The Frequency Assignment Problem

A thesis submitted for the degree of
Doctor of Philosophy

by
Angela Erika Koller

Department of Mathematical Sciences
Brunel University
West London
2004

Dedicated to

Murmel

Dad

The Frequency Assignment Problem

Angela Erika Koller

Submitted for the degree of Doctor of Philosophy

2004

Abstract

This thesis examines a wide collection of frequency assignment problems.

One of the largest topics in this thesis is that of $L(2,1)$-labellings of outerplanar graphs. The main result in this topic is the fact that there exists a polynomial time algorithm to determine the minimum $L(2,1)$-span for an outerplanar graph. This result generalises the analogous result for trees, solves a stated open problem and complements the fact that the problem is NP-complete for planar graphs. We furthermore give best possible bounds on the minimum $L(2,1)$-span and the cyclic-$L(2,1)$-span in outerplanar graphs, when the maximum degree is at least eight.

We also give polynomial time algorithms for solving the standard constraint matrix problem for several classes of graphs, such as chains of triangles, the wheel and a larger class of graphs containing the wheel.

We furthermore introduce the concept of one-close-neighbour problems, which have some practical applications. We prove optimal results for bipartite graphs, odd cycles and complete multipartite graphs.

Finally we evaluate different algorithms for the frequency assignment problem, using domination analysis. We compute bounds for the domination number of some heuristics for both the fixed spectrum version of the frequency assignment problem and the minimum span frequency assignment problem. Our results show that the standard greedy algorithm does not perform well, compared to some slightly more advanced algorithms, which is what we would expect.

In this thesis we furthermore give some background and motivation for the topics being investigated, as well as mentioning several open problems.

Contents

Acknowledgements

This research was partially funded by the EPSRC.

First of all I would like to say thanks to my supervisor, Steve Noble. I am very grateful that he took me on as a PhD student, that he got me hooked on graph theory, especially the frequency assignment problem and all its variations, and that he always made time to answer my questions and helped me out when I got stuck. Also I'm very thankful for his encouragment to attend workshops and conferences, especially the BCC in 2001 for various reasons.

I also would like to thank everybody within the Department of Mathematical Sciences, for creating such a friendly atmosphere which made it so enjoyable to study. A special thanks goes to Vinnie, Asfia, Idil, Anja and Tilo, Rebekah, Beverly, Anne, Alexei, Claire and Nobby, Marianna and Marion.

Thanks to M. Imran who made this template available on the internet.

My former lecturer, Prof. Dr. Suess-Gebhard, encouraged a number of students to go abroad for further studies. I was one of them and I would like to thank her for that.

I am ever so thankful to my parents who put the idea of starting a PhD in my head and supported me in every possible way. A big thank you also goes to my good friends back home who were willing to spend lots of money on phone bills and the occasional flight tickets to maintain the friendship: thanks Alex, Muecke, Flutschi, Elke, Sabine and Claudia!

A very special thank you goes to Anders. He has been very patient with me especially in the past few months, and has supported, encouraged and inspired me throughout the time. Thanks BB!

Chapter 1

Introduction

This thesis studies a collection of problems based on variations of the frequency assignment problem.

The radio spectrum is a limited resource which is required for both new and expanding radio services. This makes the allocation of spectrum increasingly difficult and important. Although different radio services differ significantly in their specifications, most services have a common requirement that discrete radio frequencies are assigned to users. The frequency assignment problem is concerned with assigning frequencies in as efficient a way as possible, while guaranteeing a specified level of service. Given the world-wide demand for radio frequencies for voice, as well as data communication, there has been tremendous interest in this problem over the last few years.

In a model for mobile radio services the infrastructure is fixed and the area being served is divided into a number of cells each of which has its own transmitter, the location of which is predetermined. The spectrum is divided into a number of discrete frequencies. Each transmitter is allocated a number of frequencies in order to serve the users in its cell. If closely situated transmitters operate on similar frequencies, the users in the corresponding cells may not receive a signal of sufficient quality. Therefore the frequencies on which neighbouring transmitters can operate are constrained. The objective is to minimise the amount of resource used in an assignment. This may be the number of different frequencies but more often it is

the span of the assignment, that is the difference between the highest and lowest frequencies used.

Of the alternative mathematical formulations of these problems, the most natural one uses graph theory with vertices representing transmitters and edges joining pairs of transmitters which are constrained. Weights on the edges give the required separation of frequencies assigned to the corresponding transmitters and weights on the vertices give the demand for frequencies at each transmitter. Thus the problems become generalisations of the well-known graph colouring problem and many results from this area have been used in frequency assignment. It is straightforward to show that even in very restricted cases, to solve the problems exactly is computationally intractable.

We begin by giving some notation and definitions on graphs and graph colouring. Since some of our results are concerned with complexity we also give a brief introduction in Chapter 2. We then move on to give some background knowledge and definitions of the frequency assignment problem. There are two main types of frequency assignment problems, namely the fixed spectrum version and the minimum span frequency assignment problem. We mainly study the minimum span, except in Chapter 6 where we also give results for the fixed spectrum version.

In Chapter 3 we study the most general frequency assignment problems, the constraint matrix problems. First we give a definition and then spend some time on known results to give a flavour of the variety and complexity of these problems. For instance a great deal of work has been done investigating the complexity of constraint matrix problems on graphs of bounded treewidth. It is known that for any fixed $k \geq 3$ computing the minimum span for a constraint matrix problem based on graphs of treewidth at most k is an NP-hard problem. The complexity when $k = 2$ is open. We make some progress towards resolving this by solving the minimum span problem for constraint matrix problems on different types of underlying graphs using a dynamic programming approach.

In Chapter 4 we introduce the cyclic channel metric and again start off by giving definitions and introducing some known results. In section 4.3 we modify a known

result using the cyclic channel metric and therefore give an upper bound on the span of a constraint matrix problem. The last section introduces the one-close neighbour problem which attempts to resolve one of the inaccuracies of our model of interference and has an attractive formulation. We give bounds for the one-close-neighbour problem on different types of underlying graphs, such as the odd cycle and the complete graph.

The next chapter, Chapter 5, studies a variation of the frequency assignment problem, the $L(2,1)$-labelling. The main idea is that transmitters that are close to each other should not receive the same frequency and transmitters that are "very" close to each other should have frequencies at least two apart. The $L(2,1)$-labelling was first discussed in 1988 by Roberts and Griggs in [20] and is one of the most studied mathematical problem motivated by frequency assignment. We describe the polynomial time algorithm, developed by Chang and Kuo [14] that finds an optimal $L(2,1)$-labelling of trees. This is important for us as the techniques used in the proof as well as the result itself forms the basis for the main result in this chapter. Since this result first appeared, a great deal of effort has been spent in trying to determine whether a similar result can be achieved for either graphs of bounded treewidth or series-parallel networks (graphs with tree-width at most two). The motivation of our main result stems from this and also a paper by Calamoneri and Petreschi [13], where they consider a subclass of series-parallel networks, namely outerplanar graphs. The authors give an upper bound for the optimal $L(2,1)$-span when G is outerplanar. In Section 5.4 we state their results, illustrate why there are some problems with their bound and give an analogous result using the cyclic channel metric. In the last section we introduce a polynomial time algorithm to find both the cyclic and non-cyclic $L(2,1)$-span for outerplanar graphs. This section is joint work with Steve Noble and Anders Yeo.

Chapter 6 focuses on a different aspect of graph theory. We introduce the domination number of an algorithm, which is a measure of the quality of a heuristic. Domination analysis was first studied in context with the travelling salesman problem (see [19]) and we adapted these ideas and implemented them for both the fixed

spectrum and minimum span versions of the frequency assignment problem. We begin with a survey of results on domination analysis before we then give bounds for the domination number of two greedy heuristics for the fixed spectrum version. These results are based on [35]. The final section in this chapter introduces a greedy algorithm that essentially produces a solution to the minimum span frequency assignment problem by producing an ordering of the vertices. We then give an upper bound on the domination number of this algorithm.

In Chapter 7 we give a brief conclusion and restate some open problems.

As an appendix we present a table of notation used throughout this thesis.

Chapter 2

Graphs, Graph Colouring, Complexity and Frequency Assignment Problem

In this chapter we will mainly give definitions and notation that are used throughout this thesis. The first section covers some basic graph theoretical notation, basic results for graph colouring and we will introduce the cyclic channel metric. In the next section we will give some motivation for the frequency assignment problem and how it is connected to graph theory.

2.1 Graphs and graph colouring

We assume that the reader has a knowledge of basic graph theory and refer the reader to [5], [53] or [54]. All of our problems involve simple, connected graphs G with no loops or multiple edges. We use the standard notation $G = (V, E)$, where V denotes the set of vertices and E denotes the set of edges. We will usually denote $|V|$ by n. Sometimes we use the notation $V(G)$ $(E(G))$ when we talk about the vertex set (edge set) of a graph G.

The set of vertices that are adjacent to vertex v is called the *neighbourhood* of v and is denoted by $N(v)$. The *closed neighborhood* of v is denoted by $N[v]$ and is

given by $N[v] = N(v) \cup \{v\}$.

A *digraph* or directed graph D consists of a vertex set $V(D)$ and an arc set $A(D)$, where $A(D)$ is a finite set of ordered pairs of elements of $V(D)$. If $(v, w) \in A(D)$, then we say that there is an arc from v to w and also denote this by $v \to w$.

Given an undirected graph G, an *acyclic orientation* of G is a digraph formed from G by replacing each undirected edge $\{v, w\}$ by one of the two arcs (v, w) or (w, v) in such a way that no directed cycles are formed. We will use the notation (G, ω) to denote the digraph formed from G by applying the acyclic orientation ω. When G is a complete graph then we may think of ω as being a total ordering on the vertices of G.

The *length of a path* for an orientation of a weighted graph is the sum of all weights of edges that are lying on that directed path.

A *subgraph* $H = (V', E')$ of a graph $G = (V, E)$ is a graph with $V' \subseteq V$ and $E' \subseteq E$. H is a proper subgraph of G if $E' \neq E$. A *subdigraph* is defined similarly.

The *degree* of a vertex v in a graph G is denoted by $d_G(v)$ or just $d(v)$. The maximum and minimum degrees of a graph G are denoted by $\Delta(G)$ and $\delta(G)$ respectively.

A graph is planar if it can be drawn in a plane without edges crossing.

A graph G is called *outerplanar* if there exists a plane drawing of G where all the vertices lie on the outermost face.

A graph G is *l-partite* if the vertex set $V(G)$ may be partitioned into disjoint subsets $A_1(G), A_2(G), \ldots, A_l(G)$ and for each edge e the endpoints of e lie in different subsets. The sets $A_1(G), A_2(G), \ldots, A_l(G)$ are called the *partite* sets of G.

An l-partite graph G, with partite sets $A_1, A_2, \ldots, A_l$ is a *complete multipartite graph* if for every i, j with $1 \leq i < j \leq l$, each vertex in the partite set A_i is joined to each vertex in the set A_j by exactly one edge. A *bipartite* graph is a 2-partite graph. A *k-nearly bipartite graph* is a graph G from which it is possible to obtain a bipartite graph by deleting at most k vertices and their incident edges.

We denote by $\lceil x \rceil$ the smallest integer q such that $q \geq x$, and by $\lfloor x \rfloor$ the largest integer q such that $q \leq x$.

Given a set E' of edges, $G\backslash E'$ denotes the graph formed from G by *deleting* all the edges in E'. The *contraction* of a set E' of edges is of G is denoted by G/E'. H is a *minor* of G if it can be obtained from G by deleting and contracting edges and deleting vertices. A graph $G : A$ is said to be *induced* by A, a set of vertices, if the graph has vertex set A and edge set consisting of those edges of G with both endpoints in A.

Let $G = (V, E)$ be a graph. A vertex colouring of G is a mapping $f : V \to \{0, 1, \ldots, k-1\}$ such that if v and w are adjacent then $f(v) \neq f(w)$. The *chromatic number* $\chi(G)$ of G, denotes the smallest possible value of k for which there exists a vertex colouring.

2.2 Complexity

We will only give a brief explanation of the main definitions on complexity. We refer the reader to [18] or [43] for more information.

Let $f : \mathbb{Z}^+ \to \mathbb{Z}^+$. Then the following notation is defined .

1. The set of all functions which are eventually dominated by a constant multiple of f is called $O(f(n))$. More formally,

$$O(f(n)) = \{g : \mathbb{Z}^+ \to \mathbb{Z} | \ \exists c, N, \ \forall i \geq N, \ g(i) \leq cf(i)\}.$$

2. The set of all functions which eventually dominate a positive constant multiple of f is called $\Omega(f(n))$. More formally,

$$\Omega(f(n)) = \{g : \mathbb{Z}^+ \to \mathbb{Z} | \ \exists N, c > 0, \ \forall i \geq N, \ g(i) \geq cf(i)\}.$$

3. We say that a function $g(n)$ has order $f(n)$ if it dominates and is dominated by $f(n)$. The set of functions of order $f(n)$ is called $\Theta(f(n))$. More formally,

$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n)).$$

We say that an algorithm $\mathcal{A}$ runs in polynomial time if there is a polynomial p such that for any input of size n the number of operations of $\mathcal{A}$ is at most $p(n)$.

Even though $O(f(n))$ stricly speaking is a set of functions, we will also use it to denote an order (for example g is at most $O(f(n))$, or $g \in O(f(n))$, means that g is bounded from above by a constant times $f(n)$)).

Formal complexity theory is largely concerned with decision problems. To describe these, we first need a couple of preliminary defintions. An *alphabet* is a finite set of symbols. The set of all finite strings over an alphabet Σ is denoted by Σ^*. A *language* over an alphabet Σ is a subset of Σ^*. A problem is a *decision problem* if it is equivalent to determining whether an input x is a member of a language L. For a discussion of what is meant by equivalent see [18].

The complexity class NP is defined as follows. A language L over alphabet Σ belongs to the complexity class NP if there is a constant c and a binary relation $\mathcal{A}$ computable in polynomial time such that

$$L = \{x \in \Sigma^* : \text{ there exists y with } |y| \in O(|x|^c) \text{ such that } \mathcal{A}(x, y)\}.$$

We abuse our notation slightly by saying that decision problems corresponding to languages in P (NP) are members of P (NP).

Informally a decision problem is a member of NP if and only if there is an algorithm that will verify that the answer is YES, if it is so, in polynomial time, but will never say YES when the answer should be NO.

An important concept is that of reduction between two languages. If L_1, L_2 are languages over alphabets Σ_1, Σ_2 we say that L_1 is *polynomially reducible* to L_2, which we denote by $L_1 \leq L_2$, if there exists a polynomial time computable function $f : \Sigma_1^* \to \Sigma_2^*$ such that

$$x \in L_1 \Leftrightarrow f(x) \in L_2.$$

The hardest problems in NP are the *NP-complete* problems. A language L is NP complete if

1. $L \in NP$ and

2. For every $L' \in NP$, $L' \leq L$.

A similar notion of hardness exists for problems other than decision problems. To explain this we first need to define an apparently weaker form of reduction.

We say that a problem π_1 is Turing reducible to a problem π_2, denoted by $\pi_1 \leq_T \pi_2$ if there is an algorithm to solve π_1 that runs in polynomial time providing it has access to an oracle for π_2, that is a subroutine that will solve an instance of π_2 in unit time.

A problem π_1 is said to be *NP-hard* if for all $L \in NP$, we have $L \leq_T \pi_1$. Any NP-complete problem is NP-hard.

2.3 Frequency assignment problem

The frequency assignment problem has been well studied in many publications for example [12,17,34,36,42]. See [1,39] for good and comprehensive surveys.

Wireless communication plays an important role both in civil and military applications. In order to establish a connection, a transmitter and a corresponding receiver have to be tuned (assigned) to the same frequency. The frequency assignment problem therefore deals with the tuning of several wireless connections. Naturally, depending on the location of the sender and receiver and the frequency they are tuned to, interference is likely to occur. Since the spectrum of frequencies is a limited resource, it has become important to assign the frequencies in an optimum or near-optimum manner, that is use as small a range of frequencies as possible in such a way that the interference is kept as small as possible.

In practise, there are two main types of frequency assignment problems that occur: the minimum span frequency assignment and the fixed spectrum frequency assignment (also sometimes called the minimum interference problem).

We mainly study variations of the minimum span frequency assignment problem. The transmitters are assigned with frequencies out of a range of discrete frequencies $\{0, 1, ..., \sigma - 1\}$. The value σ is known as the *span*.

In order to model interference we apply constraints to the frequencies assigned to sets of transmitters. We will only consider constraints applied to pairs of trans-

mitters. All these constraints are represented in a constraint matrix $C = (c_{ij})$ where for all i, j the frequencies f_i and f_j assigned to transmitters i, j respectively must satisfy $|f_i - f_j| \geq c_{ij}$.

The simplest interference model is based solely on the assumptions that the constraint between two transmitters depends on the geographical distance between them and that the constraints are monotonically decreasing as distance increases. More specifically, there exist constants $t_1, \ldots, t_k$ with $t_1 \geq t_2 \geq \ldots \geq t_k$ such that $|f_i - f_j| \geq l$ if $d_{ij} < t_l$ where $l = 1, \ldots, k$ and d_{ij} is the distance between transmitters i and j. Defining the constraints in that way are called *distance-based constraints.* We can also define the constraints in a slightly different way, to give *frequency-based constraints.* Here we have a set of integers $k_1, \ldots, k_m$ with $k_1 \geq k_2 \geq \ldots \geq k_m$ such that $c_{ij} = k_{d_{ij}}$. Note that for any set of distance-based constraints there is an equivalent set of frequency-based constraints. In this thesis we mainly study the distance-based constraints.

The minimum span FAP is finding the minimum value of σ for which there exists a frequency assignment in which no interference occurs at all.

In Section 6.3 we study the second model, the fixed spectrum problem. Here the transmitters are assigned with frequencies out of a fixed range of σ channels, 0, 1, ..., σ-1.

Due to the given fixed span of frequencies (which is usually not big enough) it is almost always the case that in any assignment some interference does occur.

For various reasons there can be the situation where a particular constraint between say transmitters i and j must not be broken. In order to take this matter into account, weight w_{ij} is put on constraint c_{ij}. The weights are intended to reflect the importance of the constraints. Often all constraints are equally important and all weights are equal to one.

Depending on whether weights are applied to constraints or not, the objective is to minimize the number of constraints broken or to minimize the sum of weights of constraints broken, respectively. Given an assignment f we denote its cost $c(f)$ as the sum of weights of constraints broken.

The fixed spectrum version of the problem is arguably the more important because, in practice, regulators assign blocks of channels to particular operators or companies and later the actual assignments are made using frequencies from the given blocks.

Both the problems are modelled using a graph with a vertex for each transmitter and an edge between pairs of transmitters that are constrained. From now on we will use a mixture of graph theoretic and radio frequency terminology depending on which seems more appropriate at the time.

Chapter 3

Constraint Matrix Problems

Constraint matrix problems are the most general minimum span frequency assignment problems in which we consider only interference between pairs of transmitters.

We are given a matrix C with non-negative integers c_{ij} which denote the minimum channel separation between the frequencies assigned to transmitters i and j.

To express the problem in graph theoretic terminology, we construct an edge-weighted graph $G = (V, E)$ where each vertex v represents a transmitter and an edge between two vertices represents potential interference. The weight c_{ij} on the edge between v_i and v_j is the necessary channel separation in order to avoid interference. Sometimes it is convenient to take G to be the complete graph K_n, in which case pairs of transmitters where there is no constraint are joined by an edge with weight 0. At other times G contains precisely those edges corresponding to pairs of transmitters for which the frequencies to be assigned are constrained. In this case we refer to G as the *underlying graph* of the constraint matrix problem.

A *frequency assignment with span* σ is a function $f : V \to \{0, 1, \ldots, \sigma - 1\}$. A frequency assignment is feasible; if for all i, j, $|f(v_i) - f(v_j)| \geq c_{ij}$.

The *minimum span* of a constraint matrix problem is the minimum value of σ for which there exists a feasible frequency assignment with span σ.

If all the constraints are zero or one, then the minimum span is equal to the chromatic number of the underlying graph. This shows that the minimum span

problem is NP-hard. However we shall see that the addition of weights other than one makes the constraint matrix problem considerably more difficult than finding the chromatic number.

In this chapter we will first introduce some known results for the constraint matrix problems and then use a connection between constraint matrix problems and acyclic orientations of a graph to find the minimum span for constraint matrix problems corresponding to various classes of graph.

3.1 General results

Determining whether for a given constant $\sigma \geq 3$ the minimum span of a constraint matrix problem is at most σ is NP-complete.

To see this recall that the minimum span of a constraint matrix problem in which all the weights on edges are equal to zero or one corresponds to the chromatic number of the underlying graph. For any $c \geq 3$ determining whether the chromatic number of a graph is at most c is well-known to be NP-complete.

In [41] the authors give some results on the minimum span for the constraint matrix problem for different types of underlying graph. We abuse our notation somewhat by referring to the span of a graph G where we really mean the span of a constraint matrix problem with underlying graph G.

Denote the maximum constraint that occurs in G by c_{max} and

$$l_{min} = \min \{c_{ij} + c_{jk} | \{v_i, v_j\}, \{v_j, v_k\} \in E(G)\}.$$

Just as bipartite graphs are, in some sense, the simplest graphs to colour, it is also easy to compute the minimum span for the constraint matrix problem of bipartite graphs.

Theorem 3.1.1 The minimum span for a bipartite graph G is $c_{max} + 1$.

Proof: By assigning every vertex of one partite set with $f(u) = 0$ and every vertex of the second partite set with $f(v) = c_{max}$, we get a feasible assignment with span $c_{max} + 1$. □

Recall, a 1-nearly bipartite graph is a graph G for which there exists a vertex v such that $G - v$ is bipartite. Colouring a 1-nearly bipartite graph is easy because it is clear that at most three colours suffice. In view of this, it is perhaps surprising, that the constraint matrix problem becomes hard for these graphs.

Theorem 3.1.2 [41] Given a constraint matrix problem for a 1-nearly bipartite graph G and an integer m, deciding whether the minimum span of G is at most m is NP-complete.

It is easy to find the minimum span for odd cycles.

Theorem 3.1.3 The minimum span for an odd cycle C_{2n+1} is $\max\{c_{max}, l_{min}\} + 1$.

One well-studied class of graphs for which many NP-hard problems become solvable in polynomial and often linear time, is the class of graphs with bounded treewidth. These were introduced by Robertson and Seymour in their work on graph minors, see [47] and [48]. A good survey is [9].

Definition 3.1.4 [10] A *tree decomposition* of a graph $G = (V, E)$ is a pair (X, T) with $T = (I, F)$ a tree, and $X = \{X_i | i \in I\}$ a family of subsets of V, one for each node of T, such that

- $\bigcup_{i \in I} X_i = V$;
- for all edges $\{v, w\} \in E$ there exists an $i \in I$ with $v \in X_i$ and $w \in X_i$;
- for all $i, j, k \in I$: if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The *width* of a tree decomposition $((I, F), \{X_i | i \in I\})$ is $\max_{i \in I} |X_i| - 1$.

