985 [72] and there is a famous conjecture regarding fullerenes, called the *face-spiral conjecture* which claims that the drawing of fullerenes can be unwound in a spiral manner [81] starting from one face and circulating around that face until all faces are traversed. This conjecture has been used for one of the generators which later was proven that do not generate exhaustively after finding a fullerene on 380 vertices with no face-spirals. Recently, it is proven that 380 is the minimum size of a counterexample [17, 47]. In Chapter 6, we extend the scope of this conjecture from 3-connected cubic $\{5,6\}$-angulation to all 3-connected cubic connected planar graphs whose face size are at most 6. These graphs can be partitioned to 19 different families based on the multiplicity of faces whose size is at most 5. We have found that the conjecture does not hold for these families by finding counterexamples. We showed that the counterexamples for 11 of these families are minimal by an exhaustive search.

Chapter 2

Recursive Generation of k-Angulations

2.1 Introduction

A simple plane graph with the maximal number of faces is a triangulation (otherwise we can add a chord to increase the face count). Thus, in a plane graph $|F| \leqslant 2|E|/3$. If a simple k-regular graph is planar by Euler's formula, $2 = |V| - |E| + |F| \leqslant \frac{2}{k}|E| - |E| + \frac{2}{3}|E| = |E|(\frac{2}{k} - \frac{1}{3})$. Therefore, simple plane k-regular graphs only exist for $k = 3, 4, 5$. Several studies considered generating different families of simple plane 3-regular graphs both theoretically and implementation-wise [34, 104, 4, 24, 21, 23, 20]. Simple plane 4-regular graphs are also considered in different studies [79, 75, 87, 18]. The next family, i.e., 5-regular is targeted in [29, 51, 52].

Simple k-angulations are the plane graphs in which every face size is k. Every simple k-angulation is the dual of a plane k-regular graph (that may not be simple). Although we have the limit of $k < 6$ for the existence of simple plane k-regular graphs, there is no such limit for simple k-angulations. Generation of k-angulations is studied for triangulations ($k = 3$) [11, 21, 20, 76, 88, 89, 90, 7, 5, 25, 6, 22], quadrangulations ($k = 4$) [8, 91, 18, 87, 20] and pentangulations ($k = 5$) [52]. But for $k > 5$ there is no result in the literature. In this chapter, we introduce a generic recursive generator for the following families of k-angulations with $k \geqslant 5$.

- 2-connected simple
- 1-connected simple
- 1-connected with $\delta > 1$

Tables 2.1, 2.2 and 2.3 show the number of k-angulations for $5 \leqslant k \leqslant 10$ up to some orders that we have managed to generate and the list of actual graphs can be downloaded at [61].

Instead of the traditional definition of graphs as pair of vertices and edges, we have defined a graph G as an incidence structure $G = (V, E, I)$ in which V and E are vertices and edges of G and $I \subseteq V \times E$ is its incidence relation. This definition allows us to discuss edges independent of vertices and changing the labels of vertices does not change their incident edges.

2.2 Generation of k-angulations

In order to generate the set of k-angulations, recursive generation can be used. We define the set $\mathcal{F}$ as the set of (k, f)-angulations. Also, we introduce three pairs of operations which we will use to define the expansions and reductions. For the sake of clarity we define the operations on the directed version of the actual graphs obtained from their rotation system. The first operation is e_1 (Figure 2.1(a)) which attaches a new 1-valent vertex to an existing vertex of the graph. As a result the size of the face containing the new vertex is increased by 2. The second operation is e_2 (Figure 2.1(b)) which expands a $(k-2)$-face into a k-face by splitting a vertex into two. The third operation is e_3 (Figure 2.1(c)) which takes a $(k-2)$-face (F in the figure) whose vertices are all adjacent to a single vertex x. Then it converts the $(k-2)$-face into a k-face by adding an extra vertex and rearranging its neighbourhood (H_1 and H_2 could be faces or separating cycles). Mathematically, the rearrangement is made of modification of rot of x, y, z, t and adding vertex u. For example assuming $\operatorname{rot}(z) = (h_2, \cdots, g_2, \cdots, z_1 = \operatorname{inv}(g_1), \cdots, z_t = \overrightarrow{zw})$ in the left hand side of Figure 2.1(c), after expansion we have $\operatorname{rot}(z) = (h_3, \cdots, e_1, z_1, \cdots, z_t)$. These three operations have the following preconditions:

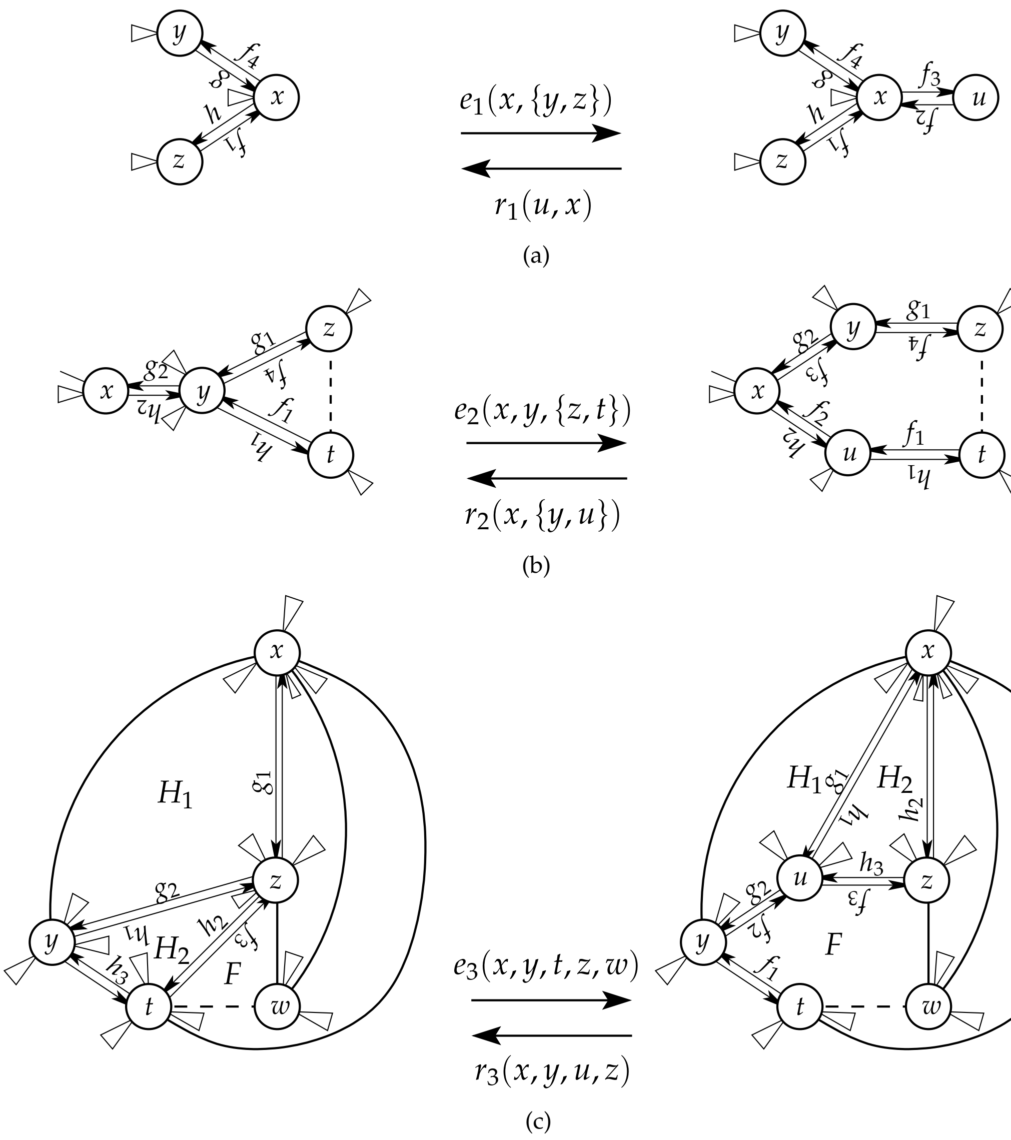
- $C_{e_1}(G; x, y, z)$:
  - y and z are consecutive neighbours of x.
  - y and z could be the same vertices if $d_G(x) = 1$.
- $C_{e_2}(G; x, y, z, t)$:
  - x, z and t are distinct.
  - x is adjacent to y.
  - z and t are consecutive neighbours of y.
- $C_{e_3}(G; x, y, t, z, w)$:
  - w, z and t are consecutive vertices of a face F.
  - Every vertex of F is adjacent to x.
  - y is adjacent to x, z and w.

Now we can define the expansions E_1, E_2 and E_3 as:

$$E_1(G) = \{e_1(G; x, \{y, z\}) : x, y, z \in V(G) \wedge C_{e_1}(G; x, y, z)\} \tag{2.1}$$

$$E_2(G) = \{e_2(G; x, y, \{z, t\}) : x, y, z, t \in V(G) \wedge C_{e_2}(G; x, y, z, t)\} \tag{2.2}$$

$$E_3(G) = \{e_3(G; x, y, w, z, t) : x, y, w, z, t \in V(G) \wedge C_{e_3}(G; x, y, w, z, t)\} \tag{2.3}$$



(a)

(b)

(c)

Figure 2.1: Operations to convert a $(k-2)$-face to a k-face and vice versa

Defining graphs based on incidence structure allows us to say that each of these operations only affects one face and leaves the rest unchanged. For example considering e_3, we need to add a vertex and one edge (two directed edges). If we add the two new directed edges inside the face F (Figure 2.1(c) on the right) as the directed edges $f_1 = \overrightarrow{wy}$ and $f_2 = \overrightarrow{yu}$, then the rest of the faces remain unchanged. So we can guarantee that the only affected face is F. The same idea is applicable to e_1 and e_2 as well.

Each application of E_1, E_2 and E_3 to a graph increases the size of one face by 2 and does not change the face count. Therefore, if the corresponding reductions are applied enough times on a k-angulation all faces eventually can be reduced to either

3-faces or 4-faces depending on the parity of k.

Lemma 2.1. *Assume G is a simple plane graph with a vertex v of degree one in a k-face F. Then $G' = r_1(G; u, x)$ is a simple plane graph with the same faces as G except for F which is converted to a $(k-2)$-face. As the result of the reduction, G' has at most the same number of 1-valent vertices as G.*

Proof. The only face that is affected by the operation is F which will lose two directed edges after the operation. Thus its size will reduced to $k-2$. The simplicity of G' is trivial as removal of a vertex does not add either loops or multiple edges. Also G' is planar because the number of vertices and edges are reduced by one while the number of faces is unchanged so $|V(G')| - |E(G')| + |F(G')| = |V(G)| - |E(G)| + |F(G)|$.

Also $d_{G'}(x) = d_G(x) - 1$ and $\forall w \in V(G' \backslash x)\ d'_G(w) = d_G(w)$. So after removal of u, the number of 1-valent vertices does not increase. □

Lemma 2.2. *Assume G is a non-trivial simple plane graph and F is a t-face ($t > 4$) with vertices $v_1, v_2, \ldots, v_t$ in clockwise order. Then if the following conditions hold, $G' = r_2(G; x, \{w, u\})$ is a simple plane graph with the same faces as G except for F which is converted to a $(t-2)$-face. Also the number of 1-valent vertices of G and G' are the same.*

1. *$d_G(x) > 2$.*
2. *x is the only common neighbour of w and u.*
3. *w and u are not adjacent.*

Proof. Consider the graph $G' = r_2(G; x, \{w, u\})$. We first show that G' does not have any multiple edges. Assume to the contrary that there are two edges between v and p. At least one of v and p should be in y because the operation does not add an edge to any other vertex. Without loss of generality we can assume $p = y$. After applying the operation we have $N_{G'}(y) = N_G(w) \cup N_G(u)$. So to have a multiple edge with an endpoint on y there should be a vertex which is both in $N_G(w)$ and $N_G(u)$. But by the assumption of the lemma $N_G(w) \cup N_G(u) = \{x\}$ and as we merge darts $\overrightarrow{xw}$ and $\overrightarrow{xu}$, there is no multiple edge in G'. Also G' does not have any loops because the only possibility is having a loop on y. But by the assumption, w and u are not adjacent in G so G' is simple.

The operation maintains the planarity, since the only affected face after the operation is F which is converted to a $(t-2)$-face. So the number of faces remain unchanged, but the numbers of vertices and edges are reduced by 1. So $|V(G')| - |E(G')| + |F(G')| = |V(G)| - |E(G)| + |F(G)|$ which means it is still planar. Moreover, the number of 1-valent vertices of G' is the same as G because $d_{G'}(x) = d_G(x) - 1 \geqslant 2$, $d_{G'}(y) > d_G(w)$ and $\forall v \in V(G' \backslash \{x, y\})\ d'_G(v) = d_G(v)$. □

Lemma 2.3. *Assume G is a non-trivial plane graph and F is a t-face ($t > 4$) with vertices $v_1, v_2, \ldots, v_t$ in clockwise order. If there is a vertex x that is adjacent to all the vertices of $V(F)$, then $G' = r_3(G; x, w, y, u, z)$ is a simple plane graph with the same faces as G, except* *for F which is converted to a $(t-2)$-face. Also the number of* 1*-valent vertices of G' and G are the same.*

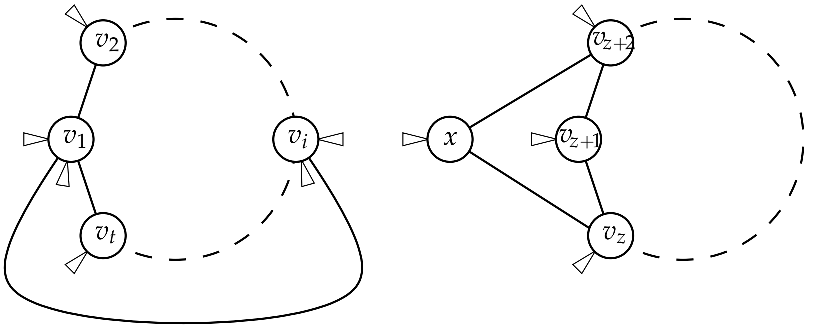
Proof. It is clear that no loops are created. To show that G' does not have any multiple edges, let a be a vertex inside the cycle H_1 (if there is any). After the operation the cyclic ordering of the neighbours of a would be the same except for the fact that any occurrence of x, z and u (if existed) will be replaced by z, w and y; respectively. The new edge between a and w does not make multiple edges because they were not previously adjacent (by the Jordan curve theorem). Also if there are multiple edges in G' between any of the pairs (a, y) or (a, z) there should be the same situation for (a, z) or (a, x) in the original graph which is contradiction. The same proof applies for vertices inside the cycle H_2. Apart from these vertices the fact that no multiple edges could be created for vertices x, y, w and z is trivial and the rest of the graph is the same as G. So the graph remains simple.

The operation e_3 maintains the planarity, since the number of faces remain unchanged, but the numbers of vertices and edges are reduced by 1. Finally, the only modified vertices are x, y, z and w, but all of them have degree at least 3 in G'. So the number of 1-valent vertices remains the same. □

Lemma 2.4. *Assume G is a non-trivial plane graph with no* 1*-valent vertex and F is a t-face $(t > 4)$ with t distinct vertices, such that there is no vertex of G which is adjacent to all vertices of $V(F)$. Then there are vertices x, w and u in F such that $G' = r_2(G; x, \{w, u\})$ is a plane graph with the same faces as G except for F which is converted to a $(t-2)$-face.*

Proof. Let $D = \{v \in V(F) : |N(v) \cap V(F)| > 2\}$. Assume the vertices of F in clockwise order are $v_1, v_2, \ldots, v_t$ (indices are chosen from $\mathbb{Z}_t$).

If D is not empty then there is a vertex of F which is adjacent to at least three vertices of F. Without loss of generality assume v_1 is that vertex. As G is simple, v_1 has a neighbour in $V(F)$ apart from v_2 and v_t, say v_i (Figure 2.2(a)). Now by the Jordan curve theorem, v_2 and v_k are neither adjacent nor have any common neighbour apart from v_1. So by Lemma 2.2, $r_2(G; v_1, \{v_2, v_t\})$ is applicable.



(a) (b)

Figure 2.2: Cases where vertices are distinct and they do not share a neighbour

If D is empty consider the set $N = \{v_z \in V(F) : N(v_z) \cap N(v_{z+2}) \neq \{v_{z+1}\}\}$. If $N = \emptyset$, any two vertices v_z and v_{z+2} of F are not neighbours and do not have any common neighbour apart from v_{z+1}. In this case as G is not trivial there should be a vertex v_z of degree more than two and by Lemma 2.2, $r_2(G; v_z, \{v_{z+1}, v_{z-1}\})$ is applicable.

So the remaining case is when $D = \emptyset$ and $N \neq \emptyset$. Assume $v_z \in N$ and $x \neq v_{z+1}$ is a member of $N(v_z) \cap N(v_{z+2})$ (Figure 2.2(b)). By the assumption of the lemma x cannot be adjacent to all vertices in $V(F)$. So $A = \{v \in V(F) : x \in N(v)\}$ and $B = V(F) \backslash A$ are both non-empty. Thus there is a vertex $v_a \in A$ such that $v_{a+1} \in B$. In this case, by Jordan curve theorem, v_{a+2} and v_a do not have any common neighbour and are not adjacent either. So by Lemma 2.2, $r_2(G; v_{a+1}, \{v_a, v_{a+2}\})$ is applicable. □

Lemma 2.5. *Assume G is a non-trivial plane graph with no 1-valent vertices and F is a t-face ($t > 4$) with $|V(F)| < t$. Then there are $x, w, u \in V(G)$ such that $G' = r_2(G; x, \{w, u\})$ is a simple plane graph with the same faces as G except for F which is converted to a $(t-2)$-face.*

Proof. Assume vertices of F in clockwise order are $v_1, v_2, \ldots, v_t$ (indices are chosen from $\mathbb{Z}_t$). By the assumption of the lemma there are indices i and j such that $v_i = v_j$, $d(v_i) > 2$ and $i \neq j$. Then by the Jordan curve theorem v_{i+1} and v_{i-1} do not have any common neighbour apart from v_i and they cannot be adjacent (see Figure 2.3 noting that possibly $v_{i+1} = v_{j-1}$ or $v_{i-1} = v_{j+1}$). So by Lemma 2.2, $r_2(G; v_i, \{v_{i-1}, v_{i+1}\}, v_{i-1})$ is applicable. □

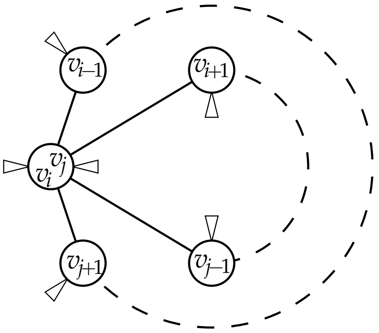


Figure 2.3: Cases where at least two vertices are the same

Corollary 2.6. *Let G be a non-trivial plane graph with no 1-valent vertex and F be a t-face ($t > 4$) of G, then F can be reduced by r_2 or there is a vertex x adjacent to all vertices of F.*

Proof. Assume $v_1, v_2, \ldots, v_t$ are the vertices of F in clockwise order and there is no vertex adjacent to all of its vertices. If there are two distinct indices i and j such that $v_i = v_j$, by Lemma 2.5, we can apply r_2 on F to convert it to a face of size $t-2$; otherwise by Lemma 2.4, r_2 is applicable. □

Theorem 2.7. *Any non-trivial plane graph G having a t-face F with $t > 4$ is reducible by at least one of e_1, e_2 and e_3 to a simple plane graph with the same faces as G except for F which is converted to a $(t-2)$-face.*

Proof. If F has a 1-valent vertex, by Lemma 2.1, e_1 can be applied; otherwise if there is there is no vertex adjacent all vertices of F, by Corollary 2.6 it can be reduced by e_2. In the remaining case by Lemma 2.3, we can use e_3, which completes the proof. □

Theorem 2.8. *Any non-trivial plane graph G with no 1-valent vertex but a t-face F with $t > 4$ is reducible by e_2 or e_3 to a plane graph with the same faces as G except for F which is converted to a $(t-2)$-face.*

Proof. Similar to the proof of Theorem 2.7. □

Now assuming $\mathcal{U}_p^{k,f}$ to be the set of all $\{k_1, \cdots, k_t\}$-angulations having f faces in which all k_i are at in the interval $[3, k]$ and have the same parity as k, we have the following results:

Theorem 2.9. *The triple $(\mathcal{I}^f, \mathcal{E}, \mathcal{U}_p^{k,f})$ recursively generates the set of all (k, f)-angulation in which $(\mathcal{I}^f, \mathcal{E})$ is either $(\mathcal{Q}_f \cup \{\mathbf{C}_k\}, \{E_1, E_2\})$ or $(\mathcal{T}_f \cup \{\mathbf{C}_k\}, \{E_1, E_2, E_3\})$ depending on the parity of k. Moreover, each graph is generated with $f \times \lfloor \frac{k-3}{2} \rfloor$ expansions.*

Proof. Each application of Theorem 2.7 on a t-face $(t > 4)$, converts that face to a $(t-2)$-face while keeping other faces unchanged. So applying r_1, r_2 and/or r_3 to the original graph $\lfloor \frac{k-3}{2} \rfloor$ times, on each face converts all k-faces to triangles or quadrangles. So $(\mathcal{I}^f, \{E_1, E_2, E_3\}, \mathcal{U}_p^{k,f})$ generate all (k, f)-angulations but for even values of k we want to show that $\{E_1, E_2\}$ is enough. To show this property we prove that for even values of k, every graph in $\mathcal{U}^{k,f}$ is bipartite and then we use that to show e_3 is not used in the generation process.

Face Count	k					
	5	6	7	8	9	10
2	3	4	16	28	114	233
3	-	18	-	875	-	50449
4	46	222	10892	70633	2874966	21826951
5	-	3732	-	7884253	-	
6	4305	88252	41983898			
7	-	2361465	-		-	
8	830420	69105036				
9	-		-		-	
10	211549760					

Table 2.1: Number of connected (k, f)-angulations

Let k be an even number, to prove that every graph in $\mathcal{U}_p^{k,f}$ is bipartite we use induction on the summation of face sizes of the graphs. The base case is well-known result that simple quadrangulations are bipartite. For the induction hypothesis, we assume that every graph in $\mathcal{U}_p^{k,f}$ with face size summation up to t is bipartite. Then let $G \in \mathcal{U}_p^{k,f}$ with face size summation equal to $t + 2$ which has a face F_G that is reducible by r_3. Assume $G' = r_3(G; x, y, u, z)$ which reduces F_G to a $F_{G'}$. Now by definition of C_{e_3}, every vertex of $F_{G'}$ is adjacent to x that means G' has a 3-cycle and so is not bipartite which is a contradiction. Therefore, G cannot be reduced by r_3 which means $(\mathcal{I}^f, \{E_1, E_2\}, \mathcal{U}_p^{k,f})$ generates all (k, f)-angulations with even k. □

Corollary 2.10. *All* $2k$*-angulations are bipartite.*

Corollary 2.11. *All* $\{2k_1, 2k_2, \cdots, 2k_t\}$*-angulations are bipartite.*

Theorem 2.12. *The triple* $(\mathcal{I}^f, \mathcal{E}, \mathcal{U}_p^{k,f})$ *recursively generates the set of all* (k, f)*-angulation with no 1-valent vertex in which* $(\mathcal{I}^f, \mathcal{E})$ *is either* $(\mathcal{Q}_f \backslash \{\mathbf{P}_3\} \cup \{\mathbf{C}_k\}, \{E_2\})$ *or* $(\mathcal{T}_f \cup \{\mathbf{C}_k\}, \{E_2, E_3\})$ *based on the parity of k. Moreover, each graph is generated with* $f \times \lfloor \frac{k-3}{2} \rfloor$ *expansions.*

Proof. Similar to the proof of Theorem 2.9. □

Face Count	k					
	5	6	7	8	9	10
2	1	1	1	1	1	1
3	-	1	-	1	-	1
4	3	5	6	8	10	12
5	-	12	-	31	-	68
6	30	89	203	454	864	1630
7	-	600	-	6608	-	41485
8	855	6139	32402	130840	544579	1577516
9	-	66481	-	3118563	-	
10	47698	792680	3256885626			
11	-	9813724	-		-	
12	3324907					
13	-		-		-	
14	269714526					

Table 2.2: Number of connected (k, f)-angulations with $\delta > 1$.

Theorem 2.13. *The set of all 2-connected (k, f)-angulations is generated recursively from the triple $(\mathcal{I}^f, \mathcal{E}, \mathcal{U}_p^{k,f})$ in which $(\mathcal{I}^f, \mathcal{E})$ is either $(\mathcal{Q}_f \backslash \{\mathbf{P}_3\} \cup \{\mathbf{C}_k\}, \{E_2\})$ or $(\mathcal{T}_f \cup \{\mathbf{C}_k\}, \{E_2, E_3\})$ depending on the parity of k. Moreover, each graph is generated with $f \times \lfloor \frac{k-3}{2} \rfloor$ expansions.*

Proof. Immediate result of Theorem 2.12. □

Face Count	k					
	5	6	7	8	9	10
2	1	1	1	1	1	1
3	-	1	-	1	-	1
4	3	5	6	8	10	12
5	-	12	-	34	-	75
6	30	89	221	491	977	1832
7	-	600	-	7327	-	48308
8	855	6139	37033	146631		
9	-	66481	-		-	
10	47698	792680				
11	-	9813724	-		-	
12	3324907					
13	-		-		-	
14	269714526					

Table 2.3: Number of 2-connected (k, f)-angulations

2.3 Implementation

Theorems 2.9, 2.12 and 2.13 in conjunction with the canonical construction path method [83] can be used to generate all non-isomorphic k-angulations. To employ CCP, we need to define some terms which will be introduced in Section 2.3.1.

2.3.1 Adapting the Generator to CCP

Let the symmetric group of degree n be S_n. We take the group $\Gamma = S_1 \times S_2 \times S_3 \times \cdots$, where the action on a graph G is such that the factor S_n permutes the vertices on graphs of order n.

Let $\mathcal{G}$ be the set of all labeled plane graphs in $\mathcal{U}_p^{k,f}$ and $G \in \mathcal{G}$, we define the set of *lower objects* of G, denoted by $L(G)$, as the union of disjoint sets $L_1(G)$, $L_2(G)$ and $L_3(G)$ defined as follows:

$$L_1(G) = \{\langle G, t, x\rangle : r_1(G; t, x) \in \mathcal{G}\} \tag{2.4}$$

$$L_2(G) = \{\langle G, x, \{w, u\}\rangle : r_2(G; x, \{w, u\}) \in \mathcal{G}\} \tag{2.5}$$

$$L_3(G) = \{\langle G, x, w, y, u, t\rangle : r_3(G; x, w, y, u, t) \in \mathcal{G}\} \tag{2.6}$$

Similarly, we define the set of *upper objects* of G, written as $U(G)$, to be the union of disjoint sets $U_1(G)$, $U_2(G)$ and $U_3(G)$ defined as follows:

$$U_1(G) = \{\langle G, x, \{y, z\}\rangle : e_1(G; x, \{y, z\}) \in \mathcal{G}\} \tag{2.7}$$

$$U_2(G) = \{\langle G, x, y, \{z, t\}\rangle : e_2(G; x, y, \{z, t\}) \in \mathcal{G}\} \tag{2.8}$$

$$U_3(G) = \{\langle G, x, y, z, t, w\rangle : e_3(G; x, y, z, t, w) \in \mathcal{G}\} \tag{2.9}$$

Using these sets we define the set of all lower and upper objects denoted by $\check{\mathcal{G}} = \check{\mathcal{G}}_1 \cup \check{\mathcal{G}}_2 \cup \check{\mathcal{G}}_3$ and $\widehat{\mathcal{G}} = \widehat{\mathcal{G}}_1 \cup \widehat{\mathcal{G}}_2 \cup \widehat{\mathcal{G}}_3$, respectively in which $\check{\mathcal{G}}_i = \bigcup_{G \in \mathcal{G}} L_i(G)$ and $\widehat{\mathcal{G}}_i = \bigcup_{G \in \mathcal{G}} U_i(G)$. Also, we define the set of parents of an upper object $\widehat{G}$, denoted by $p(\widehat{G})$, as follows. If $\widehat{G} = \langle G, x, \{y, z\}\rangle \in \widehat{\mathcal{G}}_1$, then $p(\widehat{G})$ is the set of all lower objects $\langle G', x', t'\rangle \in \check{\mathcal{G}}_1$ such that $e_1(G', x', t') = G^g$ for some $g \in \Gamma$. For $\widehat{G} \in \widehat{\mathcal{G}}_2 \cup \widehat{\mathcal{G}}_3$, $p(\widehat{G})$ is defined in a similar fashion.

We also need to extend the action of Γ to the lower and upper objects. For each lower or upper object, the action of $g \in \Gamma$ is defined as the tuple obtained by the action of g on elements of that object. If an element of the tuple is a set, we act g on the elements of the set. For example $\langle G, x, \{y, u\}\rangle^g = \langle G^g, x^g, \{y, u\}^g\rangle = \langle G^g, x^g, \{y^g, u^g\}\rangle$.

The orders of lower and upper objects are defined as the order of their graph (first element). We call a function I whose domain is the set of lower objects an *invariant*, if $I(l) = I(l^g)$ for every $g \in \Gamma$. More specifically, an invariant c which maps lower objects to vectors with elements in a totally ordered set with the criteria that $c(l_1) = c(l_2) \Leftrightarrow \exists g \in \Gamma : l_1^g = l_2$ is called a *canonical code* for the lower objects. Using the lexicographic ordering for comparing canonical codes, we define a function m for labeled plane graphs as $m(G) = \{l \in L(G) : \forall l' \in L(G) : c(l) \leqslant c(l')\}$, i.e. $m(G)$ is the set of all lower objects in $L(G)$ with the minimum canonical code, members of $m(G)$ are called *canonical reductions* of G.

Now we can employ CCP using these definitions and by Corollary 1.12 we can generate all families that were discussed in Section 2.2 without isomorphic copies. There are generic ways to optimize the running time of the generation which are in Lines 7 and 11 of the algorithm. Firstly, we can reduce the time required for the computation of m and then we can remove upper objects which are not going to be accepted (passing the condition of Line 11 of Algorithm 1.1). These two approaches are discussed in Sections 2.3.2 and 2.3.3.

2.3.2 Optimization of Canonical Code Comparison

The first issue for the implementation is how to define the canonical code for the lower objects. For this purpose we use a deterministic BFS code defined in [23] (See Definition 4.9 for the details) which we denote by $\mathrm{Bcode}(G;e)$. This code has the property that $\mathrm{Bcode}(G;e) = \mathrm{Bcode}(G';e')$ if and only if there is an order-preserving isomorphism between G and G' which maps the dart e of G to the dart e' of G'. Note that any function having this property can be used for the rest of discussion.

To employ the BFS code for plane isomorphism we have to consider the mirror image of graphs as well. So we define

$$\mathrm{BC}(G;e,d) = \begin{cases} \mathrm{Bcode}(G;e), & d = 1 \\ \mathrm{Bcode}(\mathrm{Mir}(G);e), & d = -1 \end{cases}$$

In the first attempt we use the BFS code to define a function c_1 for canonical coding. Firstly, for a $l \in \check{\mathcal{G}}$ we define $W(l)$ as follows:

- **If** $l = \langle G, u, x \rangle \in \check{\mathcal{G}}_1$**:** Let $e = \overrightarrow{ux}$, we define:

$$W(l) = \min\{\mathrm{BC}(G;e,1), \mathrm{BC}(G;e,-1)\}.$$

- **If** $l = \langle G, x, \{w, u\} \rangle \in \check{\mathcal{G}}_2$**:** Assume $e_1 = \overrightarrow{xw}$ and $e_2 = \overrightarrow{xu}$. Then taking $d = 1$ if $\sigma(e_1) = e_2$, and $d = -1$ otherwise, we define:

$$W(l) = \min\{\mathrm{BC}(G;e_1,d), \mathrm{BC}(G;e_2,-d)\}.$$

- **If** $l = \langle G, x, w, y, u, z \rangle \in \check{\mathcal{G}}_3$**:** Assume $e_1 = \overrightarrow{yu}$ and $e_2 = \overrightarrow{uz}$. Then taking $d = 1$ if $\phi(e_1) = e_2$ and $d = -1$ otherwise, we define:

$$W(l) = \mathrm{BC}(G;e_1,d).$$

Finally, we define the canonical code c_1 such that for $l \in L_i(G)$, $c_1(l) = [i, W(l)]$. In practice comparing lower objects using c_1 could be very slow as the complexity of computing the code is $O(n)$. To reduce this time we define some easily computable invariants $f_1, f_2, \cdots, f_t$ and use them in combination with w to define codes of the form $[f_1(l), f_2(l), \cdots, f_t(l), W(l)]$. Then based on the lexicographic definition of code comparison in Chapter 1, we can check the invariants first and only compute w, only if all invariants gave the same values.

Assuming $F(l)$ to be the face affected by the application of the reduction preserved in l, we define the following invariants. We did not specify many invariants for E_3 because it is used negligibly often in comparison to the other operations in practice,

$$I_s(l) = \begin{cases} d(x), & l = \langle G, u, x\rangle \in \check{\mathcal{G}}_1 \\ d(x), & l = \langle G, x, \{w, u\}\rangle \in \check{\mathcal{G}}_2 \\ 0, & \text{otherwise} \end{cases}$$

$$I_e(l) = \begin{cases} \min\{d(y), d(z)\}, & l = \langle G, u, x\rangle \in \check{\mathcal{G}}_1 \wedge \{\overrightarrow{xy}, \overrightarrow{xz}\} = \{\sigma(\overrightarrow{xu}), \sigma^{-1}(\overrightarrow{xu})\} \\ \min\{d(w), d(u)\}, & l = \langle G, x, \{w, u\}\rangle \in \check{\mathcal{G}}_2 \\ 0, & \text{otherwise} \end{cases}$$

$$I_a(l) = \begin{cases} \text{\# of darts of } F(l) \text{ whose inverse is in } k\text{-faces}, & l \notin \check{\mathcal{G}}_3 \\ 0, & \text{otherwise} \end{cases}$$

$$I_i(l) = \begin{cases} 1, & l \in \check{\mathcal{G}}_1 \\ 2, & l \in \check{\mathcal{G}}_2 \\ 3, & l \in \check{\mathcal{G}}_3 \end{cases}$$

$$I_f(l) = \text{ size of } F(l)$$

Finally, we define $c(l)$, the canonical code of a lower objects l, as

$$c(l) = [I_f(l), I_i(l), I_a(l), I_s(l), I_e(l), W(l)].$$

Amongst the invariant defined above, I_f plays the most important role for $k > 6$. Let G be a graph in the process of the generation that has exactly one face F of size in the interval $[5, k-1]$. Then by Theorem 2.7, F can be reduced by either e_1, e_2 or e_3. In such case I_f makes sure that F is the selected face, so we can filter reductions made from other faces and the canonical reduction is one of the reductions of F. This rule guarantees that all intermediate graphs have at most one intermediate face i.e. a face which is neither a k-face nor a triangle/quadrangle depending to the parity of k. Thus, we can fix t in each step and only look for the reductions on t-faces. Then, the intermediate graphs can only be $\{(3, f_1), (t, 1), (k, f_2)\}$-angulations and this can reduce the number of intermediate graphs.

For example, Figure 2.4 shows how much $I_f(l)$ filters better than $-I_f(l)$ for $(9, 8)$-angulations. In this case the total number of intermediate graphs for c_b (395204638) is less than half of c_a (807769744). Note that c_a makes sure that we expand all faces from triangles to 5-faces, then expanding all to 7-faces and finally to 9-faces. Also not having $I_f(l)$ at all is trivially worst than $-I_f(l)$ because removing it means that in each step every t-face with $t < k$ can be expanded and this increases the number of intermediate graphs.

