Adversarial and Online Algorithms

by

Matthew Askes

A thesis
submitted to Victoria University of Wellington
in fulfilment of the requirements for the degree of
Master of Science
in Mathematics.

Victoria University of Wellington
2022

Abstract

In this thesis we explore a variety of online and adversarial algorithms. We primarily explore the following online and adversarial algorithms; the perfect code game, adversarial online colouring, the chain decomposition game, and strongly online graphs.

The *perfect code game* is a new adversarial game played on graphs in which players take turns constructing perfect codes. We provide definitions for perfect codes and the perfect code game, along with some motivation from coding theory. We will prove upper bounds for both cycle and path graphs. We also prove an upper bound for graphs of bounded pathwidth. Finally, we explore the perfect code game in graphs of bounded degree.

We will create a new game called the *adversarial online colouring game*, this game takes elements from both online and adversarial algorithms. We will begin with some discussion and a definition of adversarial online colouring. We will then prove several results related to graph degree. We conclude with a proof that the adversarial online colouring game on trees is determined only by the number of colours and the number of vertices (assuming that both players have a chance at winning).

The *chain decomposition game* is another adversarial game, but this time played on partial orders. We introduce the chain decomposition game and demonstrate two results relating to upper and lower bounds for the game. We also prove a result on the online adversarial version of the game.

We introduce *strongly online* graphs and graph colouring as a new algorithmic parameterization to online graph colouring. A strongly online graph is an online graph where at each stage s we can see a ball of increasing radius about each vertex. We will prove several bounds (upper and lower) on the online chromatic number of strongly online graphs. For example, we show that every strongly online graph can be coloured in twice its chromatic number. We prove that every strongly online graph with even pathwidth k can be online coloured with $2k + 1$ colours. Then, after introducing a natural notion of *strongly online pathwidth*, we prove that there is a strongly online graph with no finite strongly online path decomposition.

Contents

List of Figures

Chapter 1

Introduction

1.1 Algorithms

Mathematical algorithms are at the heart of computer science and hence at the heart of modern society. An algorithm is a sequence of basic instructions that define some procedure. This could be something simple such as an algorithm for searching a list, or more complicated like finding an optimal delivery route for a postal service. A 'normal' algorithm has access to the entire problem structure and operates alone. However, there are many problems for which this is not true. For example, if you have a robot navigating a large maze then the robot has no knowledge about the sections that it hasn't explored. In these situations special types of algorithms are needed. It is these special algorithms we are concerned with. This thesis will be dedicated to the theory of adversarial and online algorithms on graphs and related structures. We deduce that most of the problems we study have been not considered before. These problems are the latest in a long history stretching back to Turing 1936 [1] and his work on universal computing machines.

1.1.1 Bin Packing

Consider the *bin packing problem*. Suppose you have a bunch of items of various shapes and sizes along with several bins. Each bin can hold only a limited volume. Your goal is to pack the items into the least number of bins possible. The bin packing problem is a famous NP-complete problem [2]. Given the bin packing problem, how might you solve it? Noting this is an important problem, we explain three common algorithms.

Brute force: Try every possible combination of items and bins. Then take the best one (smallest number of bins). Brute force gives you an optimal result. However, the complexity of this algorithm is $\Omega(2^n)$.

Greedy: Order the items from largest to smallest. Fix some order on the bins. Working from largest to smallest, place each item in the smallest bin that will accept the item. In effect we are greedily picking the biggest item and placing it in a bin. The greedy algorithm has complexity ($O(n \log n)$). The solutions generated using the greedy algorithm use at most $^{11}/_{6}$ times the optimal number of bins [3].

First fit: Fix some order on the bins. Place each item as it arrives in the smallest bin that will accept the item. First fit is a fast algorithm (O(n)) but gives solutions worse than the greedy algorithm. The solutions generated use at most 1.7 times the optimal number of bins [4].

First fit is an example of an *online* algorithm. Suppose you, your family, and some friends are going on holiday. You are in charge of packing the cars. However, no-one is organised and they give you items one at a time. Whenever you are given an item you must irrevocably decide which car to put the item in. You could use the first fit algorithm to pack the cars. You can do this by placing an item is the first car that it will fit in.

1.1.2 Planar Graphs

Another problem we consider is planar graph colouring. A *planar* graph is a graph that can be drawn on the plane without edges crossing. Every planar graph contains a 4-colouring [5]. But, can we algorithmically colour a planar graph with four colours? yes we can, if the graph is finite.

You could try every possible colouring until you found one that worked. But what if the graph was countably infinite? Could we still find a colouring? It turns out we cannot. Specifically we have the following theorem.

Theorem 1.1 (Bean 1976 [6])**.** *There is a 3-colourable, computable*[1] *planar graph which, for all k has no computable k-colouring.*

1.2 Definitions Related to Graphs

We need to take a moment to define some concepts related to graphs.

A *graph* $G = (V, E)$ is a set of vertices $V(G)$ along with a set of edges $E(G)$. An *edge* $(u, v) \in E$ is a pair of vertices. A *simple graph* is a graph with at most 1 edge between each pair of vertices and no loops (edges starting and ending at the same vertex). We will only be considering countable, simple, non-directed graphs. We will refer to both infinite and finite graphs, this should be clear from context. As a rule of thumb, we deal with finite graphs in adversarial situations, and (countably) infinite graphs in online situations.

The *neighbourhood* of a vertex v is the set of all vertices adjacent to v, is denoted $N(v) = \{u \in V : (u, v) \in E\}$. The *closed neighbourhood* of a vertex v is the neighbored of v along with v, and is denoted $N[v] = N(v) \cup \{v\}$. The *degree* of a vertex v is the number of neighbours of v, and is denoted $\delta(v) = |N(v)|$. The degree of a graph is the largest degree of all its vertices and is denoted $\Delta(G) = \max_{v \in V} \delta(v)$. The degree of a graph may also be referred to as just Δ when it is clear which graph we are referring to.

Let $G = (V, E)$ be a graph, and $U \subset V$. The *neighbourhood* of U is the set of all vertices not in U that are adjacent to a vertex in U, and is denoted $N(U) = \{v \in (V \setminus U) : \exists_{u \in U}(u, v) \in E\}$. The *closed neighbourhood* of U is the neighbourhood of U along with U, and is denoted $N[U] = N(U) \cup U$. For H the subgraph of G induced by U we will abuse notation slightly and write $N(H)$ to mean the subgraph of G induced by $N(U)$. We will also write $N[H]$ to mean the subgraph induced by $N[U]$.

[1]We are omitting a formal definition of computable. For an overview see Section 5.1, or Turing 1936 [1]

We formally define a colouring as follows.

Definition 1.2 (Proper k-Vertex Colouring)**.** Let $C = \{1, \dots, k\}$ be a set of colours. A *k-vertex colouring* of a graph $G = (V, E)$ is a mapping $c\colon V \to C$. A *proper k-vertex colouring* of $G = (V, E)$ is a mapping $c\colon V \to C$ such that for any two vertices $u, v \in V$, if $(u, v) \in E$, then $c(u) \neq c(v)$.

When referring to graph colourings we will henceforth be referring to proper k-vertex colourings for some k. A colouring $c : V \to C$ is called *partial* if it is not defined on all the vertices of a graph. That is, c is a partial function.

Definition 1.3 (Chromatic Number)**.** The *chromatic number*, $\chi(G)$, of a graph G is the smallest integer k such that G has a proper k-vertex colouring.

1.3 Adversarial Algorithms

An *adversarial algorithm* (a.k.a. competitive algorithms and games) is a two-player[2] game where the players have differing/conflicting goals. For example, suppose it is Alice's job to manage deliveries for a pâtisserie, Le Petit Flour. Alice's goal is to deliver pastries to every café in her route as quickly possible. That is, she needs to create an efficient delivery schedule. In a non-adversarial situation, Alice would simply have a system or algorithm for determining which café she delivers to next. However, suppose instead she had a partner, Bob, who gets to decide every second delivery. Bob has been bribed by a rival bakery and is attempting to sabotage Le Petit Flour. He sabotages Le Petit Flour by choosing cafés in such a way that the delivery schedule becomes infeasible. This is an adversarial situation and Alice will need a new algorithm that will minimize the damage done by Bob. Such an algorithm would be an adversarial algorithm.

It is useful to think of such adversarial algorithms as a game between two players, Alice and Bob. Alice and Bob would take turns constructing an object (e.g. a function). When it makes it clearer we will use Alice to refer to the algorithm attempting to construct an object, and Bob as the algorithm as the attempting to destroy/hinder the construction. For example, if we are trying to adversarial k-colour a graph, then Alice wants a colouring using at most k colours and Bob trys to force more than k colours.

1.3.1 The Colouring Game

Adversarial games model many structures in computing where we imagine an opponent trying to hinder our progress. One common example of an adversarial situation is the *colouring game*. Let G be a graph, and C a set of colours. Beginning with Alice, Alice and Bob take alternating turns. On their turn they choose an uncoloured vertex, v, and assign v a colour from C such that no two adjacent vertices in G have the same colour. This continues until one of two win conditions are met. First, Alice wins if all the vertices are coloured. Second, Bob wins if there is a vertex that cannot be coloured with the available colours.

Definition 1.4 (Game Chromatic Number)**.** For a graph G the *game chromatic number*, $\chi_g(G)$, is the smallest number of colours such that Alice will always win the colouring

[2]It is possible to have more than two players, but we will only concern ourselves with two player versions.

game on G.

Every finite planar graph has a 4-colouring algorithm. What if we are adversarial colouring the planar graph? Then we would need at least 17 colours to adversarial colour every planar graph (currently, it is not known if this bound is tight).

Theorem 1.5 (Zhu 2008 [7] and Kierstead and Trotter 1994 [8]). *For any planar graph G, it's game chromatic number is $\chi_g(G) \leq 17$. Further, there is a planer graph with game chromatic number 7.*

In Chapter 3 we will explore a further algorithmic parameterization of the colouring game.

1.3.2 General Definition of Adversarial

An *adversarial problem* is a tuple (I, S, q, t) such that I is a finite structure encoded as a string in some alphabet, S is the set of all valid (partial) solutions viewed as strings in some (possibly different) alphabet, $q : S \to \mathbb{N}$ is a function mapping each solution to a score, and $t \in \mathbb{N}$ a target score. For example, in the colouring game, I is a code for a finite graph, and S the set of all codes for valid (partial) colourings on I, $q(s)$ would be the number of colours used by $s \in S$ and t is some target score.

For σ and τ strings encoding some structures, we abuse notion slightly and write $\sigma \prec \tau$ if the structure encoded by τ extends the structure encoded by σ. For example, if σ and τ are graphs then $\sigma \prec \tau$ means σ is a subgraph of τ. Or, if σ and τ are functions then $\sigma \prec \tau$ means σ is τ restricted to σ's domain.

An *algorithm* or *strategy* for an adversarial problem (I, S, q, t) is a computable function $\varphi_I : S \to S$, such that for all $\sigma \in S$, we have $\varphi_I(\sigma) \in S$ and $\sigma \prec \varphi_I(\sigma)$. For example, in the colouring game a strategy is a way of extending any (partial) colouring of I to another colouring of I, such that the new colouring is valid.

A *solution* (a.k.a *winning strategy*) for an adversarial problem (I, S, q, t) is a strategy φ_I such that for any strategy ψ_I the function $\varphi_I \circ \psi_I$ has the following properties,

1. For all $\sigma \in S$, $\varphi_I \circ \psi_I(\sigma) \in S$
2. For all $\sigma \in S$, $\sigma \prec \varphi_I \circ \psi_I(\sigma)$
3. For all $\sigma \in S$, $q(\varphi_I \circ \psi_I(\sigma)) \leq t$

The strategy φ_I can be interpreted as a strategy for Alice, who is attempting to construct some solution σ such that the score of sigma ($q(\sigma)$) is at most t. Because if the score goes above t, then Bob will win. For example, in the colouring game a solution is a strategy for Alice, such that no matter what strategy Bob employs the colourings are always proper and never use more than t colours.

1.4 Online Algorithms

Consider again, Le Petit Flour. Bob has been found out and was promptly fired. In an attempt to repair the damage done by Bob, Alice has a new way of scheduling deliveries. When a café places an order, Alice will irrecoverably assign them a delivery slot. This is an *online* situation because the cafés are arriving in an online manner. Unfortunately for Alice, this means the efficiency of her delivery schedule gets much worse.

An online situation can be considered any situation where a structure is arriving as a sequence of small distinct pieces. Online situations are abound in computer science. They can be found in everything from robot navigation to network configuration. Any situation that has partial information can be considered online.

An online algorithm, **A**, is a strategy that responds to a sequence of inputs such that each output is an extension of the previous and is irrevocable. After seeing t inputs, the algorithm must give the t-th output.

It is useful to consider that the inputs are generated by a malevolent entity. This turns the online situation into a game (again played between Alice and Bob). Alice is responding to pieces of data determined by Bob.

As an example of an online problem we will define online graph colouring along with some results.

1.4.1 Online Colouring

The most common type of online situation we will deal with is online graph colouring.

Definition 1.6 (Online Graph)**.** An *online (presentation of a) graph* $G^{\prec}$ is a countable (possibly infinite) graph $G = (V, E)$ and $\prec$ a linear order on V. The order $\prec$ is called the *presentation order* of G. Let $G_s^{\prec} = (V_s, E_s)$ denote the online subgraph induced by the $\prec$-least s elements $V_s = \{v_1 \prec \cdots \prec v_s\}$ in $G^{\prec}$.

We call the subgraph $G_s^{\prec}$ induced by $\bigcup_{t \preceq s}\{v_t\}$ of G, the *processed* subgraph.

An online algorithm colours the online graph $G^{\prec}$ in stages. At stage s the vertex v_s is revealed and must be irrevocably assigned a colour using only the information in $G_s^{\prec}$. If it is clear from context which graph and ordering we are referring to we may write $G_s^{\prec}$ as G_s.

As we have seen, when adversarial colouring planar graphs, more colours are needed. How many extra colours are need to online colour planar graphs? To online colour a planar graphs with n vertices requires $O(\log(n))$ many colours. Thus an infinite graph would require infinitely may colours to online colour.

1.4.2 General definition of Online

An online structure is some structure A along with a *filtration* (or *online presentation*) $\{A_1, A_2, A_3 \dots\}$ where A_s has universe $\{1, \dots, n\}$ and is the induced substructure of A as identified by $\{1, \dots, s\}$, such that $A = \bigcup_s A_s$.

An *online problem* is a triple (I, S, s) where I is the space of inputs (i.e. the filtration) viewed as strings in some alphabet of some online structure, S is the set of all possible outputs viewed as strings in some (possibly different) alphabet, and s is a function which maps each input to the set of valid solutions.

For example, in online colouring, I will be codes for finite graphs, and S codes for colourings. Then s maps each graph to valid colourings. This leads us to the following definition.

Definition 1.7 (Downey, Melnikov and Ng 2021 [9]). An *online solution* (a.k.a. online algorithm) for an online problem (I, S, s) is a computable function $f\colon I \to S$ with the following properties,

1. For all $\sigma \in S$, $f(\sigma) \in s(\sigma)$
2. If $\sigma \prec \tau$, then $f(\sigma) \prec f(\tau)$

An online algorithm is a function that for every input gives a valid solution, and if one input extends another then so do the corresponding outputs.

1.5 Pathwidth and Treewidth

In general online algorithms on general graphs are very hard. In an attempt to make the problems easier we look for important parameterizations. For both adversarial and online algorithms, various parameterizations about shape are important. Kierstead 1998 [10] and Gasarch 1998 [11] both give ample evidence for this. For example, for a graph treewidth, pathwidth, degeneracy (see Section 3.1 for definition), and maximum degree are all important. This is because they provide measured improvements in online and adversarial algorithms. As we will see in later chapters, graphs of bounded treewidth (or pathwidth) often have good (relative to general graphs) bounds for both adversarial and online algorithms

The concept of treewidth was an important by-product of the famous work by Robertson and Seymour 1986 [12]. Treewidth offers us a useful way of parameterizing problems using graph structure. Fundamentally treewidth is a measure of how 'tree-like' a graph is. So, a graph with treewidth 1 is a tree, and the higher the tree width the less 'tree-like' the graph.

There are many graph parameters that are either equivalent to treewidth or have bounded treewidth. For example, a graph with treewidth k has vertex separation number k and interval thickness $k + 1$. However, we will be focusing primarily on treewidth. For an overview of these alternative parameters, see Bodlaender 1998 [13].

One of the great benefits of treewidth is many hard problems become easier on graphs with bounded treewidth. More specifically, certain NP-complete and NP-hard problems have polynomial solutions for bounded treewidth (but still exponential in treewidth). For example, the following problems have polynomial solutions for bounded treewidth.

Independent set [14].
Find the smallest subset X of V such that no two vertices in X are adjacent.

Vertex cover [15].
Find the smallest subset X of V such that every edge is incident to a vertex in X.

Dominating set [14].
Find the smallest subset X of V such that every vertex is adjacent to a vertex in X.

An algorithm is called *fixed parameter tractable* (FPT) if it's running time is $f(k) \cdot n^{O(1)}$ where f is an arbitrary function depending only on k and n is the input size. Many problems when parameterized by treewidth have FPT algorithms [16]. We note that finding the treewidth of a graph is NP-complete. However if a graph has known pathwidth then there are polynomial algorithms to find tree decompositions [17].

It also turns out that many real life structures have small treewidth. For example, Yamaguchi, Aoki and Mamitsuka 2003 [18] analysed 9712 chemical compounds from the LIGAND database and found that all bar 1 had treewidth 3 or less (the lone exception had treewidth 4). Hence, we find that many real world problems that are hard have efficient algorithms when they are parameterized by treewidth.

1.5.1 Treewidth

We will now formally define treewidth.

Definition 1.8 (Tree decompsition)**.** A *tree decomposition* $(\mathcal{X}, T)$ of a graph $G = (E, V)$ is a tree, T, along with a collection of subsets of V, $\mathcal{X} = \{X_1, \ldots, X_n\}$, indexed by vertices in T such that $\bigcup_i X_i = V$ and X has the following properties,

1. For every edge $(u, v) \in E$, there is a bag X_i such that $u, v \in X_i$.
2. If x,y, and z are vertices in T and z is on the unique path from x to y, then $X_x \cap X_y \subseteq X_z$.

The *width* of a tree decomposition is one less than the maximum size of any bag in $\mathcal{X}$.

Definition 1.9 (Treewidth)**.** The *treewidth* of a graph G is the minimum width of a tree decomposition of G. A graph with *bounded treewidth* k is a graph with treewidth less than or equal to k.

Definition 1.10 (k-tree)**.** A *k-tree* is a graph that has treewidth k and has maximal edges with respect to treewidth.

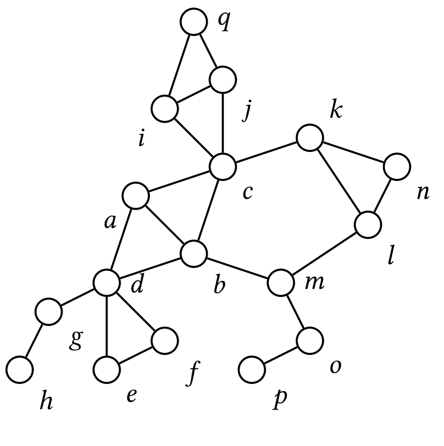


Figure 1.1: A graph of treewidth 2

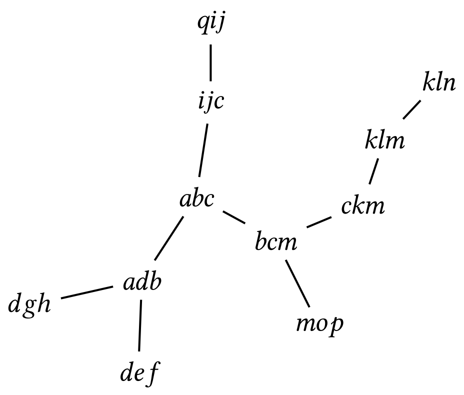


Figure 1.2: A tree decomposition of Figure 1.1

In the offline case, every graph with treewidth k can be $k+1$ coloured. But in the colouring game $3k + 2$ [19] colours are needed. And an online algorithm requires $O(\log(n))$ many colours to colour a graph with n vertices [6].

