List Coloring Some Classes of 1-Planar Graphs

by

Sam Barr

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2021

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Sam Barr is the sole author of Chapters 1, 2, 6, and 7, which were written under the supervision of Professor Therese Biedl and were not written for publication.

Sam Barr is the sole author of Chapter 3, which was written under the supervision of Professor Therese Biedl. Portions of this chapter are based on unpublished work by Professor Therese Biedl, Professor Anna Lubiw, and Owen Merkel [15].

Chapters 4 and 5 were joint work with Professor Therese Biedl and were manuscripts written for publication, see [9, 10].

Abstract

In *list coloring* we are given a graph G and a *list assignment* for G which assigns to each vertex of G a list of possible colors. We wish to find a coloring of the vertices of G such that each vertex uses a color from its list and adjacent vertices are given different colors. In this thesis we study the problem of list coloring 1-planar graphs, i.e., graphs that can be drawn in the plane such that any edge intersects at most one other edge. We also study the closely related problem of simultaneously list coloring the vertices and faces of a planar graph, known as *coupled list coloring*.

We show that 1-planar bipartite graphs are list colorable whenever all lists are of size at least four, and further show that this coloring can be found in linear time. In pursuit of this result, we show that the previously known edge partition of a 1-planar graph into a planar graph and a forest can be found in linear time.

A wheel graph consists of a cycle of vertices, all of which are adjacent to an additional center vertex. We show that wheel graphs are coupled list colorable when all lists are of size five or more and show that this coloring can be found in linear time. Possible extensions of this result to planar partial 3-trees are discussed.

Finally, we discuss the complexity of list coloring 1-planar graphs, both in parameterized and unparameterized settings.

Acknowledgements

I would like to thank Professor Therese Biedl for her support and guidance during my graduate studies. I would also like to thank Professor Anna Lubiw and Professor Luke Postle for agreeing to be readers for my thesis. Many thanks to Owen Merkel for sharing his manuscript.

I am continuously grateful for the support of my family in all my endeavors.

Table of Contents

List of Figures viii

List of Tables xi

1 Introduction 1

1.1 Our Results and Organization 4

2 Definitions 5

2.1 Graphs 5

2.2 Graph Drawing 7

2.3 Graph Coloring 14

2.4 Width Parameters 16

3 4-List Coloring 1-Planar Bipartite Graphs in Linear Time 20

3.1 All 1-Planar Bipartite Graphs are 4-List Colorable 21

3.2 Finding the 4-List Coloring Efficiently 23

3.2.1 Orienting the Edges 24

3.2.2 $(\Delta^+ +1)$-List Coloring Bipartite Graphs Efficiently 25

3.3 Complexity of 3-List Coloring 1-Planar Bipartite Graphs 30

4 Partitioning the Edges of a 1-Planar Graph Efficiently **36**
4.1 Ackerman's Proof . . . 37
4.2 An Alternate Existence Proof . . . 39
4.3 Efficient Implementation . . . 44
4.3.1 Data Structure Interface . . . 44
4.3.2 Preprocessing . . . 45
4.3.3 Handling the Quadrangles around a Vertex . . . 48
4.3.4 Putting it All Together . . . 50

5 Coupled List Coloring Planar Graphs of Small Treewidth **54**
5.1 Simple Coupled Choosability Results . . . 54
5.2 Treewidth of Coupled Graphs . . . 57
5.3 Coupled Graphs and Optimal 1-Planar Graphs . . . 61
5.4 Coupled List Coloring Subgraphs of Wheels . . . 66
5.5 Towards Planar Partial 3-Trees . . . 72

6 Complexity of List Coloring **76**
6.1 Definitions . . . 77
6.2 Para-NP-Hard problems . . . 78
6.3 Problems in XP . . . 79
6.4 W[1]-Hard Problems . . . 80
6.5 Problems in FPT . . . 83
6.6 Π_2^p-Complete Problems . . . 84

7 Conclusion **88**

References **91**

List of Figures

1.1 An example of a 1-planar graph. We have also assigned a list of three possible colors to each vertex of the graph, with the bold color in each list denoting a valid coloring from this list assignment. 3

2.1 Examples of graphs drawn in the plane. 7

2.2 Examples of triangular faces. The three angles in each face have been marked with solid arrows, while edge-sides have been marked with dotted arrows. Note that the face on the right has only two vertices on its boundary. . . . 8

2.3 A crossing point p of a 1-plane graph, with the kite edges shown as dotted edges. 11

2.4 A planar graph, its dual, and the graph induced by the intermediate edges. Note the dual parallel edges resulting from the cutting pairs of edges, and the parallel intermediate edges resulting from the cut vertices. 12

2.5 Examples of faces of a planar graph, with the dual vertex, dual edges, and intermediate edges drawn in. Observe that all new cells in f are 3-cells. . 13

2.6 A graph G (left) with a tree decomposition (right) of width two. One confirms that the tree decomposition follows the three rules of Definition 2.4.1. As G is not a tree, we know that $tw(G) = 2$. 17

2.7 The graph K_4 (left) with a branch decomposition (right). Two arcs in the branch decomposition have been labeled with their respective separator. This branch decomposition demonstrates that K_4 has branchwidth at most three. (In fact, it is exactly three.) 18

3.1 A 1-planar bipartite graph that is not 3-choosable. The numbers on each vertex represent the list assignment L for each vertex. The red (solid ellipse) vertices and blue (rectangular) vertices form one part of the bipartite graph, and the black (dashed ellipse) vertices form the other part. The blue (rectangular) vertices, labeled x and y, are connected to all of the black (dashed ellipse) vertices. Next to each red (solid ellipse) vertex we have labelled the pair of colors for x and y that that vertex blocks. 30

3.2 The construction in Lemma 3.3.3. 32

3.3 Transformation applied to each edge in the reduction of Theorem 3.3.5. . 33

4.1 The construction used in Lemma 4.2.1. (The face f' is not shown.) New vertices are represented with circles, and new edges are represented with dashed edges. 40

4.2 Some configurations where we contract z_1 and z_3. Note that in (a), the quadrangle f is the outer face. 41

4.3 The gadget added to every quadrangle of H, shown for both a quadrangle with four distinct vertices on its boundary (left) and one with three distinct vertices on its boundary (right). 45

5.1 The graph K_4 (left) and subgraph H (right). 55

5.2 A vertex x resulting from subdividing the edge uv, along with the two faces f_1 and f_2 adjacent to x. 56

5.3 A primal edge xy in the coupled graph of a plane graph G (left), and the transformation applied to the leaf of the branch decomposition of G containing xy used in Theorem 5.2.3 (right). 59

5.4 A cylinder graph $Z_{8,4}$ (left), along with a cylinder graph $Z_{16,3}$ shown as a subgraph of $C(Z_{8,4})$ (right). 60

5.5 A grid graph $R_{4,7}$ (left), along with a grid graph $R_{7,6}$ shown as a subgraph of $C(R_{4,7})$. 60

5.6 A vertex $d \in D$ in Q (left), along with its neighbors. From construction of Q, each quadrangle adjacent to d corresponds to a crossing pair of C (shown with dotted edges) which we use to find the face f of G (right). 65

5.7 The graph W_9 (left) and X_9 (right). Circled numbers indicate a lower bound on the list lengths in L' defined in the cases below. 67

5.8 The graph X_9. Circled numbers indicate a lower bound on the list lengths in L''. Solid edges show the graph X'_9. 69

5.9 The graphs X_5 and X_6. 70

5.10 The graph X_9. Some of the vertices have been labelled with the 4-list assignment defined in the proof of Theorem 5.4.2. 71

5.11 A planar partial 3-tree. Dotted edges show the Apollonian network G. We can see that G is an Apollonian network: Start with the triangle on the outer face. Stellate this triangle with the vertex labeled with 1. Then stellate one of the new faces with the vertex labeled with 2. Continue this process with the vertices in the order they are labeled. 72

5.12 A Halin-graph G (black solid; the tree is bold), and the dual graph G^* (blue dashed) which is a stellated outerplanar graph (the outerplanar graph is bold). Taking both, and adding the intermediate edges (red dotted) gives graph $C(G)$. 74

5.13 An IO graph G consists of an outerplanar graph (circles) and an independent set (squares). Dotted edges are added to obtain G^+, and some of the wheels used to build G^+ are shaded. 74

6.1 The relationship between the parameterized complexity classes para-NP, XP, W[1], and FPT. Figure is based on [33]. 78

6.2 A graph with a proper 4-coloring (left), with the constructed instance of list coloring (right). On the left vertices are labeled with $a, b, c, \ldots$, and color classes are denoted by the shape and color of the vertex. Figure is based on [32]. 81

6.3 A 1-planar graph G constructed by taking four disjoint K_4s, and adding two vertices adjacent to every vertex in one of the K_4s. We also give a list assignment for G where every list is of size 5, except for one list which is of size 4. 85

List of Tables

3.1 The complexity of k-Coloring for planar graphs compared to the complexity of k-ListColoring for 1-planar bipartite graphs. 35

4.1 All possible cases for the quadrangle f with facial circuit $\langle z_0, z_1, z_2, z_3 \rangle$. We either indicate which case in the proof of Theorem 4.2.3 would be chosen, or indicate the lemma that demonstrates that this case is impossible. . . . 43

6.1 Known results and open problems for the complexity of k-Choosability on planar, 1-planar, and optimal 1-planar multigraphs. 87

Chapter 1

Introduction

Graphs are a fundamental data structure in computer science, with many applications when modeling roadway maps, social networks, and telephone lines. A graph consists of *vertices* (also known as *nodes*) and *edges* between the vertices.

In graph coloring, one wishes to assign to every vertex of a graph a color such that adjacent vertices have different colors, using as few colors as possible. A graph is said to be k-*colorable* when it can be colored using k colors, and the *chromatic number* of a graph is the least k such that the graph is k-colorable. Graph coloring is a very well studied problem in graph theory and computer science [57, 59]. There are many practical applications of graph coloring, including register allocation in compilers [22], job scheduling [61], and radio frequency assignment [44].

List coloring is a generalization of graph coloring where every vertex of the graph is given a list of colors that it can use. The problem of list coloring was first studied by Vizing [86] and by Erdős, Rubin, and Taylor [29]. A graph is said to be k-*choosable* if it is colorable for any choice of list of size k for each vertex. The *list chromatic number* of a graph is the least k such that the graph is k-choosable. It is easy to see that if a graph is k-choosable then it is k-colorable—simply assign each vertex the same list of k colors. This does not work the other way around, and in general the list chromatic number of a graph is not bounded by the chromatic number of a graph. As an example, bipartite graphs are all 2-colorable, but have an unbounded list chromatic number. (This was observed by both Erdős et al. and by Vizing in the above papers.)

Vizing proved that the only cycles that are 2-choosable are the even cycles. Erdős et al. completely characterized the 2-choosable graphs. These graphs can be recognized

and colored in linear time. Since the introduction of the problem, list coloring has been extensively studied, see e.g. [3, 54, 70, 71, 72, 89].

In this thesis, we study list coloring in special graphs, namely those that have a particular kind of drawing. We first recall that *planar graphs* are graphs that can be drawn in the plane with a point for each vertex and a curve between the endpoints for each edge such that the curves for each edge do not cross over each other. Planar graphs have been a popular research topic for many years, as many problems that are difficult on general graphs become easy on planar graphs (see e.g. [66]). In particular, the colorability and choosability of planar graphs is well studied. Famously, it was proved that every planar graph can be colored with four colors [4, 5, 6]. This is tight, as the complete graph K_4 is planar and is not 3-colorable. The original proof of 4-colorability led to a $O(n^4)$ time algorithm, which was later improved to $O(n^2)$ time [74]. Thomassen proved that every planar graph is 5-choosable [82], and his proof leads to an $O(n^2)$ time algorithm to find the coloring. This was later improved to linear time [68]. Thomassen's result is also tight [63, 87]. There are many other results on the choosability of planar graphs under restrictions on the lengths of short cycles, see e.g. [40, 49, 60, 69, 83, 88].

The main graph class we study in this thesis is the class of *1-planar graphs*, a generalization of planar graphs where in the drawing of the graph in the plane we allow any edge to intersect at most one other edge. (A formal definition of "drawing" will be given in Chapter 2.) These graphs were first studied by Ringel [73], who was interested in the problem of simultaneously coloring the vertices and faces of a planar graph, also known as *coupled coloring*. (The graph resulting from this coloring problem is a so-called optimal 1-planar graph.) Since Ringel's work, and especially in recent years, research interest in 1-planar graphs has grown greatly. See [56] for an annotated bibliography from 2017 on 1-planar graphs, and [47] for a book from 2020 that covers 1-planar graphs, and more generally the field of so-called near planar graphs.

In his original paper, Ringel [73] showed that every 1-planar graph is 7-colorable. This was later improved to 6-colorable by Borodin [19]. (See [20] for an improved proof of this result by Borodin that is written in English.) The latter proof leads to a polynomial time algorithm to find the coloring. Furthermore, 6-colorability is tight for 1-planar graphs: The complete graph K_6 is 1-planar, and is not 5-colorable. Archdeacon [7] studied coupled colorability, and showed that planar bipartite graphs are 5-coupled colorable. Berman and Shank [13] characterized the planar graphs that are 4-coupled colorable.

It is not hard to see that any 1-planar graph is 8-choosable, since every 1-planar graph has a vertex of degree at most 7 (see also Corollary 2.3.2). It is unknown whether 1-planar graphs are 7-choosable. Wang and Lih [90] studied the problem of simultaneously list

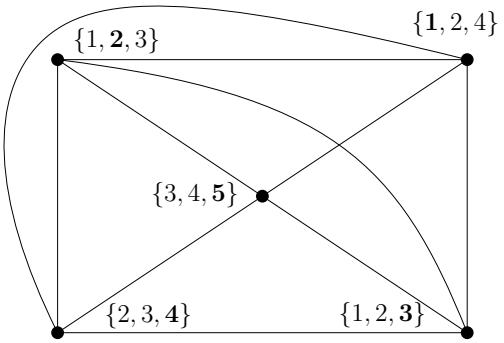


Figure 1.1: An example of a 1-planar graph. We have also assigned a list of three possible colors to each vertex of the graph, with the bold color in each list denoting a valid coloring from this list assignment.

coloring the vertices and faces of a graph (also known as *coupled choosability*) and show that lists of size 7 suffice. In particular this shows that so-called optimal 1-planar graphs are 7-choosable. Their proof leads to a polynomial time algorithm to find the coloring. It is unknown if this result is tight. Very recently, Yang et al. [91] showed that so-called IC-planar graphs (a subclass of 1-planar graphs) are 6-choosable.

We list here a few more results for coupled choosability of subclasses of planar graphs. These in turn imply list coloring results for subclasses of 1-planar graphs. As mentioned above, Wang and Lih [90] show that every planar graph is 7-coupled choosable. They also prove other coupled choosability results: They show that maximal planar graphs and planar graphs of maximum degree three are 6-coupled choosable. They also prove that all K_4-minor free graphs are 5-coupled choosable. Hetherington [45] provides another proof of the same K_4-minor free graph result, and also proves 5-coupled choosability for classes of so-called near outerplanar graphs.

There are numerous variations on coloring problems that impose further conditions. We list here a few results that specifically studied such results for 1-planar graphs, but do not otherwise consider such variations in this thesis. Zhang and Li [92] studied dynamic list colorings of 1-planar graphs, where at least two distinct colors must appear in the neighborhood of a vertex with two or more neighbors. List edge coloring, list total coloring (simultaneous coloring of edges and vertices), and other variations have also been studied for 1-planar graphs [79, 80, 93, 94].

1.1 Our Results and Organization

In this thesis, we study list coloring on several classes of 1-planar graphs. To do so, we first review in Chapter 2 definitions and notation for graph theory, graph drawing, and graph coloring that will be used in this thesis. We also define the major graph classes that will be discussed throughout the thesis. Then we show the following:

- In Chapter 3 we show that 1-planar bipartite graphs are 4-choosable. The existence of such a coloring follows quite easily from known results for list coloring bipartite graphs; our main result here is to show that this coloring can be found in linear time. We also show that 4-choosability is tight; in particular testing whether a 1-planar bipartite graph with given lists of length 3 has a valid coloring is NP-hard.
- A crucial ingredient for the results of Chapter 3 is the ability to partition the edges of a 1-planar graph into a planar graph and a forest. It was previously known that such a partition exists, but it was not known to be doable in linear time. In Chapter 4 we give a linear time algorithm to find this edge partition.
- In Chapter 5 we study the problem of coupled choosability for planar graphs of small treewidth. In doing so, we study the structure of coupled graphs, in particular upper bounding their treewidth and characterizing their relationship to optimal 1-planar graphs. We then characterize the coupled choosability of wheel graphs, doing so with an eye towards the coupled choosability of planar partial 3-trees. We then discuss possible extension of our results to various subclasses of planar partial 3-trees.
- In Chapter 6 we give many results on the complexity of list coloring 1-planar graphs and of coupled choosability, focusing mainly on these problems when parameterized by treewidth.

We close in Chapter 7 with a summary of the results of this paper, and discuss several open questions.

Chapter 2

Definitions

In this chapter we go over notation and terminology for graphs and graph coloring that will be used throughout this thesis.

2.1 Graphs

A graph G is a structure consisting of a set of *vertices* $V(G)$ and a collection of *edges* $E(G)$. Each edge $e \in E(G)$ consists of a pair of vertices $x, y \in V(G)$, and we write $e = xy$. The vertices x and y are the *endpoints* of e, and x and y are said to be *adjacent*. The *degree* $d(x)$ of a vertex x is the number of edges e such that x is an endpoint of e. Note that we specifically count the number of edges, not the number of adjacent vertices. When it would not be clear from context, we use $d_G(x)$ to denote the degree of vertex x in the graph G.

An edge $e \in E(G)$ is a *loop* if both endpoints of e are the same vertex; i.e. $e = xx$ for some vertex $x \in V(G)$. Two edges $e, e' \in E(G)$ are *parallel* if they both have the same endpoints, i.e. $e = xy$ and $e' = xy$ for two vertices $x, y \in V(G)$. A graph with no loops and no parallel edges is known as a *simple graph*, and a graph that is permitted to have loops and parallel edges is known as a *multigraph*. Note that in a multigraph it is possible for the degree of a vertex to be strictly larger than the number of adjacent vertices. A *path* is a list of vertices $x_1, x_2, \ldots, x_k$ such that $x_i x_{i+1}$ is an edge for $1 \leq i \leq k-1$ and $x_i \neq x_j$ when $i \neq j$. A *walk* is a path where we allow repeating vertices. Two vertices $x, y \in V(G)$ are *connected* if there is a path (equivalently, if there is a walk) $x, v_1, \ldots, v_k, y$. A graph G is *connected* if all vertices in G are connected, and disconnected otherwise. More generally,

a graph G is *k-vertex-connected* if G remains connected after removing any $k-1$ vertices. Similarly, G is *k-edge-connected* if G remains connected after removing any $k-1$ edges. A *cut vertex* of a graph G is a vertex $x \in V(G)$ such that removing x from G creates a disconnected graph. A *bridge* is an edge $e \in E(G)$ such that removing e from G creates a disconnected graph.

Except where explicitly stated, all graphs in this thesis are connected. In certain chapters we will assume simplicity; we will make it clear when it is assumed. We will use the variable n to refer to the number of vertices in a graph, and m to refer to the number of edges in a graph. (The graph they refer to will be clear from context.)

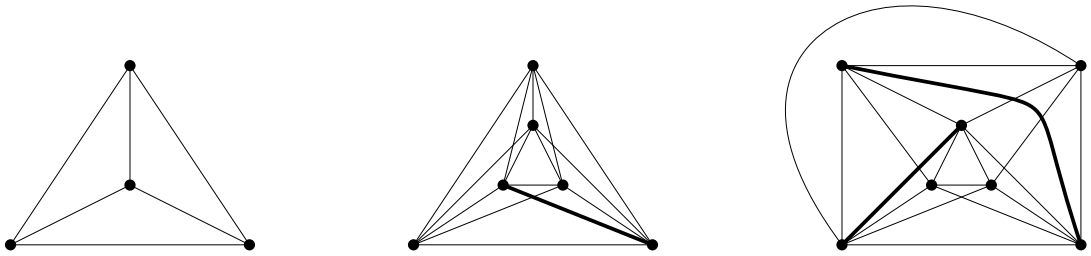
A *cycle* is a list of vertices $x_1, x_2, \ldots, x_k$ such that $x_i \neq x_j$ when $i \neq j$, $x_i x_{i+1}$ is an edge for $1 \leq i \leq k-1$, and additionally $x_1 x_k$ is an edge. A connected graph with no cycles is a *tree*, and a disconnected graph with no cycles is a *forest*. A *circuit* is a cycle where we allow repeating vertices.

The *complete graph* K_n is the simple graph with n vertices where all vertex-pairs are adjacent: for any two vertices $x, y \in V(K_n)$ there is an edge $xy \in E(K_n)$. A graph G is *bipartite* if $V(G)$ can be partitioned into two sets A, B such that for any edge $e \in E(G)$, one endpoint of e is in A and the other is in B. The *complete bipartite* graph $K_{a,b}$ is the simple bipartite graph with partition A, B where $|A| = a$ and $|B| = b$ such that for any vertices $x \in A$ and $y \in B$, there is an edge $xy \in E(K_{a,b})$.

A graph H is a *subgraph* of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, and we write $H \subseteq G$. The graph G is a *supergraph* of H. Let $A \subseteq V(G)$. By $G[A]$ we denote the subgraph of G with vertex set A and edge set $\{e = xy \in E(G) : x, y \in A\}$. We say that $G[A]$ is an *induced subgraph* of G. For an edge set $B \subseteq E(G)$, we similarly define an induced subgraph $G[B]$ with edge set B and vertex set $\{x \in V(G) : x \text{ is the endpoint of an edge in } B\}$. For a vertex $x \in V(G)$, we use $G - x$ to denote the induced subgraph $G[V(G) \setminus \{x\}]$, and similarly for an edge $e \in E(G)$ we define $G - e := G[E(G) \setminus \{e\}]$.

An *independent set* in a graph G is a set of vertices $A \subseteq V(G)$ such that for any vertices $a, b \in A$, we have that a and b are not adjacent in G.

Let G be a graph and let $e = xy$ be an edge of G. We can *subdivide* the edge e by deleting e from G and adding a new vertex $z \notin V(G)$ with two edges xz and zy. If H is obtained from G by subdividing edges of G, then H is a *subdivision* of G. Let a, b be two vertices of G. We *contract* a and b by creating a new vertex c, adding an edge vc for each edge va and vb that exists, and deleting a and b. If a and b were adjacent, then c has a loop, and if a and b were both adjacent to a vertex v, then c will have parallel edges to v. Note that we permit contracting vertices that are not adjacent. If G has an edge $e = ab$, then we write G/e for the graph obtained by contracting a and b. We say that a graph H



(a) The graph K_4, a planar graph.

(b) The graph K_6, a 1-planar graph.

(c) The graph K_7, which is not 1-planar.

Figure 2.1: Examples of graphs drawn in the plane.

is a *minor* of a graph G if H can be obtained from G by deleting edges and vertices and contracting edges (here we delete loops and parallel edges resulting from the contractions). If G does not contain H as a minor, then we say that G is *H-minor free*.

A *directed* graph is a graph D where for any edge $e = xy \in E(D)$, we distinguish x to be the *tail* of e, and y to be the *head* of e. In this way the edge e has a direction, and we say that e *goes from x to y*. The *outdegree* of a vertex x, denoted $d^+(x)$, is the number of directed edges $xy \in E(G)$ coming out of x. We call x a *sink* if $d^+(x) = 0$. Given an undirected graph G, we can construct a directed graph D by assigning to each undirected edge $e = xy \in E(G)$ one of x, y to be the head and the other to be the tail. This is known as *orienting* the edge e. The directed graph D is an *orientation* of G.

We assume that all graphs are given with the following data structure: Every vertex v has an *incidence list* listing all edges incident to v. This is a doubly linked list that knows its length. Each edge knows its two endpoints and has pointers to its to occurrences in the incidence lists.

2.2 Graph Drawing

A common way to visualize a graph is to draw it in the plane, with the vertices represented by dots, and the edges drawn as curves connecting two dots. Formally, a *drawing* Γ for a graph G is a function which maps each vertex $x \in V(G)$ to a distinct point $\Gamma(x) \in \mathbb{R}^2$, and maps each edge $e = xy \in V(G)$ to a curve $\Gamma(e)$ whose endpoints are $\Gamma(x)$ and $\Gamma(y)$ and whose interior does not intersect $\Gamma(z)$ for any vertex z. See Figure 2.1 for examples of graphs drawn in the plane.

Figure 2.2: Examples of triangular faces. The three angles in each face have been marked with solid arrows, while edge-sides have been marked with dotted arrows. Note that the face on the right has only two vertices on its boundary.

In Figure 2.1(a), we see that the edges have been drawn such that the curves for any two edges do not intersect each other (except at common endpoints) whereas in Figures 2.1(b) and 2.1(c) we see that the bold edges cross over other edges. A graph G that can be drawn with no edges intersecting (except at common endpoints) is known as a *planar graph*. It is possible to test whether a graph is planar in linear time [48]. There could be many ways to draw G in the plane such that no edges intersect, so if one particular drawing Γ of G in the plane is fixed we call G a *plane graph*. The maximal regions of $\mathbb{R}^2 \setminus \Gamma$ are known as the *faces* of G, denoted $F(G)$. The unique face whose region is unbounded is known as the *outer face*. (The plane graph in Figure 2.1(a) has four faces.) If a graph G can be drawn such that every vertex lies on the boundary of the outer face, then G is an *outerplanar* graph. We say that two faces $f, f' \in F(G)$ are *adjacent* if they share an edge on their boundaries.

For algorithmic purposes, we require an efficient way to store a drawing of a planar graph in the plane. We do so via *rotation systems*. A *rotation system* of a graph G assigns to each vertex $x \in V(G)$ a *cyclic order* π_x of the edges incident to x. We can see that a planar drawing of a plane graph G specifies a rotation system of G, with the cyclic order π_x of a vertex x listing the adjacent vertices in clockwise order. Likewise, given a rotation system of a graph G that comes from a planar drawing, it is possible to recover the faces of a planar drawing of G (see e.g. [64] for details). If along with the rotation system an outer face is specified, we can fully recover a planar drawing of G that has this rotation system and outer face. A *planar embedding* of a graph G is this rotation system and specified outer face.

Let G be a plane graph given with planar drawing Γ. For an edge $e = xy$, while walking from x to y, we have a face f_L to the left side and a face f_R to the right side (possibly $f_L = f_R$). We say that f_L and f_R are *incident* to e, and call these incidences *edge sides*. For a face f of G, we define the *degree* of f to be the number of edge-sides on the boundary of f. We also give another, equivalent definition for the degree of a face: For a vertex x of

G with incident cyclic order of edges $e_0, \ldots, e_{d-1}$, an *angle* is a triple $\langle e_i, f, e_{i+1} \rangle$ for some face f, $i \in \{0, \ldots, d-1\}$, and addition modulo d such that f is on the right edge side of e_i and the left edge side of e_{i+1} when traversing these edges away from x. The *degree* of a face is the number of angles contained in a face. We note that in a plane multigraph, the number of vertices on the boundary of a face is not always the same as the degree of a face. (See Figure 2.2.) We define a few special kinds of faces: A face of degree 2 is a *bigon*, a face of degree 3 is a *triangle*, and a face of degree 4 is a *quadrangle*. A *plane (multi)triangulation* is a plane (multi)graph G where every face is a triangle, and a *plane (multi)quadrangulation* is a plane (multi)graph G where every face is a quadrangle. Any plane (multi)quadrangulation is bipartite [81].

The *Euler characteristic* relates the number of vertices, edges, and faces in a planar graph. See e.g. [66] for a proof.

Theorem 2.2.1. *Let G be a connected (not necessarily simple) plane graph with n vertices, m edges, and f faces. Then $n - m + f = 2$.*

From the Euler characteristic, the following lemma can be derived.

Lemma 2.2.2. *Let G be a connected planar graph, $n \geq 3$, which permits a drawing where every face has degree at least three. Then G has at most $3n - 6$ edges and at most $2n - 4$ faces. These are equalities if and only if G is a plane multitriangulation.*

A simple planar graph with exactly $3n - 6$ edges is known as a *maximal planar graph*. Every simple planar graph can be turned into a maximal planar graph by adding edges.

We review some graph operations that do not affect planarity. *Stellating* a face f of a planar graph is the process of adding a new vertex s inside the region of f, and adding an edge from s to every vertex on the boundary of f. We add a *chord* to a face f by adding an edge between two non-consecutive vertices on the boundary of f and drawing the edge through the region of f. Finally, for any edge e the graph obtained by contracting e is planar.