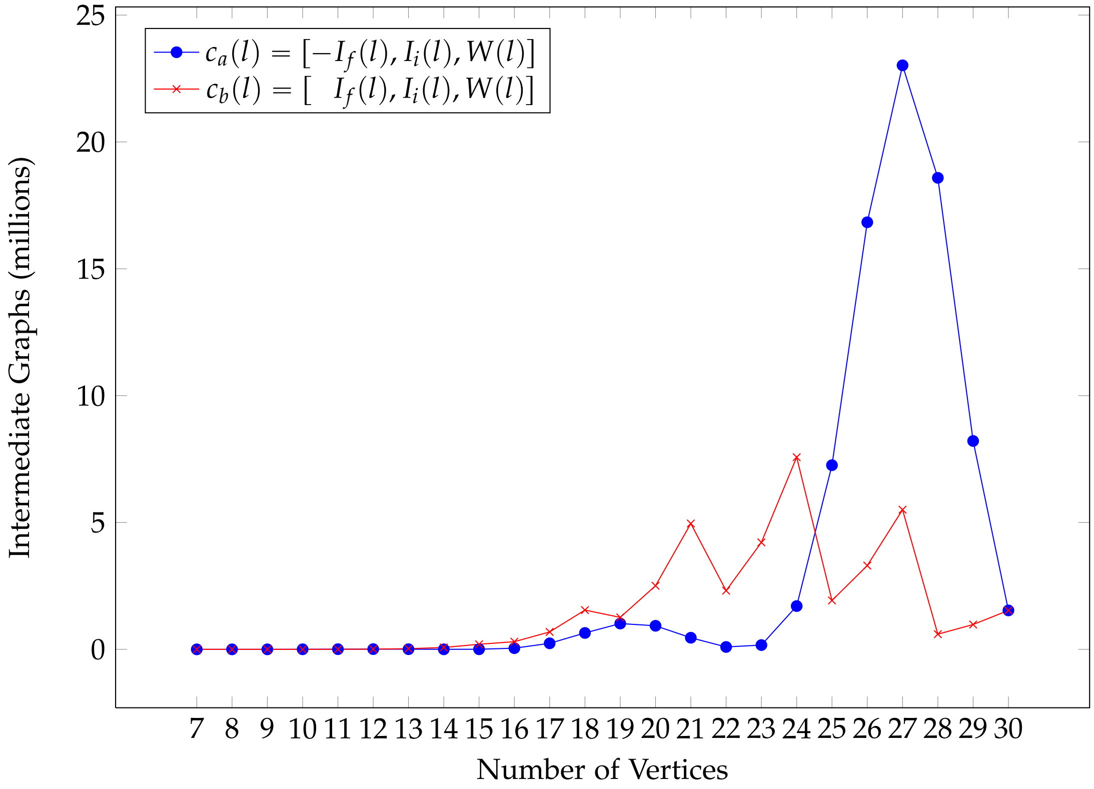


Figure 2.4: Number of intermediate graphs in generation of $(9, 8)$-angulations with two different canonical codes.

2.3.3 Optimizing by Looking Ahead

Considering Algorithm 1.1, let $\widehat{G}$ be an upper object, $\check{G} \in f'(\widehat{G})$ and $\check{G} \in L(G')$ i.e. G' is a descendant of G. Now let $\check{H} \in m(G')$. By definition of our canonical code, $c(\check{H})$ has the minimum code amongst all lower objects of G'. So if $c(\check{H})$ is smaller than $c(\check{G}')$, $\widehat{G}$ will be rejected (See Line 11). In some cases, it is not very difficult to realize that a child is going to be rejected without applying the expansion. Thus another type of optimization which can be used is to avoid making children if they are not made from the inverse of their canonical reduction. This optimization can be done by removing unnecessary upper objects from A.

The first part of the canonical code c is the invariant i_f. Let F_G be the face of G which is expanded by $\widehat{G}$, $F_{G'}$ be the corresponding face in G' and F_H be the face reduced by $\check{H}$. Now if $|F_{G'}| > |F_H|$, then $c(\check{G}') > c(\check{H})$ and $\widehat{G}$ will be rejected. So $\widehat{G}$ can be accepted only if there is no other face F_H such that $|F_G| - 2 > |F_H| > 4$; otherwise F_H is reducible with a lower code. So we can remove the upper objects from $U(G)$ for which such F_H exists.

The second part of c is about prioritizing operations. Note that any usage of e_1 adds a vertex of degree one and other operations do not modify degrees of 1-valent vertices, so as soon as applying e_1 on a face, in every decedent (not necessarily immediate) of G, that face can be reduced by r_1. Also applying e_2 on a face guarantees that no vertex can be adjacent to all vertices of the affected face. These properties also allows to remove some ineffective upper objects.

Finally, we can use third part of c to see if choosing a specific expansion does not have a canonical reduction in its parents. Assume an upper object $\widehat{G} \in U(G)$ expands a face F using e. Also assume in the resulting graph, there is another face which is reducible using inverse of e and has more k-face neighbours than F. This means that $\widehat{G}$ cannot be accepted so we can remove it from $U(G)$. These three look-ahead rules helps us to considerably improve the running time of the generation.

2.4 Conclusions

In this chapter we discussed how k-angulations can be generated recursively from triangulations or quadrangulations. Then we optimised the generator using a careful definition of canonical code for the graphs used in the generation tree in addition to looking ahead and discovering the children which are not going to be accepted and pruning the generation tree.

We defined the recursive generation such that intermediate graphs are not required to belong to the target family (in this chapter k-angulations). This approach allowed us to start from triangulations or quadrangulations, but as we discussed in Section 1.3, this extra flexibility impacts the performance. So to improve this result one could think of another approach which starts with a set of irreducible k-angulations and define the expansions such the intermediate graphs be k-angulations too. Such a generator is quite likely to be more efficient as potentially it could have very few intermediate graphs in comparison.

A natural extension of this study is to generate to not only k-angulations which have only faces of size k, but also $\{k_1, k_2, \cdots, k_t\}$-angulations which include plane graphs with all face sizes in the set $\{k_1, k_2, \cdots, k_t\}$. This extension will be discussed in detail in Chapter 3. We also hope the recursive generation discussed in this chapter will inspire induction proofs for some properties of k-angulations.

Chapter 3

Recursive Generation of Plane Graphs based upon their Face Sequences

3.1 Introduction

face sequence

The *face sequence* of a plane graph is the non-increasing sequence of its face sizes. In Section 2.2 we showed how to generate k-angulations recursively, but with the operations we designed we can go even further to recursively generate all plane graphs whose face sequence contains only odd numbers or only even numbers. So we can have the following theorems with the same proof as Theorems 2.9, 2.12 and 2.13.

Theorem 3.1. *The set of all simple $\{(k_1, f_1), \cdots, (k_n, f_n)\}$-angulations with all k_is having the same parity can recursively be generated from the triple $(\mathcal{I}^f, \mathcal{E}, \mathcal{U}_p^{k,f})$ in which $f = \sum_{i=1}^{n} f_i$ and $(\mathcal{I}^f, \mathcal{E})$ is either $(\mathcal{Q}_f \cup \{\mathbf{C}_k\}, \{E_1, E_2\})$ or $(\mathcal{T}_f \cup \{\mathbf{C}_k\}, \{E_1, E_2, E_3\})$ depending on the parity of k_1.*

Theorem 3.2. *The set of all simple $\{(k_1, f_1), \cdots, (k_n, f_n)\}$-angulations with no 1-valent vertex and all k_is having the same parity can recursively be generated from the triple $(\mathcal{I}^f, \mathcal{E}, \mathcal{U}_p^{k,f})$ in which $f = \sum_{i=1}^{n} f_i$ and $(\mathcal{I}^f, \mathcal{E})$ is either $(\mathcal{Q}_f \backslash \{\mathbf{P}_3\} \cup \{\mathbf{C}_k\}, \{E_2\})$ or $(\mathcal{T}_f \cup \{\mathbf{C}_k\}, \{E_2, E_3\})$ based on the parity of k_1.*

Theorem 3.3. *The set of all 2-connected simple $\{(k_1, f_1), \cdots, (k_n, f_n)\}$-angulations with all k_is having the same parity can recursively be generated from the triple $(\mathcal{I}^f, \mathcal{E}, \mathcal{U}_p^{k,f})$ in which $f = \sum_{i=1}^{n} f_i$ and $(\mathcal{I}^f, \mathcal{E})$ is either $(\mathcal{Q}_f \backslash \{\mathbf{P}_3\} \cup \{\mathbf{C}_k\}, \{E_2\})$ or $(\mathcal{T}_f \cup \{\mathbf{C}_k\}, \{E_2, E_3\})$ based on the parity of k_1.*

In this chapter we show how these results can be extended to generate these classes of simple $\{k_1, \cdots, k_n\}$-angulation without any limit on the parities of k_is.

- 2-connected simple
- 1-connected simple
- 1-connected with $\delta > 1$

The generator that we introduced in Section 2.2 starts from triangulations or quadrangulations as the irreducible objects and expands the faces to the desired sizes and that is why we were able to easily extend them to Theorems 3.1, 3.1 and 3.3 because each operation just increased the size of one face by two without affecting other faces. But in order to generalize these to plane graphs with given face sequence with no parity restriction we need some extra operations which will be discussed in Section 3.2.

3.2 Generation of Plane Graphs based on their Face Sequences

In order to extend Theorems 3.1, 3.1 and 3.3 to all simple $\{k_1, \cdots, k_n\}$-angulations without the parity limit our idea is to have a pre-generator to make $\{3, 4\}$-angulations and then use the same idea as Section 2.2 to extend 3-faces and 4-faces to build odd and even sized faces, respectively. To generate $\{3, 4\}$-angulations we start by triangulations which can be generated very fast using *plantri* [20]. Then, we try to add as many 4-faces as we need.

The first approach to generate $\{(3, f_3), (4, f_4)\}$-angulations is to generate all $(3, f_3 + 2 \cdot f_4)$-angulations and then remove some edges to make 4-faces. Removal of each edge would merge two adjacent 3-faces into a 4-face. This approach could be useful when there are few even sized faces in the desired family. But if we have many even sized faces in comparison, this process becomes very time consuming because each $\{3, 4\}$-angulation can be generated from many different ways and the number of intermediate graphs becomes too large.

To have an estimate of the ratio between the number of irreducible graphs (tringulations) and the final ones ($\{3, 4\}$-angulations with many 4-faces in comparison), we used the ratio between number of triangulations and quadrangulations which can be found in Chart 3.1. For example, for 21 vertices we have 28,615,703,421,545 triangulations and just 57,974,895,671 quadrangulations.

So instead of removing edges to make 4-faces, we preferred to add new faces of size 4 to graphs from the previous steps using the operations presented in Figure 3.2. Let $\mathcal{F}$ be the set of all $\{(k_1, f_1), \cdots, (k_n, f_n)\}$-angulations. We use the operations e_1, e_2 and e_3 with their inverse r_1, r_2 and r_3 defined in Section 2.2 and Figure 2.1 plus two new pairs of operations to achieve this goal. The first operation e_4 takes a path of length two and inflate it to add a 4-face (Figure 3.2(a)) and e_5 rearranges three separating cycles/faces H_1, H_2 and H_3 as in Figure 3.2(b) to add a 4-face. The operations e_4 and e_5 have these preconditions: These operations have the following preconditions:

- $C_{e_4}(G; w, y, z)$:
    - w and z are neighbours of y.
    - $d_G(w) > 1$ and $d_G(z) > 1$.
- $C_{e_5}(G; x, u, y, w)$:

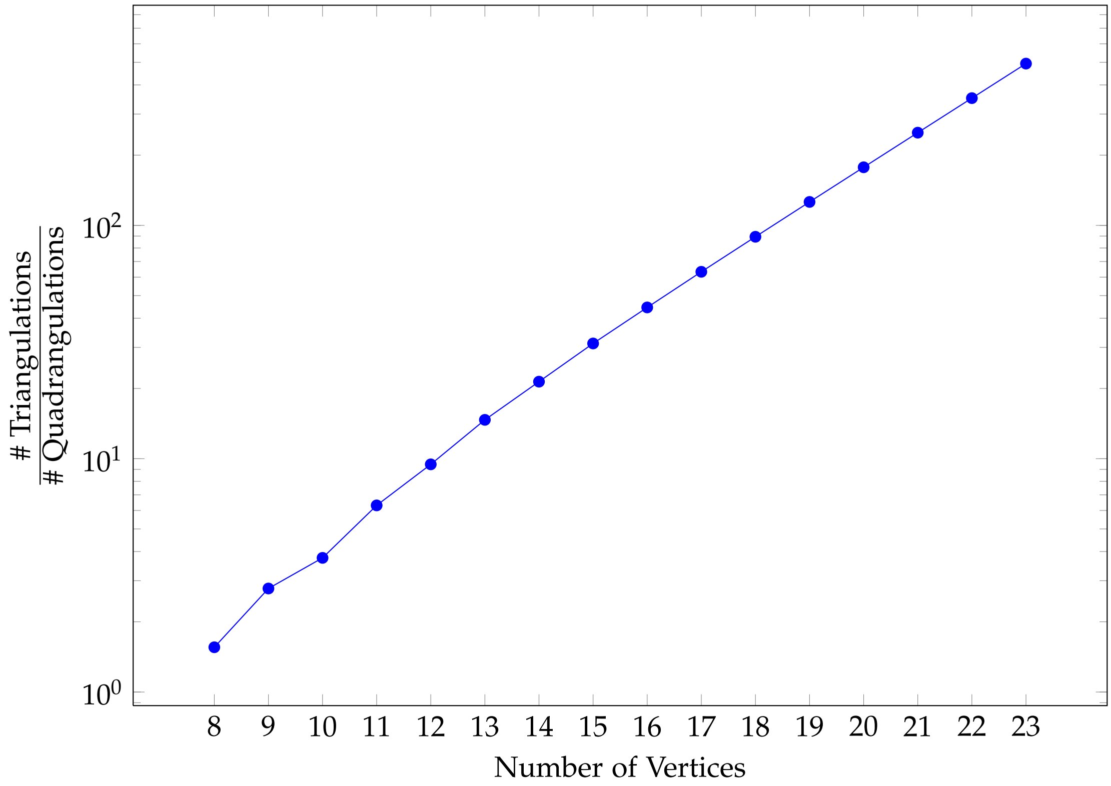


Figure 3.1: Ratio between number of simple triangulations and quadrangulations.

  - x, y and w are pairwise adjacent.
  - u is adjacent to x, y and w.

Now we can define the expansions E_4 and E_5 as:

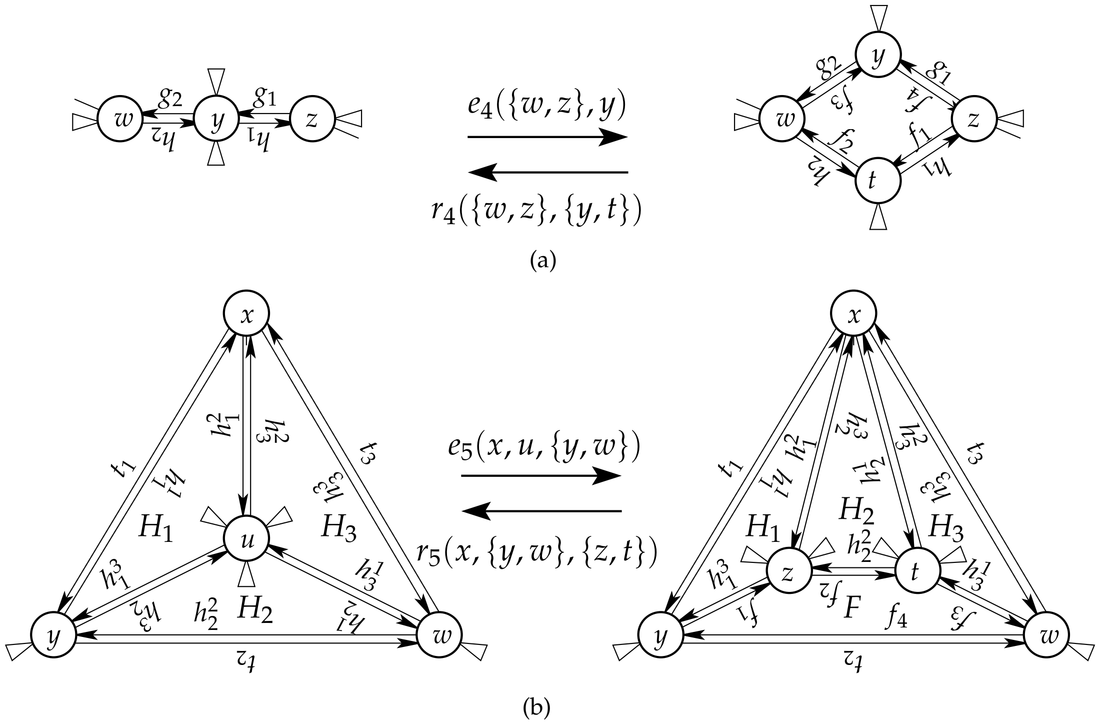
$$E_4(G) = \{e_4(G; \{w,z\}, y) : w, y, z \in V(G) \wedge C_{e_4}(G; w, y, z)\} \tag{3.1}$$

$$E_4(G) = \{e_5(G; x, u, \{y, w\}) : x, y, w, u \in V(G) \wedge C_{e_5}(G; x, u, y, w)\} \tag{3.2}$$

As graphs are defined as incidence structures and the faces as the orbit of darts, each of the operations defined above only create/delete a 4-face and does not affect the rest of faces.

Lemma 3.4. *Assume G is a non-trivial simple $\{(3, f_3), (4, f_4)\}$-angulation with a 4-face $F = (wyzt)$ in which y and z are neither adjacent nor have a common neighbour. Also let $d_G(w), d_G(z) \geqslant 3$, then $G' = r_4(\{w,z\}, \{y,t\})$ is a simple $\{(3, f_3), (4, f_4 - 1)\}$-angulation and all faces of G' and G are the same except F being removed in G'.*

Proof. The only face that is affected by the operation is F which will be removed. The simplicity of G' is trivial because y and t are not adjacent and do not have any common neighbours. Also G' is planar because the number of vertices, edges and faces are reduced by 1, 2 and 1, respectively so $|V(G')| - |E(G')| + |F(G')| = |V(G)| - |E(G)| + |F(G)|$. □



(a)

(b)

Figure 3.2: Operations to add/remove 4-faces to/from $\{3,4\}$-angulations

Lemma 3.5. *Assume G is a simple $\{(3,f_3),(4,f_4)\}$-angulation with a 4-face $F=(yztw)$ whose vertices all adjacent to a vertex x. Then $G'=r_5(x,\{y,w\},\{z,t\})$ is a simple $\{(3,f_3),(4,f_4-1)\}$-angulation and all faces of G' and G are the same except F being removed in G'.*

Proof. The only face that is affected by the operation is F which will be removed. Assume to the contrary that G' is not simple. Trivially, G' does not have a loop so there should be multiple edges between two vertices say v and s in G' (Figure 3.2(b)).

- **If** $x\in\{s,v\}$**,** then in G, x should have been adjacent to a vertex inside H_2 because the new neighbours of x are all in H_2, but this is impossible to the JCT.
- **If** $y\in\{s,v\}$ **(the same for** w**),** then there are multiple edges between s and v in G too, because $N_{G'}(y)\subseteq N_G(y)$.
- **If** $z\in\{s,v\}$ **(the same for** t**),** then by JCT z has multiple edges towoards a vertex $z'\in\{s,v\}$ in H_1 or H_2 which means there are multiple edges between u and z' in G as well.

Moreover, G' is planar because the number of vertices, edges and faces are reduced by 1, 2 and 1, respectively so $|V(G')|-|E(G')|+|F(G')|=|V(G)|-|E(G)|+|F(G)|$. □

Theorem 3.6. *Assume G is a non-trivial simple $\{(3,f_3),(4,f_4)\}$-angulation with a 4-face then there is a face F which can be removed by either r_4 or r_5 to convert it to a simple $\{(3,f_3),(4,f_4-1)\}$-angulation while keeping the rest of faces unchanged.*

Proof. Let $F = (yztw)$ be a 4-face. If there is a vertex x adjacent to all vertices of F then by Lemma 3.5 the result is obtained. So assume there is no such vertex. By JCT at most one of edges yt or zw can be in $E(G)$ and if one of them say yt does exist, by Lemma 3.4, $G' = r_4(\{y,t\},\{z,w\})$ is a desired graph. Similarly, if there is a vertex outside of F adjacent to yt or zw, r_4 is applicable.

In the final case there is no edge and no common neighbours for pairs (y,t) and (z,w). In this case we only need to show at least vertices of one these pairs have degree more than 2 to be able to use Lemma 3.4. Assume to the contrary that this is not true. As G is not trivial at least one of the vertices of F has more than 2 vertices say w so by the assumption $d_G(z) = 2$ and at least one of y and t are 2-valent as well. Without loss of generality we assume $d_G(y) = 2$. This means apart from F, vertices w, y and z belong to another face say F'. But this violate either the fact that G is non-trivial, simplicity of G or $|F'| > 4$ which means G is not a $\{3,4\}$-angulation. □

Theorem 3.7. *The set of all simple $\{(k_1,f_1),\cdots,(k_n,f_n)\}$-angulations is recursively be generated from the triple $(\mathcal{T}_f \cup \mathcal{Q}_f \cup \{\mathbf{C}_k\},\{E_1,E_2,E_3,E_4,E_5\},\mathcal{U}^{k,f})$ in which $f = \sum_{i=1}^n f_i$.*

Proof. We can prove this by induction on $\sum_{i=1}^n k_i \times f_i$. Assume G is a $\{(k_1,f_1),\cdots,(k_n,f_n)\}$-angulation. If G is trivial then $G = \mathbf{C}_k$. Otherwise if G has a face F whose size is more than 4, then it could be reduced using either E_1, E_2 or E_3 by Theorem 2.7 to a simple $\{(k'_1,f'_1),\cdots,(k'_n,f'_n)\}$-angulation with $\sum k'_i \times f'_i = \sum(k_i \times f_i) - 2$. The next case is when G is $\{(3,f_3),\cdots,(4,f_4)\}$-angulation with $f_3, f_4 > 0$ which by Theorem 3.6 can be reduced to a simple $\{(3,f_3),\cdots,(4,f_4-1)\}$-angulation. Finally, for the rest of graphs (basis of induction) G must have only 3-faces or only 4-faces so $G \in \mathcal{T}_f \cup \mathcal{Q}_f$. □

Theorem 3.8. *The set of all simple $\{(k_1,f_1),\cdots,(k_n,f_n)\}$-angulations with no 1-valent is recursively be generated from the triple $(\mathcal{T}_f \cup \mathcal{Q}_f \backslash P_3 \cup \{\mathbf{C}_k\},\{E_2,E_3,E_4,E_5\},\mathcal{U}^{k,f})$ in which $f = \sum_{i=1}^n f_i$.*

Proof. Similar to the proof Theorem 3.7. □

Theorem 3.9. *The set of all simple 2-connected $\{(k_1,f_1),\cdots,(k_n,f_n)\}$-angulations is recursively be generated from the triple $(\mathcal{T}_f \cup \mathcal{Q}_f \backslash P_3 \cup \{\mathbf{C}_k\},\{E_2,E_3,E_4,E_5\},\mathcal{U}^{k,f})$ in which $f = \sum_{i=1}^n f_i$.*

Proof. Immediate result of Theorem 3.8. □

3.3 Implementation

Theorems 3.7, 3.8 and 3.9 in conjunction with the canonical construction path method [83] can be used to generate all non-isomorphic k-angulations. To employ CCP we need to define some terms which will be introduced in Section 3.3.1.

3.3.1 Adapting the Generator to CCP

Let the symmetric group of degree n be S_n. We take the group $\Gamma = S_1 \times S_2 \times S_3 \times \cdots$, where the action on a graph G is such that the factor S_n permutes the vertices on graphs of order n.

Let $\mathcal{G}$ be the set of all labeled plane graphs in $\mathcal{U}^{k,f}$ and $G \in \mathcal{G}$, we define the set of *lower objects* of G, denoted by $L(G)$, as the union of disjoint sets $L_1(G)$, $L_2(G)$ and $L_3(G)$, defined in Equations 2.4, 2.5 and 2.6, with $L_4(G)$ and $L_5(G)$ defined as follows:

$$L_4(G) = \{\langle G, \{w,z\}, \{y,t\}\rangle : r_4(G; \{w,z\}, \{y,t\}) \in \mathcal{G}\} \tag{3.3}$$

$$L_5(G) = \{\langle G, x, \{y,w\}, \{z,t\}\rangle : r_5(G; x, \{y,w\}, \{z,t\}) \in \mathcal{G}\} \tag{3.4}$$

Similarly, we define the set of *upper objects* of G, written as $U(G)$, to be the union of disjoint sets $U_1(G)$, $U_2(G)$ and $U_3(G)$, defined in Equations 2.7, 2.8 and 2.9, with $U_4(G)$ and $U_5(G)$ defined as follows:

$$U_4(G) = \{\langle G, \{w,z\}, y\rangle : e_4(G; \{w,z\}, y) \in \mathcal{G}\} \tag{3.5}$$

$$U_5(G) = \{\langle G, x, u, \{y,w\}\rangle : e_5(G; x, u, \{y,w\}) \in \mathcal{G}\} \tag{3.6}$$

Using these sets we define the set of all lower and upper objects denoted by $\check{\mathcal{G}} = \bigcup_{i=1}^{5} \check{\mathcal{G}}_i$ and $\hat{\mathcal{G}} = \bigcup_{i=1}^{5} \hat{\mathcal{G}}_i$, respectively in which $\check{\mathcal{G}}_i = \bigcup_{G \in \mathcal{G}} L_i(G)$ and $\hat{\mathcal{G}}_i = \bigcup_{G \in \mathcal{G}} U_i(G)$. Also, we extend the set of parents of an upper object $\widehat{G}$, denoted by $p(\widehat{G})$, which was defined in Section 2.3.1 in the to $\widehat{G} \in \hat{\mathcal{G}}_4 \cup \hat{\mathcal{G}}_5$ and the action of Γ to the lower and upper objects in $\hat{\mathcal{G}}_4 \cup \hat{\mathcal{G}}_5$ in the similar fashion to the way used in Section 2.3.1.

The orders of lower and upper objects are defined as the order of their graph (first element). We call a function I whose domain is the set of lower objects an *invariant*, if $I(l) = I(l^g)$ for every $g \in \Gamma$. More specifically, an invariant c which maps lower objects to vectors with elements in a totally ordered set with the criteria that $c(l_1) = c(l_2) \Leftrightarrow \exists g \in \Gamma : l_1^g = l_2$ is called a *canonical code* for the lower objects. Using the lexicographic ordering for comparing canonical codes, we define a function m for labeled plane graphs as $m(G) = \{l \in L(G) : \forall l' \in L(G) : c(l) \leqslant c(l')\}$, i.e. $m(G)$ is the set of all lower objects in $L(G)$ with the minimum canonical code, members of $m(G)$ are called *canonical reductions* of G.

Now we can employ CCP using these definitions and by [83, Theorem 1] we can generate all families that were discussed in Section 2.2 without isomorphic copies. There are generic ways to optimize the running time of the generation which are in Lines 7 and 11 of the algorithm. Firstly, we can reduce the time required for the computation of m and then we can remove upper objects which are not going to be accepted (passing the condition of Line 11 of Algorithm 1.1). These two approaches are discussed in Sections 2.3.2 and 2.3.3.

3.3.2 Optimization of Canonical Code Comparison

The first issue for the implementation is how to define the canonical code for the lower objects.

In the first attempt we define a function c_1 for canonical coding. Firstly, for a $l \in \breve{\mathcal{G}}$ we extend the function $W(l)$ defined in Section 2.3.2 to lower objects in $\breve{\mathcal{L}}_4$ and $\breve{\mathcal{L}}_5$ as follows:

- **If** $l = \langle G, \{w,z\}, \{y,t\} \rangle \in \breve{\mathcal{G}}_4$**:** Assume $e_1^y = \overrightarrow{yz}$, $e_2^y = \overrightarrow{yw}$, $e_1^t = \overrightarrow{tw}$, $e_2^t = \overrightarrow{tz}$. Then taking $d = 1$ if $\sigma(\alpha(e_2^t))) = \alpha(e_1^y)$, and $d = -1$ otherwise, we define:

$$W(l) = \min\{\mathrm{BC}(G; e_1^y, d), \mathrm{BC}(G; e_2^y, -d), \mathrm{BC}(G; e_1^t, d), \mathrm{BC}(G; e_2^t, -d)\}.$$

- **If** $l = \langle G, x, \{y,w\}, \{z,t\} \rangle \in \breve{\mathcal{G}}_5$**:** Assume $e_1 = \overrightarrow{zt}$, $e_2 = \overrightarrow{tz}$ and $e_3 = \overrightarrow{tw}$. Then taking $d = 1$ if $\phi(e_3) = e_2$ and $d = -1$ otherwise, we define:

$$W(l) = \min\{\mathrm{BC}(G; e_1, d), \mathrm{BC}(G; e_2, -d)\}.$$

Then, we define the canonical code c_1 such that for $l \in L_i(G)$, $c_1(l) = [i, W(l)]$. In practice comparing lower objects using c_1 could be very slow as the complexity of computing the code is $O(n)$. To reduce this time we define some easily computable invariants $f_1, f_2, \cdots, f_t$ and use them in combination with w to define codes of the form $[f_1(l), f_2(l), \cdots, f_t(l), W(l)]$. Then based on the lexicographic definition of code comparison in Chapter 1, we can check the invariants first and only compute w, only if all invariants gave the same values.

Assuming $F(l)$ to be the face affected by the application of the reduction preserved in l, we define the following invariants. Noted that as E_3, E_5 are used negligibly often in comparison to the other operations in practice, we did not define many invariants for it.

$$I_s(l) = \begin{cases} d(x), & l = \langle G, u, x \rangle \in \breve{\mathcal{G}}_1 \\ d(x), & l = \langle G, x, \{w,u\} \rangle \in \breve{\mathcal{G}}_2 \\ \min\{d(w), d(z)\}, & l = \langle G, \{w,z\}, \{y,t\} \rangle \in \breve{\mathcal{G}}_4 \\ 0, & \text{otherwise} \end{cases}$$

$$I_a(l) = \begin{cases} \text{number of darts of } F(l) \text{ whose inverse belong to } k\text{-faces}, & l \notin \breve{\mathcal{G}}_3 \cup \breve{\mathcal{G}}_5 \\ 0, & \text{otherwise} \end{cases}$$

$$I_f(l) = \text{ size of } F(l)$$

$$I_e(l) = \begin{cases} \min\{d(y), d(z)\}, & l = \langle G, u, x\rangle \in \breve{\mathcal{G}}_1 \wedge \{\overrightarrow{xy}, \overrightarrow{xz}\} = \{\sigma(\overrightarrow{xu}), \sigma^{-1}(\overrightarrow{xu})\} \\ \min\{d(w), d(u)\}, & l = \langle G, x, \{w, u\}\rangle \in \breve{\mathcal{G}}_2 \\ \min\{d(y), d(t)\}, & l = \langle G, \{w, z\}, \{y, t\}\rangle \in \breve{\mathcal{G}}_4 \\ 0, & \text{otherwise} \end{cases}$$

$$I_i(l) = \begin{cases} 1, & l \in \breve{\mathcal{G}}_1 \\ 2, & l \in \breve{\mathcal{G}}_2 \\ 3, & l \in \breve{\mathcal{G}}_3 \\ 4, & l \in \breve{\mathcal{G}}_4 \\ 5, & l \in \breve{\mathcal{G}}_5 \end{cases}$$

Finally, we define $c(l)$, the canonical code of a lower objects l, as

$$c(l) = [I_f(l), I_i(l), I_a(l), I_s(l), I_e(l), W(l)].$$

3.3.3 Optimizing by Looking Ahead

The canonical code defined in this chapter is an extension of the canonical code defined in Section 2.3.2 and because of that the same lookahead rules as in Section 2.3.3 can be employed to optimize the generation and we do not address them again here as the discussion would be the same.

3.4 Conclusions

In this chapter we discussed how simple plane graphs with specified face sizes can be generated recursively from triangulations or quadrangulations. Then we optimised the generator using a careful definition of canonical code for the graphs used in the generation tree in addition to looking ahead and discovering the children which are not going to be accepted and pruned the generation tree.

We also hope the recursive generation discussed in this chapter will inspire induction proofs for some properties of simple plane graphs with given face sequences.

Chapter 4

Isomorphism Rejection and Canonical Testing of 2-Connected Plane Graphs

4.1 Background

This chapter is a joint study with G. Brinkmann and B. D. McKay.

4.2 Introduction

If a planar graph is 3-connected, by Whitney's theorem it has a unique embedding [122] otherwise it can have several different embeddings on the plane. For example Figure 4.1 presents two non-isomorphic embedding of the same graph. It is very easy to verify this claim as Figure 4.1(a) contains a face with two 1-valent vertices but Figure 4.1(b) does not. Programs like *plantri* [20] which generate many families of plane graphs (embeddings), output the graphs isomorph-free up to plane isomorphism i.e., different embeddings of planar graphs. But if we want to have the graphs isomorph-free up to abstract isomorphism, we may get isomorphic copies in the output in the case that they are not 3-connected.

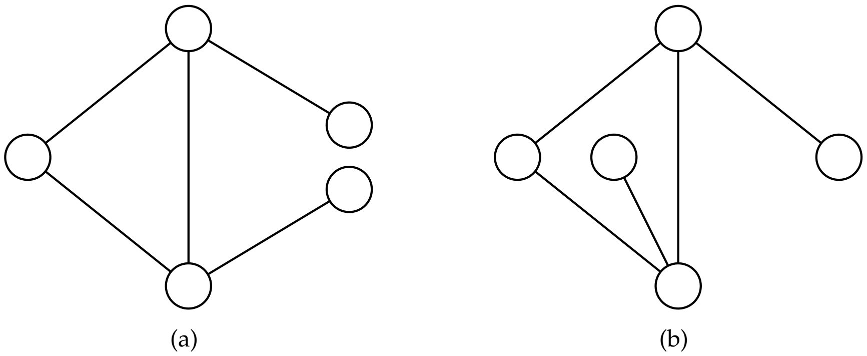


Figure 4.1: Two isomorphic graphs which are not plane-isomorphic

To generate such a family of graphs without abstract isomorphic copies, two approaches can be used: generating all such plane graphs and then remove isomorphic copies or generate all such graphs and then eliminate the ones which are not planar. For example to obtain the list of 2-connected planar graphs we can either run abstract isomorphism test on the output of 2-connected plane graphs or filter non-planar ones from the set of 2-connected graphs. If we can find a fast approach for abstract isomorphism rejection for plane graphs, the first approach could be much more efficient as families of planar graphs are usually exponentially smaller than the corresponding non-planar family. For example Figure 4.2 shows the number of simple 2-connected plane [23, 20, 111], planar [99, 43, 110] and generic graphs [100, 112].

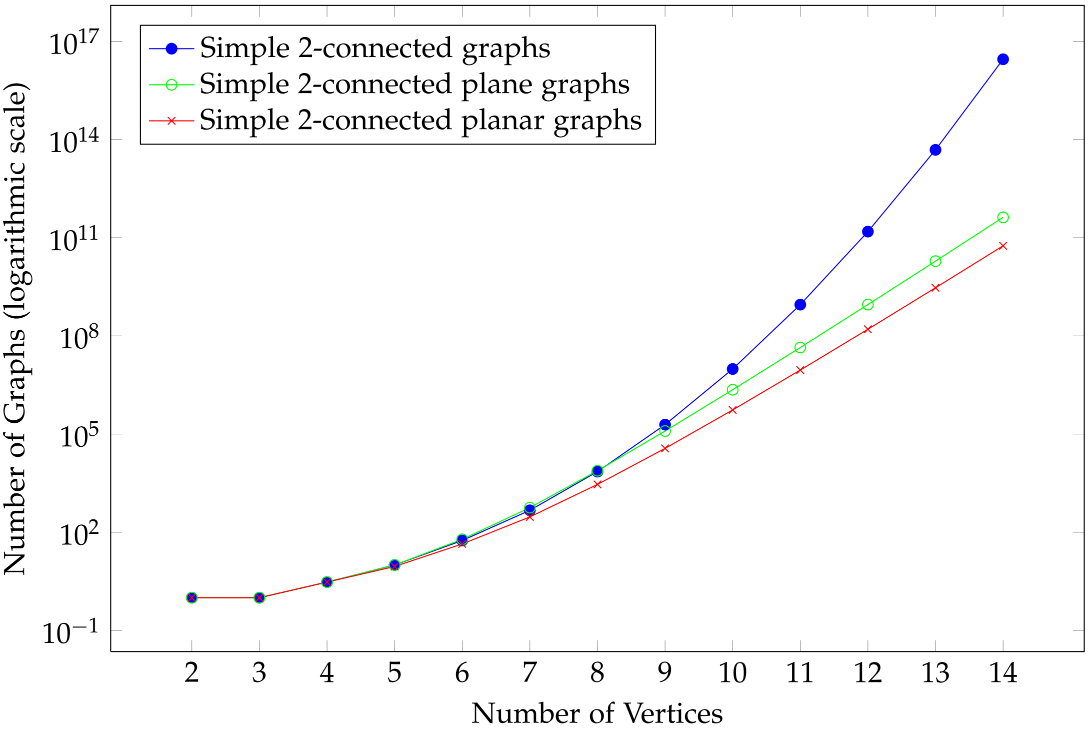


Figure 4.2: Number of simple 2-connected planar, plane and generic graphs.