Definition 3.1.5 The *treewidth* of a graph G is the minimum width over all tree decompositions of G.

Given a graph G and an integer k, deciding whether the treewidth of G is at most k is NP-complete [4], however for fixed k, recognising graphs with treewidth at most k can be solved in linear time [8].

Furthermore for any fixed k the chromatic number of a graph with treewidth at most k can be found in polynomial time, see [3]. A similar result for the constraint matrix problem is unlikely to be true. Before we make this statement more precise we first give a careful definition of the problem.

Problem 3.1.6 Constraint matrix for treewidth k (CM_k)

Input: A constraint matrix with the underlying graph having treewidth at most k, integer m.

Question: Is the span at most m?

The following theorem is from [41].

Theorem 3.1.7 For all $k \geq 3$, CM_k is NP-complete.

Consequently, unless $P = NP$, there is no polynomial time algorithm for the constraint matrix problem for graphs of bounded treewidth at least three.

Graphs of treewidth one are just trees so using Theorem 3.1.1, CM_1 is in P. The complexity of CM_2 is open. We take a small step to resolve this question in Section 3.2 where we consider graphs that we call chains of triangles.

One natural question worth considering is whether knowing the chromatic number of a graph and a corresponding colouring achieving the chromatic number helps in solving the constraint matrix problem. More precisely we wish to consider the complexity of the following problem for each k.

Given an instance of a constraint matrix problem as a graph G with chromatic number equal to k, a k-colouring of G and an integer m, determine whether $\sigma \leq m$.

We study this problem by considering graphs of bounded treewidth and by making use of the following problem.

Problem 3.1.8 Constraint matrix for treewidth k with colouring (CMC_k)

Input: A constraint matrix with the underlying graph having treewidth equal to k, integer m and a $(k+1)$-colouring of G.

Question: Is the span at most m?

To study this problem we first consider the chromatic number of graphs of treewidth k. We will show that a graph with treewidth k has chromatic number at most $k+1$ but first we need a definition and two more lemmas.

Definition 3.1.9 If $\chi(G) = k$ but $\chi(H) < k$ for every proper subgraph H of G, then G is *k-critical.*

Lemma 3.1.10 If H is connected and k-critical, then $\delta(H) \geq k-1$.

Proof: Suppose for contradiction that $\delta(H) < k-1$. Let v be a vertex of degree $\delta(H)$ and let u be a neighbour of v. Since H is k-critical $\chi(H - \{u,v\}) = k-1$. Hence the graph formed from H by deleting v is $k-1$ colourable. Since $d(v) < k-1$ we see that H is $k-1$ colourable giving a contradiction. □

Lemma 3.1.11 [51] If G is connected then $\chi(G) \leq \max_H \delta(H) + 1$ where H runs over all connected subgraphs of G.

Proof: Let $k = \chi(G)$ and let H' be a connected k-critical subgraph of G. Clearly H' exists since if we keep successively deleting edges of G in such a way that the chromatic number does not decrease and we never disconnect the graph except to create isolated vertices then eventually we must reach a k-critical graph. Then by Lemma 3.1.10, $\chi(G) - 1 \leq \delta(H') \leq \max_H \delta(H)$, where H runs over all connected subgraphs of G. □

Theorem 3.1.12 If a graph G has treewidth k, then $\chi(G) \leq k+1$.

Proof: Suppose G has treewidth k. We may assume that G is connected. By Lemma 3.1.11, it suffices to show that $\max_H \delta(H) \leq k$ where H runs over all connected subgraphs of G. We prove this by induction on $|V|$.

Consider a tree decomposition (X,T) of G with treewidth k.

Let l be a leaf of T and let l' be its neighbour in T. If $X_l \subseteq X_{l'}$ then we may delete l from T and remove X_l from X, and still have a tree decomposition of G. Therefore we may assume that we have a tree decomposition with a leaf l having neighbour l' such that $X_l \backslash X_{l'} \neq \varnothing$.

Let $v \in X_l \backslash X_{l'}$ then v is not contained in any other node X_i, otherwise the third property of the Definition 3.1.4 would be violated. Recall that the width of a tree decomposition $((I,F),\{X_i|i \in I\})$ is $\max_{i\in I}|X_i|-1$. This gives us that $|X_l| \leq k+1$, and so v has degree at most k in any subgraph H of G. Hence any connected subgraph H of G either contains v giving $\delta(H) \leq k$ or is a subgraph of $G-v$ in which case the result follows by induction. □

We use these results to prove the following.

Theorem 3.1.13 For $k \geq 3$, CMC_k is NP-complete.

Proof: Clearly CMC_k is a member of NP. We show it is NP-complete by making a reduction from CM_k.

Given an instance (G, m_0) of CM_k where G is a weighted graph with treewidth at most k and $m_0 \geq 2$, we form an instance of CMC_k. We form H from G by adding a disjoint copy of K_{k+1}. An edge $\{v_i, v_j\}$ of H has weight $(k+2)c_{ij}$ if $\{v_i, v_j\}$ is an edge of G. Otherwise if $\{v_i, v_j\}$ is an edge in the copy of K_{k+1} it receives weight 1.

By Theorem 3.1.12 we know that $\chi(H) \leq k+1$. Furthermore a $k+1$ colouring of H can be constructed in linear time since H has treewidth at most k. Note that $\sigma(G) \leq m_0$ if and only if $\sigma(H) \leq (k+2)(m_0-1)+1$. Since the reduction can be carried out in linear time, we have shown that CMC_k is NP-complete for $k \geq 3$. □

We are now ready to prove that our original problem is NP-complete.

Theorem 3.1.14 Fix $k > 2$. Given a constraint matrix problem for a graph G with chromatic number k, a k-colouring of G and an integer m, determining whether $\sigma(G) \leq m$ is NP-complete.

Proof: Clearly the problem is a member of NP. For $k \geq 4$, CMC_{k-1} is a special case so the result follows from Theorem 3.1.13.

If $k = 3$ then the result follows from 3.1.2. □

3.2 Solving special cases of the minimum span problem

Our approach to these problems will make use of the following theorem [29].

Theorem 3.2.1 (Based on the theorem of Gallai, Roy and Vitaver [54]) The *minimum span* of a constraint matrix problem with underlying graph G is the minimum over all acyclic orientations ω of the length of the longest directed path in (G, ω).

Proof: If f is a feasible frequency assignment with span σ for G then we may build an acyclic orientation ω of G as follows. If $f(v_i) \leq f(v_j)$ then orient the edges $\{v_i, v_j\}$ from v_i to v_j and otherwise orient the edges from v_j to v_i. It is easy to see that σ is at least the length of the longest directed path in (G, ω). On the other hand let ω be the acyclic orientation of G minimising the length l of the longest directed path in (G, ω). Construct a frequency assignment with span l by setting $f(v_i)$ equal to the length of the longest directed path in (G, ω) into v_i. If $(v_i v_j)$ is an arc of (G, ω) then $f(v_j) \geq f(v_i) + c_{ij}$ and so f is feasible. Hence the result follows.

□

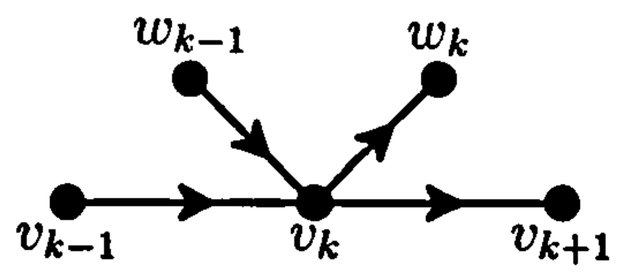
This theorem means that rather than looking for assignments we can restrict ourselves to looking at acyclic orientations. We will describe an algorithm that finds the acyclic orientation ω minimising the longest path in (G, ω) for a number of types of graph.

A *chain T of triangles* is a graph G with vertex set $V = \{v_1, \ldots, v_T, w_1, \ldots, w_{T-1}\}$ such that $v_1, \ldots, v_T$ form a path and for each i, w_i is adjacent to v_i and v_{i+1}.

We now show that it is possible to find the acyclic orientation with the shortest longest path and thus solve the minimum span problem on G.

If ω is an acyclic orientation of a chain of triangles with a vertex v_k as in Figure 3.1 we say that v_k is a *through* vertex of ω. More precisely, a vertex v_k is a through vertex if the edges from v_{k-1} and w_{k-1} are both oriented into (out of) v_k and the edges to w_{k+1} and v_{k+1} at the same time are oriented out of (into) v_k.

Figure 3.1: Through vertex v_k



A key observation is that in order to minimise the length of the longest path over all acyclic orientations, it suffices to consider only those where there are no through vertices. This follows by noting that if we have an acyclic orientation with a through vertex v_k as in Figure 3.1 then we may reverse the orientation on all the edges with both endpoints in $v_1, \ldots, v_k, w_1, \ldots, w_{k-1}$ and obtain an acyclic orientation where the longest path has not increased.

To solve the problem, we use dynamic programming. Typically if we are solving a problem on a graph H using dynamic programming we construct a sequence of graphs

$$H_1, \ldots, H_M = H.$$

For each graph in turn we construct a list of states or configurations and then calculate the optimal solution for that graph subject to some additional constraints specified by the configuration. In order to have a polynomial time algorithm the number of graphs in the sequence must be bounded by a polynomial in n and similarly the number of configurations for each graph must be bounded by a polynomial in n. Furthermore we must be able to compute the optimal solution for each configuration from the optimal solutions for the previously computed configurations in polynomial time. Finally we must be able to compute the optimal solution to the problem in polynomial time from the last set of configurations.

We now apply this to the constraint matrix problem where the underlying graph consists of chains of triangles.

Let G be a chain of T triangles. Let $G_N = G : \{v_1, \ldots, v_N, w_1, \ldots, w_{N-1}\}$. Let A_N be the set of acyclic orientations of G_N, with no through vertex. Let lp_ω denote the length of the longest directed path in an acyclic orientation ω.

A configuration of G_N consists of two paths, a directed path into v_N, a directed path out of v_N and the orientation of the edges $\{v_{N-1}, v_N\}$ and $\{w_{N-1}, v_N\}$. As we are looking at an acyclic orientation the two paths will be disjoint apart from the vertex v_N.

We say that an acyclic orientation ω *respects* a configuration τ if the orientation of $\{v_{N-1}, v_N\}$ and $\{w_{N-1}, v_N\}$ in ω are as specified by τ and the two directed paths that are part of τ are the longest directed paths into and out of v_N in ω. If ω respects τ we write $\tau \leq \omega$. The cost of τ is given by

$$\text{cost}(\tau) = \min_{\omega \in A_n : \tau \leq \omega} lp_\omega.$$

At each stage we record the list of all configurations, together with their cost and an acyclic orientation of G_N achieving the minimum cost of τ.

Consider the longest path P out of v_N. Let v_i be the vertex such that P passes through v_i but not through v_{i-1}. Suppose first that $i < N$. We now count how many possibilities there are for P. Clearly if P does not pass through any of $w_i, \ldots, w_{N-1}$ then there is only one possibility for the part of P up to v_i, namely the path $v_N, v_{N-1}, \ldots, v_i$.

Suppose then that P passes through w_j where $i \leq j \leq N-1$. There is again only one possibility for the path P up to v_i, namely $v_N, \ldots, v_{j+1}, w_j, v_j, \ldots, v_i$. It can easily be seen that once the path passes through one w_j, the orientation of the path from v_j to v_i is forced and since ω contains no through vertex P cannot pass through another w_k without creating a cycle.

There are at most three possibilities for the part of P after v_i. Either it is empty, or it is the arc (v_i, w_{i-1}) or it is the arc (v_i, w_i).

Hence there are at most $O(N^2)$ possibilities for P.

Once P is determined (and passes through v_{N-1}) then there are at most 2 possibilities for the longest path into v_N.

Similarly, if P does not pass through v_{N-1}, but Q, the longest path into v_N passes through v_{N-1} then there are at most $O(N^2)$ choices for P and Q.

Finally there are 2 configurations where neither the longest path starting from v_N nor the longest path into v_N pass through v_{N-1}.

Hence there are $O(N^2)$ configurations at each stage.

The configurations and their associated cost are easily computed for G_2 by enumerating the 6 acyclic orientations of a triangle. To compute the configurations and their cost for G_N given the corresponding information for G_{N-1}, we run through all the configurations for G_{N-1} and add all possible acyclic orientations of the three edges $\{w_N, v_N\}$, $\{v_{N-1}, w_N\}$, $\{v_{N-1}, v_N\}$, that do not result in a through vertex.

For each configuration τ of G_{N-1} and orientations of the new edges, it is easy to compute which configuration τ' of G_N is obtained. The cost is either the cost of τ or the length of the longest path passing through v_{N-1}. This can easily be computed from τ. We keep a record of the least cost of each configuration of G_N obtained so far together with the corresponding acyclic orientations and update them as necessary. The final solution is obtained by finding the configuration of G_T with least cost.

In each of the $T-1$ stages, we run through $O(T^2)$ configurations and treat each one in constant time. Hence the overall running time is $O(T^3)$.

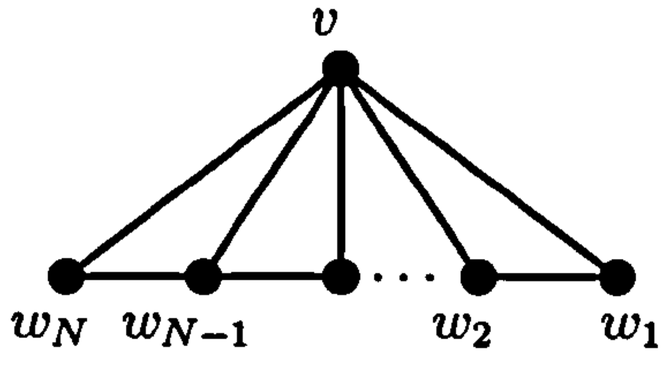
Let us now consider a similar problem. We want to find the minimum span of the wheel W_n. Again we will use dynamic programming to find the acyclic orientation in which the length of the longest directed path is minimised.

Consider the wheel W_n, where we denote the vertices along the outside by $w_1, \ldots, w_n$ and the vertex in the center by v. We are also given a corresponding constraint matrix.

In order to solve this problem we will start off by looking at a portion of a wheel, W'_N, which is obtained from W_n by deleting the edge $\{w_1, w_n\}$ and $\{w_{N+1}, \ldots, w_n\}$. See Figure 3.2 for W'_N.

As before, let A_N be the set of acyclic orientations of W'_N. A configuration of W'_N consists of six directed paths, the orientation of two edges and two boolean variables. The six directed paths are made of one beginning at each of v, w_1, w_N and one ending at each of v, w_1, w_N (the paths are not necessarily disjoint). The orientations are those of $\{v, w_1\}$ and $\{v, w_N\}$ and the boolean variables indicate whether there is a directed path from w_1 to w_N or from w_N to w_1.

Figure 3.2: W'_N



An acyclic orientation $\omega \in A_N$ respects a configuration $\tau \in W'_N$ if the six directed paths are the longest directed paths into and out of the relevant vertices, the two edges $\{v, w_1\}$ and $\{v, w_N\}$ are oriented as described and the boolean variables correctly indicate the presence of directed paths between w_1 and w_N. If ω respects τ, we write $\tau \leq \omega$.

The cost of τ is defined by

$$\text{cost}(\tau) = \min_{\omega \in A_n : \tau \leq \omega} lp_\omega.$$

We now show that there are $O(N^6)$ possible configurations of W'_N.

There are $O(N^2)$ choices for the longest path out of vertex v and also $O(N^2)$ choices for the longest path into vertex v. Now, if $w_1 \to w_2$ then there are $O(N)$ choices for the longest path out of w_1 by choosing the largest i, such that $w_1, w_2, \ldots, w_i$ form an initial segment of the longest path from w_1. Once w_i is chosen there are now only two possibilities depending on whether $w_i \to v$ or $v \to w_i$.

If $w_1 \to w_2$ then, if there is a path into w_1 then it must consist of the longest path into v together with the arc (v, w_1). Hence whenever $w_1 \to w_2$ there are $O(N)$ choices for the longest path into and out of w_1. Clearly the case when $w_2 \to w_1$ is exactly the same. Similarly there are $O(N)$ choices of the longest path into and out of w_N. Hence there are $O(N^6)$ possible sets of paths forming different configurations. The remaining information in a configuration whether $w_1 \to v$, $v \to w_1$, $w_N \to v$ or $v \to w_N$ and whether a path goes from w_1 to w_N or w_N to w_1 is needed so no cycles will be created by moving from state S_N to state S_{N+1}. There is only a constant number of possibilities for these. Hence there are $O(N^6)$ configurations.

At each stage we list each possible configuration τ together with the acyclic orientation $\omega(\tau)$ minimising lp_ω amongst those acyclic orientations ω such that $\tau \leq \omega$.

We begin by running through all acyclic orientations of W_2' and calculating the corresponding configurations. To generate the configurations of W_{N+1}' together with their associated acyclic orientations and costs, we run through all the configurations of W_N' one by one and try all possible orientations of the new edges v, w_{N+1} and w_N, w_{N+1}.

Given a configuration τ of W_N' first note that if $\tau \leq \omega$ then there is a directed path in ω from v to w_N (w_N to v) if and only if $v \to w_N$ ($w_N \to v$). Hence it is easy to determine whether there is an acyclic orientation respecting τ to which we may add a particular orientation of $\{v, w_{N+1}\}$ and $\{w_N, w_{N+1}\}$. If there is no such acyclic orientation then we move on to the next possibility.

Otherwise we may easily compute the longest paths to and from v, w_1, w_{N+1} by using τ and the orientation of the two new edges. Similarly we may easily determine whether there is a path from w_1 to w_{N+1} or w_{N+1} to w_1. Thus we can compute the specifications of a new configuration, τ'.

Let ω be formed by taking $\omega(\tau)$ and adding the orientation of the two new edges. The longest directed path in ω is either the longest directed path in $\omega(\tau)$ or passes through either v or w_{N+1}. So its length can easily be computed. If lp_ω is smaller than the best cost of τ' currently found then we update the cost of τ' by setting it to lp_ω and we let ω be the acyclic orientation associated with τ'.

In the final step we have to determine the orientation of the edge $\{w_1, w_n\}$. We run through all configurations of W_n'. Depending on whether these configurations include a directed path from w_1 to w_n or w_n to w_1 there may be one or two ways to orient $\{w_1, w_n\}$. The length of the longest path in an acyclic orientation formed in this way can be computed from the cost of the configuration of W_n' and the lengths of the longest paths to and from w_1 and w_N.

At each stage we run through $O(n^6)$ configurations and treat each one in constant time. Hence the total running time is $O(n^7)$.

3.3 General example

We will now consider a more general example and show that in polynomial time we can determine the minimum span. This example actually includes finding the span of a wheel, however we do not get as good a complexity or as simple a proof as we did in the previous section.

Consider a constraint matrix problem on a graph G where the vertex set V can be partitioned into subsets $Q = \{q_1, \ldots, q_l\}$ and $W = \{w_1, \ldots, w_k\}$. Note that $n = |V| = k+l$. The only edges allowed between vertices in W are those of the form $\{w_i, w_{i+1}\}$ for $i = 1, 2, \ldots, k-1$. Note that these edges do not have to be present. In other words, the vertices of W form a collection of paths.

Proposition 3.3.1 For fixed l, the minimum span of G can be found in time bounded by a polynomial in n.

Proof: We first count the number of possible paths in G from a vertex u to v. Let $X \subseteq Q$ and let $(x_1, x_2, \ldots, x_r)$ be an ordering of the vertices in X. We will now give an upper bound on the number of paths P, from u to v, where $V(P) \cap Q = X$ and P visits the vertices in X in the order $(x_1, x_2, \ldots, x_r)$. If we know the vertices immediately before each $x_i \in X$ and immediately after each such x_i, then we have uniquely determined our path. Therefore the upper bound is $n^{2r} \leq n^{2l}$. We now have to determine how many distinct sets X and orderings of X there are. Note that for a given $r = |X|$ there are at most $l(l-1)\ldots(l-r+1) \leq l!$ ways of picking the elements in X and ordering them. As there are at most $l+1$ possible values of r, we get that there are at most $(l+1)l! = (l+1)!$ ways of picking an ordered subset X of Q. This implies that there are at most $(l+1)!n^{2l}$ distinct paths from u to v.

Let $G_i = G : (V - \{w_{i+1}, w_{i+2}, \ldots, w_k\})$, for each $i = 1, 2, \ldots, k$. Let ω be an acyclic orientation of G_i, let $lp_\omega(u, v)$ denote the length of the longest path from u to v in (G_i, ω) and let $lp^{in}_\omega(v)$ ($lp^{out}_\omega(v)$) denote the length of the longest path into (out of) v in (G_i, ω). For each i and each possible set of the following values:

- $lp_\omega(q_a, q_b)$, $1 \leq a, b \leq |Q|$, $a \neq b$;

- $lp_\omega(q_a, w_i)$ and $lp_\omega(w_i, q_a)$, $1 \leq a \leq |Q|$;
- $lp^{in}_\omega(w_i)$ and $lp^{out}_\omega(w_i)$;
- $lp^{in}_\omega(q_a)$ and $lp^{out}_\omega(q_a)$, $1 \leq a \leq |Q|$,

we will find the acyclic orientation ω which minimises the length of the longest directed path in (G_i, ω) amongst those acyclic orientations for which the longest paths into, out of and between the relevant vertices are as in the list.

In other words if two different acyclic orientations agree on all the values in our above list, then we only remember the one where the longest path is minimum. We do not record the other orientation.

The total number of acyclic orientations is of exponential order, however we will now show that the number of acyclic orientations that give distinct values above is in fact polynomial.

Taking the length of longest paths from a certain vertex to another, we have shown that there are at most $(l+1)!n^{2l}$ possibilities. Note that we consider at most $l(l-1)+2l = l^2+l$ such longest paths above.

Looking now at the length of longest paths in or out of a certain vertex, we know that there are at most $n(l+1)!n^{2l}$ possibilities, as the other endpoint of the path can be chosen from at most n vertices. Again we note that we consider at most $2+2l$ such values.

Therefore the total number of acyclic orientations that must be recorded is

$$\left(((l+1)!n^{2l})^{l^2+l}((l+1)!n^{2l+1})^{2+2l}\right).$$

In the first stage we run through all acyclic orientations ω of G_1 recording those which for each set of values in our list, minimise the longest path in (G, ω).

At the ith stage we take each acyclic orientation ω in the list for G_{i-1} and run through all orientations of the edges incident with w_i. Let ω' be the orientation formed by adding to ω the orientation of all the edges incident with w_i. We use the values for $lp_\omega(q_a, q_b)$, $lp_\omega(q_a, w_{i-1})$ and $lp_\omega(w_{i-1}, q_a)$ to determine whether ω' is an acyclic orientation.

Let $Q' = Q \cup \{w_{i-1}\}$. Let $Q'_{out} = \{v \in Q' : (w_i, v)$ is an arc of $(G_i, \omega')\}$ and $Q'_{in} = \{v \in Q' : (v, w_i)$ is an arc of $(G_i, \omega')\}$.