Let G be a connected plane graph. The *dual* graph G^* of G is the graph with vertex set $F(G)$, where two faces f, f' are adjacent in G^* whenever there is an edge e such that one edge-side is incident to f and the other edge-side is incident to f'. (The dual graph is undefined if G is not connected.) The original graph G is known as the *primal* graph. The dual graph of a planar graph is planar, and every edge e of G corresponds to a *dual edge* e^* of G^*. The dual graph need not be simple, however. If G has a bridge, then G^* has a loop. Moreover, if G contains a *cutting pair of edges* (two edges which would disconnect the graph if removed), then G^* will contain parallel edges.

While the drawings given in Figures 2.1(b) and 2.1(c) are both not planar drawings (and indeed neither graph is a planar graph), a further distinction can be made. In Figure 2.1(b), every edge intersects at most one other edge. In contrast, one of the bold edges in Figure 2.1(c) intersects two other edges. We say that the drawing in Figure 2.1(b), where every edge crosses at most one other edge, is a *1-planar drawing*. When we talk about a drawing of a graph that has crossings, we assume that the drawing is what is known as a *good drawing*. See [77] for a full definition of the many properties of a good drawing; for our purposes here it suffices to know that no three edges cross in one point, that edges that share an endpoint do not intersect, that no two edges intersect more than once, and that no edge intersects itself. We do, however, allow loops to be crossed because such crossings arise naturally later in Lemma 2.2.4. For graphs with a 1-planar drawing, the drawing can be modified to be a good drawing without increasing the number of crossings on a single edge [67]. If a graph G has a 1-planar drawing, then G is known as a *1-planar graph*. If a 1-planar drawing Γ of G is fixed, then G is referred to as a *1-plane graph*. Two intersecting edges of a 1-plane graph are known as a *crossing pair*, and the point where they intersect is a *crossing point*. We call the maximal regions of $\mathbb{R}^2 \setminus \Gamma$ the *cells* of G. (We use "cells" for a 1-plane graph to differentiate from the "faces" of a plane graph.)

The *planarization* of a 1-planar graph G is a graph $G^\times$ obtained by, for each crossing pair of edges ab and cd, deleting these edges and adding a new vertex p adjacent to a, b, c, d. The planarization of a 1-planar graph is a planar graph, and there is a 1-1 mapping between the faces of $G^\times$ and the cells of G. Let c be a cell of G, and let $c^\times$ be the corresponding face in $G^\times$. A *corner* in c is an angle in $c^\times$. A corner in a cell c occurs either at a vertex on the boundary of c or at a crossing point on the boundary of c. A cell c may be referred to as a *bigon* if the corresponding face $c^\times$ is a bigon, and likewise for the terms *triangle* and *quadrangle*. We say that a (1-)planar drawing Γ is a (≥ 3)*-cell drawing* if every cell has at least three corners. In particular, by Lemma 2.2.2, a plane graph G with a (≥ 3)-cell drawing and $n \geq 3$ has at most $3n - 6$ edges since every face has degree at least three.

Let G be a 1-plane graph and p a crossing point of G formed by edges ab and cd. The edge ac, if it exists, is a *kite edge* if it is on the boundary of a cell whose corners are exactly the vertices a and c and the crossing point. Likewise, the edges ad, bc, bd, if they exist, may be kite edges. (See the dotted edges in Figure 2.3).

Any simple 1-planar graph has at most $4n - 8$ edges [18]. A simple 1-planar graph is called an *optimal 1-planar graph* if it has exactly $4n - 8$ edges. Note that not every planar graph is the subgraph of an optimal 1-planar graph. In particular the complete graph K_6 is 1-planar (see Figure 2.1(b)), but is not the subgraph of an optimal 1-planar graph since it has $15 = 4n - 9$ edges but no edge can be added while maintaining simplicity. A 1-plane graph G is *planar maximal* if any edges added to G while keeping the specified drawing

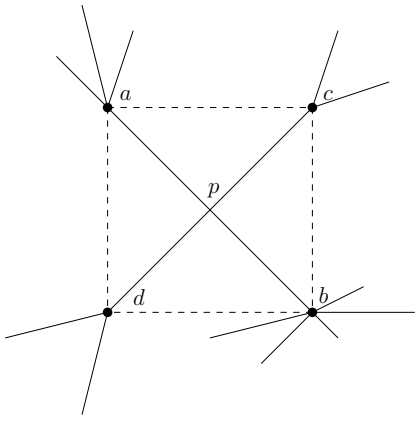


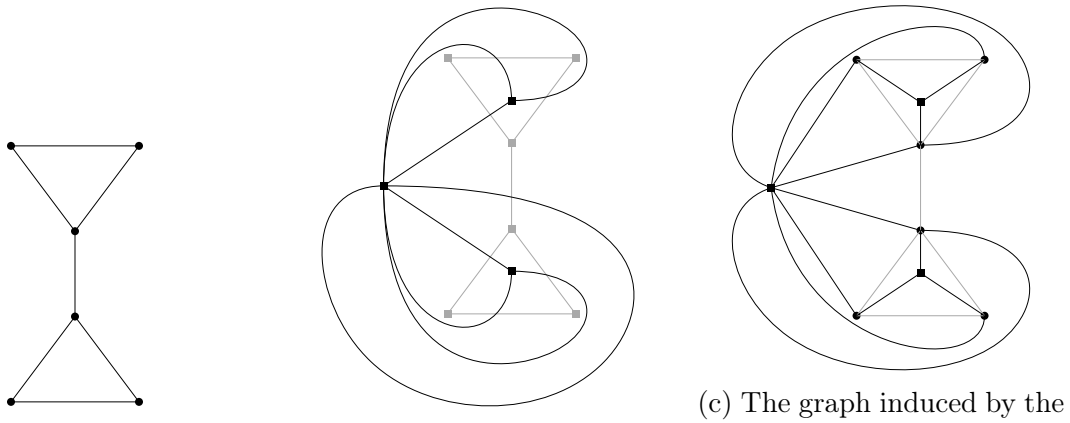
Figure 2.3: A crossing point p of a 1-plane graph, with the kite edges shown as dotted edges.

would either form a cell with at most two corners or would add a new crossing. Every cell in a planar maximal 1-plane graph has at most three corners (and possibly some cells have only one or two corners). The following was known (see e.g. [2]), but we give details here because we specifically allow for multigraphs and because we need some properties of the resulting drawing later.

Lemma 2.2.3. *Let G be a 1-plane graph. We can construct a planar maximal 1-plane supergraph G^+ of G in linear time by adding edges to G. Moreover, if the drawing of G was a (≥ 3)-cell drawing, then so is the one of G^+.*

Proof. Let p be a crossing point of G created by edges ab and cd. Add all kite edges around p if they do not already exist. As a result, any cell containing a crossing point has degree three. (This may add parallel edges to G, but does not create bigons.) To finish the construction, triangulate all faces of the planarization $G^\times$ with degree greater than three (these faces correspond to cells of G that do not contain a crossing point). Let G^+ be the resulting graph; it can be constructed from G in linear time. It is clear that our construction does not add any cells with at most two corners, and hence if G was given with a (≥ 3)-cell drawing, then the drawing of G^+ is also a (≥ 3)-cell drawing.

We now argue that G^+ is planar maximal. Any edge added would have to be added within some cell c of G^+ as we cannot add new crossings. If c contains a crossing point p on its boundary, then c is the result of adding edges around a crossing point and has exactly two vertices on its boundary. Therefore any edge added within c would create a bigon or a loop. Otherwise, c has only vertices on its boundary. It is clear that any edge e



(a) A planar graph. (b) The dual graph. (c) The graph induced by the intermediate edges.

Figure 2.4: A planar graph, its dual, and the graph induced by the intermediate edges. Note the dual parallel edges resulting from the cutting pairs of edges, and the parallel intermediate edges resulting from the cut vertices.

we could add within c would create a bigon or a loop, as otherwise we would have added e when triangulating the faces of $G^\times$. □

A *1-planar embedding* of a 1-planar graph G is a planar embedding of the planarization $G^\times$, with the crossing vertices of $G^\times$ marked as such and storing the two original edges of G that were crossed. From a 1-planar embedding of G we can recover a 1-planar drawing. Given a graph, it is NP-hard to determine whether it is 1-planar [38]. Thus, for all algorithms in this thesis involving 1-planar graphs, we assume that the 1-planar graph is given with a 1-planar embedding.

We now explain how planar graphs naturally induce 1-planar graphs. Let G be a connected plane graph. The *coupled graph* $C(G)$ of G is constructed by taking the union of G and its dual G^* and adding edges between incident faces and vertices. The edge set of $C(G)$ can be partitioned into the primal edges of G, the dual edges of G^*, and the *intermediate* edges representing incidences between vertices and faces. (We add repeated edges for repeated incidences.) Note that the intermediate edges do not necessarily induce a simple graph: If G has a cut vertex, then $C(G)$ will have parallel intermediate edges (see Figure 2.4(c)). Observe that the subgraph of $C(G)$ induced by the intermediate edges is bipartite, with the primal vertices forming one partition and the dual vertices forming the other. We also note here that for any plane graph G, we have $C(G) = C(G^*)$. Ringel observed that $C(G)$ is 1-planar [73], and by inspecting his proof one sees that the resulting

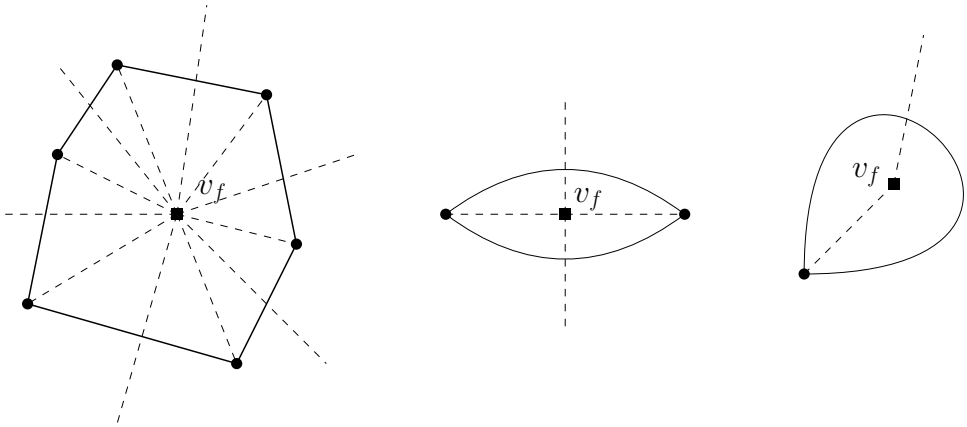


Figure 2.5: Examples of faces of a planar graph, with the dual vertex, dual edges, and intermediate edges drawn in. Observe that all new cells in f are 3-cells.

1-planar drawing of $C(G)$ is a (≥ 3)-cell drawing. We here expand his proof to permit non-simple graphs.

Lemma 2.2.4. *Let G be a plane graph that is not necessarily simple and for which the given drawing is not necessarily a (≥ 3)-cell drawing. Then $C(G)$ has a good 1-planar (≥ 3)-cell drawing.*

Proof. For every face f of G, add the dual vertex v_f representing f inside the region of f. From there, draw the dual edges such that each dual edge e^* only intersects the corresponding primal edge e. This divides f into cells with four corners. Lastly, we draw the intermediate edges without any crossings. If vertex x was incident to face f, then after placing v_f and the dual edges we have a cell with corners v_f, c, x, c' (c and c' are crossings, not vertices). Drawing the intermediate edge $v_f x$ divides this cell into two cells that each have three corners. (See Figure 2.5.)

It is clear that the resulting drawing Γ_C of $C(G)$ is a 1-planar drawing. It is moreover a (≥ 3)-cell drawing as any cell of Γ_C was obtained by splitting some face f as described above; we only create cells with three corners. Moreover, we see that Γ_C is a good drawing as a crossing pair is always formed by a primal edge and a dual edge, and hence the edges do not share an endpoint. □

2.3 Graph Coloring

Let G be a graph. We say that G is *k-colorable* if we can assign an integer $\phi(x)$ to every vertex $x \in V(G)$ such that $1 \leq \phi(x) \leq k$ and for any adjacent vertices $x, y \in V(G)$, we have that $\phi(x) \neq \phi(y)$. The *chromatic number* of G, denoted $\chi(G)$, is the least k such that G is k-colorable.

A *list assignment* L for a graph G is an assignment of a list of colors to each vertex of G. For a color c, we define $L^{-1}(c) := \{x \in V(G) : c \in L(x)\}$ to be all vertices of G whose list contains the color c. We say that G is *L-colorable* if we can assign to each vertex x of G a color $\phi(x)$ such that for every vertex x, we have $\phi(x) \in L(x)$, and for every edge $e = xy$ of G, we have $\phi(x) \neq \phi(y)$. A list assignment L is a *k-list assignment* if $|L(x)| \geq k$ for every vertex x of G. A graph G is *k-choosable* if G is L-colorable for any k-list assignment L. (Having larger lists can only make finding a coloring easier, so we often assume equality.) The least k such that G is k-choosable is the *list chromatic number* of G, denoted $\chi^L(G)$. We define several decision problems related to graph coloring.

ListColoring
Input: A graph G with list assignment L.
Output: Does G admit an L-coloring?

k-ListColoring
Input: A graph G with k-list assignment L.
Output: Does G admit an L-coloring?

k-Coloring
Input: A graph G.
Output: Is G k-colorable?

k-Choosability
Input: A graph G.
Output: Is G k-choosable?

We note that k-Coloring reduces to k-ListColoring and that k-ListColoring reduces to ListColoring, but in general these problems do not reduce to k-Choosability (nor vice-versa). Indeed, we will see situations where the complexity of k-Choosability differs from that of k-ListColoring and ListColoring.

We list here trivial complexity results for these three problems. We assume familiarity with the complexity classes P and NP, as well as the theory of NP-hard and NP-complete problems. The problems k-Coloring, k-ListColoring, and ListColoring are clearly in NP. Erdős et al. [29], in characterizing 2-choosability, showed that 2-Choosability is in P. Likewise, it is possible to reduce 2-ListColoring to 2-SAT, and hence it also is in P. As k-Coloring is NP-hard for $k \geq 3$ [52], we therefore have that ListColoring is NP-hard and k-ListColoring is NP-hard for $k \geq 3$. The complexity of k-Choosability for larger k will be discussed in Chapter 6. Notably, k-Choosability is an example of a problem that may not be in NP.

Let G be a connected plane graph. A *coupled list assignment* L is a list assignment for the coupled graph $C(G)$. We define a *k-coupled list assignment* to be a coupled list assignment where all lists are of size exactly k. We say that G is *L-coupled choosable* if $C(G)$ is L-colorable, ignoring any loops in $C(G)$. (We ignore loops when coupled coloring, otherwise a graph with a bridge would have no coloring.) We say that G is *k-coupled choosable* if G is L-coupled choosable for any k-coupled list assignment L. The *coupled list chromatic number* $\chi_{vf}^{L}(G)$ is the smallest k such that G is k-coupled choosable. Since $C(G) = C(G^*)$ for the dual graph G^*, we therefore have $\chi_{vf}^{L}(G) = \chi_{vf}^{L}(G^*)$. We define some decision problems for coupled list coloring which are analogous to the previously defined problems for list coloring.

CoupledListColoring
Input: A plane graph G with coupled list assignment L.
Output: Does G admit an L-coupled coloring?

k-CoupledListColoring
Input: A plane graph G with k-coupled list assignment L.
Output: Does G admit an L-coupled coloring?

k-CoupledChoosability
Input: A plane graph G.
Output: Is G k-coupled choosable?

For small values of k, k-CoupledChoosability is in P. One can easily show that the only graph that is 2-coupled choosable is the graph consisting of a single vertex. (Recall that $\chi_{vf}^{L}(G)$ is only defined if G is connected.) Moreover, the only graphs that are 3-coupled choosable are trees (see also Lemma 5.1.1). Wang and Lih [90] proved that

all planar graphs are 7-coupled choosable, and hence k-CoupledListColoring and k-CoupledChoosability are in P for $k \geq 7$. Further discussion of the complexity of coupled list coloring is delayed to Chapter 6.

We conclude this section with a well known result for list coloring that will be useful later in this thesis, and also to help familiarize the reader with list coloring.

Theorem 2.3.1. *Let G be a graph with vertex order $v_1, \ldots, v_n$, and let $G_i := G[v_1, \ldots, v_i]$ for $i = 1, \ldots, n$. Let L be a list assignment for G such that $|L(v_i)| \geq d_{G_i}(v_i) + 1$ for $i = 1, \ldots, n$. Then G is L-colorable, and the coloring can be found in linear time.*

Proof. We proceed by induction on n. In the base case $n = 1$ and G consists of a single vertex x with $|L(x)| = 1$. Then trivially G is L-colorable.

Otherwise, $n > 1$. The vertex v_n has neighbors $u_1, \ldots, u_{d(v_n)}$ in $G = G_n$. Observe that the graph G_{n-1} with vertex ordering $v_1, \ldots, v_{n-1}$ satisfies the premise of the theorem, and so by the inductive hypothesis let ϕ be an L-coloring of G_{n-1}. We have now colored all vertices of G aside from v_n, and so it remains to find a valid color for v_n. We must find a color $c \in L(v_n)$ not used by any of the neighbors u_i. We have that

$$|L(v_n) \setminus \{\phi(u_1), \ldots, \phi(u_{d(v_n)})\}| \geq |L(v_n)| - d(v_n) = 1,$$

and hence such a color c can always be found. Therefore ϕ can be extended to an L-coloring of G, and G is L-colorable.

Following the steps of this proof leads to a linear time algorithm to find the coloring since we spend $O(d(v))$ time at each vertex v and $\sum_{v \in V(G)} d(v) = 2m$. □

We say that a graph G is *k-degenerate* if every subgraph of G has a vertex of degree at most k. As a corollary of Theorem 2.3.1, we have that a graph that is k-degenerate is $(k+1)$-choosable. To see this, we construct the necessary vertex order for a k-degenerate graph G by picking v_n to be a vertex of G of degree at most k, and then finding the rest of the vertex order inductively on the graph $G - v_n$. As 1-planar graphs are 7-degenerate [18], we also have the following corollary.

Corollary 2.3.2. *All 1-planar graphs are 8-choosable.*

2.4 Width Parameters

In preparation for results in Chapters 5 and 6, we review here two notions of width for a graph. The *treewidth* of a graph, informally, measures how close a graph is to a tree. The treewidth of a graph is defined via its *tree decomposition*.

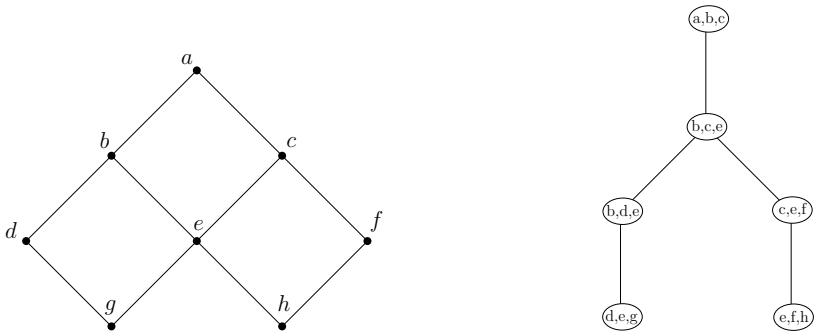


Figure 2.6: A graph G (left) with a tree decomposition (right) of width two. One confirms that the tree decomposition follows the three rules of Definition 2.4.1. As G is not a tree, we know that $tw(G) = 2$.

Definition 2.4.1. Let G be a graph. A *tree decomposition* of G is a pair $\langle \mathcal{T}, X \rangle$, where $\mathcal{T}$ is a tree, $X : V(T) \to 2^{V(G)}$ maps each vertex of $\mathcal{T}$ to a set of vertices of G, and

- (*vertex coverage*) for every vertex x of G, there is a node t of $\mathcal{T}$ such that $x \in X(t)$,
- (*edge coverage*) for every edge xy of G, there is a node t of $\mathcal{T}$ such that $x, y \in X(t)$, and
- (*connectivity condition*) for nodes r, s, t of T, if s lies on the unique path from r to t in $\mathcal{T}$, then $X(r) \cap X(t) \subseteq X(s)$.

The set $X(t)$ is referred to as the *bag* of t. The *width* of a tree decomposition $\langle \mathcal{T}, X \rangle$ is $\max_{t \in V(\mathcal{T})} |X(t)| - 1$. The *treewidth* $tw(G)$ of a graph G is the minimum width of any tree decomposition of G. See Figure 2.6 for an example of a tree decomposition.

A graph of treewidth at most k is known as a *partial k-tree*. We note that the class of partial 1-trees is exactly the forests, the class of partial 2-trees is exactly the class of K_4-minor free graphs, and that a graph that is a cycle has treewidth exactly two.

We also note here that subdividing the edges of a graph cannot increase the treewidth. For trees this is clear, as a subdivision of a tree is still a tree. Otherwise, let G be a graph with a tree decomposition $\langle \mathcal{T}, X \rangle$, and let G' be obtained from G by subdividing an edge $xy \in E(G)$ into two edges xz and zy, where z is new a vertex not present in G. We construct a tree decomposition $\langle \mathcal{T}', X' \rangle$ of G' with the same width as $\langle \mathcal{T}, X \rangle$. By edge coverage, $\mathcal{T}$ has a node t with $x, y \in X(t)$. Add a new node t' adjacent to t with $X'(t') := \{x, y, v\}$. It is not hard to see that this is a tree decomposition of G', and moreover it has the same width as $\langle \mathcal{T}, X \rangle$, since G has treewidth at least two.

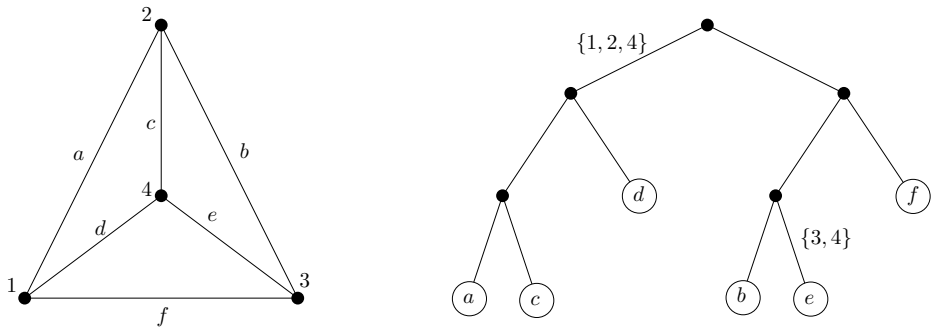


Figure 2.7: The graph K_4 (left) with a branch decomposition (right). Two arcs in the branch decomposition have been labeled with their respective separator. This branch decomposition demonstrates that K_4 has branchwidth at most three. (In fact, it is exactly three.)

We now define the *branchwidth* of a graph, which is defined by recursively partitioning the edge set of a graph.

Definition 2.4.2. A *recursive partition* of a non-empty set A is defined recursively. In the base case, A contains exactly one element. Otherwise, the recursive partition of A is a partition into two non-empty sets A_1, A_2 along with a recursive partition of A_1 and A_2.

A recursive partition can be naturally represented by a rooted binary tree $\mathcal{T}$: if A has only one element, then $\mathcal{T}$ consists of only a leaf storing that one element. Otherwise, $\mathcal{T}$ consists of a node with two children that represent the recursive partitions of A_1 and A_2. In this way, we have a rooted binary tree with a bijection from the leaves to the set A.

Definition 2.4.3. A *branch decomposition* of a graph G is a recursive partition $\mathcal{T}$ of $E(G)$.

Let G be a graph and $\mathcal{T}$ a branch decomposition of G; each leaf of $\mathcal{T}$ is mapped to a unique edge of G. Let α be an arc of $\mathcal{T}$. (We use *arcs* of $\mathcal{T}$ to disambiguate from *edges* of G.) Observe that $\mathcal{T} - \alpha$ has two connected components. The *separator* $\sigma(\alpha)$ defined by α is the set of vertices x which are incident to edges contained in leaves of both of the components of $\mathcal{T} - \alpha$. The *width* of $\mathcal{T}$ is the size of the largest separator of any arc of $\mathcal{T}$. The *branchwidth* $bw(G)$ of a graph G is the minimum width of any branch decomposition of G. See Figure 2.7 for an example of a branch decomposition.

A *star graph* is a graph with at most one vertex of degree at most two. A graph G has branchwidth one if and only if every connected component of G is a star graph. The graphs of branchwidth at most two are the K_4-minor free graphs. In particular, trees have

branchwidth at most two [76]. It is also known that the branchwidth and treewidth of a graph are closely related:

Lemma 2.4.4 (Robertson and Seymour [76])**.** *Let G be a graph with $bw(G) > 1$. Then $bw(G) - 1 \leq tw(G) \leq \frac{3}{2}bw(G) - 1$.*

Chapter 3

4-List Coloring 1-Planar Bipartite Graphs in Linear Time

In this chapter, we show that 1-planar bipartite graphs are 4-choosable, and then present an algorithm to find this coloring in linear time. We further show that this result is tight, and moreover that 3-list coloring 1-planar bipartite graphs is hard. The work in this chapter, particularly the algorithm presented in Section 3.2.2, is based on an unpublished manuscript by Biedl, Lubiw, and Merkel [15]. We have greatly simplified the description of the algorithm, and improved the run-time.

We first review some known results for list coloring bipartite graphs. Erdős et al. [29] characterized the 2-choosable graphs; these graphs are all planar and bipartite. They and Vizing [86] demonstrated that the list chromatic number of bipartite graphs is unbounded. Alon and Tarsi [3] proved that planar bipartite graphs are 3-choosable. Gutner [42] studied a generalization of list coloring on directed graphs with no odd directed cycles, and proved hardness results for k-Choosability for bipartite graphs. Biedl et al. [15] proved that the 3-list coloring of a planar bipartite graph can be found in linear time, and also provide an algorithm for 3-list coloring toroidal bipartite graphs (bipartite graphs that can be drawn on a torus without crossings) in time $O(n^{1.5})$. Campos et al. [21] studied various coloring problems on bipartite graphs of small diameter, and showed that k-ListColoring, $k \geq 3$, is NP-complete on complete bipartite graphs. There are also some results for problems related to list coloring: Esperet [30] studied the dynamic list coloring problem on bipartite graphs, and Galvin [14] studied the problem of list coloring the edges of bipartite multigraphs.

One technique used for list coloring is to repeatedly color a so-called kernel of a directed

graph (see e.g. [3, 42]). We modify here the definition of a kernel slightly (calling it a *pseudo-kernel*) because this will simplify some proofs. We need some definitions. Let D be a directed graph. A set of vertices $K \subseteq V(D)$ is called *pseudo-absorbing* if for every vertex $x \in V(D) \setminus K$, either there is an edge xy for some $y \in K$, or x is not adjacent to any vertex in K. A *pseudo-kernel* is a non-empty vertex set K such that K is both pseudo-absorbing and an independent set.[1]

Let D be a directed graph and $x \in V(D)$. Recall that the outdegree of x (denoted $d^+(x)$) is the number of edges $xy \in E(D)$. Let L be a list assignment for D and let c be a color present in some list of L. The *colored outdegree with respect to c of x*, denoted $d_c^+(x)$, is the number of edges xy such that $c \in L(y)$. We say that x is a *colored sink with respect to c* if $c \in L(x)$ and $d_c^+(x) = 0$.

In this chapter, we assume that the input graph is simple, since there are no loops in a bipartite graph, and we can eliminate parallel edges without affecting colorability.

3.1 All 1-Planar Bipartite Graphs are 4-List Colorable

Lemma 3.1.1 was known to Alon and Tarsi [3] (attributed to Bondy, Boppana, and Siegel). We recreate the proof here both to demonstrate the algorithmic properties and to restate the result for pseudo-kernels.

Lemma 3.1.1. *Let D be a directed graph and let L be a list assignment for D such that $|L(x)| \geq d^+(x) + 1$ for every vertex $x \in V(D)$. If every induced subgraph of D has a pseudo-kernel, then D is L-colorable.*

Proof. The proof proceeds by induction on n. If $n = 1$, then trivially D is L-colorable.