We can extend the notion of canonical labelling to embeddings of plane graphs and define a *canonical embedding* as an invariant mapping every planar graph G to an specific embedding of it and we may refer to that as the canonical embedding of G.

canonical embedding

In the scope of graph generation, there are four questions regarding abstract isomorphism testing and canonicality of plane graphs:

Q1. Whether two planar (or plane) graphs are isomorphic or not?

Q2. How we can define a canonical code for planar (or plane) graphs?

Q3. Whether a plane graph is embedded canonically or not?

Q4. What is the canonical embedding of a planar graph?

The first question has been targeted by several researchers and there are different sequential [55, 56, 57, 120, 35] and parallel [46, 60] algorithms for this purpose. For 3-connected planar graphs there is an algorithm by Weinberg [120] which is $O(n^2)$ order-preserving isomorphism check which can be easily extended to plane-isomorphism and because 3-connected plane graphs have unique order-preserving embedding on the plane these two types of isomorphism become equivalent. In 1974 Hopcroft and Wong showed that this problem is linear time [57] although they noted that the linear time is theoretical and not for practical purposes. Later Kukluk, Holder and Cook [73] in 2004 designed a practical isomorphism test algorithm in $O(n^2)$. They compared their implementation with other isomorphism tests and showed that for planar graphs with not many edges, their method is quite fast.

Answering Q2 can solve Q1 as well: note that two graphs are isomorphic if and only if their canonical codes are same. In particular, Kukluk, Holder and Cook used this idea for isomorphism testing. They exploit a combination of SPQR-trees [9] and Weinberg's method to design an $O(n^2)$ canonical code computation with the following steps:

1. Make the SPQR-trees for both graphs.
2. Compute a code for both trees which only depends on the tree and 3-connected components.
3. If the codes for both graphs are the same, they are isomorphic; otherwise they are not.

Therefore, after the second stage their algorithm produces a canonical code which the original planar graph can be reconstructed from. For generic graphs there are many canonical labelling algorithms in the literature which are very fast like *nauty* and *traces* [82, 85, 94, 84], *bliss* [67, 66] and *saucy* [28]. But planarity or embedding information are very strong properties which can be utilized to speed up the process.

To our knowledge, for Q3 and Q4 there are no answers in the literature. If a generator outputs every embedding of a given family of graphs to remove abstract-isomorphic copies we can remove every output whose embedding is not canonical. The fact that every embedding appears in the output guarantees that we have at least one copy in the filtered result and by definition of canonical labelling, exactly one of the outputs has it which means exactly one output from each abstract-isomorphism class will be in the filtered result.

In this chapter we design a canonical embedding for 2-connected plane graphs up to *abstract-isomorphism* which can answer Q3. Note that for a 3-connected graph every canonical labelling up to plane-isomorphism is a canonical labelling for abstract-isomorphism as well. To specify the canonical embedding, firstly, we define a bijection Rep mapping every embedding to a string, called the *representation* of that embedding, such that the original embedding can easily be reconstructed from that string. Then we define an embedding to be canonical if it has the lexicographically smallest representation amongst the set of representations of every embedding of that graph. The representation of the canonical embedded graph is called the *canonical representation* of the class abstract-isomorphic embeddings which is an trivially an invariant.

canonical representation

The way that we define the representations allows us to solve Questions Q1, Q2 and Q4 for plane graphs too but not directly planar graphs. We use embedding information to speed up the process, so if the input does not include the embedding, first the input graphs should be embedded in some way (not necessarily in the canonical way) and then our approach can find the canonical embedding (Q4) and canonical code (Q2) which can be used for isomorphism testing (Q1) as well.

A practically important feature of our algorithm is that it gives us instant information while the computation is being done. As we mentioned earlier an embedding is canonical if it has the smallest value of representation. We define representations recursively from some partial representations which can be computed locally based on some subgraphs of the original graph (Definition 4.22). Then we define the term canonical in the way that an embedding is canonical, if all partial representations are canonical too (Theorems 4.34, 4.38 and 4.39). So as soon as finding a non-canonical partial representation, one can realise that the whole embedding is not canonical.

The way that we define partial subgraphs is based on connectivity, as Whitney proved, 3-connected plane graphs have unique embedding on the plane up to plane isomorphism [122]. So with any definition of "canonical embedding", each 3-connected subgraph, is canonically embedded. We define a term 2-block in Section 4.3 which allows us to find 3-connected components of graphs, then we check whether all of them are canonically embedded and if so, we use three operations defined in Section 4.4.1 to replace each of those components with an edge. Then we define the original graph to be embedded canonically if and only if the obtained graph using reduction is canonically embedded too.

For the sake of convenience in the rest of this chapter we define the canonical embedding up to order-preserving isomorphism, instead of plane isomorphism. This can be resolved easily as we can define a plane graph G to be canonical embedding up to plane isomorphism if either G or $\mathrm{Mir}(G)$ is canonically embedded up to order-preserving isomorphism.

In this chapter we put labels on the edges and use them to build representation of plane graphs. For this purpose we define a *plane labelled graph* as a tuple $(V, E, I, D_E, \alpha, \sigma, L)$ in which $(V, E, I, D_E, \alpha, \sigma)$ is a plane graph and $L : D_E \to \mathbb{N}$ is a function which maps each dart to its label. In this section we refer to plane labelled graphs simply as plane graphs unless it is strictly mentioned otherwise.

plane labelled graph

4.3 2-Blocks of 2-Connected Graphs

A 3-connected planar graph has a unique embedding but 2-connected graphs could have different embeddings. But still any 3-connected subgraph of a 2-connected graph has a unique embedding. So this might raise an idea to partition the graph into 3-connected parts and then work on their relation to come up with a canonical embedding. The motivation of this section is this idea and for a 2-connected plane graph a tree called 2-block tree is defined which contains the structure of the 3-connected components.

The same idea for dividing the graph into 3-connected components is studied in [73] in which the graphs are considered as planar graphs not plane graphs and they defined an isomorph check for planar graphs. But it cannot be used for isomorph rejection of plane graphs because the canonical code defined is not based on the embeddings so one can not realise if a specific embedding is canonical or not. In this work a representation will be defined for each embedding which makes it possible to check if an embedding has the canonical representation or not.

Definition 4.1. *Assume T is a 2-cut of a 2-connected graph G and C is one of the components of $G \backslash T$. Now by removing all edges with both endpoints in T from the induced subgraph of $T \cup V(C)$ a graph is obtained which is called an* attached component *of T in G.*

attached component

Definition 4.2. *Assume T is a 2-cut of a 2-connected graph G and A is one of its attached components, the graph $B_{A,T}$ which is made by adding an edge between vertices of T to A is called a* semi-2-block *of G. Furthermore, $B_{A,T}$ is called a* 2-block *of G, if it is 3-connected. The edge which is added to A is called the* virtual edge *of $B_{A,T}$.*

semi-2-block
2-block
virtual edge

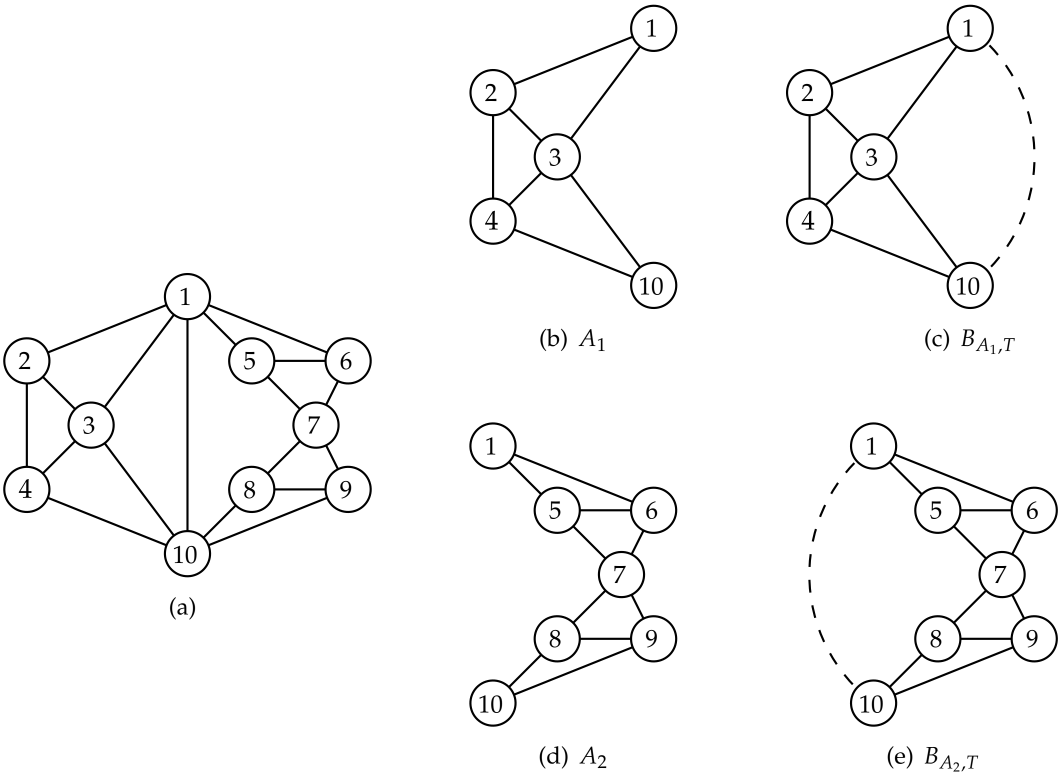
Example 4.3. Consider the graph G presented in Figure 4.3(a) and the cut set $T = \{1, 10\}$. The attached components of T are A_1 and A_2 shown in Figures 4.3(b) and 4.3(d). Furthermore, the semi-2-blocks $B_{A_1,T}$ and $B_{A_2,T}$ are shown in Figures 4.3(c) and 4.3(e), respectively. As $B_{A_1,T}$ is 3-connected it is a 2-block but $B_{A_2,T}$ is not because $\{1, 7\}$ is a 2-cut.

Lemma 4.4. *Assume G is a 2-connected graph and $B = B_{A,T}$ is one of its semi-2-blocks and assume $x, y \in B$. Then for any path P in G from x to y, by replacing any subpath outside of B with an edge whose endpoints are both in T another path P' is obtained which connects x to y in B and $V(P') \subseteq V(P)$.*

Proof. Assume P is a path in G between x and y. If $V(P) \subseteq V(B)$ then the result is trivial. So assume there is a vertex z in $P \backslash B$, then the subpath from x to z should pass through a vertex $t_1 \in T$; because T is a cut set. Also the subpath from z to y should also pass through the other vertex of T say t_2. Now replacing the part $t_1 \to z \to t_2$ in P with the edge $t_1 t_2$ a new path is obtained in B with no additional vertices. It should be noted that after this modification there is no vertex in the resulting path outside of B otherwise P goes through t_1 or t_2 more than once. □

Lemma 4.5. *Every semi-2-block of a 2-connected graph is a 2-connected graph.*

Proof. Assume G is a 2-connected graph and $B = B_{A,T}$ is one of its semi-2-blocks and assume $x, y \in V(B)$. As G is 2-connected there are two vertex-disjoint paths in G connecting x to y say P_1 and P_2. Now by Lemma 4.4 there are paths P'_1 and P'_2 from x to y in B such that $V(P'_1) \subseteq V(P_1)$ and $V(P'_2) \subseteq V(P_2)$. So the vertices of P'_1 and P'_2 remain disjoint. Thus there are two internally vertex-disjoint paths from x to y in B which means B is 2-connected. □



(a)

(b) A_1

(c) $B_{A_1,T}$

(d) A_2

(e) $B_{A_2,T}$

Figure 4.3: Example for attached components, semi-2-blocks and 2-blocks

Lemma 4.6. *Assume $\mathcal{R}$ is the binary relation such that $(G_1, G_2) \in \mathcal{R}$ when G_2 is a semi-2-block of G_1. Then $\mathcal{R}$ is transitive.*

Proof. Assume G is a 2-connected graph, $B_1 = B_{A_C,T_1}$ is one of its semi-2-blocks and $B_2 = B_{A_{C'},T_2}$ a semi-2-block of B_1. In order to prove the lemma it should be shown that T_2 is a 2-cut of G and A_2 is one of its attached components. Let $x \in C'$ and $y \in V(G) \backslash V(B_2)$. Let P be an arbitrary path from y to x. Now we consider two cases:

Case $y \in V(B_1)$**:** By Lemma 4.4 there should be a path P' with $V(P') \subseteq V(P)$ such that $E(P') \subseteq E(B_1)$. By the assumption, T_2 is a 2-cut of B_1 and $y \notin V(B_2)$. So P' should pass through a vertex of T_2 so $V(P) \cap T_2 \neq \emptyset$.

Case $y \notin V(B_1)$**:** As T_1 is a 2-cut of G and $x \in V(B_2) \subset V(B_1)$, P should go through a vertex in T_1 say t_1. Now the subpath of P from t_1 to x goes through a vertex of T_2 similar to the previous case so $V(P) \cap T_2 \neq \emptyset$.

Considering these two cases it can be concluded that any path from a vertex in $V(G) \backslash V(B_2)$ to a vertex of C' passes through T_2 and $|T_2| = 2$. So T_2 is a 2-cut of G which means B_2 is a semi-2-block of G. It should be noted that there is a path between any two vertices in the induced subgraph of G induced by C'; otherwise $A_{C'}$ could not be an attached component of B_2. □

2-block tree

Definition 4.7. *Let G be a 2-connected graph. A tree with root G defined as follows is called a* 2-block tree *of G: If a node N is 3-connected or $\mathbf{K}_3$, it is a leaf; otherwise it is not a leaf and the set of its children is*

$$\mathcal{N} = \{B_{A,T} : A \textit{ is an attached component of } T \textit{ in } N\}$$

for a 2-cut T of N. It should be noted that 2-block tree of a graph is not necessarily unique because at each node different choices for 2-cuts might be available.

Theorem 4.8. *Let G be a 2-connected graph and $\mathcal{T}$ be a 2-block tree of G. Then the leaves of $\mathcal{T}$ which have at least 4 vertices are precisely the 2-blocks of G.*

Proof. Assume $\mathcal{L}$ is the set of leaves of $\mathcal{T}$ which have at least 4 vertices and $\mathcal{B}$ is the set of 2-blocks of G. Then the theorem is equivalent to proving $\mathcal{L} = \mathcal{B}$.

First of all, if the tree has just one node then $G = B$ and the result is trivial. So assume B is a leaf of $\mathcal{T}$ having more than three vertices and also has a parent. By the definition of the tree it should be a semi-2-block of its parent node. Now by induction on the height of the tree and Lemma 4.6 one could check that B is a semi-2-block of G. Moreover, by the definition of leaves, B is 3-connected because it has more than 3 vertices. So B is a 2-block of G and the arbitrary choice of B allows us to conclude $\mathcal{L} \subseteq \mathcal{B}$.

For proving $\mathcal{B} \subseteq \mathcal{L}$ let $B = B_{A,T}$ be an arbitrary 2-block of G and N be a node such that $V(B) \subseteq V(N)$ but for all children N' of N, $V(B) \nsubseteq V(N')$. If $N = B$ then it is 3-connected and so is a leaf. This means $B \in \mathcal{L}$. So assume $V(B) \subsetneq V(N)$. Let x and y be the two vertices of T, $z \in V(N) \backslash V(B)$, $N' = N \backslash (V(B) \backslash T)$. As N is connected, $z \notin V(B)$ and T separates $V(B) \backslash T$ from $V(N) \backslash V(B)$ there should be two paths in N' from z to x and y. So by removing the possible cycles from the walk $y \to z \to x$ a path P in N' from y to x is obtained.

Now assume a and b are two arbitrary distinct vertices in B. As B is 3-connected there are three vertex-disjoint paths from a to b in B. If none of them uses the virtual edge e between x and y (which is not necessarily in N) then the same paths exists between them in N; otherwise e used in exactly one of the paths say P_1. Then by replacing e in P_1 with the path P obtained above three vertex-disjoint paths between a and b are found. Thus there is no way to split a and b with a 2-cut which means all vertices of B should be in one of the children of N. But by the assumption for any children N' of node N, $V(B) \nsubseteq V(N')$. This means that N does not have any children and is a leaf. So N is 3-connected which also means $B = N$ because it does not have any 2-cut that B could be made from one of its attached components so $B \in \mathcal{L}$. So in all situations $B \in \mathcal{L}$ which proves $\mathcal{B} \subseteq \mathcal{L}$ and as a result $\mathcal{L} = \mathcal{B}$. □

4.4 Representation of Plane Graphs

In this section we define the representation of 2-connected plane graphs denoted by Rep in a way that the embedding can be reconstructed from it. This definition is recursive and in each step, using three operations, we replace some subgraphs of a given graph with two labelled edge encoding the corresponding subgraphs, until the graph become either a cycle or 3-connected.

Then we define a string which represents the exact embedding of the labelled 3-connected or cycle graph. The string captures two deterministic traversal of the graph based on both clockwise and counter-clockwise choice of vertices neighbours.

To make this idea more efficient in terms of the length of the representation string, instead of labelling edges with the strings, a table is defined which assigns a number to each string. So each edge is labelled with a number. In this way every code is kept just once even if the part of the graph corresponding to that part occurs many times in the process. But as a down-side, to reconstruct the graph both the resulting graph and the table are required.

The first concept to define for achieving representations is Bcode$(G; e)$ for a dart e of a plane graph G (Definition 4.9). The code is obtained from a deterministic traversal of the graph which is used for encoding some parts of the graph during the computation of representations. The default label for all edges are set to 0 in the beginning. Also the vertices are supposed to have a value called their *colour* which is an invariant. By default we use degree as the colour function unless it is specified differently.

colour

Definition 4.9. *Let $G = (V, E, I, D_E, \alpha, \sigma, L)$ be a connected labelled plane graph with n vertices in which each vertex has a colour. For a dart of G say e, the* bfs code *of e denoted by* Bcode(e) *or* Bcode$(G; e)$ *is defined as follows. First a breath-first search is run on G starting from the head of e and assign an index to each vertex from* 1 *to n, consecutively as they are discovered during the search. To make the bfs search deterministic, for each vertex the edges are traversed in clockwise order starting from the first edge of that vertex which is the inverse of the edge that the vertex is discovered from except the first vertex whose starting edge is e. After this indexing, the* Bcode(e) *is defined as a vector*

bfs code of darts

Bcode(e)

$$\left[c^1, \left(r^1_1, l_1(1)\right), \ldots, \left(r^1_{d(1)}, l_{d(1)}(1)\right), 0, \cdots, c^n, \left(r^n_1, l_1(n)\right), \cdots, \left(r^n_{d(n)}, l_{d(n)}(n)\right), 0\right] \tag{4.1}$$

in which $c^i, l_j(i)$ and r^i_j are the colour, the label of the j-th edge and the index of j-th neighbour of the vertex indexed i.

Definition 4.10. *Considering* min *as the lexicographically minimum, the* bfs code *of a plane graph is defined as*

Bcode(G)

bfs code of a plane graph

$$\text{Bcode}(G) = \min_{e \in D_E(G)} \text{Bcode}(G; e) \tag{4.2}$$

This definition of Bcode is only dependent on the rotation system, labels and colours hence it is an invariant under order-preserving isomorphism.

The operations which we will define reduce $|V| + |E|$ as discussed above. Each operation replaces a subgraph G' of a graph which can be separated from the rest of the graph by a pair of vertices say (u, v) with two darts. Theses new darts are inverse to each other and join u and v. Also the darts $\overrightarrow{uv}$ and $\overrightarrow{vu}$ will be labelled to encode G' and also show which direction $\overrightarrow{uv}$ and/or $\overrightarrow{vu}$ has produced the smallest code for

G'. Note that if the codes computed from u and v are the same the labels of the darts will be the same too.

starred edge

flagged edge

To encode subgraphs as labels we have two special cases that we have to consider; otherwise two isomorphic graphs could be reduced to non-isomorphic ones. Let $c_u < c_w$ be the codes defined for vertcies u and v of G', respectively. Now if in $\mathrm{Mir}(G')$ the codes for u and v are c'_u and c'_v such that $c'_u = c_v$ and $c'_v = c_u$, which means u and v are equivalent under taking mirror from G' then we mark both $\overrightarrow{uv}$ and $\overrightarrow{vu}$ with a an $*$ symbol and refer to those edges as *starred edges*. The second case is when there is another embedding of G' apart from $\mathrm{Mir}(G')$ for which the code obtained from u (or v) is smaller than the c_u (or c_v). In such a case we mark the dart going out of u (or v) with a $\perp$ symbol and call it a *flagged edges*.

unacceptable Bcode

unacceptable $\mathrm{Bcode}_\mathrm{M}$

A Bcode or a $\mathrm{Bcode}_\mathrm{M}$ is *unacceptable* if there is a flagged edge e for which index computed in the code for $\mathrm{head}(e)$ is smaller than the index of $\mathrm{tail}(e)$.

marked plane graph

Definition 4.11. *A* marked plane graph *is a tuple* $(V, E, I, D_E, \alpha, \sigma, L, S, F)$ *such that* $(V, E, I, D_E, \alpha, \sigma, L)$ *is a labelled plane graph and* $S \cup F \subseteq D_E$. *The sets* S *and* F *are the set of starred and flagged darts of the graph with the property that*

$$\forall e \in S : \mathrm{inv}(e) \in S.$$

Also we define the flagged edge closure of G, *written as* $\overline{F}_G$ *or* $\overline{F}$ *as*

$$\overline{F}_G = \{e \in D_E : e \in F \vee \mathrm{inv}(e) \in F\}$$

Assume G is made from its parent G_p from an operation. When we take the mirror from a graph G we can also think of applying the mirror of the operation on the mirror of G_p. But there is a case which need to be considered. Let e be an starred edge in G which is made by the operation and assume it is made from a subgraph G' of G_p. Now when we take mirror from G_p it implies the mirror on G' too. As e is an starred edge, the code obtained from $\mathrm{head}(e)$ was the same as the code for $\mathrm{tail}(e)$ but in $\mathrm{Mir}(G')$. So now that we have the mirror of G_p and G' instead of G_p and G', after applying the operation we have to swap the labels of e and $\mathrm{inv}(e)$ which leads us to the next definition for Mir^* which we should use as the mirror instead of Mir for marked graphs.

$\mathrm{Mir}^*(G)$

Definition 4.12. *Let* G *be a labelled plane graph,. Then the* $\mathrm{Mir}^*(G)$ *has the same rotation system and labels as* $\mathrm{Mir}(G)$ *except for the starred edges whose labels are swapped with their inverse.*

bfs mirror code of edges

$\mathrm{Bcode}_\mathrm{M}(e)$

Definition 4.13. *In the same way as* Bcode *but using counter clockwise order for neighbours the bfs mirror code of e*bfs mirror code *of edges is defined as* $\mathrm{Bcode}(\mathrm{Mir}^*(G); e)$ *which is denoted by* $\mathrm{Bcode}_\mathrm{M}(e)$ *or* $\mathrm{Bcode}_\mathrm{M}(G; e)$. *Also*

$$\mathrm{Bcode}_\mathrm{M}(G) = \min_{e \in D_E(G)} \mathrm{Bcode}_\mathrm{M}(G; e) \tag{4.3}$$

We can extend the definition of isomorphism of labelled multigraphs to marked graphs with the next definition.

Definition 4.14. *Two marked plane graphs* $G_1 = (V_1, E_1, I_1, D_{E_1}, \alpha_1, \sigma_1, L_1, S_1, F_1)$ *and* $G_2 = (V_2, E_2, I_2, D_{E_2}, \alpha_2, \sigma_2, L_2, S_2, F_1)$ *are called* marked-isomorphic*, if there is a bijection* $\pi : V_1 \cup E_1 \cup D_{E_1}(G_1) \to V_2 \cup E_2 \cup D_{E_2}(G_2)$ *such that*

marked isomorphism

1. *The mapping* π *is a plane isomorphism between* G_1 *and* G_2.
2. $\forall e \in D_{E_1}(G) : e \in S_1 \Leftrightarrow \pi(e) \in S_2$.
3. $\forall e \in D_{E_1}(G) : \{\text{label}(e), \text{label}(\text{inv}(e))\} = \{\text{label}(\pi(e)), \text{label}(\text{inv}(\pi(e)))\}$.
4. $\forall e \in D_{E_1}(G) : e \notin (S_1 \cup \overline{F}_1) \wedge \pi(e) \notin (S_2 \cup \overline{F}_2) \Rightarrow \text{label}(e) = \text{label}(\pi(e))$.

In the case that neither of the graphs has any starred or flagged edges, by the last condition of the above definition the bijection should preserve the labels of darts which makes the third condition redundant and also we get the natural extension of plane isomorphism for plane labelled graphs.

The next definition is another code called $\text{Bcode}^*(G;e)$ which is very similar to $\text{Bcode}(G;e)$ with one difference. Whenever it reaches an edge of which neither itself nor its inverse has been visited, it chooses the best between current embedding of the edge or its mirror version (considers the mirror of the subgraph which is replaced by this edge). If the mirror version is better, it replaces that edge and its inverse with their mirror. Using this code one can determine if all edges with label more than 1 represent the best embedded of their corresponding subgraphs which will be discussed later on. It should be noted that for starred edges taking mirror is the same as exchanges label of the edge with its inverse.

Definition 4.15. *The* $\text{Bcode}^*(G;e)$ *is defined in the same way as* $\text{Bcode}(G;e)$ *as:*

$\text{Bcode}^*(G;e)$

$$\left[c^1, \left(r^1_1, l^*_1(1)\right), \cdots, \left(r^1_{d_1}, l^*_{d_1}(1)\right), 0, \cdots, c^n, \left(r^n_1, l^*_1(n)\right), \cdots, \left(r^n_{d_n}, l^*_{d_n}(n)\right), 0\right] \tag{4.4}$$

in which considering $e_j(i)$ *to be the j-th edge of the vertex indexed i as defined in* $\text{Bcode}(G;e)$, $l_j(i) = \text{label}(e_j(i))$, $l'_j(i) = \text{label}(\text{inv}(e_j(i)))$, $\text{index}(v)$ *is the index of vertex v amongst the vertices in Equation 4.4 and* $l^*_j(i)$ *is defined as:*

$$l^*_j(i) = \begin{cases} l_j(i), & e_i(j) \notin S \cup \overline{F} \\ \min\left\{l_j(i), l'_j(i)\right\}, & e_i(j) \in S \cup \overline{F} \wedge \text{index}(\text{tail}(e_j(i))) > i \\ \max\left\{l_j(i), l'_j(i)\right\}, & e_i(j) \in S \cup \overline{F} \wedge \text{index}(\text{tail}(e_j(i))) < i \end{cases}$$

Similarly using counter clockwise ordering for neighbours we can define $\text{Bcode}^*_{\text{M}}(G;e)$*. Also in the same way as* $\text{Bcode}^*(G)$ *and* $\text{Bcode}^*_{\text{M}}(G)$ *we can define* $\text{Bcode}^*(G)$ *and* $\text{Bcode}^*_{\text{M}}(G)$ *as follows.*

$\text{Bcode}^*_{\text{M}}(G;e)$
$\text{Bcode}^*(G)$
$\text{Bcode}^*_{\text{M}}(G)$

$$\text{Bcode}^*(G) = \min_{e \in D_E(G)} \text{Bcode}^*(G;e), \tag{4.5}$$

$$\text{Bcode}^*_{\text{M}}(G) = \min_{e \in D_E(G)} \text{Bcode}^*_{\text{M}}(G;e) \tag{4.6}$$

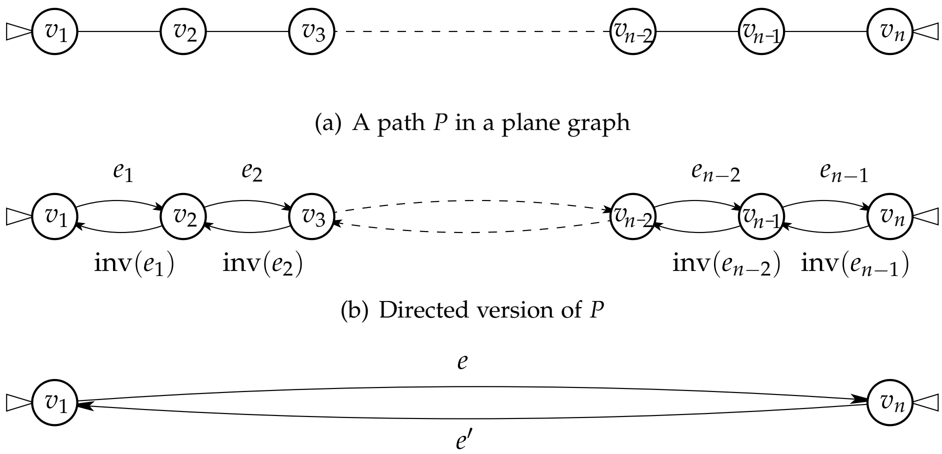
4.4.1 Reduction Operations

The operations that we define in this section replace a subgraph G' of a graph G with an edge (two darts). We choose the subgraphs in a way that they only share two vertices with the rest of the graph. So replacing them with a single edge does not disturb the structure of the graphs. The darts are labelled with vectors that encode the corresponding subgraphs although these vectors will be replaced by numbers in practice for optimization purposes. The codes are based on Bcode but could be modified in some cases.

BfsRep(G)

Definition 4.16. *If G is a plane graph with no vertices of degree 2 and no consecutive multiedges; then* $\text{BfsRep}(G)$ *is defined as* $\text{BfsRep}(G) = \text{Bcode}(G)$.

The first operation called the path reduction (See Definition 4.17 and Figure 4.4) is an operation which replaces a path made of 2-valent vertices with a single edge.



(a) A path P in a plane graph

(b) Directed version of P

(c) Result of $\mathcal{PR}$ on P

Figure 4.4: Path Reduction Operation

path reduction operation
$\mathcal{PR}(P, L)$

Definition 4.17. *Let the path $P = v_1e_1v_2e_2 \dots v_{n-1}e_{n-1}v_n$ be an induced subgraph of a plane labelled graph G and l be a number. Also assume $b_1 =$* $\text{Bcode}(P; e_1)$ *and $b_2 =$* $\text{Bcode}(P; e_1)$. *Now the* path reduction *operation $\mathcal{PR}(P, L)$ is defined as follows:*

1. *Remove all edges and vertices of P except v_1 and v_n from G.*
2. *Add two labelled darts from v_1 to v_n and vice versa, respectively named e and e'. In terms of embedding, e and e' occupy the previous position of e_1 and* $\text{inv}(e_{n-1})$ *in* $\text{rot}(v_1)$ *and* $\text{rot}(v_n)$, *respectively. To label e and e' the following cases should be considered.*
   (a) *If $b_1 = b_2$, label both e and e' with l.*
   (b) *If $b_1 < b_2$, label e and e' with l and $l + 1$, respectively.*

(c) If $b_1 > b_2$, label e and e' with $l+1$ and l, respectively.

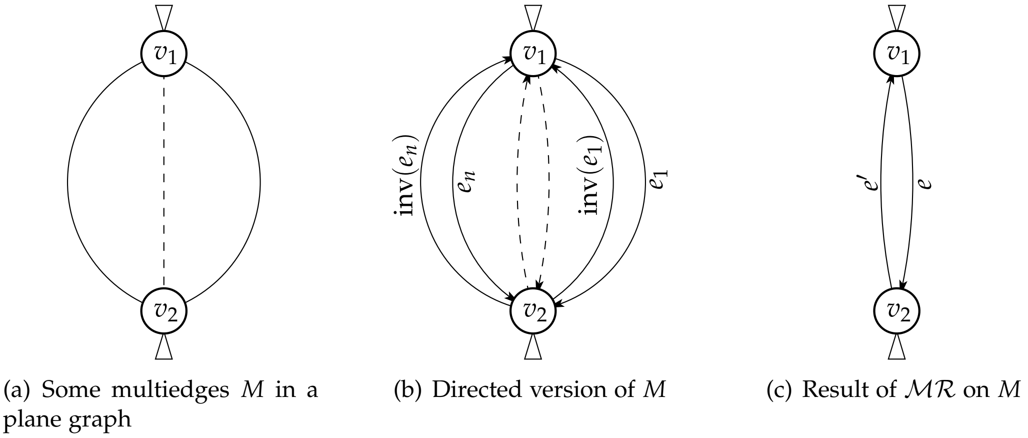
The edges e and e' are starred if $b_1 \neq b_2$ and $\text{Bcode}(P; e_1) = \text{Bcode}_\text{M}(P; \text{inv}(e_{n-1}))$. Also assuming $b_1^* = \text{Bcode}^*(P; e_1)$ and $b_2^* = \text{Bcode}^*(P; e_{n-1})$; darts e and e' are flagged if $b_1^* < b_1$ or b_1^* is unacceptable; and $b_2^* < b_2$ or b_2^* is unacceptable, respectively. Moreover, the $\text{BfsRep}(P)$ *is defined as* $\text{BfsRep}(P) = \min\{b_1, b_2\}$. BfsRep$(P)$

The second operation called the multiedges reduction (See Definition 4.18 and Figure 4.5) replaces a sequence of consecutive edges between two vertices with a single edge. It should be noted that the operation is applicable on a set of multiedges between two vertices v_1 and v_2 only if they are consecutive in $\text{rot}(v_1)$ and $\text{rot}(v_2)$; otherwise it should be applied on each consecutive sequence of edges separately.

But before we define the multiedges reduction operation, we need three new codes for a marked plane graph M with only two vertices which are defined as follows. Let v be one of the vertices of M and $e \in \text{rot}(v)$. Now $\text{Mcode}(M, e)$ and $\text{Mcode}_\text{M}(M, e)$ are the sequence of label of darts in $\text{rot}(v)$ in G and $\text{Mir}^*(G)$ starting from e. Also the code $\text{Mcode}^*(M, v)$ is the increasing sequence of $l^*(e)$ for all $e \in \text{rot}(v)$ in which: Mcode(M, e) Mcode$_\text{M}(M, e)$ Mcode$^*(M, v)$

$$l^*(e) = \begin{cases} \text{label}(e), & e \notin S \cup \overline{F} \\ \min\{\text{label}(e), \text{label}(\text{inv}(e))\}, & \text{otherwise} \end{cases}$$

A $\text{Mcode}(M, e)$ or a $\text{Mcode}_\text{M}(M, e)$ is *unacceptable* if there is a flagged edge $e' \in \text{rot}(\text{head}(e))$ for which index computed in the code for $\text{head}(e')$ is smaller than the index of $\text{tail}(e')$. unacceptable Mcode unacceptable Mcode$_\text{M}$



(a) Some multiedges M in a plane graph

(b) Directed version of M

(c) Result of $\mathcal{MR}$ on M

Figure 4.5: Multiedges Reduction Operation

Definition 4.18. *Let v_1 and v_2 be two vertices of a 2-connected plane labelled graph G with $|V(G)| > 2$, M be the induced subgraph of G induced by some contiguous multiedges between v_1 and v_2 in the rotation system and l is a number. Then assume edges of M are $e_1, \ldots, e_n$*

($n \geqslant 2$) in the same order as the rotation system. So we have $\mathrm{rot}(v_1) = (e_1, \ldots, e_n, x_1, \ldots, x_k)$ *and* $\mathrm{rot}(v_2) = (\mathrm{inv}(e_n), \ldots, \mathrm{inv}(e_1), y_1, \ldots, y_t)$ *for some edges* x_i $(0 \leqslant i \leqslant k)$ *and* y_i $(0 \leqslant i \leqslant t)$*. Firstly, we define* b_1 *and* b_2 *as follows.*

$$b_1 = \begin{cases} \mathrm{Mcode}(M; e_1), & k \neq 0 \\ \min_{e \in \mathrm{rot}(v_1)} \mathrm{Mcode}(M; e), & \textit{otherwise} \end{cases}$$

$$b_2 = \begin{cases} \mathrm{Mcode}(M; \mathrm{inv}(e_n)), & t \neq 0 \\ \min_{e \in \mathrm{rot}(v_2)} \mathrm{Mcode}(M; e), & \textit{otherwise} \end{cases}$$

multiedges reduction operation

$\mathcal{MR}(M, l)$

Now the multiedges reduction *operation* $\mathcal{MR}(M, l)$ *is defined as follows:*

1. *Remove all edges of M from G.*

2. *Add two labelled darts from* v_1 *to* v_2 *and vice versa, respectively named e and* e'*. In terms of embedding after adding these two darts* $\mathrm{rot}(v_1) = (e, x_1, \ldots, x_k)$ *and* $\mathrm{rot}(v_2) = (e', y_1, \ldots, y_t)$*. To label e and* e' *the following cases should be considered.*

   (a) *If* $b_1 = b_2$*, label both e and* e' *with l.*

   (b) *If* $b_1 < b_2$*, label e and* e' *with l and* $l + 1$*, respectively.*

   (c) *If* $b_1 > b_2$*, label e and* e' *with* $l + 1$ *and l, respectively.*

The edges e and e' *are starred if* $k \neq 0$*,* $b_1 \neq b_2$ *and* $\mathrm{Mcode}(M; e_1) = \mathrm{Mcode}_{\mathrm{M}}(M; \mathrm{inv}(e_n))$*. Also assuming* $b_1^* = \mathrm{Mcode}^*(M; v_1)$ *and* $b_2^* = \mathrm{Mcode}^*(M; v_2)$*; darts e and* e' *are flagged if* $b_1^* < b_1$ *or* b_1^* *is unacceptable; and* $b_2^* < b_2$ *or* b_2^* *is unacceptable, respectively. Moreover the* $\mathrm{BfsRep}(M)$ *is also defined as* $\mathrm{BfsRep}(M) = \min\{b_1, b_2\}$*.*

$\mathrm{BfsRep}(M)$

The third operation called the block reduction (See Definition 4.19 and Figure 4.6) replaces a 2-block with a single edge. To prove that this operation is well-defined based on the rotation system, it should be shown that the edges of the 2-block adjacent to v_1 and v_2 are consecutive (in Figure 4.6(b)); otherwise there are different position in $\mathrm{rot}(v_1)$ and $\mathrm{rot}(v_2)$ that the new edges e and e' (in Figure 4.6(c)) can occupy. This property will be result of Lemma 4.20.