Now if ω' is an acyclic orientation, we compute

$$lp_{\omega'}(q_a, q_b) = \max\{lp_\omega(q_a, q_b), \max_{v_1 \in Q'_{in}, v_2 \in Q'_{out}} \{lp_\omega(q_a, v_1) + lp_\omega(v_2, q_b) + c_{v_1 w_i} + c_{w_i v_2}\}\}$$

where $c_{v_1 w_i}$, $c_{w_i v_2}$ are the lengths of the edges joining v_1 to w_i and w_i to v_2 respectively.

Similarly we may compute $lp_\omega(q_a, w_i)$, $lp_\omega(w_i, q_a)$ for all a, $lp^{in}_{\omega'}(w_i)$, $lp^{out}_{\omega'}(w_i)$ and finally $lp^{in}_{\omega'}(q_a)$, $lp^{out}_{\omega'}(q_a)$ for all a. Clearly all these values may be computed in polynomial time for fixed l.

Note that the length of the longest path in (G_i, ω') may easily be computed using the length of the longest path in (G_i, ω) and the values $lp^{in}_{\omega'}(w_i)$ and $lp^{out}_{\omega'}(w_i)$. We record any acyclic orientation ω', together with the length of its longest path, if it is the shortest one found so far with a particular list of lengths of longest paths.

After the final stage when $i = k$, we run through all the acyclic orientations of G_k, in the list that we have constructed, and choose the one minimising the length of the longest path.

We have shown that we may find the minimum span of G in time $O(n^c)$ where c depends only on l. □

Chapter 4

Cyclic Channel Metric

In the previous chapter we studied labelling problems where the constraints were in the form of lower bounds on the absolute difference of the frequencies received by adjacent vertices. In this chapter we introduce the cyclic channel metric.

We then mention some results using the absolute differences and then generate analogous results using the cyclic channel metric. Finally we introduce the one-close-neighbour problem and give some results on the minimum span of various classes of graphs in this problem.

4.1 Notation and definitions

Suppose we have a graph $G = (V, E)$ and a frequency assignment on G with span σ, that is a function $f : V \to \{0, 1, \ldots, \sigma - 1\}$. Then the cyclic channel metric is defined by

Definition 4.1.1

$$|f(v_i) - f(v_j)|_\sigma = \min\{|f(v_i) - f(v_j)|, \sigma - |f(v_i) - f(v_j)|\}.$$

One of the advantages behind the cyclic channel metric is, that we can assign several channels to one vertex v_i, since we can just add multiples of the span to $f(v_i)$. Another motivation for using the cyclic channel metric is that all the channels used have the same status, since they all have two neighbours.

In this section we will consider the following problem. We are given a constraint matrix problem with underlying graph G and are required to find a minimum span feasible frequency assignment except now the constraints are specified using the cyclic channel matrix, that is we require for all i, j

$$|f(v_i) - f(v_j)|_\sigma \geq c_{ij}.$$

Whenever we are talking about a span derived by the cyclic channel metric, we will denote this by σ_c.

4.2 Known results

Earlier we have seen that if all the weights in a constraint matrix problem are zero or one then finding the minimum span of a feasible frequency assignment is equivalent to finding the chromatic number of the underlying graph. The same is true when the constraints are defined using the cyclic channel metric.

Minimum span problems defined using the cyclic channel metric also have a connection with the circular chromatic number of a graph, originally called *star-chromatic number*, see [52].

Definition 4.2.1 The *circular chromatic number* of a graph G is the infimum of all $\chi \in \mathbb{R}$ such that there exists $f : V \to [0, \chi)$ with $|f(v_i) - f(v_j)|_\chi \geq 1$.

In [52] it is proved that the circular chromatic number always has to be rational. The author shows that if the circular chromatic number is written in its lowest terms, the numerator is less than or equal to the number of vertices.

The connection between circular chromatic number and constraint matrix problems using the cyclic channel metric is as follows. Take a graph G with $V(G) = \{v_1, \ldots, v_n\}$, and construct a constraint matrix C where $c_{ij} = p$ if v_i and v_j are adjacent in G and $c_{ij} = 0$ otherwise. Then

$$\sigma_c(C) = \lceil p\chi_c(G) \rceil .$$

In Chapter 3, where the metric is the absolute difference between frequencies, we see that if we require a feasible frequency assignment with the additional restriction that $f(v_1) \leq f(v_2) \ldots \leq f(v_n)$, then the minimum span can be computed in time $O(n^2)$ since all we must do is find a longest path in an acyclic digraph.

In [40], the author shows that a similar result holds for the cyclic channel metric. More precisely it is shown that if we specify a cyclic ordering on the vertices and require a frequency assignment for which the assigned frequencies respect this cyclic ordering then the minimum span can be found in time $O(n^2 + mn)$. However this result is considerably more complicated.

Leese and Noble study in [37] constraint matrix problems formed in the following way. They take a graph G with vertex set $\{v_1, v_2, \ldots, v_n\}$ and for fixed integers $0 \leq q \leq p$ set

$$c_{ij} = \begin{cases} p & \text{if } d(v_i, v_j) = 1, \\ q & \text{if } d(v_i, v_j) = 2, \\ 0 & \text{otherwise.} \end{cases}$$

When the graph is a tree, then the minimum span for all values of p and q is as follows.

Theorem 4.2.2 Let T be a tree with maximum degree $\Delta \geq 1$.

$$\sigma_c(T, p, q) = 2p + (\Delta - 1)q.$$

Another result is on even cycles.

Theorem 4.2.3 For any C_{2n} we have

$$2p + q \leq \sigma_c(C_{2n}, p, q) \leq 2p + 2q.$$

Exact descriptions of the minimum span for even and odd cycles for all values of p and q can also be found in [37].

The above results were motivated by the results in [30] where the authors give the cyclic spans of the line, triangular and square lattices.

4.3 The greedy algorithm

In Chapter 2 we define the minimum and maximum vertex degree. For the next problem we need yet another type of degree, the weighted degree of a vertex.

Definition 4.3.1 The *weighted degree* of a vertex v_i is defined by

$$\mathrm{dw}(v_i) = \sum_{j:\{i,j\}\in E} c_{ij}.$$

When the constraints are defined as the straightforward absolute difference, then we know from [38] that the span is bounded as follows.

Theorem 4.3.2 Given a constraint matrix problem with underlying graph G (where constraints depend on absolute difference)

$$\sigma(G) \leq \max\{\mathrm{dw}(v_i) + 1 | 1 \leq i \leq n\}.$$

We will modify this result and give an upper bound on the span of a constraint matrix problem in which the cyclic channel metric is used.

Proposition 4.3.1 Given a constraint matrix problem with underlying graph G, with $V(G) = \{v_1, \ldots, v_n\}$, and in which constraints depend on the cyclic channel metric,

$$\sigma_c(G) \leq \max_i\{\mathrm{dw}(v_i) + \max\{c_{ij} | v_j \in N(v_i)\}\}.$$

Before we prove the result we first define the many-passes greedy algorithm.

Definition 4.3.3 For the *many-passes greedy algorithm* we assume that we are given a fixed ordering $(v_1, \ldots, v_n)$ of the vertices. Initially all vertices are unassigned. At the first stage of the algorithm, starting at vertex v_1, we run through all the vertices assigning frequency 0 whenever possible. At the $(k+1)$th stage, again starting at vertex v_1, we run through all the vertices in the given order and assign frequency k whenever possible.

Proof of Proposition 4.3.1: Using the proof from [38] we see that in the many-passes greedy algorithm every vertex v_i in G receives a frequency f_i which is at most $\mathrm{dw}(v_i)$. To see this consider the case where the many-passes greedy algorithm is about to assign frequency f_i to vertex v_i. For each frequency $f \in \{0, \ldots, f_i - 1\}$, there exists $v_j \in N(v_i)$ such that $f(v_j) \leq f$ and $f(v_j) + c_{ij} \geq f + 1$, otherwise vertex v_i would have received a frequency f from one of the previous passes. Thus for all $v_j \in N(v_i)$, v_j forbids c_{ij} frequencies at v_i and so

$$f_i \leq \sum_{v_j \in N(v_i)} c_{ij} = \mathrm{dw}(v_i).$$

We now show that this assignment is a feasible frequency assignment with span

$$\sigma'_c = \max_i \{\mathrm{dw}(v_i) + \max\{c_{ij} | v_j \in N(v_i)\}\}.$$

Let $\{v_i, v_j\}$ be an edge of G and suppose that $f(v_j) < f(v_i)$. By the above we know that $f(v_i) \geq f(v_j) + c_{ij}$. It remains to show that $\sigma'_c - (f(v_i) - f(v_j)) \geq c_{ij}$.

Now

$$f(v_i) + c_{ij} \leq f(v_i) + \max\{c_{ik} | v_k \in N(v_i)\} \leq \mathrm{dw}(v_i) + \max\{c_{ik} | v_k \in N(i)\} \leq \sigma'_c.$$

Hence $\sigma'_c \geq f(v_i) - f(v_j) + c_{ij}$ and the result follows. □

4.4 One-close-neighbour problem

The model of interference introduced in Section 2.3 considers only the interference caused by pairs of transmitters and assumes that providing each of these interferences is not too large then a user will receive an acceptable signal.

In reality the level of interference experienced by a user is the sum of the levels of interference between the transmitter which the user is trying to receive and the other transmitters.

The interference caused by a pair of transmitters is a decreasing function of their channel separation. This suggests that it may be worthwhile to look for frequency assignments in which each transmitter may have at most one potentially interfering

transmitter receiving a relatively close channel as long as all the other potentially interfering transmitters receive distant channels.

This leads us to consider the *one-close-neighbour problem.* Suppose we are given a graph G and non-negative integers p and q. We wish to find a frequency assignment f such that for each vertex v_i,

$$|f(v_i) - f(v_j)|_\sigma \geq p$$

for all $v_j \in N(v_i)$, and

$$|f(v_i) - f(v_j)|_\sigma \geq p + q$$

for all but one $v_j \in N(v_i)$.

In other words we need to find a labelling where we allow one neighbour of a vertex to have a moderately close label and the remaining vertices have to have greater separation. Note that we are free to choose which pairs of vertices may receive labels at least p frequencies apart.

With this model we are not trying to solve all the practical problems occurring within the frequency assignment problem, however the one-close-neighbour problem is slightly closer to reality than the model of interference. Also we are interested in a new mathematical formulation of the frequency assignment problem.

It is easy to see that the complexity of the one-close-neighbour problem is NP-hard in the case when $p = 1$ and $q = 0$, as this would correspond to vertex colouring. However we do not know what the complexity is if we put $p = q = 1$.

4.4.1 Bipartite graph

First let us consider a bipartite graph G and the one-close-neighbour problem. Note that if $\Delta(G) = 1$ then we may construct an assignment where all pairs of adjacent vertices are only separated by p channels. Hence $\sigma_c = 2p$.

Theorem 4.4.1 The minimum span for a bipartite graph G with $\Delta(G) > 1$ is $\sigma_c(G) = 2p + 2q$.

Proof: By assigning every vertex of one partite set with $f(u) = 0$ and every vertex of the second partite set with $f(v) = p+q$, we get a feasible assignment with span $2p+2q$. Note that at least one pair of vertices must be separated by $p+q$, so $\sigma_c \geq 2(p+q)$. □

4.4.2 Odd cycle

Consider the one-close-neighbour problem for the odd cycle C_{2n+1}.

Theorem 4.4.2 For all $n \geq 1$, $\sigma_c(C_{2n+1}) = \max\{2p+2q, 2p+q+\lceil\frac{p+q}{n}\rceil\}$.

Proof: Since $n \geq 1$, some constraint has to be at least $p+q$, which implies that $\sigma_c(C_{2n+1}) \geq 2(p+q)$, as we are considering cyclic constraints.

Let $V(C_{2n+1}) = \{v_0, v_1, \ldots, v_{2n}\}$ and let f be an optimal frequency assignment on C_{2n+1}, with respect to our given constraints. Let σ denote $\sigma_c(C_{2n+1})$ for simplicity. Let $c_i = (f(i+1) - f(i)) \pmod{\sigma}$, for all $i = 0, 1, \ldots, 2n$ (vertex v_{2n+1} is equal to vertex v_0). Note that for all i we have $p \leq c_i \leq \sigma - p$, and that either $p+q \leq c_i \leq \sigma-p-q$ or $p+q \leq c_{i+1} \leq \sigma-p-q$ also holds for all i. Let $C = \sum_{i=0}^{2n} c_i$. By the above restrictions on the constraints we see that the following holds:

$$\begin{aligned} C &\geq (n+1)(p+q) + np \\ C &\leq (n+1)(\sigma-p-q) + n(\sigma-p). \end{aligned}$$

Furthermore we note that $C = f(2n+1) - f(0) \equiv 0 \pmod{\sigma}$. Hence $C = m\sigma$, for some integer m. From the above it follows straightaway, that

$$\begin{aligned} m\sigma &\geq (2n+1)p + (n+1)q \\ \text{and} & \\ (2n+1-m)\sigma &\geq (2n+1)p + (n+1)q. \end{aligned}$$

Hence

$$\sigma \geq \left\lceil \max\left\{ \frac{(2n+1)p+(n+1)q}{m}, \frac{(2n+1)p+(n+1)q}{2n+1-m} \right\} \right\rceil.$$

Thus

$$\begin{aligned} \sigma &\geq \min_{1\leq m\leq n} \left\lceil \max\left\{ \frac{(2n+1)p+(n+1)q}{m}, \frac{(2n+1)p+(n+1)q}{2n+1-m} \right\} \right\rceil \\ &= \left\lceil \frac{(2n+1)p+(n+1)q}{n} \right\rceil \\ &= 2p+q+\left\lceil \frac{p+q}{n} \right\rceil. \end{aligned}$$

We have now shown that $\sigma \geq \max\{2p+2q, 2p+q+\lceil \frac{p+q}{n} \rceil\}$. In order to show equality, we let $\sigma = \max\{2p+2q, 2p+q+\lceil \frac{p+q}{n} \rceil\}$. We will now define a frequency assignment g, which fulfills the given constraints with span σ. Let $a = \sigma - 2p - 2q$, and note that $a \geq 0$ and $a \geq \frac{p+q}{n} - q$. The second of these inequalities gives us that $0 \leq na + (n-1)q - p$. Therefore it is easy to see that the following holds.

$$0 \leq na + (n-1)q - p \leq (n+1)a + n(a+2q).$$

Now let $e_1, e_2, \ldots, e_{2n+1}$ be integers such that $0 \leq e_l \leq a$, when l is odd, and $0 \leq e_l \leq a+2q$, when l is even, and $\sum_{l=1}^{2n+1} e_l = na + (n-1)q - p$ (this is possible, by the above equation). Define g, as follows, for all $i = 0, 1, \ldots, n$.

$$\begin{aligned} g(v_{2i}) &= 2ip + iq + \sum_{l=1}^{2i} e_l \pmod{\sigma} \\ g(v_{2i+1}) &= (2i+1)p + (i+1)q + \sum_{l=1}^{2i+1} e_l \pmod{\sigma}. \end{aligned}$$

Note that $g(v_{2n+1}) = (2n+1)p+(n+1)q+[na+(n-1)q-p] = n(2p+2q+a) \equiv 0 = g(v_0) \pmod{\sigma}$. Furthermore g is a feasible frequency assignment with span σ as the following holds:

$$\begin{aligned} p+q &\leq g(v_{2i+1}) - g(v_{2i}) \pmod{\sigma} \\ &= p+q+e_{2i+1} \leq \sigma - (p+q) \quad \text{for } i = 0,1,2\ldots,n \\ p &\leq g(v_{2i}) - g(v_{2i-1}) \pmod{\sigma} \\ &= p+e_{2i} \leq \sigma - p \quad \text{for } i = 1,2,3\ldots,n. \end{aligned}$$

□

4.4.3 Complete graph

We now consider the complete graph K_n for the one-close-neighbour problem.

Theorem 4.4.3 For all integers $n \geq 3$ the following holds

$$\sigma_c(K_n) = np + q\left\lceil \frac{n}{2} \right\rceil .$$

Note that $\sigma_c(K_1) = 0$ and $\sigma_c(K_2) = 2p$.

In order to prove the above theorem we need the following lemma.

Lemma 4.4.4 Given a constraint matrix problem on a graph G, then

$$\sigma_c \geq \min_{\pi} \sum_{i=1}^{n} c_{\pi(i)\pi(i+1)}$$

where π is an ordering of $V(G)$ and $\pi(n+1) = \pi(1)$.

Proof of Theorem: By symmetry we can consider the problem as a constraint matrix problem (with cyclic channel metric) on a complete graph, such that there is a set of edges forming a maximum matching which have constraint p and the rest of the edges have constraint $p+q$.

Label the vertices $v_1, \ldots, v_n$ so that $v_1v_2, v_3v_4, \ldots$ are matching edges. A shortest TSP tour in G is $v_1v_2, \ldots, v_nv_1$ so Lemma 4.4.4 shows that

$$\sigma \geq p\lfloor \frac{n}{2} \rfloor + (p+q)(n - \lfloor \frac{n}{2} \rfloor) = pn + \left\lceil \frac{n}{2} \right\rceil q.$$

On the other hand, a feasible assignment is

$$\begin{aligned} f(v_{2i-1}) &= (i-1)(2p+q) \\ f(v_{2i}) &= (i-1)(2p+q)+p. \end{aligned}$$

This has span $pn + \lceil \frac{n}{2} \rceil q$.

□

4.4.4 Complete multipartite graphs

We will now derive the optimal span for complete multipartite graphs. As a b-partite graph of order b is a complete graph, the results below generalise Theorem 4.4.3 from Section 4.4.3. We first need the following lemma.

Lemma 4.4.5 If G is a complete b-partite graph, with a partite sets of size one, $0 \leq a \leq b$, then

$$\sigma_c(G) \leq (p+q)b - \lfloor \frac{a}{2} \rfloor q.$$

Proof: Let G be a complete multipartite graph, with partite sets $V_1, V_2, \ldots, V_b$. Assume without loss of generality that $V_1, V_2, \ldots, V_a$ have size 1, and all other partite sets have size greater than 1. Now assign every vertex in V_i $(i = 1, 2, \ldots, b)$ frequency $(i-1)p + q\lfloor \frac{i-1}{2} \rfloor$, when $i \leq a$ and frequency $(i-1)(p+q) - q\lfloor \frac{a}{2} \rfloor$, when $i > a$. We note that these frequencies form a feasible frequency assignment with span $(p+q)b - \lfloor \frac{a}{2} \rfloor q$. Therefore $\sigma_c(G) \leq (p+q)b - \lfloor \frac{a}{2} \rfloor q$. □

We will now present two alternative proofs of Theorem 4.4.6 below. The first proof is more direct but quite technical. The second proof is a bit shorter but perhaps it does not give as good an intuition of what is going on. As the proofs are not too long, we will include both proofs here.

Theorem 4.4.6 The optimal span for a complete b-partite graph, with a partite sets of size one, is

$$\sigma_c(G) = (p+q)b - \lfloor \frac{a}{2} \rfloor q.$$

Proof 1 of Theorem 4.4.6: Let G be a complete multipartite graph, with partite sets $V_1, V_2, \ldots, V_b$, and assume that a partite sets have size 1. By Lemma 4.4.5 we see that $\sigma_c(G) \leq (p+q)b - \lfloor \frac{a}{2} \rfloor q$. In order to show equality, we assume that f is an optimal feasible frequency assignment, such that the number of distinct frequencies used is minimum. Denote $\sigma_c(G)$ by just σ, for simplicity, and let $[i,p]$ denote the interval $\{i, i+1, i+2, \ldots, p\}$, where all numbers are taken modulo σ (so p may be less than i). We now prove the following claims.

Claim 1: There are no vertices u and v from the same partite set V_i, such that there are vertices from other partite sets with frequencies in the interval $[f_u, f_v]$ and in the interval $[f_v, f_u]$.

Proof: Assume two vertices u and v from the same partite set V_i do not fulfill the statement in Claim 1. The claim follows trivially if u and v have only one neighbour so suppose this is not the case. Clearly either $[f(u)-p+1, f(u)+p+q-1]$ or $[f(u)-p-q+1, f(u)+p-1]$ contains only frequencies assigned to vertices from V_i. Similarly either $[f(v)-p+1, f(v)+p+q-1]$ or $[f(v)-p-q+1, f(v)+p-1]$ contains only frequencies assigned to vertices from V_i.

First assume that $[f(u)-p+1, f(u)+p+q-1]$ and $[f(v)-p+1, f(v)+p+q-1]$ contain only frequencies assigned to vertices from V_i. If in fact $[f(v)-p-q+1, f(v)+p+q-1]$ contains only frequencies assigned to vertices from V_i then we may just reassign every vertex receiving a frequency in $[f(u)-p+1, f(u)+p+q-1]$ with $f(v)$. The case when $[f(u)-p-q+1, f(u)+p+q-1]$ contains only frequencies assigned to vertices from V_i is similar. Hence we may suppose there is some vertex $u' \notin V_i$ receiving a frequency in $[f(u)-p-q+1, f(u)-p]$ and some vertex $v' \notin V_i$ receiving a frequency in $[f(v)-p-q+1, f(v)-p]$. In order to satisfy the constraints with u and v there can be at most one vertex receiving a frequency in $[f(u)-p-q+1, f(u)-p]$ and at most one vertex receiving a frequency in $[f(v)-p-q+1, f(v)-p]$. Reassign every vertex receiving a frequency in $[f(u)-p+1, f(u)+p+q-1]$ with $f(v)$ and reassign v' with $f(u)$. It is easy to check that all the constraints are satisfied and that at least one fewer frequency is used, a contradiction.

This completes the proof of Claim 1, as all remaining cases can be handeled analogously.

Claim 2: All vertices from the same partite set have the same frequency.

Proof: For partite set V_i let x_i and y_i be vertices in V_i ($x_i = y_i$ is possible), such that all vertices in V_i have frequency in $[f(x_i), f(y_i)]$. We now consider the following two possibilities.

If $a = 0$, then let all vertices in V_i get frequency $f(x_i)$. Note that this implies that any two vertices in different partite sets get frequencies at least $p + q$ apart, without changing the span. So if two vertices in the same partite set have different frequencies, we obtain a contradiction against the number of distinct frequencies used.

So now consider the case when $a \geq 1$. Let w and z be vertices from partite sets of size 1 ($w = z$ is possible), such that these vertices are the only vertices from partite sets of size 1 with frequencies in $[f(w), f(z)]$. Let V_i be a partite set such that $[f(x_i), f(y_i)] \subset [f(w), f(z)]$. We will show that $f(x_i) = f(y_i)$, which would complete the proof of Claim 2 as we have considered arbitrary w, z and V_i. So assume that $f(x_i) \neq f(y_i)$.

Let V^* be a set containing all partite sets, V_r, with $f(x_r) \in [f(w), f(z)]$.

If w is the only vertex with frequency in $[f(w), f(w) + p + q - 1]$, then let all vertices in $V_r \in V^*$ get frequency $f(x_r)$. This rearrangement of frequencies will fulfill our constraints, and will use less distinct frequencies, a contradiction.