Otherwise, let $\mathcal{L} := \cup_{x \in V(D)} L(x)$ be the set of all possible colors. Pick some color $c \in \mathcal{L}$. By hypothesis, the induced subgraph $D[L^{-1}(c)]$ has some pseudo-kernel K_c. Color every vertex of K_c with c. This is feasible because every vertex in K_c has the color c in its list and because by definition the vertices of a pseudo-kernel are pairwise not adjacent. Let $D' := D \setminus K_c$ be the graph of vertices which remain to be colored, and define a new list assignment

$$L'(x) := \begin{cases} L(x) \setminus \{c\} & \text{if } x \text{ has a neighbor in } K_c, \\ L(x) & \text{otherwise.} \end{cases}$$

[1] The statements "a directed graph D has a kernel" and "a directed graph D has a pseudo-kernel" are in fact equivalent; this generalization allows us to eventually more easily state and prove our algorithm.

Only vertices in $L^{-1}(c) \setminus K_c$ have had a color removed from their list, but because K_c is pseudo-absorbing in $D[L^{-1}(c)]$, any vertex which had c removed from its list must have had an edge to a neighbor in K_c, and had its outdegree decreased by 1. Hence we can apply induction on D' with the list assignment L' and get an L'-coloring for D'. As no vertex adjacent to K_c uses the color c, this can be extended to an L-coloring of D. □

One can easily show that every directed bipartite graph has a pseudo-kernel (and therefore that any induced subgraph of a directed bipartite graph has a pseudo-kernel), and so Lemma 3.1.1 applies to directed bipartite graphs. (Note that this is also a consequence of Richardson's theorem (see e.g. [12]) which states that any directed graph with no odd directed cycles has a kernel.)

Lemma 3.1.2. *Every directed bipartite graph D has a pseudo-kernel.*

Proof. Let A, B be the two partitions of D. If A contains a sink v, then take v to be the pseudo-kernel of D: Trivially a single vertex is an independent set, and as v is a sink, every vertex x adjacent to v is adjacent via a directed edge xv. Otherwise, every vertex in A has an edge into B, and we take the set B to be the pseudo-kernel of D. □

Lemmas 3.1.1 and 3.1.2 give the following list coloring result for directed bipartite graphs:

Theorem 3.1.3. *Let D be a directed bipartite graph and L a list coloring for D such that $|L(x)| \geq d^+(x) + 1$ for every vertex x of D. Then D is L-colorable.*

This also implies the following corollary:

Corollary 3.1.4. *Let D be a directed bipartite graph with maximum outdegree Δ^+. Then D is $\Delta^+ + 1$-choosable.*

Moreover, by following the proofs of Lemmas 3.1.1 and 3.1.2, we have the following algorithm for list coloring directed bipartite graphs:

Algorithm 3.1.5. *Let D be a directed bipartite graph with partition A, B, and let L be a list assignment for D. For each color c present in some list in L,*

- *while there is a colored sink x with respect to c in A, color x with c, remove c from the lists of all neighbors of x, and delete x, then*

- *color all vertices in $B \cap L^{-1}(c)$ with the color c, delete these vertices from D, then remove c from the lists of all vertices in A.*

If $|L(x)| \geq d(x) + 1$ for every vertex x of the graph D, then Algorithm 3.1.5 always succeeds. Thus, in order to find a list coloring for a bipartite graph, we wish to orient the edges of a bipartite graph such that every vertex has a suitably small outdegree. The following lemma is a special case of a known result, proved independently several times [25, 34, 43].

Lemma 3.1.6. *For some given $k \in \mathbb{N}$, a graph G has an orientation such that $d^+(x) \leq k$ for every $x \in V(G)$ if and only if for every subgraph $H \subseteq G$, we have $|E(H)| \leq k|V(H)|$.*

Others have studied how quickly this orientation can be found. Venkateswaran [85] gave in $O(m^2)$ time algorithm, and Blumenstock [17] improved this to $O(m^{1.5}\sqrt{\log \log k})$ time. We will not review these algorithms here since (as discussed later) we can find the orientation more efficiently for 1-planar bipartite graphs.

By Lemma 3.1.6, any graph with low edge density for all subgraphs can have its edges oriented to have small outdegree. While bipartite graphs in general have unbounded edge density, for 1-planar bipartite graphs the edge density is bounded.

Lemma 3.1.7 (Karpov [53])**.** *Every 1-planar bipartite graph with $n \geq 4$ vertices has at most $3n - 8$ edges.*

With this, we have the desired coloring for 1-planar bipartite graphs.

Theorem 3.1.8. *Every 1-planar bipartite graph is 4-choosable.*

Proof. Let G be a 1-planar bipartite graph. By Lemmas 3.1.6 and 3.1.7, G has an orientation such that every vertex has outdegree at most 3. Then by Corollary 3.1.4, G is 4-choosable. □

3.2 Finding the 4-List Coloring Efficiently

We now wish to efficiently find the 4-list coloring of a 1-planar bipartite graph, in particular in linear time. The proof of Theorem 3.1.8 does lead to an algorithm for finding a 4-list coloring:

Algorithm 3.2.1. *Let G be a 1-planar bipartite graph and let L be a 4-list assignment for G. First, orient the edges of G such that every vertex has outdegree at most three, then use Algorithm 3.1.5 to find an L-coloring of G.*

This algorithm is already polynomial time:

- The orientation can be found in time $O(m^{1.5})$ since $k = 4$ [17].
- Given the orientation, a feasible coloring can be found in polynomial time by using of Algorithm 3.1.5.

In pursuit of a linear time algorithm, we need to be able to orient the edges of a 1-planar bipartite graph such that any vertex has maximum outdegree three in linear time. Moreover, it is not clear how one would implement Algorithm 3.1.5 in linear time. Hence, more work is required to establish the desired run-time.

3.2.1 Orienting the Edges

It is easy to see that a forest can be oriented to have maximum outdegree one in linear time by choosing a root for each tree, traversing down each tree and orienting each edge towards the root of the tree. Likewise, if we can partition the edges of graph G into k forests, we can use this partition to orient the edges of G in linear time such that every vertex has outdegree at most k. The minimum number of forests that a graph G can be partitioned into is the *arboricity* of G. The Nash-Williams theorem states that the arboricity of a graph is directly correlated with the edge density of all subgraphs of a graph.

Theorem 3.2.2 (Nash-Williams [65])**.** *A graph G has arboricity k if and only if for every nonempty subgraph $H \subseteq G$, $|E(H)| \leq k\,(|V(H)|-1)$.*

Note the similarity between this theorem and Lemma 3.1.6. By the Nash-Williams theorem and Lemma 3.1.7, a 1-planar bipartite graph can be partitioned into three trees. It is known that the partition of the Nash-Williams theorem can be found in polynomial time [35], but in general a linear time algorithm is not known. However, it is known that a planar bipartite graph can be partitioned into two forests in linear time [14]. Hence, if we could partition a 1-planar bipartite graph into a planar (bipartite) graph and a forest, then we would have our split into three trees in linear time. Ackerman [1] proved that such a partition exists for any 1-planar graph, but doing it in linear time turns out to be

non-trivial. In Chapter 4, we will reprove Ackerman's result that a 1-planar graph can be split into a planar graph and a forest, and obtain a linear time algorithm from our new proof. Therefore, using the result from Chapter 4, we obtain:[2]

Theorem 3.2.3. *Let G be a 1-planar bipartite graph. The edges of G can be oriented such that every vertex has outdegree at most 3, and this orientation can be found in linear time.*

3.2.2 $(\Delta^+{+}1)$-List Coloring Bipartite Graphs Efficiently

We now know that, given a 1-planar bipartite graph, we can in linear time orient the edges such that every vertex has outdegree at most three. It remains to show that we can then use this orientation to find a 4-list coloring in linear time. As discussed above, although Algorithm 3.1.5 would lead to a polynomial time algorithm, it is not obviously linear. To make it a linear time algorithm, we need to be able to repeatedly find colored sinks with respect to a color c and find the vertices $B \cap L^{-1}(c)$ for some color c in amortized linear time, where B is a partition of the bipartite graph. We show here how to do this, and in fact, our algorithm works for any directed bipartite graph D with color lists that are longer than the outdegree. We present an algorithm for finding such a list coloring of a directed bipartite graph D in time $O(\Delta^+ \cdot m)$, where Δ^+ is the maximum outdegree of D. (This improves slightly on the run-time of $O((\Delta^+)^2 m)$ presented in [15].) When the algorithm is applied to 4-list coloring 1-planar bipartite graphs, we will have $\Delta^+ \leq 3$ and hence the algorithm will have run-time $O(m)$.

Let D be a directed bipartite graph with maximum outdegree Δ^+. Let A, B be the partition of D, and let L be a list assignment for D such that $|L(x)| \geq d^+(x) + 1$ for every vertex x of D. Let $\mathcal{L} := \cup_{x \in V(D)} L(x)$ be the set of all colors used in L. For every vertex x of D we assume that $|L(x)| = d^+(x) + 1$ (decreasing the size of the lists can only make finding a coloring more difficult), and we therefore have

$$|\mathcal{L}| = \left| \bigcup_{x \in V(D)} L(x) \right| \leq \sum_{x \in V(D)} (d^+(x) + 1) = n + m.$$

We assume that the colors are stored as positive integers, and that none of these integers is larger than $|\mathcal{L}|$.

Our crucial idea for small run-time is to parse always the smallest color c present in any list and to do appropriate pre-sorting. Namely, we construct an array `ColorPairs` of

[2] Chapter 4 does not use any results from this chapter (and in particular does not reference Theorem 3.2.3), so there is no issue of circular reasoning.

pairs $\langle x, b\rangle$ for each vertex x and color $b \in L(x)$, and sort all these pairs by their second element. This will make it very easy to find the set $L^{-1}(c)$, since these correspond to the first entries in `ColorPairs`. (For later colors it will also be easy using suitable updates; see below.) We will also need to be able to test whether $c \in L(x)$ for a vertex x in constant time. We therefore sort all of the lists $L(x)$ for each vertex x. This allows us to check in constant time whether $c \in L(x)$ by only looking at the first element in the list.

We show how to do this preprocessing in linear time. We first construct the unsorted array `ColorPairs`. By the same analysis used to achieve the upper bound on $|\mathcal{L}|$, the size of `ColorPairs` is at most $n + m$. We then use bucket sort with $|\mathcal{L}|$ buckets to sort all of these pairs by their second element. This can be done in time $O(|\mathcal{L}| + |\texttt{ColorPairs}|) = O(m + n) = O(m)$. After sorting, we make one pass through `ColorPairs` in order to sort all of the lists $L(x)$. See Procedure 1 for details.

Procedure 1: Initialize(D, L)

Result: Initialize and sort `ColorPairs`, and sort the list of every vertex

```
ColorPairs := []
// Initialize and sort ColorPairs
for vertex x ∈ V(D) do
    for color c ∈ L(x) do
        ColorPairs.add(⟨x, c⟩)
BucketSort(ColorPairs)
// Re-initialize all lists, then pass through ColorPairs to sort
   them
for vertex x ∈ V(D) do
    L(x) := []
for color pair ⟨x, c⟩ ∈ ColorPairs do
    L(x).add(c)
```

Again letting c be the smallest color present in any list, Procedure 2 shows the process for coloring the set of vertices $V_c := L^{-1}(c)$, following Algorithm 3.1.5. We first calculate and store the colored outdegree of every vertex in $V_c \cap A$. All vertices where this colored outdegree is 0 are colored sinks with respect to c, and we add these colored sinks to a queue. We iterate through this queue. For each vertex x in the queue, we assign the color c to x. After x is colored, we traverse all edges yx going into x. Presuming $c \in L(y)$, we remove c from the list of y, and then traverse all edges zy going into y. Because $x \in A$, we have that $y \in B$ and $z \in A$, so if $c \in L(z)$ then we had computed the colored outdegree $d_c^+(z)$.

Since we removed c from the list of y, this colored outdegree decreases, so we update the stored value $d_c^+(z)$. If z becomes a colored sink, we add it to the queue. We then delete x from the graph. Once there are no more colored sinks in A (and thus the queue is empty), we loop over all vertices v in V_c that have not been deleted or had c removed from their list. If v is in B, we assign the color c to v. Otherwise, v is in A, and we remove the color c from $L(v)$. Note that we specifically do not update `ColorPairs` during Procedure 2.

Lastly, Procedure 3 demonstrates the entire procedure for $(\Delta^+{+}1)$-list coloring a bipartite graph. After initialization, we loop over the `ColorPairs` array. We repeatedly collect all vertices that have not been deleted and have the color c in their list, where c is at that moment the smallest color, and color those vertices with Procedure 2. It is possible that a vertex in `ColorPairs` was deleted in a previous call to Procedure 2; when we encounter such a vertex we simply skip over it. We continue this until we have made one pass through `ColorPairs`. Procedure 3 follows Algorithm 3.1.5, and so it will always find a valid coloring. It remains to justify the claimed run-time.

Runtime. As previously argued, Procedure 1 takes time $O(m)$. Ignoring the calls to Procedure 2, Procedure 3 makes one pass over `ColorPairs`, which can be done in time $O(m)$. We now analyze the run time of Procedure 2 across all calls. We first observe that a vertex v will appear in a set V_c for some color c at most $|L(v)| = d^+(v)+1$ times. Therefore the last loop of Procedure 2 (lines 19–24), ignoring the time to delete any vertices or edges (since an element can be deleted at most once), in total takes time $O(m)$. In the first loop (lines 2–6), calculating $d_c^+(x)$ can be done in time $O(d^+(x))$: First set $d_c^+(x)$ to zero, then loop over all outgoing edges xy and increment $d_c^+(x)$ if $c \in L(y)$. Because c is the smallest color and all the lists are sorted, $c \in L(y)$ can be tested in $O(1)$ time. Thus, across all calls to Procedure 2, the first loop will take time proportional to

$$\sum_{x \in V(D)} (d^+(x))^2 \leq \Delta^+ \cdot \sum_{x \in V(D)} d^+(x) = \Delta^+ \cdot m.$$

Lastly, we consider the run-time of the loop where we color all of the colored sinks (lines 7–18) over all executions of Procedure 2. A vertex in A can be a colored sink at most once because it then gets colored immediately, and hence the loop of line 7 will run at most n times. The middle loop (lines 10–16) runs at most m times, since the edges we iterate over will be deleted when the sink is deleted from the graph. Lastly, when an edge zy is iterated over in the innermost loop (lines 12–16), we know that a color has just been removed from $L(y)$. Hence, this edge will appear in this loop at most $d^+(y) + 1 \leq \Delta^+{+}1$ times across the entire algorithm. Therefore, the innermost loop runs $O(\Delta^+ \cdot m)$ times

Procedure 2: Color(V_c, c)

Result: Color some vertices of V_c with the color c, following Algorithm 3.1.5.

```
1  initialize a queue q
2  for vertex x ∈ V_c do
3      if x ∈ A then
4          calculate and store d_c^+(x)
5          if d_c^+(x) == 0 then
6              q.add(x)
7  while q is not empty do
8      x := q.pop()
9      color x with c
10     for edge yx ∈ E(D) do
11         if c ∈ L(y) then
12             remove c from L(y)
13             for edge zy ∈ E(D) do
14                 if c ∈ L(z) then
15                     d_c^+(z) := d_c^+(z) − 1
16                     if d_c^+(z) == 0 then
17                         q.add(z)
18     delete x from D
19 for vertex v ∈ V_c do
20     if c ∈ L(z) and v has not been deleted then
21         if v ∈ A then
22             remove c from L(v)
23         else
24             color v with c
25             delete v from D
```

Procedure 3: DirectedBipartiteListColoring(D, L)

Result: L-list color a directed bipartite graph D.

```
Initialize (D, L)
i := 0
while i ≤ |ColorPairs| do
    ⟨_, c⟩ := ColorPairs[i]                    // retrieve the smallest color
    V_c := ∅
    loop
        ⟨x, c'⟩ := ColorPairs[i]
        if c' == c and x has not been deleted then
            V_c.add(x)
            i := i + 1
        else
            Color(V_c, c)
            break
```

across the whole algorithm. Thus, the claimed run-time has been proven, and we have the following theorem.

Theorem 3.2.4. *Let D be a directed bipartite graph with maximum outdegree Δ^+ and L be a list assignment for D such that $|L(x)| \geq d^+(x) + 1$ for every vertex x of D. For any such input,* DirectedBipartiteListColoring(D, L) *computes an L-coloring of D in $O(\Delta^+ \cdot m)$ time.*

This implies one of the main results of this chapter.

Theorem 3.2.5. *There is a linear time algorithm that, for any graph G with 4-list assignment L, computes an L-coloring of G in linear time.*

As mentioned above, our run-time of $O(\Delta^+ \cdot m)$ in Theorem 3.2.4 is an improvement on a previously known algorithm with run-time $O((\Delta^+)^2 \cdot m)$ [15]. It is unknown whether this could further be improved to $O(m)$, i.e., whether there is an algorithm for this list coloring whose runtime does not depend on Δ^+. Such an improvement would likely require a different approach than we take: Our algorithm is bottlenecked by Procedure 2, where any edge e of D is touched $O(\Delta^+)$ times. (In fact, it is possible to construct adversarial inputs where $\Omega(m)$ edges are touched $\Omega(\Delta^+)$ times.)

3.3 Complexity of 3-List Coloring 1-Planar Bipartite Graphs

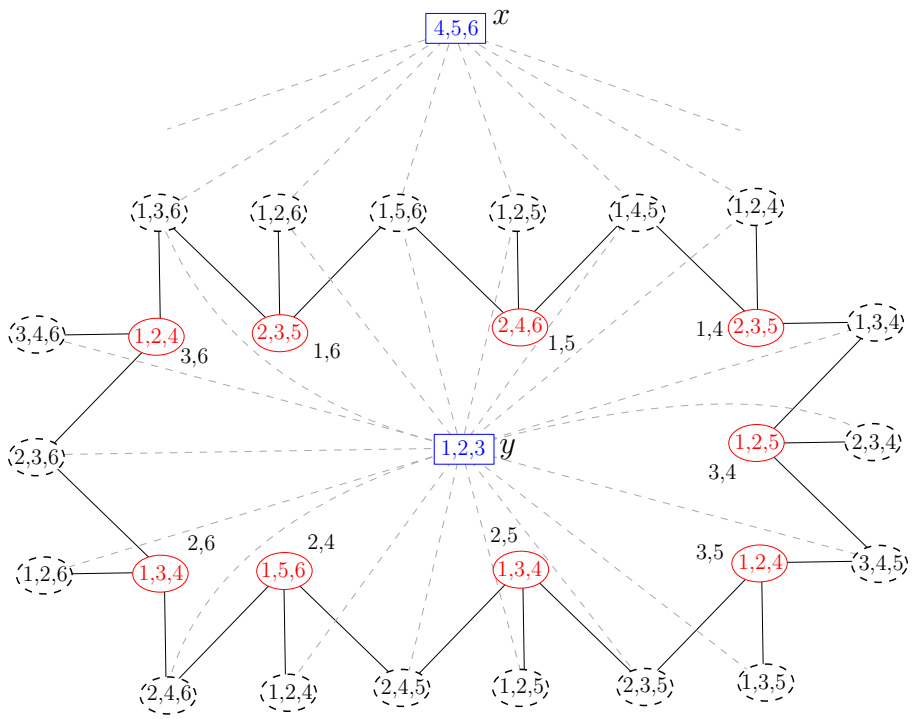


Figure 3.1: A 1-planar bipartite graph that is not 3-choosable. The numbers on each vertex represent the list assignment L for each vertex. The red (solid ellipse) vertices and blue (rectangular) vertices form one part of the bipartite graph, and the black (dashed ellipse) vertices form the other part. The blue (rectangular) vertices, labeled x and y, are connected to all of the black (dashed ellipse) vertices. Next to each red (solid ellipse) vertex we have labelled the pair of colors for x and y that that vertex blocks.

In general, our proved upper bound of 4-choosability is tight for 1-planar bipartite graphs.

Theorem 3.3.1. *There exists a 1-planar bipartite graph with 29 vertices that is not 3-choosable.*

Proof. Let G be the graph with list assignment L seen in Figure 3.1. From the given drawing we can see that G is 1-planar. Let the vertices x and y be as labeled in Figure 3.1.

Each red (solid ellipse) vertex prevents some choice of colors for x and y: For each pair of colors $a \in L(x)$ and $b \in L(y)$, there is a red vertex $r_{a,b}$ that has list $\{\alpha, \beta, \gamma\}$ that is disjoint from $\{a, b\}$. The vertex $r_{a,b}$ has three neighbors, and these neighbors are black (dashed ellipse) vertices. Moreover, these three neighbors are all adjacent to both x and y and have lists $\{\alpha, a, b\}$, $\{\beta, a, b\}$, and $\{\gamma, a, b\}$. As we have used the colors a and b for x and y, we are forced to use the colors α, β, γ for the three neighbors of $r_{a,b}$. But then there is no feasible choice of color for $r_{a,b}$, and therefore no feasible coloring of G uses a and b for x and y.

We have labelled each red vertex in Figure 3.1 with the pair of colors that it blocks; one confirms that each red vertex blocks this pair of colors as described, and that there is a red vertex for each of the $3 \times 3 = 9$ pairs of colors for x and y. □

We now show that 3-list coloring 1-planar bipartite graphs is hard in general. To do so, we give a few lemmas that will be needed both for Theorem 3.3.5 and in Chapter 6. From the construction in Theorem 3.3.1, we have the following result.

Lemma 3.3.2. *For any color r, there exists a 1-plane bipartite graph H_r with a fixed vertex x on the outer face and a 3-list assignment L_r such that H_r is L_r-colorable, and any L_r-coloring of H_r must assign r to x.*

Proof. By renaming, we may assume that $r \notin \{1, \ldots, 6\}$. Let G be the graph with list assignment L seen in Figure 3.1. Recall that G is not L-colorable. Let $H_r := G$, and construct L_r from L by replacing "6" by "r" in the list of x. To see that H_r is L_r-colorable, pick r for x and an arbitrary color for y. After choosing these colors, every other vertex in the graph has at least two colors available, as no other vertex in the graph has r in its list. Let $H' := H_r - x - y$. We argue that H' is 2-choosable, and do so by finding an orientation of H' with maximum outdegree one. By Corollary 3.1.4, with such an orientation we will know that H' is 2-choosable. (It is also possible to verify the 2-choosability of H using the characterization of 2-choosability [29].) This graph H' is the graph of red and black (solid and dashed ellipse) vertices. Observe that this graph is an even cycle along with some vertices of degree one. For each vertex v of degree one, orient the unique edge e incident to v such that e goes out of v; now v has outdegree one. Then we orient the edges of the even cycle in the clockwise direction such that every vertex in the even cycle has outdegree exactly one. This gives the desired orientation of H'. Therefore, an L_r-coloring of H_r can be found. Note that if we did not choose color r for x, and instead used 4 or 5, then no coloring is possible, since this would be an L-coloring of G. □

Lemma 3.3.3. *Let G be a graph, L be a list assignment for G, and $e = uv$ be an edge of G such that $|L(u)| = 2$. Let G' be the graph obtained from G by subdividing e twice and L' be the list assignment for G' given by*

$$L'(x) := \begin{cases} L(x) & x \in V(G) \\ L(u) & \text{otherwise} \end{cases}.$$

Then G is L-colorable if and only if G' is L'-colorable.

Proof. Let $\{a, b\} := L(u)$. Let u', u'' be the new vertices resulting from the two subdivisions of e, with edges uu', $u'u''$, $u''v$. Note that these are the only edges incident to u' and u'', and that $L(u') = L(u'') = \{a, b\}$. (See Figure 3.2.)

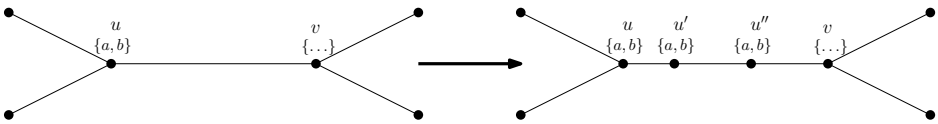


Figure 3.2: The construction in Lemma 3.3.3.

($\Rightarrow$) Let ϕ be an L-coloring of G. We construct an L'-coloring ϕ' of G'. Assume without loss of generality that $\phi(u) = a$. For vertices $x \in V(G) \setminus \{u', u''\}$, we choose $\phi'(x) := \phi(x)$. By the edge uu', we are forced to pick $\phi'(u') := b$. Similarly, we are forced to pick $\phi'(u'') := a$. Since ϕ is a valid coloring of G, and u and v are adjacent in G, we know that $\phi(v) \neq a$. Hence ϕ' is a valid L'-coloring of G'.

($\Leftarrow$) Let ϕ' be an L'-coloring of G'. We construct an L-coloring ϕ of G. Assume without loss of generality that $\phi'(u) = a$. For vertices $x \in V(G)$, we define $\phi(x) := \phi'(x)$. By the edges $uu', u'u'', u''v$ of G', it must be that $\phi'(u') = b$ and $\phi'(u'') = a$. Hence $\phi'(v) \neq a$, and ϕ is a valid L-coloring of G. □

Lemma 3.3.4. *Let G be a graph and L a list assignment for G such that for every edge $e = uv$ of G, at least one endpoint x of e satisfies $|L(x)| = 2$. Then there exists a 1-planar graph G' that is a subdivision of G and has a list assignment L' such that G is L-colorable if and only if G' is L'-colorable. The graph G' is obtained by subdividing edges of G an even number of times and can be constructed in polynomial time. For a vertex $x \in V(G')$, if $x \in V(G)$ then $L'(x) = L(x)$, and otherwise $|L'(x)| = 2$.*

Proof. Fix a good drawing Γ of G in the plane. Let e be an edge of G. Suppose that e intersects $k \geq 2$ edges in the drawing Γ. Since e is incident with a vertex with list of size

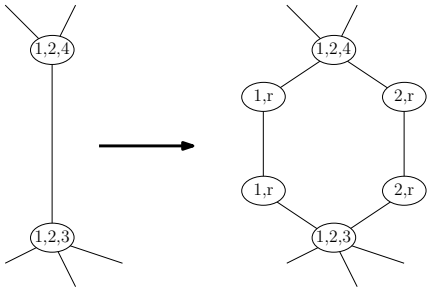


Figure 3.3: Transformation applied to each edge in the reduction of Theorem 3.3.5.

two, we can apply Lemma 3.3.3 to e for $\lfloor \frac{k}{2} \rfloor$ times, drawing each new edge resulting from the subdivisions such that it intersects at most one other edge. Repeat this process for every edge of G. By construction the resulting graph G' is 1-planar, and by Lemma 3.3.3, it is colorable if and only if G is colorable.

Since Γ is a good drawing, any two edges of G intersect at most once in Γ. Hence G has $O(m^2)$ crossings, and we perform $O(m^2) = O(n^4)$ subdivisions to construct G'. Therefore G' can be constructed in polynomial time. □

Theorem 3.3.5. 3-ListColoring *is NP-hard for 1-planar bipartite graphs.*

Proof. We reduce from 3-ListColoring complete bipartite graphs $K_{n,n}$. This problem is known to be NP-hard [21]. Let G be a complete bipartite graph and L a 3-list assignment for G. We construct a 1-planar bipartite graph $\hat{G}$ with 3-list assignment $\hat{L}$ such that G is L-colorable if and only if $\hat{G}$ is $\hat{L}$-colorable, and first have two intermediate constructions.

We first construct a bipartite graph G' with list assignment L'. Let r be a color not present in any list of G. We begin by adding every vertex x of G to G', with $L'(x) := L(x)$. Let x, y be adjacent vertices of G. For every color $c \in L(x) \cap L(y)$, we add two vertices u and u' and edges xu, uu', and $u'y$. Define $L(u) := L(u') := \{c, r\}$. (See Figure 3.3.) Since G is bipartite, G' is also bipartite. However, at this time G' is not necessarily 1-planar, and not every vertex has a list of size exactly three. We will later amend this.

We now show that G is L-colorable if and only if G' is L'-colorable. Let ϕ be an L-coloring for G. We construct an L'-coloring ϕ' of G'. We first define $\phi'(x) := \phi(x)$ for every vertex x of G. Let x, y be two adjacent vertices of G. We must assign colors to the paths between x and y added due to colors in $L(x) \cap L(y)$. If $L(x) \cap L(y) = \emptyset$, we are done. Otherwise, let $u, u' \in V(G')$ such that $xuu'y$ is a path in G' and $L(u) = L(u') = \{c, r\}$ where $c \in L(x) \cap L(y)$. If $\phi(x) \neq c$ and $\phi(y) \neq c$, then u and u' can be colored with c and r. Otherwise, suppose $\phi(x) = c$. Since ϕ is a proper coloring of G, we know that $\phi(x) \neq \phi(y)$,

and hence $\phi(y) \neq c$. We thus define $\phi'(u) := r$ and $\phi'(u') := c$ and obtain an L'-coloring of G'.