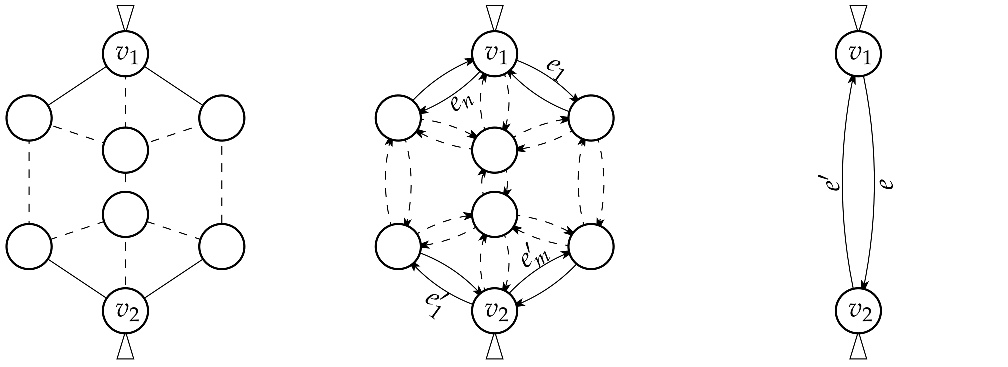
Definition 4.19. *Let* $B = B_{A,T}$ *be a 2-block of a 2-connected plane labelled graph G and* e_v *is one of its virtual edges and* v_1, v_2 *its endpoints and let l be a number. Also assume we have* $\mathrm{rot}(v_1) = (e_1, \ldots, e_n, x_1, \ldots, x_k)$ *and* $\mathrm{rot}(v_2) = (e'_1, \ldots, e'_m, y_1, \ldots, y_t)$ *such that* $\mathrm{tail}(e_i) \in B$ *and* $\mathrm{tail}(e'_j) \in B$ *for all* $1 \leqslant i \leqslant n$ *and* $1 \leqslant j \leqslant m$*. Moreover assume* $b_1 = \mathrm{Bcode}(B; e_1)$ *and* $b_2 = \mathrm{Bcode}(B; e'_1)$*. Now the* 2-block reduction *operation* $\mathcal{BR}(B, l)$ *is defined as follows:*

2-block reduction operation

$\mathcal{BR}(B, l)$

1. *Remove all edges and vertices of B except* v_1 *and* v_2*.*

2. *Add two labelled darts from* v_1 *to* v_2 *and vice versa, respectively named e and* e'*. In terms of embedding after adding these two darts* $\mathrm{rot}(v_1) = (e, x_1, \ldots, x_k)$ *and* $\mathrm{rot}(v_2) = (e', y_1, \ldots, y_t)$*. To label e and* e' *the following cases should be considered.*



(a) Attached component of a 2-block $B_{A,T}$ in a plane graph

(b) Directed version of A

(c) Result of $\mathcal{BR}$ on $B_{A,T}$

Figure 4.6: Block Reduction Operation

(a) If $b_1 = b_2$, label both e and e′ with l.

(b) If $b_1 < b_2$, label e and e′ with l and $l + 1$, respectively.

(c) If $b_1 > b_2$, label e and e′ with $l + 1$ and l, respectively.

The edges e and e′ are starred if $b_1 \neq b_2$ and $\text{Bcode}(B; e_1) = \text{Bcode}_\text{M}(B; e'_m)$. *Also assuming* $b_1^* = \min\{\text{Bcode}^*(B; e_1), \text{Bcode}^*_\text{M}(B; e_n)\}$ *and* $b_2^* = \min\{\text{Bcode}^*(B; \text{inv}(e_n)), \text{Bcode}^*_\text{M}(B; \text{inv}(e_1))\}$*; darts e and e′ are flagged if $b_1^* < b_1$ or b_1^* is unacceptable; and $b_2^* < b_2$ or b_2^* is unacceptable, respectively. Moreover the* BfsRep(B) *is also defined as* BfsRep$(B) = \min\{b_1, b_2\}$. BfsRep(B)

Lemma 4.20. *Let $B = B_{A,T}$ be a 2-block of a 2-connected plane graph G, e be one of its virtual edges,* $e^n = \text{next}_{V(B)}(e)$ *and* $e^p = \text{prev}_{V(B)}(\text{inv}(e))$. *Also assume* $v = \text{head}(e)$ *and* $\text{rot}(G; v) = (e^n, x_0, \ldots, x_k, e^p, y_0, \ldots, y_t)$. *Then* $E(B; v)\backslash e = \{x_0, \ldots, x_k\}$.

Proof. Let $X = \{x_0, \ldots, x_k\}$, $Y = \{y_0, \ldots, y_t\}$ and $v' = \text{tail}(e)$. As the order of edges in the rotation systems are not changed while making B, $Y \cap E(B) = \varnothing$ and $E(B; v)\backslash e \subseteq X$. Assume by contrary that $E(B; v)\backslash e \neq X$. Thus there is an edge $e' \in X$ such that $e' \notin E(B; v)\backslash e$. Now consider $v^n = \text{tail}(e^n)$ and $v^p = \text{tail}(e^p)$. As B is 3-connected there are three internally disjoint paths between v^n and v^p so at least there is a path P between them in B which does not go through v and v'. But the fact that G is 2-connected allows x to have two internally disjoint path to v' and by JTC any path from x to v' goes through a vertex in the cycle $e^n \to P \to \text{inv}(e^p)$ (See Figure 4.7). This means that x is reachable from a vertex of P after removing v and v'. But all vertices of P are in B so x is in the same attached component as other vertices of B and thus $x \in B$ which is a contradiction. □

4.4.2 Formal Definition of Representation

Definition 4.21. *The* BfsRep(G) *of a graph G is* unacceptable *if the minimum code(s) from* unacceptable BfsRep

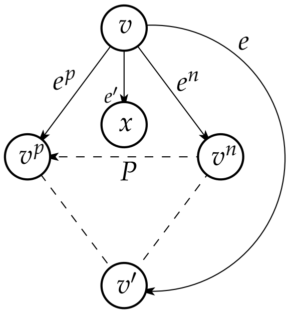


Figure 4.7: Edges of 2-blocks are consecutive

which it is computed is(are) unacceptable.

For example in Definition 4.17 BfsRep$(P) = \min\{b_1, b_2\}$. So if $b_1 < b_2$ and b_1 is unacceptable, BfsRep(P) is too. Note that if $b_1 = b_2$, then BfsRep(P) is unacceptable only if both b_1 and b_2 are unacceptable.

Rep(G)

Definition 4.22. *The representation of a plane graph G with labelled edges is denoted by* Rep(G) *which is defined in Algorithm 4.1.*

Theorem 4.23. *The algorithm Rep is a well-defined mathematical function on the set of all plane graphs.*

Proof. If G is a cycle or a simple 3-connected graph the result is obtained as Bcode(G) is a well-defined function because the rotation system is reconstructible from it. For the rest of the cases the result can be proven by the induction on the number of edges of *Rep*. We need to consider three cases:

If G has some 2-valent vertices, its maximal induced path subgraphs are unique because it is not a cycle. Also by the maximality the paths share no edges and vertices except their endpoints so the order of applying path reductions does not change the output graph.

Else If G has some multiedges, its maximal induced multiedges subgraphs are unique. Thus $C = -\sum_{i=1}^{n} |E(S_i)|^2$ is well-defined too. Also by the maximality the multiedges does not share any edges so the order of applying multiedges reductions does not change the output graph.

Else G is not 3-connected and has no 2-valent vertex. So the set of 2-blocks of G by Theorem 4.8 is the set of leaves of any 2-block tree of it. Also no 2-blocks share the same edge except possibly their virtual edges otherwise the 2-blocks would have made from the same attached component and so were equal. Thus the order of applying 2-block reductions does not change the output graph. Note that we only use the 2-blocks with just one virtual edge, otherwise replacing them with edges are not possible as the endpoint of new edges would be removed because of some other 2-blocks.

Algorithm 4.1 Computes Representation of Plane Graphs

```
function Rep($G$: Plane graph)
    $\mathcal{RT}$ = unknown                                        ▷ Recurision type
    $\mathcal{S}$ = unknown                                         ▷ Subgraphs to be reduced
    $\mathcal{R}$ = unknown                                         ▷ Reduction operator

    if $G$ has at least a 2-valent vertex and is not a cycle then
        $\mathcal{S}$ = list of all maximal induced path subgraphs
        $\mathcal{RT} \leftarrow Path$ and $\mathcal{R} = PR$                          ▷ Path reduction
    else if $G$ has some consecutive multiple edges then
        $\mathcal{S}$ = list of all maximal induced multiedges subgraphs
        $\mathcal{RT} = MultiEdge$ and $\mathcal{R} = MR$                  ▷ Multiedges reduction
    else if $G$ is 2-connected then
        $\mathcal{S}$ = list of all 2-blocks of $G$ with only one virtual edge
        $\mathcal{RT} = TwoBlock$ and $\mathcal{R} = BR$                      ▷ 2-block reduction
    else
        return $[-1, \text{Bcode}(G)]$
    end if

    Sort $\mathcal{S}$ based on the BfsRep of its items
    for each $D \in \mathcal{S}$ do
        Index($D$) = $|\{D' : \text{BfsRep}(D') < \text{BfsRep}(D) \text{ and } D' \in \mathcal{S}\}|$
    end for
    $A = []$                                                        ▷ Empty vector
    for $i = 1 \rightarrow |\mathcal{S}|$ do
        if BfsRep($\mathcal{S}[i]$) is unaccepted then
            $A = A$ concatenate with $[\text{BfsRep}(\mathcal{S}[i]), 1]$
        else
            $A = A$ concatenate with $[\text{BfsRep}(\mathcal{S}[i]), 0]$
        end if
    end for

    $G' = G$
    $k = \max_{e \in D_E(G)} \text{label}(e) + 1$
    for $i = 1 \rightarrow |\mathcal{S}|$ do
        $G' = \mathcal{R}(G'; \mathcal{S}[i], k + 2 \times \text{Index}(\mathcal{S}[i]))$          ▷ Apply reduction
    end for

    if $\mathcal{RT} = Path$ then
        return $[-4, A, Rep(G')]$
    else if $\mathcal{RT} = MultiEdge$ then
        $C = -\sum_{i=1}^{n} |E(S_i)|^2$
        return $[-3, C, A, Rep(G')]$
    else if $\mathcal{RT} = TwoBlock$ then
        return $[-2, A, Rep(G')]$
    end if
end function
```

Based on these discussions the reduced graph is well-defined which has less edges and by the induction hypothesis its representation is well-defined. □

Definition 4.24. *A 2-connected plane graph G can be categorised as one of types $\{P, M, B, C, T\}$ defined as:*

C-**Type** *: If G a cycle graph.*

P-**Type** *: If G is not a cycle but has at least a 2-valent vertex.*

B-**Type** *: If G is simple but is neither 3-connected nor has a 2-valent vertex.*

M-**Type** *: If G does not have any 2-valent vertex but has some multiple edges.*

T-**Type** *: If G is a 3-connected simple graph.*

Lemma 4.25. *Any 2-connected plane graph has exactly one of types $\{P, M, B, C, T\}$.*

Proof. To show that any graph as at least one of these types, we prove the following equivalent claim. If G is a 2-connected graph which is neither of the types $\{P, M, B, C\}$, then G is a T-type graph. If G has none of those types, it does not have a 2-valent vertex; otherwise it would be either P-type or C-type. Also G is simple otherwise it would be M-type. Moreover G is 3-connected otherwise it would be a B-type graph. So G is simple and 3-connected which means it is a T-type graph. The proof that G has at most one of the types is straightforward as the definition of each type contradicts the others. So no graph could have two of these types. □

Theorem 4.26. *Every plane graph can be reconstructed from its representation uniquely.*

Proof. We use induction on the number of recursion used to build the representation of a graph G. For the base case that no recursion is used in Algorithm 4.1 (G is C-type or T-type), then the representation is $[-1, \text{Bcode}(G)]$ based on Line 16 but the rotation system and labels can be reconstructed from Bcode(G).

Now assume all graphs whose representation is computed by at most $n \geqslant 0$ recursion are reconstructible from their representation. Now let G be a plane graph whose representation computation needs $n + 1$ recursion (G is of type P, B or M).

In all these three cases we can reconstruct G from its representation as follows. The first element of the representation indicates which type G is: -4, -3 and -2 indicate P, M and B types, respectively (Lines 39,42 and 44). Also, by the induction hypothesis we can reconstruct G', the graph used for the recursion, from its representation. Then, we can use the part A of the representation to find BfsRep of those sub-objects (paths, multiedges and 2-blocks). Finally, by replacing edges whose labels belongs to the interval $[k, k + 2 \times (|\mathcal{S}| - 1)]$ with their corresponding sub-objects, the graph G will be obtained. □

4.5 Canonicity Check

In this section we introduce our canonicity check algorithm based on representation of graphs (Definition 4.22). First in Definition 4.27 the canonicity is defined then how canonicity check works for C-Type and T-Type (Section 4.5.4), P-Type (Section 4.5.1), M-Type (Section 4.5.2) and B-Type (Section 4.5.3).

Definition 4.27. *A 2-connected plane graph is said to have the* canonical embedding *or to be canonical if it has the least representation amongst the representations of all its embeddings.*

canonical embedding

Lemma 4.28. *For a marked plane graph G, if there is a subgraph X in the set S of Algorithm 4.1 whose* BfsRep *is unacceptable, then G is not canonical.*

Proof. If BfsRep(X) is not canonical, by the definition, there is another embedding of X or one of its subgraphs which has a smaller code. So replacing that subgraph with its better embedding, decreases the code of that subgraph, BfsRep(X) and as a result $Rep(G)$. □

Lemma 4.29. *For any edge e of a plane graph G*

$$\text{Bcode}^*(e) \leqslant \text{Bcode}(e)$$

Proof. Lets assume Bcode(e) and Bcode$^*(e)$ are defined as in Definitions 4.9 and 4.15.

$$\text{Bcode}(e) = \left[c^1, \left(r_1^1, l_1(1)\right), \cdots, \left(r_{d_1}^1, l_{d_1}(1)\right), 0, \cdots, c^n, (r_1^n, l_1(n)), \cdots, (r_{d_n}^n, l_{d_n}(n)), 0\right]$$

$$\text{Bcode}^*(e) = \left[c^1, \left(r_1^1, l_1^*(1)\right), \cdots, \left(r_{d_1}^1, l_{d_1}^*(1)\right), 0, \cdots, c^n, (r_1^n, l_1^*(n)), \cdots, (r_{d_n}^n, l_{d_n}^*(n)), 0\right]$$

Now considering the first difference between these two sequences. The difference happens where $l_j(i) \neq l_j^*(i)$ for some i and j as the rest of values are the same in both sequences. This means that $e_j(i) \in S \cup \overline{F}$ and $l_j(i) \neq l_j^*(i)$ so $l'_j(i) = l_j^*(i)$. Assume to the contrary that Bcode$^*(G;e) >$ Bcode$(G;e)$. So $l'_j(i) = l_j^*(i) > l_j(i)$ which means $i' = \text{index}(\text{tail}(e_j(i))) < i$ which is the third case in the definition of $l_j^*(i)$. Now considering the j' for which $e_{j'}(i') = \text{inv}(e_j(i))$ we have $i = \text{index}(\text{tail}(e_{j'}(i'))) > i'$ and so $l_{j'}^*(i') = l'_{j'}(i')$ which is the second case in the definition of $l_{j'}^*(i')$. Thus $l_{j'}^*(i') = \text{label}(\text{inv}(e_{j'}(i'))) = \text{label}(e_j(i)) = l_j(i)$. But we already knew that $l_j(i) < l'_j(i) = l_{j'}(i')$ which means $l_{j'}^*(i') < l_{j'}(i')$ and this contradicts the assumption that $l_j^*(i) \neq l_j(i)$ has been the first difference because $i' < i$. □

Lemma 4.30. *For any plane graph G there is an embedding G^* marked-isomorphic to G such that* Bcode(G^*) = Bcode$^*(G)$.

Proof. Let e is one of the edges of G for which Bcode$^*(G;e)$ = Bcode$^*(G)$. Now by exchanging labels of all $e_j(i)$ for which $l_j^*(i) \neq l_j(i)$ with their inverse G^* is obtained. Note that the exchanging only occurs if $e_j(i) \notin S \cup \overline{F}$ which ensures G^* is marked-isomorphic to G. □

Lemma 4.31. *Let G_1 be a marked plane graph and G_2 be a copy of it except for some starred and/or flagged edges the labels are exchanged with their inverse. Then* $\text{Bcode}^*(G_1; e) = \text{Bcode}^*(G_2; e)$ *for all edges e.*

Proof. For an arbitrary edge e assume $\text{Bcode}^*(G_1; e)$ be the sequence

$$\left[c^1, \left(r_1^1, l_1^*(1)\right), \cdots, \left(r_{d_1}^1, l_{d_1}^*(1)\right), 0, \cdots, c^n, (r_1^n, l_1^*(n)), \cdots, (r_{d_n}^n, l_{d_n}^*(n)), 0\right].$$

Now considering $\text{Bcode}^*(G_2; e)$ we have the same values for $c(i)$ and $r(i)$. So we should check $l_j^*(i)$ to see if it gives the same value for both G_1 and G_2. If $e_j(i) \notin S \cup \overline{F}$, then $l_j^*(i) = l_j(i)$ which is the same for both graphs. In the remaining case the result is the same because $\{l_j(i), l'_j(i)\} = \{l'_j(i), l_j(i)\}$ so the minimum and maximum will be unchanged. Thus $\text{Bcode}^*(G_1; e) = \text{Bcode}^*(G_2; e)$. □

Corollary 4.32. *Let G_1 be a marked graph and G_2 be a copy of it except for some starred and/or flagged edges the labels are exchanged with their inverse. Then* $\text{Bcode}^*(G_1) = \text{Bcode}^*(G_2)$.

Proof. By Lemma 4.31 for all edges $\text{Bcode}^*(G_1; e) = \text{Bcode}^*(G_2; e)$ which means:

$$\text{Bcode}^*(G_1) = \min_{e \in D_E(G_1)} \text{Bcode}^*(G_1; e) = \min_{e \in D_E(G_2)} \text{Bcode}^*(G_2; e) = \text{Bcode}^*(G_2). \qquad \square$$

Based on the categorisation of Definition 4.24 the canonicity check can be subdivided into checking for each category similar to the following pseudocode. The procedures used in this pseudocode are defined in Sections 4.5.1 to 4.5.4.

4.5.1 Canonicty of P-Type Plane Graphs

In this section it is discussed how a P-type plane graph can be checked if it is canonical or not.

Theorem 4.33. *If G is a canonical P-type graph, then for any maximal induced subpath $P = v_0e_1v_1 \ldots e_nv_n$ of G, then* $\text{BfsRep}(P) = \min\{\text{Bcode}^*(P; e_1), \text{Bcode}^*(P; \text{inv}(e_n))\}$.

Proof. Assume to the contrary that G_1 is canonical but there is a path $P_1 = v_0e_1v_1 \ldots e_nv_n$ such that $\text{BfsRep}(P_1) \neq \text{Bcode}^*(P_1; e_1)$ and $\text{BfsRep}(P_1) \neq \text{Bcode}^*(P_1; \text{inv}(e_n))$. So based on the definition of BfsRep and Bcode* the first difference point in the sequences $\text{BfsRep}(P_1)$ and $\text{Bcode}^*(P_1; e_1)$ is when we have an starred edge e such that $\text{label}(\text{inv}(e)) < \text{label}(e)$. Thus $\text{Bcode}^*(P_1; e_1) < \text{BfsRep}(P_1)$ and similarly the first difference in $\text{BfsRep}(P_1)$ and $\text{Bcode}^*(P_1; \text{inv}(e_n))$ is when an edge e' is reached such that $\text{label}(\text{inv}(e')) < \text{label}(e')$ which means $\text{Bcode}^*(P_1; \text{inv}(e_n)) < \text{BfsRep}(P_1)$.

Let G_2 be the copy of G_1 except in e, $\text{inv}(e)$, e' and $\text{inv}(e')$ for which their label are exchanged with their inverse. Then assume P_2 is the path $e_1, \ldots, e_n$ in G_2. As e is an starred edge, G_2 is isomorphic to G_1 as changing these labels is the same as taking mirror for the subgraph replaced by e and $\text{inv}(e)$. Also as in G we had label(inv(e)) < label(e) and label(inv(e')) < label(e'), Bcode($P_2; e_1$) < Bcode(P_1, e_1) and Bcode(P_2; inv(e_n)) < Bcode(P_1, inv(e_n)) which means BfsRep(P_2) < BfsRep(P_1).

Now based on Definition 4.22, the $Rep(G_1) = [-4, A_1, Rep(G'_1)]$ and $Rep(G_2) = [-4, A_2, Rep(G'_2)]$ for some A_1, A_2, G'_1 and G'_2. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be the value of $\mathcal{S}$ in Definition 4.22 for G_1 and G_2 respectively. Based on the way G_2 is defined from G_1 it can be obtained that $\mathcal{S}_1 \backslash \mathcal{S}_2 = \{\text{BfsRep}(P_1)\}$ and $\mathcal{S}_2 \backslash \mathcal{S}_1 = \{\text{BfsRep}(P_2)\}$. But BfsRep($P_2$) < BfsRep($P_1$) that means $A_2 < A_1$ because A_1 and A_2 are made from $\mathcal{S}_1$ and $\mathcal{S}_2$. So as a result $Rep(G_2) < Rep(G_1)$ which is a contradiction with the assumption that G_1 is canonical. □

Theorem 4.34. *Let G be a P-type plane labelled graph. Then G is canonical if and only if:*

- *For every maximal induced subpath $P = v_0 e_1 v_1 \ldots e_n v_n$ of G,* BfsRep(P) $\in$ {Bcode*($P; e_1$), Bcode*(P; inv(e_n))}.
- *None of the maximal induced subpath P of G have unacceptable* BfsRep.
- *The graph G' obtained after applying reductions in Definition 4.22 is canonical.*

Proof. (⇒) If G is canonical by Theorem 4.33 and Lemma 4.28 it satisfies the first and second conditions. To prove the third condition assume to the contrary that G' is not canonical then there is another embedding of G' say G'_c which is canonical. So $Rep(G'_c) < Rep(G')$. Now applying the inverse of reductions on the G'_c a graph G_c is obtained which is isomorphic to G and so has the same set of maximal paths. Thus $Rep(G) = [-4, A, Rep(G')]$ and $Rep(G_c) = [-4, A, Rep(G'_c)]$ for the same value of A because of condition two. But by canonicity of G_c, $Rep(G'_c) < Rep(G')$ which means $Rep(G_c) < Rep(G)$ and this contradicts the assumption that G is canonical

(⇐) Let G be a plane graph satisfying the criteria and G_a be an arbitrary embedding of G. As G and G_a are isomorphic they have the same maximal subpaths and so $Rep(G) = [-4, A, Rep(G')]$ and $Rep(G_a) = [-4, A, Rep(G'_a)]$ for the same values A because none of the maximal subpaths is unacceptable. Also G' is isomorphic to G'_a. So canonicity of G' guarantees that $Rep(G') \leqslant Rep(G'_a)$ and as a result $Rep(G) \leqslant Rep(G_a)$. Therefore, G has the least representation amongst all embedding which means G is canonical. □

Theorem 4.34 gives a simple way to recognise if a P-type graph is canonical or not. It is enough to apply the path reductions and then check if the obtained graph is canonical or not. Algorithm 4.2 shows how this idea can be implemented.

4.5.2 Canonicty of M-Type Plane Graphs

In this section it is discussed how a M-type plane graph can be checked if it is canonical or not. Two necessary conditions for a M-type plane graph to be canonical is proven. Then adding another criterion, a necessary and sufficient condition for canonicity of M-type plane graphs is defined.

Algorithm 4.2 Canonical Testing for P-Type Graphs

```
function IsCanonPath($G$: Plane graph)
    $\mathcal{P}$ = list of all maximal induced path subgraphs
    for each $P \in \mathcal{P}$ do
        $r(P) = \text{BfsRep}(P)$
        if $r(P) \notin \{\text{Bcode}^*(P; e_1), \text{Bcode}^*(P; \text{inv}(e_n))\}$ then
            return false
        end if
        if $r(P)$ is unaccaptable then
            return false
        end if
    end for

    for each $P \in \mathcal{P}$ do
        $\text{Index}(P) = |\{P' \in \mathcal{P} : r(P') < r(P)\}|$
    end for

    $k = \max_{e \in D_E(G)} \text{label}(e) + 1$
    for $i = 1 \to |\mathcal{P}|$ do
        $G' = \mathcal{PR}(G'; \mathcal{P}[i], k + 2 \times \text{Index}(\mathcal{P}[i]))$        ▷ Apply path reduction
    end for

    return IsCanon($G'$)
end function
```

Theorem 4.35. *Let G be a M-type graph containing some multiedges from vertices v_1 and v_2 which are $e_1, \ldots, e_n$ in the same order as $\text{rot}(v_1)$. Then G is not canonical; unless $e_1, \ldots, e_n$ are contiguous.*

Proof. Let G be one such graph. We prove $e_1, \ldots, e_n$ should be consecutive; otherwise G is not canonical. Assume $e_1, \ldots, e_n$ are not contiguous and $\text{rot}(v_1)$ and $\text{rot}(v_2)$ are

$$\begin{aligned}\text{rot}(v_1) &= (e_1, x_{1,0}, \ldots, x_{1,i_1}, e_2, x_{2,0}, \ldots, x_{2,i_2}, \ldots, e_n, x_{n,0}, \ldots, x_{n,i_n})\\ \text{rot}(v_2) &= (\text{inv}(e_n), y_{n,0}, \ldots, y_{n,j_n}, \ldots, \text{inv}(e_2), y_{2,0}, \ldots, y_{2,j_2}, \text{inv}(e_1), y_{1,0}, \ldots, y_{1,j_1})\end{aligned}$$

Now consider graph G' obtained from G by modification of $\text{rot}(v_1)$ and $\text{rot}(v_2)$ as

$$\begin{aligned}\text{rot}(v_1) &= (e_1, \ldots, e_n, x_{1,0}, \ldots, x_{1,i_1}, \ldots, x_{n,0}, \ldots, x_{n,i_n})\\ \text{rot}(v_2) &= (\text{inv}(e_n), \ldots, \text{inv}(e_1), y_{n,0}, \ldots, y_{n,j_n}, \ldots, y_{1,0}, \ldots, y_{1,j_1})\end{aligned}$$

this way $e_1, \ldots, e_n$ become consecutive and G' is still planar and also isomorphic to G. Now if G has no contiguous multiple edges between no two vertices, then based on the definition $Rep(G)$ is of form $[-2, \ldots]$ or $[-1, \ldots]$ but as G' has some consecutive multiedges $Rep(G')$ is of form $[-3, C', \ldots]$ which means G is not canonical. So assume G has at least some consecutive multiedges which means $Rep(G)$ is of form

$[-3, C, \ldots]$. But still $Rep(G) > Rep(G')$ because $C > C'$. □

Theorem 4.36. *Let G be a M-type graph containing some contiguous multiedges from vertices v_1 and v_2. Then if G is canonical,* $\text{BfsRep}(M) = \min\{\text{Mcode}^*(M; v_1), \text{Mcode}^*(M; v_2)\}$.

Proof. Assume for the contrary that $L^s < L'^s$ but $L \neq L^s$ for some G satisfying the theorem criteria. Now for all $1 \leqslant i \leqslant j \leqslant n$ exchanging e_i with e_j and correspondingly e'_{n-i} with e'_{n-j} in rot(v_1) and rot(v_2), a new embedding G' of G is obtained.

Now considering their representation, they are the same except subsequence A in Definition 4.22. Moreover it is not hard to see that A for G' is less than G because the set $\mathcal{S}$ is the same for both except in the changed part for which BfsRep computed for the modified part is reduced. So $Rep(G') < Rep(G)$ which means G is not canonical.

The same discussion can be applied for the case when $L^s = L'^s$ to prove that if neither $L = L^s$ nor $L' = L'^s$ hold, then the graph is not canonical. □

Theorem 4.37. *Let G be a canonical M-type containing some contiguous multiedges from vertices v_1 and v_2. Also assume M is a subgraph of G induced by maximal set of edges $e_1, \ldots, e_n$ from v_1 to v_2. Then* $\text{BfsRep}(M) \in \{\text{Bcode}^*(M; e_1), \text{Bcode}^*(M; \text{inv}(e_n))\}$.

Proof. Let G_1 be a canonical M_1-type graph and M a subgraph of G_1 satisfying the criteria except $\text{BfsRep}(M_1) \neq \text{Bcode}^*(M_1; e_1)$ and $\text{BfsRep}(M_1) \neq \text{Bcode}^*(M_1; \text{inv}(e_n))$. So based on the definition of BfsRep and Bcode* the first difference point in the sequences $\text{BfsRep}(P_1)$ and $\text{Bcode}^*(P_1; e_1)$ is when we have an starred edge e such that label(inv(e)) < label(e). Thus $\text{Bcode}^*(M_1; e_1) < \text{BfsRep}(M_1)$ and similarly the first difference in $\text{BfsRep}(M_1)$ and $\text{Bcode}^*(M_1; \text{inv}(e_n))$ is when an edge e' is reached such that label(inv(e')) < label(e') which means $\text{Bcode}^*(M_1; \text{inv}(e_n)) < \text{BfsRep}(M_1)$.

Let G_2 be the copy of G_1 except in e, inv(e), e' and inv(e') for which their label are exchanged with their inverse. Then assume M_2 is the subgraph of G induced by edges $e_1, \ldots, e_n$. As e is an starred edge, G_2 is isomorphic to G_1 as changing these labels is the same as taking mirror for the subgraph replaced by e and inv(e). Also as in G we had label(inv(e)) < label(e) and label(inv(e')) < label(e'), $\text{Bcode}(M_2; e_1) < \text{Bcode}(M_1, e_1)$ and $\text{Bcode}(M_2; \text{inv}(e_n)) < \text{Bcode}(M_1, \text{inv}(e_n))$ which means $\text{BfsRep}(M_2) < \text{BfsRep}(M_1)$.

Now based on the Definition 4.22, the $Rep(G_1) = [-3, C, A_1, Rep(G'_1)]$ and $Rep(G_2)$ $= [-4, C, A_2, Rep(G'_2)]$ for some C, A_1, A_2, G'_1 and G'_2. Now based on Definition 4.22, $S_1 \backslash S_2 = \{\text{BfsRep}(M_1)\}$ and $S_2 \backslash S_1 = \{\text{BfsRep}(M_2)\}$. But $\text{BfsRep}(M_2) < \text{BfsRep}(M_1)$ that means $A_2 < A_1$ and as result $Rep(G_2) < Rep(G_1)$ which contradicts the assumption that G_1 is canonical. □

Theorem 4.38. *Let G be a canonical M-type graph. Then G is canonical if and only if:*

- *All multiedges satisfy the criteria of Theorems 4.35, 4.36 and 4.37.*
- *None of the maximal multiedges M of G have unacceptable* BfsRep.
- *The graph G′ obtained after applying reductions in Definition 4.22 is canonical.*

Proof. ($\Rightarrow$) If G is canonical then it satisfy the criteria of Theorems 4.35, 4.36 and 4.37; and Lemma 4.28. To prove the third condition assume to the contrary that G' is not canonical then there is another embedding of G' say G'_c which is canonical. Now applying the inverse of reductions on the G'_c a graph G_c is obtained which has the same set of multiple edges and so is isomorphic to G, but $Rep(G_c) < Rep(G)$ because $Rep(G) = [-3, C, A, Rep(G')]$ and $Rep(G_c) = [-3, C, A, Rep(G'_c)]$ for the same values C, S and A; but $Rep(G'_c) < Rep(G')$ because G'_c is canonical and G' is not. So $Rep(G_c) < Rep(G)$ which contradicts the assumption that G is canonical

($\Leftarrow$) Assume G satisfies the criteria and assume G_c is the embedding of G which is canonical. By the fact that G_c is canonical it satisfies the criteria of Theorems 4.35, 4.36 and 4.37. Also considering that G is isomorphic to G_c, the set $\mathcal{S}$ defined in Definition 4.22 would be the same for G and G_c. So $Rep(G) = [-3, C, A, Rep(G')]$ and $Rep(G_c) = [-3, C, A, Rep(G'_c)]$ for the same values C and A because none of the subgraphs is unacceptable; but G' is canonical by the assumption so $Rep(G') \leqslant Rep(G'_c)$. Thus $Rep(G) \leqslant Rep(G_c)$ which means G is canonical. □

Theorem 4.38 gives a simple way to recognise if a M-type graph is canonical or not. One could check every maximal induced multiedges to see if the edges are consecutive and in the correct order. If at least one of them violates the criteria the graph can be rejected instantly. If none of the subgraphs cause rejectio, the reduced graph after applying all multiedges reductions can be used to check whether the original graph was canonical or not. Algorithm 4.3 shows how we can determine if a M-type graph is canonical or not.

4.5.3 Canonicty of B-Type Plane Graphs

In this section we discuss how a B-type plane graph can be checked if it is canonical or not.

Theorem 4.39. *A B-type plane graph G is canonical if and only if*

- *All 2-blocks of G are canonical.*
- *None of the 2-blocks B of G have unacceptable* BfsRep*.*
- *The graph G' obtained after applying reductions in Definition 4.22 is canonical.*

Proof. ($\Rightarrow$) If G is canonical by Lemma 4.28 it satisfies the second condition. Assume to the contrary that G is canonical and B is one its 2-blocks which is not canonical. Now consider the graph G_c obtained from G by replacing B with its canonical embedding B_c. Therefore, $[-2, A, Rep(G')] = Rep(G) > Rep(G_c) = [-2, A_c, Rep(G'_c)]$. The reason behind is that A and A' are sorted elements of two sets say S and S_c such that $S \backslash S_c = \{\text{BfsRep}(B)\}$ and $S_c \backslash S = \{\text{BfsRep}(B_c)\}$. But $Rep(B) > Rep(B_c)$ which means $A > A_c$ and as a result $Rep(G) > Rep(G_c)$. This contradicts the assumption that G is canonical.