If z is the only vertex with frequency in $[f(z) - p - q + 1, f(z)]$, then we get a contradiction, analogously to above. So assume that there are vertices, apart from w and z, with frequencies in both $[f(w), f(w)+p+q-1]$ and $[f(z)-p-q+1, f(z)]$. This implies that $w \neq z$ and that the only vertices with frequencies in $[f(w) - p - q + 1, f(w)] \cup [f(z), f(z) + p + q - 1]$ are w and z. Now let $f(z) = f(w) + p$, and let all vertices in V_r get frequency $f(x_r) + p + q$, for all $V_r \in V^*$. If $r, r' \in V^*$ then $|f(x_r)-f(x_{r'})| \geq p+q$ because $|V_r| \geq 2$, so no constraints are broken between vertices of V_r and $V_{r'}$. Note that no vertex gets a frequency in $[f(z)+1, f(z)+p+q-1]$ as there is no r such that $f(x_r) \in [f(z)-p-q+1, f(z)]$. Hence there are no constraints broken

with vertices receiving frequencies outside of the interval $[f(w), f(z)]$. Similarly there is no vertex receiving a frequency in $[f(w)+p+1, f(w)+2p+q-1]$ so no constraints are broken with z.

Note that if f' was the frequency of z before we changed it, then no vertex gets a frequency in $[f'+1, f'+p+q-1]$, as there was no $f(x_r) \in [f'-p-q+1, f']$ (there was some $f(y_r) \in [f'-p-q+1, f']$, but these frequencies have now been changed). Therefore our new assignment of frequencies fulfils our constraints, and uses less distinct frequencies. This contradiction completes the proof of Claim 2.

As all vertices in the same partite set get the same frequency, we note that the constraints with separation p must be between vertices forming partite sets of size one. Hence there are at most $\lfloor \frac{a}{2} \rfloor$ such constraints. Using Lemma 4.4.4 we see that $\sigma \geq (p+q)b - q\lfloor \frac{a}{2} \rfloor$, since any cyclic ordering of the vertices of G must include at least b pairs of adjacent vertices. □

Proof 2 of Theorem 4.4.6: Let G be a complete b-partite graph with a partite sets of size 1. By Lemma 4.4.5 we see that $\sigma_c(G) \leq (p+q)b - \lfloor \frac{a}{2} \rfloor q$.

Now let G' be a b-partite graph, with a partite sets of size one and $b-a$ partite sets of size two. We will show that $\sigma_c(G') \geq (p+q)b - \lfloor \frac{a}{2} \rfloor q$, which would prove the desired result as G' is a subgraph of G.

Let f be an optimal feasible frequency assignment for G'. Let σ_c^f be the span of f. Let H be an edge-coloured graph with $V(H) = V(G')$ and the following edges. Vertices x and y belong to the same partite set in G' if and only if there is a red edge between vertices x and y in H. Vertices u and w belong to different partite sets in G' and $|f(u) - f(v)|_\sigma < p+q$, if and only if there is a blue edge between u and v in H. The graph H only contains the above defined red and blue edges. Note that no two red edges are adjacent and no two blue edges are adjacent, so the maximum degree in H is at most 2.

Clearly all connected components of H are either a cycle of even length or a path. Let P_1 be the set of all connected components containing more blue edges than red (that is they are paths beginning and ending in blue edges). Let P_2 be the

set of all connected components that are paths not in P_1, including isolated vertices, and let C be the set of all connected components that are cycles.

Orient the edges of paths in P_2 so that those with length at least one, end with a red edge, and let S contain every second vertex on all the paths in P_2, starting from the first vertex (that is, we take the first, third, fifth,.... vertex) and every second vertex in all cycles in C. Furthermore let S also contain every second vertex from the paths in P_1 as well as both end points in all paths in P_1. (Note that there are two choices for the vertices to add to S in the case of cycles and paths in P_1 and it does not matter which we choose.)

S contains all vertices not adjacent to a red edge and exactly one endpoint of each red edge. Hence S contains exactly one member of each partite set. $H : S$ contains exactly $|P_1|$ blue edges. Since S contains at least $2|P_1|$ vertices not adjacent to a red edge, we have $2|P_1| \leq a$ and so $|P_1| \leq \lfloor \frac{a}{2} \rfloor$ which means that there are at most $\lfloor \frac{a}{2} \rfloor$ pairs of vertices $\{u, v\} \subseteq S$ such that $|f(u) - f(v)|_{\sigma_c'} < p + q$.

Let $G'' = G' : S$ and let the constraint on an edge $\{u, v\}$ of G'' be p if $|f(u) - f(v)|_{\sigma_c'} < p + q$ and $p + q$ otherwise. Note that G'' is a complete graph and all constraints in G'' require a separation of at leat $p+q$ except for at most $\lfloor \frac{a}{2} \rfloor$ constraints which require a separation of p. Exactly as in the proof of Theorem 4.4.3, we see that $\sigma_c(G'') \geq (p+q)b - \lfloor \frac{a}{2} \rfloor q$. Now $\sigma_c(G'') \leq \sigma_c(G')$ so the result follows. □

Chapter 5

$L(2,1)$-Labelling

In this chapter we study a variation of the frequency assignment, where the main idea is that if transmitters are close to each other then they should receive different frequencies and transmitters that are "very" close to each other are assigned frequencies that are at least two apart. This is known as the $L(2,1)$-labelling of a graph and was first discussed in 1988 by Roberts and Griggs [20].

5.1 Notation and definitions

Definition 5.1.1 An $L(2,1)$-labelling of a graph G is a function $L : V \to \mathbb{Z}^+ = \{0,1,2,3\dots\}$ such that whenever u and v are adjacent we have $|L(u) - L(v)| \geq 2$ and whenever u and v have a common neighbour then $L(u) \neq L(v)$.

An $L(2,1)$-labelling is a k-$L(2,1)$-labelling if for all $v \in V$, we have $L(v) < k$. We denote $\sigma(G)$, also called the $L(2,1)$-span of G, by the minimum k such that there exists a k-$L(2,1)$-labelling. We define $\lambda(G)$ to be $\sigma(G) - 1$, which implies that $\lambda(G)$ is the largest allowed label in a $\sigma(G)$-$L(2,1)$-labelling.

Lemma 5.1.2 [14] $\lambda(G) \leq \lambda(H)$ for any subgraph G of a graph H.

Since the $L(2,1)$-labelling is in fact a proper vertex colouring with the additional property at distance two, $\lambda(G)$ has the following connection to the chromatic number $\chi(G)$.

Definition 5.1.3 The k-th power of a simple graph G is the simple graph G^k with vertex set $V(G)$ and edge set $\{\{v_i, v_j\} : d(v_i, v_j) \leq k\}$.

Lemma 5.1.4 [14] $\chi(G) - 1 \leq \lambda(G) \leq 2\chi(G^2) - 2$ for any graph G.

5.2 Known results for the $L(2,1)$-labelling

In [20] Griggs and Yeh found the exact λ for paths (P_n) and cycles (C_n) on n vertices.

$$\lambda(P_n) = \begin{cases} 2 & \text{if } n = 2 \\ 3 & \text{if } n = 3 \\ 3 & \text{if } n = 4 \\ 4 & \text{if } n \geq 5. \end{cases}$$

$\lambda(C_n) = 4$, for any n.

The exact λ for wheels (W_n) on $n+1$ vertices was shown in [55] to be $\lambda(W_n) = n+2$, when $n \geq 4$.

A lower bound for $\lambda(G)$ in terms of the maximum degree is given by

Lemma 5.2.1 [20] $\lambda(G) \geq \Delta + 1$ for any graph G of maximum degree Δ. If $\lambda(G) = \Delta + 1$, then for any $\sigma(G)$-$L(2,1)$-labelling f and any vertex v of maximum degree Δ, $f(v) = 0$ or $\Delta + 1$. In this case, for any $x \in V(G)$, $N[x]$ contains at most two vertices of degree Δ.

And an upper bound is

Lemma 5.2.2 [14] $\lambda(G) \leq \Delta^2 + \Delta$ for any graph G of maximum degree Δ.

Another bound has been proven by Jonas, which for the case of $\Delta = 3$ improves the above bound from 12 to 11.

Lemma 5.2.3 [33] $\lambda(G) \leq \Delta^2 + 2\Delta - 4$ for $\Delta(G) \geq 2$.

5.3 Polynomial algorithm for $\lambda(T)$

Griggs and Yeh proved in [20] that for a tree T with maximum degree Δ, $\lambda(T) = \Delta+1$ or $\Delta+2$. They then conjectured that it is NP-complete to determine whether $\lambda(T) = \Delta + 1$.

Chang and Kuo however found in [14] a polynomial algorithm to verify whether $\lambda(T) = \Delta + 1$.

We will now describe the main ideas of the algorithm.

To start of with, it is a good idea to check two properties of the given tree.

Firstly, does the tree have a vertex v whose closed neighbourhood $N[v]$ contains at least three vertices of degree Δ? If so, then we know straightaway by Lemma 5.2.1, that $\lambda(T) = \Delta + 2$.

Secondly, does there exist a leaf u whose unique neighbour v has degree $d(v) < \Delta$? If so, then $T - u$ still has maximum degree Δ. By Lemma 5.1.2 and the same arguments used to proof Lemma 5.2.1, we get

$$\lambda(T-u) \leq \lambda(T) \leq \max\{\lambda(T-u), d(v)+2\} \leq \lambda(T-u)$$

and therefore $\lambda(T) = \lambda(T-u)$. Since finding $\lambda(T)$ is now the same as finding $\lambda(T-u)$, we can continue this step until any leaf of the tree is a neighbour of a vertex of degree Δ.

Whether or not we have checked the above properties, we are now given a tree T with $|V(T)| \geq 2$ and maximum degree Δ.

Let T_{v_i} be the subtree rooted at v_i. For technical reasons we now add a vertex v_i', which will be adjacent to v_i only. Call this new tree T'_{v_i}.

The main part of the algorithm consists of a stage for each vertex in turn. We run through the vertices of T working upwards form the leaves to the root. At the stage where we consider v_i we construct the set S_i of all pairs (a,b), $0 \leq a, b \leq \Delta+1$, such that there exists an $L(2,1)$ labelling of T'_{v_i} with v_i' receiving label a and v_i receiving label b.

If v_i is a leaf then S_i will be $\{(a,b) : 0 \leq a, b \leq \Delta+1, |a-b| \geq 2\}$.

Otherwise $(a,b) \in S_i$ if $|a-b| \geq 2$ and if the children of v_i are $w_1, \ldots, w_s$ then there exist distinct labels $c_1, \ldots, c_s$ such that for all j, $c_j \neq a$ and $(b, c_j) \in S_j$.

In order to determine if $(a,b) \in S_i$ we construct the bipartite graph G, with partite sets $A = \{w_1, w_2, \ldots, w_s\}$, $B = \{0, 1, \ldots, \Delta+1\}$ and edge set

$$E = \{(w_j, c) : c \neq a \text{ and } (b,c) \in S_j\}.$$

Then $(a,b) \in S_i$ if and only if G has a matching of size s. Any bipartite matching algorithm may be used.

Working our way up in the tree we can decide whether $\lambda(T) = \Delta + 1$, in polynomial time.

The running time of the algorithm is the time taken to solve $O(n\Delta^2)$ bipartite matching problems on graphs with $O(\Delta)$ vertices. Using the standard Hungarian algorithm this gives an overall complexity of $O(n\Delta^5)$.

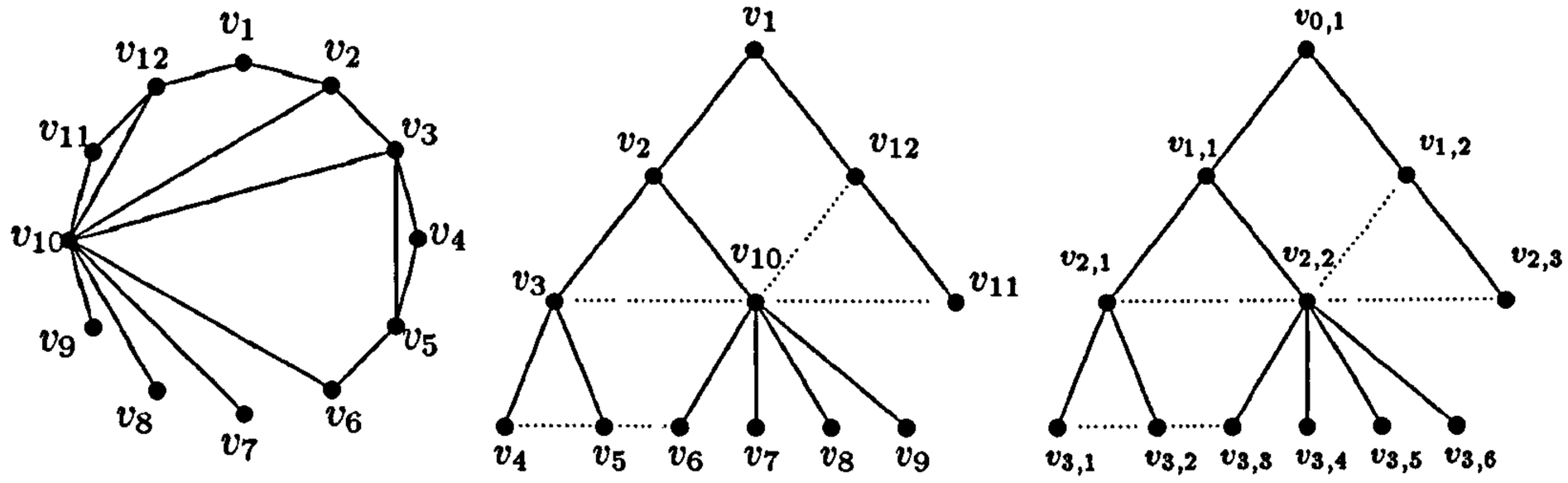
5.4 Bounds for outerplanar graphs

In [11] Bodlaender et al. proved that for an outerplanar graph G, $\lambda(G)$ is at most $\Delta(G)+9$. For $\Delta(G) \geq 8$ this was improved to $\Delta(G)+3$ in [13], by Calamoneri and Petreschi. The proof in [13] is correct for $\Delta(G) \geq 12$, however we will illustrate why there are certain problems when $\Delta(G) \leq 11$. Before we illustrate these problems, we will give an outline of the proof of the main result in [13]. We will also give an analogous result to the main result in [13] which uses the cyclic channel metric. In order to do this we need a few definitions and lemmas.

Let G be an outerplanar graph with $V(G) = \{v_1, v_2, \ldots, v_n\}$. The ordering $(v_1, v_2, \ldots, v_n)$ is said to be an *outerplanar-ordering* of $V(G)$, if the following holds: If all edges $v_i v_{i+1}$, for $i = 1, 2, \ldots, n$ $(v_{n+1} = v_1)$ are added then the graph remains outerplanar, and while traversing the outer face (in a clockwise direction and starting from v_1 say), the vertices are visited in the order $v_1, v_2, \ldots, v_n$. See for example Figure 5.1 for an example.

Let G be an outerplanar graph with outerplanar-ordering $(v_1, v_2, \ldots, v_n)$. An *ordered breadth first search* tree (OBFS) is a breadth first search tree started at v_1,

Figure 5.1: This is an example of an outerplanar graph, G, with its OBFS tree, and the labelling of the OBFS tree. The dotted lines illustrate edges in G, which are not in the OBFS tree.



where vertices with lower index are always chosen before vertices with higher index. See Figure 5.1 for an example of an outerplanar graph and its OBFS. The dotted lines indicate edges of G that are not tree edges.

Assume that the root v_1 of the OBFS is at level 0, the children of v_1 are at level 1, etc. Furthermore the left to right ordering of each level induces a numbering of the nodes, so let $v_{l,i}$ denote the i'th node on level l. See Figure 5.1 for an example. In [13] the following properties of an OBFS for an outerplanar graph are given.

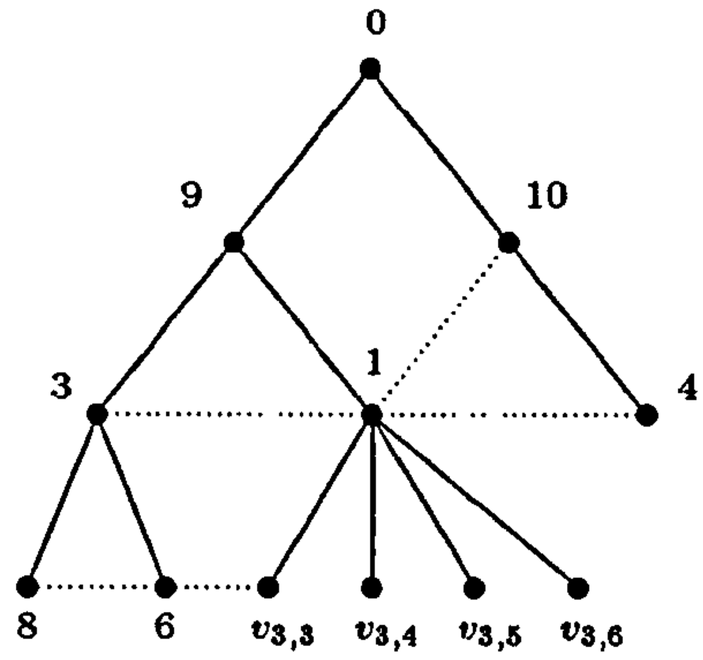
Let $v_{l-1,k}$ be any node, and assume that its children (in the OBFS tree) are $v_{l,i}, v_{l,i+1}, \ldots, v_{l,j}$. Now $v_{l-1,k}$ is also connected to its father, $v_{l-2,s}$ (unless it is the root) and it may also be connected to $v_{l,i-1}$, $v_{l-1,k-1}$, $v_{l-1,k+1}$ and $v_{l-2,s+1}$ in G.

We define $W_\Delta = (V, E)$ to be the outerplanar graph which is defined as follows (see [13]). $V = \{v_0, v_1, v_2, \ldots, v_\Delta\}$ and $E = \{v_0 v_i \mid 1 \leq i \leq \Delta\} \cup \{v_i v_{i+1} \mid 1 \leq i \leq \Delta - 1\}$. The following lemma is now needed.

Lemma 5.4.1 [13] If we label v_0 in W_Δ with any label from $\{0, 1, \ldots, \Delta+2\}$, then we can label the remaining vertices in W_Δ to give an $L(2,1)$-labelling of W_Δ with span at most $\Delta + 3$.

We will outline the proof of Theorem 5.4.2 below. After the proof we will give

Figure 5.2: This is a partial $L(2,1)$-labelling of the outerplanar graph G, depicted in Figure 5.1.



outerplanar graphs showing that the proof actually doesn't hold for $\Delta \in \{8,9\}$. We will then illustrate that the proof would need to include some extra conditions for it to be valid when $\Delta \in \{10,11\}$.

Theorem 5.4.2 [13] If G is an outerplanar graph with $\Delta(G) \geq 8$, then the $L(2,1)$-span is at most $\Delta(G)+3$.

Outline of the proof in [13]: We first find an OBFS tree of G. Using Lemma 5.4.1, we can label the first two levels with labels from $\{0,\ldots,\Delta+2\}$ (in fact we can do it with labels from $\{0,\ldots,\Delta+1\}$). This can be done as if we only consider the first two levels of our OBFS tree, then they induce a subgraph of $W_{\Delta(G)}$. We will now proceed by induction so assume that all vertices in levels $0,1,\ldots,l-1$ are already labelled as well as all vertices $v_{l,1}, v_{l,2}, \ldots, v_{l,i-1}$ where either $i=1$ or $v_{l,i-1}$ and $v_{l,i}$ have different parents in the OBFS tree. Let $v_{l,i}, v_{l,i+1}, \ldots, v_{l,j}$ be all the vertices in level l which have the same parent, $v_{l-1,k}$, as $v_{l,i}$ in the OBFS tree.

Using a slight generalisation of Lemma 5.4.1 it is now claimed that we can label all the vertices $v_{l,i}, v_{l,i+1}, \ldots, v_{l,j}$ with the labels $\{0,1,\ldots,\Delta(G)+2\}$, such that the $L(2,1)$ constraints on all labelled vertices are satisfied. The generalisation of Lemma 5.4.1 deals with the case when certain labels are forbidden (by previously labelled

vertices) and is the reason for the restriction $\Delta(G) \geq 8$. As we will illustrate that the above algorithm does not work immediately for $\Delta \leq 12$, we will not go into more detail of the algorithm here, and instead refer the interested reader to [13]. □

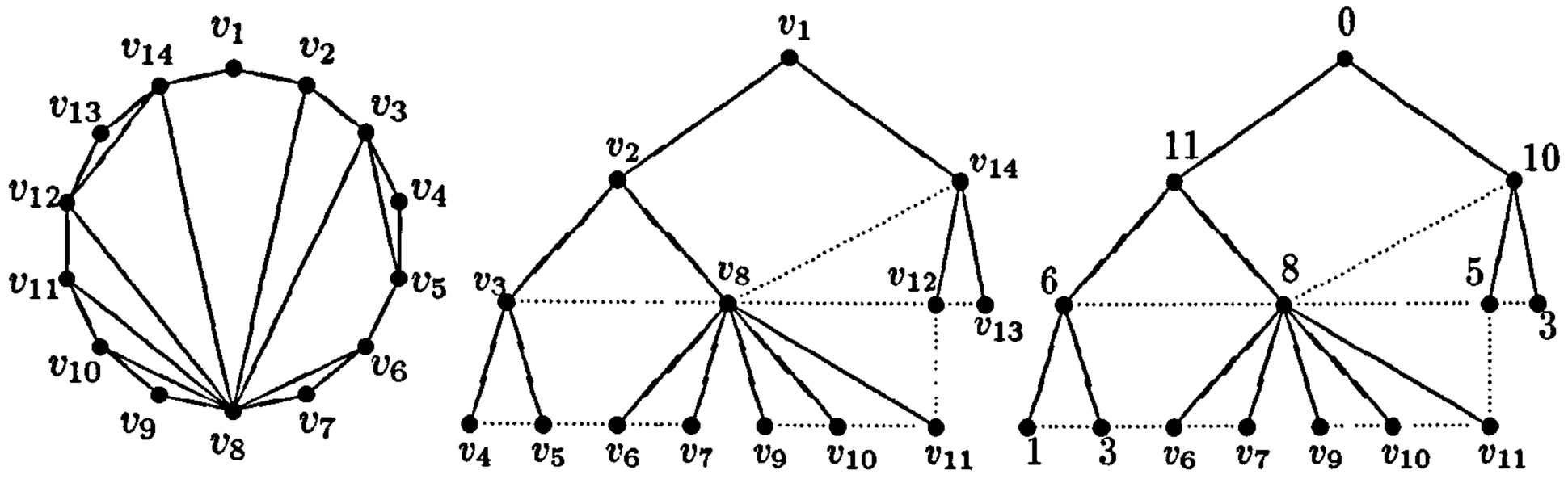
In Figure 5.2 we see an OBFS tree of the outerplanar graph G depicted in Figure 5.1. As $\Delta(G) = 8$ and we have only used labels from $\{0, 1, \ldots, 10\}$ so far, we should be able to label the remaining 4 vertices with labels from $\{0, 1, \ldots, 10\}$. However $v_{3,3}$ cannot receive any label from $\{0, 1, \ldots, 10\}$, a contradiction. Note that in the above example the vertices $v_{3,4}$, $v_{3,5}$ and $v_{3,6}$ are not used and could be deleted, if the example was part of some bigger OBFS tree, where there was a vertex of degree eight somewhere else in the outerplanar graph.