Let ϕ' be an L'-coloring of G'. For every vertex x of G, we define $\phi(x) := \phi'(x)$. Let x, y be adjacent vertices of G. We wish to show that ϕ' assigns different colors to x and y. If $\phi'(x) \notin L(x) \cap L(y)$, then clearly $\phi'(x) \neq \phi'(y)$. Otherwise, suppose $\phi'(x) = c \in L(x) \cap L(y)$. Then there are vertices $u, u' \in V(G')$ such that $xuu'y$ is a path in G' and $L(u) = L(u') = \{c, r\}$. As ϕ' is a proper coloring, it must be that $\phi'(u) = r$ and $\phi'(u') = c$. We therefore know that $\phi'(y) \neq c$, and so $\phi'(x) \neq \phi'(y)$. In either case, ϕ' assigns different colors to x and y, and so ϕ is a valid L-coloring of G.

From G' and L' we construct a 1-planar bipartite graph G'' with list assignment L''. By construction, every edge of G' is incident to a vertex with a list of size two. We therefore apply Lemma 3.3.4 to obtain a 1-planar graph G'' with list assignment L'' such that G' is L'-colorable if and only if G'' is L''-colorable. Recall that the construction of Lemma 3.3.4 works by subdividing edges an even number of times, and hence G'' is also bipartite. Moreover, the size of lists are not increased, and hence every list in L'' is still of size two or three.

Lastly, we augment G'' and L'' into a 1-planar bipartite graph $\hat{G}$ with 3-list assignment $\hat{L}$. Let s be a color not present in any list of G''. Let u be a vertex of G'' with a list of size two. We add s to $L'(u)$. We then add a copy of the graph H_s with list assignment L_s from Lemma 3.3.2, adding an edge from u to the vertex of H_s that must be colored with s. Doing this preserves the 1-planarity and bipartiteness of the graph. Moreover, by the forced coloring of each copy of H_s, the L-colorability is preserved.

All transformations of graphs and lists can be done in polynomial time, so this proves NP-hardness as desired. □

In summary, we know that 1-planar bipartite graphs are 4-choosable, and that the coloring can be found in linear time. In contrast, finding a 3-list coloring of a 1-planar bipartite graph is NP-hard. We end this chapter by observing a curious coincidence between the complexity of k-Coloring planar graphs and k-ListColoring 1-planar bipartite graphs, demonstrated in Table 3.1.

While the fastest known algorithm for finding a 4-coloring of a planar graph runs in time $O(n^2)$ [74], we are able to find the 4-list coloring of a 1-planar bipartite graph in linear time.

k	Planar k-Coloring	1-Planar Bipartite k-ListColoring
2	P	P
3	NP-hard [37]	NP-hard (Thm. 3.3.5)
4	Always Possible [5]	Always Possible (Thm. 3.1.8)

Table 3.1: The complexity of k-Coloring for planar graphs compared to the complexity of k-ListColoring for 1-planar bipartite graphs.

Chapter 4

Partitioning the Edges of a 1-Planar Graph Efficiently

In this chapter, we show that it is possible to partition the edges of a 1-planar graph into a planar graph and a forest in linear time. As discussed in Chapter 3, this will allow us to orient the edges of a 1-planar bipartite graph with maximum outdegree 3 in linear time. Thus, this chapter completes the proof of Theorem 3.2.5, and concludes the description of the algorithm for finding a 4-list coloring of a 1-planar bipartite graph in linear time.

The existence of this edge partition was previously proved by Ackerman [1]. Following the steps of his proof, it is clear that the partition can be found in polynomial time. Ackerman further claims that the edge partition can be found in linear time, but provides no details. However, it is not clear how one would achieve a linear time (or even $O(n \log n)$ time) algorithm from his proof; we will discuss this in more detail below. (This was confirmed in private communication with Ackerman.)

We list here some previously known edge partition results for 1-planar graphs and other classes of so-called near planar graphs. As mentioned above, Ackerman [1] established that the edges of a 1-planar graph can be partitioned into a planar graph and a forest. This was an extension of an earlier result from Czap and Hudák [24], who proved the result for optimal 1-planar graphs. Lenhart et al. [58] showed that optimal 1-planar graphs can be partitioned into a maximal planar graph and a planar graph of maximum degree four (the bound of four is shown to be optimal). Bekos et al. [11] proved edge partition results for some k-planar graphs (graphs that can be drawn in the plane such that any edge crosses at most k other edges). Di Giacomo et al. [26] proved edge partition results for so-called NIC-graphs, a subclass of 1-planar graphs. With the exception of Ackerman [1], these

proofs either immediately lead to a linear time algorithm or the paper explicitly proves that the partition can be found in linear time.

In pursuit of a linear time algorithm, we re-prove Ackerman's result, using a different approach so as to avoid some problematic situations so that a linear time algorithm can then be established. We will then show that our proof can be implemented in linear time. A crucial ingredient for this will be a data structure by Holm, Italiano, Karczmarz, Łącki, Rotenberg and Sankowski [46] which allows for efficiently contracting edges of planar graphs. To our knowledge, this data structure has not been implemented. Because of this, we also show that the partition can be computed in $O(n \log n)$ time using a simpler data structure based on incidence lists.

Both Ackerman's and our proof operate heavily on the quadrangle faces of a planar graph. However, Ackerman's proof is only concerned with quadrangles that have exactly four distinct vertices on their boundary, whereas our proof operates on quadrangles that may have repeating vertices on their boundary (see the outer face in Figure 4.2(a)). We define a quadrangle f to be a *simple quadrangle* if it has four distinct vertices on its boundary. (The quadrangles of a simple graph will all be simple quadrangles.) The *facial circuit* of a quadrangle f is a 4-tuple $\langle z_0, z_1, z_2, z_3 \rangle$ such that each z_i is on the boundary of f and there are edges $z_i z_{i+1}$ and $z_i z_{i-1}$ (arithmetic modulo 4) that form an angle in f. Note that z_0, z_1, z_2, z_3 need not be distinct if f has loops or parallel edges, or if it is incident to a bridge. We say that z_0 and z_2 are *opposing vertices in* f (we will often omit mentioning the face when it is clear from context). Likewise z_1 and z_3 are opposing vertices in f. Note that it is possible for a vertex to oppose itself in a quadrangle (in Figure 4.2(a), the vertex $z_0 = z_2$ opposes itself).

We define here a graph operation that Ackerman uses in his proof. Let f be a face of a plane graph, and let a and b be two vertices on the boundary of f. We can *contract a and b through f* by adding an edge $e := ab$ drawn inside the region of f and contracting this edge e. This preserves planarity but destroys the face f.

For ease of notation, we define an abbreviation for our partition problem.

Definition 4.0.1. A *PGF-partition* of a graph G is a partition of $E(G)$ into two sets A and B such that $G[A]$ is a planar graph and $G[B]$ is a forest.

4.1 Ackerman's Proof

To establish the difficulties of achieving a linear time algorithm, we briefly review here Ackerman's proof for the existence of a PGF-partition.

Let G be a planar maximal 1-plane multigraph without loops. We assume that G is planar maximal as by Lemma 2.2.3 any 1-plane graph can be made planar maximal in linear time by adding edges, and adding edges can only make partitioning more difficult. We assume that there are no loops since they can be removed and added later to the planar part of the partition. On the other hand we specifically allow parallel edges since it is not clear how they could be handled otherwise. We remove all crossing pairs of G and call the resulting graph H the *(planar) skeleton* of G. Observe that the faces of H are either bigons, triangles, or quadrangles: These bigons and triangles were cells in G that only had vertices on their boundary (i.e. did not have crossing points on their boundary). The quadrangles of H were created by removing a pair of crossing edges from G, and in this way there is a 1-1 mapping between the quadrangles of H and the crossing pairs of G. Moreover, by this 1-1 mapping and the fact that the two edges forming a crossing pair do not share an endpoint, one can see that the quadrangles of H are in fact simple quadrangles. Ackerman establishes the following.

Lemma 4.1.1 (Ackerman [1])**.** *Let G be a 1-planar graph, and let H be the skeleton of G. If we can add a chord to every quadrangle of H such that the chords induce a forest, then G has a PGF-partition.*

Proof. Let C be the set of chords added to H. By the 1-1 mapping between quadrangles of H and crossing pairs of G, we know that exactly one edge from each crossing pair of G is contained in C. In particular, each edge $e \in C$ forms a crossing pair with some edge e' of G. Let C' be the set of these edges e'. By assumption $G[C]$ is a forest. Moreover, the graph $H \cup C'$ is the graph H plus a chord added to each quadrangle of H, and so is a planar graph. As $H \cup C'$ is also the graph induced by the edge-set $E(G) \setminus C$, this gives us the desired partition. □

Thus, in order to prove the existence of a PGF-partition, it suffices to show (typically by induction on the number of quadrangles) that such a set of chords can be found. For the induction to go through, Ackerman additionally forbids the chords from containing a path between two adjacent pre-specified vertices x, y, and he requires all quadrangles to be simple.

Theorem 4.1.2 (Ackerman [1])**.** *Let H be a planar multigraph without loops such that every face has degree at most four and all quadrangles are simple. Let xy be an edge of H. Then we can add a chord to every simple quadrangle of H such that the subgraph induced by the chords is a forest and does not contain a path between x and y.*

Proof. Proceed by induction on the number of quadrangles in H. If H has no quadrangles, then the statement is trivial. Otherwise, let f be a quadrangle with facial circuit $\langle z_0, z_1, z_2, z_3 \rangle$. By assumption f is simple, so the four vertices z_i are distinct.

Case 1: The only face containing z_0 and z_2 is f; in particular z_0 and z_2 are not adjacent. Contract z_0 and z_2 through f. Let H' be the graph resulting from this contraction. Observe that H' has one fewer quadrangle than H. All other quadrangles remain simple since f was the only face containing z_0 and z_2. Apply induction on H' with the same edge xy to receive a set of chords C', and then further add the chord z_0z_2. Chords have now been added to every quadrangle of H, and one verifies that we have not added a cycle or a path from x to y in the chords.

Case 2: There is some face $f' \neq f$ containing z_0 and z_2, but the only face containing z_1 and z_3 is f. Proceed as in Case 1, except contract z_1 and z_3.

Case 3: None of the above. Then there is a face $f' \neq f$ containing z_0 and z_2, and there is a face $f'' \neq f$ containing z_1 and z_3. Ackerman argues that $f' = f''$ (see also Lemma 4.2.2), and therefore f' is also a simple quadrangle. Observe that G can be split into four connected subgraphs H_0, H_1, H_2, H_3, where H_i contains z_i and z_{i+1} (addition modulo 4) on the boundary (see Figure 4.2(e)). One of these subgraphs, say H_0, will contain the edge xy. Apply induction on H_0 with xy, and apply induction on the other H_i with the edge z_iz_{i+1}. After induction, add the chords z_0z_2 in f, and z_1z_3 in f'. One verifies that the added chords do not add a path between x and y and that the chords do not create a cycle. □

4.2 An Alternate Existence Proof

While Ackerman's proof clearly leads to a polynomial time algorithm for finding the partition, it is not obviously linear since distinguishing between the cases and contracting are not obviously doable in constant time:

1. We need to test whether a given pair of vertices share more than one face.
2. The graph changes via contractions, and it is not obvious whether the existing data structures for efficiently contracting edges in planar graphs (e.g. [46]) would support (1) in constant time.
3. In Case 3 of Ackerman's proof, we need to identify the four subgraphs H_0, H_1, H_2, H_3. Furthermore, we need to determine which of these subgraphs contains the pair x, y. Neither operation is obviously doable in constant time.

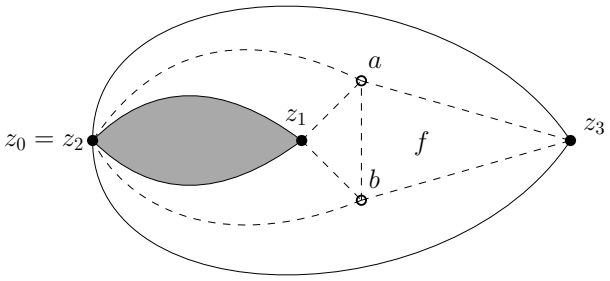


Figure 4.1: The construction used in Lemma 4.2.1. (The face f' is not shown.) New vertices are represented with circles, and new edges are represented with dashed edges.

We now give a different proof of the existence of a PGF-partition that either avoids these issues or addresses explicitly how to resolve them. The biggest change is how we handle Case 3 of Ackerman's proof. Ackerman used a split into four graphs, which is necessary in order to maintain that all quadrangles are simple. We prove a more general statement that permits non-simple quadrangles and hence avoids having to split the graph. Moreover, we generalize Ackerman's "forbidden edge" xy by choosing chords in such a way that the chords do not induce a path between *any* pair of vertices that were adjacent in the original graph H. Doing so simplifies the induction since we no longer need to keep track of where the vertices x and y are. Before we state this result we need a few helper-results that hold for all quadrangles (simple or not). They are easily proved by finding (after minor modifications) a planar drawing of K_5 if the conclusion is violated.

Lemma 4.2.1. *Let H be a plane multigraph without loops, let f be a quadrangle of H, and let $\langle z_0, z_1, z_2, z_3\rangle$ be the facial circuit of f. If $z_i = z_{i+2}$ (addition modulo 4) for some i, then $z_{i+1} \neq z_{i+3}$, and there is no face $f' \neq f$ that contains z_{i+1} and z_{i+3}.*

Proof. (see Figure 4.2(a)) Up to renaming, we may assume that $i = 0$, so $z_0 = z_2$. Assume for contradiction that $z_1 = z_3$. Then f consists of several parallel edges between $z_0 = z_2$ and $z_1 = z_3$, and thus f is not a quadrangle.

Assume for contradiction that there is some face $f' \neq f$ which contains both z_1 and z_3. Add two new vertices a and b inside the face f, along with edges az_0, az_1, az_3, bz_0, bz_1, and bz_3. (See Figure 4.1.) Along with this, add a chord z_1z_3 through the face f'. All these steps maintain the planarity of H. Moreover, the five vertices z_0, z_1, z_3, a, b are pairwise adjacent. But this forms a K_5 which is not planar, a contradiction. □

The following lemma was shown (without being stated explicitly) in Case 3 of Ackerman's proof.

Lemma 4.2.2. *Let H be a plane multigraph without loops, let f be a quadrangle of H, and let $\langle z_0, z_1, z_2, z_3\rangle$ be the facial circuit of f. If z_i and z_{i+2} (addition modulo 4) are both on some face $f' \neq f$ for some i, then no face $f'' \neq f, f'$ contains both z_{i+1} and z_{i+3}.*

Proof. (see Figure 4.2(b-e)) Up to renaming we may assume that $i = 0$, so z_0 and z_2 are on f and f'. Suppose for contradiction that such a face f'' exists. By Lemma 4.2.1, $z_1 \neq z_3$ and $z_0 \neq z_2$. Stellate f with a new vertex x, add an edge z_0z_2 through f', and add an edge z_1z_3 through f''. The original multigraph H was planar, and all of these operations preserve planarity. However, the five vertices z_0, z_1, z_2, z_3, and x are pairwise adjacent and form a K_5, which is not planar, a contradiction. □

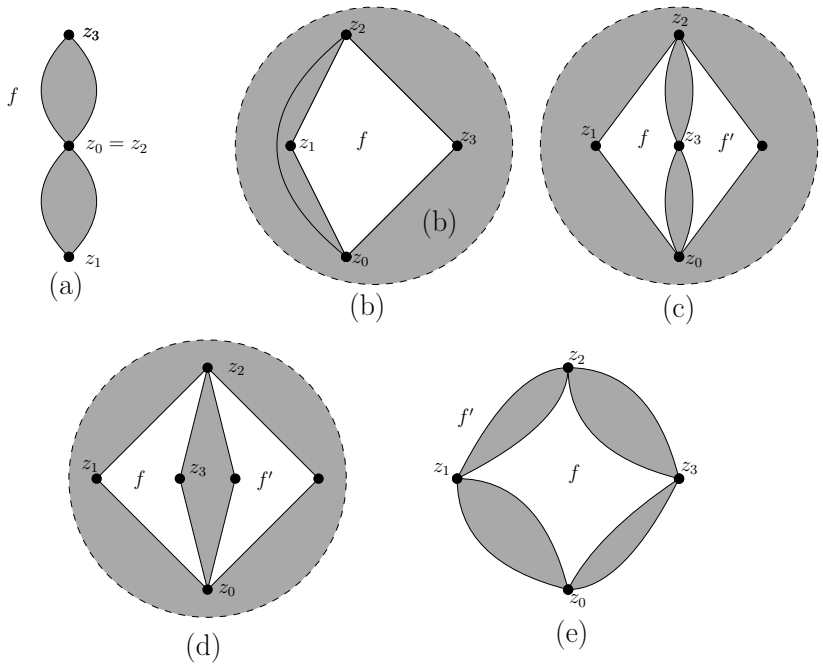


Figure 4.2: Some configurations where we contract z_1 and z_3. Note that in (a), the quadrangle f is the outer face.

Now we reprove the existence of quadrangle-chords that form a forest.

Theorem 4.2.3. *Let H be a plane multigraph without loops such that every face has degree at most 4. Then it is possible to add a chord to every quadrangle of H such that the graph induced by the chords is a forest. Moreover, for any edge ab of H, there is no path from a to b in the chords.*

Proof. As in Ackerman's proof, we prove the claim by induction on the number of quadrangles in H and remove each quadrangle by contracting an opposing pair of vertices in the quadrangle. If H has no quadrangles, the claim is trivial. Otherwise, let f be a quadrangle of H with facial circuit $\langle z_0, z_1, z_2, z_3 \rangle$. We first pick two opposing vertices of f to contract.

Case 1: This case covers when we choose to contract z_1 and z_3, and has three sub-cases. We contract z_1 and z_3 whenever

Case 1.a $z_0 = z_2$ are the same vertex, or

Case 1.b z_0 and z_2 are adjacent, or

Case 1.c z_0 and z_2 are opposing vertices of some quadrangle $f' \neq f$. [1]

Figure 4.2 illustrates possible configurations of face f where Case 1 applies: Case 1.a applies to (a), Case 1.b applies to (b), and Case 1.c applies to (c,d,e). Note that Case 1.c covers Case 3 of Ackerman's proof, where z_0, z_1, z_2, z_3 all belong to two simple quadrangles f, f' (see also Figure 4.2(e)). Our contraction turns f' into a non-simple quadrangle, but our proof can handle this.

Case 2: Otherwise, we contract z_0 and z_2.

Table 4.1 demonstrates when we pick Case 1 and when we pick Case 2, and crucially shows cases which are impossible by Lemmas 4.2.1 and 4.2.2. To see that these lemmas apply in the second row and column, observe that a pair of adjacent vertices x, y must be on some face that has the edge xy on its boundary. In particular, if x, y are opposing vertices on some quadrangle f, then additionally they are also on the boundary of some face $f' \neq f$ where the edge xy is on the boundary of f'.

Let z_i, z_{i+2} be two vertices chosen for contraction and let H' be the graph resulting from contracting z_i and z_{i+2}. By Table 4.1, z_i and z_{i+2} are not adjacent, and they are distinct. Therefore our contraction has destroyed the quadrangle f and not added any loops, so we can apply induction on H'. Let C' be the set of chords added to H'. By the inductive hypothesis, C' induces a forest and for any edge $ab \in H'$, there is no path from a to b in C'. Uncontract z_i and z_{i+2}, and add a chord $e := z_i z_{i+2}$ between them. Define $C := C' \cup \{e\}$. We now verify that C satisfies all conditions.

[1] In fact, our proof does not require Case 1.c to be separated out; we could equally have contracted z_0 and z_2 in this case.

	$z_0 = z_2$	$z_0 \neq z_2$; $z_0z_2 \in E(H)$	$z_0 \neq z_2$; z_0, z_2 are opposing in f'	Otherwise
$z_1 = z_3$	Impossible (Lemma 4.2.1)	Impossible (Lemma 4.2.1)	Impossible (Lemma 4.2.1)	Case 2
$z_1 \neq z_3$; $z_1z_3 \in E(H)$	Impossible (Lemma 4.2.1)	Impossible (Lemma 4.2.2)	Impossible (Lemma 4.2.2)	Case 2
$z_1 \neq z_3$; z_1, z_3 are opposing in f''	Impossible (Lemma 4.2.1)	Impossible (Lemma 4.2.2)	Impossible if $f' \neq f''$ (Lemma 4.2.2), Case 1.c otherwise	Case 2
Otherwise	Case 1.a	Case 1.b	Case 1.c	Case 2

Table 4.1: All possible cases for the quadrangle f with facial circuit $\langle z_0, z_1, z_2, z_3\rangle$. We either indicate which case in the proof of Theorem 4.2.3 would be chosen, or indicate the lemma that demonstrates that this case is impossible.

Let ab be an edge of H. Assume for contradiction that there is a path π from a to b in C. By the inductive hypothesis π must use e. Furthermore, e cannot be the edge ab since z_i and z_{i+2} are not adjacent, so π must use some edges from C'. Let $c_1, \ldots, c_{k_1}, e, c_{k_1+1}, \ldots, c_{k_2}$ be π for some $1 \leq k_1 \leq k_2$. But then $c_1, \ldots, c_k$ would be a path from a to b within C' in $H' = H/e$, a contradiction.

Assume for contradiction that C contains a cycle in H. Since e is not a loop in H, the cycle must use edges from C'. Let $e, c_1, \ldots, c_k$ be the cycle, $k \geq 2$. Then $c_1, \ldots, c_k$ would induce a cycle within C' in $H' = H/e$, a contradiction. □

We therefore have the following algorithm for finding a PGF-partition of a 1-planar graph.

Algorithm 4.2.4. *Let G be a 1-planar multigraph without loops. Let G^+ be a planar maximal supergraph of G and let H be the skeleton of G^+. For each quadrangle f of H with facial circuit $\langle z_0, z_1, z_2, z_3\rangle$, if z_0 and z_2 are adjacent, or $z_0 = z_2$, or z_0 and z_2 are opposing on a quandrangle $f' \neq f$, contract z_0 and z_2. Otherwise, contract z_0 and z_2. Define a partition A, B of G as follows: the partition A is all edges $ab \in E(G)$ such that the pair a, b was never contracted, and the partition B is all other edges of G.*

4.3 Efficient Implementation

It is still not immediately clear how one would implement Algorithm 4.2.4 in order to achieve linear run-time, since as in Ackerman's proof we need to repeatedly test how many quadrangles two vertices share. However, if we are more careful about the order in which we contract each quadrangle, an efficient implementation can be achieved. The crucial idea will be to pick some vertex x and contract all quadrangles incident to x. This will allow us to store additional information relative to x and hence speed up testing which case applies. We note that this idea alone would not suffice to make Ackerman's proof run in linear time, as one would still need to find a way to implement his Case 3 efficiently, where he splits the graph into four subgraphs and determines which subgraph contains a given pair of vertices.

4.3.1 Data Structure Interface

As mentioned earlier, one of the major ingredients to achieve fast run-time is to use the data structure by Holm et al. [46] for contraction in planar graphs, but we will also provide an alternative which uses a simpler data structure, at the cost of a slower run-time. We will discuss these later (in Section 4.3.4) when we analyze the run-time, but note here the two operations provided by [46] that will be needed:

- $x =$ `contract`(e) takes a reference to an edge e, contracts e, and returns the vertex resulting from the contraction.

  Note that contracting e creates a loop in the graph, while the proof of Theorem 4.2.3 assumed that the graph has no loops. We could remove loops (the data structure by Holm et al. can report newly created loops after `contract`), but this turns out to be unnecessary: We will only contract edges at artificial gadgets inserted into the graph, and the created loops are at quadrangles that are destroyed afterwards and those will not pose problems.

- `neighbors`(x) returns an iterator over $\{\langle xv, v\rangle : xv \in E\}$ where E is the edge set of the graph. In other words, it returns an iterator to tuples containing each edge incident to x and the endpoint of this edge. In our pseudocode we will often not need the edges, and so we use `neighbors'` to get an iterator over only the vertices. No guarantee is given as to the order of the neighbors.

  We assume that `neighbors`(x) has $O(1)$ run-time and that the returned list can be iterated over in $O(d(x))$ time. Since edge-contraction can create parallel edges, it is

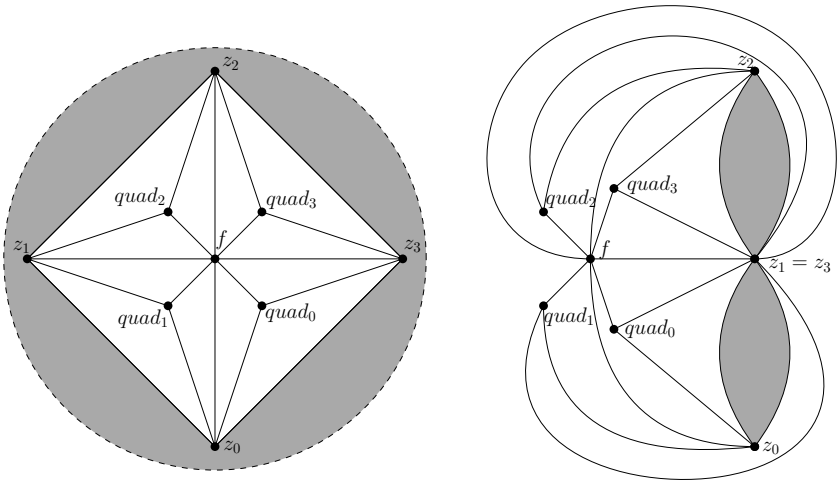


Figure 4.3: The gadget added to every quadrangle of H, shown for both a quadrangle with four distinct vertices on its boundary (left) and one with three distinct vertices on its boundary (right).

possible that `neighbors`(x) contains parallel edges, and hence the second element of the tuple need not be unique. Again this will not pose problems later.

In Section 4.3.2, we will add labels and other meta-data to vertices of our graph. We make no assumptions as to how the meta-data are updated when two vertices are contracted, and so we will maintain those manually.

4.3.2 Preprocessing

We take as input a 1-plane multigraph G without loops. Recall that when we are given a 1-plane graph, it is given by specifying its planarization via the rotational clockwise order of edges at the vertices, and assuming that vertices of the planarization resulting from crossing points are marked as such. By Lemma 2.2.3, we can construct a planar maximal supergraph G^+ in linear time by adding edges. This also gives us the planarization $(G^+)^\times$ of G^+. As before we use H to denote the skeleton of G^+, but we do not construct it explicitly. Instead, notice that the vertices of $(G^+)^\times$ marked as crossings correspond uniquely to the quadrangles of H. For this reason, we mark all these vertices with a label *quad* and we call such a vertex a *quadrangle-vertex* (whereas the corresponding face of H is called a *quadrangle-face*)

Our proof of Theorem 4.2.3 relies heavily on having faces, while the data structure of Holm et al. makes no provisions for accessing faces. For this reason, we keep the quadrangle-vertices in the graph as representatives of the quadrangle-faces. We also need to know the order of the vertices along the boundary of a quadrangle, which again the data structure does not support. Therefore at any quadrangle-vertex f we stellate each of the four incident triangular faces. (See Figure 4.3.) Since we have yet not contracted any edges, we have access to the rotational clockwise order of edges at f; we can hence label the added vertices with $quad_i$ (for $0 \leq i \leq 3$) in clockwise order around f. We use $H^\diamondsuit$ for the graph that results after all these modifications (it can be viewed as the skeleton H with a "diamond"-gadget inserted into each quadrangle-face). Clearly a 1-plane multigraph can only have $O(m)$ crossing points, and therefore $H^\diamondsuit$ has size $O(n+m)$ and can be constructed from G in linear time.

The added vertices will make it possible to implement the operation "contract z_i and z_{i+2} within quadrangle f" used in Theorem 4.2.3 via edge-contractions at the corresponding quadrangle-vertex f: We contract each $quad_j$ into f for $0 \leq j \leq 3$, and then contract z_i and z_{i+2} into f. See Procedure 4 for details. We also assume that the quadrangle-vertex f has references to the two original edges in G that crossed; when doing this contraction within f we can hence also record the corresponding edge $z_i z_{i+2}$ for inclusion in the forest part of the partition.

Procedure 4: ContractThrough(u, v, f)

Result: Contract two vertices u and v in H through a quadrangle-face f, and return the resulting vertex y.