To prove the third criteria, assume to the contrary that G' is not canonical then there is another embedding of G' say G'_c which is canonical. Now applying the

Algorithm 4.3 Canonical Testing for M-Type Graphs

```
function IsCanonMultiEdge($G$: Plane graph)
    $\mathcal{M}$ = list of all maximal induced multiedges subgraphs
    for each $M \in \mathcal{M}$ do
        if edges of $M$ are not consecutive in $G$ then
            return false                                          ▷ Check Theorem 4.35
        end if

        $v$ = one of the vertices of $M$ and $v'$ = the other vertex of $M$
        $b^* = \min\{\text{Mcode}^*(M; v), \text{Mcode}^*(M; v')\}$
        if $b^*$ is unaccaptable then
            return false
        end if
        if $G = M$ then
            $b_1 = \min_{e \in \text{rot}(v)} \text{Mcode}(M; e)$
            $b_2 = \min_{e \in \text{rot}(v')} \text{Mcode}(M; e)$
        else
            $e$ = the $\overrightarrow{vv'}$ dart in $M$ whose previous edge is not in $M$
            $e'$ = the $\overrightarrow{v'v}$ dart in $M$ whose previous edge is not in $M$
            $b_1 = \text{Mcode}(M; e)$
            $b_2 = \text{Mcode}(M; e')$
        end if
        $r(M) = \min b_1, b_2$
        if $r(B)$ is unaccaptable then
            return false
        end if
        if $b^* \neq r(M)$ then
            return false
        end if
    end for

    for each $M \in \mathcal{M}$ do
        $\text{Index}(M) = |\{M' \in \mathcal{M} : r(M') < r(M)\}|$
    end for
    $k = \max_{e \in D_E(G)} \text{label}(e) + 1$
    for $i = 1 \to |\mathcal{M}|$ do
        $G = \mathcal{MR}(G; \mathcal{M}[i], k + 2 \times \text{Index}(\mathcal{M}[i]))$      ▷ Apply path reduction
    end for

    return IsCanon($G'$)                                          ▷ Check Theorem 4.38
end function
```

inverse of reductions on the G'_c a graph G_c is obtained which has the same set of multiple edges and so is isomorphic to G, but $Rep(G_c) < Rep(G)$ because $Rep(G) = [-2, A, Rep(G')]$ and $Rep(G_c) = [-2, A, Rep(G'_c)]$ for the same value of A because of

Algorithm 4.4 Canonical Testing for B-Type Graphs

```
function IsCanon2Block($G$: Plane graph)
    $\mathcal{B}$ = list of all 2-blocks
    for each $B \in \mathcal{B}$ do
        Let $\{v, v'\}$ be the 2-cut that $B$ is attached to
        $e_s$ = the $\overrightarrow{vv'}$ dart in $B$ whose previous edge is not in $B$
        $e_e$ = the $\overrightarrow{vv'}$ dart in $B$ whose next edge is not in $B$
        $e'_s$ = the $\overrightarrow{v'v}$ dart in $B$ whose previous edge is not in $B$
        $e'_e$ = the $\overrightarrow{v'v}$ dart in $B$ whose next edge is not in $B$

        $b^* = \min\{\text{Bcode}^*(B; e_s), \text{Bcode}^*_{\text{M}}(B; e_e), \text{Bcode}^*(B; e'_s), \text{Bcode}^*_{\text{M}}(B; e'_e)\}$
        if $b^*$ is unaccaptable then
            return false
        end if

        $b_1 = \text{Bcode}(B; e_s)$ and $b_2 = \text{Bcode}(B; e'_s)$
        $r(B) = \min\{b_1, b_2\}$
        if $r(B)$ is unaccaptable then
            return false
        end if
        if $b^* \neq r(B)$ then
            return false
        end if
    end for

    for each $B \in \mathcal{B}$ do
        $\text{Index}(B) = |\{B' \in \mathcal{B} : r(B') < r(B)\}|$
    end for

    $k = \max_{e \in D_E(G)} \text{label}(e) + 1$
    for $i = 1 \rightarrow |\mathcal{B}|$ do
        $G' = \mathcal{BR}(G'; \mathcal{B}[i], k + 2 \times \text{Index}(\mathcal{B}[i]))$          ▷ Apply 2-block reduction
    end for

    return IsCanon($G'$)
end function
```

the second condition; but $Rep(G'_c) < Rep(G')$ because G'_c is canonical and G' is not. So $Rep(G_c) < Rep(G)$ which contradicts the assumption that G is canonical.

($\Leftarrow$) Assume G satisfies the criteria and let G_c be the canonical embedding of G, so by the previous discussions, all 2-blocks of G_c are canonical and the graph G'_c obtained from G_c after reduction is canonical too. Also the set of 3-connected subgraphs is independent of the embedding so the set of 2-blocks of G and G_c are the same graphs and as all are canonical, they have the same embedding. So

$Rep(G) = [-2, A, Rep(G')]$ and $Rep(G_c) = [-2, A, Rep(G'_c)]$ for the same value of A because none of the 2-blocks is unacceptable; but G' is canonical by the assumption so $Rep(G') \leqslant Rep(G'_c)$. Thus $Rep(G) \leqslant Rep(G_c)$ which means G is canonical. □

4.5.4 Canonicty of T-Type and C-Type Plane Graphs

In this section we discuss how a T-type or a C-type plane graph can be checked if it is canonical or not.

An cycle graph with no flagged and starred edge has only one embedding on the sphere. But if C has some starred edges, then swapping the label of any of them with its inverse edge produces isomorphic graphs.

Theorem 4.40. *Any cycle graph C is canonical if and only if for* $\text{Bcode}(C) = \text{Bcode}^*(C)$ *and* $\text{Bcode}(C)$ *is acceptable.*

Proof. ($\Rightarrow$) Assume C is a canonical cycle graph, by Lemma 4.30 there is an embedding C^* such that $\text{Bcode}(C^*) = \text{Bcode}^*(C)$ and by Lemma 4.29 we have $\text{Bcode}(C) \geqslant \text{Bcode}^*(C)$ which means $\text{Bcode}(C) \geqslant \text{Bcode}(C^*)$. So C is not canonical unless $\text{Bcode}(C) = \text{Bcode}(C^*) = \text{Bcode}^*(C)$. Note that by Lemma 4.28 if $\text{Bcode}(C)$ is not acceptable then it is not canonical.

($\Leftarrow$) Let C be a cycle graph for which $\text{Bcode}(C) = \text{Bcode}^*(C)$ and C_c be the canonical embedding of C. As a cycle graph has a unique embedding except the flagged and starred edges that can swap their labels by their inverse, by Corollary 4.32 $\text{Bcode}^*(C_c) = \text{Bcode}^*(C)$. But $\text{Bcode}^*(C_c) \leqslant \text{Bcode}(C_c)$ by Lemma 4.29 which means $\text{Bcode}(C) \leqslant \text{Bcode}(C_c)$ and so C is canonical. □

Algorithm 4.5 Canonical Testing for C-Type Graphs

```
function IsCanonCycle($G$: Plane graph)
    $b^* = \text{Bcode}^*(G)$
    if $b^*$ is unaccaptable then
        return false
    end if
    $b = \text{Bcode}(G)$
    if $b$ is unaccaptable then
        return false
    end if
    if $b = b^*$ then
        return true
    else
        return false
    end if
end function
```

Theorem 4.41. *A simple 3-connected marked graph G is canonical, if and only if* $\text{Bcode}(G) = \text{Bcode}^*(G) \leqslant \text{Bcode}^*_{\text{M}}(G)$ *and* $\text{Bcode}(G)$ *is acceptable.*

Proof. ($\Rightarrow$) Assume G is a canonical 3-connected simple graph, by Lemma 4.30 there is an embedding G^* such that $\text{Bcode}(G^*) = \text{Bcode}^*(G)$ and by Lemma 4.29 we have $\text{Bcode}(G) \geqslant \text{Bcode}^*(G)$ which means $\text{Bcode}(G) \geqslant \text{Bcode}(G^*)$. So G is not canonical unless

$$\text{Bcode}(G) = \text{Bcode}(G^*) = \text{Bcode}^*(G). \tag{4.7}$$

Also let $G_M = \text{Mir}^*(G)$ and G_M^* be the embedding of G_M such that $\text{Bcode}(G_M^*) = \text{Bcode}^*(G_M)$. As G is canonical, $\text{Bcode}(G) \leqslant \text{Bcode}(G_M^*) = \text{Bcode}^*(G_M) = \text{Bcode}^*_{\text{M}}(G)$. Combining this result with Equation 4.7 and Lemma 4.28, the desired result is obtained.

($\Leftarrow$) Let G be a 3-connected marked graph for which $\text{Bcode}(G) = \text{Bcode}^*(G) \leqslant \text{Bcode}^*_{\text{M}}(G)$ and G_c be the canonical embedding of G. A 3-connected graph has at most two embedding (mirror of each other) except for the flagged and starred edges that can swap their labels by their inverse. By Corollary 4.32, $\text{Bcode}^*(C_c) = \text{Bcode}^*(C)$ because $\text{Bcode}(G)$ is acceptable and $\text{Bcode}^*(G) \leqslant \text{Bcode}^*_{\text{M}}(G)$. But $\text{Bcode}^*(G_c) \leqslant \text{Bcode}(G_c)$ by Lemma 4.29. Therefore, $\text{Bcode}(G) \leqslant \text{Bcode}(G_c)$ and G is canonical. □

Algorithm 4.6 Canonical Testing for T-Type Graphs

```
function IsCanonTriConnected($G$: Plane graph)
    $b^* = \min\{\text{Bcode}^*(G), \text{Bcode}^*_{\text{M}}(G)\}$
    if $b^*$ is unaccaptable then
        return false
    end if
    $b = \text{Bcode}(G)$
    if $b^*$ is unaccaptable then
        return false
    end if
    if $b = b^*$ then
        return true
    else
        return false
    end if
end function
```

Finally, using Lemma 4.25 which shows there are exactly five types of 2-connected plane graphs we can design the Algorithm 4.7 to check canonicity of a 2-connected plane graph.

4.6 Conclusions

In this chapter we discussed how a 2-connected planar graph can be embedded canonically and how we can do an isomorphism rejection using the canonical embedding that we defined. Software *canemb* [63] contains the implementation of a filter for non-canonically embedded 2-connected graphs based on the theories of this chapter.

Algorithm 4.7 Canonical Testing for 2-Connected Plane Graphs

```
function IsCanon($G$: Plane graph)
    if $G$ is $P$-Type then
        return IsCanonPath($G$)
    else if $G$ is $M$-Type then
        return IsCanonMultiEdge($G$)
    else if $G$ is $B$-Type then
        return IsCanon2Block($G$)
    else if $G$ is $C$-Type then
        return IsCanonCycle($G$)
    else if $G$ is $T$-Type then
        return IsCanonTriConnected($G$)
    end if
end function
```

The fifth and sixth columns of Table 4.1 contain the number of 2-connected planar graphs computed using *canemb* and their running time which confirm the previous known results [23, 20, 111] too.

In addition, we have checked the 2-connected planar bipartite graphs too and the numbers of graphs and computation times can be found in Table 4.2. The number of planar bipartite graphs up to $n = 14$ has been known [109]. Using *canemb* we have extended this results to $n = 16$.

$\lvert V \rvert$	2-Connected	2-Connected Plane	Generation Time	2-Connected Planar	Filtering Time
2	1	1	<1s	1	<1s
3	1	1	<1s	1	<1s
4	3	3	<1s	3	<1s
5	10	10	<1s	9	<1s
6	56	61	<1s	44	<1s
7	468	564	<1s	294	<1s
8	7123	7593	<1s	2893	<1s
9	194066	123874	<1s	36496	2s
10	9743542	2262877	2s	545808	34s
11	900969091	44190279	30s	9029737	16m
12	153620333545	904777809	14m	159563559	355m

Table 4.1: Number of 2-connected generic, plane and planar graphs; and the time for generation of plane graphs using *plantri* and filtering isomorphic copies using *canemb*

	Bipartite				
$\lvert V\rvert$	2-Connected	2-Connected Plane	Generation Time	2-Connected Planar	Filtering Time
4	1	1	<1s	1	<1s
5	1	1	<1s	1	<1s
6	5	4	<1s	4	<1s
7	8	6	<1s	6	<1s
8	42	28	<1s	30	<1s
9	146	77	<1s	92	<1s
10	956	386	<1s	521	<1s
11	6643	1787	<1s	2781	<1s
12	65921	10354	<1s	18161	<1s
13	818448	62040	<1s	121835	2s
14	13442572	404093	<1s	869379	16s
15	287665498	2725484	6s	6361801	2.5m
16	8099980771	19078248	42s	47802651	24m

Table 4.2: Number of 2-connected bipartite generic [107], plane [109] and planar graphs; and the time for generation of bipartite plane graphs using *plantri* and filtering isomorphic copies using *canemb*

We mainly focused on how we can determine if a plane graph has the canonical embedding or not. But one can find the canonical embedding from any given embedding with the same recursive approach. It is enough in each step to replace subgraphs with edges whose label are based on the Bcode* and Bcode$^*_M$ instead of Bcode for P-type, B-type, C-type and T-type; and Mcode* instead of Mcode for M-type marked plane graphs. Also for testing if two plane graphs are abstract-isomorphic or not, one can compute their canonical representation (the representation of their canonical embedding) and check if they are equal or not.

block graph

A possible future direction is to extend this approach to all connected plane graphs. This extension could be done using block graphs. The *block graph of G* is the graph obtained from G by replacing every maximal 2-connected component with a vertex, and two vertices in the resulting graph are adjacent if there is an edge between two vertices of their corresponding 2-connected components. One can check the canonicity of each 2-connected component using our approach and then extend the result to its block graph. This could be specially for graphs with order more than 10. Figure 4.8 shows that after $n > 10$ the ratio of generic graphs to plane graphs increases dramatically.

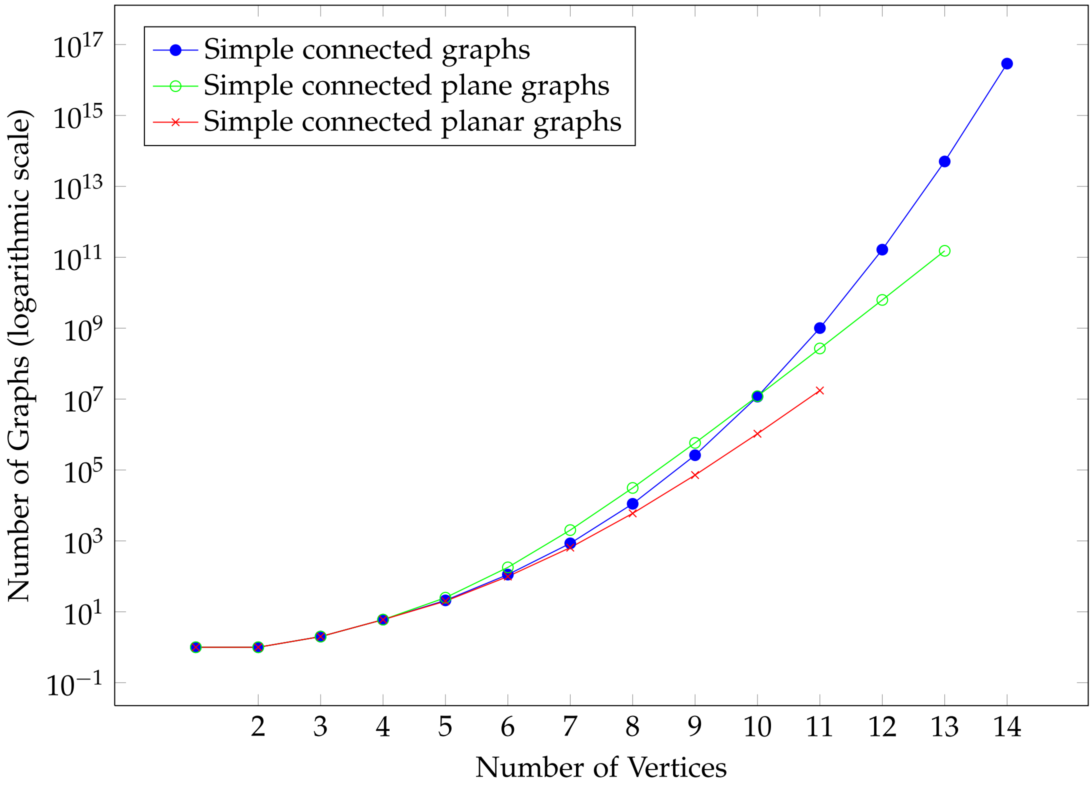


Figure 4.8: Number of simple connected generic, plane and planar graphs [23, 108, 113].

Chapter 5

Recursive Generation of 4-Face Deflatable Hypohamiltonian Graphs

5.1 Background

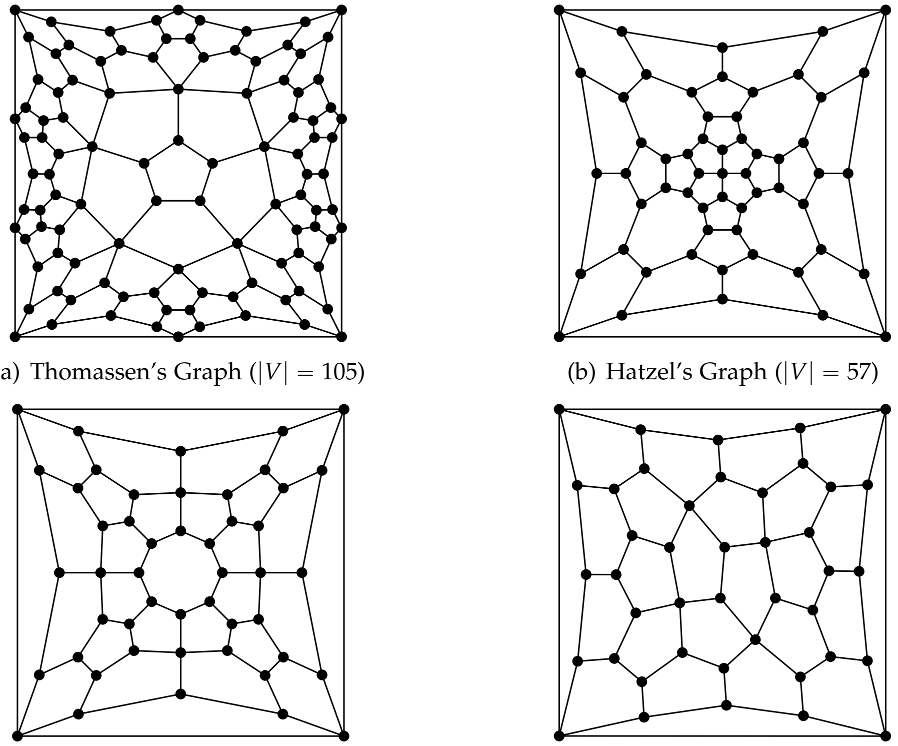
This chapter contains the material published in “Planar Hypohamiltonian Graphs on 40 Vertices” which is a joint study with B. D. McKay, P. R. J. Östergård, V. H. Pettersson and C. T. Zamfirescu [64].

5.2 Introduction

Chvátal [26] asked in 1973 whether there exist *planar* hypohamiltonian graphs, and there was a conjecture that such graphs might not exist [49]. However, an infinite family of planar hypohamiltonian graphs was later found by Thomassen [115], the smallest among them having order 105. This result was the starting point for work on finding the smallest possible order of such graphs, which has led to the discovery of planar hypohamiltonian graphs of order 57 (Hatzel [53] in 1979), 48 (C. Zamfirescu and T. Zamfirescu [126] in 2007), and 42 (Wiener and Araya [123] in 2011). These four graphs are depicted in Figure 5.1.

Grinberg [48] proved a necessary condition for a plane graph to be Hamiltonian. All graphs in Figure 5.1 have the property that one face has size 1 modulo 3, while all other faces have size 2 modulo 3. Graphs with this property are natural candidates for being hypohamiltonian, because they do not satisfy Grinberg’s condition. However, we will prove that this approach cannot lead to hypohamiltonian graphs of order smaller than 42. Consequently we seek alternative methods for finding planar hypohamiltonian graphs. In particular, we construct a certain subset of graphs with girth 4 and a fixed number of 4-faces in an exhaustive way. This collection of graphs turns out to contain 25 planar hypohamiltonian graphs of order 40.

In addition to finding record-breaking graphs of order 40, we shall prove that planar hypohamiltonian graphs exist for all orders greater than or equal to 42 (it is proved in [123] that they exist for all orders greater than or equal to 76). Similar results are obtained for *hypotraceable* graphs. We show that there is a planar hypotraceable graph of order 154 and of all orders greater than or equal to 156; the old records were 162 and 180, respectively [123].



(a) Thomassen's Graph ($|V| = 105$)

(b) Hatzel's Graph ($|V| = 57$)

(c) Zamfirescu and Zamfirescu's Graph ($|V| = 48$)

(d) Wiener and Araya's Graph (WA_{42})

Figure 5.1: Planar hypohamiltonian graphs of order 105, 57, 48, and 42

T. Zamfirescu defined $\overline{C_k^i}$ and $\overline{P_k^i}$ to be the smallest order for which there is a planar k-connected graph such that every set of i vertices is disjoint from some longest cycle and path, respectively [127]. Some of the best bounds known so far were $\overline{C_3^1} \leqslant 42$, $\overline{C_3^2} \leqslant 3701$, $\overline{P_3^1} \leqslant 164$ and $\overline{P_3^2} \leqslant 14694$, which were found based on a planar hypohamiltonian graph on 42 vertices [123]. We improve upon these bounds using our graphs to $\overline{C_3^1} \leqslant 40$, $\overline{C_3^2} \leqslant 2625$, $\overline{P_3^1} \leqslant 156$ and $\overline{P_3^2} \leqslant 10350$.

The chapter is organized as follows. In Section 5.3 we define Grinbergian graphs and prove theorems regarding their hypohamiltonicity. In Section 5.4 we describe generation of certain planar graphs with girth 4 and a fixed number of 4-faces, and show a summary of hypohamiltonian graphs found among them. In Section 5.5 we present various corollaries based on the new hypohamiltonian graphs. The paper is concluded in Section 5.6.

5.3 Grinbergian graphs

Consider a plane hypohamiltonian graph $G = (V, E)$, and let $\kappa(G)$, $\delta(G)$, and $\lambda(G)$ denote the vertex-connectivity, minimum degree, and edge-connectivity of G, respectively. We will use tacitly the following fact.

Theorem 5.1. $\kappa(G) = \lambda(G) = \delta(G) = 3$.

Proof. Since the deletion of any vertex in V gives a Hamiltonian graph, we have $\kappa(G) \geqslant 3$. Tutte [118] proved that every 4-connected planar graph is Hamiltonian, so $\kappa(G) \leqslant 3$. Thomassen [116] showed that V must contain a vertex of degree 3, so $\delta(G) \leqslant 3$. The result now follows from Whitney's Theorem [121, Theorem 4.1.9]. □

The set of vertices adjacent to a vertex v is denoted by $N(v)$. Let $n = |V|$, $m = |E|$, and f be the number of faces of the plane graph G. They satisfy Euler's formula $n - m + f = 2$. A *k-face* is a face of G bounded by k edges. We define

$$I_j := \{i \geqslant 3 : i \equiv j \bmod 3\},$$

and let $\mathcal{P}_j$ be the family of k-faces with $k \in I_j$.

Grinbergian graph

We call a graph *Grinbergian* if it is 3-connected, planar and of one of the following two types.

Type 1 Every face but one belongs to $\mathcal{P}_2$.

Type 2 Every face has even order, and the graph has odd order.

The motivation behind such a definition is that Grinbergian graphs can easily be proven to be non-Hamiltonian using Grinberg's Theorem. Namely, their face sizes are such that the sum in Grinberg's Theorem cannot possibly be zero. Thus, they are good candidates for hypohamiltonian graphs.

Our definition of Grinbergian graphs contains two types. One could ask, if there are other types of graphs that can be guaranteed to be non-Hamiltonian with Grinberg's Theorem based on only their sequence of face sizes. The following theorem shows that our definition is complete in this sense.

Theorem 5.2. *Consider a 3-connected simple planar graph with n vertices ($n \leqslant 42$) and F_i i-faces for each i. Then there are non-negative integers f_i, f_i' ($f_i + f_i' = F_i$) satisfying the equation $\sum_i (i-2)(f_i - f_i') = 0$ if and only if the graph is not Grinbergian.*

Proof. Since the graph is simple and 3-connected, every face must have at least 3 edges. Applying [121, Theorem 6.1.23] to the dual of the graph gives $\sum_i F_i = 2e \leqslant 6f - 12$, where f is the number of faces. Thus, the average face size is at most $6 - (12/f)$. In addition, the size of a face has to be smaller than or equal to the number of vertices in the graph.

Given a sequence of face sizes F_i, the problem of finding coefficients f_i, f_i' that satisfy the equation can be reduced to a simple knapsack problem. Namely, note that $\sum_i (i-2)(f_i - f_i') = \sum_i (i-2)(F_i - 2f_i') = \sum_i (i-2)F_i - \sum_i 2(i-2)f_i'$, so solving the equation corresponds to solving an instance of the knapsack problem where we have F_i objects of weight $2(i-2)$, and we must find a subset whose total weight is $\sum_i (i-2)F_i$. The result can be then verified with an exhaustive computer search over all sequences of face sizes that fulfill the above restrictions. □

It should be noted that the result in Theorem 5.2 most likely holds for all n, but for our purposes it suffices to prove it for $n \leqslant 42$.

By Grinberg's Theorem, Grinbergian graphs are non-Hamiltonian. Notice the difference between our definition and that of Zaks [125], who defines *non-Grinbergian* graphs to be graphs with every face in $\mathcal{P}_2$. We call the faces of a Grinbergian graph not belonging to $\mathcal{P}_2$ *exceptional*.

Theorem 5.3. *Every Grinbergian hypohamiltonian graph is of Type 1, its exceptional face belongs to* $\mathcal{P}_1$*, and its order is a multiple of* 3.

Proof. Let G be a Grinbergian hypohamiltonian graph. There are two possible cases, one for each type of Grinbergian graphs.

Type 1: Let the j-face F be the exceptional face of G (so $j \notin I_2$), and let v be a vertex of F. Vertex v belongs to F and to several, say h, faces in $\mathcal{P}_2$. The face of $G - v$ containing v in its interior has length $3h + j - 2 \pmod 3$, while all other faces have length $2 \pmod 3$. Since G is hypohamiltonian, $G - v$ must be Hamiltonian. Thus, $G - v$ cannot be a Grinbergian graph, so $3h + j - 2 \in I_2$, whence $j \in I_1$.

Type 2: As G contains only cycles of even length, it is bipartite. A bipartite graph can only be Hamiltonian if both of the parts have equally many vertices. Thus, it is not possible that $G - v$ is Hamiltonian for every vertex v, so G cannot be hypohamiltonian and we have a contradiction.

Hence, G is of Type 1, and its exceptional face is in $\mathcal{P}_1$. Counting the edges we get $2m \equiv 2(f-1) + 1 \pmod 3$, which together with Euler's formula gives

$$2n = 2m - 2f + 4 \equiv 2f - 1 - 2f + 4 \equiv 0 \pmod 3,$$

so n is a multiple of 3. □

Lemma 5.4. *In a Grinbergian hypohamiltonian graph G of Type 1, all vertices of the exceptional face have degree at least* 4.

Proof. Denote the exceptional face by Q. Now assume that there is a vertex $v \in V(Q)$ with degree 3, and consider the vertex $w \in N(v) \backslash V(Q)$. (Note that $N(v) \backslash V(Q) \neq \varnothing$, because G is 3-connected.) Let k be the degree of w. Now consider the graph G' obtained by deleting w from G. Denote the number of vertices in the faces of G that contain w by N_i ($1 \leqslant i \leqslant k$); we have $N_i \equiv 2 \pmod 3$. The number of vertices in the face of G' containing w in its interior is now $m = \sum_i (N_i - 2) \equiv 0 \pmod 3$. Assume that G' is Hamiltonian. The graph G' contains only faces in $\mathcal{P}_2$ except for one face in $\mathcal{P}_1$ and one in $\mathcal{P}_0$. The face in $\mathcal{P}_1$ and the face in $\mathcal{P}_0$ are on different sides of any Hamiltonian cycle in G', since the cycle must pass through v. The sum in Grinberg's Theorem, modulo 3, is then $(m-2) + 1 \equiv 2 \pmod 3$ or $-(m-2) - 1 \equiv 1 \pmod 3$, so G' is non-Hamiltonian and we have a contradiction. □

In Section 5.4, we will use these properties to show that the smallest Grinbergian hypohamiltonian graph has 42 vertices.

5.4 Generation of 4-face deflatable hypohamiltonian graphs

We define the operation *4-face deflater* denoted by $\mathcal{FD}_4$ which squeezes a 4-face of a plane graph into a path of length 2 (see Figure 5.2). The inverse of this operation is called *2-path inflater* which expands a path of length 2 into a 4-face and is denoted by $\mathcal{PI}_2$. In Figure 5.2 each half line connected to a vertex means that there is an edge incident to the vertex at that position and a small triangle allows zero or more incident edges at that position. For example v_3 has degree at least 3 and 4 in Figures 5.2(a) and 5.2(b), respectively. The set of all graphs obtained by applying $\mathcal{PI}_2$ and $\mathcal{FD}_4$ on a graph G is denoted by $\mathcal{PI}_2(G)$ and $\mathcal{FD}_4(G)$, respectively.

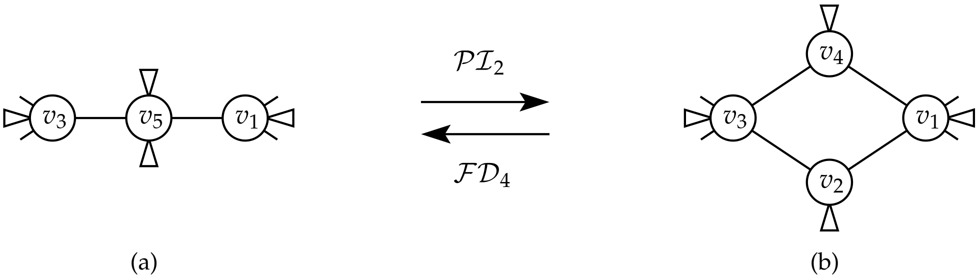


Figure 5.2: Operations $\mathcal{FD}_4$ and $\mathcal{PI}_2$

Let $\mathcal{D}_5(f)$ be the set of all simple connected plane graphs with f faces and minimum degree at least 5, which can be generated using the program *plantri* [22]. Let us denote the dual of a plane graph G by G^*. We define the family of *4-face deflatable graphs* (not necessarily simple) with f 4-faces and n vertices, denoted by $\mathcal{M}^4_f(n)$, recursively as:

$$\mathcal{M}^4_f(n) = \begin{cases} \{G^* : G \in \mathcal{D}_5(n)\}, & f = 0; \\ \\ \bigcup_{G \in \mathcal{M}^4_{f-1}(n-1)} \mathcal{PI}_2(G), & f > 0. \end{cases} \tag{5.1}$$

It should be noted that applying $\mathcal{PI}_2$ to a graph increases the number of both vertices and 4-faces by one. Then, we can filter $\mathcal{M}^4_f$ for possible hypohamiltonian graphs and we define $\mathcal{H}^4_f$ based on it as:

$$\mathcal{H}^4_f(n) = \{G \in \mathcal{M}^4_f(n) : G \text{ is hypohamiltonian}\}. \tag{5.2}$$

The function $\mathcal{H}^4_f(n)$ can be defined for $n \geqslant 20$ because the minimum face count for a simple planar 5-regular graph is 20 (icosahedron). Also it is straightforward to check that $f \leqslant n - 20$ because $\mathcal{H}^4_f(n)$ is defined based on $\mathcal{H}^4_{f-1}(n-1)$ for $f > 0$.

To test hamiltonicity of graphs, we use depth-first search with the following pruning rule: If there is a vertex that does not belong to the current partial cycle, and has fewer than two neighbours that either do not belong to the current partial cycle or are an endpoint of the partial cycle, the search can be pruned. This approach can be implemented efficiently with careful bookkeeping of the number of neighbours that do not belong to the current partial cycle for each vertex. It turns out to be reasonably fast for small planar graphs.

Finally, we define the set of *4-face deflatable hypohamiltonian graphs* denoted by $\mathcal{H}^4(n)$ as:

$$\mathcal{H}^4(n) = \bigcup_{f=0}^{n-20} \mathcal{H}^4_f(n). \tag{5.3}$$

Using this definition for $\mathcal{H}^4_f(n)$, we are able to find many hypohamiltonian graphs which were not discovered so far. The graphs found on 105 vertices by Thomassen [115], 57 by Hatzel [53], 48 by C. Zamfirescu and T. Zamfirescu [126], and 42 by Wiener and Araya [123] are all 4-face deflatable and belong to $\mathcal{H}^4_0(105)$, $\mathcal{H}^4_1(57)$, $\mathcal{H}^4_1(48)$ and $\mathcal{H}^4_1(42)$, respectively.

We have generated $\mathcal{H}^4_f(n)$ exhaustively for $20 \leqslant n \leqslant 39$ and all possible f but no hypohamiltonian graph was found, which means that for all $n < 40$ we have $\mathcal{H}^4_f(n) = \varnothing$. For $n > 39$ we were not able to finish the computation for all f due to the amount of required time. For $n = 40, 41, 42, 43$ we finished the computation up to $f = 12, 12, 11, 10$, respectively. The only values of n and f for which $\mathcal{H}^4_f(n)$ was non-empty were $\mathcal{H}^4_5(40)$, $\mathcal{H}^4_1(42)$, $\mathcal{H}^4_7(42)$, $\mathcal{H}^4_4(43)$ and $\mathcal{H}^4_5(43)$. More details about these families are provided in Tables 5.1, 5.2 and 5.3. Based on the computations we can obtain the Theorems 5.5, 5.6, 5.7 and 5.8. The complete list of graphs generated is available to download at [62].

4-Face Count	Face Sequence	Degree Sequence	Count
5	$5 \times 4, 22 \times 5$	$30 \times 3, 10 \times 4$	4
		$31 \times 3, 8 \times 4, 1 \times 5$	10
		$32 \times 3, 6 \times 4, 2 \times 5$	9
		$33 \times 3, 4 \times 4, 3 \times 5$	2
		All	25

Table 5.1: Facts about $\mathcal{H}^4_5(40)$

Theorem 5.5. *There is no planar* 4*-face deflatable hypohamiltonian graph of order less than* 40.

Theorem 5.6. *There are at least* 25 *planar* 4*-face deflatable hypohamiltonian graphs on* 40 *vertices.*

4-Face Count	Face Sequence	Degree Sequence	Count
1	$1 \times 4, 26 \times 5$	$34 \times 3, 8 \times 4$	5
		$35 \times 3, 6 \times 4, 1 \times 5$	2
7	$7 \times 4, 22 \times 5$	$30 \times 3, 12 \times 4$	4
		$31 \times 3, 10 \times 4, 1 \times 5$	28
		$32 \times 3, 8 \times 4, 2 \times 5$	57
		$33 \times 3, 6 \times 4, 3 \times 5$	49
		$33 \times 3, 7 \times 4, 1 \times 5, \times 6$	11
		$34 \times 3, 4 \times 4, 4 \times 5$	10
		$34 \times 3, 5 \times 4, 2 \times 5, 1 \times 6$	5
		$34 \times 3, 6 \times 4, 2 \times 6$	6
		$35 \times 3, 4 \times 4, 1 \times 5, 2 \times 6$	2
All	All	All	179

Table 5.2: Facts about $\mathcal{H}_1^4(42)$ and $\mathcal{H}_7^4(42)$

Theorem 5.7. *There are at least* 179 *planar* 4*-face deflatable hypohamiltonian graphs on* 42 *vertices.*

Theorem 5.8. *There are at least* 497 *planar* 4*-face deflatable hypohamiltonian graphs on* 43 *vertices.*

Lemma 5.9. *Let G be a hypohamiltonian planar graph whose faces are at least* 5*-faces except one which is a* 4*-face. Then any G' in $\mathcal{FD}_4(G)$ has a simple dual.*

Proof. As G is a simple 3-connected graph, the dual G^* of G is simple, too. Let $G' \in \mathcal{FD}_4(G)$ and assume to the contrary that G'^* is not simple.