In fact we will show that the slightly larger outerplanar graph G depicted in Figure 5.3, with maximum degree nine, cannot be labelled with twelve labels using the approach in [13]. Its corresponding OBFS tree can be seen in Figure 5.3. If we have already partially labelled the vertices of G as seen in Figure 5.3, then we will show that the five non-labelled vertices cannot be labelled with labels from $\{0, 1, \ldots, 11\}$. Assume that we can label them with labels from $\{0, 1, \ldots, 11\}$. Note that the vertex v_6 can only receive label 0. Vertex v_7 can now receive label 2 or 4. First assume that it receives the label 4, which implies that the vertices v_9, v_{10} and v_{11} have to receive the labels $1, 2, 3$, which is impossible, as the vertex receiving label 2 will be adjacent to either 1 or 3. Therefore v_7 has to be labelled with 2. Now vertex v_{11} has to receive label 1, which implies that the labels 3 and 4 have to be assigned to v_9 and v_{10}, which is the desired contradiction.

With some additional work we could make the proof of Theorem 5.4.2 work for $\Delta \geq 10$. However we will not give the details here as this follows from Theorem 5.4.4, and the proof would be of comparable length to the proof of Theorem 5.4.4. The proof of Theorem 5.4.2 does not work in its current state for $\Delta \leq 11$, as the following example shows.

Consider the OBFS tree in Figure 5.4. It satisfies all the conditions given for an OBFS tree given in [13], however it is not an outerplanar graph. So to avoid such an example we would need to give extra conditions on an OBFS tree. We will do that

Figure 5.3: This is an example of an outerplanar graph, G, with its OBFS tree, and a partial $L(2,1)$-labelling of G. The vertices $v_6, v_7, v_9, v_{10}, v_{11}$ cannot be labelled with labels from $\{0, 1, \ldots, 11\}$, in order to complete an $L(2,1)$-labelling of G.



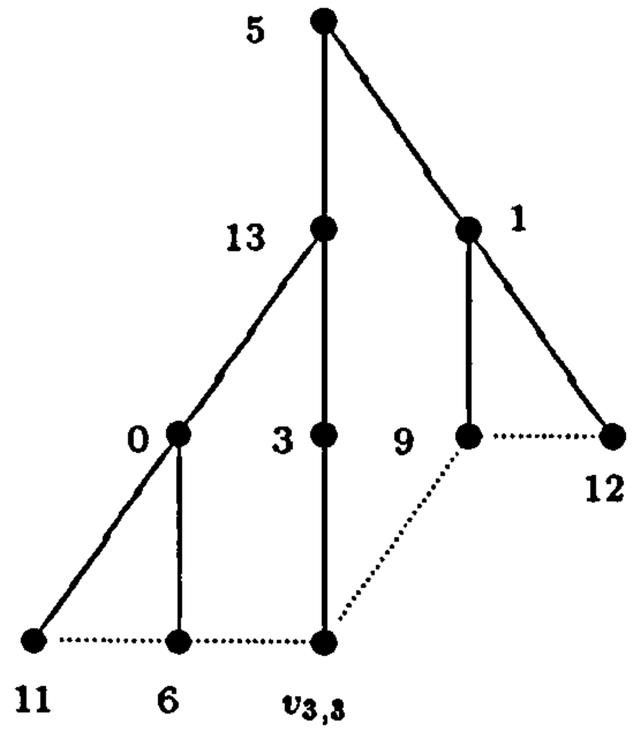
later in this section, however we will first show that if we do not forbid such OBFS trees, then the proof of Theorem 5.4.2 is not true for any $\Delta(G) \leq 11$. Imagine that we add vertices anywhere so that the maximum degree is eleven, and that the vertices are given the labels shown in Figure 5.4. Now note that $v_{3,3}$ cannot receive any label from $\{0, 1, \ldots, 13\}$, a contradiction. The proof in [13] is however correct for $\Delta(G) \geq 12$.

By adding the property below for an OBFS tree for an outerplanar graph, the proof in [13] can be made true for all $\Delta(G) \geq 10$, and we have shown an example with $\Delta = 9$, for which it does not work. We first need the following well known lemma (see for example exercise 20, page 91, in [16]).

Lemma 5.4.3 No outerplanar graph contains a $K_{3,2}$-minor.

Proof: First note that if G is an outerplanar graph and e is an edge of G then $G - \{e\}$ is clearly outerplanar. Suppose $(v_1, \ldots, v_n)$ is an outerplanar ordering of G. Now let $e = \{v_i, v_j\}$ where $i < j$. Then an outerplanar ordering of $G/\{e\}$ is given by $(v_1, \ldots, v_{i-1}, v^*, v_{i+1}, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n)$ where v^* is the new vertex formed by contracting $\{v_i, v_j\}$. Hence using induction we see that every minor of an outerplanar graph is outerplanar. Since $K_{3,2}$ is not outerplanar it follows that no

Figure 5.4: This is an OBFS tree which satisfy the OBFS-properties in [13]. It is partially labelled, in such a way that the vertex $v_{3,3}$ cannot receive any label from $\{0, 1, 2, \ldots, 13\}$. Note that the graph is not outerplanar.



outerplanar graph contains a $K_{3,2}$-minor. □

We now show that the situation in Figure 5.4 cannot occur. Let $v_{l-1,k}$ be any vertex in an OBFS, and assume that its children are $v_{l,i}, v_{l,i+1}, \ldots, v_{l,j}$. If the edge $v_{l,i-1}v_{l,i}$ and the edge $v_{l,j}v_{l-1,k+1}$ exist in G then there must be some $p \in \{i, i+1, \ldots, j-1\}$ such that the edge $v_{l,p}v_{l,p+1}$ does not exist in G. In order to prove this assume that it is false, and that $v_{l',r}$ is a vertex with maximum l' ($l' < l$) such that at least two of $v_{l-1,k-1}$, $v_{l-1,k}$ and $v_{l-1,k+1}$ are descendants of $v_{l',r}$ in the OBFS tree. Now note that there are three vertex disjoint paths from $v_{l',r}$ to $v_{l,i}$ (one using the edge $v_{l-1,k-1}v_{l,i-1}$, one using the edge $v_{l-1,k}v_{l,i}$ and one using the edge $v_{l-1,k+1}v_{l,j}$), all having length at least two. However this implies that G has a $K_{3,2}$ minor, which is impossible by Lemma 5.4.3.

We note that the above condition forbids OBFS trees as in Figure 5.4. However the example in Figure 5.3 still shows that the proof of Theorem 5.4.2 does not work for $\Delta(G) = 9$.

Consequently we move on to the next theorem which generalises Theorem 5.4.2 as the proof of Theorem 5.4.2 definitely does not hold for $\Delta(G) \leq 9$. Before we state

the theorem we need a few more definitions. Some of these will not be needed until Section 5.5. A *partial* $L(2,1)$*-labelling* of a graph G is, for some $A \subseteq V$, a function $L : A \to \mathbb{Z}^+$, such that whenever u and v are adjacent we have $|L(u) - L(v)| \geq 2$ and whenever u and v have a common neighbour in G then $L(u) \neq L(v)$. Note that if the only common neighbours of u and v are in $V \setminus A$, we still require $L(u) \neq L(v)$. A partial σ-$L(2,1)$-*labelling* is defined analogously.

In the following we may drop the $L(2,1)$ and just refer to a labelling or a partial labelling. We will also drop the partial when the context is clear. Given partial labellings $L_1 : A_1 \to \mathbb{Z}^+$ and $L_2 : A_2 \to \mathbb{Z}^+$, we say $L_1 \subseteq L_2$ if $A_1 \subseteq A_2$ and for all $v \in A_1$, we have $L_1(v) = L_2(v)$. If $L_1 \subseteq L_2$ we say that L_1 may be extended to L_2 and that L_2 extends L_1. If we have partial labellings $L_1 : A_1 \to \mathbb{Z}^+$ and $L_2 : A_2 \to \mathbb{Z}^+$ such that for all $v \in A_1 \cap A_2$ we have $L_1(v) = L_2(v)$ then the partial labelling $L_1 \cup L_2 : A_1 \cup A_2 \to \mathbb{Z}^+$ is given by $(L_1 \cup L_2)(v) = L_1(v)$ if $v \in A_1$ and $(L_1 \cup L_2)(v) = L_2(v)$ if $v \in A_2$. If $L : A_1 \to \mathbb{Z}^+$ is a partial labelling and $A_2 \subseteq A_1$ then $L|A_2$ is the partial labelling obtained by restricting the domain of L to A_2. We now introduce the corresponding concepts for the version of the problem where we use cyclic constraints. Given a graph, G, a *cyclic-*σ*-*$L(2,1)$*-labelling* of G is a function, $L : V(G) \to \{0, 1, \ldots, \sigma - 1\}$ such that if u and v are adjacent, $|f(u) - f(v)|_\sigma \geq 2$ and if u and v have a common neighbour then $f(u) \neq f(v)$. The *cyclic-*$L(2,1)$*-span* of a graph G is the minimum σ for which there exists a cyclic-σ-$L(2,1)$-labelling of G.

A partial cyclic-σ-$L(2,1)$ labelling is defined in the obvious way and all the definitions involving partial labellings can be carried over to the version of the problem involving cyclic labellings. In the following we will often omit the σ when it is clear from the context.

In order to explain the main idea behind the next theorem we first define $i^*(1)$ as follows. Let $(v_1, v_2, \ldots, v_n)$ be an outerplanar-ordering of an outerplanar graph G where all indices are taken modulo n. Let $N(v_2) - \{v_1\} = \{v_{a_1}, v_{a_2}, \ldots, v_{a_A}\}$, where $a_1 < a_2 < \ldots < a_A$, and let $N(v_1) - \{v_2\} = \{v_{b_1}, v_{b_2}, \ldots, v_{b_B}\}$, where $b_1 < b_2 < \ldots < b_B$. If $A > 0$ then let $i^*(1) = a_A$, otherwise if $B > 0$ then let

$i^*(1) = b_1$, and if $A = B = 0$, then let $i^*(1) = 3$. Before we show why this is a useful parameter, we note that the "1" in $i^*(1)$ indicates that we are considering v_1 (and v_2), when determining $i^*(1)$. We could also define $i^*(j)$ for any $j = 1, 2, \ldots, n$ in an analogous way, but since we do not need it below, we will not go through the details of this. The reason why $i^*(1)$ is an important parameter, is the following. Let L be any labelling of $N[v_1, v_2, v_{i^*(1)}]$, such that L is a partial labelling of G. Let $G_1 = G : \{v_2, v_3, \ldots, v_{i^*(1)}\}$ and let $G_2 = G : \{v_{i^*(1)}, v_{i^*(1)+1}, \ldots, v_n, v_1\}$. Let L_1 be any $L(2,1)$-labelling of G_1, and let L_2 be any $L(2,1)$-labelling of G_2, such that $L_1 \cup L_2$ has the same labels as L on $N[v_1, v_2, v_{i^*(1)}]$. Then $L_1 \cup L_2$ is an $L(2,1)$-labelling of G. The reason for this is that there are no edges joining a vertex in $V(G_1) - \{v_2, v_{i^*(1)}\}$ to a vertex in $V(G_2) - \{v_1, v_{i^*(1)}\}$ in G. Furthermore all paths of length two from $V(G_1) - \{v_2, v_{i^*(1)}\}$ to $V(G_2) - \{v_1, v_{i^*(1)}\}$ in G must pass through v_1, v_2 or $v_{i^*(1)}$. See Figure 5.5 for an example.

Theorem 5.4.4 If G is an outerplanar graph, with $\Delta(G) \geq 3$, then the *cyclic* $-$ $L(2,1)$-span is at most $\Delta(G)+8$. Furthermore if $\Delta(G) \geq 9$, then the *cyclic*$-L(2,1)$-span equals $\Delta(G) + 3$.

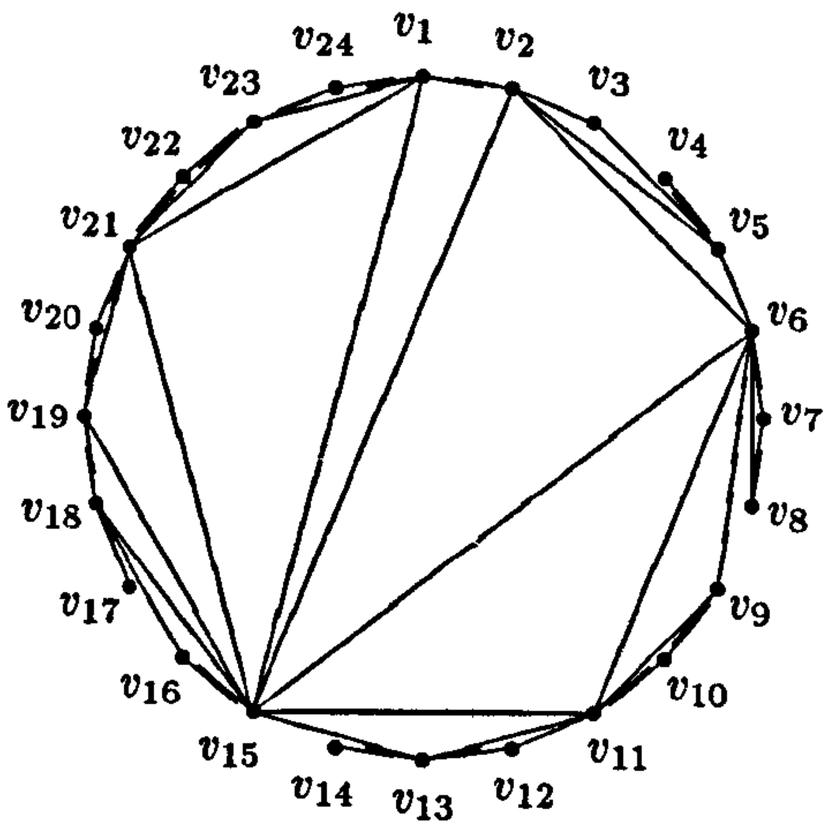
Proof: First note that providing $\Delta(G) > 0$, $\Delta(G) + 3$ is a lower bound. This follows by considering a vertex of degree $\Delta(G)$ and its neighbours.

We first consider the case when $\Delta(G) \geq 9$. Let $(v_1, v_2, \ldots, v_n)$ be an outerplanar ordering of the vertices of G. We will show that if we can construct a labelling L of $N[v_1, v_2]$ such that L is a partial-cyclic-$(\Delta + 3)$-$L(2,1)$-labelling of G then L may be extended to a cyclic-$(\Delta + 3)$-$L(2,1)$-labelling of G.

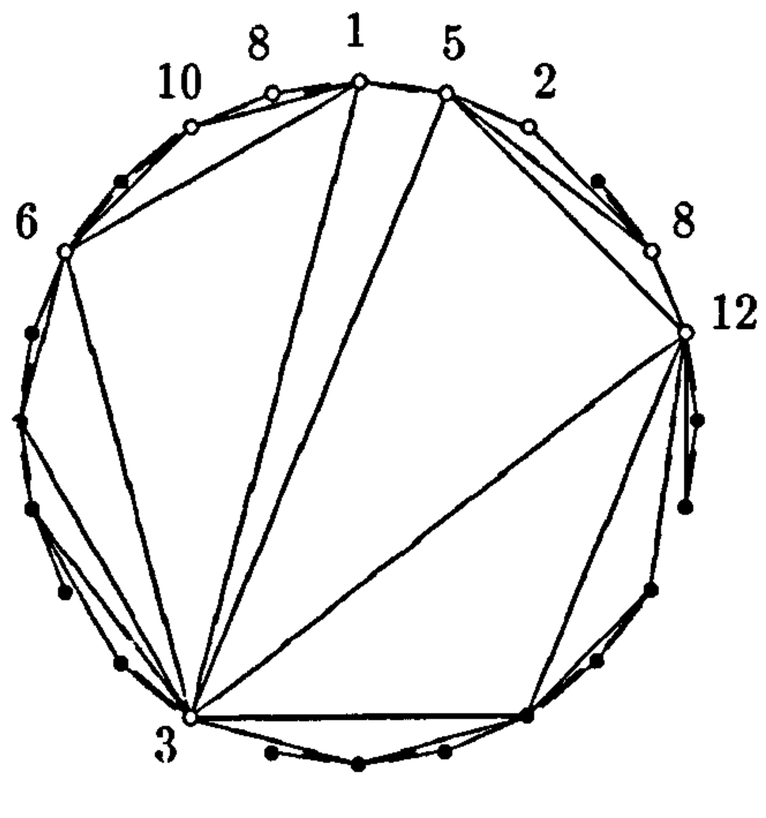
Let $i = i^*(1)$ as defined before the theorem. We will now show that we may extend L to a labelling L' of $N[v_1, v_2, v_i]$ so that L' is a partial-cyclic-$(\Delta + 3)$-$L(2,1)$-labelling of G. By induction we can then extend L' to labellings L_1 and L_2 of $G_1 = G : \{v_2, \ldots, v_i\}$ and $G_2 = G : \{v_i, \ldots, v_n, v_1\}$. Then by the observations before the theorem, these two labellings can be put together to give us a cyclic-$(\Delta + 3)$-$L(2,1)$-labelling of G.

Let $N(v_i) - \{v_1, v_2\} = \{v_{i_1}, v_{i_2}, \ldots, v_{i_r}\}$, such that $2 < i_1 < i_2 < \ldots < i_r \leq n$. If

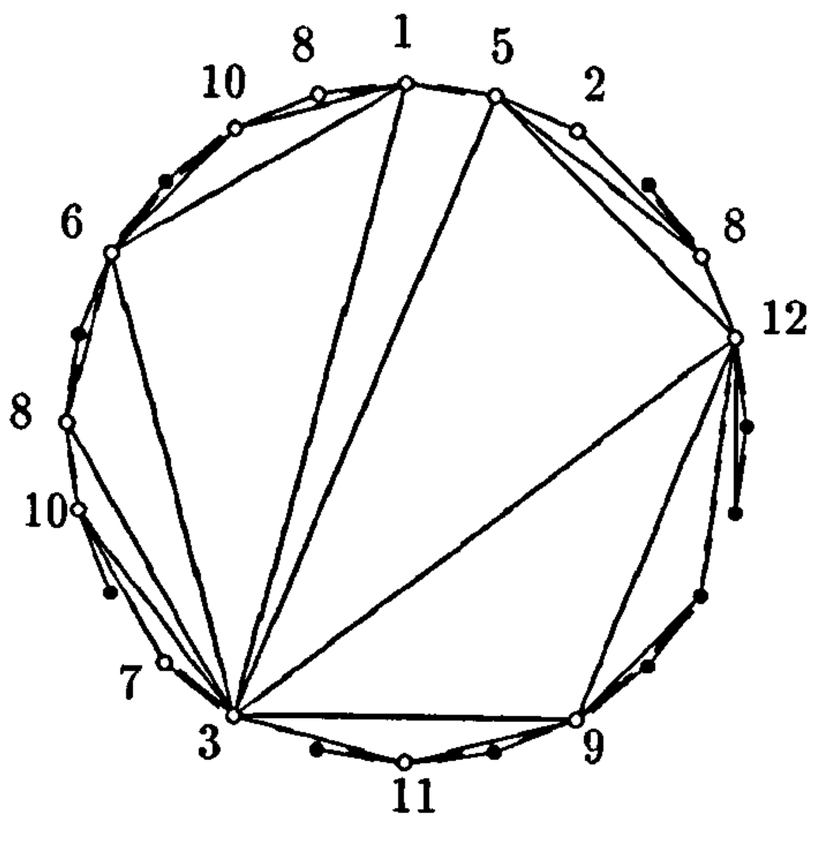
Figure 5.5: In each picture the solid vertices are yet to be labelled. Note from the first picture that $i^*(1) = 15$. In the second picture we have labelled $N[v_1, v_2]$. In the third picture we have labelled $N[v_1, v_2, v_{15}]$. The last picture shows the graphs $G_2 = G : \{v_{i^*(1)}, v_{i^*(1)+1}, \ldots, v_n, v_1\}$ and $G_1 = G : \{v_2, v_3, \ldots, v_{i^*(1)}\}$.



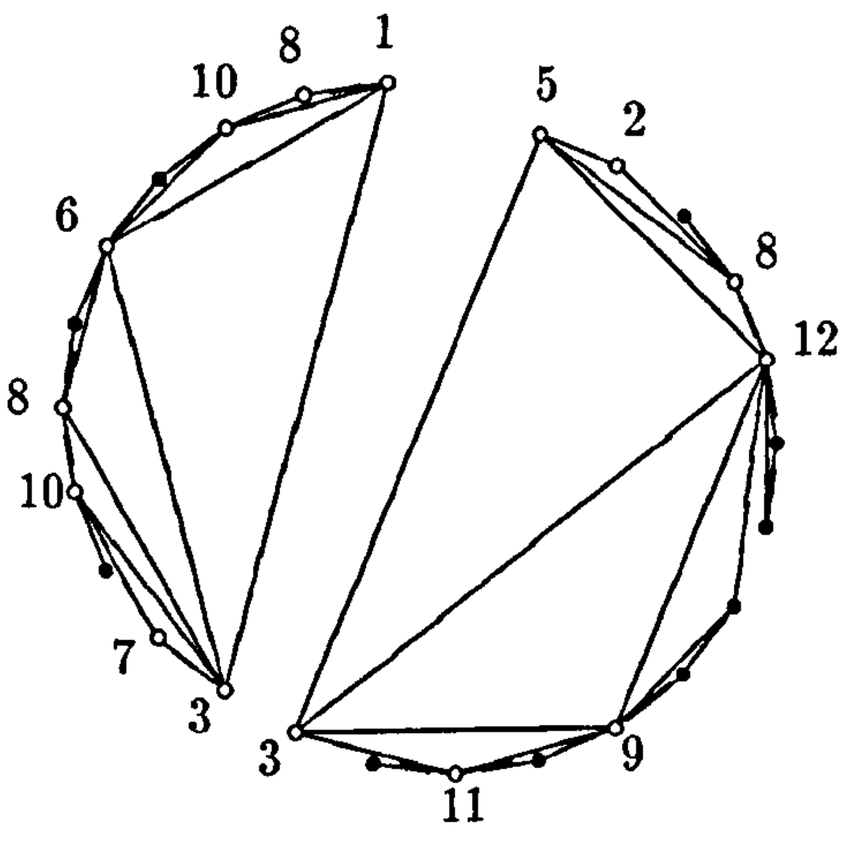
Picture 1



Picture 2



Picture 3



Picture 4

v_i is not already labelled (which implies that $i = 3$) then label it by any label. This is possible as in this case only v_1 and v_2 are labelled and there is no edge from $\{v_1, v_2\}$ to $V - \{v_1, v_2\}$. If $c(v_i)$ denotes the label of v_i then let $X = \{c(v_i)-1, c(v_i), c(v_i)+1\}$ where these labels are taken modulo $\Delta(G)+3$. If $v_1 \in N(v_i)$ then add $c(v_1)$ to X and if $v_2 \in N(v_i)$ then add $c(v_2)$ to X. Now let $Col = \{0, 1, \ldots, \Delta(G)+2\} - X$. Note that $|Col| \geq r$ and $|Col| \geq 7$. We will now show how to label $v_{i_1}, v_{i_2}, \ldots, v_{i_r}$ with labels from Col, such that all cyclic-$L(2,1)$-constraints hold (note that v_{i_1} and/or v_{i_r} may already be labelled).