1.5.2 Pathwidth

Pathwidth is a parameter related to treewidth. While treewidth measures how 'tree-like' a graph is pathwidth measures how 'path-like' a graph is.

Definition 1.11 (Path Decomposition)**.** A *path decomposition* $\mathcal{X} = \{X_1, \ldots, X_n\}$ is a series of bags of vertices such that,

1. Every vertex is in some bag.
2. For every edge (u, v) there is a bag that contains both u and v.
3. If X_i, X_j, X_k are all bags such that $i < j < k$ then $X_j \subseteq (X_i \cap X_k)$.

That is a path decomposition is a tree decomposition where the underlying tree is a path graph.

Definition 1.12 (Pathwidth)**.** The *pathwidth* of a graph G is the minimum width of any path decomposition of G. A graph with *bounded pathwidth* k is a graph with pathwidth less than or equal to k.

Definition 1.13 (k-path)**.** A *k-path* is a graph that has pathwidth k and has maximal edges with respect to pathwidth k.

For example, the graph in Figure 1.3 has the path decomposition (set notation omitted for clarity),

$$ab,\ be,\ bd,\ bg,\ bf,\ bc,\ ch,\ hi,\ hj$$

and the graph in Figure 1.4 has the path decomposition,

$$abcd,\ dceh,\ defh,\ dfgh,\ dghi$$

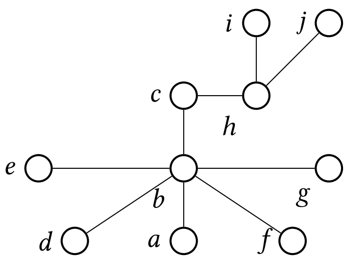


Figure 1.3: A graph of pathwidth 1

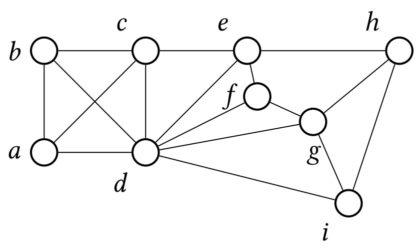


Figure 1.4: A graph of pathwidth 3

Many results for graphs of bounded treewidth have better bounds when restricted to graphs bounded pathwidth. For example, for a graph with pathwidth k,

- The colouring game requires $3k + 1$ colours (see Theorem 1.14).
- Strongly online colouring requires $2k + 1$ colours (if k is even, see Theorem 5.15).
- Online colouring requires $3k + 1$ colours [20].

Interval graphs

Interval graphs are an equivalent characterisation to pathwidth. That is, every graph with pathwidth k is a subgraph of an interval graph with interval width $k+1$.

Interval graphs are graphs defined from a series of closed intervals in the real numbers. A graph $G=(V,E)$ is an *interval graph* if every vertex v_i in V is associated with an interval $(a_i,b_i)\subset\mathbb{R}$ and $(v_i,v_j)\in E$ if and only if the intervals (a_i,b_i) and (a_j,b_j) intersect. For an example see Figure 1.5.

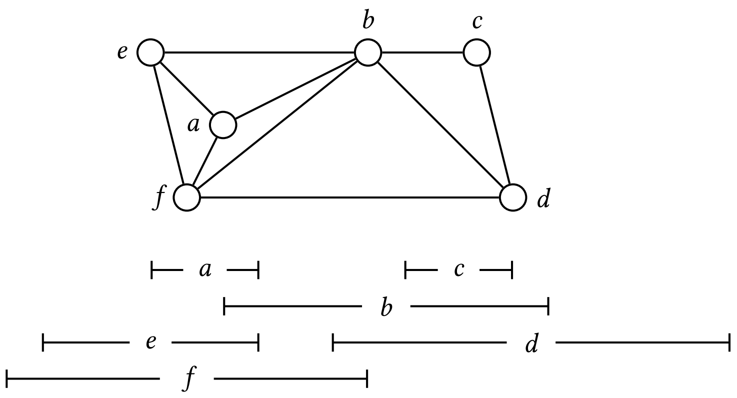


Figure 1.5: An interval graph with clique width 4

The *clique width* of an interval graph is the size of its maximum clique. The *interval width* of a graph G is the minimum clique width of all interval graphs that contain G as a subgraph. It is shown in Faigle et al. 1993 [21] that for any graph with interval width w, we have $\chi_g(G)\leq 3w-2$. Building on this result, Kierstead 2000 [22] uses an algorithm called the Activation Strategy to prove the same result. The pathwidth of a graph is one less than its interval width [13]. This then gives us Theorem 1.14.

Theorem 1.14. *For G be a graph of pathwidth k,* $\chi_g(G)\leq 3k+1$

1.6 Chapters

We begin in Chapter 2 by introducing an adversarial game based on the dominating set and the independent set games. This adversarial game is called the *perfect code game*. We explore some motivation from coding theory before proving several results related to path and cycle graphs. We then conclude with a theorem about pathwidth and a theorem about bounded degree.

In Chapter 3, we explore a new area, the intersection of online and adversarial algorithms. We define a new algorithmic parameterisation called *adversarial online*. As far as this author is aware, there has been no previous study in this area. We begin with some notes about online colouring along with definitions for adversarial online colouring. We explore some alternative ways of presenting the graphs. Next we prove several results related to bounded degree. To conclude we look at a theorem about adversarial online colouring trees.

In Chapter 4, we briefly explore the *chain decomposition game*, where we attempt to construct chain decompositions of partial orders. We make note on the relation between the decomposition game and the game colouring number of incompatibility graphs. We also make some comments on online chain decomposition. We finish this chapter with some notes on adversarial online colouring.

Finally, in Chapter 5, we explore a new notion related to online algorithms, and *strongly online graphs*. We begin with some background from highly computable graph theory. We provide a definition of strongly online graphs. Then, we prove a result about strongly online trees. We then prove a lower bound for general graphs related to their chromatic number. We explore several results about strongly online colouring graphs with bounded pathwidth. To finish, we introduce a natural notion of a strongly online path decomposition and prove several related theorems.

Chapter 2

The Perfect Code Game

In this chapter, we will begin our analysis of adversarial games and algorithms on graphs. We do so with a game we call *the perfect code game*. In this game players take turns constructing a perfect code. All results in this chapter are new and are due to this author.

We begin by providing definitions for perfect codes and the perfect code game, along with some motivation from coding theory. We will then prove upper bounds for both cycle and path graphs. Next we show an upper bound for graphs of bounded pathwidth. Finally, we explore the perfect code game in graphs of bounded degree.

2.1 Perfect Codes

Suppose it is Alice's job to construct a radio network in a rural area. She places radio towers at people's houses. Each tower provides radio coverage to itself and the neighbouring homes. The problem is, if two towers have overlapping coverage, then the interference between the towers interferes with the whole network. Thus, no two towers can provide coverage to the same house. Is it possible to place towers in such a way that every house receives coverage? Consider a graph such that the houses are vertices and two houses share an edge if a tower placed at one provides coverage to the other. In such a graph, a set of houses that provide cover to the entire area is an example of a *perfect code*.

Definition 2.1 (Perfect Code)**.** For a graph $G = (V, E)$, a *perfect code* in G is an independent subset C of V such that every vertex in V is either in C or adjacent to exactly one vertex in C.

Further motivation comes from areas such as coding theory (see Section 2.1.1) and resource allocation in computer systems (for example, see Livingston and Stout 1988 [23]).

There is no standard notation for perfect codes. The name perfect code comes from the study of error correcting codes in coding and information theory. The the terminology for and the idea of perfect codes was introduced by Biggs 1973 [24]. Perfect codes also arise from work in graph theory where they are known as *efficient dominating sets* [25], *independent perfect dominating sets* [26], or *perfect dominating sets* [27].

2.1.1 Coding and Information Theory

Perfect codes have other uses beyond graph theory. One such area is coding theory. As further motivation we will briefly explore these alternative uses/origin.

Suppose we are trying to send a message to a friend. One way to do this is to associate each symbol in our message to a binary string and transmit this string. Over a robust network the chance the message is corrupted along the way is low. However, if we are sending a message a long way or over an unreliable network our message could end up scrambled. For example, suppose you are trying to send a message to a spacecraft on the edge of our solar system. We need some way way of encoding our message that is capable of detecting and possibly correcting errors. This is the realm of coding theory.

A *code* C is a subset of $\mathbb{F}_r^n$ where $\mathbb{F}_r$ is a finite field with r elements. The strings in C are called *codewords*. For any two codewords x and y the *Hamming distance* between x and y is the number of places in which they differ. For example, $(1, 1, 1, 1)$ and $(1, 1, 2, 2)$ have Hamming distance 2.

A code C is e-error detecting if, whenever a codeword incurs at least 1 and at most e errors, the resulting string is not a codeword. A code C is e-error correcting if, when given a string x with at most e errors, the codeword with minimum Hamming distance to x is the intended codeword.

Definition 2.2 (Perfect Code in Vector Space)**.** A code C is called *perfect* if for every code $u \in \mathbb{F}_r^n$ there is exactly one codeword with Hamming distance 1 from u.

A perfect code differs from a 1-error correcting code in that it requires every string to be within Hamming distance 1 from a codeword.

Definitions 2.1 and 2.2 are equivalent via *Hamming graphs*. The Hamming graph $H(n, r)$ is a graph with vertices indexed by elements of $\mathbb{F}_r^n$ and an edge between two vertices if the corresponding strings have Hamming distance 1. In order to find a perfect code in $\mathbb{F}_r^n$ we can find a perfect code in the graph $H(n, r)$, and vice versa. For example see Figure 2.1.

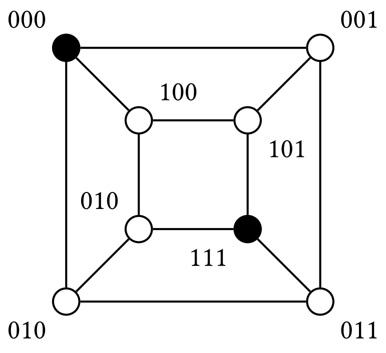


Figure 2.1: The Hamming graph $H(3, 2)$ with a perfect code

We can now see that, as demonstrated in Biggs 1973 [24], graphs are a natural setting for perfect codes. Henceforth, we will only focus on perfect codes in graphs. For more information about codes and coding theory we direct the reader to Hamming 1986 [28].

2.2 Perfect Code Game

To inspire the idea of the *perfect code game* suppose the local council assigns Alice a partner, Bob. Alice and Bob take turns to place towers. But Bob has his own podcast and hates radio towers. So he decides to make it his personal mission to disrupt the network. He must still obey the rules, that is, he must place towers such that no house is covered twice. However, Bob will attempt to place towers in such a way that the resulting network is not a perfect code. Is it possible for Alice to place towers in such a way that forces coverage of the entire area? If not, then how much of the area can Alice guarantee will be covered?

Definition 2.3 (Perfect Code Game)**.** Let $G = (V, E)$ be a graph, $C \subseteq V$ a (possibly partial) perfect code that we initialise to the empty set, and t a target score. In the *perfect code game* Alice and Bob take turns adding a vertex v to C such that $C \cup v$ is independent and dominates at least one new vertex exactly once. The game continues until either C forms a perfect code or cannot be enlarged further. At the end of the game, let s be the number of vertices not dominated by C. Alice wins if $s \leq t$, and Bob wins if $t < s$.

Note that if we have a target score $t = 0$ then Alice wins if and only if C forms a perfect code.

Definition 2.4 (Perfect Code Game Number)**.** For a graph G, the *perfect code game number* $\mathrm{PCode}_g(G)$ is the minimum target score in the perfect code game such that Alice has a winning strategy[1] when Alice has the first move. Similarly, $\mathrm{PCode}'_g(G)$ is the *perfect code game number* on G when Bob has the first move.

2.3 Path and Cycle Graphs

The path P_n is the graph with n vertices, $v_1, \ldots, v_n$, where for all $i < n$, there is an edge joining v_i and v_{i+1}. A cycle is a path with the extra edge (v_1, v_n), turing the path into a circle.

Path and cycle graphs are very simple graphs and have simple upper bounds. However, before we bound the perfect code game number for paths and cycles, we introduce Lemma 2.5.

We say that a vertex v is *blocked* if it is undominated and cannot be played when playing the perfect code game. That is, $v \notin C$ and $N(v) \subseteq N(C)$, where C is a (possibly partial) perfect code. If Alice or Bob can play a vertex v that results in a vertex u being blocked, then we say v blocks u.

Lemma 2.5. *Let P_{n+2} be a path graph with $n + 2$ vertices such that $n \geq 5$. If $C \subset V(P_{n+2})$ is a perfect code containing exactly the leftmost and rightmost vertices of P_{n+2} (i.e. $C = \{u_1, u_2\}$ in Figure 2.2), then*

$$\mathrm{PCode}_g\left(P_{n+2}\right) \leq {}^{n}/_{4} + {}^{1}/_{4}$$
$$\mathrm{PCode}'_g\left(P_{n+2}\right) \leq {}^{n}/_{4} + {}^{7}/_{4}$$

[1]A winning strategy for Alice is a strategy that guarantees that the game will end with Alice winning.

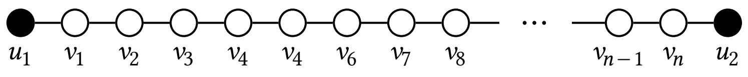
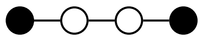
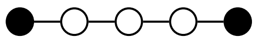
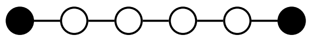
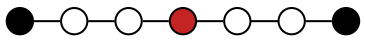
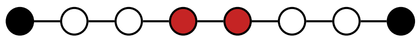
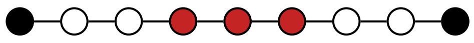


Figure 2.2: The Path Graph P_{n+2}

Proof. We will prove this by induction on n for both $\mathrm{PCode_g}$ and $\mathrm{PCode_g'}$ simultaneously on the partially dominated P_{n+2}. It is not possible for $n = 1$, as v_1 will be dominated twice, so we will demonstrate the cases $n = 2, 3, \ldots, 7$. The red vertices represent the possible moves. It is clear to see that for all $n \leq 7$ each possible move blocks the same number of vertices.

n	P_{n+2}	$\mathrm{PCode_g}(P_{n+2})$	$\mathrm{PCode_g}'(P_{n+2})$
2		0	0
3		1	1
4		2	2
5		0	0
6		1	1
7		2	2

Thus, for $n = 5, 6, 7$, $\mathrm{PCode_g}\left(P_{n+2}\right) \leq {}^{n}\!/_{4} + {}^{1}\!/_{4}$ and $\mathrm{PCode_g'}\left(P_{n+2}\right) \leq {}^{n}\!/_{4} + {}^{7}\!/_{4}$, as desired.

Note that for $n = 4$, $\mathrm{PCode_g}\left(P_{4+2}\right) = 2 \not\leq {}^{4}\!/_{4} + {}^{1}\!/_{4} = 1.25$. But, as we will see later this will not matter.

Now assume that the inductive hypothesis holds for all $k < n$. First, we will show that in the Alice start game $\mathrm{PCode_g}\left(P_{n+2}\right) \leq {}^{n}\!/_{4} + {}^{1}\!/_{4}$.

Alice's strategy is to play the vertex v_6. This splits the graph into two games. One game has $k = 5$ unplayed vertices and will always end with $\mathrm{PCode_g}\left(P_{5+2}\right) = 0$. The other has $k = n - 6$ unplayed vertices. As $\mathrm{PCode_g}\left(P_{k+2}\right) \leq \mathrm{PCode_g'}\left(P_{k+2}\right)$ we assume that Bob has the first move in the $k = n - 6$ game. Assuming $k \neq 4$, by the inductive hypothesis Alice has a winning strategy on $P_{(n-6)+2}$ with $\mathrm{PCode_g'}\left(P_{(n-6)+2}\right) \leq {}^{(n-6)}\!/_{4} + {}^{7}\!/_{4}$. Thus we have

$$\begin{aligned}\mathrm{PCode_g}\left(P_{n+2}\right) &\leq \mathrm{PCode_g}\left(P_{5+2}\right) + \mathrm{PCode_g'}\left(P_{(n-6)+2}\right)\\ &\leq 0 + {}^{(n-6)}\!/_{4} + {}^{7}\!/_{4}\\ &= {}^{n}\!/_{4} + {}^{1}\!/_{4}\end{aligned}$$

If $k = 4$, then we have $n = 10$ and

$$\begin{aligned}\mathrm{PCode_g}\left(P_{10+2}\right) &\leq \mathrm{PCode_g}\left(P_{5+2}\right) + \mathrm{PCode_g'}\left(P_{(4)+2}\right)\\ &= 0 + 2\\ &< {}^{10}\!/_{4} + {}^{1}\!/_{4} = 2.75\end{aligned}$$

Therefore for all $n \geq 5$

$$\mathrm{PCode_g}\left(P_{n+2}\right) \leq {}^{n}\!/_{4} + {}^{1}\!/_{4}$$

Next, we show that for the Bob start game $\text{PCode}'_g\left(P_{n+2}\right) \leq {}^{n}/_{4} + {}^{7}/_{4}$.

Suppose Bob plays vertex v_i.

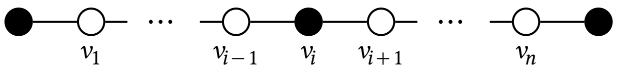


Figure 2.3: Bob splitting the graph into two games

This splits the graph into two games (see Figure 2.3). One game contains $k = i-1$ unplayed vertices and the other $k = n-i$ unplayed vertices. Alice plays her next move according to her winning strategy on $P_{(i-1)+2}$. Thus we have an Alice start game on $P_{(i-1)+2}$ and a Bob start game on $P_{(n-i)+2}$. Then assuming $i-1 \neq 4 \neq n-i$, by the induction hypothesis we have

$$\begin{aligned}
\text{PCode}_g\left(P_{n+2}\right) &= \text{PCode}_g\left(P_{(i-1)+2}\right) + \text{PCode}'_g\left(P_{(n-i)+2}\right) \\
&\leq \left({}^{(i-1)}/_{4} + {}^{1}/_{4}\right) + \left({}^{(n-i)}/_{4} + {}^{7}/_{4}\right) \\
&= {}^{(i-1+n-i)}/_{4} + 2 \\
&= {}^{(n-1)}/_{4} + 2 \\
&= {}^{n}/_{4} + {}^{7}/_{4}
\end{aligned}$$

If one of $i-1$ and $n-i$ equals 4, then the inductive hypothesis doesn't hold. Suppose that only one of them equals 4, without loss of generality assume it is $i-1=4$. In this case, Alice plays her strategy on $v_5, \ldots, v_n$, (i.e. $P_{(n-5)+2}$) and we have:

$$\begin{aligned}
\text{PCode}_g\left(P_{n+2}\right) &= \text{PCode}_g\left(P_{4+2}\right) + \text{PCode}_g\left(P_{(n-5)+2}\right) \\
&\leq 2 + \left({}^{(n-5)}/_{4} + {}^{1}/_{4}\right) \\
&= 2 + {}^{n}/_{4} - {}^{5}/_{4} + {}^{1}/_{4} \\
&= {}^{n}/_{4} + 1 \\
&\leq {}^{n}/_{4} + {}^{7}/_{4}
\end{aligned}$$

If both $i-1$ and $n-i$ equal 4, then $i=5$ and $n=9$. Thus

$$\begin{aligned}
\text{PCode}'_g\left(P_{(9)+2}\right) &\leq \text{PCode}_g\left(P_{(5-1)+2}\right) + \text{PCode}_g\left(P_{(9-5)+2}\right) \\
&= 2 + 2 \\
&= 4 \\
&= {}^{9}/_{4} + {}^{7}/_{4} \\
&= {}^{n}/_{4} + {}^{7}/_{4}
\end{aligned}$$

As desired.