If G'^* has some multiedges, then the fact that G^* is simple shows that either the two faces incident with v_1v_5 or with v_3v_5 in Figure 5.3(b) (we assume the first by symmetry) have a common edge v_8v_9 in addition to v_1v_5. Let v_1v_6 and v_1v_7 be the edges adjacent to v_1v_5 in the cyclic order of v_1. Note that $v_6 \neq v_7$ because $d(G'; v_1) \geqslant 3$ by Lemma 5.4. If v_1 and v_8 were the same vertex, then v_1 would be a cut vertex in G considering the closed walk $v_1v_6 \cdots v_8(=v_1)$. But this is impossible as G is 3-connected, so $v_1 \neq v_8$. Now we can see that $\{v_1, v_8\}$ is a 2-cut for G considering the closed walk $v_1v_6 \cdots v_8 \cdots v_7v_1$.

Also, if G'^* has a loop, with the same discussion, we can assume that the two faces incident with v_1v_5 are the same but then v_1 would be a cut vertex for G. Therefore, both having multiedges or having loops violate the fact that G is 3-connected. So the assumption that G'^* is not simple is incorrect, which completes the proof. □

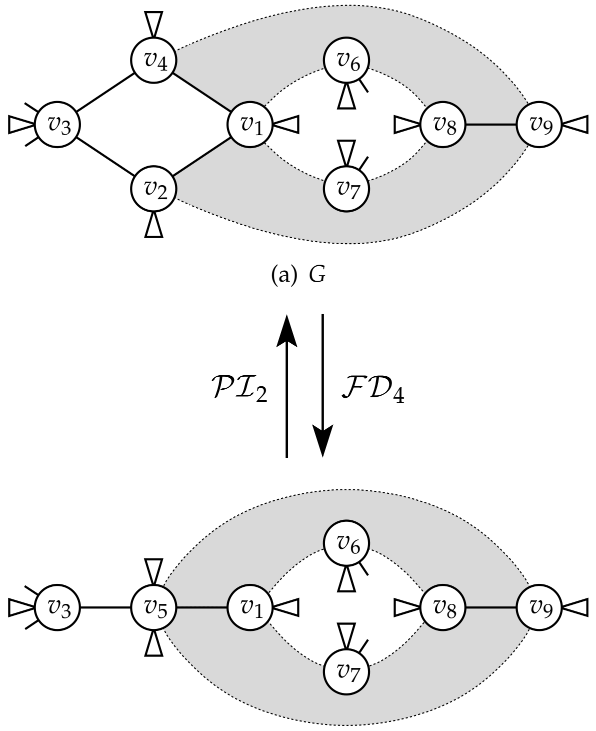
4-Face Count	Face Sequence	Degree Sequence	Count
4	$4 \times 4, 23 \times 5, 1 \times 7$	$36 \times 3, 6 \times 4, 1 \times 6$	1
		$37 \times 3, 4 \times 4, 1 \times 5, 1 \times 6$	1
5	$5 \times 4, 22 \times 5, 1 \times 8$	$34 \times 3, 9 \times 4$	8
		$35 \times 3, 7 \times 4, 1 \times 5$	20
		$36 \times 3, 5 \times 4, 2 \times 5$	19
		$37 \times 3, 3 \times 4, 3 \times 5$	1
		$37 \times 3, 4 \times 4, 1 \times 5, 1 \times 6$	1
	$5 \times 4, 24 \times 5$	$32 \times 3, 11 \times 4$	52
		$33 \times 3, 9 \times 4, 1 \times 5$	148
		$34 \times 3, 7 \times 4, 2 \times 5$	175
		$34 \times 3, 8 \times 4, 1 \times 6$	2
		$35 \times 3, 5 \times 4, 3 \times 5$	56
		$35 \times 3, 6 \times 4, 1 \times 5, 1 \times 6$	6
		$36 \times 3, 3 \times 4, 4 \times 5$	1
		$36 \times 3, 4 \times 4, 2 \times 5, 1 \times 6$	4
		$37 \times 3, 2 \times 4, 3 \times 5, 1 \times 6$	1
		$37 \times 3, 3 \times 4, 1 \times 5, 2 \times 6$	1
All	All	All	497

Table 5.3: Facts about $\mathcal{H}_4^4(43)$ and $\mathcal{H}_5^4(43)$

Theorem 5.10. *Any Type 1 Grinbergian hypohamiltonian graph is 4-face deflatable. More precisely, any Type 1 Grinbergian hypohamiltonian graph of order n is in $\mathcal{H}_0^4(n) \cup \mathcal{H}_1^4(n)$.*

Proof. Let G be a Type 1 Grinbergian hypohamiltonian graph with n vertices. By Theorem 5.3 the exceptional face belongs to $\mathcal{P}_1$ so its size is 4 or it is larger. If the exceptional face is a 4-face, then by Lemma 5.4 the 4-face has two non-adjacent 4-valent vertices. So we can apply $\mathcal{FD}_4$ to obtain a graph G' which has no face of size less than 5. So $\delta(G'^*) \geqslant 5$ and G'^* is a simple plane graph by Lemma 5.9. Thus $G'^* \in \mathcal{D}_5$ and as a result of the definition of $\mathcal{M}_f^4$, $G'^{**} = G' \in \mathcal{M}_0^4(n-1)$. Furthermore, $G \in \mathcal{M}_1^4(n)$ because $G \in \mathcal{PI}_2(G')$ and as G is hypohamiltonian, $G \in \mathcal{H}_1^4(n)$.

But if the exceptional face is not a 4-face, then by the fact that it is 3-connected and simple, G^* is simple as well and as the minimum face size of G is 5, $\delta(G^*) \geqslant 5$ which means $G \in \mathcal{M}_0^4(n)$ and so $G \in \mathcal{H}_0^4(n)$. □



(a) G

(b) G'

Figure 5.3: Showing that $\mathcal{FD}_4(G)$ has a simple dual

Corollary 5.11. *The smallest Type 1 Grinbergian hypohamiltonian graph has* 42 *vertices and there are exactly* 7 *of them on* 42 *vertices.*

Proof. By Theorem 5.10 any Type 1 Grinbergian graph belongs to $\mathcal{H}_0^4(n) \cup \mathcal{H}_1^4(n)$ but according to the results presented in the paragraph preceding Theorem 5.5, we have $\mathcal{H}_0^4(n) \cup \mathcal{H}_1^4(n) = \varnothing$ for all $n < 42$. So there is no such graph of order less than 42. On the other hand, we have $\mathcal{H}_0^4(42) = \varnothing$ and $|\mathcal{H}_1^4(42)| = 7$ which completes the proof. □

5.5 Results

We present one of the planar hypohamiltonian graphs of order 40, discovered by us in Figure 5.4, and the complete list of 25 in Figure 5.6.

Theorem 5.12. *The graph shown in Figure 5.4 is hypohamiltonian.*

Proof. We first show that the graph is non-Hamiltonian. Assume to the contrary that the graph contains a Hamiltonian cycle, which must then satisfy Grinberg's Theorem.

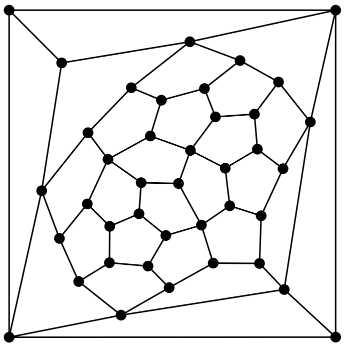


Figure 5.4: A planar hypohamiltonian graph on 40 vertices ($H_{40,1}$)

The graph in Figure 5.4 contains five 4-faces and 22 5-faces. Then

$$\sum_i (i-2)(f_i - f_i') \equiv f_4' - f_4 \equiv 0 \pmod 3,$$

where $f_4 + f_4' = 5$. So $f_4' = 1$ and $f_4 = 4$, or $f_4' = 4$ and $f_4 = 1$. Let Q be the 4-face on a different side from the four others.

Notice that an edge belongs to a Hamiltonian cycle if and only if the two faces it belongs to are on different sides of the cycle. Since the outer face of the embedding in Figure 5.4 has edges in common with all other 4-faces and its edges cannot all be in a Hamiltonian cycle, that face cannot be Q.

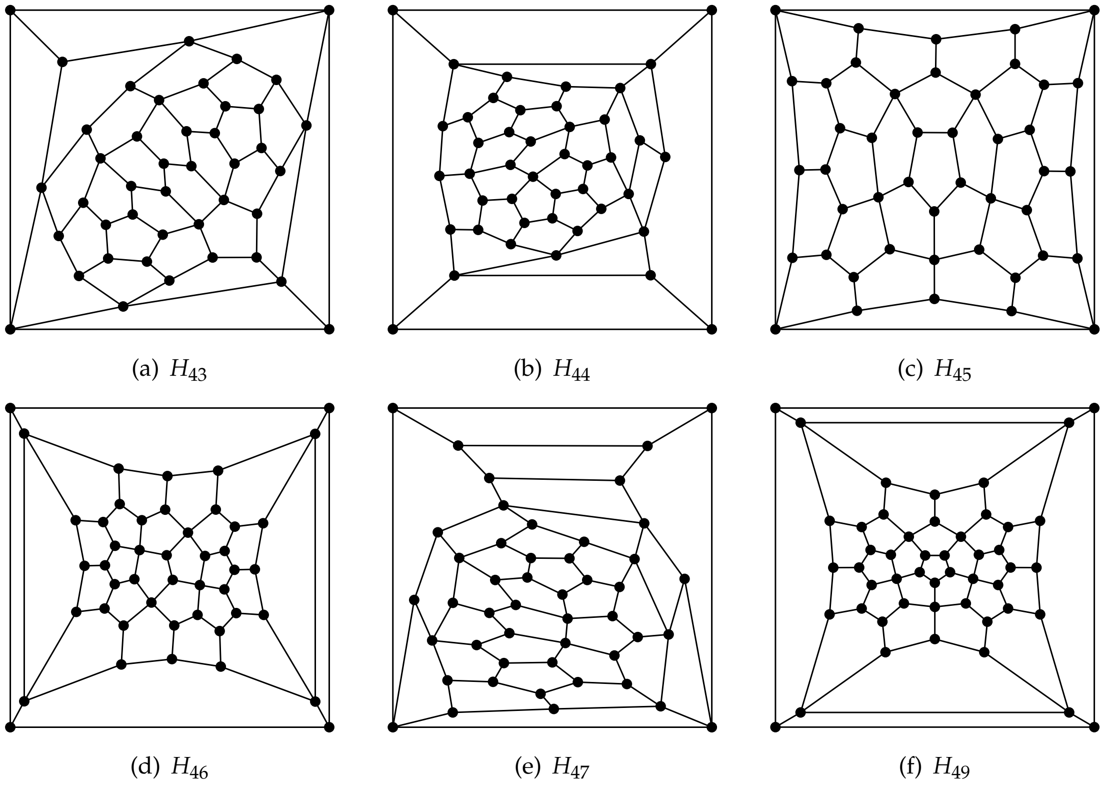
If Q is any of the other 4-faces, then the only edge of the outer face in the embedding in Figure 5.4 that belongs to a Hamiltonian cycle is the edge belonging to Q and the outer face. The two vertices of the outer face that are not endpoints of that edge have degrees 3 and 4, and we arrive at a contradiction as we know that two of the edges incident to the vertex with degree 3 are not part of the Hamiltonian cycle. Thus, the graph is non-Hamiltonian. Finally, for each vertex of the graph, Figure 5.8 shows a cycle omitting the vertex. □

We now employ operation Th_H (See Definition 1.14 and Figure 1.6), defined by Thomassen [117], for producing infinite sequences of hypohamiltonian graphs. Wiener and Araya use this operation to show that planar hypohamiltonian graphs exist for every order greater than or equal to 76. That result is improved further in Theorem 5.14.

Lemma 5.13. *The graphs H_{43}, H_{44}, H_{45}, H_{46}, H_{47} and H_{49} in Figure 5.5 are all hypohamiltonian.*

Proof. Using a computer search, it can easily be checked that the graphs H_{43} and H_{45} are not Hamiltonian. By Lemma 1.16 the rest of them are also non-Hamiltonian because $H_{44} \in \text{Th}_\text{H}(H_{40,1})$, $H_{46} \in \text{Th}_\text{H}(WA_{42})$, $H_{47} \in \text{Th}_\text{H}(H_{43})$ and $H_{49} \in \text{Th}_\text{H}(H_{45})$.

Figures 5.7, 5.9, 5.10, 5.11, 5.12 and 5.13 shows the vertex-omitting Hamiltonian cycles of them, respectively. □



(a) H_{43}

(b) H_{44}

(c) H_{45}

(d) H_{46}

(e) H_{47}

(f) H_{49}

Figure 5.5: Planar hypohamiltonian graphs of order 43, 44, 45, 46, 47 and 49

Theorem 5.14. *There exist planar hypohamiltonian graphs of order n for every $n \geqslant 42$.*

Proof. Lemma 5.13 shows the graph H_{43}, H_{44}, H_{45}, H_{46}, H_{47} and H_{49} are hypohamiltonian. Also each of H_{44}, H_{46}, H_{47} and H_{49} have a 4-cycle whose vertices are 3-valent so by repeated application of the operation Th_{H} (Definition 1.14) and Theorem 1.17 there is a planar hypohamiltonian graphs on $n \geqslant 44$. Adding the fact that H_{43} and WA_{42} are hypohamiltonian, the result is obtained. □

Whether there exists a planar hypohamiltonian graph on 41 vertices remains an open question.

Wiener and Araya [123] further prove that there exist planar hypotraceable graphs on $162 + 4k$ vertices for every $k \geqslant 0$, and on n vertices for every $n \geqslant 180$. To improve on that result, we make use of the following theorem, which is a slight modification of [114, Lemma 3.1].

Theorem 5.15. *There exist planar hypotraceable graphs on* 154 *vertices, and on n vertices for every $n \geqslant 156$.*

Proof. All the graphs obtained in the proof of Theorem 5.14 have a vertex with degree 3. Consequently, Theorem 1.19 can be applied to those graphs to obtain planar hypotraceable graphs of order n for $n = 40 + 40 + 40 + 40 - 6 = 154$ and for $n \geqslant 40 + 40 + 40 + 42 - 6 = 156$. □

The graphs considered in this work have girth 4. In fact, by the following theorem we know that any planar hypohamiltonian graphs improving on the results of the current work must have girth 3 or 4. Notice that a planar hypohamiltonian graph can have girth at most 5, since a planar hypohamiltonian graph has a simple dual, and the average degree of a simple plane graph is less than 6.

Let H be a cubic graph and G be a graph containing a cubic vertex $w \in V(G)$. We say that we *insert G into H*, if we replace every vertex of H with $G - w$ and connect the endpoints of edges in H to the neighbours of w.

Corollary 5.16. *We have*

$$\overline{C_3^1} \leqslant 40, \quad \overline{C_3^2} \leqslant 2625, \quad \overline{P_3^1} \leqslant 156 \quad \text{and} \quad \overline{P_3^2} \leqslant 10350.$$

Proof. The first of the four inequalities follows immediately from Theorem 5.12. In the following, let G be the planar hypohamiltonian graph from Figure 5.4.

For the second inequality, insert G into Thomassen's graph H from [117, p. 38]. This means that each vertex of H is replaced by G minus some vertex of degree 3. Since every pair of edges in H is missed by a longest cycle [101], in the resulting graph G' any pair of vertices is missed by a longest cycle. This property is not lost if all edges originally belonging to H are contracted.

In order to prove the third inequality, insert G into K_4. We obtain a graph in which every vertex is avoided by a path of maximal length.

For the last inequality, consider the graph H from the second paragraph of this proof and insert H into K_4, obtaining H'. Now insert G into H'. Finally, contract all edges which originally belonged to H'. □

5.6 Conclusions

Despite the new planar hypohamiltonian graphs discovered in the current work, there is still a wide gap between the order of the smallest known graphs and the best lower bound known for the order of the smallest such graphs, which is 18 [1]. One explanation for this gap is the fact that no extensive computer search has been carried out to increase the lower bound.

It is encouraging though that the order of the smallest known planar hypohamiltonian graph continues to decrease. It is very difficult to conjecture anything about the smallest possible order, and possible extremality of the graphs discovered here. It would be somewhat surprising though if no extremal graphs would have nontrivial automorphisms (indeed, the graphs of order 40 discovered in the current work have no nontrivial automorphisms). An exhaustive study of graphs with prescribed automorphisms might lead to the discovery of new, smaller graphs.

The smallest known *cubic* planar hypohamiltonian graph has 70 vertices [2]. We can hope that the current work inspires further progress in that problem too.

5.A List of All 4-Face Deflatable Hypohamiltonian Graphs on 40 Vertices

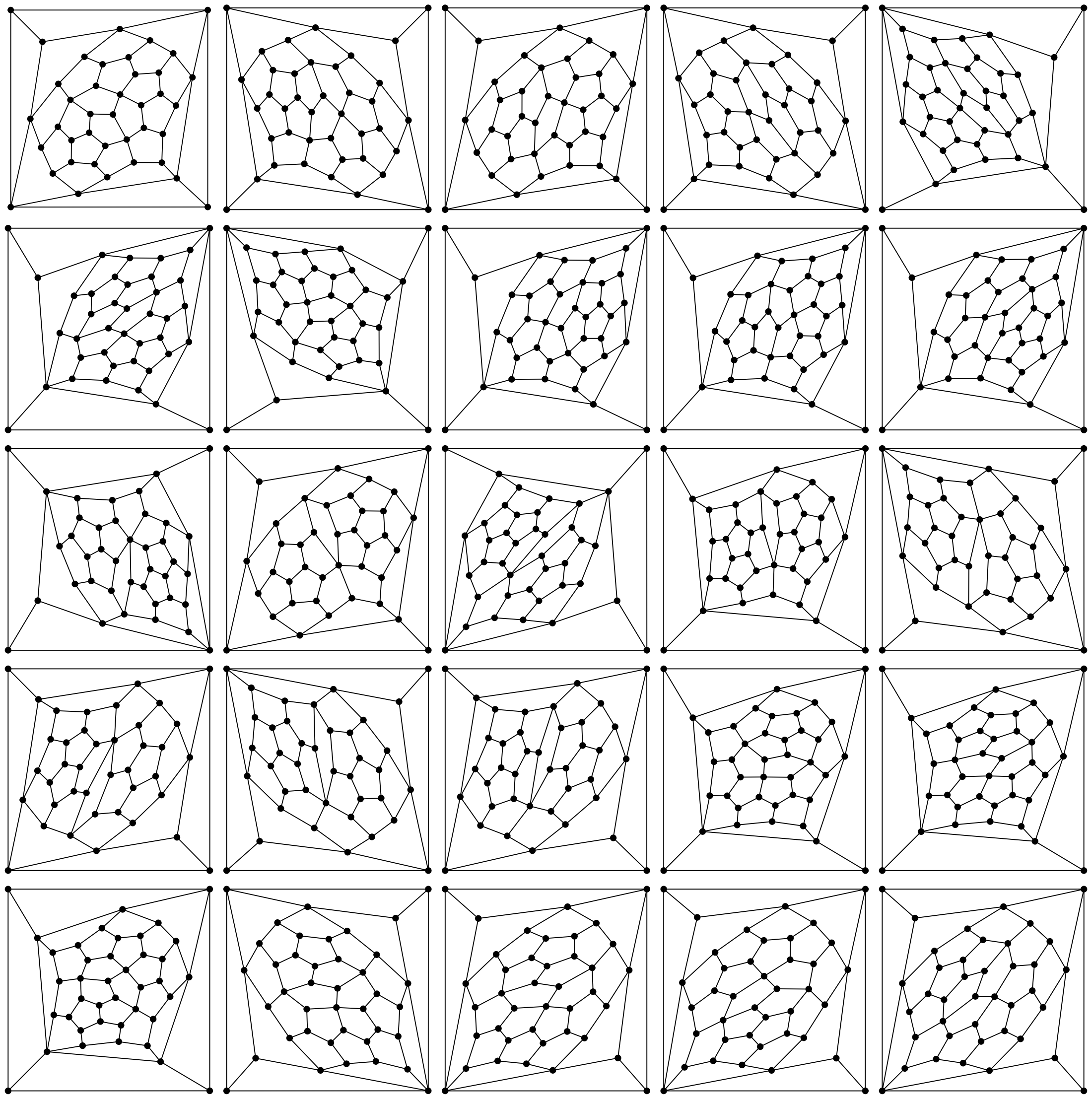


Figure 5.6: List of 4-face deflatable hypohamiltonian graphs on 40 vertices

5.B Vertex-Omitting cycles of $H_{40,1}$, H_{43}, H_{44}, H_{45}, H_{46}, H_{47} and H_{49}