First consider the case when $r \geq 7$. Now greedily label $v_{i_r}, v_{i_{r-1}}, \ldots, v_{i_8}$, in that order, such that all constraints hold. It is not difficult to see that there are enough labels available in each step. Now let Col denote the remaining labels, which are left after deleting the labels we have just used from the set. Note that $|Col| \geq 7$. If v_{i_1} is not labelled and/or if v_{i_7} is not labelled, then label these. This can still be done. We now need to label $v_{i_2}, v_{i_3}, \ldots, v_{i_6}$, such that no two adjacent vertices in the sequence $v_{i_1}, v_{i_2}, \ldots, v_{i_7}$ get adjacent labels and all the labels used for $v_{i_1}, v_{i_2}, \ldots, v_{i_7}$ are distinct and lie in Col. Note that v_{i_1} may be adjacent to a vertex with a label in Col (this vertex would be adjacent to v_2), so v_{i_2} cannot receive this label. But there is only one such forbidden label at v_{i_2}. Analogously there may be one forbidden label at v_{i_6}.

Define the labels $h_1, h_2, \ldots, h_7$ so that $\{h_1, h_2, \ldots, h_7\} \subseteq Col$, $h_1 = c(v_{i_1})$ and $h_1 < h_2 < \ldots < h_s$ and $h_{s+1} < h_{s+2} < \ldots < h_7 < h_1$. We can now look up in the table below to see how to label $v_{i_2}, v_{i_3}, \ldots, v_{i_6}$, by doing the following. Consider the entry $(1, a)$ if $c(v_{i_7}) = h_a$. Since at most one extra label is forbidden at v_{i_2} and at most one label is forbidden at v_{i_6}, we can pick one of the three options for $(1, a)$, as in the following example. Let $Col = \{3, 5, 6, 7, 9, 10, 11\}$, and $c(v_{i_1}) = 7 = h_1$ which means that $h_1 = 7$, $h_2 = 9$, $h_3 = 10$, $h_4 = 11$, $h_5 = 3$, $h_6 = 5$ and $h_7 = 6$. Assume that $c(v_{i_7}) = 5 = h_6$, and that we may not use labels $10 = h_3$ at either v_{i_2} or v_{i_6}. Then we use the sequence $1, 5, 3, 7, 4, 2, 6$ as it starts with 1, ends with 6 and does not have 3 at the second or sixth place. This sequence tells us to label the vertices as follows: $c(v_{i_2}) = h_5 = 3$, $c(v_{i_3}) = h_3 = 10$, $c(v_{i_4}) = h_7 = 6$, $c(v_{i_5}) = h_4 = 11$ and

$c(v_{i_6}) = h_2 = 6$. This always gives the desired labelling. We now need to consider the case when $r < 7$.

(1,7)	1,3,5,2,6,4,7 1,4,6,2,5,3,7 1,5,3,6,4,2,7	(1,6)	1,3,5,2,7,4,6 1,4,7,2,5,3,6 1,5,3,7,4,2,6
(1,5)	1,4,7,2,6,3,5 1,6,3,7,4,2,5 1,3,6,2,7,4,5	(1,4)	1,3,6,2,5,7,4 1,5,2,7,3,6,4 1,6,3,5,7,2,4
(1,3)	1,4,6,2,7,5,3 1,5,7,4,2,6,3 1,6,4,2,5,7,3	(1,2)	1,3,6,4,7,5,2 1,4,7,5,3,6,2 1,5,3,6,4,7,2

Assume that $r < 7$. Assume that $i_1 < i_2 < \ldots < i_t < i < i_{t+1} < i_{t+2} < \ldots < i_r$, where $0 \leq t \leq r$. Note that if $0 < t < r$, then we may use adjacent labels on v_{i_t} and $v_{i_{t+1}}$. If $t = 0$, then note that we have no restrictions on v_{i_1}, and if $t = r$, then we have no restrictions on v_{i_r}. If $r \leq 2$, then we can easily label the vertices greedily, so assume that $r \geq 3$. As in the case when $r \geq 7$, we consider the three options under $(1, a)$ in the above table, and proceed as is illustrated in the following example.

As before let Col contain the labels $h_1, h_2, \ldots, h_7$, such that $c(v_{i_1}) = h_1$ and $h_1 < h_2 < \ldots < h_s$ and $h_{s+1} < h_{s+2} < \ldots < h_7 < h_1$. Consider the entry $(1, a)$ if $c(v_{i_r}) = h_a$. Since at most one extra label is forbidden at v_{i_2} and at most one label is forbidden at $v_{i_{r-1}}$, we can pick one of the three options for $(1, a)$, as in the following example. Let $Col = \{3, 5, 6, 7, 9, 10, 11\}$, and $c(v_{i_1}) = 7 = h_1$ which means that $h_1 = 7$, $h_2 = 9$, $h_3 = 10$, $h_4 = 11$, $h_5 = 3$, $h_6 = 5$ and $h_7 = 6$. Assume that $c(v_{i_r}) = 5 = h_6$, and that we may not use labels $10 = h_3$ at either v_{i_2} or $v_{i_{r-1}}$. Then we use the sequence $1, 5, 3, 7, 4, 2, 6$ as it starts with 1, ends with 6 and does not have 3 at the second or sixth place. This sequence tells us to label the vertices as follows: $c(v_{i_2}) = h_5 = 3$, $c(v_{i_3}) = h_3 = 10$, $c(v_{i_4}) = h_7 = 6$,... until we have labelled v_{i_t}. Then we let $c(v_{i_{r-1}}) = h_2 = 9$, $c(v_{i_{r-2}}) = h_4 = 11$,... until we have labelled $v_{i_{t+1}}$. We note that the vertices v_{i_t} and $v_{i_{t+1}}$ may get adjacent labels, but that this is allowed due to the arguments above.

When $3 \leq \Delta \leq 8$, the proof follows similar lines. We take a labelling of $N[v_1, v_2]$ and extend it as before to a labelling of $N[v_1, v_2, v_i]$ so that all cyclic-$L(2,1)$ constraints hold. Choose i and define r and Col as before. Whenever $|Col| \geq 7$, the proof above works. If $|Col| < 7$ then $\Delta \leq 3$ and case-checking shows that the labelling of $N[v_1, v_2]$ may always be extended to a labelling of $N[v_1, v_2, v_i]$ so that all the cyclic-$L(2,1)$ constraints hold. □

Since a cyclic-σ-$L(2,1)$ labelling is always a (non-cyclic) σ-$L(2,1)$-labelling the following result concerning the non-cyclic $L(2,1)$ span follows immediately.

Corollary 5.4.5 If G is an outerplanar graph with $\Delta(G) \geq 3$, then the $L(2,1)$-span is at most $\Delta(G) + 8$. Furthermore if $\Delta(G) \geq 9$, then the $L(2,1)$-span is at most $\Delta(G) + 3$. The $L(2,1)$-span is always at least $\Delta(G) + 2$.

5.5 A polynomial time algorithm to find the $L(2,1)$-span for outerplanar graphs

The following results are joint work with Steve Noble, Brunel University, and Anders Yeo, Royal Holloway.

Motivated by Section 5.4 we will now show that it is possible in polynomial time to determine both the cyclic and non-cyclic $L(2,1)$-spans of an outerplanar graph G in polynomial time. This clearly generalises the same results for trees, which was stated as an open problem, and even conjectured to be NP-hard, before it was shown to be polynomial in [14]. The problem for outerplanar graphs was stated in [11] by Bodlaender et al.

The fact that the problem is NP-hard for planar graphs (see [11]), also makes our result for outerplanar graphs more interesting. Our polynomial time algorithm is quite long and technical.

We begin by describing a recursive algorithm to compute the $L(2,1)$-span of an outerplanar graph G that runs in polynomial time for certain values of $\Delta(G)$ and n. Later we explain how the algorithm may be modified so that it always runs in

polynomial time.

Lemma 5.5.1 Let G be an outerplanar graph. There exists an algorithm, which finds the $L(2,1)$-span, in $O(n\Delta(G)^{10\Delta(G)})$ time.

Proof: By Corollary 5.4.5 and the remark in the beginning of section 5.2 we note that the $L(2,1)$-span is $\Delta(G)+k$ for some $k \in \{2,3,\ldots,8\}$. Let $\sigma = \Delta(G)+k$, where $1 < k < 9$ is arbitrary. Let $v_1, v_2, \ldots, v_n$ be an outerplanar-ordering of the vertices of G. We will now give an algorithm that decides in $O(n\Delta(G)^{10\Delta(G)})$-time if there exists a σ-$L(2,1)$-labelling of G. This would prove the theorem as we only have to run the algorithm for 7 different values of k to determine the $L(2,1)$-span.

Our algorithm will find all possible labellings of $N[v_1, v_2]$ that can be extended to a σ-$L(2,1)$-labelling of G. If there exists no such labellings of $N[v_1, v_2]$, then clearly there exists no σ-$L(2,1)$-labelling of G and otherwise there does. Our algorithm is recursive, and will produce a (possibly empty) list of labellings of $N[v_1, v_2]$.

If $|V(G)| \leq 2$ then we just return all possible labellings of G. Otherwise let $i = i^*(1)$ (see the definitions of $i^*(1)$ above Theorem 5.4.4). Now recursively solve the problem for the two graphs $G_1 = G : \{v_i, v_2, v_3, \ldots, v_{i-1}\}$ and $G_2 = G : \{v_1, v_i, v_{i+1}, \ldots, v_n\}$. In G_1, we think of v_i as playing the same role as v_1 does in G, so in other words we look for labellings of $N[v_i, v_2]$ that may be extended to G_1. Similarly, in G_2, we think of v_i as playing the same role as v_2 does in G. Let $\mathcal{L}_1$ contain all possible labellings of $N_{G_1}[v_2, v_i]$ that can be extended to a σ-$L(2,1)$-labelling of G_1, and let $\mathcal{L}_2$ contain all possible labellings of $N_{G_2}[v_1, v_i]$ that can be extended to a σ-$L(2,1)$-labelling of G_2.

We now construct a list $\mathcal{L}$ of labellings of $N_G[v_1, v_2]$ by considering all pairs of labellings consisting of one from $\mathcal{L}_1$ and one from $\mathcal{L}_2$. Let $\mathcal{L}$ be empty initially.

If $L_1 \in \mathcal{L}_1$ and $L_2 \in \mathcal{L}_2$, then we can check (in $O(|N_{G_1}[v_2, v_i]| \cdot |N_{G_2}[v_1, v_i]|)$ time) whether $L = L_1 \cup L_2$ is a partial-$L(2,1)$-labelling of G. If it is then add the restriction of L to $N[v_1, v_2]$ into our list $\mathcal{L}$. Once we have tried all pairs L_1 and L_2, we stop and return $\mathcal{L}$.

We will now prove that the algorithm works. If L is an $L(2,1)$-labelling of G,

then $L|N[v_1, v_2]$, will belong to our list $\mathcal{L}$. This is easily proved using induction on our recursive step. Similarly if some labelling L belongs to our list $\mathcal{L}$, then this can be extended to an $L(2,1)$-labelling of G. This is also proved by induction since the pair of labellings L_1 and L_2 that caused us to add L to our list $\mathcal{L}$ can be extended to labellings L_1' and L_2' of G_1 and G_2 respectively. Now $L_1' \cup L_2'$ extends L. Therefore the algorithm works correctly.

We now compute the complexity of the algorithm. The maximum time that our algorithm uses on an outerplanar graph with n vertices and maximum degree less than or equal to Δ, will be denoted by $F(n, \Delta)$. Note that $|\mathcal{L}_1| \leq \sigma^{2\Delta+2}$, as $|N_{G_1}[v_2, v_i]| \leq 2\Delta + 2$. Similarly $|\mathcal{L}_2| \leq \sigma^{2\Delta+2}$. As we need at most $O((2\Delta+2)^2)$ time to check if $L_1 \cup L_2$ is an $L(2,1)$-labelling of $G : \{N[v_1, v_2, v_i]\}$, this implies the following for some constant q and the i found above (note that $3 \leq i \leq n$).

$$\begin{aligned} F(n,\Delta) &\leq F(i-1,\Delta) + F(n+2-i,\Delta) + q \cdot (2\Delta+2)^2 \cdot (\sigma^{2\Delta+2})^2 \\ &\leq F(i-1,\Delta) + F(n+2-i,\Delta) + q \cdot (2\Delta+2)^2 \cdot (\Delta+k)^{4\Delta+4}. \end{aligned}$$

As $1 < k < 9$, and $F(j, \Delta)$ is bounded by a constant, when $j \leq 4$, we have

$$F(n,\Delta) \in O(n \cdot (2\Delta+2)^2 \cdot (\Delta+8)^{4\Delta+4}).$$

If $\Delta(G) < 8$, then this implies that $F(n, \Delta) \in O(n)$, so assume that $\Delta(G) \geq 8$, in which case we have

$$F(n,\Delta) \in O(n \cdot 9(\Delta)^2 \cdot (2\Delta)^{4.5\Delta}) \subseteq O(n \cdot (\Delta^2)^{4.5\Delta+1}) \subseteq O(n \cdot \Delta^{10\Delta}).$$

□

Clearly our algorithm runs in polynomial time when Δ is bounded by a constant and it is easy to modify the algorithm to deal with the version of the problem with cyclic constraints. This gives us the following result

Theorem 5.5.2 There is a polynomial time algorithm to determine the cyclic-$L(2,1)$-span for outerplanar graphs.

Proof: Given an outerplanar graph G, if $\Delta(G) \geq 9$ then from Theorem 5.4.4, the cyclic-$L(2,1)$-span is $\Delta(G) + 3$. Otherwise we use the algorithm described in Lemma 5.5.1 suitably modified to deal with cyclic constraints. □

From now on we deal only with the non-cyclic version of the constraints. The next result gives us a class of outerplanar graphs for which the algorithm in Lemma 5.5.1 runs in polynomial time.

Corollary 5.5.3 Let G be an outerplanar graph, with $\Delta \leq 1000$ or with $\frac{\ln(n)}{\ln(\Delta)} \geq 0.01\Delta$, then there exists a polynomial time algorithm to find the $L(2,1)$-span.

Proof: If $\Delta \leq 1000$, then Lemma 5.5.1 gives us an $O(n)$ and hence polynomial time algorithm. If $\ln(n)/\ln(\Delta) \geq 0.01\Delta$, then we see that

$$n \geq \Delta^{0.01\Delta}$$

and so

$$n^{1000} \geq \Delta^{10\Delta}.$$

Therefore Lemma 5.5.1 gives us an $O(n^{1001})$ and hence polynomial time algorithm. □

The remainder of this section is devoted to the case not covered by Corollary 5.5.3. As the proofs get quite technical we will first give a rough outline of how we prove that there exists a polynomial algorithm to decide the $L(2,1)$-span of an outerplanar graph G, when $\Delta > 1000$ and $\frac{\ln(n)}{\ln(\Delta)} < 0.01\Delta$.

Lemma 5.5.4 is a very useful lemma, as it tells us when we can complete a partial-$L(2,1)$-labelling of $N[x]$, where x is some vertex in an outerplanar graph. In fact Lemma 5.5.4 is slightly stronger as it considers any subgraph of a wheel (note that $N[x]$ is always a subgraph of a wheel, with x being the central vertex in the wheel).

Definition 5.5.5 defines $F(m)$ and an integer k, which will be used later. Lemma 5.5.6 then proves some properties of $F(m)$ and k. In Definition 5.5.7 we define what an $OBFS_2$ tree is. Technically speaking an $OBFS_2$ tree is two disjoint trees

obtained by adding a vertex v^* to our outerplanar graph, then finding the $OBFS$ tree, and then deleting v^* again. We use the $OBFS_2$ tree heavily in Lemma 5.5.11, which is the main structural result. Before Lemma 5.5.11, we will give another definition and a lemma, which turn out to be useful for Lemma 5.5.11.

Lemma 5.5.11 states that if there exists a $(\Delta + 2)$-$L(2,1)$-labelling, L, of an outerplanar graph G, then the following holds. Assume we have a colouring of a number of layers in the $OBFS_2$ tree, L', such that $L(x) = L'(x)$, when L contains values which are either small or big. Then L' can be extended to a $(\Delta + 2)$-$L(2,1)$-labelling of G. In order to define what 'big' and 'small' is we use $F(m)$ and k which we mentioned earlier, as well as some technical definitions given in Definition 5.5.8.

Corollary 5.5.12 follows quite easily from Lemma 5.5.11, and is the main result used in the proof of our main theorem. Corollary 5.5.12 states that if there exists a $(\Delta + 2)$-$L(2,1)$-labelling, L, of an outerplanar graph G with outerplanar ordering $(v_1, v_2, \ldots, v_n)$ then the following holds. Let $H = \{k-1, k, \ldots, \Delta + 2 - k\}$, where k was defined in Definition 5.5.5. Let L' be a labelling of $N[v_1, v_2]$, such that L' is a partial-$(\Delta+2)$-$L(2,1)$-labelling of G and for all $j = 1, 2, \ldots, l$, either $L(v_{i_j}) = L'(v_{i_j})$ or $L(v_{i_j}), L'(v_{i_j}) \subseteq H$. Then L' can be extended to a $(\Delta + 2)$-$L(2,1)$-labelling of G (under certain conditions). Informally this means that all labels in H can be changed in $N[v_1, v_2]$, and we can still find a $(\Delta + 2)$-$L(2,1)$-labelling of G. We use this result several times when proving our main result, Theorem 5.5.13.

In Theorem 5.5.13 we finally prove that the desired polynomial algorithm exists. We do this by using a similar approach as we did in Lemma 5.5.1. However it would take too long time to record all $L(2,1)$-labellings of $N[v_1, v_2]$ which can be extended to a $(\Delta + 2)$-$L(2,1)$-labelling of G, but if we treat all labels in H (defined above) as identical, then we show that the time complexity becomes polynomial. We can then use Corollary 5.5.12 to show that the list of labellings we obtain can actually be extended to a proper $(\Delta + 2)$-$L(2,1)$-labelling of G, where the labels in H are actually given (and not considered to be identical). Furthermore if there does exist a proper $(\Delta+2)$-$L(2,1)$-labelling of G then our algorithm will produce a non-empty list of labellings of $N[v_1, v_2]$.

This was the main idea behind our algorithm. We have of course left out many details, as the proof is extremely long and technical. However, hopefully it has given an idea of how the proof will progress. We are now ready to prove our results.

Lemma 5.5.4 Let G be a graph with $V(G) = \{x, v_1, v_2, \ldots, v_k\}$. Suppose that $N[x] = V(G)$ and that $N(v_i) \subseteq \{x, v_{i-1}, v_{i+1}\}$ for all $i = 1, 2 \ldots, k$, where all indices are taken modulo k. Let $X \subset V(G) - \{x\}$ be arbitrary. Let L be an $L(2,1)$-labelling of $G - X$, and let H be any set of labels, which are not used in L and which differ by at least two from $L(x)$. If $|H| \geq |X|$ and $|H| \geq 22$, then L can be extended to an $L(2,1)$-labelling of G, by using only labels from H.

Proof: First assume that $|X| = |H|$. Suppose we label the vertices of X with labels from H one by one. At any point each vertex $v_i \in X$ may be labelled with any label in H that has not yet been used, except at most 4, namely the labels that differ by one from the labels of v_{i-1} and v_{i+1}.

Furthermore note that at any point, each label in H that has not yet been used, may be given to at least $|X| - 4$ vertices of X (it is only forbidden on each of the two neighbours of the vertices with labels differing by one from the label we are considering).

A *maximal non-labelled interval* is an interval $\{v_i, v_{i+1}, \ldots, v_l\} \subseteq X$ (indices taken modulo k), such that $v_{i-1} \notin X$ and $v_{l+1} \notin X$. As $|X| = |H| \geq 22$, either Case 1 or Case 2 below holds.

Case 1. There exist at least 8 *distinct maximal non-labelled intervals.* We can easily greedily label all vertices, except exactly one vertex in each of 8 distinct maximal non-labelled intervals. Now consider the bipartite graph, where one partite set, X', consists of the 8 unlabelled vertices, and the other partite set, H', consists of those labels, from H, which have not yet been used. The edge from $x \in X'$ to $h \in H'$ is present if it is permissible for x to receive the label h. It follows from the two observations at the beginning of the proof, that each vertex in X' has degree at least $|H'| - 4$, each vertex in H' has degree at least $|X'| - 4$ and furthermore $|H'| = |X'| = 8$. This implies (by Hall's Theorem) that there is a perfect matching,

which gives us the desired extension to an $L(2,1)$-labelling of G.

Case 2. Some maximal non-labelled interval has size at least 4. We can easily construct a labelling L' by greedily labelling all vertices, such that we are left with exactly one maximal non-labelled interval of size exactly 4. Assume without loss of generality that the non-labelled vertices are $\{v_2, v_3, v_4, v_5\}$ and let H' denote any four labels that are not used. Let $H' \cup \{L'(v_1), L'(v_6)\} = \{h_1, h_2, h_3, h_4, h_5, h_6\}$, such that $h_1 < h_2 < h_3 < h_4 < h_5 < h_6$.

If $L'(v_1) < L'(v_6)$, then we label $\{v_2, v_3, v_4, v_5\}$, using the following table, and if $L'(v_1) > L'(v_6)$, then we just reverse the labels in the table below.

If $(L'(v_1), L'(v_6)) =$	Then label v_2, v_3, v_4, v_5 with
(h_1, h_2)	h_4, h_6, h_3, h_5
(h_1, h_3)	h_4, h_6, h_2, h_5
(h_1, h_4)	h_3, h_5, h_2, h_6
(h_1, h_5)	h_4, h_2, h_6, h_3
(h_1, h_6)	h_3, h_5, h_2, h_4
(h_3, h_4)	h_1, h_5, h_2, h_6
(h_3, h_5)	h_1, h_6, h_4, h_2
(h_3, h_6)	h_1, h_5, h_2, h_4

If $(L'(v_1), L'(v_6)) =$	Then label v_2, v_3, v_4, v_5 with
(h_2, h_3)	h_4, h_6, h_1, h_5
(h_2, h_4)	h_5, h_3, h_1, h_6
(h_2, h_5)	h_4, h_1, h_6, h_3
(h_2, h_6)	h_4, h_1, h_5, h_3
(h_4, h_5)	h_2, h_6, h_1, h_3
(h_4, h_6)	h_2, h_5, h_1, h_3
(h_5, h_6)	h_2, h_4, h_1, h_3

This completes the case when $|X| = |H|$, so now assume that $|H| > |X|$. Note that if there exist at least 8 distinct maximal non-labelled intervals, then we are still done, by an analogous argument to that in Case 1 above. If there exists some maximal non-labelled interval with size at least 3, then we proceed as above in Case 2 except at the end we have at least 4 available labels, to label 3 vertices. Now use the table below, just as we did in Case 2.