Therefore for all $n \geq 5$

$$\text{PCode}'_g\left(P_{n+2}\right) \leq {}^{n}/_{4} + {}^{7}/_{4} \qquad \square$$

Both cycle and path graphs have an upper bound of $O({}^{n}/_{4})$, differing only in the constants. This comes about because once the first move is made, a cycle can be considered as a path. This is done by ignoring the blocked vertices from the first move.

Theorem 2.6. *For C_n, a cycle graph with $n \geq 6$ vertices,*

$$\begin{aligned}\mathrm{PCode}_g(C_n) &\leq {}^{n}\!/\!_{4} + 1.5\\ \mathrm{PCode}'_g(C_n) &\leq {}^{n}\!/\!_{4}\end{aligned}$$

Proof. Regardless of who plays first, after the first move the partially dominated graph is equivalent to a path graph with $n+1$ vertices where the end vertices have been played and there are $n-1$ unplayed vertices. See Figure 2.4.

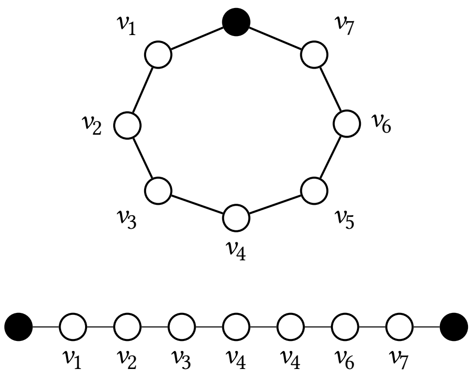


Figure 2.4: The cycle graph C_8 and its equivalent path graph, $P_{(7)+2}$

If Alice played first, then Bob gets the first move on the remaining $n-1 \geq 5$ vertices. Thus, by Lemma 2.5

$$\begin{aligned}\mathrm{PCode}_g(C_n) &\leq \mathrm{PCode}'_g\left(P_{(n-1)+2}\right)\\ &\leq {}^{(n-1)}\!/\!_{4} + {}^{7}\!/\!_{4}\\ &= {}^{n}\!/\!_{4} + 1.5\end{aligned}$$

If Bob played first, then Alice gets the first move on the remaining $n-1 \geq 5$ vertices. Thus, by Lemma 2.5

$$\begin{aligned}\mathrm{PCode}'_g(C_n) &\leq \mathrm{PCode}_g\left(P_{(n-1)+2}\right)\\ &\leq {}^{(n-1)}\!/\!_{4} + {}^{1}\!/\!_{4}\\ &= {}^{n}\!/\!_{4}\end{aligned}$$

□

Theorem 2.7. *For P_n, the path graph with $n \geq 5$ vertices*

$$\mathrm{PCode}_g(G) \leq {}^{n}\!/\!_{4} + {}^{7}\!/\!_{4}$$

Proof. We demonstrate a strategy for Alice that ensures that at the end of the game there at most ${}^{n}\!/\!_{4} + {}^{7}\!/\!_{4}$ many blocked vertices.

Consider the path graph labelled as in Figure 2.5. Alice's first move is to play the vertex v_4. The only vertex v_j with $j < 4$ that is undominated is v_1. Playing v_1 doesn't block any vertices. Thus v_1 will be played at some point, but Alice attempts to avoid playing v_1.

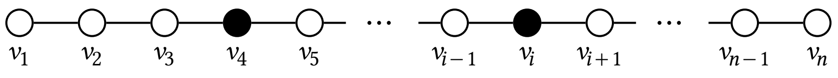


Figure 2.5: The path graph P_n

Suppose that on Bob's turn he plays the vertex v_i. There are two cases, $i < n-5$ and $i \geq n-5$.

First, assume that $i < n-5$. In this case, Alice plays v_{n-3}. The only vertex v_j for $j > n-3$ that is undominated is v_n and playing v_n doesn't block any vertices. Alice attempts to avoid playing v_n. The remaining unplayed vertices form two partially dominated Bob start games, one on $P_{(i-5)+2}$ and the other on $P_{(n-4-i)+2}$. By Lemma 2.5, Alice has winning strategies on these games. Hence

$$\begin{aligned}\mathrm{PCode}_g(P_n) &\leq \mathrm{PCode}'_g\left(P_{(i-5)+2}\right) + \mathrm{PCode}'_g\left(P_{(n-4-i)+2}\right)\\ &\leq {}^{(i-5)}/_4 + {}^{7}/_4 + {}^{(n-4-i)}/_4 + {}^{7}/_4\\ &= {}^{n}/_4 + {}^{5}/_4\end{aligned}$$

Now assume $i \geq n-5$. This means that Bob has played a vertex v_{n-j} such that $j \in [0,5]$. This turns the game into an Alice start partially dominated game on $P_{(-4+n-j)+2}$. Notice how v_{n-2} blocks the vertex v_n. By Lemma 2.5, Alice has a winning strategy on this game. Hence

$$\begin{aligned}\mathrm{PCode}_g(P_n) &\leq \mathrm{PCode}_g\left(P_{(-4+n-j)+2}\right) + 1\\ &= {}^{(-4+n-j)}/_4 + {}^{1}/_4 + 1\\ &= {}^{n}/_4 - {}^{(j+1)}/_4\\ &\leq {}^{n}/_5 + {}^{5}/_4 && \text{for all } j \in [0,5]\end{aligned}$$

Therefore $\mathrm{PCode}_g(P_n) \leq {}^{n}/_5 + {}^{5}/_4$. □

2.4 Graphs of Bounded Pathwidth

In general, finding a perfect code in a graph is NP-complete [29]. However, for graphs of bounded pathwidth there is a polynomial-time algorithm for finding perfect codes [30]. In light of this we would hope that for graphs of bounded pathwidth we could bound their perfect code game number. However, this is not the case.

Theorem 2.8. *For all $n, k > 0$ there exists a graph $G = (V, E)$ such that G has pathwidth k and $\mathrm{PCode}_g(G) \geq n$.*

Proof. Let G be the graph consisting of the complete graph K_k and $n+1$ vertices all connected to every vertex in K_k (See Figure 2.6). Consider two disjoint copies of G. Bob's first move will be to play the vertex v_{n+1} in which ever copy of G Alice did not play in. Vertex v_{n+1} dominates K_k and blocks all n vertices $v_1, \ldots, v_n$. Therefore the game ends with $\mathrm{PCode}_g(G) \geq n$. □

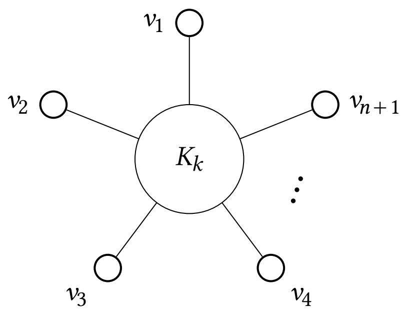


Figure 2.6: The graph G

2.5 Graphs of Bounded Degree

For graphs of bounded degree we also have unbounded perfect code game number.

Theorem 2.9. *For all $\Delta, n > 3$ there exists a graph G with maximum degree Δ and $\mathrm{PCode_g}(G) \geq n$.*

Proof. Consider the path graph P_{2n} with $2n$ vertices. For every vertex v in P_{2n} add $\Delta - 2$ vertices adjacent to v. See Figure 2.7.

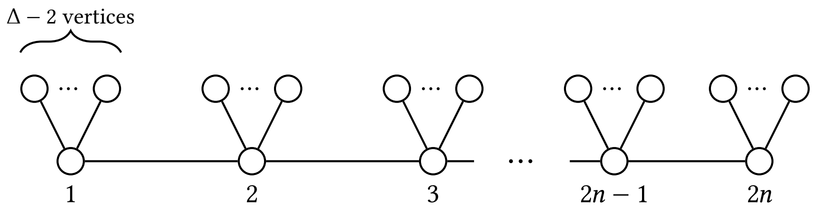


Figure 2.7: The extended path graph P_{2n}

Bob's strategy is to play any leaf that blocks $\Delta - 3$ vertices. After n rounds there will be at least $n(\Delta - 3) > n$ blocked vertices. ☐

Chapter 3

Adversarial Online Colouring

In this chapter we will continue our analysis of online and adversarial games. We explore online and adversarial games in the context of graph colouring. We do this by creating a new game called the *adversarial online colouring game.* All the results in this chapter related to adversarial online colouring are new and due to this author.

The *growth rate* of an colouring algorithm **A** is a function (normally expressed in big O notation) that bounds the number of colours needed by **A**, relative to the number of vertices. For example, Let $\mathscr{C}$ a class of graphs. If **A** requires $n/3 + 12$ colours to colour to colour any graph in $\mathscr{C}$ with n vertices, then we say the growth rate of **A** is $O(n)$.

We will begin with some discussion and a definition of adversarial online colouring. We will then prove several results related to graph degree. We conclude this chapter with a proof that the adversarial online colouring game on trees is determined only by the number of colours and the number of vertices (assuming that both players have a chance at winning).

3.1 Adversarial Online Colouring

Online colouring can be considered a game between two players, Alice and Bob. Bob presents the vertices to Alice one at a time. Upon being presented with a vertex Alice, irrevocably assigns it a colour. Adversarial online adds an additional hostile element to this game. Bob is also allowed to colour every second vertex he presents. Bob must still obey the rules of colouring. That is, he must not assign a colour c to a vertex v if any of v's neighbours have already been coloured with c. This is *Adversarial Online Colouring*.

The *performance ratio* of an online colouring algorithm is the ratio of the number of colours used in the online algorithm vs the number of colours used in an optimal offline algorithm. For example, if **A** online colours a graph G is n colours but G has an optimal m-colouring the performance ratio of **A** is n/m. When online colouring arbitrary graphs, the performance ratio is a function of the graph size. For example, online colouring a tree with n vertices has performance ratio $O(\log n)$ many colours [31].

As with many problems, when we see that the general problem is very hard, we can parameterize the problem which sometimes make it more tractable Downey and Fellows 2013 [16]. In the remainder of this chapter we will look at the situation for a number of such parameterizations. We look at bounded degree, and another situation where wew restrict the structure of the graph. One such parameterization is d-degenerate graphs[1].

Definition 3.1. A graph $G = (V, E)$ is *d-degenerate* if there is an ordering of V such that for all $v \in V$, $\left|\{u : u < v, u \in N(v)\}\right| \le d$.

For all k there exists a d-degenerate graph G such that G requires k colours to online colour. But, the growth rate for algorithms on d-degenerate graphs often less than $O(n)$, where n is the number of vertices.

Theorem 3.2 (Irani 1994 [33])**.** *For G a d-degenerate graph with n vertices, First Fit uses $O(d \log n)$ many colours to online colour G. Further, this bound is tight.*

Any graph with n vertices can be online coloured using n colours. Thus $O(n)$ is the worst possible bound. $O(n)$ is the bound for many classes of graphs in adversarial online colouring.

Consider the graph in Figure 3.1. It has pathwidth 2, treewidth 1, and is 1-degenerate, but requires 9 colours to be adversarial online coloured.

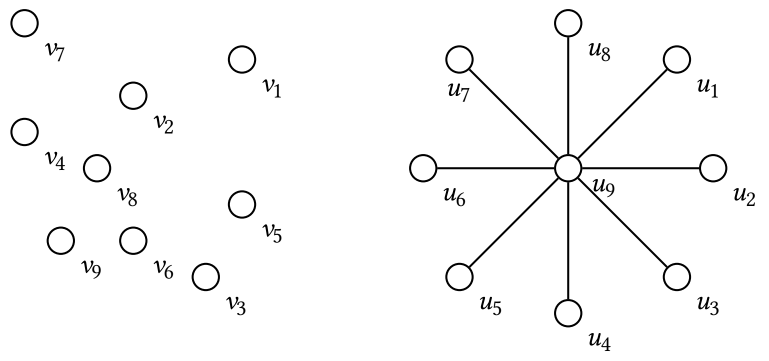


Figure 3.1: The star graph S_9 along with 9 isolated vertices

The strategy for Bob is to present all isolated vertices to Alice, while he colours the vertices $u_1, \dots, u_8$ the colours 1 through 8. He then presents the vertex u_9, which must have colour 9. By using larger star graphs and more isolated vertices we can force more and more colours. These obstructions lead us to Proposition 3.3.

Proposition 3.3. *For all k there exists a graph with pathwidth (treewidth, or degeneracy) k and n vertices that requires at least $O(n)$ many colours to adversarial online colour.*

3.2 Alternative Presentations

Most classical (offline) algorithms about graphs with bounded pathwidth, treewidth, etc., rely on first decomposing the graph into some parsed structure. These algorithms will then typically do some dynamic programming on the parsed structure. This approach

[1] d-degenerate graphs are an wider class of graphs that includes bounded pathwidth and treewidth, and planar graphs. Interestingly many results from other graph classes generalise to d-degenerate, such as dominating set generalising from planar graphs [32].

culminates in Courcelle's Theorem for graphs of bounded treewidth [34]. An example of such a parsed structure is a tree decomposition.

As an example, we consider the online graph $G^{\prec}$ with pathwidth k presented as an implicit path decomposition. That is, if $G^{\prec}$ is an online graph with a path decomposition $\mathscr{P} = \{P_1, P_2, \dots\}$, then the presentation order $(<)$ of G presents every vertex in P_1, then $P_2 \setminus P_1$, then $P_3 \setminus (P_2 \cup P_1)$ and so on. This has the effect of presenting G as a series of bags. G can then be adversarial online coloured in $k+1$ colours. This is because no vertex that is presented will have more than k many already presented neighbours. This is the same bound as the online version. The reason for this is that by presenting the graph in as an implicit decomposition manner we bound the number of neighbours we have seen when a vertex is presented. It is bounded degree by stealth.

Always-connected adversarial online is a variation of adversarial online where at each stage the given graph must be connected. Even this restriction is $O(n)$. This can be seen by changing the star graph in Figure 3.1 for a cycle graph with a vertex in the centre and turning the isolated vertices in to a clique with one vertex connected to the cycle. Such a graph can be presented as always connected but still require $O(n)$ colours.

3.3 Maximum Degree

We will now look at a series of structural parameterizations. The first parameterization we will look at is maximum degree. By bounding the degree of a graph we get a lower bound on the number of colours needed. Theorem 3.4 essentially says that Bob will win if there are less than half as many colours as the max degree.

Theorem 3.4. *Let G be a graph with maximum degree Δ. Then at least $\lceil {}^{(\Delta-1)}/_{2} \rceil + 1$ many colours are required to adversarial online colour G. Thus adversarial online colouring is* $O(\Delta)$.

Proof. Let G be a graph with maximum degree Δ. It suffices to show there is a winning strategy for Bob with $\lceil {}^{(\Delta-1)}/_{2} \rceil$ colours. Let v be a vertex with $\delta(v) = \Delta$ and $u_1, \dots, u_\Delta$ be the neighbours of v. Bobs' strategy is as follows.

- Present u_1 to Alice. Which, without loss of generality will get colour 1.
- Colour u_2 with colour 2.
- Present any uncoloured u_i to Alice and colour any uncoloured u_j with an unused colour.

When this is finished v will have

$$2 + \left\lfloor \frac{\Delta - 2}{2} \right\rfloor \geq \left\lceil \frac{(\Delta - 1)}{2} \right\rceil$$

uniquely coloured neighbours. Therefore v cannot be coloured and Bob wins. ☐

In Theorem 3.4 our strategy involved Alice colouring half and Bob colouring the other half of the neighbours of a vertex. Another strategy is to make Alice colour some vertices that are not in the neighbours of some vertex. This allows Bob to force more needed colours, relative to the maximum degree. If Bob wants to colour all the neighbours of a vertex without interference from Alice then he needs one more than 3 times the degree of that vertex. This is witnessed by Theorem 3.5.

Theorem 3.5. *Let $G = (V, E)$ be a graph. If there exists a vertex $v \in V$ such that $|V| \geq 3\delta(v) + 1$ and for all $u \in N(v)$, $\delta(u) \leq \delta(v)$ then G requires at least $\delta(v)$ many vertices to adversarial online colour.*

Proof. Let v be a vertex such that $|V| \geq 3\delta(v) + 1$ and for all $u \in N(v)$, $\delta(u) \leq \delta(v)$. Let $\delta(v) = k$.

Denote $N(v) = \{v_1, v_2, \ldots, v_k\}$.

In every round, Bob presents a vertex in $V \setminus N[v]$ to Alice, and colours the least i such that v_i is uncoloured.

Suppose we want to colour v_i. Thus $v_1, \ldots, v_{i-1}$ have all been coloured. $\delta(v_i) \leq k$ and there are at least

$$|V| - |N[v]| = 3k - (k + 1) = 2k - 1$$

vertices in $G \backslash N[v]$.

The vertex v_i is connected to $\delta(v_i) - 1 \leq k - 1$ vertices in $G \backslash N[v]$, and $i - 1$ vertices in $G \backslash N[v]$ have already been coloured. Thus there must be at least

$$(2k - 1) - (k - 1) - (i - 1) = k - i + 1 \geq 1$$

uncoloured vertices in $G \backslash N[v]$ that are not adjacent to v_i. Let u be such a vertex. See Figure 3.2

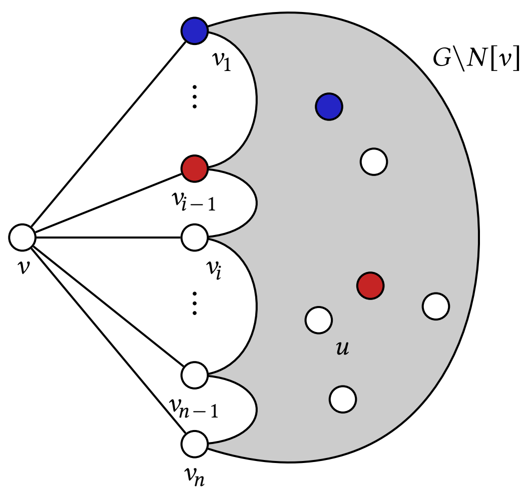


Figure 3.2: The partially coloured graph, G

Present u to Alice. If Alice colours u a colour from the set $\{1, \dots, i-1\}$, then Bob colours v_i with the colour i. Otherwise Alice has coloured u a colour not seen before. Without loss of generality let it be i. Bob colours v_i with the colour i.

Thus each vertex v_i has colour i. Hence when all the vertices in $N(v)$ have been coloured, v has k many uniquely coloured neighbours. Thus v requires a $k+1$-th colour. Therefore G requires $k+1$ colours to adversarial online colour. □

We also get Corollary 3.6 because the second condition in Theorem 3.5 is vacuously true when $|V| \geq 3\Delta + 1$.

Corollary 3.6. *Let $G = (V, E)$ be a graph with maximum degree Δ and $|V| \geq 3\Delta + 1$. Then G requires $\Delta + 1$ many colours to adversarial online colour.*

3.4 Bipartite Graphs

A graph $G = (V, E)$ is called *bipartite* if V can be partitioned into two sets A and B such that there are no edges joining two vertices in in the same set. The complete bipartite graph $K_{n,m}$ is a bipartite graph with vertex sets A and B where $|A| = n, |B| = m$, and every vertex in A is connected to every vertex in B.

Definition 3.7 (Matching)**.** Let $G = (V, E)$ be a graph. A *matching* $M \subseteq E$ is a set of independent edges, that is a set of edges that share no common vertices. We write $u = M(v)$ (and $v = M(u)$) to denote that $(u, v) \in M$. A matching is *perfect* if every vertex in V is incident to some edge in the matching.

Theorem 3.8. *Let $G = (A, B, E)$ be the complete bipartite graph $K_{n,n}$ with all the edges from a perfect matching from A to B removed. Then G requires n colours to adversarial online colour.*

Proof. Let A, B be a partition of $K_{n,n}$ with all the edges from a perfect matching, M, from A to B removed Bob's strategy is as follows.

- Present some uncoloured $v \in A$ to Alice.
- Bob colours $M(v)$ what ever colour v has.