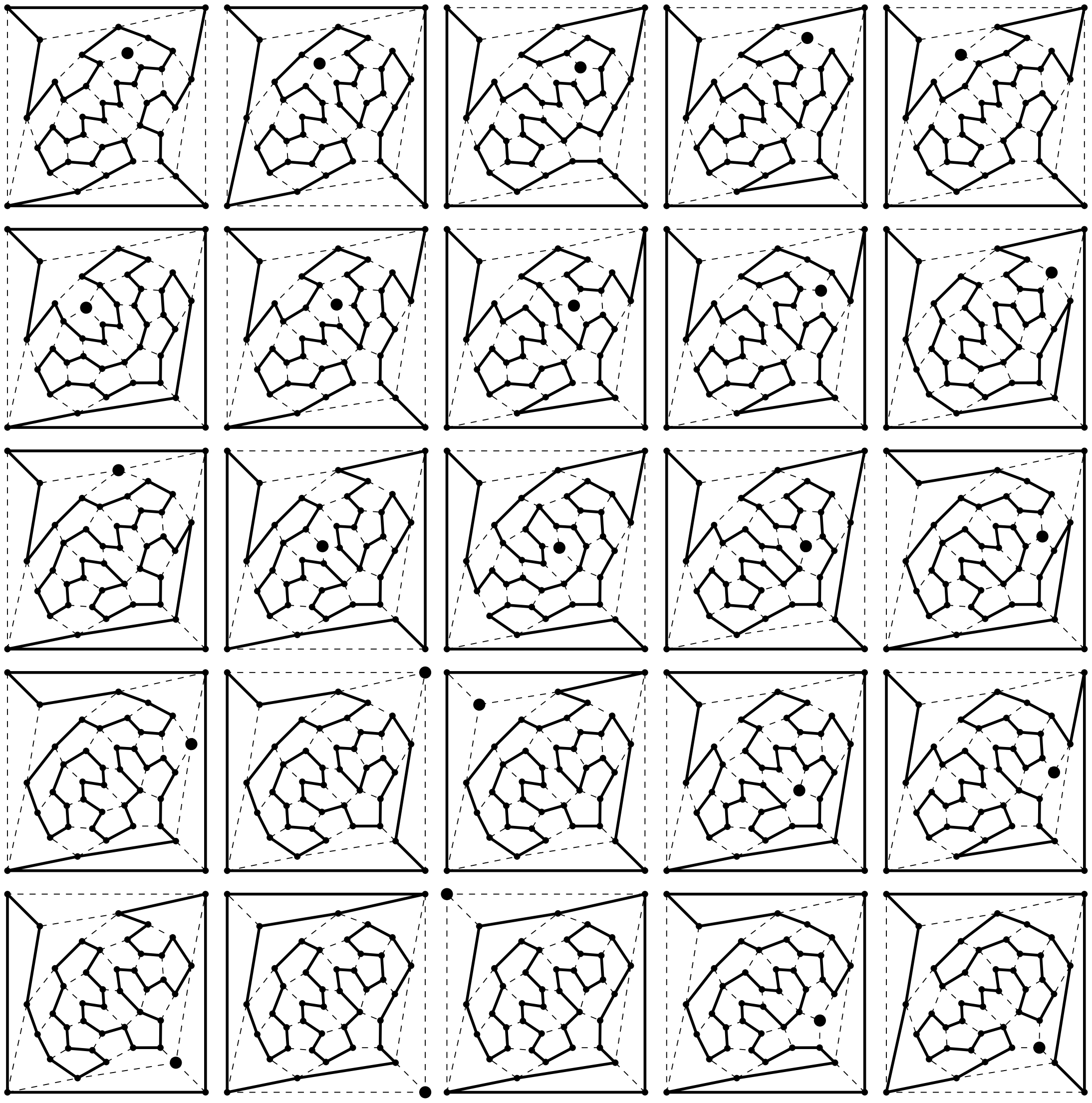


Figure 5.7: All vertex-omitting cycles of H_{43} up to automorphism

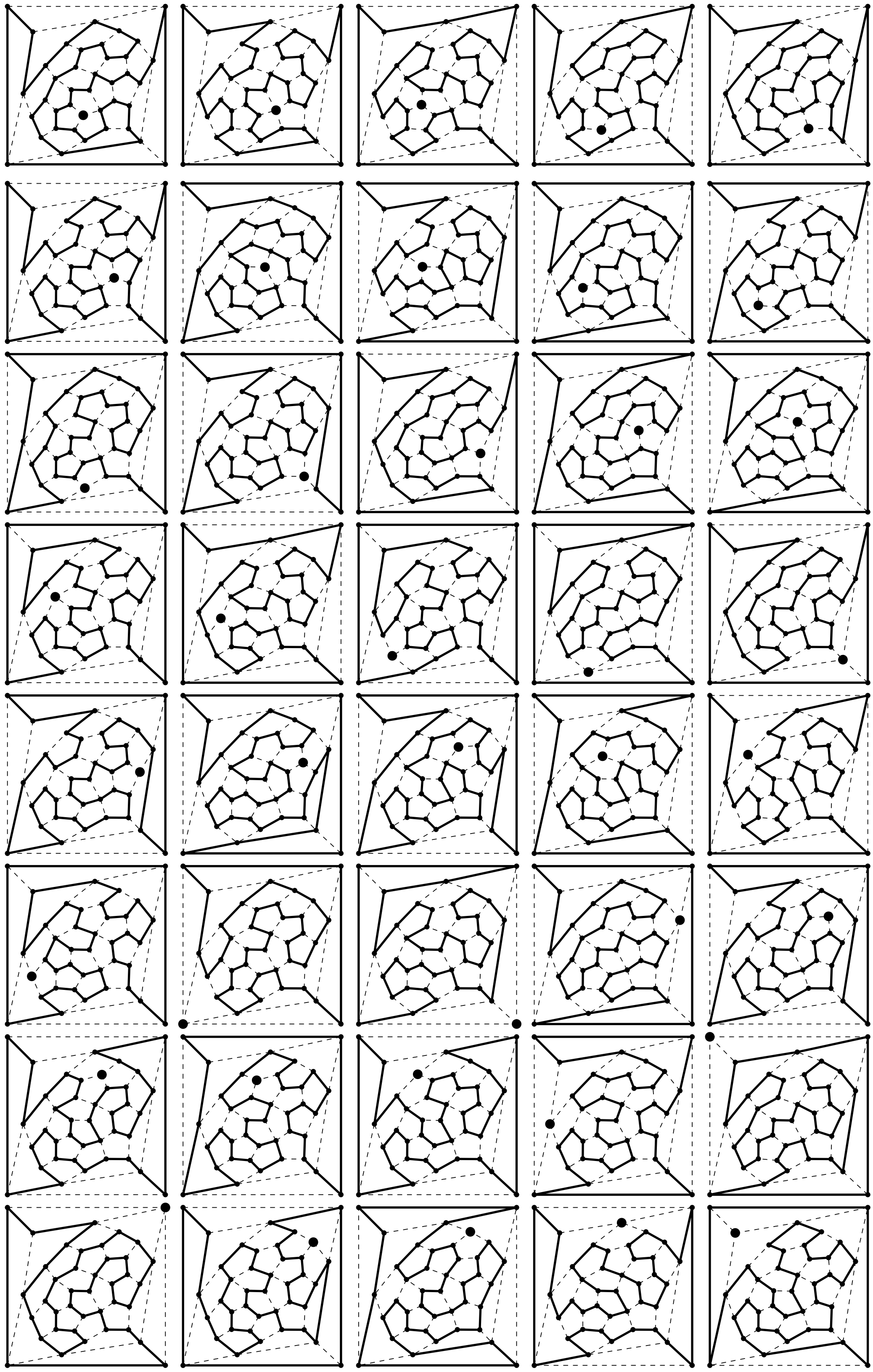


Figure 5.8: Vertex-omitting cycles of $H_{40,1}$

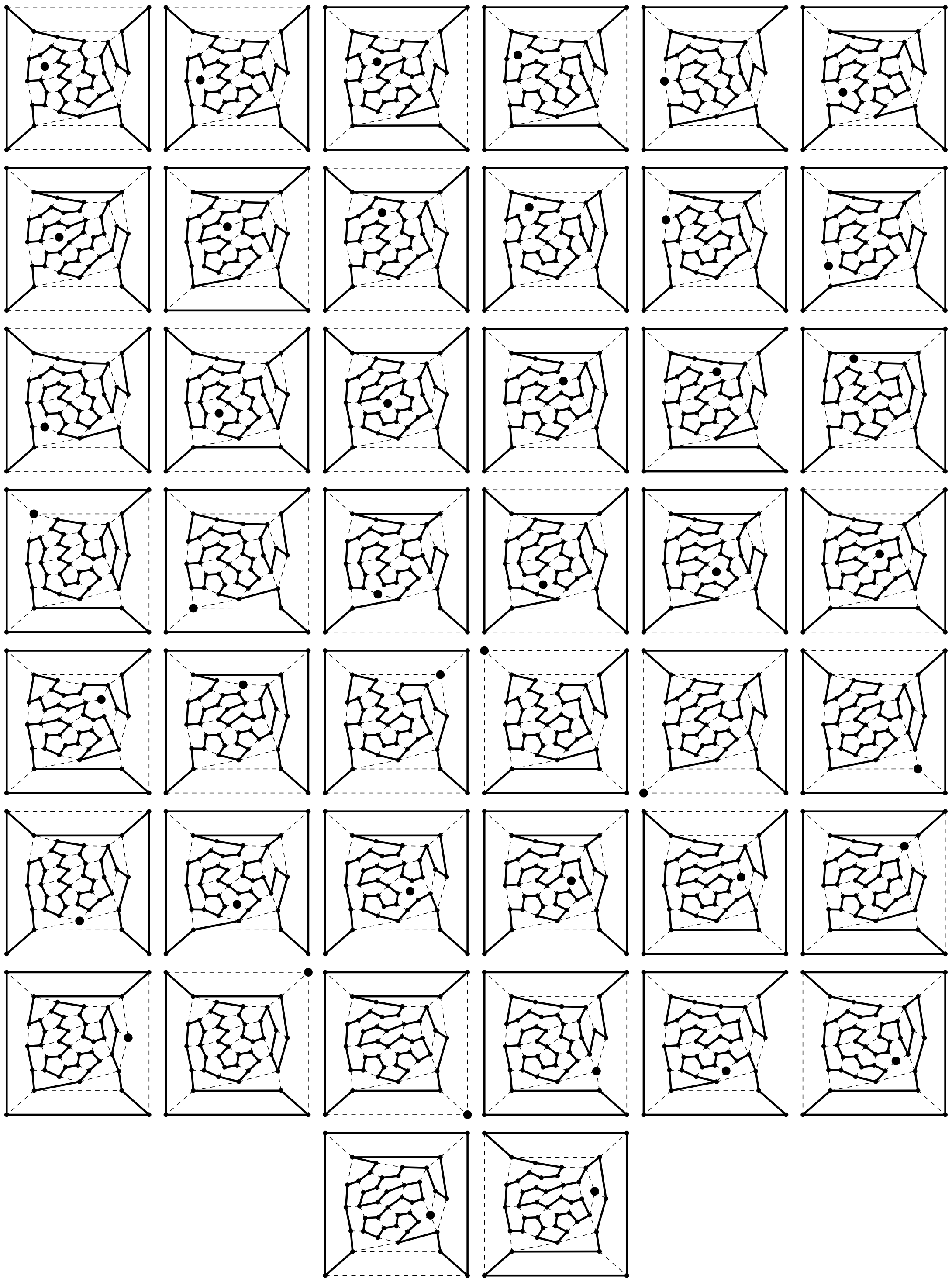


Figure 5.9: All vertex-omitting cycles of H_{44}

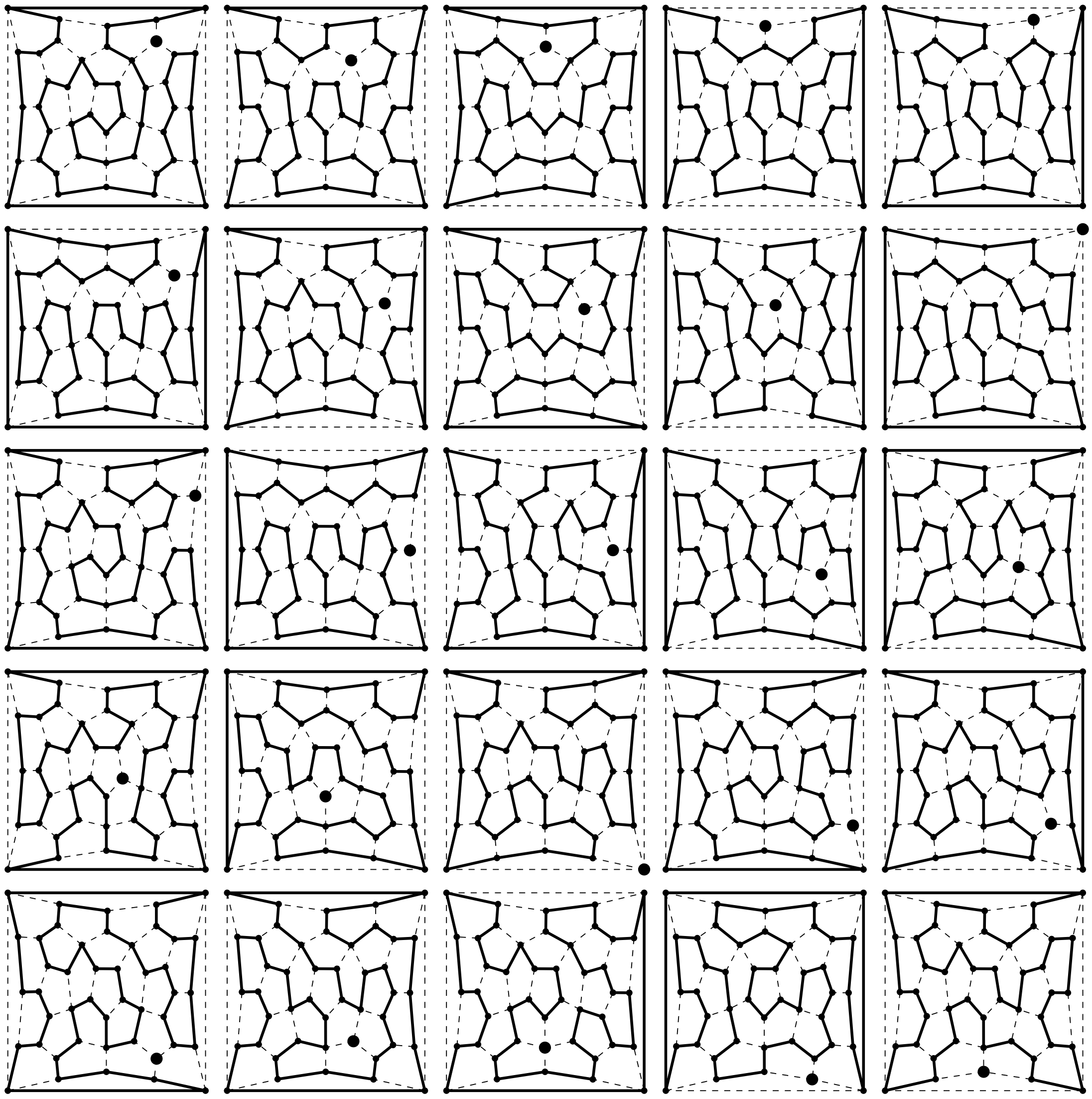


Figure 5.10: All vertex-omitting cycles of H_{45} up to automorphism

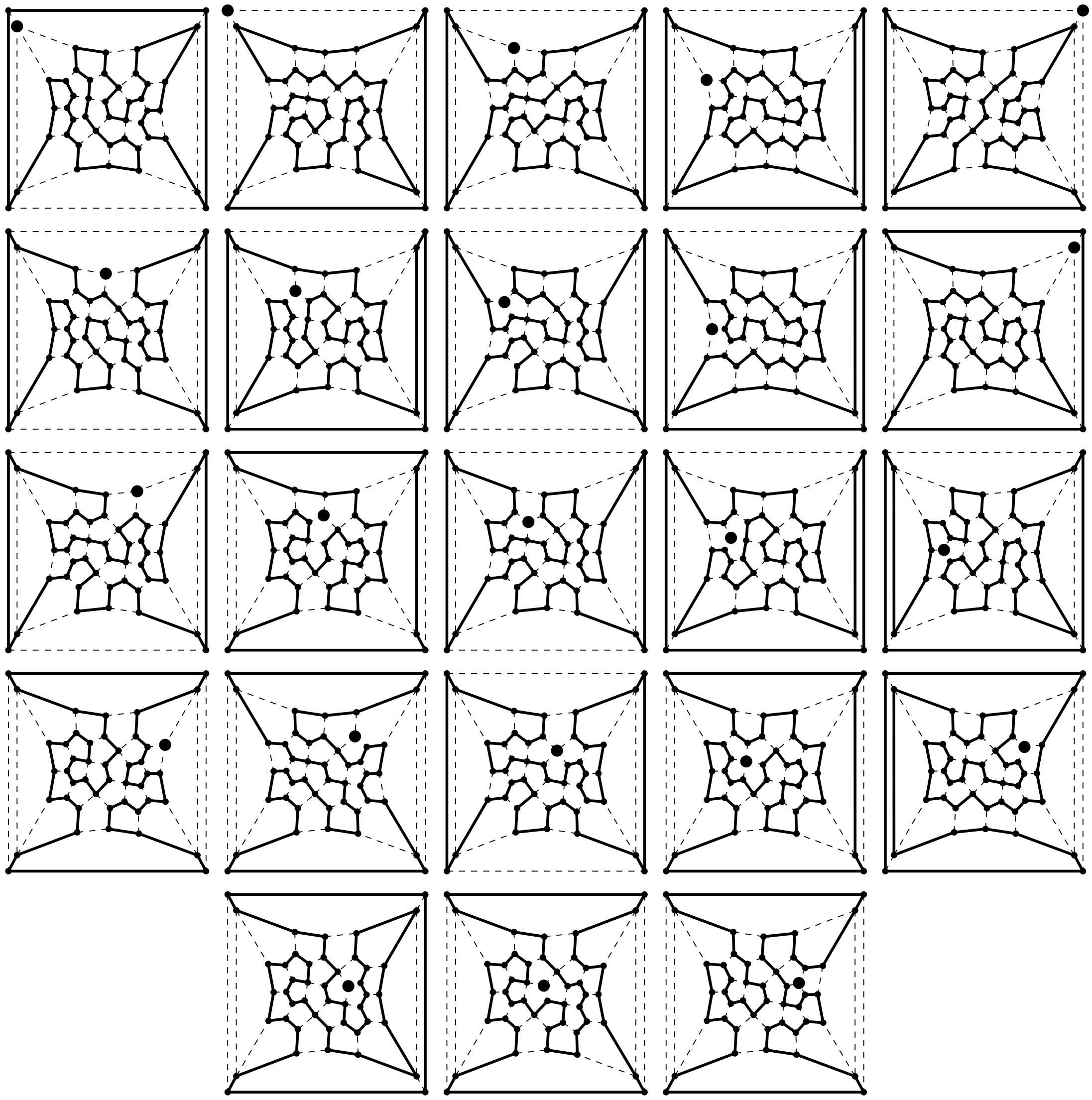


Figure 5.11: All vertex-omitting cycles of H_{46} up to automorphism

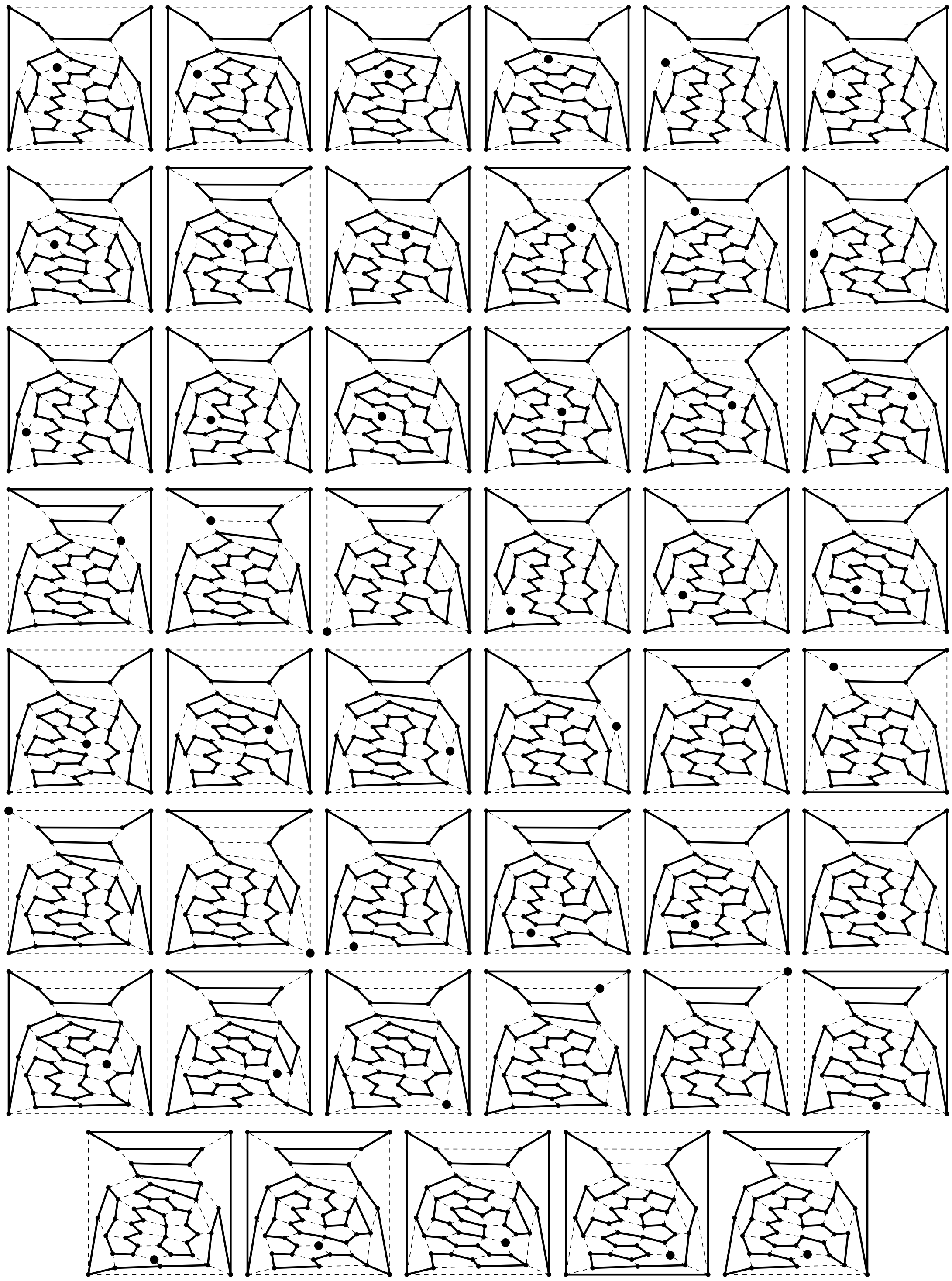


Figure 5.12: All vertex-omitting cycles of H_{47}

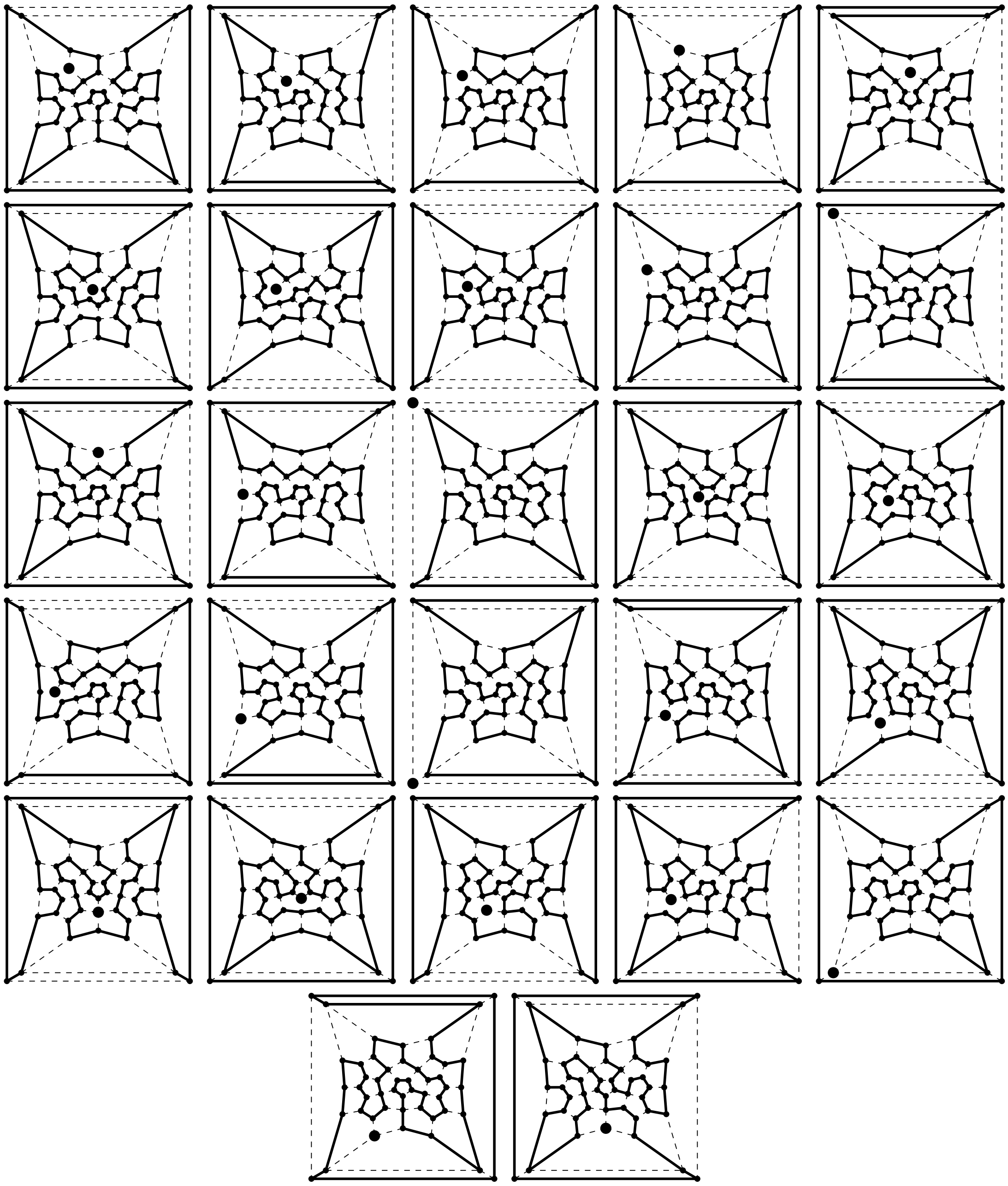


Figure 5.13: All vertex-omitting cycles of H_{49} up to automorphism

Chapter 6

Face-spiral Codes in 3-Connected Cubic Plane Graphs with no Large Face

6.1 Background

This chapter contains the material published in "Face-spiral codes in cubic polyhedron graphs with face sizes no larger than 6" which is a joint study with P. W. Fowler and G. Brinkmann [38].

6.2 Introduction

polyhedron

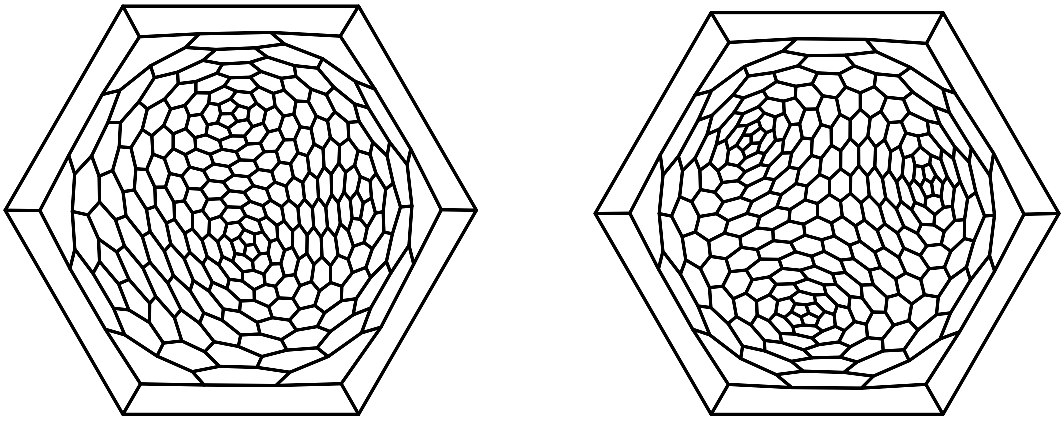
A *polyhedron* is a 3-connected planar graph. By Whitney's theorem each polyhedron has a unique embedding on the plane up to plane isomorphism [122]. So the polyherdons can be referred to as 3-connected plane graphs.

Cubic polyhedron structures built from carbon and other atoms are of current interest in chemistry, physics and materials science for many reasons. They are exemplified by the fullerenes [72, 70, 40, 41], and also occur as skeletons of the polyhedron hydrocarbons [33, 69, 32, 93] known collectively as 'spheroalkanes' [97]. They are studied as models for electron-precise clusters involving other elements (e.g., clusters with pairwise replacement of carbon atoms by BN) [105]. They occur as motifs in supramolecular frameworks [106], act as finite models for many of the forms of carbon that have emerged since the discovery of the *fullerenes* (cubic polyhedrons which have only 5-faces and 6-faces) [58, 119, 102, 78, 71, 59, 92] and for chemically plausible 'spheroarene' [96] generalisations of the fullerene class [39, 3, 42, 31, 68].

fullerene

face-spiral conjecture
face-spiral

The *face-spiral conjecture* for fullerenes claims that the surface of every fullerene can be unwound in a *face-spiral* of edge-sharing 5-faces (pentagons) and 6-faces (hexagons) such that each face in the spiral after the second, has an edge in common with the previous face and another with the first face in the proceeding faces which has an edge that has an open edge (the edge only belongs to one of the previous faces) with the condition that a face-spiral must contain each face exactly once [81, 40, 41]. The stronger version of this conjecture replaces "fullerences" with "cubic polhral"; and "5-faces" and "6-faces" with "faces". Both conjectures are known to be false and the smallest known fullerenes with no face-spirals have 380 (Figure 6.1(a)) [80] and 384 vertices (Figure 6.1(b)) [124]. Recently, it is proven that these two are the smallest fullerenes with no face-spirals and there is no counterexample upto 400 vertices [17, 47]. An example of failed spiral for the smallest fullerene counterexample is presented in Section 6.2.



(a) $|V| = 380$

(b) $|V| = 384$

Figure 6.1: The fullerenes with up to 400 vertices having no face-spirals

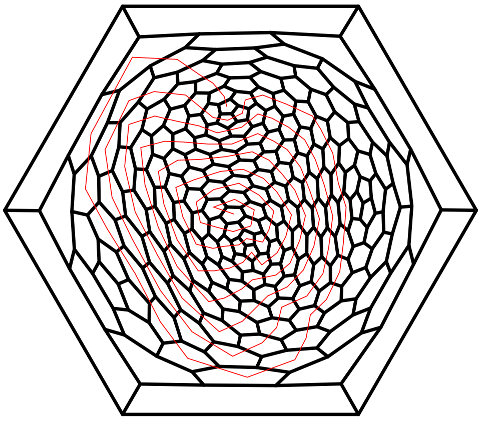


Figure 6.2: An example of failed spiral

Although it has been known for a while that not all fullerences have face-spirals, this idea has been used to design the first generator (non-exhaustive) for fullerences [40] which was improved later to an exhaustive generator in [14, 15].

6.3 Spirals and classes of cubic polyhedra

If the maximum face size is restricted to six, and graphs without multiple edges are considered (simple $\{3, 4, 5, 6\}$-angulations), simple counting with Euler's theorem face-signatures

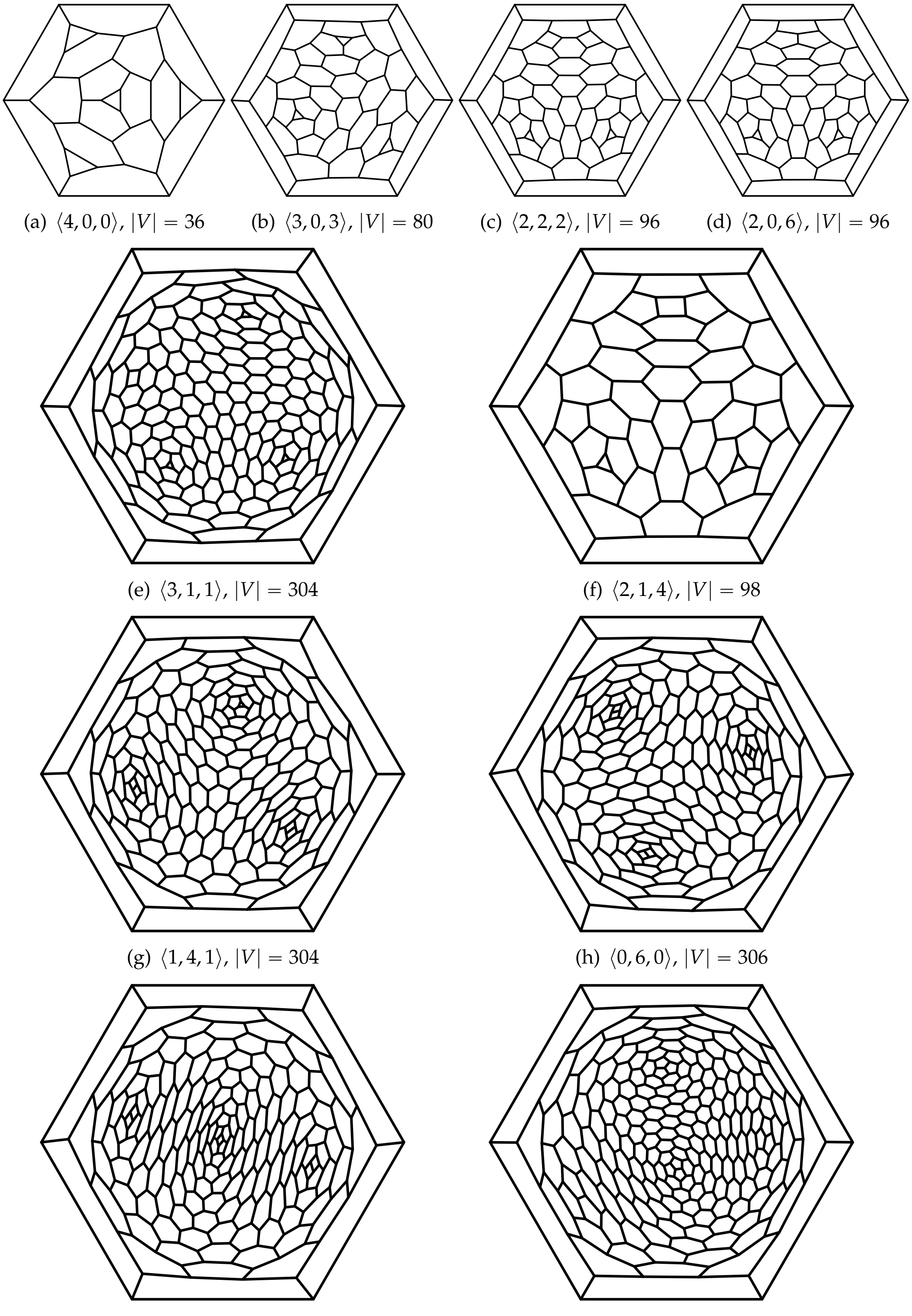
for polyhedra gives 19 distinct *face-signatures* $\langle f_3, f_4, f_5\rangle$, where f_3, f_4 and f_5 are the respective numbers of triangle, quadrangle and pentagonal faces.

In the following, we explore each face-signature class and attempt to provide a minimal unspirallable example, or at least to place bounds on the size of the smallest unspirallable polyhedron in the class. The initially plausible suggestion that fullerenes may be the 'best' cubic polyhedra for the spiral conjecture proves to be incorrect with respect to vertex numbers: while the smallest non-spiral fullerene has 380 vertices, the smallest non-spiral polyhedron in the class $\langle 2, 3, 0\rangle$ has as many as 2170 vertices. If, instead, we consider the number of structures in the class that are smaller than the minimal counterexample, fullerenes can, however, still be claimed to be best suited for spiral coding.

Two methods are used here for finding counterexamples. The first method is exhaustive generation of each family, at each vertex number, followed by a check of the results for spirals. The counterexamples resulting from this approach are presented in Table 6.1. We used the program *CGF* by Harmuth [19, 50] which can be obtained as part of the package *CaGe* [16]. The non-spiral examples were all tested independently by the two programs used elsewhere [17] to check fullerenes without spirals and for the case of fullerenes only, the generation step itself was also checked with an independent program.

Face Signature	Order	Figure
$\langle 4, 0, 0\rangle$	36	Figure 6.3(a)
$\langle 3, 1, 1\rangle$	304	Figure 6.3(e)
$\langle 3, 0, 3\rangle$	80	Figure 6.3(b)
$\langle 2, 3, 0\rangle$	2170	Figure 6.6
$\langle 2, 2, 2\rangle$	96	Figure 6.3(c)
$\langle 2, 1, 4\rangle$	98	Figure 6.3(f)
$\langle 2, 0, 6\rangle$	96	Figure 6.3(d)
$\langle 1, 4, 1\rangle$	304	Figure 6.3(g)
$\langle 0, 6, 0\rangle$	306	Figure 6.3(h)
$\langle 0, 5, 2\rangle$	304	Figure 6.3(i)
$\langle 0, 0, 12\rangle$	380	Figure 6.3(j)

Table 6.1: Minimal counterexamples for classes with no face-spirals found by exhaustive generation. In each case, there is a unique counterexample with the given vertex number within the class.



(a) $\langle 4,0,0\rangle$, $|V| = 36$ (b) $\langle 3,0,3\rangle$, $|V| = 80$ (c) $\langle 2,2,2\rangle$, $|V| = 96$ (d) $\langle 2,0,6\rangle$, $|V| = 96$

(e) $\langle 3,1,1\rangle$, $|V| = 304$ (f) $\langle 2,1,4\rangle$, $|V| = 98$

(g) $\langle 1,4,1\rangle$, $|V| = 304$ (h) $\langle 0,6,0\rangle$, $|V| = 306$

(i) $\langle 0,5,2\rangle$, $|V| = 304$ (j) $\langle 0,0,12\rangle$, $|V| = 380$

Figure 6.3: Minimal counterexamples for classes with no face-spirals

In cases where, within reasonable time limits, no counterexamples could be found by exhaustive generation, we modified existing counterexamples from other classes using operations in Figure 6.4. Using these four operations we were able to construct counterexamples for all remaining values of f_3, f_4 and f_5 and the result is presented in Table 6.2.

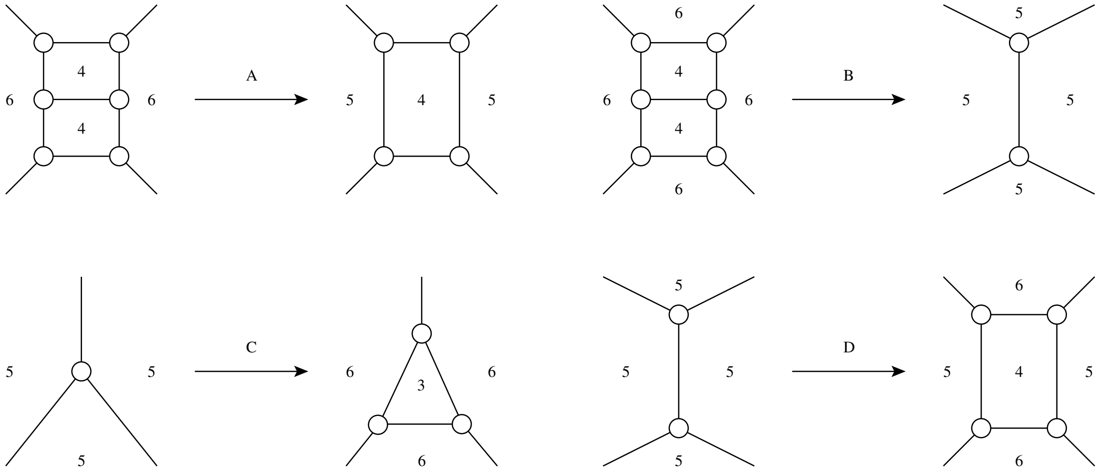


Figure 6.4: Operations used for converting graphs between face-signature classes

Counterexample		Parent		
Signature	Order	Signature	Order	Operation
$\langle 1,3,\ 3\rangle$	302	$\langle 1,4,\ 1\rangle$	$304^{\#}$	A
$\langle 1,2,\ 5\rangle$	300			A^2
$\langle 0,4,\ 4\rangle$	302	$\langle 0,5,\ 2\rangle$	$304^{\#}$	A
$\langle 0,3,\ 6\rangle$	300			A^2
$\langle 0,1,10\rangle$	386	$\langle 0,0,12\rangle$	$384^{\star}$	D
$\langle 0,2,\ 8\rangle$	388			D^2
$\langle 0,1,\ 9\rangle$	382	$\langle 0,0,12\rangle$	$380^{\#}$	C
$\langle 1,1,\ 7\rangle$	326	$\langle 1,3,\ 3\rangle$	$330^{\dagger}$	B

Table 6.2: Counterexamples for classes with no face-spirals generated by modification. The parents marked with # are unique minimal counterexamples within their own family. The parent marked with $\star$ is the unique second smallest counterexample within the fullerene family [124, 17]. The parent marked with † is a non-minimal counterexample for $\langle 1,3,3\rangle$ which is shown in Figure 6.5.

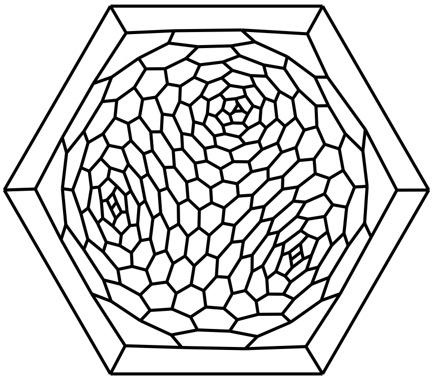


Figure 6.5: Parent of the counterexample for the class $\langle 1, 1, 7\rangle$

6.4 Conclusions

One obvious comment on the results presented here is that non-spiral cases are found reasonably early for all but one of the 19 classes, with one set of classes having non-spiral counterexamples of order 100 or less, and another having counterexamples in the range 300 to 400. The outstanding exception is the class $\langle 2, 3, 0\rangle$ which requires about five times as many vertices as the smallest fullerene counterexample. This example is an egregious exception (Figure 6.7). It is natural to wonder why it needs so many vertices.

Although this is perhaps not the most precisely defined of questions, we can at least note that all the other counterexamples have one of two rough shapes: either a characteristic roughly tetrahedral cluster of defects, or a trigonal-sandwich structure. The class $\langle 2, 3, 0\rangle$ does not allow either of these groupings. For polyhedra in this class, the total defect of 12 is made up of contributions 3, 3, 2, 2, 2. A triangular shape would require distribution of these defects in 3 groups of defect 4 each (not possible). Similarly, a tetrahedral shape would require 4 groups of defect 4 each (again not possible) The eventual first counterexample in $\langle 2, 3, 0\rangle$ includes four groups of defect 3, 2, 3 and 4, respectively, and spring embedding [16] suggests a starfish-like shape, with four arms (Figure 6.8).

For cubic graphs that also allow faces of size larger than 6, counterexamples occur early, and are abundant [13]. These results suggest the conjecture that every infinite class of cubic polyhedra described by allowed and forbidden face sizes contains non-spiral elements.

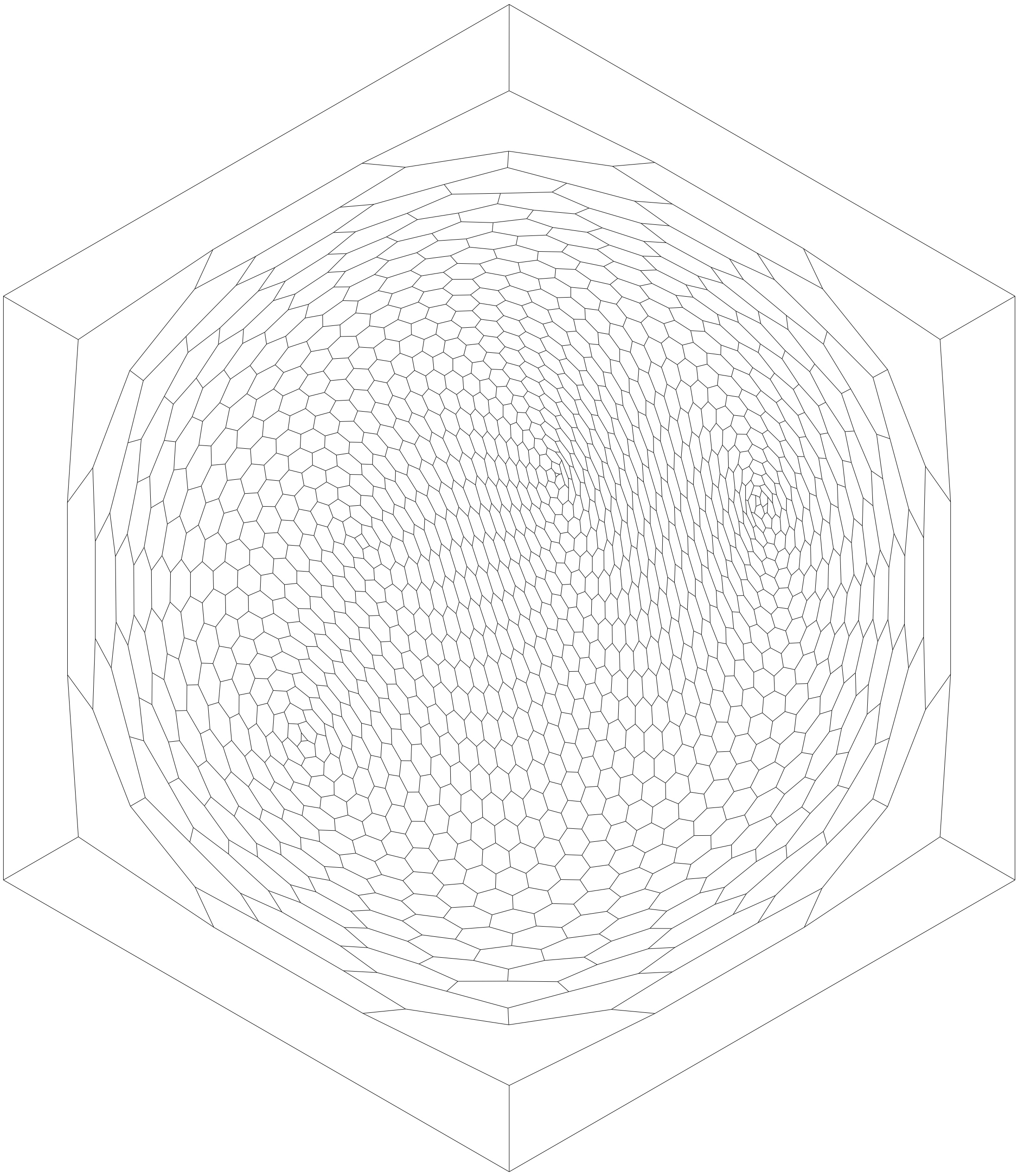


Figure 6.6: Minimal counterexample for sequence $\langle 2, 3, 0\rangle$ with 2170 vertices

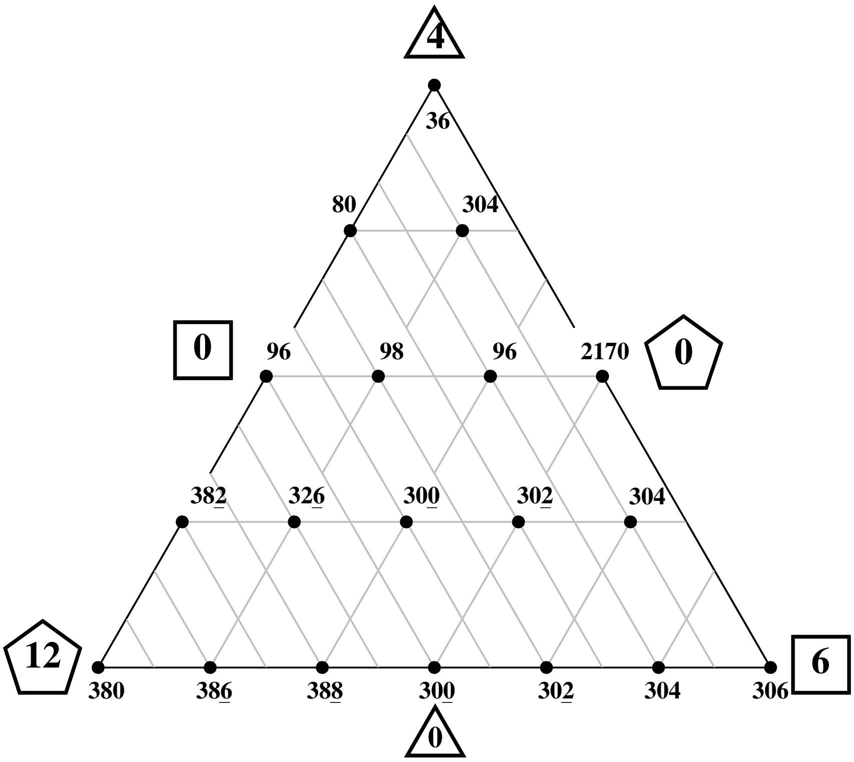


Figure 6.7: The landscape of spiral counterexamples. In the triangular coordinate system, the vertices of the master triangle represent 'pure' types $\langle 4, 0, 0\rangle$, $\langle 0, 6, 0\rangle$ and $\langle 0, 0, 12\rangle$, and in general the values p_3, p_4, and p_5 are proportional to the lengths of perpendiculars to the triangle sides. Each black dot represents a counterexample with the number of vertices indicated; minimal counterexamples are labelled in bold face; those numbers marked with an underline are not claimed to be minimal.