`// pre:` f `is labelled` *quad*

`// pre:` u `and` v `are opposing on the face of` H `corresponding to` f

1 Find the original edge uv of G that is stored with f.
2 Record edge uv as belonging to the forest of the partition.
3 **for** $\langle e, w \rangle$ *in* `neighbors`(f) **do**
4 **if** v *equals* w ***or*** v *equals* u ***or*** w *has label* $quad_i$ *for some* $0 \leq i \leq 3$ **then**
5 $y :=$ `contract`(e)
6 Remove labels $quad, quad_i$ from y
7 **return** y

The labels $quad_i$ also make it possible to recover the clockwise order $\langle z_0, \ldots, z_3 \rangle$ of vertices on the quadrangle-face corresponding to f in constant time: If f is a simple quadrangle, then the neighbors of $quad_i$ are z_{i-1}, the quadrangle-vertex f, and z_i. The

vertices $quad_i$ and $quad_{i+1}$ only have the neighbors z_i and f in common. This allows us to recover the facial circuit. Special care must be taken if the quadrangle f is not simple, because if $z_{i-1} = z_{i+1}$, then $quad_i$ and $quad_{i+1}$ will have all three neighbors in common. (See Figure 4.3(right).) However, in such a case we can still determine z_{i-1} as in the simple case, and this allows us to recover z_i. See Procedure 5 for details.

Procedure 5: FacialCircuit(f)

Result: Reconstruct the facial circuit of a quadrangle-face, given the corresponding quadrangle-vertex.

`// pre:` f `is a vertex of` $H^{\diamondsuit}$ `with label` $quad$

for *vertex* v *in* `neighbors'`(f) **do**
  for $i = 0, \ldots, 3$ **do**
    if v *has label* $quad_i$ **then** $f_i := v$
for $i = 0, \ldots, 3$ **do**
  $N_i :=$ `neighbors'`$(f_i) \cap$ `neighbors'`(f_{i+1})
  if $|N_i|$ *equals* 2 **then** $z_i := N_i \setminus \{f\}$

`// Special case for non-simple quadrangles`

for $i = 0, \ldots, 3$ **do**
  if $|N_i|$ *equals* 3 **then** $z_i := N_i \setminus N_{i+2}$
return $\langle z_0, z_1, z_2, z_3 \rangle$

We also add the following meta-data to each vertex of $H^{\diamondsuit}$:

- A boolean `adj`, initialized to **false**.
- A boolean `in_worklist`, initialized to **false**.
- An integer `opposing`, initialized to 0.

The main idea of our algorithm is to iteratively contract all the quadrangles incident to some vertex x. As we do this, we will use `adj` to mark vertices that are adjacent to x, `in_worklist` to mark unprocessed quadrangles incident to x, and we will use the value `opposing` for a vertex y to keep track of the number of quadrangle-faces where y is the opposing vertex of x.

4.3.3 Handling the Quadrangles around a Vertex

The main subroutine of our algorithm handles all quadrangles incident to some vertex x by contracting each of them using the criteria laid out in Theorem 4.2.3 to decide which vertices to contract. To do so, we will initialize and maintain a work-list of faces incident to x that we need to contract (using `in_worklist` to avoid putting duplicate quadrangles into the worklist). We also mark vertices in $H^{\diamondsuit}$ as `adj` and increment `opposing` as needed. This can be done in $O(d(x))$ time by retrieving all neighbours of x via `neighbours`. Procedure 6 shows the details.

Now we iteratively contract all the faces in the worklist according to Theorem 4.2.3; with `adj` and `opposing` we can determine the correct case in constant time. See Procedure 7 for details. In Case 1, we need to update some values of opposing: the vertex z_2 which was opposing x now no longer opposes x in f (since f was destroyed), so we decrement z_2.`opposing`. Likewise the new vertex v resulting from the contraction will be opposing x in all those quadrangles in which z_1 and z_3 previously opposed x, so we set v.`opposing` correspondingly. In Case 2, when we contract some vertex z_2 into x, we need to add the quadrangles incident to z_2 to our worklist, for which we can re-use Procedure 6.

Lastly, once our worklist is empty (and hence there are no more quadrangles incident to x), we reset the meta-data of x and its neighbors, so that when repeating the procedure with a different vertex as x there are no stray vertices with meta-data set to erroneous values. See Procedure 8 for details.

Procedure 6: InitializeAtOneVertex(y, `worklist`)

Result: Adds all quadrangle-vertices incident to a vertex y to a worklist, taking care not to put duplicates in the worklist.

```
// pre:  y is a vertex of H
// pre:  y equals x (the vertex we are currently working on), or y
   will be contracted into x
for vertex v ∈ neighbors'(y) do
    v.adj := true
    if v has label quad and not v.in_worklist then
        v.in_worklist := true
        worklist.push(v)
        ⟨z0, z1, z2, z3⟩ := FacialCircuit(v)
        while y ≠ z0 do relabel ⟨z0, z1, z2, z3⟩ := ⟨z1, z2, z3, z0⟩
        z2.opposing += 1
```

Procedure 7: HandleQuadsAtOneVertex(x)

Result: Contract all the quadrangle-faces incident to a vertex x

```
worklist := []
InitializeAtOneVertex(x, worklist)                          // see Proc. 6
for quadrangle-vertex f in worklist do
    ⟨z0, z1, z2, z3⟩ := FacialCircuit(f)                    // see Proc. 5
    while x ≠ z0 do relabel ⟨z0, z1, z2, z3⟩ := ⟨z1, z2, z3, z0⟩
    if z2 equals x or z2.adj is true or z2.opposing ≥ 2 then      // Case 1
        opposing1 := z1.opposing
        opposing3 := z3.opposing
        v := ContractThrough(z1, z3, f)                     // see Proc. 4
        v.adj := true
        v.opposing := opposing1 + opposing3
        z2.opposing −= 1
    else                                                        // Case 2
        InitializeAtOneVertex(z2, worklist)
        x := ContractThrough(x, z2, f)
CleanupAtOneVertex(x)                                       // see Proc. 8
```

Procedure 8: CleanupAtOneVertex(y)

Result: Cleanup the metadata at a vertex y and its neighbors.

`// pre:` y `is a vertex of` H

for *vertex* $v \in$ `neighbors'`$(y) \cup \{y\}$ **do**
    v.`adj` := **false**
    v.`in_worklist` := **false**
    v.`opposing` := 0

4.3.4 Putting it All Together

The following summarizes our algorithm: after preprocessing, and for as long as there is a quadrangle-face f left, process all quadrangles at a vertex x on f and record all edges that belong to the forest along the way. See Procedure 9 in for a detailed description. We have correctness by Algorithm 4.2.4.

Procedure 9: FindPGFPartition(G)

Result: Find a PGF-partition of a 1-plane graph G.

Add edges to make G planar maximal
Compute planarization $G^\times$, mark vertices of crossings with *quad*
`// Construct` $H^\diamondsuit$ `from` $G^\times$
foreach *vertex* f *marked quad* **do**
    Insert four vertices in four incident faces of f
    Mark these vertices with $quad_0, \ldots, quad_3$ according to embedding
Initialize edge-contraction data structure with $H^\diamondsuit$
while *there remains a vertex* f *labeled quad* **do**
    x := some neighbor of f not labeled $quad_i$
    `HandleQuadsAtOneVertex(`x`)`
Return all edges that were recorded as forest F, and $G \setminus F$ as planar graph

Runtime It remains to analyze the run-time. For now, we ignore the time required to perform the contractions and analyze the time for handling all quadrangles at one vertex x. Initialization takes $O(d(x))$ time, and most other steps take constant time per handled quadrangle, with one notable exception: When we contract some vertex z_2 into x, we must update the worklist, which takes time $O(d(z_2))$. Complicating matters further, z_2 may

actually be the result of prior contractions, so its degree may be more than what it was in $H^\diamondsuit$, and we must ensure that degrees of vertices are not counted repeatedly.

To account for the work, consider the graph $H_f^\diamondsuit$ that results from $H^\diamondsuit$ after all of the quadrangle-vertices have been contracted. For a vertex $x \in V\left(H_f^\diamondsuit\right)$, let $s(x)$ be the set of vertices of $H^\diamondsuit$ that were contracted into x, either directly (when handling the quadrangles at x) or indirectly (i.e., if they had been contracted into one of the vertices z_2 that later get contracted into x). Crucially, note that if x, x' are two vertices that are parameters during calls to Procedure 7 (i.e., we contract all quadrangles incident to the vertex), then $s(x)$ and $s(x')$ are disjoint. This holds because once we are done with the first of them (say x), all vertices in $s(x)$ have been combined with x and no longer have any incident quadrangles. Since we only contract vertices into x' that are incident to quadrangles, none of the vertices in $s(x)$ becomes part of $s(x')$.

Hence for each vertex x of $H_f^\diamondsuit$, the amount of work done in Procedure 7 is proportional to the sum of the degrees $d_{H^\diamondsuit}(y)$ for each $y \in s(x)$. Since $H^\diamondsuit$ has $O(n)$ edges, and all other parts of the algorithm take constant time per quadrangle-vertex, the total amount of work done is at most

$$O\left(\sum_{x \in V\left(H_f^\diamondsuit\right)} \sum_{y \in s(x)} d_{H^\diamondsuit}(y)\right) = O\left(\sum_{x \in V(H^\diamondsuit)} d_{H^\diamondsuit}(x)\right) = O(|E(H^\diamondsuit)|) = O(n+m)$$

hence the algorithm is linear (ignoring the time for contractions).

Now we consider the run-time of possible data structures for contractions. Our first approach is to use the usual data structure for a graph introduced in Chapter 2 using incidence lists, where every vertex has a list of incident edges, each edge knows both of its endpoints, and every list knows its length. We can implement `neighbors`(x) in constant time by simply returning an iterator to the incidence list at x. Contracting two vertices u and v can be done in $O(\min\{d(u), d(v)\})$ time by re-attaching the edges of the vertex with smaller degree to the vertex with larger degree.

Lemma 4.3.1. *Let G be a multigraph implemented with incidence lists. Then any number of contractions can be done in $O(m \log m)$ time.*

Proof. Consider some edge e of G. We determine how many times one of the endpoints of e is changed. Suppose that during a contraction, an endpoint v of e is changed to the vertex u. Then it must have been that $d(u) \geq d(v)$. The vertex u' resulting from the contraction will have degree $d'(u) = d(u) + d(v) \geq 2d(v)$. Therefore, the degree of one of

the endpoints of e has doubled. Since the maximum degree is m, an endpoint of e can be changed at most $2 \log m$ times. Therefore, the total amount of times an endpoint of some edge is changed is $O(m \log m)$. The time for all contractions is proportional to the number of endpoint changes and hence also $O(m \log m)$. □

Note that this is analogous to implementing UNION-FIND in $O(n \log n)$ time, a well known result (see e.g. Section 4.6 of [55]). With this we have our first result.

Theorem 4.3.2. *There exists an algorithm that, for any input graph G implemented with incidence lists that comes with a 1-planar embedding, finds a PGF-partition of G in $O(m \log m)$ time.*

If G is simple, then $m \leq 4n - 8$, and hence the algorithm runs in time $O(n \log n)$. To improve this run-time, we appeal to the following result by Holm et al.

Theorem 4.3.3 (Holm et al. [46])**.** *Let G be a (not necessarily simple) planar graph with n vertices and m edges. Then there exists a data structure that supports* `contract` *and* `neighbours` *and that can be initialized in $O(n + m)$ time. Any call to* `neighbors` *can be processed in worst case constant time, and any sequence of $\Theta(m)$ calls to* `contract` *can be performed in time $O(n + m)$.*

Using this data structure, we have the main result of this chapter.

Theorem 4.3.4. *There exists an algorithm that, for any input graph G that comes with a 1-planar embedding, finds an edge partition of G into a planar graph and a forest in $O(n + m)$ time.*

Again, if G is simple, then this algorithm runs in time $O(n)$. We note here that the algorithm we have presented requires (regardless of the underlying data structure used) that we are given the embedding of the 1-planar graph. Indeed, the proof itself (both ours and Ackerman's) relies heavily on properties of the embedding. Finding this partition is the only step in 4-list coloring a 1-planar bipartite graph that requires the embedding. This leads to the following open problem.

Open Problem 4.3.5. *Is there a polynomial time algorithm to find a partition of a 1-planar graph into a planar graph and a forest without being given an embedding?*

Even if in general there is no polynomial time algorithm, it is possible that we can find a PGF-partition without an embedding when the graph is bipartite. If no polynomial time

algorithm is found for this, it is possible that there is a linear time algorithm for 4-list coloring a 1-planar bipartite graph that avoids having to find the PGF-partition. Recall that we needed this partition to orient the edges of the 1-planar bipartite graph such that the vertices have outdegree at most three. Without an embedding, the fastest algorithm known to us to find this orientation runs in time $O(m^{1.5})$ [17].

Open Problem 4.3.6. *Is there a linear time algorithm to find a 4-list coloring of a 1-planar bipartite graph without being given am embedding?*

Chapter 5

Coupled List Coloring Planar Graphs of Small Treewidth

In this chapter we study the problem of coupled list coloring planar graphs of small treewidth. In doing so we study the structure of coupled graphs (the graphs of vertex and face adjacencies of planar graphs), in particular characterizing their relationship with optimal 1-planar graphs. We then characterize the coupled choosability of wheel graphs, with an eye towards the coupled choosability of planar partial 3-trees.

5.1 Simple Coupled Choosability Results

We first give a few simple results for coupled choosability. These will be helpful in Section 5.4 when we study the coupled choosability of subgraphs of wheel graphs, and they will also help to familiarize the reader with the problem of coupled choosability. Here we revisit the characterization of the graphs that are 3-coupled choosable (this was briefly mentioned in Chapter 2).

Lemma 5.1.1. *Let G be a connected planar graph. Then G is 3-coupled choosable if and only if G is a tree. Moreover, for any tree G with 3-list assignment L, we can find an L-coloring of G in linear time.*

Proof. ($\Rightarrow$) Suppose G is not a tree. Then G contains a cycle and therefore must have at least two faces. In particular, G has two faces f, g adjacent via an edge $e = xy$. In $C(G)$, we have that f, g, x, y are pairwise adjacent: we chose f, g and x, y such that they

are adjacent, and the vertices x and y both lie on the faces f and g. Therefore, the coupled graph $C(G)$ contains K_4 as a subgraph, and so G is not 3-coupled choosable (since K_4 is not even 3-colorable).

($\Leftarrow$) Let G be a tree and let L be a 3-coupled list assignment for G. We first color the unique face f of G with an arbitrary color of $c \in L(f)$. It remains to list color the vertices of G. We cannot use the color c for any vertex of G, and therefore define lists L' for the vertices of G given by $L'(x) := L(x) \setminus \{c\}$. We have that $|L'(x)| \geq 2$ for every vertex $x \in V(G)$, and G is 2-list colorable because it is a tree, so an L'-coloring of G can be found. We then therefore have an L-coloring of G. Clearly this is a linear time algorithm. □

Now we study how graph operations affect the coupled list chromatic number. In contrast to list coloring, where the list chromatic number of a graph is an upper bound for the list chromatic number of its subgraphs, there is no clear relationship between the coupled list chromatic number of a graph and the coupled list chromatic number of its subgraphs. Indeed, it is possible for a subgraph to have a larger coupled list chromatic number.

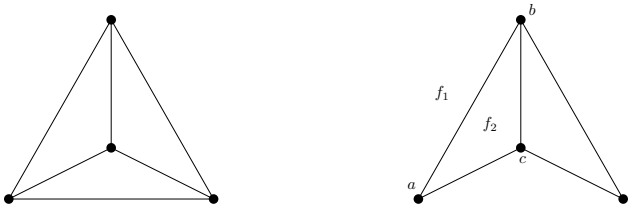


Figure 5.1: The graph K_4 (left) and subgraph H (right).

Theorem 5.1.2. *There exists a plane graph G with subgraph H such that the coupled list chromatic numbers satisfy $\chi_{vf}^L(H) > \chi_{vf}^L(G)$.*

Proof. Define $G = K_4$, and let H be the result of deleting one edge from K_4. From Theorem 10 of [90] we know that $\chi_{vf}^L(K_4) = 4$. But in the graph $C(H)$ and in the naming of Figure 5.1, we have that a, b, c, f_1, f_2 are all pairwise adjacent: the vertices a, b, c are pairwise adjacent in H, the faces f_1, f_2 share an edge, and a, b, c all lie on the boundaries of both f_1 and f_2. Therefore $C(H)$ contains K_5 as a subgraph, and it must be that $\chi_{vf}^L(H) \geq 5 > 4 = \chi_{vf}^L(K_4)$. (Note that H is outerplanar, so in fact $\chi_{vf}^L(H) = 5$; see Corollary 5.1.7.) □

Other graph operations are better behaved in this respect. For instance, there is a clear relationship between the coupled choosability of some graph G and the coupled choosability of any subdivision of G.

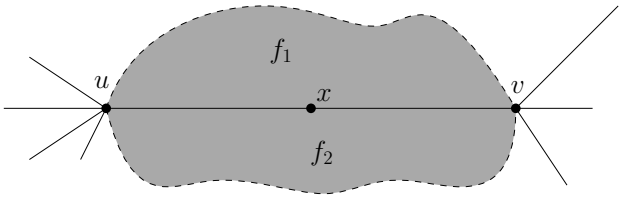


Figure 5.2: A vertex x resulting from subdividing the edge uv, along with the two faces f_1 and f_2 adjacent to x.

Lemma 5.1.3. *For any plane graph G, any subdivision H of G is $\max\{5, \chi^L_{vf}(G)\}$-coupled choosable.*

Proof. Let L be a list assignment for H such that $|L(x)| = \max\{5, \chi^L_{vf}(G)\}$ for every vertex and face of H. We prove the statement by induction on the number of subdivisions performed on G to obtain H. If H is the result of subdividing the edges of G zero times, then $H = G$ and so trivially any L-coupled coloring of G is an L-coupled coloring of H.

Otherwise, H was the result of performing $k+1$ subdivisions on G for some $k \geq 0$. In particular, H is the result of subdividing a single edge of some graph H', where H' was the result of performing k subdivisions on G. Let $uv \in E(H')$ be the edge of H' that was subdivided, and let x be the vertex which was added. By the inductive hypothesis, H' is $\max\{5, \chi^L_{vf}(G)\}$-coupled choosable. Color the faces of H and the vertices $V(H) \setminus \{x\}$ according to how they would be colored in H'. Then we only need to color the remaining vertex x. Note that x has degree two with neighbors u and v. Let f_1 and f_2 be the two faces adjacent to the edge uv in H'. Then u, v, f_1, and f_2 are the only vertices and faces that are adjacent (respectively incident) to x. (See Figure 5.2.) Hence, after coloring the vertices and faces of H', x still has at least $|L(x)| - 4 \geq 5 - 4 = 1$ color left and can be colored. □

This implies another result. For a planar graph G, subdividing an edge corresponds in the dual graph G^* to duplicating edges to form bigons (recall that these are faces of degree 2). Since $\chi^L_{vf}(G) = \chi^L_{vf}(G^*)$ we therefore have

Corollary 5.1.4. *Let G be a plane graph and H be the result of duplicating some edges of G to form bigons. Then H is $\max\{5, \chi^L_{vf}(G)\}$-coupled choosable.*

We present a similar result for adding a vertex of degree one to a graph.

Lemma 5.1.5. *Let G be a planar graph, and let H be G plus a new vertex of degree one. Then H is $\max\{3, \chi^L_{vf}(G)\}$-coupled choosable.*

Proof. Let x be the new vertex, and let L be a list assignment for H such that $|L(y)| = \max\{3, \chi^L_{vf}(G)\}$ for every vertex and face of H. Color the faces and vertices of $H - x$ according to how they would be colored in G. It remains to color x. Since x is adjacent to only one vertex and incident to only one face in H, after coloring the vertices and face of $H - x$, x still has at least $|L(x)| - 2 \geq 3 - 2 = 1$ color left and can be colored. □

Note that for the last three results, the coloring of H can be found in constant time given a suitable coloring of G. We now use these results to prove that all K_4-minor free graphs are 5-coupled choosable, and that this coloring can be found in linear time. The existence of this coloring was previously known [45, 90], but the linear time result is new.

Theorem 5.1.6. *All K_4-minor free graphs are 5-coupled choosable, and the coloring can be found in linear time.*

Proof. It is known (see e.g. [27]) that every K_4-minor free graph G can be constructed from some tree T via a series of duplicating edges, subdividing edges, and adding vertices of degree one. Then by Lemmas 5.1.1, 5.1.3, and 5.1.5, and by Corollary 5.1.4, we have that G is 5-coupled choosable.

It is known how to find this construction from a tree in linear time if the graph G is 2-connected [84], and this can be extended to 1-connected graphs by running this construction on the maximal 2-connected components of G. Since the 3-coupled list coloring of the tree can be found in linear time, and each of our expansion steps takes constant time, we can find the coloring of G in linear time. □

Recall that an outerplanar graphs is a planar graph with all vertices on the outer face. Since outerplanar graphs are K_4-minor free, we have the following corollary.

Corollary 5.1.7. *All outerplanar graphs are 5-coupled choosable, and the coloring can be found in linear time.*

5.2 Treewidth of Coupled Graphs

Recall the definition of treewidth and branchwidth from Section 2.4. In this section we study the relationship between $tw(G)$ and $tw(C(G))$ for a planar graph G. Trivially we have that $tw(C(G)) \geq tw(G)$ as $C(G)$ is a supergraph of G. We wish to show that $tw(C(G))$ can also be upper bounded in terms of $tw(G)$. In Chapter 6, this will allow us to prove results on the parameterized complexity of coupled list coloring problems when

parameterized by the treewidth of the planar graph. Towards our goal, we first show that if a planar graph has small branchwidth then the coupled graph has small branchwidth. Recall that bounds on the branchwidth imply bounds on the treewidth as follows:

Lemma 5.2.1 (Robertson and Seymour [76]). *Let G be a graph with $bw(G) > 1$. Then $bw(G) - 1 \leq tw(G) \leq \frac{3}{2}bw(G) - 1$.*

To prove the bound on the branchwidth, we use that for most planar graphs, the branch decomposition of the primal graph can be used to construct a branch decomposition of the dual graph. It is a direct consequence of the work from Mazoit and Thomassé [62], who studied the branchwidth of planar graphs by way of branch decompositions of matroids.

Lemma 5.2.2 (based on [62]). *Let G be a planar graph that is not a tree and let G^* be the dual of G. Let $\mathcal{T}$ be a branch decomposition of G and let $\mathcal{T}^*$ be the branch decomposition of G^* obtained by starting with the tree $\mathcal{T}$ and, for each leaf t of $\mathcal{T}$ mapped to an edge $e \in E(G)$, instead map it to the dual edge $e^* \in E(G^*)$. Then the width of $\mathcal{T}^*$ is equal to the width of $\mathcal{T}$.*

Mazoit and Thomassé state the result for planar graphs without bridges, but within their paper they also discuss that the crucial ingredient for the proof fails only if G is a tree of branchwidth two. In the case where the graph G is a tree, it may be that $bw(G) = 2$, but the dual graph G^* of a tree is a single vertex with many loops, which trivially has branchwidth one. As a direct corollary of Lemma 5.2.2, for any planar graph G that is not a tree, we have $bw(G) = bw(G^*)$. With this, we can bound the branchwidth of the coupled graph.

Theorem 5.2.3. *Let G be a connected planar graph with $n \geq 2$ vertices that is not a tree. Then $bw(C(G)) \leq 2bw(G)$.*

Proof. Let $\mathcal{T}$ be a branch decomposition of G of minimal width and $\mathcal{T}^*$ be the branch decomposition of G^* of Lemma 5.2.2. We will construct a branch decomposition $\mathcal{T}_c$ of $C(G)$.

We begin with $\mathcal{T}$. Let t be a leaf of $\mathcal{T}$ mapped to an edge $e = xy$ of G. Let $e^* = fg$ be the dual edge of e. In the coupled graph these four vertices form a K_4 (or a K_3 with some parallel edges and a loop in the case where e is a bridge). We replace t with a recursive partition of the edges $\{e, e^*, xf, xg, yf, yg\}$. (See Figure 5.3.) If any of the edges xf, xg, yf, yg were included in a previous recursive edge partition when constructing $\mathcal{T}_c$, then simply exclude them here. It is easy to see that $\mathcal{T}_c$ is indeed a branch decomposition of $C(G)$. It remains to prove the claimed upper bound on the width of $\mathcal{T}_c$.

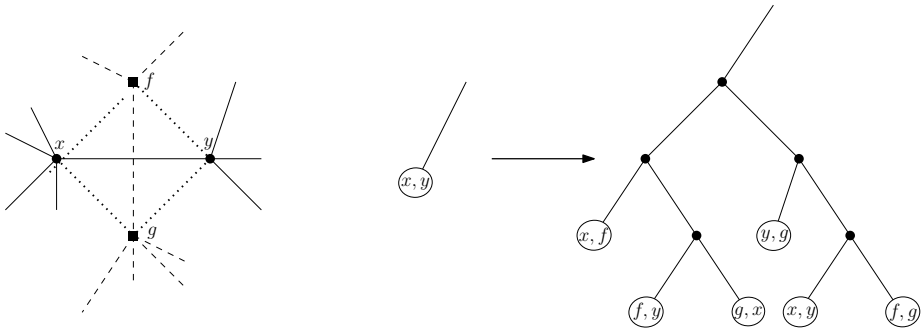


Figure 5.3: A primal edge xy in the coupled graph of a plane graph G (left), and the transformation applied to the leaf of the branch decomposition of G containing xy used in Theorem 5.2.3 (right).

Let a be an arc of $\mathcal{T}_c$. We first examine the case where a was an original arc of $\mathcal{T}$. The separator $\sigma(a)$ can be partitioned into two sets $\sigma_G(a)$ and $\sigma_{G^*}(a)$, where the former consists of primal vertices and the latter consists of dual vertices. These are the separators of a in $\mathcal{T}$ and $\mathcal{T}^*$, respectively. Therefore we have

$$|\sigma(a)| = |\sigma_G(a)| + |\sigma_{G^*}(a)| \leq bw(G) + bw(G^*) \leq 2bw(G).$$

Otherwise, a is an arc added in one of the new recursive partitions. One of the components of $\mathcal{T}_c - a$ is a subtree of a newly added recursive partition. At most four vertices of $C(G)$ are incident to edges of this recursive partition, and therefore $|\sigma(a)| \leq 4$. Because G is not a tree and $n \geq 2$, we have $bw(G) \geq 2$, and therefore have the desired result. □

We note that the construction used in Theorem 5.2.3 does not work for tree graphs: The complete bipartite graph $K_{1,n}$ is a tree and has branchwidth one [76], but the constructed branch decomposition for $C(K_{1,n})$ would have width three.

Theorem 5.2.4. *Let G be a connected planar graph, $n \geq 2$. Then $tw(C(G)) \leq 3tw(G)-1$.*

Proof. If G is not a tree, this follows from Lemma 5.2.1 and Theorem 5.2.3. If G is a tree, then $tw(G) = 1$ and $C(G)$ is a tree along with an additional vertex x^* connected to every vertex of the tree and several loops at x^*. A tree decomposition of width two can be constructed by taking a tree decomposition of G of width one and adding x^* to every bag of the decomposition. □

We also establish lower bounds for $tw(C(G))$ and $bw(C(G))$. (In fact, our result for the branchwidth is tight.) For graphs G and H, the *Cartesian product* $G\Box H$ is a graph

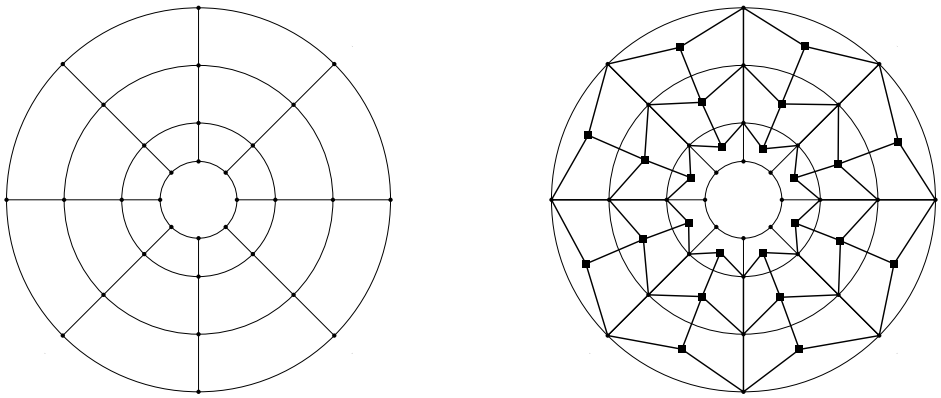


Figure 5.4: A cylinder graph $Z_{8,4}$ (left), along with a cylinder graph $Z_{16,3}$ shown as a subgraph of $C(Z_{8,4})$ (right).