By doing this, after every round every vertex is incident to another colour. Thus the last two vertices must be coloured the n-th colour. □

As every vertex in $K_{n,n}$ has degree n, Alice can always colour $K_{n,n}$ with n colours. This means Theorem 3.8 is the worst possible bound.

One theorem about bipartite graphs that we will need later is the following famous theorem by Hall.

Hall's Marriage Theorem (Hall 1935 [35])**.** *Let $G = (U, V, E)$ be a bipartite graph. There is a matching from U to V if and only if for every $A \subseteq U$*

$$|N(A)| \geq |A|$$

3.5 Trees

Trees have neither bounded degree, pathwidth, nor degeneracy. This might lead us to suspect that adversarial online colouring of trees could be bounded by one of these parameters, but this is not the case. A tree is adversarial online k-colourable if and only if it has at most $2k-2$ vertices (so long as k is not too small or large).

Theorem 3.9. *Fix a tree $T = (V, E)$ with maximum degree Δ and k such that $\Delta \geq k > \Delta/2 + 1$. Then T can be adversarial online coloured in k colours if and only if*

$$|V| \leq 2k - 2$$

We have $\Delta \geq k \geq \Delta/2$ because, Alice will trivially win if $k > \Delta$ and Bob will win if $k \leq \Delta/2 + 1$ by Theorem 3.4. So to avoid trivialities we restrict k.

Proof. We show the forward direction by contrapositive. So, assume $|V| > 2k-2$. It suffices to show there is a winning strategy for Bob with k colours.

As k is at most the maximum degree of G there is some vertex $v \in V$ such that $\delta(v) \geq k$. Fix some such $v \in V$. Fix some $X \subseteq N(v)$ such that $|X| = k$. Note that

$$\begin{aligned} |V - X - v| &= n - k - 1 \\ &\geq (2k-1) - k - 1 \\ &= k - 2 \end{aligned}$$

Therefore we can take some $Y \subseteq V - X - v$ such that $|Y| = k - 2$. Every vertex in Y is connected to at most one vertex in X. Thus for every $A \subseteq Y$, the set A is not connected to at least $k - 1 > |Y| \geq |A|$ vertices in X. Therefore by Hall's theorem, there is an anti-matching[2] M from Y to X (ignoring all anti-edges in X and Y). Note that $|M| = k - 2$. See Figure 3.3.

Let u_1, u_2 be the two vertices in $X - \text{range}(M)$. Bob's strategy is as follows.

1. Present u_1 to Alice. Without loss of generality, u_1 will get colour 1.
2. Colour u_2 colour 2.
3. For each $(a, b) \in M$, present $a \in Y$ to Alice. Let $c(a) = i$. If i has not been used in $N(v)$, then Bob colours $b \in N(v)$ with i. If i has already been used in $N(v)$, then Bob colours b with any colour not used in $N(v)$.

Step 3 ensures that for every $b \in \text{range}(M)$, Bob colours b a colour not seen in $N(v)$. Thus, when Bob has coloured all of range(M) v will have k uniquely coloured neighbours. Hence whenever v is revealed, it cannot be coloured, and Bob wins. Therefore we have a winning strategy for Bob.

We will now show the reverse direction by induction on n, the number of vertices in the tree. That is, we show that for any $\Delta \geq k > \Delta/2 + 1$, if $n \leq 2k - 2$, T can be adversarial online coloured in k colours.

[2]An *anti-matching* in a graph G is a matching in the complement graph of G.

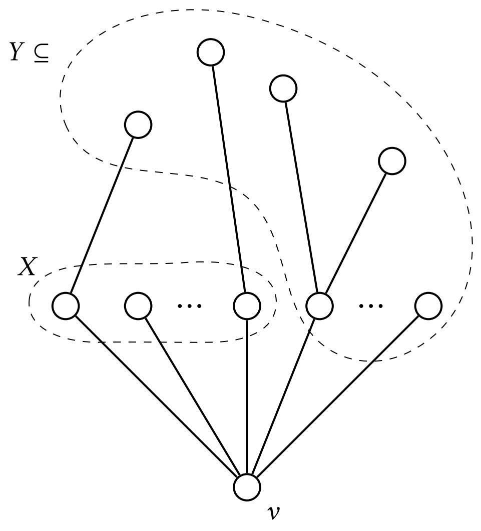


Figure 3.3: The structure of the anti-matching in T

For the base case, assume that $n = \Delta + 1$. Then T is a star graph and can be adversarial online coloured in $\Delta/2 + 1 < k$ colours.

For the inductive step, assume that any tree with $n \leq 2k - 2$ vetices can be adversarial online coloured in k colours. Let $T = (V, E)$ be a tree such that $|V| = n + 1 \leq 2k - 2$.

Let $v \in V$ be a vertex of degree at least k. We will show that when v is revealed it has at most $k - 1$ uniquely coloured neighbours. Without loss of generality, we can assume that v is the last vertex coloured.

Let u be the vertex coloured immediately before v. If $u \notin N(v)$, then $T \backslash u$ has n vertices. Therefore by the inductive hypothesis, $T \backslash u$ can be adversarial online coloured in k colours. That is v has at most $k-1$ uniquely coloured neighbours in $T \backslash u$. Hence when u is coloured, v has at most $k - 1$ uniquely coloured neighbours in T. See Figure 3.4.

Note that when u is revealed it has at most $\delta(u) - 1$ revealed neighbours. Further,

$$\begin{aligned}\delta(u) - 1 &\leq \left|V - N[v]\right| \\ &\leq (n+1) - k - 1 \\ &\leq (2k-2) - k - 1 \\ &= k - 3\end{aligned}$$

Suppose that $u \in N(v)$. Either Alice coloured u or Bob coloured u. First, assume Alice coloured u. By the inductive hypothesis, $T \backslash u$ can be adversarial online coloured in k colours. That is, v has at most $k - 1$ uniquely coloured neighbours. As u is adjacent to at most $k - 3$ coloured vertices there is at least one colour that u is not adjacent to. Alice colours u the least colour that has already been used. This will not increase the number of uniquely coloured neighbours of v. Therefore v will have at most $k - 1$ uniquely coloured neighbours.

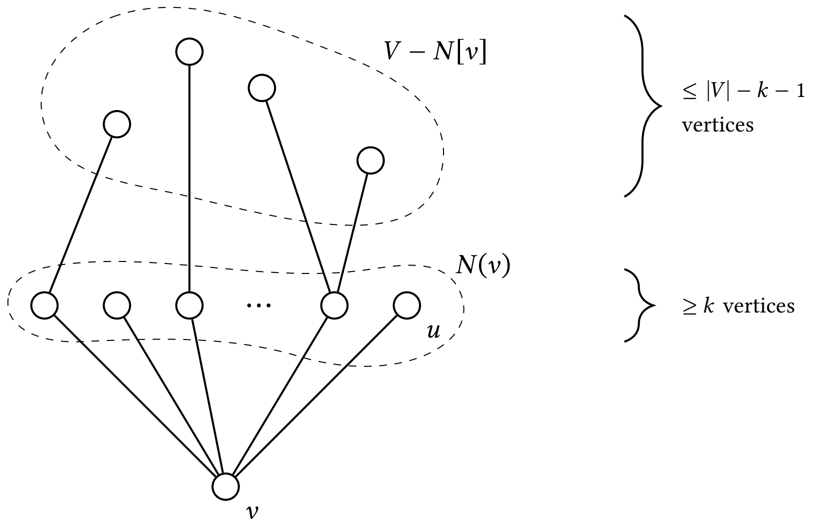


Figure 3.4: The structure of T

Now suppose Bob coloured u. As Bob plays on even turns there must have been an odd number of vertices in T. As $2k - 2$ is always even, $n + 1 \leq 2k - 3$. Therefore $T \backslash u$ has $n \leq 2k - 4 = 2(k - 1) - 2$ vertices. Hence $T \backslash u$ can be adversarial online coloured in $k - 1$ colours. That is, v has at most $k - 2$ uniquely coloured neighbours. Thus regardless of the colour of u, v will have at most $k - 1$ uniquely coloured neighbours.

Therefore every vertex in T has at most $k - 1$ uniquely coloured neighbours revealed before it, and thus T can be adversarial online coloured in k colours. $\square$

Chapter 4

The Chain Decomposition Game

In this chapter, we explore another adversarial game, but this time played on partial orders. We call this game the *chain decomposition game*. It is worth noting that this game could also accurately be called the *Dilworth game*. This name would be appropriate considering that the game is an extension of a famous theorem by Dilworth.

Partial orders can be found in all corner of mathematics. Partial orders have their uses in everything from algebra, to model theory, and set theory. The applications go beyond and through to computer science and operations research. One important structural aspect of partial orders is called a chain decomposition. Chain decompositions are a way of decomposing the partial order into linear subsets. A seminal theorem relating to chain decompositions is Dilworth's Theorem. One benefit of Dilworth's Theorem is that it allows for a theory of dimension for partial orders [36]. This theorem has been extensively studied with many proofs and application. For an overview of some of these see Bogart, Greene and Kung 1990 [37]. Even online (and effective) versions have been studied in great detail (some examples include [38, 39, 40, 41]). However, one aspect that has been overlooked is an adversarial version. We call this adversarial version the chain decomposition game.

We will begin this chapter with some definitions and a proof of Dilworth's Theorem. We will then introduce the chain decomposition game and demonstrate two results relating to upper and lower bounds for the game. To conclude we prove a result on the online adversarial version of the game.

4.1 Chain Decompositions

Before we can state Dilworth's theorem, we need some definitions.

Definition 4.1. A relation $\leq$ on a set of elements P is a *partial order* if it is,

1. Reflexive: For all $x \in P$ $x \leq x$.
2. Antisymmetric: If $x \leq y$ and $y \leq x$ then $x = y$.
3. Transitive: If $x \leq y$ and $y \leq z$ then, $a \leq z$.

Definition 4.2. A *chain* in a partial order P is a subset such that all the elements are comparable. An *anti-chain* in a partial order P is a subset such that all the elements are incomparable.

Definition 4.3. A *chain decomposition* of a partial order P is a set of chains $D_1, D_2, \dots, D_n$ such that all D_i are disjoint and $\bigcup_i D_i = P$.

Definition 4.4. The *width* of a partial order P is the size of the largest anti-chain in P and is denoted $\omega(P)$.

Dilworth's Theorem (Dilworth 1950 [42]). *A finite partially ordered set P can be decomposed into w chains, where w is the width of P.*

Over the years there have been many new proofs of Dilworth's Theorem. For example, in Dantzig and Hoffman 1956 [43], the authors prove Dilworth's Theorem using a linear programing approach. Fulkerson 1956 [44] provides a proof based off a theorem by König. We present a proof by Galvin 1994 [45] that provides a good clear overview.

Proof (Galvin 1994 [45]). We show this by induction on the size of P. The theorem is trivially true when P contains a single element. So, assume P is non-empty and let x be any maximal element in P. Suppose $P' = P \setminus \{x\}$ has width n. Then by the induction hypothesis, P can be covered in n chains, $C_1, \dots, C_n$.

For each $i \in [1, n]$ let x_i be the maximal element in C_i such that x_i is contained in an n-element antichain. We claim that $A = \{x_1, \dots, x_n\}$ is an antichain. Assume for a contradiction that A is not an antichain. Then there are $x_i, x_j \in P$ such that $x_i \le x_j$. There is some antichain A' of size n that contains x_j but not x_i. There must be some element $x_i' \in A' \cap C_i$. By the maximality of x_i, we get $x_i' \le x_i$. Thus $x_i' \le x_i \le x_j$, which contradicts the fact that A' is an antichain. Therefore A is an antichain.

If $A \cup \{x\}$ is an antichain, then $\{x\}, C_1, \dots, C_n$ is a chain cover of size $(n + 1)$ and we are done. Otherwise, assume $A \cup \{x\}$ is not an antichain. We show there is a chain cover of size n. There is some $i \in [1, n]$ such that $x > x_i$. Then $D = \{x\} \cup \{y \in C_i : y \le x_i\}$ is a chain. By our choice of x_i, the set $P \setminus D$ contains no n-element antichain. Therefore by induction, $P \setminus D$ is the union of $n - 1$ chains since $A \setminus \{x_i\}$ is an antichain of size $n - 1$ in $P \setminus D$. Hence P can be covered in n chains, as desired. $\square$

4.2 Adversarial Game

The *chain decomposition game* is played on a partial order P as follows. Fix n boxes. The value n serves as a parameter determining how many chains we are constructing. n is chosen before the game begins. Starting with Alice, the players take turns placing some element in P into a box such that all the elements in each box are comparable. The game ends when one of two win conditions are meet. First, Alice wins if all the elements have been placed into boxes. In this case, the boxes represent a chain decomposition. Second, Bob wins if on any turn there is an element that cannot be placed into any box.

Definition 4.5. The *game width* of a partial order P, denoted $\omega_g(P)$, is the smallest number of boxes needed such that Alice has a winning strategy on P.

Unfortunately infinitely many chains would be required for Alice to always win the chain decomposition game on every partial order. The following theorem was first proved by Krawczyk and Walczak 2015 [46] in the context of colouring incompatibility graphs (defined later).

Theorem 4.6 (Krawczyk and Walczak 2015 [46]). *For all k there exists a partial order P of width 2 such that $k \leq \omega_g(P)$.*

Chain decompositions and colouring certain types of graphs are intimately related. We have the following definition.

Definition 4.7 (Gilmore and Hoffman 1964 [47]). Let P be a partial order. The *incomparability graph* of P is a graph $G = (V, E)$ such that V is the set of elements in P and $(u, v) \in E$ if and only if $(u, v) \notin P$ and $(v, u) \notin P$.

That is, the incompatibility graph of a partial order P is a graph where two vertices are connected if the elements corresponding to these vertices are not related in P.

In a partial order for every pair of comparable elements x and y either $x \leq y$ or $x \geq y$. In a graph each vertex is only adjacent to it's neighbours. Hence, a partial order contains more information than it's incomparability graph. One example where this fact is used is in the proof of Theorem 1.14.

Fora partial order P, there is a one to one correspondence between chain decompositions of P and colourings of the incomparability graph of P. This means the game width and the game chromatic number are the same.

Theorem 4.8 (Folklore see [48]). *Let P be a partial order with width w and G the incompatibility graph of P. Then*

$$\omega_g(P) = \chi_g(G)$$

Proof. It suffices to show that there is a reduction from chain decomposition to colouring in the incomparability graph, and vice versa.

Fix P some partial order and G it's incomparability graph. Let $C_1, \dots, C_w$ be a chain decomposition of P. Every element in C_i is comparable, and so in G forms an independent set. Thus if each C_i is coloured with the colour i, we have a w-colouring of G.

Let $c : V \to [1, w]$ be a colouring of G. As every element in $c^{-1}(i)$ is independent, $c^{-1}(i)$ forms a chain in P. Thus each colour from c determines a chain in P. Therefore we have a chain decomposition of P into w chains. □

Theorem 4.8 effectively says that the chain decomposition game is exactly the same as the colouring game on incomparability graphs. For example, because interval orders[1] have incomparability graphs as interval graphs, we get Corollary 4.9.

Corollary 4.9. *Let P be a finite interval order with width w. Then*

$$\omega_g(P) \leq 3w - 2$$

Proof. By Faigle et al. 1993 [21], every interval order of interval width w has $\chi_g(G) \leq 2w-2$. Therefore $\omega_g(P) \leq 3w - 2$ by Theorem 4.8. □

[1]An interval order is a partial order whose incomparability graph is an interval graph.

4.3 Online Partial Orders

All proofs of Dilworth's Theorem seem to need global understanding of the structure of the partial order. As we mentioned in the introduction in *online* situations we will only have partial information and need to assign, in this case, a chain to each presented vertex of the online partial ordering. We recall the definition of an online partial order.

Definition 4.10. An *online partial order* $P^{\prec}$ is a partial order P and a linear order on the elements of P, $\prec$. Let $P_s^{\prec}$ be the partial order induced by the first s elements in $P^{\prec}$.

Definition 4.11. Let $P^{\prec}$ be an online partial order. An *online chain decomposition* D of $P^{\prec}$ is a sequence of chain decompositions $D_1, D_2, \ldots$ where each D_i is a chain decomposition of the first i elements in $P^{\prec}$, and D_{i+1} is an extension of D_i.

Theorem 4.8 might lead us to suspect that the online width of a partial order is the same as the online chromatic number of its incomparability graph. However, this is not the case.

The first upper bound on the online chain decomposition was from Kierstead 1981 [38]. They show there exists an online algorithm that partitions any online partial order of width w into at most $(5^w-1)/4$ chains. This is a rather large exponential bound. Subexponential online algorithms have been found. The first was found by Bosek and Krawczyk 2010 [40] ($w^{16\log w}$ many chains). Later this bound was improved by Bosek and Krawczyk 2015 [49] to $w^{13\log w}$ many chains. Then again by Bosek et al. 2018 [50] to $w^{6.5\log w+7}$. Currently the best upper bound is due to Bosek and Krawczyk 2021 [39]. They show $w^{O(\log\log w)}$ many chains are needed.

In the next section, we will look at a new variation of online algorithms for Dilworth's Theorem, where we imagine an online algorithm that is also adversarial.

4.4 Online Adversarial

With the online and adversarial versions in mind, we can take a look at the adversarial online version of chain decompositions. Unfortunately, there is not much to be said about adversarial online chain decompositions.

Theorem 4.12. *For all k there is an interval order of width* 2 *that cannot be online adversarial partitioned into fewer than k chains.*

Proof sketch. Let P be the ladder order with $2k$ vertices, see Figure 4.1. P has width 2. Present x_1 to Alice. Alice places x_1 in chain 1, We (Bob) places y_1 in chain 1.

For stage i Present Alice x_i. The element x_i is incomparable to y_{i-1} and thus cannot go into chain i. Thus x_i goes into chain i. Bob places y_i into chain i. Thus by induction there will be k many chains as a new chain is needed everytime Alice is presented an element. Each chain can be seen as a 'rung' in the ladder (the pair x_i and y_i). □

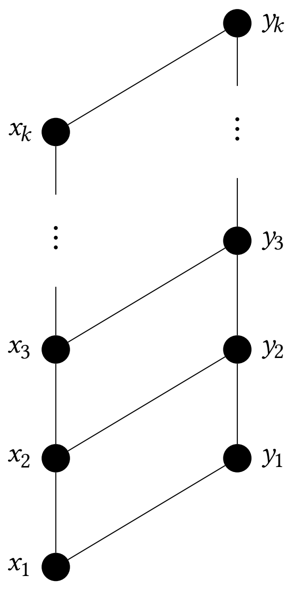


Figure 4.1: The Hasse diagram of the ladder Poset

Chapter 5

Strongly Online Graphs

In the preceding chapters, we have seen the effect of structural parameterizations (such as bounded pathwidth) of graphs on the online performance of online algorithms. In this chapter we introduce *strongly online* graphs and graph colouring as a new algorithmic parameterization to online graph colouring. This new idea is an analog of notions from computable graph theory [51]. All the results in this chapter that relate to strongly online graphs are new and due to this author unless otherwise stated.

Online algorithms have no information about future events[1]. At any stage we can only see an induced subgraph. It is this subgraph that we have already coloured. However, in many situations and problems we would expect more information about future vertices. For example, in online bin packing we have no knowledge of what items we will be placing into bags before they arrive. However, it is reasonable to expect that we would have some information about the upcoming items. We may be able to see the items before they arrive, or they may be arriving in a predictable pattern. This extra information we are provided with would also change your choice of algorithms. For example, if you are told that all the items will arrive in order of descending size then by using a greedy algorithm you can get a more optimal packing.

As with many situations we would expect to have more information about future events than simply an induced subgraph. This leads us to introduce the notation of *strongly online algorithms* and graphs. A strongly online graph is an online graph where at each stage we can see the neighbours of every vertex we could see at the previous stage along with the neighbours of the vertex we must colour at this stage.

The addition of this extra information allows us to get better bounds than normal online colouring of graphs.

In this chapter we will only consider countable locally finite graphs. That is, graphs $G = (V, E)$ where $|V| \leq \aleph_0$ and for any vertex $v \in V$, $\delta(v)$ is finite.