6.A A 3D embedding of the minimal counterexample from the class $\langle 2, 3, 0 \rangle$

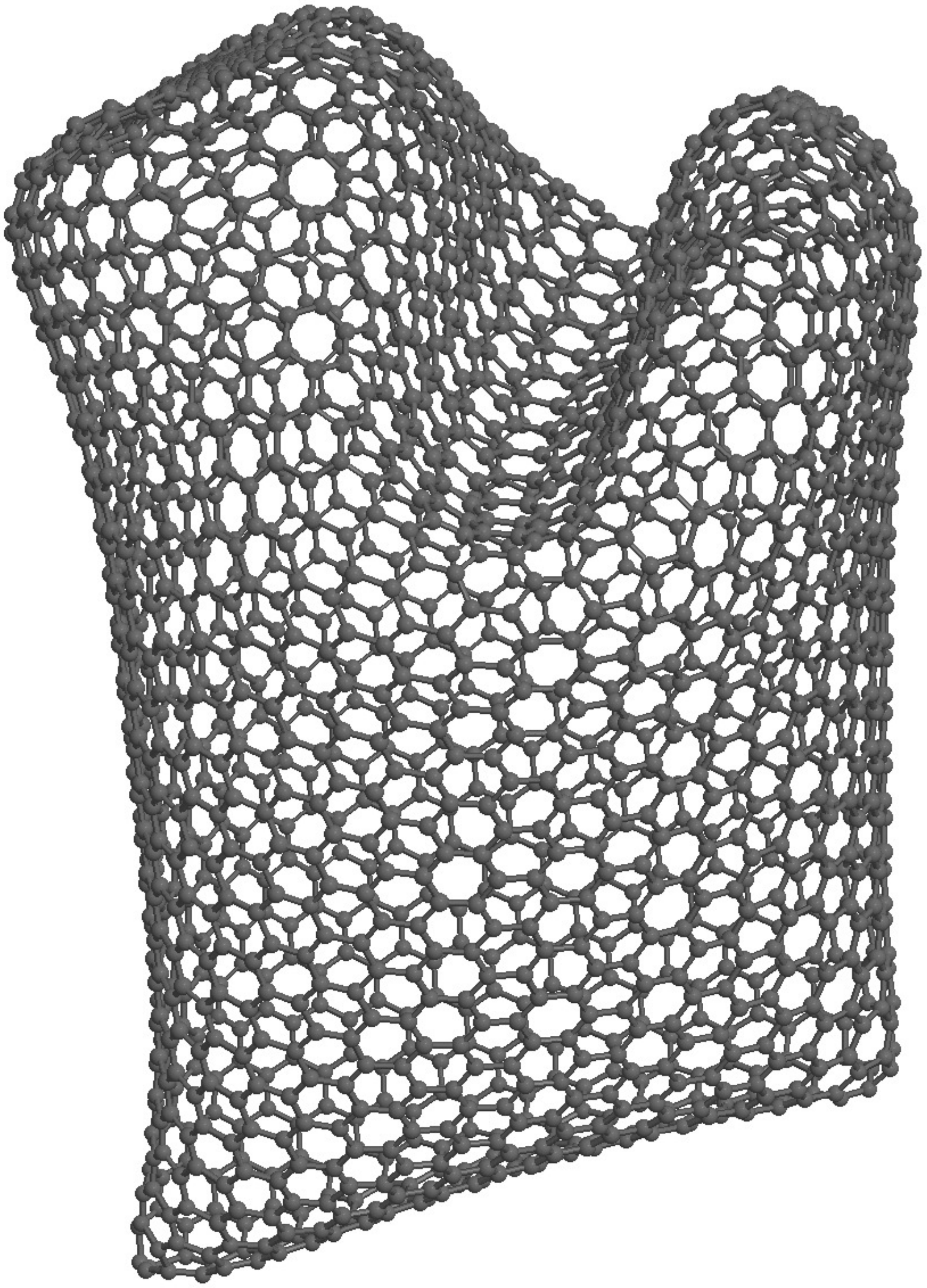


Figure 6.8: A 3D embedding of the minimal counterexample from the class $\langle 2, 3, 0 \rangle$

Conclusion

In this thesis we discussed recursive algorithms for generation of three families of graphs, namely: k-angulations, $\{k_1, k_2, \cdots, k_t\}$-angulations and 4-face deflatable hypohamiltonian graphs. The last generator allowed us to find the smallest known planar hypohamiltonian graphs with 40 vertices while the previous smallest known has 42 vertices. Then we designed a method for isomorphism checking and canonical labelling of 2-connected plane graphs and finally we analysed the face-spiral conjecture for fullerenes on a larger family of plane graphs and showed that it does not hold.

In the second chapter we discussed how k-angulations can be generated recursively from triangulations or quadrangulations. Then we optimised the generator using a careful definition of canonical code for the graphs used in the generation tree in addition to looking ahead and discovering the children which are not going to be accepted and pruning the generation tree.

We defined the recursive generation such that intermediate graphs are not required to belong to the target family (in this chapter k-angulations). This approach allowed us to start from triangulations or quadrangulations, but this extra flexibility impacts the performance. So to improve this result one could think of another approach which starts with a set of irreducible k-angulations and define the expansions such the intermediate graphs be k-angulations too. Such a generator is quite likely to be more efficient as potentially it could have very few intermediate graphs in comparison.

A natural extension of this study is to generate not only k-angulations which have only faces of size k, but also $\{k_1, k_2, \cdots, k_t\}$-angulations which include plane graphs with all face sizes in $\{k_1, k_2, \cdots, k_t\}$. This extension is discussed in detail in the fourth chapter where we showed how simple plane graphs with specified face sizes can be generated recursively from triangulations or quadrangulations. Then we optimised the generator using a careful definition of canonical code for the graphs used in the generation tree in addition to looking ahead and discovering the children which are not going to be accepted and pruning the generation tree.

In the sixth chapter we tackled the problem of finding the smallest hypohamiltonian graph and reduced the previous record from graphs with 42 vertices to 40 vertices. Despite the new planar hypohamiltonian graphs discovered, there is still a wide gap between the order of the smallest known graphs and the best lower bound known for the order of the smallest such graphs, which is 18 [1]. One explanation for this gap is the fact that no extensive computer search has been carried out to increase the lower bound.

It is encouraging though that the order of the smallest known planar hypohamiltonian graph continues to decrease. It is very difficult to conjecture anything about the smallest possible order, and possible extremality of the graphs discovered here. It would be somewhat surprising though if no extremal graphs would have nontrivial automorphisms (indeed, the graphs of order 40 discovered in the current work have no nontrivial automorphisms). An exhaustive study of graphs with prescribed automorphisms might lead to the discovery of new, smaller graphs. The smallest known *cubic* planar hypohamiltonian graph has 70 vertices [2]. We can hope that the current work inspires further progress in that problem too.

In chapter seven we work on the face-spiral conjecture and consider not only fullerenes which have only faces of size 5 and 6, but also plane graphs with 3-faces and 4-faces and realized that the conjecture does not hold for any of these families. One obvious comment on the results presented here is that non-spiral cases are found reasonably early for all but one of the 19 classes, with one set of classes having non-spiral counterexamples of order 100 or less, and another having counterexamples in the range 300 to 400. The outstanding exception is the class $\langle 2, 3, 0 \rangle$ which requires about five times as many vertices as the smallest fullerene counterexample. This example is an egregious exception (Figure 6.7). It is natural to wonder why it needs so many vertices.

For cubic graphs that also allow faces of size larger than 6, counterexamples occur early, and are abundant [13]. These results suggest the conjecture that every infinite class of cubic polyhedra described by allowed and forbidden face sizes contains non-spiral elements.

We also hope the recursive generations discussed in this thesis will inspire other induction proofs or other generator to help solving or finding counterexamples for different problems in mathematics, computer science and chemistry.

Bibliography

1. Aldred, R. E. L.; McKay, B. D.; and Wormald, N. C., Small hypohamiltonian graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, **23**, (1997), 143–152. (cited on pages 13, 84, and 103)

2. Araya, M. and Wiener, G., On cubic planar hypohamiltonian and hypotraceable graphs. *The Electronic Journal of Combinatorics*, **18(1)**, (2011), 85. (cited on pages 84 and 104)

3. Ayuela, A.; Fowler, P. W.; Mitchell, D.; Schmidt, R.; Seifert, G.; and Zerbetto, F., C62: Theoretical evidence for a nonclassical fullerene with a heptagonal ring. *The Journal of Physical Chemistry*, **100(39)**, (1996), 15634–15636. doi:10.1021/jp961306o. (cited on page 93)

4. Barnette, D., 1969. On steinitz's theorem concerning convex 3-polytopes and on some properties of planar graphs. In *The Many Facets of Graph Theory* (Eds. G. Chartrand and S. Kapoor), vol. 110 of *Lecture Notes in Mathematics*, 27–40. Springer Berlin / Heidelberg. ISBN 978-3-540-04629-5. doi:10.1007/BFb0060102. (cited on page 21)

5. Barnette, D., On generating planar graphs. *Discrete Mathematics*, **7(3-4)**, (1974), 199–208. doi:10.1016/0012-365X(74)90035-1. (cited on pages 19 and 21)

6. Batagelj, V., 1984. An inductive definition of the class of all triangulations with no vertex of degree smaller than 5. In *Graph theory: Proceedings of the Fourth Yugoslav Seminar on Graph Theory, Novi Sad, April 15-16 1983*, 15–25. (cited on pages 19 and 21)

7. Batagelj, V., Inductive definition of two restricted classes of triangulations. *Discrete Mathematics*, **52(2-3)**, (1984), 113–121. doi:10.1016/0012-365X(84)90074-8. (cited on pages 19 and 21)

8. Batagelj, V., 1989. An improved inductive definition of two restricted classes of triangulations of the plane. In *Combinatorics and Graph Theory*, vol. 25 of *Banach Center publications*, 11–18. PWN-Polish Scientific Publishers. ISBN 9788301093006. (cited on pages 19 and 21)

9. Battista, G. and Tamassia, R., On-line maintenance of triconnected components with spqr-trees. *Algorithmica*, **15(4)**, (1996), 302–318. doi:10.1007/BF01961541. (cited on page 45)

10. Bondy, J. A., Variations on the hamiltonian theme. *Canadian Mathematical Bulletin*, **15**, (1972), 57–62. doi:10.4153/CMB-1972-012-3. (cited on page 16)

11. Bowen, R. and Fisk, S., Generation of triangulations of the sphere. *Mathematics of Computation*, **21(98)**, (1967), 250–252. (cited on pages 19 and 21)

12. Brinkmann, G., Fast generation of cubic graphs. *Journal of Graph Theory*, **23(2)**, (1996), 139–149. doi:10.1002/(SICI)1097-0118(199610)23:2<139::AID-JGT5>3.0.CO;2-U. (cited on page 10)

13. Brinkmann, G., Problems and scope of spiral algorithms and spiral codes for polyhedral cages. *Chemical Physics Letters*, **272(3-4)**, (1997), 193–198. doi:10.1016/S0009-2614(97)88009-8. (cited on pages 98 and 104)

14. Brinkmann, G. and Dress, A. W., A constructive enumeration of fullerenes. *Journal of Algorithms*, **23(2)**, (1997), 345–358. doi:10.1006/jagm.1996.0806. (cited on page 94)

15. Brinkmann, G. and Dress, A. W., Penthex puzzles: A reliable and efficient top-down approach to fullerene-structure enumeration. *Advances in Applied Mathematics*, **21(3)**, (1998), 473 – 480. doi:10.1006/aama.1998.0608. (cited on page 94)

16. Brinkmann, G.; Friedrichs, O. D.; Lisken, S.; Peeters, A.; and Van Cleemput, N., Cage - a virtual environment for studying some special classes of plane graphs - an update. *MATCH Commun. Math. Comput. Chem.*, **63(3)**, (2010), 533–552. (cited on pages 95 and 98)

17. Brinkmann, G.; Goedgebeur, J.; and McKay, B. D., The smallest fullerene without a spiral. *Chemical Physics Letters*, **522(0)**, (2012), 54–55. doi:10.1016/j.cplett.2011.11.056. (cited on pages xv, 1, 20, 94, 95, and 97)

18. Brinkmann, G.; Greenberg, S.; Greenhill, C.; McKay, B. D.; Thomas, R.; and Wollan, P., Generation of simple quadrangulations of the sphere. *Discrete Mathematics*, **305(1-3)**, (2005), 33–54. doi:10.1016/j.disc.2005.10.005. (cited on pages 9, 19, and 21)

19. Brinkmann, G.; Harmuth, T.; and Heidemeier, O., The construction of cubic and quartic planar maps with prescribed face degrees. *Discrete Applied Mathematics*, **128(2-3)**, (2003), 541–554. doi:10.1016/S0166-218X(02)00549-8. (cited on page 95)

20. Brinkmann, G. and McKay, B. D., plantri software. http://cs.anu.edu.au/~bdm/plantri/. Accessed: 2013-09-03. (cited on pages 19, 21, 36, 43, 44, and 70)

21. Brinkmann, G. and McKay, B. D., Fast generation of some classes of planar graphs. *Electronic Notes in Discrete Mathematics*, **3**, (1999), 28–31. doi:10.1016/S1571-0653(05)80016-2. 6th Twente Workshop on Graphs and Combinatorial Optimization. (cited on pages 9, 19, and 21)

22. Brinkmann, G. and McKay, B. D., Construction of planar triangulations with minimum degree 5. *Discrete Mathematics*, **301(2-3)**, (2005), 147–163. doi:10.1016/j.disc.2005.06.019. (cited on pages 9, 19, 21, and 77)

23. Brinkmann, G. and McKay, B. D., Fast generation of planar graphs. *MATCH Commun. Math. Comput. Chem.*, **58**, (2007), 323–357. Expanded version available at http://cs.anu.edu.au/~bdm/papers/plantri-full.pdf. (cited on pages xiii, 9, 21, 31, 44, 70, and 72)

24. Brinkmann, G.; McKay, B. D.; and von Nathusius, U., Backtrack search and look-ahead for the construction of planar cubic graphs with restricted face sizes. *MATCH Commun. Math. Comput. Chem.*, **48**, (2003), 163–177. (cited on pages 9 and 21)

25. Butler, J. W., A generation procedure for the simple 3-polytopes with cyclically 5-connected graphs. *Canadian Journal of Mathematics*, **26**, (1974), 686–708. doi:10.4153/CJM-1974-065-6. (cited on pages 19 and 21)

26. Chvátal, V., Flip-flops in hypohamiltonian graphs. *Canadian Mathematical Bulletin*, **16**, (1973), 33–41. (cited on pages 16, 19, and 73)

27. Collier, J. B. and Schmeichel, E. F., Systematic searches for hypohamiltonian graphs. *Networks*, **8(3)**, (1978), 193–200. doi:10.1002/net.3230080303. (cited on pages 13 and 16)

28. Darga, P. T.; Katebi, H.; Liffiton, M.; Markov, I. L.; and Sakallah, K., saucy software. http://vlsicad.eecs.umich.edu/BK/SAUCY/. Accessed: 2013-09-06. (cited on page 45)

29. Ding, G.; Kanno, J.; and Su, J., Generating 5-regular planar graphs. *Journal of Graph Theory*, **61(3)**, (2009), 219–240. doi:10.1002/jgt.20377. (cited on page 21)

30. Doyen, J. and Diest, V. V., New families of hypohamiltonian graphs. *Discrete Mathematics*, **13(3)**, (1975), 225–236. doi:10.1016/0012-365X(75)90020-5. (cited on page 16)

31. Dress, A. and Brinkmann, G., Phantasmagorical fulleroids. *MATCH Commun. Math. Comput. Chem*, **33**, (1996), 87–100. (cited on page 93)

32. Eaton, P. E.; Cassar, L.; and Halpern, J., Silver(i)- and palladium(ii)-catalyzed isomerizations of cubane. synthesis and characterization of cuneane. *Journal of the American Chemical Society*, **92(21)**, (1970), 6366–6368. doi:10.1021/ja00724a061. (cited on page 93)

33. Eaton, P. E. and Cole, T. W., Cubane. *Journal of the American Chemical Society*, **86(15)**, (1964), 3157–3158. doi:10.1021/ja01069a041. (cited on page 93)

34. Eberhard, V., *Zur Morphologie der Polyeder*. B.G. Teubner, 1891. (in German). (cited on page 21)

35. Eppstein, D., Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, **3(3)**, (1999), 1–27. doi:10.7155/jgaa.00014. (cited on page 45)

36. Faradzev, I. A., 1978. Constructive enumeration of combinatorial objects. In *Problèmes combinatoires et théorie des graphes (Colloq. Internat. CNRS, Univ. Orsay, Orsay, 1976)*, vol. 260 of *Colloq. Internat. CNRS*, 131âĂŞ135. CNRS, Paris. (cited on page 10)

37. Faradzhev, I. A., 1978. Generation of nonisomorphic graphs with a given degree sequence. In *Algorithmic Studies in Combinatorics Moscow*, 11–19. Nauka, Moscow, (in Russian). (cited on page 10)

38. Fowler, P.; Jooyandeh, M.; and Brinkmann, G., Face-spiral codes in cubic polyhedral graphs with face sizes no larger than 6. *Journal of Mathematical Chemistry*, **50**, (2012), 2272–2280. doi:10.1007/s10910-012-0029-3. (cited on page 93)

39. Fowler, P. W.; Heine, T.; Manolopoulos, D. E.; Mitchell, D.; Orlandi, G.; Schmidt, R.; Seifert, G.; and Zerbetto, F., Energetics of fullerenes with four-membered rings. *The Journal of Physical Chemistry*, **100(17)**, (1996), 6984–6991. doi:10.1021/jp9532226. (cited on page 93)

40. Fowler, P. W. and Manolopoulos, D. E., *An atlas of fullerenes*. Oxford University Press, Oxford, 1995. (cited on pages 93 and 94)

41. Fowler, P. W. and Manolopoulos, D. E., *An atlas of fullerenes*. Dover Publications Inc., New York, 2006. (cited on page 93)

42. Fowler, P. W.; Mitchell, D.; Seifert, G.; and Zerbetto, F., Energetics of fullerenes with octagonal rings. *Fullerene Science and Technology*, **5(4)**, (1997), 747–768. doi:10.1080/15363839708012229. (cited on page 93)

43. Gagarin, A.; Labelle, G.; Leroux, P.; and Walsh, T., Structure and enumeration of two-connected graphs with prescribed three-connected components. *Advances in Applied Mathematics*, **43(1)**, (2009), 46–74. doi:10.1016/j.aam.2009.01.002. (cited on page 44)

44. Garey, M. R. and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences. W. H. Freeman & Co., San Francisco, California. ISBN 0716710455, 1979. (cited on page 13)

45. Gaudin, T.; Herz, J. C.; and Rossi, P., Solution du problème no. 29. *Rev. Franç. Rech. Opérationnelle*, **8**, (1964), 214–218. (cited on page 13)

46. Gazit, H. and Reif, J., 1990. A randomized parallel algorithm for planar graph isomorphism. In *Proceedings of the Second Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '90 (Island of Crete, Greece, 1990), 210–219. ACM, New York, NY, USA. doi:10.1145/97444.97687. (cited on page 45)

47. Goedgebeur, J., *Generation Algorithms for Mathematical and Chemical Problems*. Ph.D. thesis, Department of Applied Mathematics and Computer Science, Ghent University, 2013. http://users.ugent.be/~jgoedgeb/Thesis_Jan_Goedgebeur.pdf. (cited on pages 9, 20, and 94)

48. Grinberg, E. J., Plane homogeneous graphs of degree three without hamiltonian circuits. *Latvian Math. Yearbook*, **4**, (1968), 51–58. (in Russian). English translation by Dainis Zeps, arXiv:0908.2563. (cited on pages 14 and 73)

49. Grünbaum, B., Vertices missed by longest paths or circuits. *Journal of Combinatorial Theory, Series A*, **17(1)**, (1974), 31–38. doi:10.1016/0097-3165(74)90025-9. (cited on page 73)

50. Harmuth, T., *The construction of cubic maps on orientable surfaces*. Ph.D. thesis, Bielefeld University, 2000. (cited on page 95)

51. Hasheminezhad, M.; McKay, B. D.; and Reeves, T., 2009. Recursive generation of 5-regular planar graphs. In *WALCOM: Algorithms and Computation* (Eds. S. Das and R. Uehara), vol. 5431 of *Lecture Notes in Computer Science*, 129–140. Springer Berlin Heidelberg. ISBN 978-3-642-00201-4. doi:10.1007/978-3-642-00202-1_12. (cited on pages 9 and 21)

52. Hasheminezhad, M.; McKay, B. D.; and Reeves, T., Recursive generation of simple planar 5-regular graphs and pentangulations. *Journal of Graph Algortihms and Applications*, **15(3)**, (2011), 417–436. doi:10.7155/jgaa.00232. (cited on pages 9, 19, and 21)

53. Hatzel, W., Ein planarer hypohamiltonscher graph mit 57 knoten. *Math. Ann.*, **243**, (1979), 213–216. (in German). doi:10.1007/BF01424541. (cited on pages 14, 15, 19, 73, and 78)

54. Herz, J. C.; Duby, J. J.; and Vigué, F., 1967. Recherche systématique des graphes hypohamiltoniens. In *Theory of Graphs: International Symposium (1966)*, 153–159. (cited on page 13)

55. Hopcroft, J. and Tarjan, R., A V^2 algorithm for determining isomorphism of planar graphs. *Information Processing Letters*, **1(1)**, (1971), 32–34. doi:10.1016/0020-0190(71)90019-6. (cited on page 45)

56. Hopcroft, J. and Tarjan, R., A $V \log V$ algorithm for isomorphism of triconnected planar graphs. *Journal of Computer and System Sciences*, **7(3)**, (1973), 323–331. doi:10.1016/S0022-0000(73)80013-3. (cited on page 45)

57. Hopcroft, J. E. and Wong, J. K., 1974. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74 (Seattle, Washington, USA, 1974), 172–184. ACM, New York, NY, USA. doi:10.1145/800119.803896. (cited on page 45)

58. Iijima, S., Helical microtubules of graphitic carbon. *Nature*, **354(6348)**, (1991), 56–58. (cited on page 93)

59. Iijima, S.; Yudasaka, M.; Yamada, R.; Bandow, S.; Suenaga, K.; Kokai, F.; and Takahashi, K., Nano-aggregates of single-walled graphitic carbon nano-horns. *Chemical Physics Letters*, **309(3-4)**, (1999), 165–170. doi:10.1016/S0009-2614(99)00642-9. (cited on page 93)

60. Ja'Ja, J. and Kosaraju, S. R., 1986. Parallel algorithms for planar graph. isomorphism and related problems. Technical Report SRC-TR-86-86, University of Maryland, College Park, Department of Electerical Engineering. (cited on page 45)

61. Jooyandeh, M., *k*-Angulations data. http://www.jooyandeh.com/k-angulations. Accessed: 2013-07-20. (cited on page 21)

62. Jooyandeh, M., Planar hypohamiltonian graphs data. http://www.jooyandeh.com/planar_hypo. Accessed: 2013-07-20. (cited on page 78)

63. Jooyandeh, M.; McKay, B. D.; and Brinkmann, G., canemb software. http://www.jooyandeh.com/canemb. Accessed: 2014-02-18. (cited on page 69)

64. Jooyandeh, M.; McKay, B. D.; Östergård, P. R. J.; Pettersson, V. H.; and Zamfirescu, C. T., Planar hypohamiltonian graphs on 40 vertices. *arXiv*, **1302.2698**, (2013), 1–17. http://arxiv.org/abs/1302.2698. (cited on pages 15 and 73)

65. Jordan, C., *Cours d'analyse de l'École polytechnique*. Gauthier-Villars et fils, 1893. (cited on page 8)

66. Junttila, T. and Kaski, P., bliss version 0.72. http://www.tcs.hut.fi/Software/bliss/. (cited on page 45)

67. Junttila, T. and Kaski, P., 2007. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments and the Fourth Workshop on Analytic Algorithms and Combinatorics*, 135–149. (cited on page 45)

68. Kardoš, F., Tetrahedral fulleroids. *Journal of Mathematical Chemistry*, **41(2)**, (2007), 101–111. doi:10.1007/s10910-006-9057-1. (cited on page 93)

69. Katz, T. J. and Acton, N., Synthesis of prismane. *Journal of the American Chemical Society*, **95(8)**, (1973), 2738–2739. doi:10.1021/ja00789a084. (cited on page 93)

70. Krätschmer, W.; Lamb, L. D.; Fostiropoulos, K.; and Huffman, D. R., Solid C_{60}: a new form of carbon. *Nature*, **347(6291)**, (1990), 354–358. doi:10.1038/347354a0. (cited on page 93)

71. Krishnan, A.; Dujardin, E.; Treacy, M.; Hugdahl, J.; Lynum, S.; and Ebbesen, T., Graphitic cones and the nucleation of curved carbon surfaces. *Nature*, **388(6641)**, (1997), 451–454. doi:10.1038/41284. (cited on page 93)

72. Kroto, H. W.; Heath, J. R.; O'Brien, S. C.; Curl, R. F.; and Smalley, R. E., C_{60}: buckminsterfullerene. *Nature*, **318(6042)**, (1985), 162–163. doi:10.1038/318162a0. (cited on pages 1, 19, and 93)

73. Kukluk, J. P.; Holder, L. B.; and Cook, D. J., Algorithm and experiments in testing planar graphs for isomorphism. *Journal of Graph Algortihms and Applications*, **8(3)**, (2004), 313–356. doi:10.7155/jgaa.00094. (cited on pages 45 and 47)

74. Lando, S. K. and Zvonkin, A. K., *Graphs on Surfaces and Their Applications*, vol. 141 of *Encyclopaedia of Mathematical Sciences*. Springer. ISBN 9783540002031, 2004. (cited on pages 4 and 6)

75. Lehel, J., Generating all 4-regular planar graphs from the graph of the octahedron. *Journal of Graph Theory*, **5(4)**, (1981), 423–426. doi:10.1002/jgt.3190050412. (cited on page 21)

76. Li, Z. and Nakano, S.-i., 2001. Efficient generation of plane triangulations without repetitions. In *Automata, Languages and Programming* (Eds. F. Orejas; P. Spirakis; and J. van Leeuwen), vol. 2076 of *Lecture Notes in Computer Science*, 433–443. Springer Berlin / Heidelberg. ISBN 978-3-540-42287-7. doi:10.1007/3-540-48224-5_36. (cited on pages 19 and 21)

77. Lindgren, W. F., An infinite class of hypohamiltonian graphs. *The American Mathematical Monthly*, **74(9)**, (Nov 1967), 1087–1089. doi:10.2307/2313617. (cited on page 16)

78. Liu, J.; Dai, H.; Hafner, J. H.; Colbert, D. T.; and Smalley, R. E., Fullerene 'crop circles'. *Nature*, **385**, (1997), 780–781. doi:10.1038/385780b0. (cited on page 93)

79. Manca, P., Generating all planer graphs regular of degree four. *Journal of Graph Theory*, **3(4)**, (1979), 357–364. doi:10.1002/jgt.3190030406. (cited on page 21)

80. Manolopoulos, D. E. and Fowler, P. W., A fullerene without a spiral. *Chemical Physics Letters*, **204(1-2)**, (1993), 1–7. doi:10.1016/0009-2614(93)85597-H. (cited on pages 1 and 94)

81. Manolopoulos, D. E.; May, J. C.; and Down, S. E., Theoretical studies of the fullerenes: C_{34} to c_{70}. *Chemical Physics Letters*, **181(2-3)**, (1991), 105–111. doi:10.1016/0009-2614(91)90340-F. (cited on pages 1, 19, and 93)

82. McKay, B. D., Practical graph isomorphism. *Congressus Numerantium*, **30**, (1981), 45–87. 10th Manitoba Conference on Numerical Mathematics and Computing (Winnipeg, 1980). (cited on page 45)

83. McKay, B. D., Isomorph-free exhaustive generation. *Journal of Algorithms*, **26(2)**, (1998), 306–324. doi:10.1006/jagm.1997.0898. (cited on pages xvii, 10, 12, 29, 39, and 40)

84. McKay, B. D. and Piperno, A., nauty and traces software. http://pallini.di.uniroma1.it/. Accessed: 2013-09-06. (cited on page 45)

85. McKay, B. D. and Piperno, A., Practical graph isomorphism, II. *Journal of Symbolic Computation*, **60**, (2014), 94–112. doi:10.1016/j.jsc.2013.09.003. (cited on page 45)

86. Meringer, M., 1996. *Erzeugung regulärer Graphen*. Master's thesis, Universität Bayreuth, (in German). ftp://ftp.mathe2.uni-bayreuth.de/meringer/pdf/ErzRegGraphUniBT.pdf. (cited on page 10)

87. Nakamoto, A., Generating quadrangulations of surfaces with minimum degree at least 3. *Journal of Graph Theory*, **30(3)**, (1999), 223–234. doi:10.1002/(SICI)1097-0118(199903)30:3<223::AID-JGT7>3.0.CO;2-M. (cited on pages 19 and 21)

88. Nakano, S.-i., 2001. Efficient generation of triconnected plane triangulations. In *Computing and Combinatorics* (Ed. J. Wang), vol. 2108 of *Lecture Notes in Computer Science*, 131–141. Springer Berlin / Heidelberg. ISBN 978-3-540-42494-9. doi:10.1007/3-540-44679-6_15. (cited on pages 19 and 21)

89. Nakano, S.-i., Efficient generation of triconnected plane triangulations. *Computational Geometry*, **27(2)**, (2004), 109–122. doi:10.1016/j.comgeo.2003.06.001. (cited on pages 19 and 21)

90. Nakano, S.-i. and Uno, T., 2004. More efficient generation of plane triangulations. In *Graph Drawing* (Ed. G. Liotta), vol. 2912 of *Lecture Notes in Computer Science*, 273–282. Springer Berlin / Heidelberg. ISBN 978-3-540-20831-0. doi:10.1007/978-3-540-24595-7_25. (cited on pages 19 and 21)

91. Negami, S. and Nakamoto, A., Diagonal transformations of graphs on closed surfaces. *Science Reports of the Yokohama National University. Section I, Mathematics, Physics and Chemistry*, **40**, (1993), 71–97. (cited on pages 19 and 21)

92. Novoselov, K. S.; Geim, A. K.; Morozov, S.; Jiang, D.; Zhang, Y.; Dubonos, S.; Grigorieva, I.; and Firsov, A., Electric field effect in atomically thin carbon films. *Science*, **306(5696)**, (2004), 666–669. doi:10.1126/science.1102896. (cited on page 93)

93. Paquette, L. A.; Ternansky, R. J.; Balogh, D. W.; and Kentgen, G., Total synthesis of dodecahedrane. *Journal of the American Chemical Society*, **105(16)**, (1983), 5446–5450. doi:10.1021/ja00354a043. (cited on page 93)

94. Piperno, A., Search space contraction in canonical labeling of graphs (preliminary version). *arXiv*, , (2008). http://arxiv.org/abs/0804.4881. (cited on page 45)

95. Proskurowski, A., On the generation of binary trees. *J. ACM*, **27(1)**, (Jan 1980), 1–2. doi:10.1145/322169.322170. (cited on page 10)

96. Rassat, A., Chirality and symmetry aspects of spheroarenes, including fullerenes. *Chirality*, **13(8)**, (2001), 395–402. doi:10.1002/chir.1051. (cited on page 93)

97. Rassat, A.; Scalmani, G.; Seroussi, D.; and Berthier, G., Structure and stability of spheroalkanes (ch)$_{10}$. *Journal of Molecular Structure: {THEOCHEM}*, **338(1-3)**, (1995), 31–41. doi:10.1016/0166-1280(94)04046-U. (cited on page 93)

98. Read, R. C., 1978. Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. In *Algorithmic Aspects of Combinatorics* (Eds. P. H. B. Alspach and D. Miller), vol. 2 of *Annals of Discrete Mathematics*, 107–120. Elsevier. doi:10.1016/S0167-5060(08)70325-X. (cited on page 10)

99. Read, R. C.; Wilson, R. J.; Wilson, R. J.; and Wilson, R. J., *An atlas of graphs*, vol. 21. Clarendon Press Oxford, 1998. (cited on page 44)

100. Robinson, R. W. and Walsh, T. R., Inversion of cycle index sum relations for 2- and 3-connected graphs. *Journal of Combinatorial Theory, Series B*, **57(2)**, (1993), 289–308. doi:10.1006/jctb.1993.1022. (cited on page 44)

101. Schauerte, B. and Zamfirescu, C. T., Regular graphs in which every pair of points is missed by some longest cycle. *Annals of the University of Craiova - Mathematics and Computer Science Series*, **33**, (2006), 154–173. (cited on page 84)

102. Smith, B. W.; Monthioux, M.; and Luzzi, D. E., Encapsulated c$_{60}$ in carbon nanotubes. *Nature*, **396(6709)**, (1998), 323–324. doi:10.1038/24521. (cited on page 93)

103. Sousselier, R., Probléme no. 29: Le cercle des irascibles. *Rev. Franç. Rech. Opérationnelle*, **7**, (1963), 405–406. (cited on pages 1 and 13)

104. Steinitz, E. and Rademacher, H., *Vorelsungen über die Theorie der Polyeder*. Springer Berlin, 1934. (cited on page 21)

105. Stéphan, O.; Bando, Y.; Loiseau, A.; Willaime, F.; Shramchenko, N.; Tamiya, T.; and Sato, T., Formation of small single-layer and nested bn cages under electron irradiation of nanotubes and bulk material. *Applied Physics A*, **67(1)**, (1998), 107–111. doi:10.1007/s003390050745. (cited on page 93)

106. Stephenson, A. and Ward, M. D., An octanuclear coordination cage with a 'cuneane' core-a topological isomer of a cubic cage. *Dalton Transaction*, **40**, (2011), 7824–7826. doi:10.1039/C0DT01767A. (cited on page 93)

107. The On-Line Encyclopedia of Integer Sequences, Number of 2-connected (or biconnected) graphs on n nodes with chromatic number 2. http://oeis.org/A126750. Accessed: 2014-02-23. (cited on pages xv and 71)

108. The On-Line Encyclopedia of Integer Sequences, Number of connected graphs with n nodes. https://oeis.org/A001349. Accessed: 2014-02-18. (cited on pages xiii and 72)

109. The On-Line Encyclopedia of Integer Sequences, Number of pairwise non-isomorphic biconnected planar bipartite graphs on n vertices. http://oeis.org/A122113. Accessed: 2014-02-23. (cited on pages xv, 70, and 71)

110. The On-Line Encyclopedia of Integer Sequences, Number of two-connected (or biconnected) planar graphs with n nodes. http://oeis.org/A021103. Accessed: 2013-09-03. (cited on page 44)

111. The On-Line Encyclopedia of Integer Sequences, Number of two-connected (or biconnected) simple plane graphs with n nodes. http://oeis.org/A228773. Accessed: 2013-09-03. (cited on pages 44 and 70)

112. The On-Line Encyclopedia of Integer Sequences, Number of unlabeled non-separable (or 2-connected) graphs (or blocks) with n nodes. https://oeis.org/A002218. Accessed: 2013-09-03. (cited on page 44)

113. The On-Line Encyclopedia of Integer Sequences, Number of unlabeled planar simple graphs with n nodes. https://oeis.org/A005470. Accessed: 2013-02-18. (cited on pages xiii and 72)

114. Thomassen, C., Hypohamiltonian and hypotraceable graphs. *Discrete Mathematics*, **9(1)**, (1974), 91–96. doi:10.1016/0012-365X(74)90074-0. (cited on pages xiii, 16, 17, 18, and 83)

115. Thomassen, C., Planar and infinite hypohamiltonian and hypotraceable graphs. *Discrete Mathematics*, **14(4)**, (1976), 377–389. doi:10.1016/0012-365X(76)90071-6. (cited on pages 14, 15, 16, 19, 73, and 78)

116. Thomassen, C., 1978. Hypohamiltonian graphs and digraphs. In *Theory and Applications of Graphs* (Eds. Y. Alavi and D. Lick), vol. 642 of *Lecture Notes in Mathematics*, 557–571. Springer Berlin / Heidelberg. ISBN 978-3-540-08666-6. doi:10.1007/BFb0070410. (cited on page 75)

117. Thomassen, C., Planar cubic hypohamiltonian and hypotraceable graphs. *Journal of Combinatorial Theory, Series B*, **30(1)**, (1981), 36–44. doi:10.1016/0095-8956(81)90089-7. (cited on pages xiii, 16, 82, and 84)

118. Tutte, W. T., A theorem on planar graphs. *Transaction of the American Mathematical Society*, **82(1)**, (1956), 99–116. (cited on page 75)

119. Ugarte, D., Curling and closure of graphitic networks under electron-beam irradiation. *Nature*, **359(6397)**, (1992), 707–709. doi:10.1038/359707a0. (cited on page 93)

120. Weinberg, L., A simple and efficient algorithm for determining isomorphism of planar triply connected graphs. *Circuit Theory, IEEE Transactions on*, **13(2)**, (jun 1966), 142–148. doi:10.1109/TCT.1966.1082573. (cited on page 45)

121. West, D. B., *Introduction to Graph Theory*. Prentice Hall, second edn. ISBN 9780130144003, 2001. (cited on pages 14 and 75)

122. Whitney, H., A set of topological invariants for graphs. *American Journal of Mathematics*, **55(1)**, (1933), 231–235. (cited on pages 7, 43, 46, and 93)

123. Wiener, G. and Araya, M., On planar hypohamiltonian graphs. *Journal Graph Theory*, **67(1)**, (2011), 55–68. doi:10.1002/jgt.20513. (cited on pages 14, 15, 16, 17, 19, 73, 74, 78, and 83)

124. Yoshida, M. and Fowler, P. W., Dihedral fullerenes of threefold symmetry with and without face spirals. *Journal of the Chemical Society, Faraday Transactions*, **93**, (1997), 3289–3294. doi:10.1039/A702351K. (cited on pages xv, 94, and 97)

125. Zaks, J., Non-hamiltonian non-grinbergian graphs. *Discrete Mathematics*, **17(3)**, (1977), 317–321. doi:10.1016/0012-365X(77)90165-0. (cited on page 76)

126. Zamfirescu, C. T. and Zamfirescu, T. I., A planar hypohamiltonian graph with 48 vertices. *Journal of Graph Theory*, **55(4)**, (2007), 338–342. doi:10.1002/jgt.20241. (cited on pages 14, 15, 19, 73, and 78)

127. Zamfirescu, T., A two-connected planar graph without concurrent longest paths. *Journal of Combinatorial Theory, Series B*, **13(2)**, (1972), 116–121. doi:10.1016/0095-8956(72)90048-2. (cited on page 74)

Index