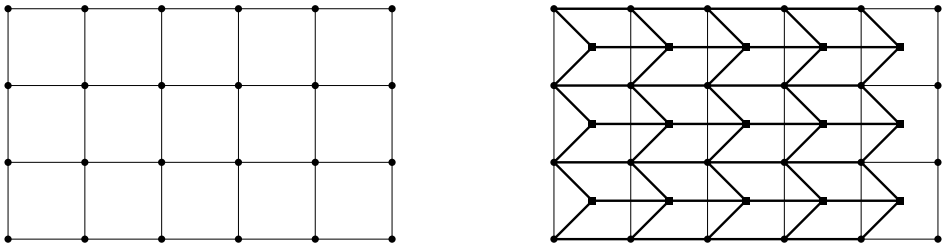


Figure 5.5: A grid graph $R_{4,7}$ (left), along with a grid graph $R_{7,6}$ shown as a subgraph of $C(R_{4,7})$.

with vertex set $V(G) \times V(H)$. Two vertices (g_1, h_1) and (g_2, h_2) of $G \square H$ are defined to be adjacent if $g_1 = g_2$ and $h_1 h_2 \in E(H)$, or $g_1 g_2 \in E(G)$ and $h_1 = h_2$. The cylinder graph $Z_{k,h}$ is defined to be the Cartesian product of a cycle on k vertices and a path on h vertices. In other words, $Z_{k,h}$ consists of h cycles of length k, connected in concentric circles via k paths of length h. (See Figure 5.4(left).) The grid graph $R_{k,h}$ is defined to be the Cartesian product of a path on k vertices and a path on h vertices. The graph $R_{k,h}$ can be seen as a $k \times h$ integer grid with all of the horizontal and vertical lines. (See Figure 5.5(left).)

Theorem 5.2.5. *For every $k \geq 1$ there is a graph G_k of treewidth k such that $C(G_k)$ has treewidth at least $2k-1$. Moreover, there is a graph H_k of branchwidth k such that $C(H_k)$ has branchwidth $2k$.*

Proof. It is known that $R_{k,h}$ has treewidth $\min\{k, h\}$ [75]. We take $G_k := R_{k,2k}$ and have $tw(G_k) = \min\{k, 2k\} = k$. We can see in Figure 5.5(right) that the coupled graph $C(R_{a,b})$

contains as a subgraph $R_{2a-1,b-1}$. Thus $C(G_k)$ contains $R_{2k-1,2k-1}$ as a subgraph, and $tw(C(G_k)) \geq 2k-1$.

It is known that a cylinder graph $Z_{k,h}$ has branchwidth $\min\{k, 2h\}$ [39]. We take $H_k := Z_{k,k+1}$, and have $bw(H_k) = \min\{k, 2k+2\} = k$. As can be seen in Figure 5.4(right), $C(H_k)$ contains $Z_{2k,k}$ as a subgraph, and therefore has branchwidth at least $2k$. By Theorem 5.2.3, we have $bw(C(H_k)) = 2k$. □

This demonstrates that Theorem 5.2.3 is tight. However, we do not know if Theorem 5.2.4 is tight, and we suspect its bound could be improved.

5.3 Coupled Graphs and Optimal 1-Planar Graphs

Recall that in this chapter we are studying coupled list coloring of planar graphs, or equivalently, list coloring the graph $C(G)$ for some planar graph G. By Lemma 2.2.4 $C(G)$ is 1-planar, but it has more structure as we will explain here. Recall that an optimal 1-planar graph is a simple 1-planar graph with $4n-8$ edges. Bodendiek, Schumacher, and Wagner [18] proved that every simple optimal 1-planar graph is the coupled graph of some planar graph. In particular, they define an *almost elementary planar graph* to be a simple planar graph in which the boundary of every face is a cycle, and any two faces share at most one edge on their boundary. Equivalently, almost elementary planar graphs are those simple planar graphs that are 2-vertex-connected and 3-edge-connected. Observe that these are exactly the simple planar graphs whose coupled graph is simple—any two faces in an almost elementary planar graph will share at most one edge, and as there are no cut vertices and no bridges there are no parallel intermediate edges and no loops in the dual graph. They then prove the following.

Theorem 5.3.1 (Bodendiek et al. [18])**.** *Let C be a (simple) graph. Then C is an optimal 1-planar graph if and only if there is an almost elementary planar graph G such that $C(G) = C$.*

The main objective of their paper was the coupled chromatic number of planar graphs, so their colorings did not use lists. They argued that this coupled chromatic number is maximized for graphs that are almost elementary, and they therefore did not need to consider planar graphs whose coupled graph is not simple. Part of their argument for this involved splitting a planar graph that is not almost elementary into almost elementary subgraphs. These subgraphs are separately colored, and the colorings are then combined by

permuting colors. This strategy would not work for list coloring, and therefore we cannot make the same assumption that our planar graphs are almost elementary, and hence cannot assume that the coupled graph is simple.

In this section we explore the structure of all coupled graphs. As optimal 1-planar graphs are defined to be simple, we generalize the concept to 1-planar multigraphs. Recall that a (≥ 3)-cell drawing is a graph drawing in which every cell has at least three corners.

Definition 5.3.2. An *optimal 1-planar multigraph* is a 1-planar multigraph with exactly $4n - 8$ edges that has a 1-planar (≥ 3)-cell drawing.

Observe that optimal 1-planar multigraphs are a superset of simple optimal 1-planar graphs, since any simple 1-planar graph trivially has a (≥ 3)-cell drawing. We now show that optimal 1-planar multigraphs are exactly the coupled graphs. Hence, what we prove is a generalization of Theorem 5.3.1. Moreover, as Bodendiek et al.'s paper is only available in German, this also is (as far as the author knows) the first time a proof of this result has appeared in English.

We first prove that a coupled graph always has exactly $4n - 8$ edges. This was proved for simple graphs in [18], but a different proof via the Euler characteristic shows that it holds even if there are bigons or loops.

Lemma 5.3.3. *Let G be a plane multigraph. Then $C(G)$ has $4|V(C(G))| - 8$ edges.*

Proof. By definition, $C(G)$ has $|V(C(G))| = |V(G)| + |F(G)| = |E(G)| + 2$ vertices by the Euler characteristic. We now count the number of edges in $C(G)$. We have $|E(G)|$ primal edges of G, and $|E(G^*)| = |E(G)|$ dual edges of G^*. A primal vertex x is incident to exactly $d_G(x)$ intermediate edges (recall that we add parallel edges to $C(G)$ if a vertex is incident to a face more than once). Therefore $C(G)$ has $\sum_{x \in V(G)} d_G(x) = 2|E(G)|$ intermediate edges. Then in total, $C(G)$ has $4|E(G)| = 4|V(C(G))| - 8$ edges, as desired. □

We proved in Lemma 2.2.4 that every coupled graph has a 1-planar (≥ 3)-cell drawing. Therefore, by Lemmas 2.2.4 and 5.3.3, every coupled graph is an optimal 1-planar multigraph.

Before showing the other direction, we first demonstrate that several known results for simple 1-planar graphs hold for 1-planar multigraphs with a (≥ 3)-cell drawing. (Indeed, the proof strategies themselves generalize.)

Lemma 5.3.4. *Let C be a 1-plane multigraph with a (good) (≥ 3)-cell drawing Γ_C, $n \geq 3$. Then the following hold:*

1. *C has at most $n-2$ crossings.*

2. *Let H be the graph obtained from C by removing one edge from each crossing pair of C. Then the induced drawing of H is a planar (≥ 3)-cell drawing.*

3. *C has at most $4n-8$ edges. In case of equality, C has exactly $n-2$ crossing points.*

4. *If C has exactly $4n-8$ edges, then C can be obtained from a plane multiquadrangulation Q by adding a pair of crossed edges to every face of Q.*

Proof. (1) We recreate the proof by Czap and Hudak [24] of this result for simple 1-planar graphs, here generalized to 1-planar multigraphs with a (≥ 3)-cell drawing. By Lemma 2.2.3, let C^+ be a planar maximal supergraph of C obtained by adding edges. Because C has a (≥ 3)-cell drawing, the drawing of C^+ is also a (≥ 3)-cell drawing. Note that C^+ has the same number of crossings as C. Let c be the number of crossings of C. The planarization $(C^+)^\times$ of C^+ therefore has $n+c$ vertices. Because every edge crosses at most one other edge, no two vertices of $V((C^+)^\times) \setminus V(C^+)$ are adjacent.

Because C^+ is planar maximal and has a (≥ 3)-cell drawing, every cell of C^+ has exactly three corners. Therefore all faces of $(C^+)^\times$ are of degree three and $(C^+)^\times$ is a plane multitriangulation. By Lemma 2.2.2, $(C^+)^\times$ has exactly $2|V((C^+)^\times)| - 4$ faces.

By construction we have $|V((C^\times)^+)| = |V(C^\times)| = n+c$. Let x be a vertex of $(C^\times)^+$ resulting from a crossing point. We know x has degree four and is incident to exactly four distinct faces of $(C^\times)^+$. Moreover, we know that these faces are triangles. As x is not adjacent to any vertex resulting from a crossing point, we have that every face of $(C^\times)^+$ has at most one such vertex on its boundary. Therefore we have $|F((C^\times)^+)| \geq 4c$, which implies $2(n+c) - 4 \geq 4c$, which implies $n - 2 \geq c$, as desired.

(2) Again, we recreate the proof from Czap and Hudak [24]. Let Γ_H be the drawing of H induced by Γ_C. Clearly H is planar since we removed one edge from each crossing pair. It remains to show that Γ_H is a (≥ 3)-cell drawing. Suppose some face f of Γ_H has only one vertex v or two vertices v_1, v_2. This face f would have resulted from merging some cells of Γ_C since Γ_C is a (≥ 3)-cell drawing. So some edge e of C crossed an edge e' on the boundary of f, where e' is either a loop vv or an edge v_1v_2. As e has only one crossing, one endpoint of e must have been v or some v_i. But Γ_C is a good drawing, and edges that share an endpoint do not cross, so this is a contradiction. Therefore Γ_H is a (≥ 3)-cell drawing.

Let H be as in (2) for the remainder of this proof.

(3) By Lemma 2.2.2, H has at most $3n-6$ edges, and so C has at most $(3n-6)+(n-2) = 4n-8$ edges since there are at most $n-2$ edges in $C-H$ by (1). Equality can hold only if $C-H$ has $n-2$ edges, so there were exactly $n-2$ crossings.

(4) As in the proof of (3), we know that H has exactly $3n-6$ edges. Then by Lemma 2.2.2, H must be a plane multitriangulation and has exactly $2n-4$ faces.

Let A be the set of edges removed from C to construct H. Each edge of A is in a crossing pair with an edge of H; let $B \subseteq E(H)$ be the set of these edges. As H is a plane multitriangulation and C has a good 1-planar drawing, no face of H can contain more than one edge-side of B on its boundary (otherwise some edge $e \in A$ would either intersect two different edges or intersect some edge with which it shares an endpoint). Moreover, by $|B| = n-2$ and $|F(H)| = 2n-4$, we determine that every face of H has exactly one edge-side of B on its boundary.

Let $Q = H - B$. For each edge $e \in B$, there were two incident faces f_1, f_2 in H (necessarily $f_1 \neq f_2$ as f_1 and f_2 can have at most one edge side of e on its boundary) which became one face in Q. The faces f_1 and f_2 each had three edge sides on their boundary, but two of those edge sides were those of e. Therefore the face f has $3+3-2 = 4$ edges sides and is a quadrangle. Since all faces of H have an edge side of B on their boundary, all faces of Q are quadrangles and so Q is a plane multiquadrangulation. By definition, C can be constructed from Q by adding a pair of crossed edges to every face of Q. □

We now prove the main result of this section.

Theorem 5.3.5. *Let C be a multigraph. Then $C = C(G)$ for some planar graph G if and only if C is an optimal 1-planar multigraph.*

Proof. ($\Rightarrow$) This follows from Lemmas 2.2.4 and 5.3.3.

($\Leftarrow$) By Lemma 5.3.4(4), let Q be a plane multiquadrangulation such that C can be constructed from Q by adding a pair of crossed edges to every face of Q. The graph Q is bipartite, and we let P, D be the partition of Q. Consider some quadrangle f of Q. Walking along the boundary of f we have vertices z_0, z_1, z_2, z_3 in clockwise order (possibly $z_0 = z_2$ or $z_1 = z_3$). In C, there is a crossing pair with edges z_0z_2 and z_1z_3. As z_i is adjacent to z_{i+1} (addition modulo 4), assuming $z_0 \in P$ it must be that $z_0, z_2 \in P$ and $z_1z_3 \in D$. Hence, every crossing pair of C consists of one edge of $C[P]$ and one edge of $C[D]$. Define $G := C[P]$. The graph G contains exactly one edge from every crossing pair of C, and therefore G is a planar graph. We observe that for any vertex $d \in D$, the neighbors of d that are in P form a circuit: Let $N \subseteq P$ be the neighbors of d that are in P; enumerated as as $p_0, \ldots, p_{k-1}$ in their cyclic order around d (possibly $p_i = p_j$ for $i \neq j$). The vertices

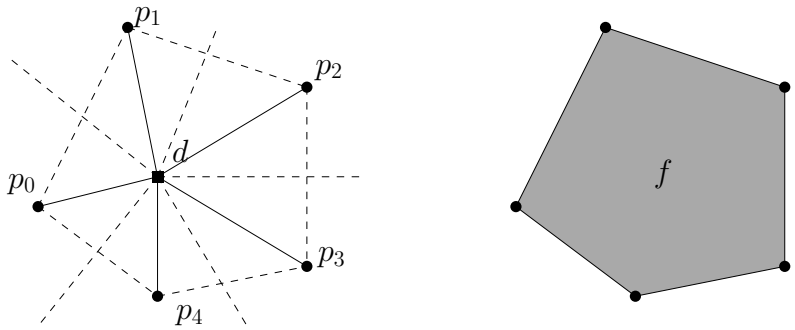


Figure 5.6: A vertex $d \in D$ in Q (left), along with its neighbors. From construction of Q, each quadrangle adjacent to d corresponds to a crossing pair of C (shown with dotted edges) which we use to find the face f of G (right).

p_i, d, p_{i+1} (addition modulo k) form an angle in a quadrangle of Q, and hence $p_i p_{i+1}$ is an edge of C. Therefore there is a circuit $p_0, \ldots, p_{k-1}$. (See Figure 5.6(left).)

In order for $C(G)$ to be meaningful, we must show that the graph G is connected. It suffices to show that for any vertices $p, p' \in P$, there is a walk between p and p' in C that only uses vertices of P. Let $\pi := \langle p{=}x_1, x_2, \ldots, x_{k-1}, x_k{=}p' \rangle$ be a walk in C between p and p' that uses the minimum possible number of vertices of D. Suppose for contradiction that $x_i \in D$ for some $2 \leq i \leq k-1$, and assume that $x_{i-1} \in P$. (It may be that $x_{i+1} \in D$.) We know that the neighbors of x_i that are in P form a circuit ρ. Using ρ, we can locally re-route π to go from x_i to x_{i+1} using only vertices from P: This is clear if $x_{i+1} \in P$, as then x_{i+1} is in ρ. Otherwise, the edge $x_{i+1}x_i$ is in a crossing pair with an edge qq' where q and q' are in ρ, and we have that q is adjacent to both x_i and x_{i+1}. Hence we reroute π by going from x_i to q using vertices in ρ, and then going from q to x_{i+1}. After this re-routing, we obtain a walk π' from p to p', and π' uses less vertices from D than π. But π was assumed to use the minimum number of vertices of D, and so this is a contradiction. Therefore π uses no vertices from D, and G is connected.

Fix the drawing of G induced by the drawing of C. We now show that $C(G) = C$. Let $d \in D$ and $N \subseteq P$ be the neighbors of d that are in P. As argued above, the neighbors of d that are in P form a circuit. Moreover, as d was connected to all vertices in this circuit, the circuit forms the boundary of a face f of G. (See Figure 5.6(right).) The face f is uniquely defined from d, and we therefore have that $V(C) = P \cup D \subseteq V(C(G))$.

The edges dp_i correspond to the intermediate edges added for the face-vertex-incidences between f and p_i. Finally, for each angle $\{p_i, d, p_{i+1}\}$, the corresponding quadrangle of Q contains a fourth angle at some $d' \in D$ (possibly $d' = d$), and the edge dd' inside this

quadrangle is the dual edge of p_ip_{i+1}, so C contains all edges of $C(G)$. We therefore have that $E(C(G)) \subseteq E(C)$ and $V(C) \subseteq V(C(G))$. Moreover, as both graphs are optimal 1-planar multigraphs and have $4n-8$ edges, we conclude that the two graphs are equal. □

In conclusion, any statements that are true for coupled graphs are therefore true for optimal 1-planar multigraphs. In particular for this thesis, any result for coupled list coloring is equivalently a result for list coloring optimal 1-planar multigraphs. For example, Theorem 5.1.6 could be viewed as a result for a subclass of optimal 1-planar multigraphs.

5.4 Coupled List Coloring Subgraphs of Wheels

In this section we investigate the coupled list chromatic number of subgraphs of wheel graphs, and show that the corresponding coloring can be found in linear time. We do this with an eye towards coupled list coloring planar partial 3-trees (we discuss this connection further in Section 5.5). It was already known that all planar partial 2-trees (such graphs are K_4-minor free) are 5-coupled choosable [45, 90], and we moreover proved in Theorem 5.1.6 that this coloring can be found in linear time. In order to prove the desired result for all subgraphs of wheels, we first determine the coupled list chromatic number of the wheel graph.

The *wheel graph* W_n, $n \geq 4$, is a plane graph constructed by taking a cycle Y_{n-1} on $n-1$ vertices, placing a new vertex x_0 inside of the cycle, and adding an edge from x_0 to every vertex of Y_{n-1}. We will label the outer face of the wheel graph as f_0. We further label the vertices in the cycle Y_{n-1} as $x_1, \ldots, x_{n-1}$, and label the inner faces as $f_1, \ldots, f_{n-1}$ such that x_i is incident to f_i and f_{i+1} for $1 \leq i < n-1$, and x_{n-1} is adjacent to f_{n-1} and f_1. It is easy to see that a wheel graph W_n has treewidth at most three: The cycle Y_{n-1} has treewidth two. We can take a tree decomposition of Y_{n-1} of width two and then add x_0 to the bag of every node in the decomposition, giving us a tree decomposition of W_n of width three. We aim to show the following:

Theorem 5.4.1. *Every wheel graph W_n, $n \geq 4$, is 5-coupled choosable, and the coloring can be found in linear time.*

Proof. For $n = 4$, W_4 is the complete graph K_4, Wang and Lih [90] proved that $\chi^L_{vf}(K_4) = 4$. So we assume $n \geq 5$. Let L be a 5-coupled list assignment for W_n. Our goal is to find an L-coupled coloring of W_n. In the coupled graph $C(W_n)$, x_0 and f_0 are adjacent to all other vertices. We will color them first and then color the rest of $C(W_n)$. We define

$X_n := C(W_n) \setminus \{x_0, f_0\}$. Observe that $|V(X_n)| = 2n - 2$. (See Figure 5.7.) When coloring x_0 and f_0, we have two cases.

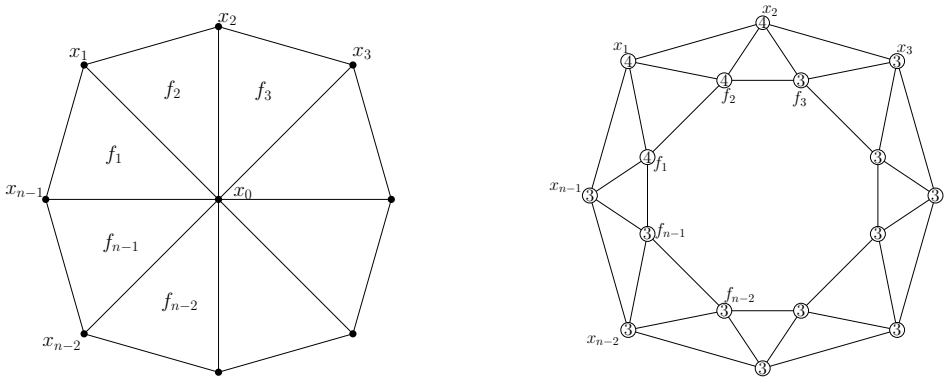


Figure 5.7: The graph W_9 (left) and X_9 (right). Circled numbers indicate a lower bound on the list lengths in L' defined in the cases below.

Case 1: $L(x_0) \cap L(f_0) \neq \emptyset$. Pick a color $a \in L(x_0) \cap L(f_0)$ and assign color a to x_0 and f_0. We cannot use the color a for any vertex in X_n, and therefore define lists L' for X_n, given by $L'(y) := L(y) \setminus \{a\}$ for every vertex $y \in V(X_n)$. We see that all lists of L' are of size at least four.

Case 2: $L(x_0) \cap L(f_0) = \emptyset$. We find suitable colors for x_0 and f_0 by imitating the method used for K_4 in [90] (adapted here to five colors). We define a set of color pairs $S := \{\{a, b\} : a \in L(x_0), b \in L(f_0)\}$. By case-assumption $|S| = 25$. For any $y \in V(X_n)$ define $S_y := \{s \in S : s \subseteq L(y)\}$. We claim that $|S_y| \leq 6$. To see this, consider the disjoint sub-lists $L_1 := L(y) \cap L(x_0)$ and $L_2 := L(y) \cap L(f_0)$. Observe that $S_y = L_1 \times L_2$. Moreover, $|L_1| + |L_2| \leq 5$, and so by integrality we have

$$|S_y| = |L_1 \times L_2| = |L_1| \cdot |L_2| \leq 6.$$

Therefore, color pairs of S appear as subsets of lists in X_n at most

$$\sum_{y \in V(X_n)} |S_y| \leq (2n - 2) \cdot 6 = 12n - 12$$

times. By $|S| = 25$, some element $\{a', b'\} \in S$ appears at most

$$\frac{12n-12}{25} < \frac{n-1}{2}$$

times as a subset of a list in X_n. Color x_0 with a' and f_0 with b'. We cannot use a' or b' for any vertex in X_n, and therefore define lists L' for X_n with $L'(y) := L(y) \setminus \{a', b'\}$ for all $y \in V(X_n)$. For a vertex y in X_n, we have that $3 \leq |L'(y)| \leq 5$. We call a vertex y a *3-vertex* if $|L'(y)| = 3$, i.e., $\{a', b'\} \subseteq L(y)$. From our choice of colors a' and b', we have

$$\frac{|\{y \in V(X_n) : y \text{ is a 3-vertex}\}|}{|V(X_n)|} < \frac{(n-1)/2}{2n-2} = \frac{1}{4}$$

This ends the description for how to color x_0 and f_0. In both cases, we have chosen their colors such that more than three quarters of the vertices of X_n are 4-vertices, i.e., have at least 4 colors in the list-assignment L' obtained by removing the colors of x_0 and f_0 everywhere. Consider the cyclic enumeration

$$\sigma := \langle f_1, x_1, f_2, x_2, \ldots, f_{n-1}, x_{n-1} \rangle$$

of the vertices of X_n. Since more than three quarters of the vertices of X_n are 4-vertices, there must be four consecutive 4-vertices in σ. Up to exchanging f_i and x_i and renumbering, we may assume that f_1, x_1, f_2, and x_2 are 4-vertices. See Figure 5.7(right) for an illustration of the list sizes in L'.

We next color f_{n-1} and x_1. We wish to color them such that at most one color from $L(f_1)$ is used. If $L'(f_{n-1}) \cap L'(x_1) \neq \emptyset$, then color f_{n-1} and x_1 with the same color. Otherwise, we have $|L'(f_{n-1}) \cup L'(x_1)| \geq 7 > |L(f_1)|$. Then there must be colors p and q for f_{n-1} and x_1 respectively such that at most one of them is in $L(f_1)$.

Two vertices adjacent to x_{n-1} have now been colored, and $|L'(x_{n-1})| \geq 3$, so x_{n-1} still has at least one valid color left, and we pick this color for x_{n-1}. We now have colors p, q, and r for f_{n-1}, x_1, and x_{n-1} (respectively) such that $|L'(f_1) \setminus \{p, q, r\}| \geq 2$. Removing these colors from the lists of their neighbors produces new lists L'' such that

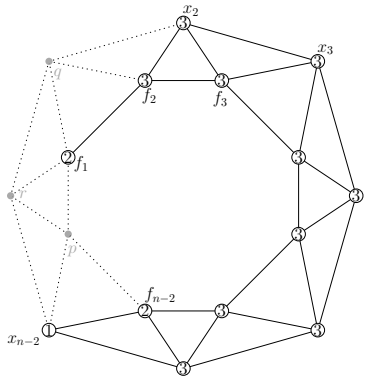


Figure 5.8: The graph X_9. Circled numbers indicate a lower bound on the list lengths in L''. Solid edges show the graph X_9'.

$$
\begin{aligned}
|L''(f_1)| &= |L'(f_1) \setminus \{p, q, r\}| \geq 4 - 2 = 2 \\
|L''(f_2)| &= |L'(f_2) \setminus \{q\}| \geq 4 - 1 = 3 \\
|L''(x_2)| &= |L'(x_2) \setminus \{q\}| \geq 4 - 1 = 3 \\
|L''(x_{n-2})| &= |L'(x_{n-2}) \setminus \{p, r\}| \geq 3 - 2 = 1 \\
|L''(f_{n-2})| &= |L'(f_{n-2}) \setminus \{p\}| \geq 3 - 1 = 2 \\
|L''(x_i)| &\geq 3 \quad (\text{for all } 3 \leq i \leq n-3) \\
|L''(f_i)| &\geq 3 \quad (\text{for all } 3 \leq i \leq n-3)
\end{aligned}
$$

Figure 5.8 illustrates these lower bounds on the list lengths in L''. We then color the rest of the graph with respect to L''. We argue that this is always feasible. Define $X_n' := X_n \setminus \{f_{n-1}, x_{n-1}, x_1\}$ to be the graph of the vertices that remain to be colored (shown in solid edges in Figure 5.8). We show that X_n' has an L''-coloring, and do so via Theorem 2.3.1. As such we must first define a vertex order for X_n':

$$\langle x_{n-2}, f_{n-2}, x_{n-3}, f_{n-3}, \ldots, x_2, f_2, f_1 \rangle .$$

The vertex x_{n-2} trivially has zero *predecessors* (neighbors proceeding it in this order), f_{n-2} has one predecessor (namely x_{n-2}), and f_1 only has one predecessor (namely f_2). Any other vertex of X_n' has two exactly predecessors. With this vertex ordering and the known lower bounds on the size of the lists in L'' given above, by Theorem 2.3.1, we can find an L''-coloring for X_n' in linear time. We therefore have a list coloring of X_n that is compatible

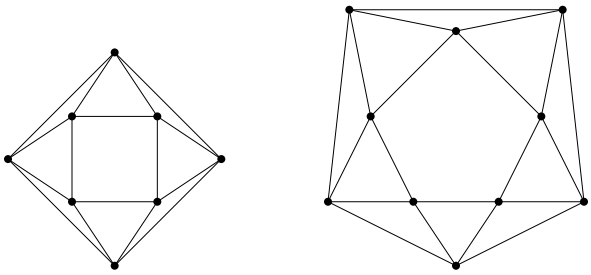


Figure 5.9: The graphs X_5 and X_6.

with the colors chosen for x_0 and f_0 chosen earlier, and so we have a coupled list coloring of W_n.

Following the steps of this proof immediately leads to a linear time algorithm to find the coloring. □

In [90], it was shown that $K_4 = W_4$ is 4-coupled choosable. One might wonder whether our result of 5-coupled choosability can be improved for any other wheel graphs. As we show now, this is not the case and W_4 is the only wheel graph that is 4-coupled choosable.

Theorem 5.4.2. $\chi^L_{vf}(W_n) = 5$ *for* $n \geq 5$.

Proof. From Theorem 5.4.1, we know that all wheel graphs are 5-coupled choosable. It remains to show that they are not 4-coupled choosable for $n \geq 5$.

For $n = 5, 6$, we consider the coupled list assignment L such that $L(y) = \{1, 2, 3, 4\}$ for every $y \in V(W_n) \cup F(W_n)$. (So these graphs are not even 4-coupled colorable. It would be possible to verify this using the characterization of 4-coupled colorability [13], but it is easier to give a direct proof.) Assume for contradiction that we have an L-coupled coloring c of W_n. If $c(x_0) \neq c(f_0)$, then this leaves two colors for coloring the triangle x_1, f_1, f_2 in X_n, impossible. Hence $c(x_0) = c(f_0)$, say they are both colored 4. Then we have an L'-coloring of X_n with lists $L'(y) := L(y) \setminus \{4\} = \{1, 2, 3\}$.