[1]It is possible for an online algorithm to have some information that looks like it is about future events. For example, online with lookahead (see Section 5.5.1) can be interpreted as seeing into the future. However lookahead is more accurately described as waiting. Another possibility is that we know that the graph has some property (e.g. the graph is a tree), this is only a promise not foresight.

Summary

We will begin this chapter by providing further motivation from and comparisons to highly computable graph theory. We will use this as a launching point to prove several bounds (upper and lower) on the online chromatic number. For example, we show that every strongly online graph can be coloured in twice its chromatic number. Next, we show how graph of bounded pathwidth have improved bounds. Proving every strongly online graph with pathwidth k, where k is even, can be online coloured with $2k+1$ colours. Then, after introducing a natural notion of strongly online pathwidth, we prove that there is a strongly online graph with no finite strongly online path decomposition.

5.1 Highly Computable Graphs

We will omit a formal definition of what it means to be *computable*, but direct the reader to Turing 1936 [1]. For our purposes, we consider a function $f\colon \mathbb{N} \to \mathbb{N}$ to be computable if we can write a finite algorithm that computes f and halts on any input. We are only considering functions with domain $\mathbb{N}$. A subset of the natural numbers is computable if there is a computable function that halts on any input and tells us if the input is in the set. For a graph $G = (V, E)$ we can consider its vertex set as a set of natural numbers, each number identifying a vertex. Then each edge in E is a pair of natural numbers, and by using a pairing function we can consider each edge as a natural number. Hence E can be considered as a subset of $\mathbb{N}$. In this way we can encode all (countable) sets (that we will consider) as subsets of $\mathbb{N}$. Thus G is computable if both V and E are computable sets of natural numbers.

Computable graphs are natural objects to construct and ponder the action of algorithms on. In this spirit we consider, when do computable graphs have computable colourings? That is, when is there an algorithm **A**, such that when given any vertex in any computable graph **A** will compute the colour of that vertex? A computable k-colouring is a computable function $f\colon V \to \{1, \dots, k\}$ such that $f^{-1}(i)$ is an independent set for all $i \in [1, k]$. It is not always the case that having a k-colouring guarantees there is a computable k-colouring. For example, while every planar graph is 4-colourable, there exists computable planar graphs which are not k-colourable for any k [6].

Another notion we are motivated to introduce from computable graph theory is, what happens when a graph is 'more computable'. For example, what if for every vertex v, there is an algorithm A such that $A(v) = \delta(V)$? The notion of 'more computable' was motivated by a desire to expand algorithms from finite combinatorics to infinite graphs. There is further motivation to find under what conditions (both algorithmic, and structural) do computable graphs have desired properties (e.g. computable colourings). For an overview of these notions see Gasarch 1998 [11]. One such notion introduced by Bean 1976 [6] is *highly computable graphs*[2]. A computable graph $G = (V, E)$ is *highly computable* if the function $\delta(v) : V \to \mathbb{N}$ is also computable. This means a highly computable graph is locally finite (every vertex has finite degree) and the neighbour relation is computable. That is, $N(v)$ is a computable set for all $v \in V$. To compute $N(v)$ we enumerate through V looking for neighbours of v, and once $\delta(v)$ many neighbours have been found we have

[2]In the literature the term *highly recursive graph* is often used. However, in keeping with more modern terminology we call these graphs highly computable.

found $N(v)$.

There has been much work in the area of computable graph theory, for some examples see Schmerl 1980 [52], Kierstead 1981 [51], or surveys Gasarch 1998 [11] and Kierstead 1998 [10]. The overall theme in this area is asking, what structure do we need to have in our graph to give us a computable colouring? Or, how bad does the colouring get when we consider more parameters?

There is a strong connection between online algorithms and computable graph theory (Kierstead 1998 [10]). In this vein many results in computable graph colourings transfer to online colouring. For example, consider the proofs of the following, where the computable version gives rise to the online version.

- Kierstead and Trotter 1981 [53]:
  - Every computable interval order of width w can be partitioned into $3w - 2$ computable chains.
  - Every online interval order of width w can be online partitioned into $3w - 2$ chains.
- Kierstead 1981 [38]:
  - Every computable partial order of width w can be partitioned into $^{(5^w - 1)}/_4$ computable chains.
  - Every online partial order of width w can be online partitioned into $^{(5^w - 1)}/_4$ chains.
- Bean 1976 [6]:
  - There is a 3-colourable, computable planar graph which, for all k, has no computable k-colouring.
  - There is a 3-colourable, online planar graph which, for all k, has no online k-colouring.

In this notion we will introduce a new *algorithmic parameterization* we call strongly online (this is defined below in Section 5.3). This is an online analog of highly computable. We would hope that in the spirit of the above results we would get a nice translation from highly computable to strongly online. However, as we will see, results about highly computable graphs do not tend to transfer to strongly online. For example, perfect, k-colourable, highly computable graphs are computably $k + 1$ colourable. However, there is a perfect, k-colourable, strongly online graph that is not $k + 1$-colourable.

5.2 Online Graphs and an Analogy

Suppose you find yourself navigating a large maze. The maze can interpreted as a graph, with vertices as cells and edges between cells if you can move from one to the other. In a classical online setting you may find a cell x. Then at later stages more cells may be reveled connecting to x. In effect new egresses have opened up in the maze. This is clearly not realistic. For this situation to be true, the maze would have to be magical.

Now suppose instead that the maze is fair (i.e. not magical). Any time a cell in the maze is revealed you can see all the egresses from this cell. The egresses that you can see do

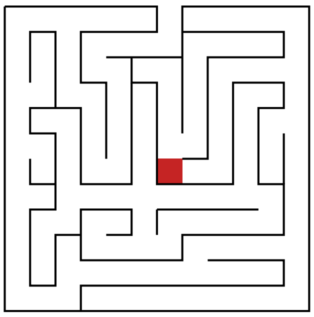


Figure 5.1: A maze

not change. In effect every time a vertex is revealed you get all the neighbours of that vertex as well. This leads us to the notion of (locally) strongly online graphs

5.3 Strongly Online Graphs

Definition 5.1 (Strongly Online Graph [Downey and Askes])**.** For $G = \lim_s G_s$ we define the *strongly online graph* G as follows.

1. A *locally strongly online* graph is an online graph $G^{\prec}$ with a sequence (filtration) $\langle H_s \rangle_{s \in \mathbb{N}}$ such that
$$H_s = N[G_s^{\prec}]$$
where v_s is the vertex that must be coloured at stage s.

2. A *strongly online* graph is an an online graph $G^{\prec}$ with a sequence (filtration) $\langle H_s \rangle_{s \in \mathbb{N}}$ such that H_1 is the subgraph induced by $N[v_1]$ and
$$H_{s+1} = N[H_s] \cup N[G_s^{\prec}]$$
where v_s is the vertex that must be coloured at stage s.

3. A (locally) strongly online graph is *strongly connected* if at each stage s, H_s is connected. That is, the next vertex presented (v_{s+1}) is in H_s.

At each stage s we must colour v_s, but we can see H_s. Note that $\lim_s H_s = \lim_s G_s = G$.

In other words, a strongly online graph is an online graph where at each stage s we can see a ball of increasing radius about each vertex. A locally strongly online graph is an online graph where we can see the neighbours of every vertex we must colour.

When colouring strongly online graphs we can *assign* a colour to vertex in H_s, or we can irrevocably *colour* a vertex in G_s. Because we must colour the vertices in the online presentation order, a simple strategy is to *assign* colours to vertices that we can see, but have not been presented (i.e. not in G_s). Henceforth, we will not make the distinction between assigning colours and colouring vertices. The assumption is that at each stage of a strongly online graph we will colour the vertex $v_s \in V(G_s \backslash G_{s-1})$ that must be coloured according to its assigned colour.

Likewise, when presenting/describing a strongly online graph and we do not specify which vertex to present next, it can be assumed we are presenting a vertex in V_{s-1}. This vertex has been revealed and its neighbours are in the next boundary ($N(V_{s-1})$). Hence, the vertex will not interfere with later operations.

Further, for simplicity we may refer to a strongly online graph as $G = \lim_s G_s$ and make no reference to a filtration. In this case G_s is assumed to be the s-th stage in the filtration.

An *online algorithm* **A** for colouring a (locally) strongly online graph $G = \lim_s G_s$ with filtration $\langle G_s, H_s \rangle$ is a computable function $\mathbf{A} : (G_s, H_s) \mapsto c$ where c is a colouring of G_s, and $\mathbf{A}(G_{s+1}, H_{s+1}){\upharpoonright}_s = \mathbf{A}(G_s, H_s)$. That is, an online algorithm is a function that takes as input G_{s+1} and the visible graph H_{s+1} and outputs a colouring of G_{s+1} that extends the colouring of G_s.

For a strongly online graph $G = \lim_s G_s$ with filtration $\langle H_s \rangle$ the *boundary* at stage s is the induced subgraph $\hat{H}_s = (\hat{V}_s, \hat{E}_s)$, where $\hat{V}_s = V_s \setminus V_{s-1}$ and $\hat{E}_s = E_s \setminus E_{s-1}$. We also note that $\hat{G}_s = G_s \backslash G_{s-1}$, and $\hat{H}_s = G_s \backslash G_{s-1}$.

5.3.1 Colouring Strongly Online Graphs

A strategy to colour highly computable graphs introduced by Bean 1976 [6] is to colour odd stages with one set of colours and even stages with another set. This strategy was used to prove Theorem 5.2 in the context of highly recursive graphs. We present a modification to the context of strongly online graphs.

Theorem 5.2. *If G is a strongly online graph, then G can be strongly online coloured in $2\chi(G)$ colours.*

Proof. Let $G = \lim_s G_s$ be a k-colourable strongly online graph.

We define the colouring at odd and even stages. In the even stages we use colours $1, \dots, k$ and in the odd stages $k+1, \dots, 2k$.

We colour $G_0 = \emptyset$ with no colours.

For the even stage $2s+2$, suppose we have coloured G_{2s+1} using $2k$ colours. Let v be the next vertex presented. Note that G_{2s+2} is the subgraph induced by $N[V(G_{2s+1}) \cup \{v\}]$. At stage $2s+2$ we can see all of G_{2s+2}. Let $\hat{G}_{2s+2} = G_{2s+2} \backslash G_s$. We assign every vertex in $\hat{G}_{2s+2}$ a colour from $1, \dots, k$. This can be done using brute force as G is k-colourable and $\hat{G}_{2s+2}$ is finite.

The odd stage is analogous, but we colour $\hat{G}_{2s+1}$ with colours from $k+1, \dots, 2k$.

This gives a colouring of G as $\hat{G}_s$ is disconnected from $\hat{G}_{s+2}$. ☐

Theorem 5.2 shows us that being strongly online is a much stronger property than being online. What happens if we restrict ourselves to locally strongly online graphs? As in the 'vanilla' online case, we can force arbitrarily many colours on trees and forests, as witnessed by Corollary 5.4 and Theorem 5.3.

Because every online algorithm is a computable function, there is list of partial computable functions, $\langle \varphi_e \rangle$, that contains all possible online algorithms. Once we can list all the online algorithms, we can diagonalise against them. We do this by constructing infinity many graphs $A_1, A_2, \dots$ such that the A_i-th graph cannot be coloured by the i-th function (φ_i) in our list. As each A_i is a strongly online graph, $A_i = \lim_s A_{i,s}$.

Let $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a pairing function. Define a strongly online graph $G = \lim_s G_s$ where $G_s = \bigcup_{t \leq s} A_{\pi^{-1}(t)}$. We then have a graph $G = \bigcup_i A_i$, and hence G cannot be online coloured by any strategy.

Theorem 5.3. *For every online algorithm* $\mathbf{A}$ *and* $k \in \mathbb{N}$ *there is a locally strongly online tree* G *that cannot be online* k*-coloured by* $\mathbf{A}$*. Indeed, for forests of height* n*, the approximation ratio is* $\Omega(\log n)$[3] *which is no better than the one for normal online trees.*

Proof. Fix $k \in \mathbb{N}$ and $\mathbf{A}$ an online algorithm. By induction on the number of colours $l \leq k$ we construct a locally strongly online forest T such that $\mathbf{A}$ does not give a locally strongly online k-colouring of T. We ensure at each stage l the vertices that need the l-th colour have not been presented yet (but will have been revealed).

For $l = 0$, any single vertex is a tree and cannot be coloured with 0 colours. Let G be a pair of vertices a and b connected by an edge, and present the vertex a. Vertex b is revealed and requires 1 colour, as desired.

Generate $2^k \cdot k^{k+1}$ many G's. By the pigeonhole principle, $2^k \cdot k^k$ of the b vertices from these trees are the same colour. Let F_0 be the set of such vertices.

By the induction hypothesis, assume that there exists $F_0 \ldots F_l$, such that, $|F_m| \geq 2^{k-l} \cdot k^{k-l}$, all vertices in F_m use the same colour, have not been presented, and are adjacent to $m - 1$ different coloured neighbours. That is, F_m uses the m-th colour.

Let $X = x_1, \ldots, x_n$ contain $2^{k-l-1} \cdot k^{k-l}$ new vertices. Connected each x_i to a unique vertex from each $F_0, \ldots, F_l$ by presenting one of x_i's new neighbours. Each x_i is adjacent to l colours and so must get a new colour. See Figure 5.2.

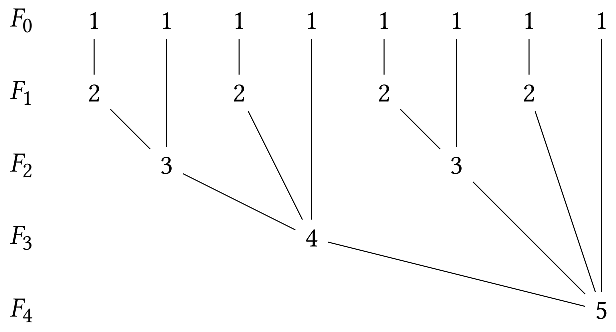


Figure 5.2: A tree that cannot be four coloured (numbers representing colours, and vertices not in a F_i omitted)

By the pigeonhole principle, $2^{k-l-1} \cdot k^{k-l-1}$ many x_i's must use the same colour. Let F_{l+1} be the set of these same coloured vertices.

As $2^{k-l-1} \cdot k^{k-l}$ is less than $1/2(2^{k-l} \cdot k^{k-l})$, each F_i decreases in size by no more than half. That is, each F_i has $|F_i| \geq 2^{k-l-1} \cdot k^{k-l-1}$, as desired.

Note that once $l = k$, we will generate $2^{k-k} \cdot k^{k-k} = 1$ tree that requires a $k + 1$-th colour. Therefore we have a forest that $\mathbf{A}$ cannot k-colour.

[3] $f(x) = \Omega(g(x))$ means that f is bounded below by g asymptotically

Finally, note that T has $O(2^k \cdot k^k)$ many vertices, and hence T requires $\Omega(\log(n))$ colours to colour. □

Corollary 5.4 then follows as a simple corollary of Theorem 5.3 by a diagonalization against all possible algorithms and numbers of colours.

Corollary 5.4. *There is an (infinite) locally strongly online forest which cannot be online finitely coloured.*

We have seen how forests cannot be locally strongly online coloured with finitely many colours. Is there any situation where we can finitely colour forests? In contrast to Corollary 5.4, strongly connected online trees can be 2-coloured. This is because in a strongly connected online tree we can use a standard strategy for colouring offline trees.

Theorem 5.5. *Every strongly connected online tree is strongly 2-colourable.*

Proof. Let $T = \lim_s T_s$ be a strongly connected online tree. We colour T by induction on s, the stage.

For the base case we colour v_1 with 1 and $N(v)$ with 2. This colours T_1.

Suppose we have coloured T_s. We must now colour T_{s+1}. Let $\hat{T}_{s+1} = T_{s+1} \backslash T_s$. Colour $\hat{T}_{s+1}$ with whichever colour we didn't use in $\hat{T}_s$.

As each v_s is part of the graph we can see, it has already been assigned a colour. And, as each $\hat{T}_s$ is independent, we have strongly 2-coloured T. □

5.4 Suspected Improvements

We have taken strong hints from highly computable graph theory about how to proceed. In this vein we will look a several results on highly computable graphs that show improvements in the context of highly computable graphs over the theorems we have seen so far. We would hope these results transfer to strongly online graphs.

In the context of highly computable graphs we can get an improvement on Theorem 5.2, as evidenced by Theorem 5.6.

Theorem 5.6 (Schmerl 1980 [52])**.** *If G is a highly computable, k-colourable graph then, G is computably $(2k-1)$-colourable.*

Further improvements can be made on highly computable graphs if we restrict the class of graphs further. Kierstead 1981 [51] takes the ideas from Theorem 5.6 and uses them to prove Theorem 5.8.

Definition 5.7 (Perfect Graph)**.** A graph G is *perfect* if for every induced subgraph H of G, $\chi(H) = w(H)$, where $w(H)$ is the size of the largest clique in H.

Theorem 5.8 (Kierstead 1981 [51])**.** *Every perfect, k-colourable, highly computable graph is computably $k+1$ colourable.*

Using Theorems 5.6 and 5.8 as templates, we can then ask the following questions.

Question 5.9. Is every k-colourable strongly online graph strongly $(2k-1)$-online colourable?

Question 5.10. Is every perfect, k-colourable, strongly online graph strongly $k+1$ online colourable?

Unfortunately, in general, the answers to questions 5.9 and 5.10 are no. As we will see in Section 5.5, there are graphs that require twice their chromatic number to colour.

5.5 Lower Bounds

A common strategy for highly computable algorithms is to colour every vertex we can see at each stage, or more precisely we colour H_s. This strategy will not work for strongly online graphs, as witnessed by Theorem 5.11.

Theorem 5.11. *For all k there is a strongly online k-colourable graph that cannot be strongly $(2k-1)$-online coloured.*

Proof. We construct the graph in stages. Let $\langle \varphi_e \rangle$ be an enumeration of all possible colourings (i.e. functions $\mathbb{N} \mapsto [2k-1]$). At stage s, we diagonalise against φ_s by constructing a strongly online graph H_s that cannot be strongly online coloured by φ_s. We then let $G = \bigcup_s H_s$.

Stage s: Note that a strongly online graph is a sequence $H_{s,1}, H_{s,2}, \ldots$. Begin by letting $H_{s,1}$ be a k-clique. Let $H_{s,t+1}$ be $H_{s,t}$ with another k-clique and the neighbours of each k-path in $H_{s,t}$. See Figure 5.3. We repeat this until there are

$$n = 2(k-1)\binom{2k-1}{k-1} + 1$$

k-paths. Let this graph be $H_{s,q}$. Let X_i be the last vertices revealed in each k-path in $H_{s,q}$. If φ_e is a valid colouring of $H_{s,t}$ then, by the pigeonhole principle, φ_s must have coloured $2k$ X_i's using the same set of colours. Let I denote the set of indices of these sets.

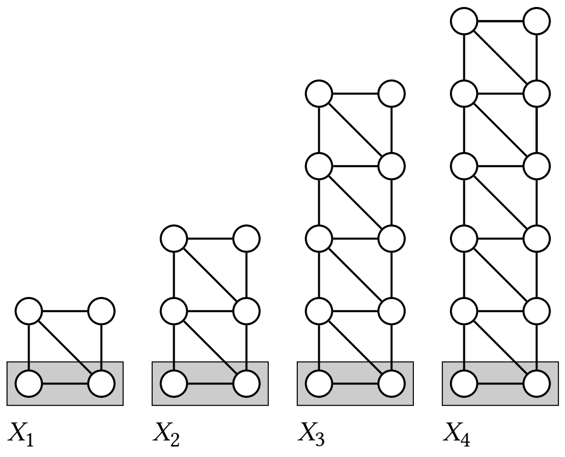


Figure 5.3: The first 4 paths for $H_{s,4}$ when $k = 3$

Let $A = (\{a_1, \ldots, a_k\}, E_A)$ be a k-clique (complete graph). We let $H_{s,q+1}$ be $H_{s,q} \cup A$ along with edges $\bigcup_{i \in I} \{(u, a_i) : u \in X_i\}$. See Figure 5.4. Without loss of generality we can assume that φ_s uses colours $1, \ldots, k-2$ to colour X_i. Then A will be coloured using $k-1, \ldots, 2k-1$.