If $(L'(v_1), L'(v_5)) =$	Then label v_2, v_3, v_4 with
(h_1, h_2)	h_5, h_3, h_6
(h_1, h_3)	h_5, h_3, h_6
(h_1, h_4)	h_5, h_3, h_6
(h_1, h_5)	h_3, h_6, h_2
(h_1, h_6)	h_3, h_5, h_2
(h_3, h_4)	h_5, h_1, h_6
(h_3, h_5)	h_6, h_4, h_1
(h_3, h_6)	h_5, h_1, h_4

If $(L'(v_1), L'(v_5)) =$	Then label v_2, v_3, v_4 with
(h_2, h_3)	h_5, h_1, h_6
(h_2, h_4)	h_5, h_1, h_6
(h_2, h_5)	h_6, h_1, h_3
(h_2, h_6)	h_5, h_1, h_3
(h_4, h_5)	h_6, h_1, h_3
(h_4, h_6)	h_1, h_5, h_2
(h_5, h_6)	h_1, h_4, h_2

If there is no maximal non-labelled intervals of size 3, and there are at most 7 distinct maximal non-labelled intervals, then we can easily label all vertices greedily. □

We now define a function F and a value k both of which are crucial later. We then prove a simple lemma.

Definition 5.5.5 For fixed Δ^* and n, define $F : \mathbb{N} \to \mathbb{R}$ by

$$F(m) = \begin{cases} m & \text{if } m \leq 2, \\ \frac{\Delta^*-2m-13}{6} F(m-2) & \text{otherwise} \end{cases}$$

and let

$$k = \left\lceil \frac{2 \ln n}{\ln(0.1\Delta^*)} + 2 \right\rceil .$$

Lemma 5.5.6 If $\Delta^* \geq 1000$, $\ln n / \ln \Delta^* < 0.01\Delta^*$, and F and k are as above then

1. If $0 < m < (\Delta^* - 13)/2$,
$$F(m) \geq \left(\frac{\Delta^* - 2m - 13}{6} \right)^{m/2-1} .$$
2. $F(k) > n$.
3. $F(\Delta^*/2 - 23) > n$.

Proof:

1. Clearly the result holds for $m = 1, 2$. By induction if $2 < m < (\Delta^* - 13)/2$, we have

$$\begin{aligned} F(m) &= \frac{\Delta^* - 2m - 13}{6} F(m-2) \\ &\geq \frac{\Delta^* - 2m - 13}{6} \left(\frac{\Delta^* - 2(m-2) - 13}{6} \right)^{(m-2)/2-1} \\ &\geq \left(\frac{\Delta^* - 2m - 13}{6} \right)^{m/2-1}. \end{aligned}$$

This completes the proof of part 1.

2. First note that

$$\frac{\ln n}{\ln(0.1\Delta^*)} = \frac{\ln n}{\ln \Delta^*} \frac{\ln \Delta^*}{\ln(0.1\Delta^*)}.$$

Since $\ln n / \ln \Delta^* < 0.01\Delta^*$ and $\Delta^* \geq 1000$ we have

$$\frac{\ln n}{\ln \Delta^*} \frac{\ln \Delta^*}{\ln(0.1\Delta^*)} < 0.01\Delta^* \frac{\ln \Delta^*}{\ln \Delta^* - \ln 10} \leq 0.01\Delta^* \frac{\ln 1000}{\ln 1000 - \ln 10} = 0.015\Delta^*.$$

Hence $\ln n / \ln(0.1\Delta^*) < 0.015\Delta^*$. Now since $\Delta^* \geq 1000$, we have

$$\frac{17}{300}\Delta^* - 4 > 0.$$

Hence

$$0.1\Delta^* < \frac{47\Delta^*}{300} - 4 < \frac{\Delta^* - 2(0.03\Delta^* + 3) - 13}{6}.$$

Thus

$$\begin{aligned} \ln n &= \frac{\ln n}{\ln(0.1\Delta^*)} \ln(0.1\Delta^*) \\ &< \frac{\ln n}{\ln(0.1\Delta^*)} \ln \left(\frac{\Delta^* - 2(0.03\Delta^* + 3) - 13}{6} \right). \end{aligned}$$

Using $\ln n / \ln(0.1\Delta^*) < 0.015\Delta^*$, we have

$$\begin{aligned} ln(n) &< \frac{\ln n}{\ln(0.1\Delta^*)} \ln\left(\frac{\Delta^* - 2(0.03\Delta^* + 3) - 13}{6}\right) \\ &< \left(\frac{\ln n}{\ln(0.1\Delta^*)} + 1 - 1\right) \ln\left(\frac{\Delta^* - 2k - 13}{6}\right) \\ &\leq \left(\frac{k}{2} - 1\right) \ln\left(\frac{\Delta^* - 2k - 13}{6}\right). \end{aligned}$$

Thus $n < F(k)$.

3. Note that the following holds, when $i > 0$ is an integer.

$$\begin{aligned} F(2i) &= 2 \times \tfrac{\Delta^*-8-13}{6} \times \tfrac{\Delta^*-12-13}{6} \times \ldots \times \tfrac{\Delta^*-2(2i)-13}{6} \\ F(2i-1) &= \tfrac{\Delta^*-6-13}{6} \times \tfrac{\Delta^*-10-13}{6} \times \ldots \times \tfrac{\Delta^*-2(2i-1)-13}{6} \end{aligned}$$

It is not difficult to see that $F(2i) > F(2i-1)$ if $2 \times \frac{\Delta^*-4i-13}{6} > \frac{\Delta^*-6-13}{6}$ and $F(2i+1) > F(2i)$ if $\frac{\Delta^*-2(2i+1)-13}{6} > 2$. This implies that $F(j)$ is an increasing function when $j < min\{\frac{\Delta^*-25}{2}, \frac{\Delta^*-7}{4}\}$. As $k + 200 < 0.03\Delta^* + 3 + 200 \leq 0.03\Delta^* + 0.22\Delta^* - 220 + 203 \leq 0.25\Delta^* - 17$, we see that $F(j)$ is increasing when $j < k + 200$.

Note that $\Delta^*/2 - 23 \geq 0.03\Delta^* + 3 > k$ as $\Delta^* \geq 1000$. Let j be any integer, such that $k \leq j \leq \Delta^*/2 - 23$. Note that $F(j) > F(j-2)$ as $\frac{\Delta^*-2j-13}{6} > 1$. Furhtermore note that the following holds, where $z = 0$ if $j - k$ is even and $z = 1$ otherwise.

$F(j) > F(j-2) > F(j-4) > \ldots > F(k+z) \geq F(k) > n$, by 2,

□

We now define the $OBFS_2$ tree of an outerplanar graph G. This is a modification of the $OBFS$ trees studied in Section 5.4.

Definition 5.5.7 Let $(v_1, v_2, \ldots, v_n)$ be an outerplanar ordering of $V(G)$. The $OBFS_2$ tree, which technically is two disjoint trees, is constructed as follows. Add

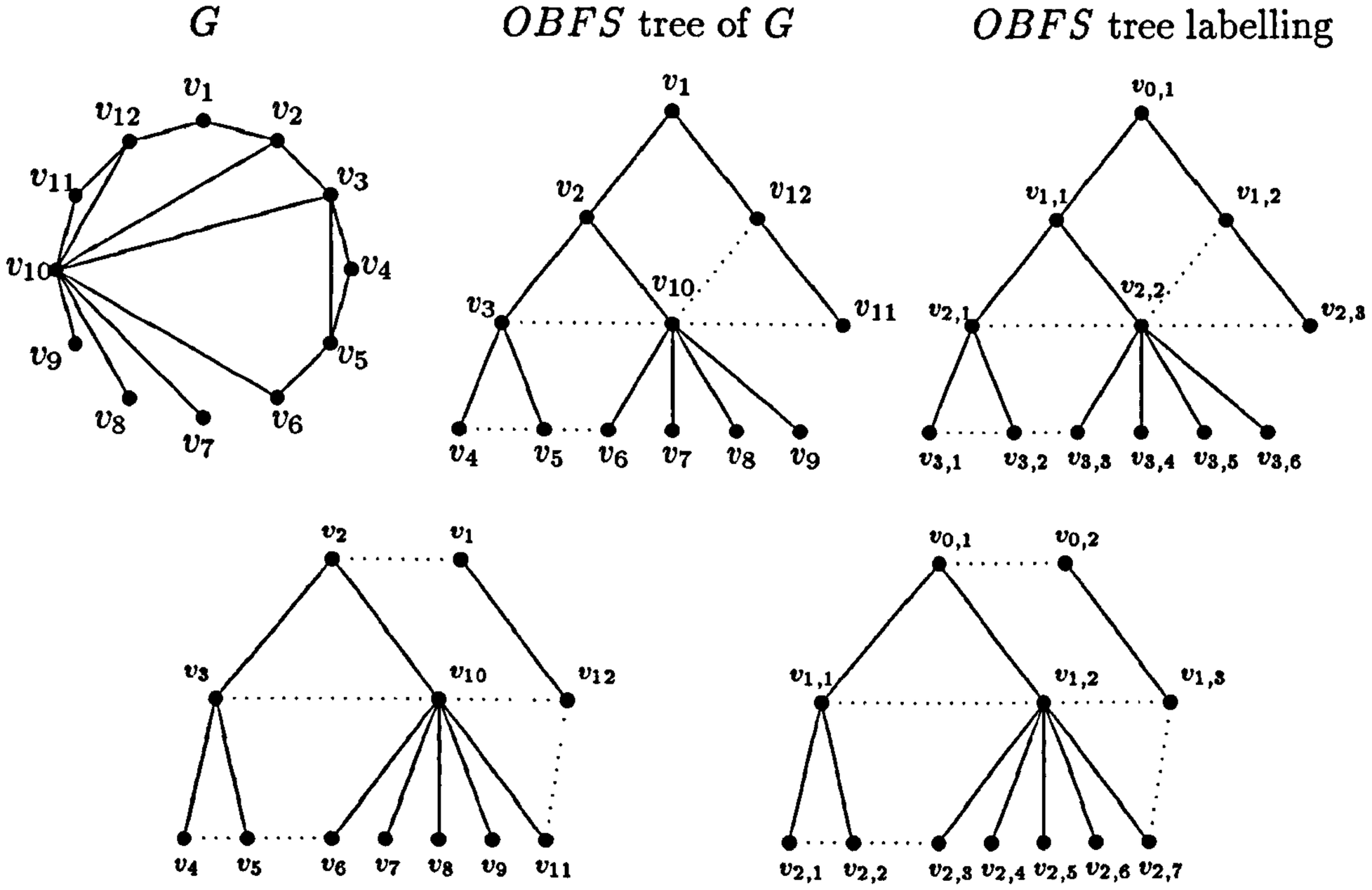


Figure 5.6: This is an example of an outerplanar graph, G, with its OBFS tree, and the labelling of the OBFS tree. The dotted lines illustrate edges in G, which are not in the $OBFS$ tree. The bottom two graphs illustrate the $OBFS_2$ tree and its labelling.

a vertex v^* to G, and the edges v^*v_1 and v^*v_2. The $OBFS_2$ tree is obtained by deleting v^* from a breadth first search tree starting at v^*, where vertices with lower index are always chosen before vertices with higher index, except v_2 is chosen before v_1.

In an $OBFS_2$ tree, we define the vertices at level 0 to be v_1 and v_2 and for $j \geq 1$ we define the vertices at level j to be the children of the vertices at level $j-1$. Analogously as for the $OBFS$ tree we have a natural ordering on the vertices in each level defined by the order in which they were added to the $OBFS_2$ tree. As before we let $v_{l,i}$ denote the i'th node on level l. See Figure 5.6 for an example.

For an $OBFS$ or $OBFS_2$ tree let $L_i(T)$ contain all vertices at level i.

Even though most of the following definitions are valid for $OBFS$ trees as well as $OBFS_2$ trees, we shall only prove them for $OBFS_2$ trees, as these are the kind of trees we will need later in the proof of our main result.

Definition 5.5.8 If T is an $OBFS_2$ tree of an outerplanar graph G, then we will need the following definitions.

(a): Define T^* as follows. Assume that T has l levels, $L_0(T), L_1(T), \ldots, L_l(T)$. Let the vertex set of T and T^* be the same and let T^* contain the edges of G, as well as the following edges (see definition 5.5.7 for a definition of the vertex labels).

$$\{v_{i,j}v_{i,j+1} \mid \text{There is a } (v_{i,j}, v_{i,j+1})\text{-path in } G \text{ only including vertices from } \{v_{i,j}, v_{i,j+1}\} \cup L_{i+1}(T) \cup L_{i+2}(T) \cup \ldots \cup L_l(T)\}$$

(b): Let T^* be defined as in (a). A (p_0, p_z)-path, $P = p_0p_1p_2 \ldots p_z$, in T^* is called a (p_0, p_z)-AD-path (AD stands for Almost Down) if for all $a < b$ we have that p_a lies on the same level as p_b or on a higher level (i.e. a level with smaller index). Furthermore P may contain at most three vertices from any level.

(c): For any vertex $v_{i,j}$ on level i in T, let $ADP(v_{i,j})$ contain all vertices in $L_i(T) \cup L_{i+1}(T) \cup \ldots \cup L_l(T)$ which are connected to $v_{i,j}$ with an AD-path in T^*. Let $adp(v_{i,j}) = |ADP(v_{i,j})|$.

(d): For all $v_{i,j} \in T$, define $mc_{\Delta^*}(v_{i,j})$ to be the smallest integer such that $F(1 + mc_{\Delta^*}(v_{i,j})) > adp(v_{i,j})$ (see Definition 5.5.5).

(e): Let $H_{\Delta^*}(j) = \{j-1, j, \ldots, \Delta^* + 2 - j\}$. Note that $\{0, 1, 2, \ldots, \Delta^* + 1\} - H_{\Delta^*}(j) = I_1 \cup I_2$, where $I_1 = \{0, 1, 2, \ldots, j-2\}$ and $I_2 = \{\Delta^* + 3 - j, \Delta^* + 4 - j, \ldots, \Delta^* + 1\}$ and that $|I_1| = |I_2| = j - 1$.

Lemma 5.5.9 Let G be an outerplanar graph with an $OBFS$ tree T. Let $(v_1, v_2, \ldots, v_n)$ be an outerplanar ordering. Define $z(i,j)$ such that $v_{i,j} = v_{z(i,j)}$ (i.e. the j'th vertex on level i in T has index $z(i,j)$ in the outerplanar ordering). For every i we have $z(i,1) < z(i,2) < z(i,3) < \ldots$

Proof: We will show the theorem by induction on i. Clearly it is true for $i = 0$ and $i = 1$. So assume the theorem holds for all levels $0, 1, 2, \ldots, i-1$. We will now show that $z(i,1) < z(i,2) < z(i,3) < \ldots$. Assume that this is not true and that $z(i,j) > z(i,j+1)$. Let $v_{i-1,s}$ be the parent of $v_{i,j}$ in T and let $v_{i-1,t}$ be the parent of $v_{i,j+1}$ in T. By the construction of a OBFS tree we note that $s < t$ (in particular $s \neq t$). By induction we know that $z(i-1,s) < z(i-1,t)$.

Note that $z(i-1,s) \notin \{2, 3, 4, \ldots, z(i,j+1)\}$ as if $z(i-1,s)$ did belong to this set then any path from v_1 to $v_{z(i,j+1)}$ would have to pass through either $v_{z(i-1,s)}$ or $v_{z(i,j)}$ (as $1 < z(i-1,s) < z(i,j+1) < z(i,j)$ and there is an edge between $v_{z(i-1,s)}$ and $v_{z(i,j)}$). However there is a path from v_1 to $v_{z(i,j+1)}$ in the tree which does not pass through either $v_{z(i-1,s)}$ or $v_{z(i,j)}$. This proves that $z(i-1,s) \notin \{2, 3, 4, \ldots, z(i,j+1)\}$. Analogously we get that $z(i-1,t) \notin \{z(i,j), z(i,j)+1, \ldots, n\}$. As $z(i-1,s) < z(i-1,t)$ the above implies that $1 < z(i,j+1) < z(i-1,s) < z(i-1,t) < z(i,j)$. However now the edges $v_{z(i,j)}v_{z(i-1,s)}$ and $v_{z(i,j+1)}v_{z(i-1,t)}$ give us the desired contradiction. □

Lemma 5.5.10 Let G be an outerplanar graph with an $OBFS_2$ tree T. Let T^* be defined as in definition 5.5.8. The following now holds.

(i) Let $v_{i,j}$ and $v_{i,k}$ be any two vertices on level i of T, such that $|j - k| \geq 2$. Then every $(v_{i,j}, v_{i,k})$-path in G, must include a vertex from $L_0(T) \cup L_1(T) \cup \ldots \cup L_i(T) - \{v_{i,j}, v_{i,k}\}$.

(ii) Let $v_{i,j}$ and $v_{i,k}$ be any two vertices on level i of T, such that $|j-k| \geq 6$. Then $ADP(v_{i,j}) \cap ADP(v_{i,k}) = \emptyset$.

Proof: We will first prove (i), so assume that i, j and k are chosen as in (i), but that there exists a $(v_{i,j}, v_{i,k})$-path P in G, which contains no vertices from $L_0(T) \cup L_1(T) \cup \ldots \cup L_i(T) - \{v_{i,j}, v_{i,k}\}$. Assume without loss of generality that $j < k$ and let $r \in 1,2$ be chosen such that there exists a $(v_{0,r}, v_{i,j+1})$-path, Q, in T. If $(v_1, v_2, \ldots, v_n)$ is an outerplanar ordering of G and $v_{0,r} = v_{i_1}$, $v_{i,j} = v_{i_2}$, $v_{i,j+1} = v_{i_3}$ and $v_{i,k} = v_{i_4}$, then $i_1 < i_2 < i_3 < i_4$ by Lemma 5.5.9. As Q is a (v_{i_1}, v_{i_3})-path we see that P, which is a (v_{i_2}, v_{i_4})-path, must contain a vertex in $V(Q)$, a contradiction.

In order to prove part (ii) let Z_q contain all vertices z, such that there exists a $(v_{i,q}, z)$-path in G : $\{v_{i,q}\} \cup L_{i+1}(T) \cup L_{i+2}(T) \cup \ldots \cup L_l(T)$. Note that $ADP(v_{i,j}) \subseteq Z_{j-2} \cup Z_{j-1} \cup Z_j \cup Z_{j+1} \cup Z_{j+2}$ and $ADP(v_{i,k}) \subseteq Z_{k-2} \cup Z_{k-1} \cup Z_k \cup Z_{k+1} \cup Z_{k+2}$, by the definition of ADP. However by (i) we see that $(Z_{j-2} \cup Z_{j-1} \cup Z_j \cup Z_{j+1} \cup Z_{j+2}) \cap (Z_{k-2} \cup Z_{k-1} \cup Z_k \cup Z_{k+1} \cup Z_{k+2}) = \emptyset$, which proves part (ii). □

Lemma 5.5.11 Let G be an outerplanar graph and let T be an $OBFS_2$ tree of G (see definition 5.5.7), with l levels. Assume that G has maximum degree at most Δ^*, where $\Delta^* \geq 1000$. Assume that $|H_{\Delta^*}(mc_{\Delta^*}(v))| \geq 50$ for all $v \in V(G)$, and that G has a (Δ^*+2)-$L(2,1)$-labelling, L.

For some i let $M_i = L_0(T) \cup L_1(T) \cup L_2(T) \cup \ldots \cup L_i(T)$. Suppose that L' is a labelling of the vertices M_i, such that L' is a partial-$L(2,1)$-labelling of G. Furthermore assume that for all $v \in M_i$ either $L(v) = L'(v)$ or $L(v) \in H_{\Delta^*}(mc_{\Delta^*}(v))$ and $L'(v) \in H_{\Delta^*}(1 + mc_{\Delta^*}(v))$. Then L' can be extended to a (Δ^*+2)-$L(2,1)$-labelling of G.

Proof: We will show this by induction on i. In fact all we need to show is that we can label all vertices in level $i+1$, such that the new labelling of levels 0 to $i+1$, satisfies the conditions of the lemma. As Δ^* is a constant throughout the proof we will omit the subscripts in $mc_{\Delta^*}(v)$ and $H_{\Delta^*}(j)$.

Let $C(v_{i,j})$ denote the children of $v_{i,j}$ in T. We will now label the vertices in $C(v_{i,1})$, $C(v_{i,2}), \ldots, C(v_{i,l_i})$ in that order. So assume that we have labelled $C(v_{i,1})$,

$C(v_{i,2}), \ldots, C(v_{i,k-1})$ and now need to label $C(v_{i,k})$. Let $C(v_{i,k})$ and $C^*(v_{i,k})$ be defined as follows.

$$\begin{aligned} C(v_{i,k}) &= \{v_{i+1,s}, v_{i+1,s+1}, \cdots, v_{i+1,s+t}\} \\ C^*(v_{i,k}) &= \{v_{i+1,s}, v_{i+1,s+1}, v_{i+1,s+t-1}, v_{i+1,s+t}\} \end{aligned}$$

(if $|C(v_{i,k})| \leq 4$, then let $C^*(v_{i,k}) = C(v_{i,k})$). Let $H^* = \{\Delta^*/2 - 20, \Delta^*/2 - 19, \ldots, \Delta^*/2 + 20\}$. We will make sure that every vertex $v \in C^*(v_{i,k})$ either gets the same label as in L or a label in H^*. We will also assume that we fulfilled this condition when we labelled the previous vertices on level $i+1$.

Let DP denote all the vertices in $L_{i+1}(T) \cup L_{i+2}(T) \cup \ldots \cup L_l(T)$ that can be reached by an AD-path from $v_{i,k}$, which only contains the vertex $v_{i,k}$ from level i (that is it starts of with an edge from level i to level $i+1$). Let m be the smallest value such that $F(1+m) > |DP|$. First label all vertices $v \in C(v_{i,k})$ where $L(v) \notin H(m)$ with the label $L'(v) = L(v)$. Then label every unlabelled vertex $v \in C^*(v_{i,k})$ with any available label from H^*. This can always be done as $|H^*| = 41$, and there are always less than 20 of these labels we cannot use.

Now let NC denote all the vertices in $C(v_{i,k})$ which have not been labelled yet. Let FB denote all labels already used to label vertices adjacent to $v_{i,k}$ as well as the label $L'(v_{i,k})$ and the labels only differing by one from $L'(v_{i,k})$. Note that we may not label any vertex in NC with a label in FB.

Let Q contain all vertices $q \in L_i(T) \cup L_{i-1}(T)$, which have a path of length one or two to a vertex in $C(v_{i,k})$. We now need the following properties.

(i): For all $c \in C(v_{i,k})$ and $q \in Q$ we have $adp(c) \leq |DP| \leq adp(q)$ and $mc(c) \leq m \leq mc(q)$.