Observe that for X_5 and X_6 (illustrated in Figure 5.9), any putative L'-coloring would be unique up to renaming the colors, since once we have colored one triangle, every other vertex can be reached via a sequence of triangles. One verifies that for these graphs (and indeed every X_k where $k - 1$ is not divisible by 3), attempting such a 3-coloring leads to a contradiction. This proves the theorem for $n = 5, 6$.

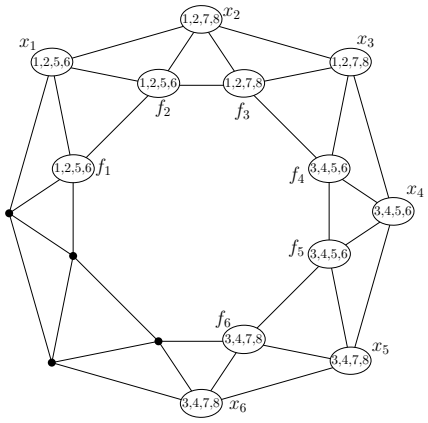


Figure 5.10: The graph X_9. Some of the vertices have been labelled with the 4-list assignment defined in the proof of Theorem 5.4.2.

For $n \geq 7$, we construct a list assignment L such that W_n is not L-coupled choosable. Set $L(x_0) = \{1, 2, 3, 4\}$ and $L(f_0) = \{5, 6, 7, 8\}$. We further define:

$$L(f_1) = L(x_1) = L(f_2) = \{1, 2, 5, 6\}$$
$$L(x_2) = L(f_3) = L(x_3) = \{1, 2, 7, 8\}$$
$$L(f_4) = L(x_4) = L(f_5) = \{3, 4, 5, 6\}$$
$$L(x_5) = L(f_6) = L(x_6) = \{3, 4, 7, 8\}$$

Observe that each of these triples forms a triangle in X_n, and for any $a \in \{1, 2, 3, 4\}$ and $b \in \{5, 6, 7, 8\}$, one of these triangles has colors $\{a, b, x, y\}$ for some colors x, y. (See Figure 5.10.) Assume for contradiction that we have an L-coupled coloring c of W_n. Up to symmetry, assume $c(x_0) = 1$ and $c(f_0) = 5$. But then f_1, x_1, and f_2 have only two colors left, and therefore cannot be colored, a contradiction. □

With this, the coupled choosability of wheel graphs is completely characterized.

Theorem 5.4.3. *For a wheel graph W_n, we have $\chi^L_{vf}(W_n) = \min\{5, n\}$.*

We now return to the subject of graphs that are subgraphs of wheels. As demonstrated in Theorem 5.1.2, non-trivial work is required to show that any subgraph of a wheel graph is 5-coupled choosable. However, the result comes quickly using results proved earlier in this chapter.

Theorem 5.4.4. *Let G be a subgraph of a wheel graph W_n, $n \geq 4$. Then G is 5-coupled choosable and the coloring can be found in linear time.*

Proof. We examine several possibilities of the structure of G.

Case 1: G is the result of deleting at least one vertex of W_n, or one edge of W_n that is on the outer face. Then G is outerplanar and therefore K_4-minor free, and so by Theorem 5.1.6 G is 5-coupled choosable and the coloring can be found in linear time.

Case 2: Otherwise. Then all vertices of W_n belong to G, but we deleted zero or more edges not on the outer face. These are edges that are incident to the center vertex (call them *spokes*). If at most two spokes remain, then G has at most 3 faces and therefore is K_4-minor free, and hence is 5-coupled choosable by Theorem 5.1.6. If $k \geq 3$ spokes remain, then G is a subdivision of W_k and we can find the subdivision vertices in linear time by scanning for vertices of degree two. By Theorem 5.4.1 and Lemma 5.1.3, G is 5-coupled choosable, and we can find the coloring in linear time. □

5.5 Towards Planar Partial 3-Trees

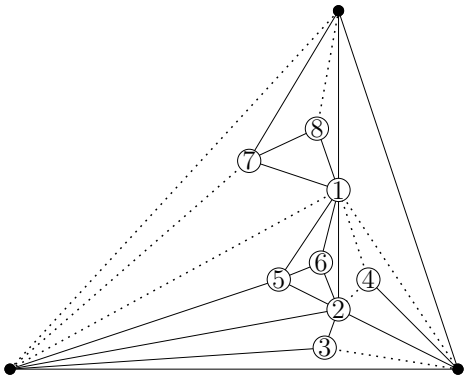


Figure 5.11: A planar partial 3-tree. Dotted edges show the Apollonian network G. We can see that G is an Apollonian network: Start with the triangle on the outer face. Stellate this triangle with the vertex labeled with 1. Then stellate one of the new faces with the vertex labeled with 2. Continue this process with the vertices in the order they are labeled.

Our investigation of wheel graphs was motivated by wanting to determine the coupled choosability number of planar partial 3-trees. We have given their definition via treewidth, but there is also a more intuitive definition using *Apollonian networks*. Apollonian networks are defined recursively as follows. A triangle is an Apollonian network. If G is an Apollonian network, and f is a face of G (necessarily a triangle) that is not the outer-face, then the

graph obtained by stellating face f is also an Apollonian network. (See Figure 5.11.) Any planar partial 3-tree can be obtained as the subgraph of an Apollonian network [16].

Any Apollonian network is a maximal planar graph, and these graphs are known to be 6-coupled choosable [90]. This does not automatically imply 6-coupled choosability of planar partial 3-trees due to Theorem 5.1.2, and we therefore offer the following conjecture:

Conjecture 5.5.1. *Every planar partial 3-tree is 6-coupled choosable.*

Towards this conjecture, we examine several classes of planar partial 3-trees that also generalize wheel graphs. One such class of graphs is the class of *Halin graphs*, which are defined by starting with a tree T and adding a cycle between the leaves of T. See also the solid edges in Figure 5.12. Wheel graphs are the special case of Halin graphs where the tree T is a star graph. A second class of planar partial 3-trees are the *stellated outerplanar graphs*, obtained by starting with some outerplanar graph G and stellating the outer face. See also the dashed edges in Figure 5.12. Wheel graphs are the special case of stellated outerplanar graphs where the outerplanar graph is a cycle. One can easily see that Halin graphs are exactly the duals of stellated outerplanar graphs. Therefore, any coupled list coloring of a stellated outerplanar graph corresponds to a coupled list coloring of a Halin graph. Unfortunately, our upper bound for the coupled choosability of wheel graphs does not in general extend to Halin graphs.

Theorem 5.5.2. *There exists a stellated outerplanar graph (equivalently a Halin graph) that is not 5-coupled colorable (in particular therefore it is not 5-coupled choosable).*

Proof. The Halin-graph G is the triangular prism, see Figure 5.12 where we also show the dual graph G^* and the coupled graph $C(G)$. The claim holds if we show that there is no 5-coloring of the vertices of $C(G)$.

Assume for contradiction that $C(G)$ had a 5-coloring; up to renaming of colors we may assume that the triangle formed by the three quadrangles of G is colored $1, 2, 3$. Let (t, t') be the edge that crosses the edge colored with 2 and 3. Vertices t, t' are colored with 1, 4 or 5; up to renaming of colors 4 and 5 hence one of them is colored 4.

Starting with these three vertices colored 2,3,4, propagate restrictions on the possible colors to other vertices of $C(G)$ along the numerous copies of K_4 (note that all vertices other than t, t' are adjacent to the one colored 1). This leads to a triangle that has only two possible colors left, a contradiction. □

In particular, this shows that we cannot replace "6" with "5" in Conjecture 5.5.1.

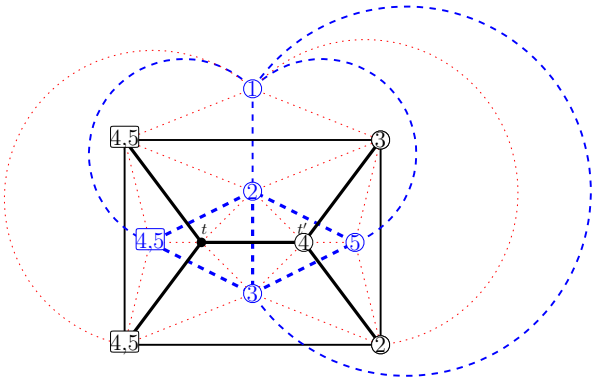


Figure 5.12: A Halin-graph G (black solid; the tree is bold), and the dual graph G^* (blue dashed) which is a stellated outerplanar graph (the outerplanar graph is bold). Taking both, and adding the intermediate edges (red dotted) gives graph $C(G)$.

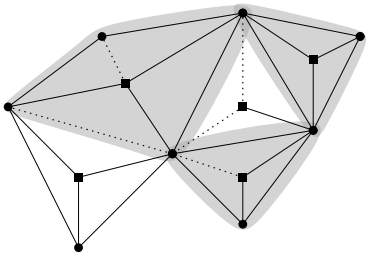


Figure 5.13: An IO graph G consists of an outerplanar graph (circles) and an independent set (squares). Dotted edges are added to obtain G^+, and some of the wheels used to build G^+ are shaded.

We now consider a second graph class that lies somewhere between wheels and planar partial 3-trees. These are *IO graphs*, which are planar graphs that can be obtained by adding an independent set to the interior faces of an outerplanar graph. (See Figure 5.13). Certainly any subgraph of a wheel is an IO graph.

Conjecture 5.5.3. *Every IO graph is 5-coupled choosable.*

We studied subgraphs of wheel graphs because they may be an important stepping stone towards Conjecture 5.5.3. In particular, consider some IO graph G obtained from an outerplanar graph O and independent set I. Let G^+ be a maximal IO graph containing G, i.e., add edges to G for as long as the result is simple and an IO graph. Then G^+ is a *tree of wheels*, where each wheel consists of a vertex $x \in I$ with its neighbours, and the wheels have been glued together at edges such that the gluing incidences form a tree. Correspondingly

G is a tree of subgraphs of wheels. It may be possible to use Theorem 5.4.4 (enhanced with further restrictions on the coloring of some parts) to prove Conjecture 5.5.3 by building a coloring of G incrementally in this tree, but this remains future work.

Chapter 6

Complexity of List Coloring

In this chapter we give complexity results for both list coloring 1-planar graphs and coupled list coloring planar graphs. Many of these results can be proved by extending previously known complexity results to 1-planar graphs. Clearly ListColoring and 3-ListColoring are NP-hard in 1-planar graphs (because already 3-ListColoring is NP-hard in bipartite 1-planar graphs, see Theorem 3.3.5). For intractable problems such as this, it is common to instead consider the complexity in terms of both the size of the input and in terms of some *parameter* of the input. For example, it is well known that finding a vertex cover of size k is an NP-complete problem. However, for fixed k, a simple recursive algorithm for finding a vertex cover runs in time $O(2^k \cdot n)$ [33], i.e., in linear time with respect to the size of the graph. This type of complexity analysis is known *parameterized complexity* (precise definitions will be given in Section 6.1).

In this chapter we mainly consider two such parameters for our analysis: When studying k-ListColoring, a natural parameter to consider is this integer k. We find that the problem remains intractable under this parameter. The other parameter we consider is the treewidth of the input graph. Many NP-complete problems, such as determining the chromatic number of a graph or finding a maximum size independent set, are known to be solvable in polynomial time on graphs with small treewidth [8]. We find that list coloring problems are solvable in polynomial time for graphs of constant treewidth, although they are still hard when parameterized by treewidth.

We recall here our Theorem 5.3.5, in particular that for any optimal 1-planar multigraph C there is a planar graph G such that $C(G) = C$. Therefore, results in this chapter for coupled list coloring are equivalently results for list coloring optimal 1-planar multigraphs.

6.1 Definitions

We assume familiarity with the complexity classes P, NP, and coNP, as well as reductions and hardness, but review here some complexity classes from the field of parameterized complexity. In parameterized complexity, we consider the running time of an algorithm in terms of both the size of the input and a *parameter* of the input. For a computational problem Q, a *parameterization* is a computable function κ that, for any input x to Q, outputs a positive integer $\kappa(x)$. We call the pair $\langle Q, \kappa \rangle$ a *parameterized problem*. For a positive integer $k \in \mathbb{N}$, we can obtain the parameterized problem Q_k by restricting Q to inputs x where $\kappa(x) = k$. The problem Q_k is known as a *slice* of $\langle Q, \kappa \rangle$. We say that an algorithm $\mathcal{A}$ is *fixed parameter tractable with respect to* κ if there is a polynomial $p : \mathbb{N} \to \mathbb{N}$ and computable function $f : \mathbb{N} \to \mathbb{N}$ such that on input x, $\mathcal{A}$ runs in time $O(f(\kappa(x)) \cdot p(|x|))$. We also call $\mathcal{A}$ an *fpt-algorithm*. A parameterized problem $\langle Q, \kappa \rangle$ is *fixed parameter tractable* if there is an algorithm for Q that is fixed parameter tractable with respect to κ. We also say that Q is *fixed parameter tractable with respect to* κ. We define *FPT* to be the complexity class of fixed parameter tractable problems.

One can think of FPT as the parameterized equivalent to the complexity class P. In this way, there are several parameterized complexity classes one can consider to be an equivalent to NP. We consider three such classes here. One is the class *XP*, which are parameterized problems that can be solved in time $O(|x|^{f(\kappa(x))})$ for any input x. The slices of a problem in XP are solvable in polynomial time. However, in contrast to FPT, the degree of this polynomial depends depends on the value of the parameter. Another class is *para-NP*, which are parameterized problems which can be solved in time $O(f(\kappa(x)) \cdot p(|x|))$ by a non-deterministic Turing machine. The slices of a problem in para-NP are in NP. We also consider (but do not define here) the class *W[1]*, which consists of parameterized problems solvable by a certain constrained class of boolean circuit. (See [33] for more details.) When proving hardness results for parameterized complexity classes, one considers *fpt-reductions*, i.e., reductions that can be computed with an fpt-algorithm.

The relationship between these parameterized complexity classes is shown in Figure 6.1. It is known that FPT is contained in W[1], that W[1] is contained in para-NP and XP, and that the classes para-NP and XP are not comparable [33]. It is conjectured that FPT $\neq$ W[1] and that FPT $\neq$ para-NP, and hence a problem being W[1]-hard or para-NP-hard is evidence that there is no fpt-algorithm for that problem.

Lastly, we recall the complexity class Π_2^p from the polynomial hierarchy. This class consists of all problems that are in coNP when given an oracle for some NP-complete problem. For hardness results for Π_2^p, one considers polynomial-time many-one reductions

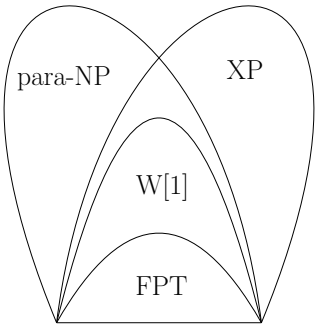


Figure 6.1: The relationship between the parameterized complexity classes para-NP, XP, W[1], and FPT. Figure is based on [33].

(the same as used when proving problems to be NP-hard). The class NP is contained in Π_2^p, and so if a problem is Π_2^p-hard it is also NP-hard. It is conjectured that $\Pi_2^p \neq$ NP [36], so a problem being Π_2^p-hard is evidence that it is not in NP.

6.2 Para-NP-Hard problems

When considering the parameterized complexity of k-list coloring, it is natural to examine the complexity when parameterized by the integer k, i.e., when parameterized by the size of the lists. In this section we show that list coloring 1-planar graphs with respect to this parameter is hard, even when list coloring optimal 1-planar multigraphs. The following theorem reduces proving that a parameterized problem is para-NP-hard to examining the slices of the problem.

Theorem 6.2.1. *[33] Let $\langle Q, \kappa \rangle$ be a parameterized problem. If for some $k \in \mathbb{N}$ the slice Q_k is NP-hard, then $\langle Q, \kappa \rangle$ is para-NP-hard.*

With this, k-ListColoring is para-NP-hard with respect to k even in bipartite 1-planar graphs by Theorem 3.3.5. To show the result for optimal 1-planar multigraphs, we need the following:

Theorem 6.2.2. 3-CoupledListColoring *is NP-hard. Equivalently,* 3-ListColoring *is NP-hard even for optimal 1-planar multigraphs.*

Proof. We reduce from 3-Coloring planar graphs, which is known to be NP-hard [37]. Let G be a planar graph. We construct a 3-coupled list assignment L for G such that G is

L-coupled colorable if and only if G is 3-colorable. For $x \in V(G)$, define $L(x) = \{1, 2, 3\}$. It remains to assign lists for the dual vertices of G^*. As G^* is a planar graph, we compute a 4-coloring c^* of G^* using the colors $\{4, 5, 6, 7\}$ [5]. Then for each $x \in V(G^*)$, pick $L(x)$ such that $|L(x)| = 3$, $|L(x)| \subseteq \{4, 5, 6, 7\}$, and $c^*(x) \in L(x)$.

Suppose that G is 3-colorable with coloring c. Then the coloring ϕ given by

$$\phi(x) := \begin{cases} c(x) & \text{if } x \in V(G) \\ c^*(x) & \text{if } x \in V(G^*) \end{cases}$$

is a valid L-coupled coloring of G.

Suppose that G is L-coupled colorable with coloring ϕ. Then since ϕ only uses colors in $\{1, 2, 3\}$ for the primal vertices of G, ϕ also defines a valid 3-coloring of G. □

Combining this with Theorem 6.2.1 gives the following.

Theorem 6.2.3. *k-CoupledListColoring is para-NP-hard when parameterized by k. Equivalently, k-ListColoring is para-NP-hard even for optimal 1-planar multigraphs.*

Note that although we have shown that k-CoupledListColoring planar graphs and that k-ListColoring 1-planar graphs are NP-hard, these problems are both solvable in polynomial time for large enough k: k-CoupledListColoring is always possible for $k \geq 7$ [90], and k-ListColoring 1-planar graphs is always possible for $k \geq 8$ (Corollary 2.3.2). Flum and Grohe observe that para-NP-complete problems are "in some sense, uninteresting from the parameterized point of view" [33], since a problem need only be hard for one parameter value in order to be para-NP-hard.

We will discuss the complexity of k-Choosability and k-CoupledChoosability for fixed values of k in Section 6.6.

6.3 Problems in XP

We now turn to considering the complexity of list coloring 1-planar graphs when parameterized by the treewidth of the input graph, and first give some positive results. Jansen and Scheffler give a polynomial time algorithm for list coloring graphs with bounded treewidth.

Theorem 6.3.1 (Jansen and Scheffler [50])**.** *Let G be a graph and L be a list assignment for G. Let t be the treewidth of G and $\mathcal{L}$ be the set of colors used in L. Then an L-coloring of G (if it exists) can be found in time $O(n \cdot |\mathcal{L}|^{t+1})$.*

For any vertex x, $L(x)$ need not be larger than $d(x)+1$ by Theorem 2.3.1. Hence, we can assume that

$$|\mathcal{L}| \leq \sum_{x \in V(G)} (d(x)+1) = O(m+n) = O(n^2).$$

Therefore, the algorithm above can be reanalyzed as running in time $n^{O(t)}$. This is polynomial if t is constant, but the algorithm is not an fpt-algorithm. It is, however, an XP algorithm.

Theorem 6.3.2. *The following problems are in XP when parameterized by the treewidth of the input graph:*

- ListColoring *1-planar graphs*
- CoupledListColoring *planar graphs.*

Proof. The first result holds immediately by the above discussion. To see that the second holds, recall that coupled list coloring a planar graph G is the same as list coloring $C(G)$, and we proved that $tw(C(G)) \in O(tw(G))$ in Theorem 5.2.4. □

6.4 W[1]-Hard Problems

In this section we show that one cannot hope for an algorithm better than that described by Jansen and Scheffler, specifically an fpt-algorithm, unless FPT=W[1]. Namely, Fellows, Fomin, Lokshtanov, Rosamond, Saurabh, Szeider, and Thomassen [31] prove that ListColoring is W[1]-hard for general graphs.

Theorem 6.4.1 (Fellows et al. [32])**.** ListColoring *is W[1]-hard when parameterized by treewidth. Furthermore, it remains W[1]-hard even in bipartite graphs where all vertices in one partition have lists of length two.*

We recreate their proof here, as the second claim was not explicitly stated in their paper.

Proof. Fellows et al. reduce from MulticolorClique: given a graph G with a proper k-coloring, test whether there exists a clique of size k in G. This problem is known to be W[1]-hard when parameterized by the integer $k \geq 3$ [31] (the problem is trivial for $k \leq 2$). From G, they construct a graph G' with treewidth at most $k-1$ and list assignment L such

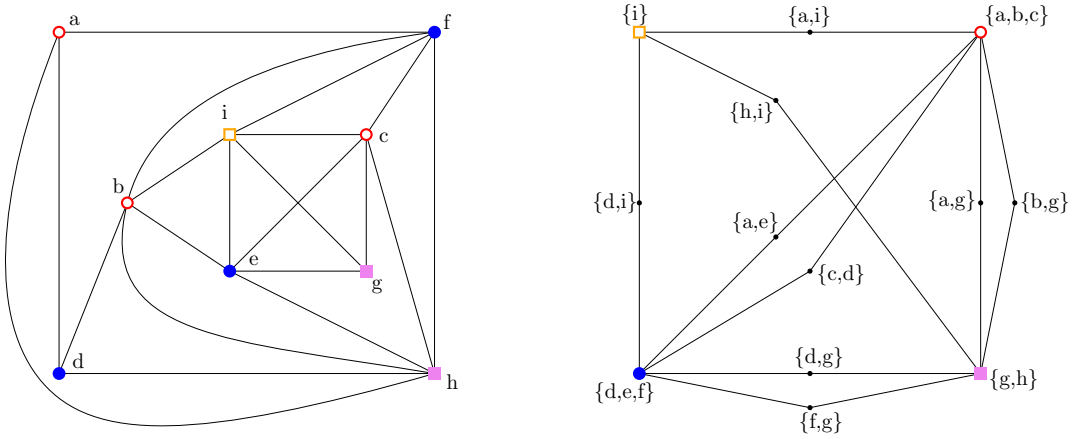


Figure 6.2: A graph with a proper 4-coloring (left), with the constructed instance of list coloring (right). On the left vertices are labeled with $a, b, c, \ldots$, and color classes are denoted by the shape and color of the vertex. Figure is based on [32].

that G has a multicolor k-clique if and only if G' is L-colorable. Let $c : V(G) \to \{1, \ldots, k\}$ be the given proper k-coloring of G.

The graph G' is constructed as follows: Begin with k nodes $v_1, \ldots, v_k$. (We use *nodes* for G' to distinguish from the vertices of G.) Assign $L(v_i) = \{x \in V(G) : c(x) = i\}$. (Note that we are using vertices of G for the colors in G'.) Consider every pair x, y of non-adjacent vertices of G such that $c(x) \neq c(y)$. Add to G' a node u adjacent to $v_{c(x)}$ and $v_{c(y)}$ and assign $L(u) := \{x, y\}$. (See Figure 6.2.) This is an fpt-reduction, as it can be constructed in polynomial time in terms of both k and the size of G.

The graph G' is bipartite, with one side the nodes $v_1, \ldots, v_k$, and the other side the degree two vertices added for non-adjacent pairs of vertices of G. All these vertices of degree two were assigned lists of size two.

We now demonstrate that G' has treewidth at most $k-1$ by constructing a tree decomposition. Start with a tree-node t^* with bag $\{v_1, \ldots, v_k\}$. Then for every node u adjacent to v_i and v_j, add a new tree-node t adjacent to t^* with bag $\{u, v_i, v_j\}$. Since $k \geq 3$, the tree-node t^* has the largest bag, being of size k, and therefore the decomposition has width $k - 1$.

It remains to show that G has a multicolor clique if and only if G' is L-colorable. Suppose that G has a multicolor clique $x_1, \ldots, x_k$ such that $c(x_i) = i$. We compute an L-coloring ϕ of G'. Begin by assigning $\phi(v_i) := x_i$. Let u be a node of G' not in $v_1, \ldots, v_k$. So $L(u) = \{x, y\}$ for two non-adjacent vertices x, y of G, and u is adjacent to nodes $v_{c(x)}$

and $v_{c(y)}$ of G'. Since x and y are not adjacent, at most one of them belongs to $x_1, \ldots, x_k$. Assume $y \notin \{x_1, \ldots, x_k\}$. Then $\phi(v_{c(y)}) \neq y$, so we can pick $\phi(u) = y$.

Suppose that G' has an L-coloring ϕ. Take $\phi(v_1), \ldots, \phi(v_k)$ to be our k-clique of G. By construction, the vertex $\phi(v_i)$ of G has color i. Consider two vertices $\phi(v_i)$ and $\phi(v_j)$ of G. Suppose that they were non-adjacent. Then by construction of G' there would be a node u adjacent to v_i and v_j with $L(u) = \{\phi(v_i), \phi(v_j)\}$. But then u must be colored the same as one of its neighbors, and ϕ could not be a valid coloring of G', a contradiction. Therefore, the vertices $\phi(v_1), \ldots, \phi(v_k)$ are all pairwise adjacent, and all of a different color class, and therefore form a multicolor clique of G. □

We note that this construction cannot be used to prove that k-ListColoring is W[1]-hard when parameterized by treewidth: As stated in the theorem, many lists constructed in the reduction have size two, but the lists that are not of size two can have unbounded size.

We now use Theorem 6.4.1 to prove a stronger result: List coloring is W[1]-hard, even when the graph is 1-planar and bipartite. We will do so by modifying the graph constructed in the proof to be 1-planar while maintaining that the graph is bipartite. The graph constructed for the reduction in Theorem 6.4.1 is not always 1-planar, for if the graph is sparse then the construction will contain K_k with all edges subdivided, which has $k + \binom{k}{2}$ vertices but $\Theta(k^4)$ crossings, and hence is not 1-planar.

Theorem 6.4.2. ListColoring *is W[1]-hard when parameterized by treewidth, even for bipartite 1-planar graphs.*

Proof. We reduce from MulticolorClique. Let G be a graph given with a proper k-coloring for some $k \geq 3$. From Theorem 6.4.1, we construct a graph G' with list assignment L such that G has a multicolor clique if and only if G' has an L-coloring. We know that G' is bipartite, and that every edge of G' is incident to a vertex with a list of size two. Recall that we showed in Lemma 3.3.4 that, because we have such a list assignment, we can subdivide each edge of G' an even number of times to obtain a 1-planar graph G'' with list assignment L' such that G' has an L-coloring if and only if G'' has an L'-coloring, and we can construct G'' and L' in polynomial time. By the construction of G'', we have that G'' is also bipartite, and that the treewidth of G'' is not greater than the treewidth of G'. Therefore, G'' is a bipartite 1-planar graph with treewidth at most $k - 1$, and G'' has an L'-coloring if and only if G has a multicolor clique. This is an fpt-reduction, as G'' can be constructed from G in polynomial time in terms of k and the size of G. □

W[1]-hardness is evidence that there is no fpt-algorithm with respect to treewidth for ListColoring, even when the graph is 1-planar and bipartite.

As the graph G'' constructed in the reduction of Theorem 6.4.2 is 1-planar and bipartite, it has at most $3n - 8$ edges [53]. Therefore G'' is not an optimal 1-planar graph. Also, while it is easy to go from an arbitrary drawing to a 1-planar drawing by subdividing edges, we do not know how to remove crossings altogether without increasing the treewidth. We therefore give the following open questions:

Open Problem 6.4.3. *Are the following problems W[1]-hard when parameterized by treewidth?*

- k-ListColoring *in arbitrary graphs.*
- CoupledListColoring *planar graphs. (Equivalently,* ListColoring *optimal 1-planar multigraphs.)*
- ListColoring *planar graphs.*

6.5 Problems in FPT

Theorem 6.4.2 shows that ListColoring, when parameterized by treewidth, is (likely) not in FPT. Surprisingly, k-Choosability (which at first glance may seem harder than ListColoring) is actually easier. Fellows et al. proved the following:

Theorem 6.5.1 (Fellows et al. [32])**.** k-Choosability *is in FPT when parameterized by treewidth.*

The proof uses Courcelle's theorem [23], which in particular implies that the algorithm runs in linear time for graphs of bounded treewidth. From Theorem 6.5.1, we have the following corollary.