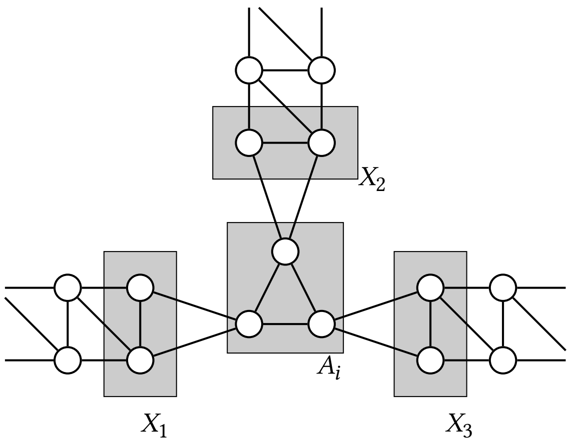


Figure 5.4: The graph $H_{s,q}$ for $k = 3$

We add $2k$ such cliques, each one connected to different X_i's. Let $\mathcal{A} = \{A_i : i \in J\}$ be the set all of these k-coloured cliques. Denote $A_i = \{a_{i,1}, a_{i,2}, \dots, a_{i,k}\}$ where, for $j \in [1, k]$, $\varphi_s\left(\bigcup_{i \in J}\{a_{i,j}\}\right)$ is a single colour. Let $B = (\{b_1, \dots, b_k\}, E_B)$ be another k-clique (complete graph). We let $H_{s,q+2}$ be $H_{s,q+1} \cup B$ along with the edges

$$\bigcup_{i \in J}\{(b_i, a_{2i-1,1}), \dots, (b_i, a_{2i-1,k-1}), (b_i, a_{2i,k}), \dots, (b_i, a_{2i,2k})\}$$

See Figure 5.5.

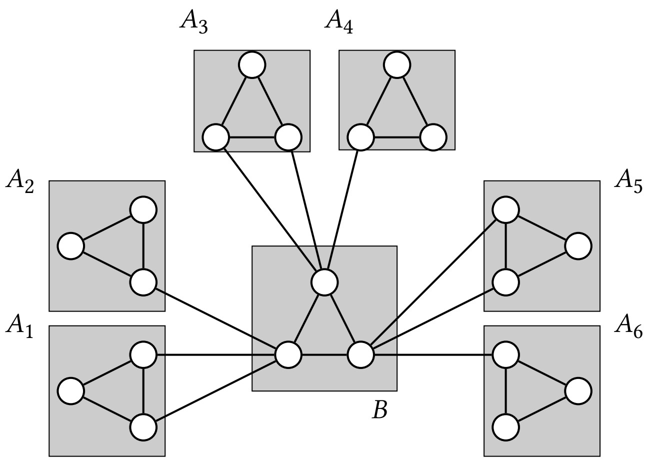


Figure 5.5: The graph H_s for $k = 3$, with colours

This ends the construction. Each vertex in B is connected to k uniquely coloured neighbours and is part of a k-clique, and thus cannot be $2k-1$ coloured. Therefore φ_e cannot strongly online colour H_s. A k-colouring of H_s can be found by colouring B then using breadth first search and the greedy strategy.

Therefore, $G = \bigcup_s H_s$ is a k-colourable graph that cannot be coloured by any strategy. $\square$

The graph constructed in Theorem 5.11 has $O\left(\binom{2k-1}{2}\right) = O((2k-1)^{k-1})$ vertices. Thus we need $O(\log n)$ colours to strongly online colour a graph with n vertices.

We can now see that the answer to question 5.9 is no. For the case of general graphs we cannot do better than twice the chromatic number. However, we can do better for more restricted classes of graphs. For example, every graph of pathwidth 1 is strongly online colourable in 3 colours (see Theorem 5.15). As we will see in Theorem 5.15, for graphs of even bounded pathwidth the answer to question 5.9 is yes.

5.5.1 Online with Lookahead

One restriction of strongly online colouring vs colouring highly computable graphs is that when strongly online colouring, there is a vertex that must be coloured at each stage. In a highly computable graph after a vertex is presented we could wait before colouring it. It is this fact that Theorems 5.6 and 5.8 use to improve on Theorem 5.2.

The added ability to wait in highly computable graphs implies that our definition of strongly online is limited. To rectify this, we propose an alternative version of strongly online.

Definition 5.12 (Strongly Online Graph with Lookahead)**.** Fix some $d \in \mathbb{N}$, and $G = \lim_s G_s$ a strongly online graph with filtration $\langle H_s \rangle_{s \in \mathbb{N}}$. The graph G has *lookahead d* if the vertex $v_s \in G_s \backslash G_{s-1}$ is the vertex that must be coloured at stage $s + d$.

In short a strongly online graph with lookahead d allows you to wait d stages before colouring the vertex v_s.

Even lookahead is not enough for strongly online graphs to have the same bounds as highly computable graphs. To demonstrate this, consider Theorem 5.11 and its proof. At no stage do we wait for a vertex to be irrevocably coloured. We are operating on the colours assigned by the function we are diagonalising against. Therefore we have Theorem 5.13.

Theorem 5.13. *For all k, d there is a strongly online k-colourable graph with lookahead d that cannot be strongly $(2k-1)$-online coloured.*

Furthermore, the graph in Theorem 5.11 is perfect, of bounded degree, and bounded treewidth. Thus, for all these classes of graphs, the bound of twice the chromatic number is tight.

Theorem 5.14. *For all k, d there is a perfect strongly online k-colourable graph with lookahead d, pathwidth $k-1$, and max degree $2k-1$ that cannot be strongly $(2k-1)$-online coloured.*

Proof sketch. It suffices to show that the graph G generated by Theorem 5.11 is perfect, has pathwidth $k-1$, and max degree $2k-1$.

As every component in G follows the same structure, we can assume, without loss of generality, that G is just one of these connected components. We begin by showing G has max degree $2k-1$. Let b be a vertex in B. Vertex b is connected to $k-1$ vertices inside B, and at most k vertices from some A_i's. Thus $\delta(b) \leq 2k-1$. Let a be a vertex in some A_i. Vertex a is connected to $k-1$ vertices from A_i, 1 vertex from B, and $k-1$ vertices from some X_i. Thus $\delta(a) \leq 2k - 1$. Let x be some vertex in some X_i. Vertex x is connected to one vertex in A_i, $k - 2$ vertices in X_i, and $k - 1$ vertices from the k-path that terminates with X_i. Thus $\delta(x) \leq 2k - 1$. The remaining vertices form parts of k-paths, and have at most $2k$ neighbours. Therefore every vertex in G has max degree $2k - 1$.

Next we show that G has treewidth k. Consider B the root of a tree decomposition. Then all the A_i's are B's children, and the X_i's the A_i's children. Each of these bags (B, A_i's, and X_i's) all have size at most k. Thus the width of this tree decomposition is $k - 1$.

It remains to show that G is perfect. In the tree decomposition of G all the bags are cliques. Bags B, A_i's, and X_i's are cliques by construction, and the other bags are cliques by inspection of the graph. Thus the only way to make G l-colourable is to remove enough vertices so that every bag has at most $l - 1$ vertices in it. Hence G is perfect. ☐

5.6 Graphs with Bounded Pathwidth

As we have seen, many results seem to hold because of parameterizations related to the shape of a graph. Graphs of bounded pathwidth seem to have a nice structure. For example, any online graph with pathwidth k can be online coloured in $3k - 2$ colours [20]. Whereas, in general, online graphs (including bounded treewidth) require arbitrarily many colours to colour.

Every graph $G = (V, E)$ with pathwidth k has degeneracy k. That is, there is an ordering of V such that each vertex has at most k neighbours smaller that it. Thus, to colour G, we can iterate along this order and greedily colour the vertices. This will use at most $k + 1$ colours because every vertex will have at most k coloured neighbours when it is coloured. Therefore every graph with pathwidth k can be coloured in $k + 1$ colours. Hence, by Theorem 5.2, we can colour any strongly online graph with pathwidth k in $2k + 2$ colours. However, we can improve on $2k + 2$ colours if k is even (see Theorem 5.15). Further, for even pathwidth this bound is tight by Theorem 5.16.

Theorem 5.15. *Every strongly online graph with pathwidth k, where k is even, is strongly online $2k + 1$-colourable.*

Proof. Let $G = \lim_s G_s$ be a strongly online graph with pathwidth k.

Let

$$C_{s,t} = \begin{cases} 1, 3, 5, 7, \ldots, 2t - 1 & \text{if } s \text{ is odd} \\ 2, 4, 6, 8, \ldots, 2t & \text{if } s \text{ is even} \end{cases}$$

At each stage s we will colour the boundary, $\hat{G}_s = (\hat{V}_s, \hat{E}_s)$, with the set of colours $C_{s,l}$ along with a $k+1$-th colour (which we will call *white* for convenience), where l is the pathwidth of $\hat{G}_s$.

When colouring $\hat{G}_s$ we will ensure the colour white is only used in a connected component in $\hat{G}_s$ that has pathwidth k. All other components will use the colours $C_{s,m+1}$, where m is the pathwidth of the connected component.

The first stage ($s = 1$) has pathwidth at most k and so can be coloured appropriately with $C_{1,k}$ and white.

Suppose we have coloured G_s, and we need to colour $\hat{G}_{s+1}$. Let H be the subgraph induced by a connected component in $\hat{G}_{s+1}$. Suppose H has pathwidth $m < k$. Then as the only neighbours of H have colours $C_{s,k}$ and white, we can colour H with $C_{s+1,m+1}$.

Now, suppose H has pathwidth k. We must now find a way to reuse the colour white. Let $\mathcal{P} = \{P_1, P_2, P_3, \dots\}$ be a path decomposition of $N[H]$. There are three cases.

(a) H is adjacent to no connected components of width k in $\hat{G}_s$.

(b) H is adjacent to at most 1 connected component of width k in $\hat{G}_s$.

(c) H is adjacent to more than 1 connected component of width k in $\hat{G}_s$.

Case (a): H is not adjacent to any white vertices. Thus any colouring of H using $C_{s+1,k}$ and white is valid.

Case (b): Let P_i be a bag in $\mathcal{P}$ that contains a white vertex from $\hat{G}_s$. Without loss of generality, assume that i is less than the index of any bag that contains only vertices from H. That is, P_i is to the left of H. We use Algorithm 5.1 to find $X \subseteq V(H)$ such that X is 1-colourable, $H \backslash X$ has pathwidth $k-1$, and no vertex in X is adjacent to a white vertex in G_s.

Algorithm 5.1 Find the vertices of H to colour white

```
1:  procedure FindVertices($i$)
2:      $j \leftarrow i$
3:      $X \leftarrow \emptyset$
4:      while $P_j \neq \emptyset$ do
5:          if $P_J \subseteq V(H)$ and $P_j \cap X = \emptyset$ then
6:              Fix some $x \in P_j \setminus P_{j-1}$
7:              $X \leftarrow X \cup \{x\}$
8:          end if
9:          $j \leftarrow j + 1$
10:     end while
11:     return $X$
12: end procedure
```

Colour all vertices in X and the remaining vertices in H, with $C_{s+1,k}$.

Case (c): Assume for a contradiction that H is connected to 3 connected components with width K, and let D,E, and F be their induced subgraphs from $\hat{G}_s$. Because each of D, E, F,and H have pathwidth k, there are bags P_d, P_e, P_f, and P_h such that $P_d \subset V(D)$, $P_e \subset V(E)$,$P_f \subset V(F)$, $P_h \subset V(H)$, and $|P_d| = |P_e| = |P_f| = |P_h| = k+1$.

Without loss of generality, assume that $d < h < e < f$. As there is a path from P_h to D_f, every bag between P_h and D_f must contain a vertex from $H \cup F$. This contradicts the fact that $h < e < f$. Therefore H is connected to two connected components of pathwidth k.

Claim 5.15.1. *Every bag $P_i \in \mathcal{P}$ such that $P_i \subseteq V(H)$ and $|P_i| = k+1$ has at most $^{k+1}/_2$ vertices that are adjacent to $k+1$ colours from $\{P_1, \dots, P_{i-1}\}$ (or $\{P_{i+1}, P_{i+2}, \dots\}$).*

Proof of claim. Fix $P_i \in \mathscr{P}$ such that $P_i \subseteq V(H)$ and $|P_i| = k+1$. Let $\{p_1, \ldots, p_n\}$ be the vertices in P_i adjacent to exactly $k+1$ colours from $\{P_1, \ldots, P_{i-1}\}$.

Let P_j be the greatest $j < i$ such that P_j contains $k+1$ colours (cannot be in H as H has not been coloured). No bag between P_j and P_i can contain white (or else there would be a third connected component with width k). Let y be the white vertex in P_j. $p_1, \ldots, p_n$ are adjacent to y. Further there is some bag P_l such that $p_1, \ldots, p_n, y \in P_l$.

Note that the maximum width of a connected component from $\hat{G}_s$ between P_l and P_i is $k - n$. Hence $p_1, \ldots, p_n$ can only be adjacent to less than $k+1-n$ colours not in P_l.

In P_{l-1} there are only $n-1$ vertices from $p_1, \ldots, p_n$. Hence the maximum number of colours from P_j that $p_1, \ldots, p_n$ is adjacent to is $k+1-n$.

If $n > {}^{k+1}/_2$, then the number of colours not from P_l and from P_j is

$$(k+1-n) + (k+1-n) < 2k+2-2({}^{k+1}/_2) = k+1$$

This contradicts the fact that $p_1, \ldots, p_n$ are adjacent to $k+1$ colours from $\{P_1, \ldots, P_{i-1}\}$. Therefore n is at most ${}^{k+1}/_2$. ■

Let A and B be the induced subgraphs of the connected components of pathwidth k that H is adjacent to.

Let P_l be the leftmost and P_r the rightmost bag in $\mathscr{P}$ such that $P_l \subseteq V(H)$ and $P_r \subseteq V(H)$. By Claim 5.15.1 and the fact that k is even, P_l and P_r have more than ${}^{k+1}/_2$ vertices adjacent to at most k colours from $\hat{G}_s$.[4] See Figure 5.6.

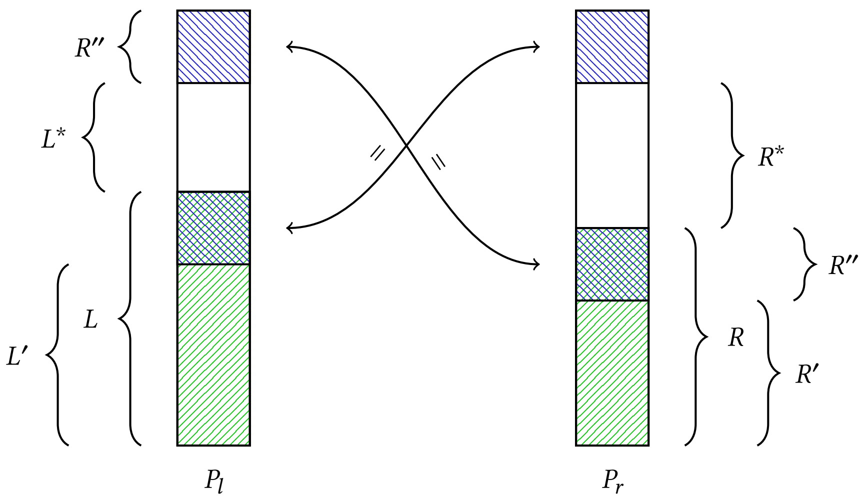


Figure 5.6: The bags P_l and P_r

We define the following subsets of P_l and P_r.

- $L \subseteq P_l$ is the subset of P_l that is not connected to $k+1$ colours from $\{P_1, \ldots, P_{l-1}\}$.

[4]This fact fails to be true if k is odd. We could have exactly half of the vertices in H connected to $k+1$ colours to the left and the other half connected to $k+1$ colours to the right. Therefore, we might not be able to colour H

- $R \subseteq P_r$ is the subset of P_r that is not connected to $k+1$ colours from $\{P_{r+1}, P_{r+2}, \dots\}$.
- $L' \subseteq L$ such that for every $x \in L'$, $x \notin P_r$.
- $R' \subseteq R$ such that for every $y \in R'$, $y \notin P_l$.
- $L^* = P_l \setminus L \setminus P_r$.
- $R^* = P_r \setminus R \setminus P_l$.
- $R'' = R \setminus R'$ ($= P_l \setminus L \setminus L^*$).

Note that $|L^*| = k + 1 - |L| - |R''|$ and $|R'| = |R| - |R''|$. Hence

$$|L^*| \le k + 1 - |{}^{k+1}/_2| - |M| < {}^{k+1}/_2 - |M| \le |R'|$$

Thus $|L^*| < |R'|$.

Therefore any colouring of H must have one colour that is used in both L' and R'. Let $x \in L'$ and $y \in R'$ be some such vertices. If both x and y are not adjacent to the colour white, then we can colour these vertices white and are done.

Suppose one of x and y is adjacent to the colour white. Then one of L^* and R^* (respectively) has one less vertex in it. Hence L' (or R') is one vertex larger. Therefore in any colouring of H, there are two colours used in both L' and R'.

By applying induction we see that either we will be able to colour H, $L = P_l$ or $R = R_r$. We then use the technique from case (b) to colour H as A or B effectively has pathwidth less than k.

It remains to show that no neighbour of H in later stages can have pathwidth k.

Claim 5.15.2. *If F is the induced subgraph of a connected component in $G \backslash G_s$ that is connected to H, then F has pathwidth less than k.*

Proof of claim. Let $\mathcal{Q} = \{Q_1, Q_2, \dots\}$ be a path decomposition of G. Because each of A, B, and H have pathwidth k, there must be bags Q_a, Q_b, and Q_h such that $Q_a \subset V(A)$, $Q_b \subset V(B)$, $Q_h \subset V(H)$, and $|Q_a| = |Q_b| = |Q_h| = k + 1$.

Assume for a contradiction that F has pathwidth k. Then there is also a bag $Q_f \subseteq V(F)$, where $|Q_f| = k + 1$.

In G there are paths Q_a to Q_h and Q_b to Q_h. $Q_f \cap V(H) = Q_f \cap V(A) = Q_f \cap V(B) = \emptyset$ and so f cannot be between a and b because the paths cannot pass through Q_f. That is, $a > f$ and $b < f$.

f cannot be greater than b because there is a path from Q_f to Q_h and $Q_b \cap (V(h) \cup V(F)) = \emptyset$. Similarly f cannot be less than a.

This is a contradiction as Q_f must exist somewhere in $\mathcal{Q}$. Therefore F cannot have pathwidth k. ■

This means we can always use k colours to colour later neighbours of H. □

The following result shows that for arbitrary graphs with pathwidth k, we cannot do better than $2k + 1$.

Theorem 5.16. *There is a strongly online graph of pathwidth k that is not strongly online $2k$-colourable.*

Proof. As in Theorem 5.11, we construct the graph in stages. Let $\langle \varphi_e \rangle$ be an enumeration of all possible colourings (i.e. functions $\mathbb{N} \mapsto [2k]$). Again the graph is constructed as disjoint components and we will diagonalise against φ_s by constructing a strongly online graph H_s with path width k that cannot be strongly online coloured by φ_s. We then let $G = \bigcup_s H_s$. We return to the construction of component H_s which is constructed in stages e.

Stage e: Recall that a strongly online graph is a sequence $H_{s,1}, H_{s,2}, \dots H_{s,e-1} \dots$. We construct a series of k-paths, each with vertices $x_1, x_2, x_3, \dots$ and all the edges of the form (x_i, x_j) such that $i - j < k$. Let $\{P_1, P_2, P_3, \dots\}$ denote the set of all such k-paths. Let $H_{s,1}$ contain P_1 with k vertices. Let $H_{s,i+1}$ contain P_{i+1} with k vertices and all P_j such that $j < i$ with an extra k vertices. See Figure 5.3 for an example when $k = 3$.