Corollary 6.5.2. *The following problems are in FPT when parameterized by the treewidth of the input graph:*

- k-Choosability *for 1-planar graphs, and*
- k-CoupledChoosability *for planar graphs.*

Proof. The first result is immediate from Theorem 6.5.1. The latter holds as testing whether a planar graph G is k-coupled choosable is the same as testing if the coupled graph $C(G)$ is k-choosable, and we proved that $tw(C(G)) \in O(tw(G))$ in Theorem 5.2.4. □

In fact, we can obtain a slightly stronger result.

Theorem 6.5.3. *The following can be computed in linear time:*

1. *The list chromatic number of a 1-planar graph G of treewidth at most t, and*
2. *the coupled list chromatic number of a planar graph G' of treewidth at most t.*

Proof. By Corollary 2.3.2 we know that every 1-planar graph is 8-choosable. We also know that all planar graphs are 7-coupled choosable [90]. Hence, for (1), we run the algorithm from Corollary 6.5.2 for $k = 1, \ldots, 8$, and for (2) we run the same algorithm for $k = 1, \ldots, 7$, and in both cases return the smallest k that is accepted. In both cases we are running a linear time algorithm a constant number of times, and hence this algorithm is also linear time. □

Because of the large constants associated with Courcelle's theorem, the associated algorithms here are not necessarily practical.

6.6 Π_2^p-Complete Problems

We conclude this section with an unparameterized hardness result for k-Choosability. Gutner [41] proves that 4-Choosability is not only NP-hard in planar graphs, but it is hard in the superclass Π_2^p. Since planar graphs are 1-planar, 4-Choosability is Π_2^p-hard also for 1-planar graphs. But 5-Choosability is in P for planar graphs (as all planar graphs are 5-choosable [82]). But what is the status of 5-Choosability in 1-planar graphs? We first show that k-Choosability and k-CoupledChoosability are both in Π_2^p. (This result was likely known before, but we provide a proof here as a warm-up to the class Π_2^p.)

Lemma 6.6.1. *k-Choosability and k-CoupledChoosability are in Π_2^p.*

Proof. We only show this for k-Choosability; the other result follows by reducing to k-Choosability in the coupled graph. We must show that it is polynomial to prove that

a graph G is not k-choosable, given the ability to guess and an oracle for an NP-complete problem. Here we guess a k-list assignment L such that G is not L-colorable, and then use an oracle for k-ListColoring, which is NP-hard and clearly in NP. □

Towards proving that 5-Choosability is Π_2^p-hard, we first need a helper lemma.

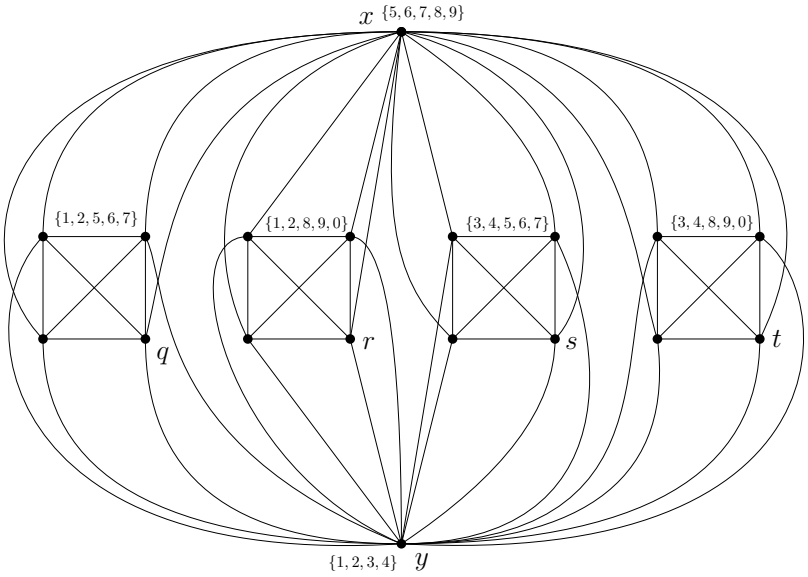


Figure 6.3: A 1-planar graph G constructed by taking four disjoint K_4s, and adding two vertices adjacent to every vertex in one of the K_4s. We also give a list assignment for G where every list is of size 5, except for one list which is of size 4.

Lemma 6.6.2. *There exists a 1-planar graph G with 18 vertices that is 5-choosable, and moreover there is a vertex y of G and a list assignment L for G such that $|L(y)| = 4$, $|L(u)| = 5$ for every vertex $u \neq y$, and G is not L-choosable.*

Proof. Let G be the graph in Figure 6.3 consisting of four K_4s and two vertices x, y adjacent to all vertices of the K_4s. We first argue that G is 5-choosable with an argument similar to the one used in Theorem 5.4.1. Let L be a 5-list assignment for G. If $L(x) \cap L(y) \neq \emptyset$, we pick the same color $c \in L(x) \cap L(y)$ for x and y. After removing c from the lists of every vertex in the disjoint K_4s, every list is of size at least four. Because K_4 is 3-degenerate, it is 4-choosable and a feasible coloring can be found.

If $L(x)$ and $L(y)$ are disjoint, then define $S = \{\{a, b\} : a \in L(x), b \in L(y)\}$. By case assumption, $|S| = 25$. For a vertex u, define $S_u := \{s \in S : s \subseteq L(u)\}$ to be the color-pairs

of S that are a subset of the list of u. As in Theorem 5.4.1, we have $|S_u| \leq 6$ by $|L(u)| = 5$. Consider the vertices q, r, s, t in the figure. By $|S_u| \leq 6$, elements of S appear at most $6 \cdot 4 = 24$ times in S_q, S_r, S_s, and S_t. By $|S| = 25$, some element $\{a, b\}$ of S is not in $S_q \cup S_r \cup S_s \cup S_t$. Choose these colors a, b for x and y and remove a, b from the lists of all other vertices. It remains to color the four disjoint K_4s.

By our choice of colors for x and y, the vertices q, r, s, t each have four colors in their lists. For any other vertex $u \neq q, r, s, t, x, y$, there are still three colors available. Therefore, each of the K_4s has one vertex with at least four colors available, while the other three vertices of the K_4 have at least three colors available. It is easy to see that with such a list assignment for K_4 we can find a vertex ordering satisfying the statement of Theorem 2.3.1: Let v_1, v_2, v_3 be the vertices with lists of size three, and let v_4 be the vertex with a list of size four. Therefore such a list assignment for K_4 always has a feasible coloring, and so G is 5-choosable.

We finally show that there is a list assignment L for G such that $|L(y)| = 4$, $|L(u)| = 5$ for every vertex $u \neq y$ of G, and that G is not L-choosable. Let L be the list assignment shown in Figure 6.3 (all vertices within one copy of K_4 have the same list). This list assignment was constructed such that any choice of colors for x and y would cause one of the disjoint K_4s to have three colors remaining at all vertices. It is impossible to color this K_4, and hence G is not L-choosable. □

Theorem 6.6.3. 5-Choosability *is* Π_2^p*-complete for 1-planar graphs.*

Proof. By Lemma 6.6.1, we know that 5-Choosability is in Π_2^p. To prove hardness, we reduce from 4-Choosability for planar graphs, which is known to be Π_2^p-complete [41]. Let H be a planar graph for which we wish to determine 4-choosability. From H we construct a 1-planar graph H^+: for every vertex u of H, we add a copy of the graph G from Lemma 6.6.2, and an edge uy from u to the vertex y of G. It is easy to see that H^+ is 1-planar. We wish to show that H is 4-choosable if and only if H^+ is 5-choosable.

Suppose that H is 4-choosable. Let L be a 5-list assignment for H^+. As the graph G is 5-choosable, we can find a feasible assignment for each copy of G in the graph H^+. Thus, at any edge uy with $u \in V(H)$ and y in a copy of G, we have colored y and we remove this color from $L(u)$. Therefore, every vertex of H has at least four colors left, and hence a feasible coloring can be found because H is 4-choosable.

Suppose that H^+ is 5-choosable. Let L be a 4-list assignment for H. We first construct a 5-list assignment L^+ for H^+: For each copy of G, we use the list assignment in Figure 6.3. (We assume that these colors are distinct from any of the colors used in L.) Let c be a new color. We add c to $L(y)$ for every copy of the vertex y in a copy of G. We also add c

to the list of every vertex of H. We now have a 5-list assignment L^+ for H^+, and because H^+ is 5-choosable, H^+ is L^+-choosable. In any L^+-coloring of H^+, the copies of the vertex y must use the color c by Lemma 6.6.2. Therefore, no vertex of H can be colored with c, and therefore must use colors from the 4-list assignment L. Therefore H is L-choosable, and since L was an arbitrary 4-assignment, H is 4-choosable. □

For planar graphs, the complexity of k-Choosability has already been completely characterized for all values of k. However, for k-Choosability for 1-planar graphs and also for k-CoupledChoosability (or equivalently, k-Choosability for optimal 1-planar multigraphs), there are still gaps to fill. Table 6.1 shows the known results and open problems here.

k	Planar Graphs	1-Planar Graphs	Optimal 1-Planar Multigraphs
2	P [29]	P [29]	P [29]
3	Π_2^p-complete [41]	Π_2^p-complete [41]	P (Lem. 5.1.1)
4	Π_2^p-complete [41]	Π_2^p-complete [41]	Unknown
5	Always Possible [82]	Π_2^p-complete (Thm. 6.6.3)	Unknown
6		Unknown	Unknown
7		Unknown	Always Possible [90]
8		Always Possible (Cor. 2.3.2)	

Table 6.1: Known results and open problems for the complexity of k-Choosability on planar, 1-planar, and optimal 1-planar multigraphs.

Chapter 7

Conclusion

In this thesis, we studied the list colorability of some classes of 1-planar graphs. We gave a linear time algorithm to 4-list color a 1-planar bipartite graph and a linear time algorithm to 5-coupled list color a wheel graph. To prove linear time for the algorithm to 4-list color a 1-planar bipartite graph, we proved that a 1-planar graph can be edge-partitioned into a planar graph and a forest in linear time. We also gave many complexity results for list coloring 1-planar graphs and coupled list coloring planar graphs.

We conclude here with a few open problems. As noted at the end of Chapter 4, the only step in finding the 4-list coloring of a 1-planar bipartite graph in Chapter 3 that requires the embedding of the graph is finding the partition of the 1-planar bipartite graph into a planar graph and a forest. (This partition was used to orient the edges of the 1-planar bipartite graph with maximum outdegree three.) This embedding must be given as finding a 1-planar embedding of a graph is NP-hard [38]. We posed the question of whether it is possible to find this partition without being given the embedding.

Open Problem 7.0.1. *Can a 1-planar graph be partitioned into a planar graph and a forest in polynomial time without being given an embedding?*

Recall some open problems from Chapter 6.

Open Problem 7.0.2. *Are the following problems W[1]-hard when parameterized by treewidth:*

- k-ListColoring *for arbitrary graphs*
- CoupledListColoring *for planar graphs*

- ListColoring *for planar graphs*

We end this chapter with some open problems around k-CoupledChoosability and k-Choosability. As discussed in Chapter 2, 2-coupled choosability and 3-coupled choosability are easily characterized. 4-coupled choosability seems similarly restrictive, as a "leaf triangle" (a triangular face adjacent to only two other faces) corresponds to a K_5 in the coupled graph. (Recall that by Theorem 5.4.2 wheel graphs W_n, $n \geq 5$, are not 4-coupled choosable and do not have such a triangular face, hence this does not characterize 4-coupled choosability.) Moreover, 4-coupled colorability (where no lists are specified) has been fully characterized [13]. This leads us to ask the following:

Open Problem 7.0.3. *Is there a characterization of 4-coupled choosability that can be tested in polynomial time?*

Such a characterization would likely help in proving our two conjectures from Chapter 5, which we phrase here as open problems.

Open Problem 7.0.4. *Are planar partial 3-trees 6-coupled choosable?*

Open Problem 7.0.5. *Are IO-graphs 5-coupled choosable?*

While Theorem 6.6.3 proves 5-Choosability to be Π_2^p-hard even for 1-planar graphs, we have no comparable results for k-CoupledChoosability. Moreover, recall from Table 6.1 that for $k = 6, 7$ the complexity of k-Choosability for 1-planar graphs is unknown.

Open Problem 7.0.6. *Is k-CoupledChoosability Π_2^p-complete for $k = 4, 5, 6$?*

Open Problem 7.0.7. *Is k-Choosability for 1-planar graph Π_2^p-complete for $k = 6, 7$?*

One can also consider list coloring problems on 1-planar graphs that don't have short cycles. For instance, it is known that planar graphs without cycles of length 3 or 4 are 3-choosable [83]. Can we get better upper bounds for the list chromatic number for triangle-free 1-planar graphs, or 1-planar graphs without cycles of length 3 or 4? Similarly, can we get better upper bounds for the coupled list chromatic number of triangle-free planar graphs?

There are several generalizations of 1-planar graphs for which list coloring problems can be studied. Some possible generalizations are 1-toroidal graphs (graphs that can be drawn on the surface of a torus with at most one crossing per edge) or k-planar graphs

(graphs that can be drawn in the plane with at most k crossings per edge). Alternatively, there are generalizations of list coloring that could be studied for 1-planar graph, such as the Alon-Tarsi number (see e.g. [51]), paintability [78], or correspondence coloring [28]. It is known that planar graphs have Alon-Tarsi number 5 [95] and are 5-paintable [78]. Since 1-planar graphs are 7-degenerate, they are both 8-paintable and have Alon-Tarsi number at most 8, but can these bounds be improved?

References

[1] Eyal Ackerman. A note on 1-planar graphs. *Discrete Applied Mathematics*, 175:104–108, 2014.

[2] Md. Jawaherul Alam, Franz J. Brandenburg, and Stephen G. Kobourov. Straight-line grid drawings of 3-connected 1-planar graphs. In *Graph Drawing*, volume 8242 of *Lecture Notes in Computer Science*, pages 83–94, Cham, 2013. Springer International Publishing.

[3] Noga Alon and Michael Tarsi. Colorings and orientations of graphs. *Combinatorica*, 12(2):125–134, 1992.

[4] Kenneth Appel and Wolfgang Haken. Every planar map is four colorable. Part I: Discharging. *Illinois Journal of Mathematics*, 21(3):429 – 490, 1977.

[5] Kenneth Appel, Wolfgang Haken, et al. Every planar map is four colorable. *Bulletin of the American Mathematical Society*, 82(5):711–712, 1976.

[6] Kenneth Appel, Wolfgang Haken, and John Koch. Every planar map is four colorable. Part II: Reducibility. *Illinois Journal of Mathematics*, 21(3):491–567, 1977.

[7] Dan Archdeacon. Coupled colorings of planar maps. In *Proceedings of the Fourteenth Southeastern Conference on Combinatorics, Graph Theory and Computing (Boca Raton, Fla., 1983)*, volume 39, pages 89–94, 1983.

[8] Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, 1989.

[9] Sam Barr and Therese Biedl. All subgraphs of a wheel are 5-coupled-choosable. In Paola Flocchini and Lucia Moura, editors, *International Workshop on Combinatorial Algorithms (IWOCA)*, volume 12757 of *Lecture Notes in Computer Science*, pages 78–91, Cham, 2021. Springer International Publishing.

[10] Sam Barr and Therese Biedl. Efficiently Partitioning the Edges of a 1-Planar Graph into a Planar Graph and a Forest. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*, volume 212 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[11] Michael A. Bekos, Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Chrysanthi Raftopoulou. Edge partitions of optimal 2-plane and 3-plane graphs. *Discrete Mathematics*, 342(4):1038–1047, 2019.

[12] Claude. Berge. *Graphs and hypergraphs.* North-Holland mathematical library ; v.6. North Holland, Amsterdam, 1973.

[13] Kenneth A. Berman and H. Shank. Full 4-colorings of 4-regular maps. *Journal of Graph Theory*, 3(3):291–294, 1979.

[14] Therese Biedl and Franz J. Brandenburg. Partitions of graphs into trees. In Michael Kaufmann and Dorothea Wagner, editors, *Graph Drawing*, volume 4372 of *Lecture Notes in Computer Science*, pages 430–439. Springer, Berlin, 2007.

[15] Therese Biedl, Anna Lubiw, and Owen Merkel. List coloring bipartite graphs embedded on a surface. Unpublished Manuscript, 2019.

[16] Therese Biedl and L.E. Ruiz Velázquez. Drawing planar 3-trees with given face areas. *Computational Geometry: Theory and Applications*, 46(3):276–285, 2013.

[17] Markus Blumenstock. Fast algorithms for pseudoarboricity. In *Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*, pages 113–126. SIAM, 2016.

[18] R. Bodendiek, H. Schumacher, and K. Wagner. Bemerkungen zu einem Sechsfarbenproblem von G. Ringel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 53:41–52, 1983.

[19] Oleg V. Borodin. Solution of the Ringel problem on vertex-face coloring of planar graphs and coloring of 1-planar graphs. *Metody Diskret. Analiz*, 41(12):12–26, 108, 1984.

[20] Oleg V. Borodin. A new proof of the 6 color theorem. *Journal of Graph Theory*, 19(4):507–521, 1995.

[21] Victor A. Campos, Guilherme C. M. Gomes, Allen Ibiapina, Raul Lopes, Ignasi Sau, and Ana Silva. Coloring problems on bipartite graphs of small diameter. *Electronic Journal of Combinatorics*, 28(2), 2021.

[22] Gregory J. Chaitin. Register allocation & spilling via graph coloring. *ACM Sigplan Notices*, 17(6):98–101, 1982.

[23] Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

[24] Július Czap and Dávid Hudák. On drawings and decompositions of 1-planar graphs. *Electronic Journal of Combinatorics*, 20(2), 2013.

[25] Hubert de Fraysseix and Patrice Ossona de Mendez. Regular orientations, arboricity, and augmentation. In Roberto Tamassia and Ioannis Tollis, editors, *Graph Drawing*, volume 894 of *Lecture Notes in Computer Science*, pages 111–118. Springer, Berlin, 1995.

[26] Emilio Di Giacomo, Walter Didimo, William S. Evans, Giuseppe Liotta, Henk Meijer, Fabrizio Montecchiani, and Stephen K. Wismath. New results on edge partitions of 1-plane graphs. *Theoretical Computer Science*, 713:78–84, 2018.

[27] Christian Doczkal and Damien Pous. Treewidth-two graphs as a free algebra. In Igor Potapov, Paul Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 60:1–60:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[28] Zdeněk Dvořák and Luke Postle. Correspondence coloring and its application to list-coloring planar graphs without cycles of lengths 4 to 8. *Journal of Combinatorial Theory, Series B*, 129:38–54, 2018.

[29] Paul Erdős, Arthur L. Rubin, and Herbert Taylor. Choosability in graphs. In *Proceedings of the West Coast Conference on Combinatorics, Graph Theory and Computing*, Congress. Numer., XXVI, pages 125–157. Utilitas Math., Winnipeg, Man., 1980.

[30] Louis Esperet. Dynamic list coloring of bipartite graphs. *Discrete Applied Mathematics*, 158(17):1963–1965, 2010.

[31] Michael Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410:53–61, 2009.

[32] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, 209(2):143–153, 2011.

[33] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.

[34] A. Frank and A. Gyárfás. How to orient the edges of a graph? In *Combinatorics (Proc. Fifth Hungarian Colloq., Keszthely, 1976), Vol. I*, volume 18 of *Colloq. Math. Soc. János Bolyai*, pages 353–364. North-Holland, Amsterdam-New York, 1978.

[35] Harold N. Gabow and Herbert H. Westermann. Forests, frames, and games: algorithms for matroid sums and applications. *Algorithmica*, 7(5-6):465–497, 1992.

[36] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. A Series of Books in the Mathematical Sciences. W. H. Freeman and Co., San Francisco, Calif., 1979.

[37] Michail R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[38] Alexander Grigoriev and Hans L. Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007.

[39] Qian-Ping Gu and Hisao Tamaki. Improved bounds on the planar branchwidth with respect to the largest grid minor size. *Algorithmica*, 64(3):416–453, 2012.

[40] Jun-Lin Guo and Yue-Li Wang. 3-list-coloring planar graphs of girth 4. *Discrete Mathematics*, 311(6):413–417, 2011.

[41] Shai Gutner. The complexity of planar graph choosability. *Discrete Mathematics*, 159(1):119–130, 1996.

[42] Shai Gutner and Michael Tarsi. Some results on (a:b)-choosability. *Discrete Mathematics*, 309(8):2260–2270, 2009.

[43] S. L. Hakimi. On the degrees of the vertices of a directed graph. *Journal of the Franklin Institute*, 279:290–308, 1965.

[44] W.K. Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

[45] Timothy J. Hetherington. Coupled choosability of near-outerplane graphs. *Ars Combinatoria*, 113:23–32, 2014.

[46] Jacob Holm, Giuseppe F. Italiano, Adam Karczmarz, Jakub Łącki, Eva Rotenberg, and Piotr Sankowski. Contracting a planar graph efficiently. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[47] Seok-Hee Hong and Takeshi Tokuyama. *Beyond Planar Graphs*. Springer Nature, Singapore, 2020.

[48] John Hopcroft and Robert Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[49] Dai-Qiang Hu, Danjun Huang, Weifan Wang, and Jian-Liang Wu. Planar graphs without chordal 6-cycles are 4-choosable. *Discrete Applied Mathematics*, 244:116–123, 2018.

[50] Klaus Jansen and Petra Scheffler. Generalized coloring for tree-like graphs. *Discrete Applied Mathematics*, 75(2):135–155, 1997.

[51] Tommy R. Jensen. *Graph Coloring Problems*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, New York, 1995.

[52] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103, 1972.

[53] D. V. Karpov. An upper bound on the number of edges in an almost planar bipartite graph. *J. Math Sci.*, 196:737–746, 2014.

[54] H.A. Kierstead. On the choosability of complete multipartite graphs with part size three. *Discrete Mathematics*, 211(1):255–259, 2000.

[55] Jon Kleinberg and Éva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., USA, 2005.

[56] Stephen G. Kobourov, Giuseppe Liotta, and Fabrizio Montecchiani. An annotated bibliography on 1-planarity. *Computer Science Review*, 25:49–67, 2017.

[57] Marek Kubale. *Graph Colorings*. Contemporary mathematics. American Mathematical Society, Providence, R.I, 2004.

[58] William J. Lenhart, Giuseppe Liotta, and Fabrizio Montecchiani. On partitioning the edges of 1-plane graphs. *Theoretical Computer Science*, 662:59–65, 2017.

[59] Rhyd Lewis. *A Guide to Graph Colouring Algorithms and Applications*. Springer International Publishing, Cham, 1st edition, 2016.

[60] Xiangwen Li. On 3-choosable planar graphs of girth at least 4. *Discrete Mathematics*, 309(8):2424–2431, 2009.

[61] Dániel Marx. Graph colouring problems and their applications in scheduling. *Periodica Polytechnica Electrical Engineering (Archives)*, 48(1-2):11–16, 2004.

[62] Frédéric Mazoit and Stéphan Thomassé. Branchwidth of graphic matroids. In *Surveys in Combinatorics 2007*, volume 346 of *London Math. Soc. Lecture Note Ser.*, pages 275–286. Cambridge Univ. Press, Cambridge, 2007.

[63] Maryam Mirzakhani. A small non-4-choosable planar graph. *Bull. Inst. Combin. Appl*, 17:15–18, 1996.

[64] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, Md., 2001.

[65] C. St. J. A. Nash-Williams. Decomposition of finite graphs into forests. *The Journal of the London Mathematical Society*, 39:12, 1964.

[66] Takao Nishizeki and Norishige Chiba. *Planar Graphs: Theory and Algorithms*. Elsevier, 1988.

[67] János Pach, Radoš Radoičić, Gábor Tardos, and Géza Tóth. Improving the crossing lemma by finding more crossings in sparse graphs. *Discrete & Computational Geometry*, 36(4):527–552, 2006.

[68] Luke Postle. Linear-time and efficient distributed algorithms for list coloring graphs on surfaces. In *IEEE Symposium on Foundations of Computer Science*, pages 929–941. IEEE Comput. Soc. Press., 2019.

[69] Luke Postle. 3-list-coloring graphs of girth at least five on surfaces. *Journal of Combinatorial Theory, Series B*, 147:1–36, 2021.

[70] Luke Postle and Robin Thomas. Five-list-coloring graphs on surfaces I. Two lists of size two in planar graphs. *Journal of Combinatorial Theory, Series B*, 111:234–241, 2015.

[71] Luke Postle and Robin Thomas. Five-list-coloring graphs on surfaces II. A linear bound for critical graphs in a disk. *Journal of Combinatorial Theory, Series B*, 119:42–65, 2016.

[72] Luke Postle and Robin Thomas. Five-list-coloring graphs on surfaces III. One list of size one and one list of size two. *Journal of Combinatorial Theory, Series B*, 128:1–16, 2018.

[73] Gerhard Ringel. Ein Sechsfarbenproblem auf der Kugel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 29:107–117, 1965.

[74] Neil Robertson, Daniel P. Sanders, Paul Seymour, and Robin Thomas. Efficiently four-coloring planar graphs. In *ACM Symposium on the Theory of Computing*, pages 571–575. ACM, 1996.

[75] Neil Robertson and Paul Seymour. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory. Series B*, 41(1):92–114, 1986.

[76] Neil Robertson and Paul Seymour. Graph minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.

[77] Marcus Schaefer. The graph crossing number and its variants: A survey. *The Electronic Journal of Combinatorics*, 20, 04 2013.

[78] Uwe Schauz. Mr. Paint and Mrs. Correct. *Electronic Journal of Combinatorics*, 16(1), 2009.

[79] Wen-yao Song, Lian-ying Miao, and Shu-jie Zhang. List edge and list total coloring of triangle-free 1-planar graphs. *Journal of East China Normal University Natural Science Edition*, (3):40–44, 2014.

[80] Lin Sun, Guanglong Yu, and Xin Li. Neighbor sum distinguishing total choosability of 1-planar graphs with maximum degree at least 24. *Discrete Mathematics*, 344(1), 2021.

[81] Yusuke Suzuki. Re-embeddings of maximum 1-planar graphs. *SIAM Journal on Discrete Mathematics*, 24(4):1527–1540, 2010.

[82] Carsten Thomassen. Every planar graph is 5-choosable. *Journal of Combinatorial Theory, Series B*, 62(1):180–181, 1994.

[83] Carsten Thomassen. 3-list-coloring planar graphs of girth 5. *Journal of Combinatorial Theory. Series B*, 64(1):101–107, 1995.

[84] Jacobo Valdes, Robert Endre Tarjan, and Eugene L. Lawler. The recognition of series parallel digraphs. *SIAM J. Comput.*, 11(2):298–313, 1982.

[85] V. Venkateswaran. Minimizing maximum indegree. *Discrete Applied Mathematics*, 143(1-3):374–378, 2004.

[86] V. G. Vizing. Coloring the vertices of a graph in prescribed colors. *Akademiya Nauk SSSR. Sibirskoe Otdelenie. Institut Matematiki. Diskretnyĭ Analiz. Sbornik Trudov*, (29, Metody Diskret. Anal. v Teorii Kodov i Shem):3–10, 101, 1976.

[87] Margit Voigt. List colourings of planar graphs. *Discrete Mathematics*, 120(1):215–219, 1993.

[88] Margit Voigt. A not 3-choosable planar graph without 3-cycles. *Discrete Mathematics*, 146(1):325–328, 1995.

[89] Weifan Wang and Ko-Wei Lih. Choosability and edge choosability of planar graphs without five cycles. *Applied Mathematics Letters*, 15(5):561–565, 2002.

[90] Weifan Wang and Ko-Wei Lih. Coupled choosability of plane graphs. *Journal of Graph Theory*, 58(1):27–44, 2008.

[91] Wanshun Yang, Yiqiao Wang, Weifan Wang, and Ko-Wei Lih. IC-planar graphs are 6-choosable. *SIAM Journal on Discrete Mathematics*, 35(3):1729–1745, 2021.

[92] Xin Zhang and Yan Li. Dynamic list coloring of 1-planar graphs. *Discrete Mathematics*, 344(5), 2021.

[93] Xin Zhang, Bei Niu, and Jiguo Yu. A structure of 1-planar graph and its applications to coloring problems. *Graphs Comb.*, 35(3):677–688, 2019.

[94] Xin Zhang, Jianliang Wu, and Guizhen Liu. List edge and list total coloring of 1-planar graphs. *Frontiers of Mathematics in China*, 7(5):1005–1018, 2012.

[95] Xuding Zhu. The Alon–Tarsi number of planar graphs. *Journal of Combinatorial Theory, Series B*, 134:354–358, 2019.