We repeat this until there are

$$n = \binom{2k}{k-1} + 1$$

k-paths. If it is the case that φ_s halts we will observe the online colouring it has given. By the pigeonhole principle there are two paths whose last $k - 1$ vertices have the same colour. Denote these vertex sets $A = \{a_1, \dots, a_k\}$ and $B = \{b_1, \dots, b_k\}$.

Without loss of generality, assume that $c(a_i) = c(b_i) = i$. Let $D = \big(\{d_1, \dots, d_{k+1}\}, E_D\big)$ be a $(k + 1)$-clique (complete graph). We let $H_{s,n+1} = H_{s,n} \cup D$ along the edges of the form (d_i, a_j) such that $i \le j$ and (d_i, b_j) such that $i > j$. This forms a single k-path See Figure 5.7.

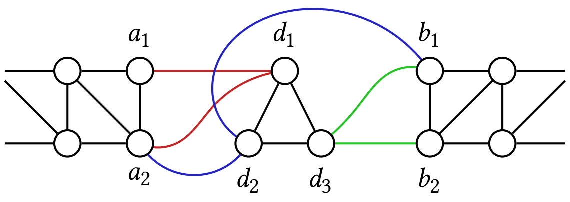


Figure 5.7: The graph $H_{s,n+1}$ with edges coloured for clarity

Each vertex in D is connected to a vertex of each colour $1, \dots, k - 1$ and thus must be coloured from the colours $k, \dots, 2k$. However as D is a $(k + 1)$-clique it requires $k + 1$ colours. Hence D cannot be coloured. Therefore H_s cannot be coloured by φ_s.

H_s is a series of k-paths and so has pathwidth k.

Therefore, $G = \bigcup_s H_s$ is a graph with pathwidth k that cannot be coloured by any strategy. ☐

In general, for graphs with odd pathwidth k we can also do no better than $2k + 2$, as witnessed by Theorem 5.17.

Theorem 5.17. *There is a strongly online graph of odd pathwidth k that is not strongly online $2k + 1$colourable.*

Proof. As in Theorem 5.11, we construct the graph in stages. Let $\langle\varphi_e\rangle$ be an enumeration of all possible colourings (i.e. functions $\mathbb{N} \mapsto [2k]$). The graph is constructed as disjoint components and we will diagonalise against φ_s by constructing a strongly online graph H_s with pathwidth k that cannot be strongly online coloured by φ_s in $2k + 1$ colours.

Note that since k is odd $k + 1 = 2l$ for some $l \in \mathbb{N}$.

Stage e: To construct H_s we begin by presenting

$$2\binom{2k+1}{k+1} + \binom{2k+1}{l} + 2$$

vertices, each the center of a path that grows in length with every presented vertex.

Next we create $\binom{2k+1}{k+1} + 1$ cliques of size $k + 1$ and $\binom{2k+1}{l} + 1$ cliques of size l.

If φ_e halts, we will observe the following. First, there will be two cliques X_1 and X_2 of size $k + 1$ which use the same colours. Second, there are two cliques Y_1 and Y_2 of size l which use the same colours. Without loss of generality, let the colours used in Y_1 and Y_2 be $1, 2, \dots, l$, and the colours used in X_1 and X_2 be $1, 2, \dots, k + 1$.

We join the vertices coloured $1, \dots, l$ in X_1 (respectively X_2) to the end of one of our paths, call this vertex a_1 (respectively a_2). We do the same for the vertices coloured $l + 1, \dots, k + 1$ in X_1 (respectively X_2), who join to the end of one of our paths, call this vertex b_1 (respectively b_2).

Next join half of Y_1 (respectively X_2) to some path, call this c_1 (respectively c_2), and the other half to some path and call it d_1 (respectively d_2).

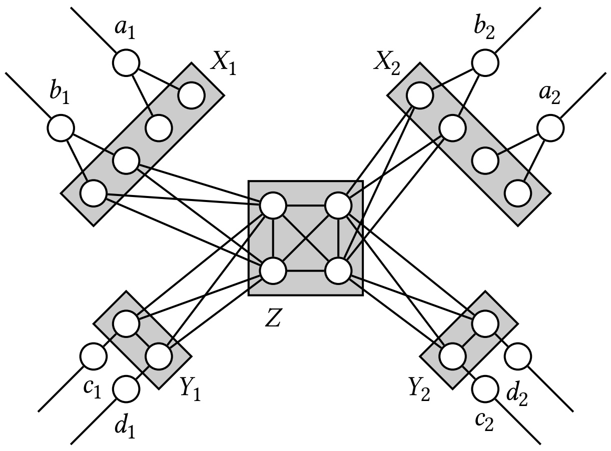


Figure 5.8: The graph H_s for pathwidth 3

Next we reveal another clique, Z, of size $k + 1$ with vertices $z_1, \dots, z_{k+1}$. For each $i \leq l$, join the vertex z_i to every vertex in Y_1 and every vertex with colour greater than l in X_1.

For $i > l$, connect z_i to every vertex in Y_2 and every vertex with colour greater than l in X_2 See Figure 5.8 for an example.

Each vertex in Z is connected to $k+1$ colours (colours $1, \dots, l$ from Y_i and colours $l+1, \dots, k+1$ from X_i) and is part of a $k + 1$ sized clique. Therefore Z cannot be coloured with k colours.

It remains to show that H_s has pathwidth k. The path decomposition of H_s contains the following bags in order, along with the interpolation of any necessary vertices.

1. $a_1 \cup N(a_1) \cap X_1$
2. X_1
3. $b_1 \cup N(b_1) \cap X_1$
4. $(N(X_1) \cap Z) \cup (N(Z) \cap X_1)$
5. The path adjacent to c_1
6. $c_1 \cup N(c_1) \cap Y_1$
7. Y_1
8. $d_1 \cup N(d_1) \cup Y_1$
9. The path adjacent to d_1
10. Z
11. The path adjacent to d_2
12. $d_2 \cup N(d_2) \cup Y_2$
13. Y_2
14. $c_2 \cup N(c_2) \cup Y_2$
15. The path adjacent to c_2
16. $(N(X_2) \cap Z) \cup (N(Z) \cap X_2)$
17. $b_2 \cup N(b_2) \cap X_2$
18. X_2
19. $a_2 \cup N(a_2) \cap X_2$

By the construction of H_s, each of these bags will contain no more than $k + 1$ vertices. Hence H_s has pathwidth k.

Therefore $G = \bigcup_s H_s$ is a strongly online graph with odd pathwidth k that cannot be $2k + 1$-coloured. □

5.6.1 Strongly Connected

By changing our presentation slightly to strongly connected, we can improve on Theorem 5.15 for both even and odd pathwidth. But first we need a lemma.

Lemma 5.18. *Let G be a connected non-empty graph and H a super-graph of G such that all vertices in H are either in G or areconnected to at least one vertex in G and the pathwidth of H is k. Then the pathwidth of $\hat{H} = H \backslash G$ has pathwidth less than k.*

Note that this lemma is essentially a single step in a strongly online graph. In a strongly online graph H would be the graph visible in the stage after G was visiable.

Proof. Let $\mathcal{P} = P_1, \dots, P_n$ be a path decomposition of H and $\mathcal{Q}$ be $\mathcal{P}$ with only vertices in $\hat{H}$.

Assume for a contradiction that $\mathcal{Q}$ has width k. Then there is a $P \in \mathcal{Q}$ such that $|P| = k+1$. Note that $P = P_i \in \mathcal{P}$. There are three cases.

(1) There are bags P_l and P_r in $\mathscr{P}$ such that $l < i$ and $r > i$.

(2) There is a bag P_l and no P_r in $\mathscr{P}$ such that $l < i$ and $r > i$.

(3) There is no bag P_l and a bag P_r in $\mathscr{P}$ such that $l < i$ and $r > i$.

If (1) is true, then there is no path from P_l to P_r that does not contain a vertex in P_i. This contradicts the fact that G is connected.

Suppose (2) is true, then there is some vertex in $v \in P_i$ such that $v \notin P_{i-1}$. Thus v is not connected to G, a contradiction.

Suppose (3) is true, then there is some vertex in $v \in P_i$ such that $v \notin P_{i+1}$. Thus v is not connected to G, a contradiction. □

Theorem 5.19. *Every strongly online graph, G, with pathwidth k that is strongly connected can be strongly online coloured in $2k$ colours.*

Proof. Let $G = \lim_s G_s$ be a strongly online graph. For each $s \geq 1$, we colour $\hat{V}_{2s}$ with the colours $1, \ldots, k$, and $\hat{V}_{2s+1}$ with the colours $k+1, \ldots, 2k$.

We proceed by induction on the stage s.

For stage 1, we colour v_s 1.

Stage 2: Let $\mathscr{P}_2$ be a path decomposition of G_2. $\mathscr{P}_2$ has width k. We colour G_2 by using a greedy strategy on $\mathscr{P}_2$ using the colours $k+1, \ldots, 2k$.

For any vertex v, denote the set of neighbours that have already been coloured as $N^+(v)$ when we attempt to colour v.

Suppose we run into a vertex v such that $N^+(v) = k$. Let B_i be the leftmost bag containing v. Any $u \in N^+(v)$ must be in B_i. Because if $u \notin B_i$, then B_i would not be the leftmost bag containing v. Note that $|B_i| = k + 1$. By Lemma 5.18, B_i cannot contain only vertices from $\hat{V}_2$ (or else $\hat{V}_2$ would have pathwidth k). Let u be a vertex in B_i that is not in $\hat{V}_2$. Flag u and pretend it does not exist while we finish colouring G_2. Now $|N^+(v)| < k$ and we can colour v.

We now have to colour all of the flagged vertices. Let $X = x_1, \ldots, x_n$ be the set of flagged vertices. G_2 has pathwidth k and each x_i is connected to $G_2 \backslash X$. Thus by Lemma 5.18, X has pathwidth less than k. Therefore we can colour X using the colours $1, \ldots, k$.

If v_s needs its colour changed after we have coloured V_2 swap two colours so that the colour of v_s is still 1.

Stage s: By Lemma 5.18 $\hat{V}_s$ has pathwidth less than k. Thus $\hat{V}_s$ can be coloured in k colours.

Therefore we can $2k$ colour G. □

5.7 Strongly Online Path Decompositions

In light of the usefulness of width metrics, we would hope that there is an online equivalent. We introduce one such possibility for graphs in the form of *strongly online pathwidth*. By adding a single new vertex to a path decomposition at each stage we can create a path decomposition of a (infinite) graph.

Originally we had hoped that if we could find a strongly online path decomposition of width $2k$, then we could use a greedy strategy to colour strongly online graphs with $2k+1$ colours. However, as we will see, this did not eventuate.

Definition 5.20 (Path Decompositon Extension)**.** Let $\mathscr{P}$ and $\mathscr{Q}$ be path decompositions. $\mathscr{Q}$ is an *extension* of $\mathscr{P}$ (denoted $\mathscr{P} \preceq \mathscr{Q}$) if there is an injection $\sigma : \mathscr{P} \to \mathscr{Q}$ such that for all $P_i \in \mathscr{P}$, we have $P_i \subseteq \sigma(P_i)$. For each P_i, P_j such that $P_i \cap P_j \neq \emptyset$, if $i < j$, then $\sigma(P_i) = Q_l$ and $\sigma(P_j) = Q_m$ for some $l < m$.

Definition 5.21 (Strongly Online Path Decomposition)**.** Let $G = \lim_s G_s$ be a strongly online graph. A *strongly online path decomposition* is a sequence $\langle \mathscr{P}_s \rangle_s$ such that

(i) Each $\mathscr{P}_s$ is a path decomposition of $G_s^{\prec}$,

(ii) $\mathscr{P}_s$ is an extension of $\mathscr{P}_{s-1}$, and

(iii) $\mathscr{P} = \lim_s \mathscr{P}_s$ is a path decomposition of G.

That is, using the information in G_s, the vertex v_s must be place inside at least one bag. Other vertices may be placed in bags (including vertices already in a bag), but vertices cannot be removed.

Definition 5.22. The *strongly online pathwidth* of a graph G is the smallest k such that for each strongly online presentation $G = \lim_s G_s$, G has a strongly online path decomposition of width at most k.

In order to construct a strongly online path decomposition of width $2k$, we can only make k mistakes in each bag. However, we can force arbitrarily many mistakes. This is because mistakes can be forced by presenting long paths that join together past the point at which we can see in the graph.

Theorem 5.23. *For all $\Delta \geq 4$, there is a graph G with pathwidth 2 and maximum degree Δ that does not have a strongly online path decomposition of width less than $\lfloor \Delta/2 \rfloor + 2$.*

Proof. Let G be the star graph S_Δ. We present the graph in Figure 5.9 from the center out.

Let each maximal path in the star starting from v be $X_1, X_2, \ldots, X_\Delta$. Once $N^2[v]$ has been presented, let $\mathscr{P}_s$ be the path decomposition at this stage s. $\mathscr{P}_s$ must have each path as a piece of the decomposition. Without loss of generality, assume that the order in which the paths appear in $\mathscr{P}_s$ is $X_1, X_2, \ldots, X_\Delta$. Present vertices $u_1, u_2, \ldots, u_{\lfloor \Delta/2 \rfloor - 1}$ such that vertex u_i joins the end of paths X_i and X_j where $j = \lceil \Delta/2 \rceil + i + 1$. See Figure 5.10.

The path(s), $X_{\lceil \Delta/2 \rceil}$ (and $X_{\Delta 2+1}$ if Δ even), in the center of $\mathscr{P}_s$ will not be joined, but will be between the i and j pair. In order to extend $\mathscr{P}_s$ to $\mathscr{P}_{s+i}$ and include u_i every bag in the paths between i and j, $\mathscr{P}_{s+i}$ must contain an extra vertex in the paths in $\mathscr{P}_s$ not joined to some u_i. Thus each vertex u_i increase the width of the path decomposition by 1. Therefore $\mathscr{P}$ must have width greater than or equal to $\lfloor \Delta/2 \rfloor + 2$. ☐

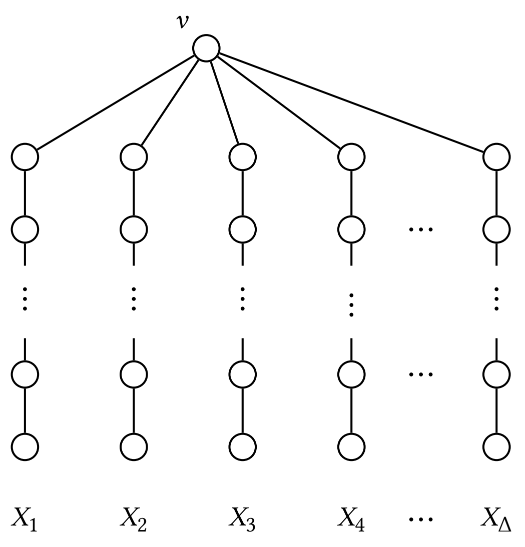


Figure 5.9: The graph G

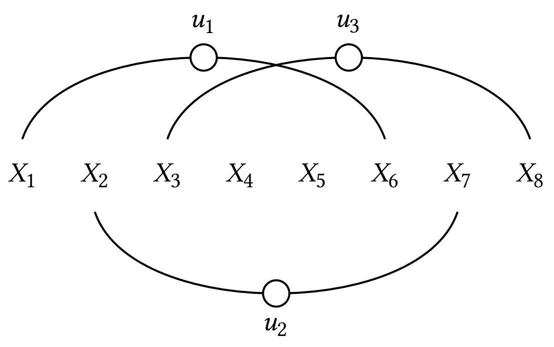


Figure 5.10: The new vertices added to the paths

Theorem 5.24. *For all $n \in \mathbb{N}$, there exists a strongly online graph G that is strongly connected, has pathwidth 2, max degree $n(n+1)$ and strongly online connected pathwidth greater than $^{n-1}/_2$.*

Proof. Fix $n \in \mathbb{N}$. We will construct in stages a graph $G = \lim_s G_s$ such that G has path pathwidth 2, max degree $n(n+1)$, and no strongly online path decomposition of width at most $^{n-1}/_2 = k$, where k is the allowed pathwidth.

Stage 1: Let G_1 be the star graph $S_{n(n+1)}$ and present the center vertex v.

Stages 2 through $2n(n+1)$: At each stage, add a new vertex onto each of the leaves, and present the next vertex in $N^2(v)$.

Stage $2n(n+1)+1$: Let $\mathcal{P}$ be the current path decomposition. Because all of the presented vertices are connected from this stage, the order of the bags that contain them cannot change. Note that in $\mathcal{P}$, all of the bags have size at most k and at least 2. Hence there must be at least $n+1$ bags and a corresponding $n+1$ vertices that belong to exactly one of these bags. Denote these bags $B_1, \dots, B_{n+1}$ and their corresponding vertices $u_1, \dots, u_{n+1}$, ordered by their appearance in $\mathscr{P}$;

Each u_i is contained in one path X_i. Present any vertex and reveal k vertices joining the paths X_i and $X_{n/2+i}$ for each $i < n/2$. This forms a series of k cycles, each one forcing another vertex in the bag $B_{n/2}$. Present the rest of the vertices and let $\mathscr{P}'$ be the final decomposition.

$\mathscr{P}'$ must have width at least $k+1$ because each cycle forced an increase in the width by 1. Therefore G has no path decomposition of width k. $\square$

The proof of Theorem 5.24 is modified from Theorem 5.23 but with max degree $n(n+1)$. Then when we join n bags from the star S_Δ we force one bag to have width at least $\lfloor {}^{n-1}/_2 \rfloor = \lfloor {}^{-3}/_4 + \sqrt{1+4\Delta} \rfloor$.

Observe that G is finite because once the center v and then $N(v)$ have been presented their order is fixed. Also note that this gives us a strongly online pathwidth of $O(\sqrt{\Delta})$. Thus we need a rather high max degree to ensure that there are n vertices each in a unique bag.

When considering general strongly online graphs, there exist graphs that do not have finite strongly online path decompositions.

Theorem 5.25.

1. *For each online algorithm* $\mathbf{A}$ *and finite number* n*, there is a strongly online (finite) graph* G *of pathwidth 2 which cannot have an online path decomposition of width* n*. We can construct* G *with at most* $O(n^4)$ *many vertices.*
2. *There is a (infinite) strongly online graph of pathwidth 2 which cannot have an online path decomposition.*

The basic idea is, for each possible σ that can rearrange the bags from stage to stage, present $n(2n+2)$ vertices that each form part of a single path. Then use σ to determine the order in which the bags containing each of the $n(2n+2)$ vertices will be ordered after we reveal all the vertices that join them. We join the paths (bags) across the middle bag n times. This forces the middle bag to have at least $n+1$ vertices in it, as σ has made the wrong choices.

Proof. Fix $n \in \mathbb{N}$ and $\mathbf{A}$ an online algorithm. Note that at each stage $\mathbf{A}$ gives us a path decomposition of G_s. Because we want to force a pathwidth greater than $n-1$ every bag has size at most n. We will force at least 1 bag to have size $n+1$. By using a pairing function we consider $\mathbf{A}$ as a way of extending a path decomposition as in Definition 5.20. We perform a computable construction.

We construct a strongly online graph $G = \lim_t G_t$.

We build G in two stages. In stage 1, we present $n(2n+1)$ vertices, each the center of a path that grows in length with every presented vertex. Once this is done, let $\mathscr{P}$ be path decomposition generated by $\mathbf{A}$ on the currently presented vertices, this signals the start of stage 2. See Figure 5.11.

In stage 2, there are two cases. First, if $\mathbf{A}(\mathscr{P})$ does not give us a valid decomposition or a valid extension of $\mathscr{P}$, then G satisfies our requirements and we stop the construction.

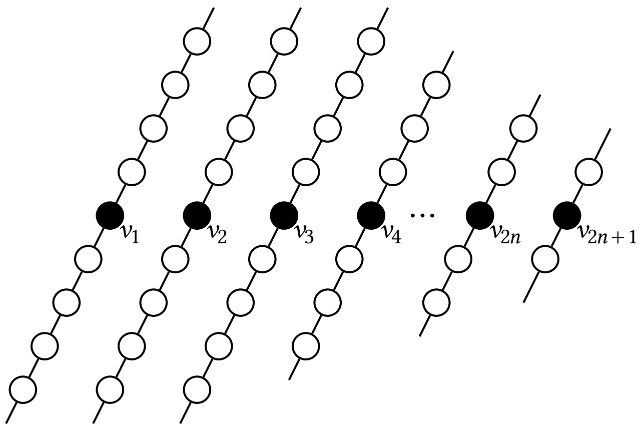


Figure 5.11: The initial $n(2n + 1)$ paths and the presented vertices (in black)

Second, $\mathbf{A}(\mathscr{P})$ is a valid decomposition. In which case, because $\mathbf{A}(\mathscr{P})$ has $n(2n+1)$ vertices divided into bags of size n, there must be $2n + 1$ bags each containing a vertex not in any of the other $2n + 1$ bags. Let these bags be $X_1, \dots, X_{2n+1}$, and the corresponding vertices $v_1, \dots, v_{2n+1}$. Without loss of generality, assume that $X_1, \dots, X_{2n+1}$ are ordered by their appearance in $\mathbf{A}(\mathscr{P})$.

Each X_i corresponds to a path in G. We reveal $2n$ vertices, each one joining the end of two paths in G_e as follows. For $i \leq n$ Join the i-th path to the $(2n + 2 - i)$-th and the $(2n + 1 - i)$-th. This forms a sort of spiral (see Figure 5.12).

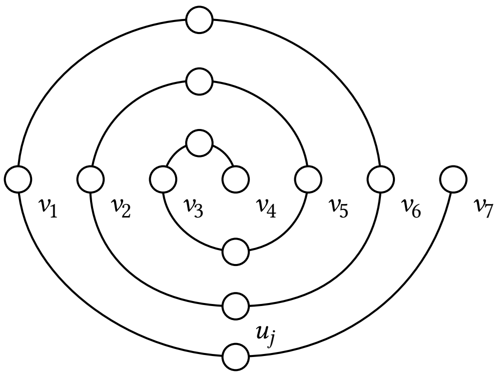


Figure 5.12: The connections between vertices (paths) for $n = 3$

We now stop revealing vertices in G and present all the vertices visible in the order they were revealed. This ends the construction of G.

We need to show that $\mathbf{A}$ does not give a valid path decomposition of G. Fix $i \leq n$. Let $u_1, u_2, \dots, u_a$ be the path v_i–v_{2n+2-i}. Let $\mathscr{Q}$ be the current path decomposition given by $\mathbf{A}$. Let $U_1, U_2, \dots, U_b$ be the bags in $\mathscr{Q}$ that contain a u_j, in the order they appear in $\mathscr{Q}$. Note that $u_1 = v_i$ and $u_a = v_{2n+2-i}$. Without loss of generality, we can assume that $U_1 = X_i$ and $U_b = U_{2n+2-i}$ (see Figure 5.13).

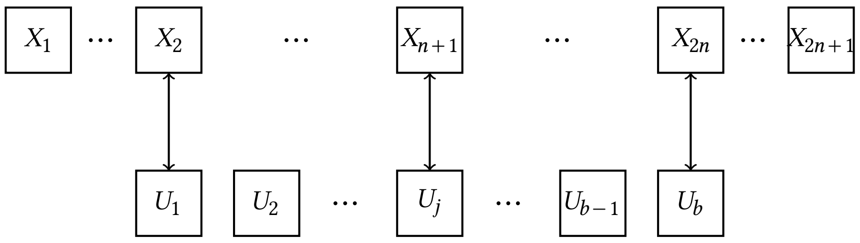


Figure 5.13: The bags in $\mathcal{Q}$ and the bags corresponding to the path v_i–v_{2n+2-i}

Thus X_{n+1} appears between U_1 and U_b. Hence there must be some U_j such that $U_j = X_{n+1}$. Therefore X_{n+1} contains some u_j. Then by letting i vary, we can see that X_{n+1} must contain $n + 1$ vertices (one from each path along with the vertex v_{n+1}). Therefore $\mathbf{A}(\mathcal{P})$ does not give a width $n - 1$ path decomposition of G.

For each i, the path v_i–v_{2n+1-i} forces a vertex into X_{n+1} just as above. However this may be the same vertex forced in by v_i–v_{2n+2-i}. Thus from the $2n$ paths only half are guaranteed to increase the size of X_{n+1}.

G is union of $n(2n + 1)$ paths. Because at each stage each path gets another two vertices the i-th path has $1 + 2(i - 1)$ vertices. Finally we add another $2n$ vertices to join these paths together. Therefore the total number of vertices is

$$\sum_{i=1}^{2n^2+n} \big(1 + 2(i - 1)\big) + 2n = n^2(1 + 2n)^2 + 2n = O(n^4)$$

Then by a diagonalization against all possible strategies and widths we can see that we have a strongly online graph with no strongly online path decomposition. □

Chapter 6

Conclusion

In this thesis, we explored a wide variety of parameterizations related to online and adversarial algorithms on graphs and related structures. We began with an exploration of the perfect code game, where we proved several results relating to graph shape. We introduced adversarial online colouring which is the intersection of adversarial and online colouring. For adversarial online colouring we proved results relating to, but not limited to, maximum degree, and trees. We then saw the chain decomposition game and how it is related to the colouring game in incomparability graphs. Finally, we explored a new algorithmic parameterization related to online algorithms, strongly online graphs. We saw how strongly online graphs relate to highly computable graphs and explored a wide variety of results from strongly online graphs of bounded pathwidth to strongly connected trees.

6.1 Other work

There is much work that we would have liked to have done, but due to time limitations did not get around to. For example, we would like to investigate online perfect codes and adversarial online versions of perfect codes and other structures. There is much work to be done in these areas as perfect codes have not been widely studied online and adversarial online is a new parameterization. We explain two more areas below.

Brookes Theorem on Strongly Online Graphs

Brookes theorem states that every graph G with max degree Δ is $(\Delta + 1)$-colourable if G is not an odd cycle nor contains a $k + 1$ clique [54]. By Tverberg 1984 [55] the highly computable version of Brookes theorem is also true. That is, every highly computable graph G with max degree Δ is computably $(\Delta + 1)$-colourable if G is not an odd cycle nor contains a $k + 1$ clique. We conjecture that this is not true for strongly online graphs.

Conjecture 6.1. *There is a strongly online graph G where G is not an odd cycle, has max degree Δ, and does not contain a $k + 1$ clique, such that G is not $(\Delta + 1)$-colourable.*

We would have liked to either proof of disprove this conjecture. However we were not able to to time restraints.

Strongly Online Partial Orders

A *strongly online partial order* is an online partial order $P^{\prec}$ where at every stage, you can see all the minimal (or maximal) neighbours of every vertex you have previously seen. A minimal (or maximal) neighbour of an element x is an element y such that $x \leq y$ (or $y \leq x$) and there is no z such that $x < z < y$ (or $x > z > y$). Strongly online partial orders mimic the behaviour of strongly online graphs. But, a strongly online version of P's comparability or incomparability graph would be different than the strongly online partial order P. Even if the vertices/elements are presented in the same order. This leads us to suspect that there could be interesting bounds related to strongly online partial orders. However, we did not have time to include this.

Bibliography

[1] A. M. Turing. “On Computable Numbers, with an Application to the Entscheidungsproblem”. In: *Proc. London Math. Soc. (2)* 42.3 (1936), pp. 230–265. DOI: 10.1112/plms/s2-42.1.230.

[2] M. R. Garey and D. S. Johnson. *Computers and intractability*. Vol. 174. freeman San Francisco, 1979.

[3] G. Dósa. “The Tight Bound of First Fit Decreasing Bin-Packing Algorithm Is FFD(I) $\leq$ 11/9OPT(I) + 6/9”. In: *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–11. DOI: 10.1007/978-3-540-74450-4_1.

[4] G. Dósa and J. Sgall. “First Fit bin packing: A tight analysis”. In: *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*. Vol. 20. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013, pp. 538–549. DOI: 10.4230/LIPIcs.STACS.2013.538.

[5] R. Thomas. “An update on the four-color theorem”. In: *Notices of the AMS* 45.7 (1998), pp. 848–859.

[6] D. R. Bean. “Effective Coloration”. In: *The Journal of Symbolic Logic* 41.2 (1976), pp. 469–480. DOI: 10.2307/2272247.

[7] X. Zhu. “Refined activation strategy for the marking game”. In: *J. Combin. Theory Ser. B* 98.1 (2008), pp. 1–18. DOI: 10.1016/j.jctb.2007.04.004.

[8] H. A. Kierstead and W. T. Trotter. “Planar graph coloring with an uncooperative partner”. In: *Journal of Graph Theory* 18.6 (1994), pp. 569–584.

[9] R. G. Downey, A. Melnikov and K. M. Ng. “Foundations of Online Structure Theory II: The Operator Approach”. In: *Logical Methods in Computer Science* Volume 17, Issue 3 (2021). DOI: 10.46298/lmcs-17(3:6)2021.

[10] H. A. Kierstead. “Recursive and on-line graph coloring”. In: *Handbook of recursive mathematics, Vol. 2*. Vol. 139. Stud. Logic Found. Math. North-Holland, Amsterdam, 1998, pp. 1233–1269. DOI: 10.1016/S0049-237X(98)80051-7.

[11] W. Gasarch. “A survey of recursive combinatorics”. In: *Handbook of recursive mathematics, Vol. 2*. Vol. 139. Stud. Logic Found. Math. North-Holland, Amsterdam, 1998, pp. 1041–1176. DOI: 10.1016/S0049-237X(98)80049-9.

[12] N. Robertson and P. D. Seymour. “Graph minors. V. Excluding a planar graph”. In: *Journal of Combinatorial Theory, Series B* 41.1 (1986), pp. 92–114.

[13] H. L. Bodlaender. “A partial k-arboretum of graphs with bounded treewidth”. In: *Theoret. Comput. Sci.* 209.1-2 (1998), pp. 1–45. DOI: 10.1016/S0304-3975(97)00228-4.

[14] S. Arnborg and A. Proskurowski. “Linear time algorithms for NP-hard problems restricted to partial k-trees”. In: *Discrete Applied Mathematics* 23.1 (1989), pp. 11–24. DOI: https://doi.org/10.1016/0166-218X(89)90031-0.

[15] Z. Bai, J. Tu and Y. Shi. “An improved algorithm for the vertex cover P_3 problem on graphs of bounded treewidth”. In: *Discrete Math. Theor. Comput. Sci.* 21.4 (2019), Paper No. 17, 13.

[16] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. 1st ed. Texts in Computer Science. Springer, London, 2013. DOI: 10.1007/978-1-4471-5559-1.

[17] H. L. Bodlaender. “A Linear Time Algorithm for Finding Tree-Decompositions of Small Treewidth”. In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*. STOC ’93. Association for Computing Machinery, 1993, pp. 226–234. DOI: 10.1145/167088.167161.

[18] A. Yamaguchi, K. Aoki and H. Mamitsuka. “Graph complexity of chemical compounds in biological pathways”. In: *Genome Informatics* 14 (Jan. 2003), pp. 376–377.

[19] J. Wu and X. Zhu. “Lower bounds for the game colouring number of partial k-trees and planar graphs”. In: *Discrete Math.* 308.12 (2008), pp. 2637–2642. DOI: 10.1016/j.disc.2007.05.023.

[20] H. A. Kierstead and W. T. Trotter. “An extremal problem in recursive combinatorics”. In: *Congressus Numerantium* 33.143-153 (1981), p. 98.

[21] U. Faigle et al. “On the game chromatic number of some classes of graphs”. In: *Ars Combin.* 35 (1993), pp. 143–150.

[22] H. A. Kierstead. “A Simple Competitive Graph Coloring Algorithm”. In: *Journal of Combinatorial Theory, Series B* 78.1 (2000), pp. 57–68. DOI: https://doi.org/10.1006/jctb.1999.1927.

[23] M. Livingston and Q. F. Stout. “Distributing Resources in Hypercube Computers”. In: *Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications: Architecture, Software, Computer Systems, and General Issues - Volume 1*. C3P. Pasadena, California, USA: Association for Computing Machinery, 1988, pp. 222–231. DOI: 10.1145/62297.62324.

[24] N. Biggs. “Perfect codes in graphs”. In: *J. Combinatorial Theory Ser. B* 15 (1973), pp. 289–296. DOI: 10.1016/0095-8956(73)90042-7.

[25] D. W. Bange, A. E. Barkauskas and P. J. Slater. “Efficient dominating sets in graphs”. In: *Applications of discrete mathematics (Clemson, SC, 1986)*. SIAM, Philadelphia, PA, 1988, pp. 189–199.

[26] G. J. Chang, C. Pandu Rangan and S. R. Coorg. “Weighted independent perfect domination on cocomparability graphs”. In: *Algorithms and computation (Hong Kong, 1993)*. Vol. 762. Lecture Notes in Comput. Sci. Springer, Berlin, 1993, pp. 506–514. DOI: 10.1007/3-540-57568-5_282.

[27] M. Livingston and Q. F. Stout. “Perfect dominating sets”. In: *Proceedings of the Twenty-first Southeastern Conference on Combinatorics, Graph Theory, and Computing (Boca Raton, FL, 1990)*. Vol. 79. 1990, pp. 187–203.

[28] R. Hamming. *Coding and Information Theory*. Prentice-Hall, 1986.

[29] D. W. Bange, A. E. Barkauskas and P. J. Slater. “Efficient dominating sets in graphs”. In: *Applications of discrete mathematics (Clemson, SC, 1986)*. SIAM, Philadelphia, PA, 1988, pp. 189–199.

[30] M. S. Chang and Y. C. Liu. “Polynomial algorithms for the weighted perfect domination problems on chordal graphs and split graphs”. In: *Information Processing Letters* 48.4 (1993), pp. 205–210. DOI: 10.1016/0020-0190(93)90147-2.

[31] A. Gyárfás and J. Lehel. “On-line and first fit colorings of graphs”. In: *Journal of Graph Theory* 12.2 (1988), pp. 217–227. DOI: https://doi.org/10.1002/jgt.3190120212.

[32] N. Alon and S. Gutner. “Linear Time Algorithms for Finding a Dominating Set of Fixed Size in Degenerated Graphs”. In: *Algorithmica* 54 (July 2009), pp. 544–556. DOI: 10.1007/s00453-008-9204-0.

[33] S. Irani. “Coloring inductive graphs on-line”. In: *Algorithmica* 11.1 (1994), pp. 53–72. DOI: 10.1007/BF01294263.

[34] B. Courcelle. “The monadic second-order logic of graphs. I. Recognizable sets of finite graphs”. In: *Information and Computation* 85.1 (1990), pp. 12–75. DOI: 10.1016/0890-5401(90)90043-H.

[35] P. Hall. “On representatives of subsets”. In: vol. s1-10. 1. 1935, pp. 26–30. DOI: =10.1112/jlms/s1-10.37.26.

[36] D. Kelly. “On the dimension of partially ordered sets”. In: *Discrete Mathematics* 35.1 (1981). Special Volume on Ordered Sets, pp. 135–156. DOI: 10.1016/0012-365X(81)90203-X.

[37] K. P. Bogart, C. Greene and J. P. S. Kung. “The Impact of the Chain Decomposition Theorem on Classical Combinatorics”. In: *The Dilworth Theorems: Selected Papers of Robert P. Dilworth*. Boston, MA: Birkhäuser Boston, 1990, pp. 19–29. DOI: 10.1007/978-1-4899-3558-8_3.

[38] H. A. Kierstead. “An effective version of Dilworth’s theorem”. In: *Transactions of the American Mathematical Society* 268.1 (1981), pp. 63–77. DOI: 10.2307/1998337.

[39] B. Bosek and T. Krawczyk. “On-line partitioning of width w posets into $w^{O(\log\log w)}$ chains”. In: *European J. Combin.* 91 (2021), p. 103202. DOI: 10.1016/j.ejc.2020.103202.

[40] B. Bosek and T. Krawczyk. “The sub-exponential upper bound for on-line chain partitioning”. In: *2010 IEEE 51st Annual Symposium on Foundations of Computer Science—FOCS 2010*. IEEE Computer Soc., Los Alamitos, CA, 2010, pp. 347–354. DOI: 10.1109/FOCS.2010.40.

[41] B. Bosek and P. Micek. “Variants of online chain partition problem of posets”. In: *Proceedings of the Second Workshop on Computational Logic and Applications (CLA 2004)*. Vol. 140. Electron. Notes Theor. Comput. Sci. Elsevier Sci. B. V., Amsterdam, 2005, pp. 3–13. DOI: 10.1016/j.entcs.2005.06.028.

[42] R. P. Dilworth. “A Decomposition Theorem for Partially Ordered Sets”. In: *Annals of Mathematics* 51.1 (1950), pp. 161–166. DOI: 10.2307/1969503.

[43] G. B. Dantzig and A. J. Hoffman. “11. Dilworth’s Theorem on Partially Ordered Sets”. In: *Linear Inequalities and Related Systems. (AM-38), Volume 38*. Princeton University Press, 1956, pp. 207–214. DOI: doi:10.1515/9781400881987-012.

[44] D. R. Fulkerson. “Note on Dilworth’s decomposition theorem for partially ordered sets”. In: *Proc. Amer. Math. Soc.* 7 (1956), pp. 701–702. DOI: 10.2307/2033375.

[45] F. Galvin. “A Proof of Dilworth’s Chain Decomposition Theorem”. In: *The American Mathematical Monthly* 101.4 (1994), pp. 352–353. DOI: 10.2307/2975628.

[46] T. Krawczyk and B. Walczak. “Asymmetric Coloring Games on Incomparability Graphs”. In: *Electronic Notes in Discrete Mathematics* 49 (Mar. 2015). DOI: 10.1016/j.endm.2015.06.108.

[47] P. C. Gilmore and A. J. Hoffman. “A Characterization of Comparability Graphs and of Interval Graphs”. In: *Canadian Journal of Mathematics* 16 (1964), pp. 539–548. DOI: 10.4153/CJM-1964-055-5.

[48] R. H. Möhring. “Algorithmic Aspects of Comparability Graphs and Interval Graphs”. In: *Graphs and Order: The Role of Graphs in the Theory of Ordered Sets and Its Applications*. Dordrecht: Springer Netherlands, 1985, pp. 41–101. DOI: 10.1007/978-94-009-5315-4_2.

[49] B. Bosek and T. Krawczyk. “A subexponential upper bound for the on-line chain partitioning problem”. In: *Combinatorica* 35.1 (2015), pp. 1–38. DOI: 10.1007/s00493-014-2908-7.

[50] B. Bosek et al. “An easy subexponential bound for online chain partitioning”. In: *Electron. J. Combin.* 25.2 (2018), Paper No. 2.28, 1–23. DOI: 10.37236/7231.

[51] H. A. Kierstead. “Recursive colorings of highly recursive graphs”. In: *Canadian Journal of Mathematics* 33.6 (1981), pp. 1279–1290. DOI: 10.4153/CJM-1981-097-8.

[52] J. H. Schmerl. “Recursive colorings of graphs”. In: *Canadian Journal of Mathematics* 32.4 (1980), pp. 821–830. DOI: 10.4153/CJM-1980-062-7.

[53] H. A. Kierstead and W. T. Trotter Jr. “An extremal problem in recursive combinatorics”. In: *Congr. Numer.* 33 (1981), pp. 143–153.

[54] R. L. Brooks. “On colouring the nodes of a network”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 37.2 (1941), pp. 194–197. DOI: `10.1017/S030500410002168X`.

[55] H. Tverberg. “On Schmerl’s effective version of Brooks’ theorem”. In: *J. Combin. Theory Ser. B* 37.1 (1984), pp. 27–30. DOI: `10.1016/0095-8956(84)90041-8`.