$adp(c) \leq |DP|$ is true as if P is a (c,u)-AD-path then $u \in DP$, by considering the path $v_{i,k}P$. We will now prove that $|DP| \leq adp(q)$.

First note that if a vertex in $C(v_{i,k})$ has an edge to $v_{i,k+1}$ in T^*, then $v_{i,k}v_{i,k+1}$ is an edge in T^*, by the definition of T^*. Analogously if there is a path from $C_{i,k}$ to $v_{i,k-1}$ then $v_{i,k}v_{i,k-1}$ is an edge in T^*. Therefore every $q \in Q$ has an AD-path, R_q to $v_{i,k}$. Now if P is an $(v_{i,k},u)$-AD-path starting with an

edge from $v_{i,k}$ to $C(v_{i,k})$, then R_qP is an (q,u)-AD-path. This implies that $|DP| \leq adp(q)$.

By the definition of m (and the above) we note that $F(1+m) > |DP| \geq adp(c)$. This implies that $m \geq mc(c)$, as $mc(c)$ was chosen as the minimum value such that $F(1+mc(c)) > adp(c)$. As $F(1+mc(q) > adp(q) \geq |DP|$, we note that $m \leq mc(q)$, since m was chosen as the minimum value such that $F(1+m) > |DP|$.

(ii): All $L(2,1)$-constraints are still satisfied.

If $v \in C(v_{i,k})$ is labelled then it either has a label in H^*, in which case no constraint is violated, by our construction, or $L(v) = L'(v) \notin H(m)$. So assume that $L(v) = L'(v) \notin H(m)$, but that a constraint with u is violated.

If $u \in L_{i+1}(T)$ then by our previous assumption either $L'(u) = L(u)$ or $L'(u) \in H^*$ (as $u \in C^*(w)$ for some $w \in L_i(T)$). However this implies that the constraint between u and v is not violated as $H^* \subseteq H(m+1)$. If $u \notin L_{i+1}(T)$ then $u \in Q$, which by (i) implies that either $L'(u) = L(u)$, or $L'(u) \in H(mc(u)+1) \subseteq H(m+1)$. However in both cases no constraint is violated, as L is a proper $L(2,1)$-labelling of G and $L'(v) \notin H(m)$, respectively.

(iii): $|H(m-1) - FB| \geq |NC|$.

Let Z denote all neighbours of $v_{i,k}$ in G which have already received a L'-label. If $z \in Z \cup \{v_{i,k}\}$ then z is either in Q or on level $i+1$. Therefore either $L'(z) = L(z)$ or both $L(z) \in H(m)$ and $L'(z) \in H(m+1) \subseteq H(m)$ (as if $z \in L_{i+1}(T)$ then $L'(z) \in H^* \subseteq H(m+1)$ and otherwise $mc(z) \geq m$ by (i)).

Therefore the vertices in $Z \cup \{v_{i,k}\}$ would add equally many labels to $H(m-1) \cap FB$ whether we considered the L-values or the L'-values. If we did consider the L-labels instead of the L'-values then there would be at least $|NC|$ labels in $H(m-1) - FB$, as all vertices in NC did actually have a L-label in $H(m-1) - FB$. So therefore there are also at least $|NC|$ labels in $H(m-1) - FB$ when we consider the L'-labels, which completes the proof of (iii).

(iv): If $|NC| > |H(m+1) - FB|$ then there are at least 6 vertices in NC, which have an mc value of at most $m-2$.

Note that $|FB \cap H(m+1)| \leq 12$, as $v_{i,k}$ may add three labels to $FB \cap H(m+1)$, and each vertex in $C^*(v_{i,k})$ may add one label to $FB \cap H(m+1)$, and if $v_{i-1,a}$ is the parent of $v_{i,k}$ in T, then each of the vertices $v_{i-1,a}, v_{i-1,a+1}, v_{i,k-1}, v_{i,k+1}, v_{i+1,s-1}$ may also add one label to $FB \cap H(m+1)$. If there is not six vertices in NC, which have an mc value of at most $m-2$, then we must have at least $|H(m+1)|-12-5$ vertices in NC which all have a mc value of at least $m-1$. Let $V' = \{v_{i+1,r_1}, v_{i+1,r_2}, \ldots, v_{i+1,r_{|H(m+1)|-17}}\}$ be $|H(m+1)|-17$ such vertices, such that $r_1 < r_2 < \ldots < r_{|H(m+1)|-17}$. Note that $ADP(v_{i+1,r_j}) \subseteq DP$ and $adp(v_{i+1,r_j}) \geq F(m-1)$ for all $j = 1, 2, \ldots, |H(m+1)|-17$. By Lemma 5.5.10 we note that the following holds.

$$\begin{aligned} F(1+m) &> |DP| \geq |\cup_{v \in V'} ADP(v')| \\ &\geq adp(v_{i+1,r_1}) + adp(v_{i+1,r_7}) + adp(v_{i+1,r_{13}}) + adp(v_{i+1,r_{19}}) + \cdots \\ &\geq \frac{|H(m+1)|-17}{6} F(m-1) \\ &= \frac{\Delta^* - 15 - 2m}{6} F(m-1) \end{aligned}$$

This is the desired contradiction to Definition 5.5.5.

If $|NC| \leq |H(m+1) - FB|$, then we can label all vertices in NC with labels from $H(m+1)-FB$, using Lemma 5.5.4, as $|NC| \leq |H(m+1)-FB|$ and $|H(m+1) - FB| \geq 48 - 12 \geq 22$ (as $m \leq mc(v_{i,k})$ and $|H(mc(v_{i,k})| \geq 50$).

Now consider the case when $|NC| > |H(m+1) - FB|$. By (iv) we can let Y denote 6 vertices in NC, which have an mc value of at most $m-2$. For each label $c^* = m-2, \Delta^*+3-m, m-1, \Delta^*+2-m$ (in that order) we do the following. If $c^* \in H(m-1) - FB$, then pick a vertex $y \in Y$, which is not adjacent to a vertex of adjacent label to c^*, and label it with label c^*. Then delete y from Y and NC, and add c^* to FB. Continue until all of the possible c^* have been tried. Note that

now $H(m-1)-FB \subseteq H(m+1)$, which implies that $|NC| \leq |H(m-1)-FB| = |H(m+1)-FB|$ by (iii) and the fact that if we decrease $|NC|$ by one then we also decreased $|H(m-1)-FB|$ by one. We are now done analogously to above (using Lemma 5.5.4). □

Corollary 5.5.12 Let G be an outerplanar graph, with maximum degree at most Δ^*, where $\Delta^* \geq 1000$ and $\frac{\ln(n)}{\ln(\Delta^*)} < 0.01\Delta^*$. Suppose G has a (Δ^*+2)-$L(2,1)$-labelling, L. Let $(v_1, v_2, \ldots, v_n)$ be any outerplanar-ordering of $V(G)$. Let $k = \left\lceil \frac{2\ln(n)}{\ln(0.1\Delta^*)} + 2 \right\rceil$.

Let L' be a labelling of $N[v_1, v_2]$, such that L' is a partial-(Δ^*+2)-$L(2,1)$-labelling of G and for all $j = 1, 2, \ldots, l$, either $L(v_{i_j}) = L'(v_{i_j})$ or $L(v_{i_j}), L'(v_{i_j}) \subseteq H(k)$. Then L' can be extended to a (Δ^*+2)-$L(2,1)$-labelling of G.

Proof: We show that we may use Lemma 5.5.11 with $i = 1$. Using Definition 5.5.8, for any vertex v, we have $adp(v) \leq n$ and so using Lemma 5.5.6 we see that $mc_\Delta(v) \leq k-1$. Similarly for all v $F(\Delta^*/2 - 23) > n \geq adp(v)$ and so $mc_{\Delta^*}(v) \leq \Delta^*/2 - 23$. Thus $|H_{\Delta^*}(mc_{\Delta^*}(v))| = \Delta + 4 - 2mc_{\Delta^*}(v) \geq 50$. Therefore all the conditions of Lemma 5.5.11 apply and L' may be extended to L (note that $H(k) \subseteq H(mc_{\Delta^*}(v)+1) \subseteq H(mc_{\Delta^*}(v))$ for all $v \in V(G)$). □

We are finally ready to prove our main theorem.

Theorem 5.5.13 Let G be an outerplanar graph. There exists a polynomial time algorithm to find the optimum $L(2,1)$ span for G.

Proof: Let $\Delta^* = \Delta(G)$. If $\Delta^* \leq 1000$ or if $\frac{\ln(n)}{\ln(\Delta^*)} \geq 0.01\Delta^*$, then we are done by Corollary 5.5.3, so assume that this is not the case. By Corollary 5.4.5 we note that the optimum $L(2,1)$ span for G is either Δ^*+2 or Δ^*+3, so we will let $\sigma = \Delta^*+2$, and decide if there is a σ-$L(2,1)$-labelling of G. Let $(v_1, v_2, \ldots, v_n)$ be an outerplanar-ordering of the vertices of G. Let $k = \left\lceil \frac{2\ln(n)}{\ln(0.1\Delta^*)} + 2 \right\rceil$. Our algorithm will now follow the approach in Lemma 5.5.1, except we label all the vertices with the labels $\{0, 1, 2, \ldots, k-1\} \cup \{\Delta^*+2-k, \Delta^*+3-k, \ldots, \Delta^*+1\} \cup \{h\}$, where h

is an artificial label. Such a labelling is said to be an H-labelling. A vertex initially labelled with h will later be given some label from $H = \{k, k+1, \ldots, \Delta^* + 1 - k\}$.

If L is a σ-$L(2,1)$-labelling of G, then let $L_H(v) = L(v)$ if $L(v) \notin H$ and $L_H(v) = h$ otherwise. Note that L_H is an H-labelling of G. Let W be a partial H-labelling of G. If there exists a σ-$L(2,1)$-labelling, L, of G, such that L_H extends W, then we say that W can be extended to a σ-$L(2,1)$-labelling (namely L).

Our algorithm will find all possible H-labellings of $N[v_1, v_2]$, that can be extended to a σ-$L(2,1)$-labelling of G. If there exist no such H-labellings of $N[v_1, v_2]$, then there exist no σ-$L(2,1)$-labelling of G and otherwise there does. Our algorithm is recursive, and will produce a (possibly empty) list of H-labellings of $N[v_1, v_2]$.

If $|V(G)| \leq 2$ then we just return all possible H-labellings of G. Otherwise let $i = i^*(1)$ (see the definitions of $i^*(1)$ above Theorem 5.4.4). Now recursively solve the problem for the two graphs $G_1 = G : \{v_2, v_3, \ldots, v_i\}$ and $G_2 = G : \{v_i, v_{i+1}, \ldots, v_n, v_1\}$. As before, in G_1, we think of v_i as playing the same role as v_1 does in G, so in other words we look for H-labellings of $N[v_i, v_2]$ that may be extended to a σ-$L(2,1)$-labelling of G_1. Similarly, in G_2, we think of v_i as playing the same role as v_2 does in G. Let $\mathcal{L}_1$ contain all possible H-labellings of $N_{G_1}[v_2, v_i]$ which can be extended to a σ-$L(2,1)$-labelling of G_1. Let $\mathcal{L}_2$ contain all possible H-labellings of $N_{G_2}[v_1, v_i]$ which can be extended to a σ-$L(2,1)$-labelling of G_2. We build a list $\mathcal{L}$ of H-labellings of $N_G[v_1, v_2]$ that can be extended to a σ-$L(2,1)$-labelling of G. Initially $\mathcal{L}$ is empty.

If $L_1 \in \mathcal{L}_1$ and $L_2 \in \mathcal{L}_2$, then we can check (in $O(|N_{G_1}[v_2, v_i]| \times |N_{G_2}[v_1, v_i]|)$ time) whether the following hold:

- L_1 and L_2 have the same label in v_i.
- Let $v \in N[v_1, v_2, v_i]$ be arbitrary. If the label of v is h, then it is adjacent to no more than $|H| - 3$ vertices of label h.

  If the label of v is $k - 1$ or $\Delta^* + 2 - k$, then it is adjacent to no more than $|H| - 1$ vertices of label h.

  Otherwise it is adjacent to no more than $|H|$ vertices of label h.

- For all non-h labels the $L(2,1)$-constraints hold.

If the above hold then add the H-labelling $L_1 \cup L_2$, restricted to $N_G[v_1, v_2]$, to our list $\mathcal{L}$. Once we have tried all pairs L_1 and L_2, we stop and return $\mathcal{L}$.

We will now prove that if our algorithm returns an H-labelling, L, of $N_G[v_1, v_2]$ then this can be extended to a σ-$L(2,1)$-labelling of G. Let L_1 be the H-labelling of $N_{G_1}[v_2, v_i]$ and L_2 be the H-labelling of $N_{G_2}[v_1, v_i]$ which caused us to add L to $\mathcal{L}$. By induction L_1 can be extended to a σ-$L(2,1)$-labelling, L_1', of G_1 and L_2 can be extended to a σ-$L(2,1)$-labelling, L_2', of G_2. First consider the H-labelling, $L' = L_1 \cup L_2$ of $N_G[v_1, v_2, v_i]$. Note that this can be extended to a proper $\sigma - L(2,1)$-labelling, L'', of $N_G[v_1, v_2, v_i]$, by first labelling v_1, v_2 and v_i with any allowed label (if they had label h in L'), and then using Lemma 5.5.4 for $N_G[v_1]$, $N_G[v_2]$ and $N_G[v_i]$ in that order (there may be some constraints between vertices in $N_G[v_1]$, $N_G[v_2]$ and $N_G[v_i]$, however these are limited to only a couple of vertices, and do not cause any problems).

We apply Corollary 5.5.12 to G_1 to see that the labelling L'' restricted to $N[v_i, v_2]$ may be extended to an $L(2,1)$-labelling L_1'' of G_1. This follows because of the existence of L_1' and the fact that for all $v \in N(v_i, v_2)$, either $L''(v) = L_1'(v)$ or $L''(v), L_1'(v) \in \{k, \ldots, \Delta^* + 1 - k\}$. Note that $\left\lceil \frac{2 \ln n(G_1)}{\ln(0.1\Delta^*)} + 2 \right\rceil \leq \left\lceil \frac{2 \ln n(G)}{\ln(0.1\Delta^*)} + 2 \right\rceil$. Similarly we may extend the labelling L'' restricted to $N[v_1, v_i]$ to an $L(2,1)$-labelling L_2'' of G_2. $L_1'' \cup L_2''$ is a σ-$L(2,1)$-labelling of G.

Now assume that there exists a σ-$L(2,1)$-labelling, L, of G. Recall $H_{\Delta^*}(j) = \{j-1, j, \ldots, \Delta^* + 2 - j\}$. We first show there is a σ-$L(2,1)$-labelling $\bar{L}$ of G such that $\bar{L}(v_1), \bar{L}(v_2) \notin \{k, \Delta^* + 1 - k\}$. Suppose L is a σ-$L(2,1)$-labelling of G in which at least one of v_1 and v_2 receives either label k or $\Delta^* + 1 - k$. We construct a labelling $\bar{L}'$ of $N[v_1, v_2]$ by first setting $\bar{L}'(v) = L(v)$ if $v \in N[v_1, v_2]$ and $L(v) \notin H_{\Delta^*}(k)$.

Then if either of v_1, v_2 have not yet been labelled in $\bar{L}'$ we give them a label in $H_{\Delta^*}(k+2)$ so that no constraints are broken. Using Lemma 5.5.4 twice we label the unlabelled vertices of first $N[v_1]$ and then $N[v_2]$ with labels from $H_{\Delta^*}(k)$.

By Corollary 5.5.12 $\bar{L}'$ may be extended to a σ-$L(2,1)$-labelling $\bar{L}$ of G.

We now show that for any labelling L, where $L(v_1), L(v_2) \notin \{k, \Delta^* + 1 - k\}$, our

algorithm will return L_H restricted to $N[v_1, v_2]$. Again we prove this by induction.

First suppose that $L(v_i) \notin \{k, \Delta^* + 1 - k\}$. Let L_1 be the restriction to G_1 of L and L_2 be the restriction to G_2 of L. By induction $(L_1)_H$ restricted to $N[v_i, v_2]$ and $(L_2)_H$ restricted to $N[v_1, v_i]$ will be returned by the algorithm applied to G_1 and G_2 respectively. By the definition of the algorithm $(L_1)_H \cup (L_2)_H$ restricted to $N[v_1, v_2]$ will be returned by the algorithm and so we are done.

Now suppose that $L(v_i) \in \{k, \Delta^* + 1 - k\}$. We construct a labelling L' of $N_G[v_1, v_2]$ as follows. First let $L'(v_1) = L(v_1)$, $L'(v_2) = L(v_2)$ and let $L'(v_i)$ be a member of $H_{\Delta^*}(k+2)$. If $v \notin \{v_1, v_2, v_i\}$ and $L(v) \notin H_{\Delta^*}(k+1)$ set $L'(v) = L(v)$.

Now we use Lemma 5.5.4 twice to label the remaining vertices of $N_G[v_1]$ and $N_G[v_2]$ with labels from $H_{\Delta^*}(k+1)$. There may be an additional constraint between a vertex $u_1 \in N_G(v_1)$ and $u_2 \in N_G(v_2)$ if both are adjacent to v_i. However it is not difficult to see that this does not create any additional problems.

Corollary 5.5.12 may now be applied to extend L' to a labelling of L'' of G.

Let L_1 be L'' restricted to G_1 and L_2 be L'' restricted to G_2. Since $L'(v_1)$, $L'(v_2)$, $L'(v_i) \notin \{k+1, \Delta^* + 2 - k\}$, by induction $(L_1)_H$ restricted to $N_{G_1}[v_i, v_2]$ will be returned by the algorithm applied to G_1 and $(L_2)_H$ restricted to $N_{G_2}[v_1, v_i]$ will be returned by the algorithm applied to G_2. Now by the definition of the algorithm $(L_1)_H \cup (L_2)_H$ restricted to $N_G[v_1, v_2]$ be returned by the algorithm and this is L_H restricted to $N_G[v_1, v_2]$. This completes the proof that the algorithm works correctly.

We will now prove that the algorithm is polynomial. This follows easily if we can show that the number of H-labellings that the algorithm returns (in each recursive step) is at most a polynomial. We first consider how many H-labellings of $N[v_1]$ are partial H-labellings of G. Any label in $\{0, 1, \ldots, k-1\} \cup \{\Delta^* + 2 - k, \Delta^* + 3 - k, \ldots, \Delta^* + 1\}$ may be assigned to one of the at most $\Delta^* + 1$ vertices in $N[v_1]$, or to no vertex at all. So for each label we have $\Delta^* + 2$ possibilities. The label h is assigned to all vertices which have not received any of the other $2k$ labels. So there exists at most $(\Delta^* + 2)^{2k}$ H-labellings of $N[v_1]$. If $k \leq 12$, then this is clearly a

polynomial, and if $k \geq 12$, then

$$\begin{aligned} k &\leq 3 + \frac{2\ln n}{\ln(0.1\Delta^*)} \\ &= 3 + \frac{2\ln n}{\ln \Delta^* - \ln 10} \frac{\ln \Delta^*}{\ln \Delta^*} \\ &\leq 3 + \frac{3\ln n}{\ln \Delta^*} \end{aligned}$$

as $\Delta^* \geq 1000$. Since $k \geq 12$ we must have $\ln n / \ln \Delta^* \geq 3$ and so $k \leq \frac{4\ln n}{\ln \Delta^*}$. This implies

$$(\Delta^* + 2)^{2k} \leq (\Delta^{*2})^{8\ln(n)/\ln(\Delta^*)} = \Delta^{*\frac{\ln(n^{16})}{\ln(\Delta^*)}} = n^{16}.$$

Hence there are at most a polynomial number of H-labellings satisfying the $L(2,1)$-constraints for $N[v_1, v_2]$, as this is at most the number of H-labellings satisfying the $L(2,1)$-constraints for $N[v_1]$ times the number of H-labellings satisfying the $L(2,1)$-constraints for $N[v_2]$. This completes the proof. □

A series parallel network is a graph recursively defined as follows. A graph on two vertices s (source) and t (target), with one edge st is a series parallel network. Let G_1 and G_2 be two series parallel networks where s_1/t_1 are the source/target of G_1 and s_2/t_2 are the source/target of G_2. Then the following graphs are also series parallel:

Serial combination: The graph G_s that emerges from identifying t_1 with s_2. The source of G_s is s_1 and the target of G_s is t_2.

Parallel combination: The graph G_p that emerges from identifying s_1 with s_2 and t_1 with t_2. The source of G_p is s_1/s_2 and the target is t_1/t_2.

An alternative definition is that series parallel networks are exactly the graphs with treewidth two. It still remains an open problem as of whether there exists a polynomial time algorithm for deciding the minimum $L(2,1)$-span of a series parallel network.

Chapter 6

Domination Analysis for Greedy Heuristics for the Frequency Assignment Problem

In this chapter we will introduce the domination number of an algorithm. This is a measure of the quality of a heuristic. The domination number was first studied in the context of the travelling salesman problem [19] and there is now a relatively large literature on this topic, see for example [26, 28, 44, 46, 49]. We have adapted ideas from the travelling salesman problem to obtain results on the domination number of greedy heuristics for both the fixed spectrum and minimum span version of the frequency assignment problem.

6.1 Notation and definitions

The domination number was introduced in an attempt to provide a measure of both the quality of a heuristic for a combinatorial optimisation problem and also the difficulty of a combinatorial optimisation problem. The domination number does not suffer from some of the inherent drawbacks associated with other established quality measures. We will explain this in more detail later.

A *combinatorial optimisation problem* π consists of a set of instances $\mathcal{I}(\pi)$. For

each instance I there exists a set of feasible solutions $Sol(I)$ and a cost function $c : Sol(I) \to \mathbb{Q}$. The objective of the problem is to find $s^* \in Sol(I)$ such that for all $s \in Sol(I)$

$$c(s^*) \leq c(s).$$

Such a problem is a minimisation problem and for each such problem one can also define the equivalent maximisation problem by reversing the inequality.

An algorithm or heuristic for a problem π takes as input a description of the instance I and outputs a member of $Sol(I)$. For an algorithm $\mathcal{A}$, let $\mathcal{A}(I)$ denote the cost of the solution returned by $\mathcal{A}$, when run on instance I. Furthermore let $OPT(I)$ denote the cost of the optimal solution of I.

The complexity class APX is the class of combinatorial optimisation problems for which there exists a polynomial time algorithm $\mathcal{A}$ and a constant r such that for all $I \in \mathcal{I}$

$$\frac{\mathcal{A}(I)}{OPT(I)} \leq r$$

in the case of minimisation problems or

$$\frac{OPT(I)}{\mathcal{A}(I)} \leq r$$

in the case of maximisation problems.

A drawback with this well-studied class is that the travelling salesman problem is not a member of APX but the related problem of finding the longest Hamiltonian Circuit in a weighted copy of K_n is a member of APX. However a "good" algorithm for the maximisation version of the problem can be transformed to a "good" algorithm for the minimisation problem by multiplying all the edge weights by minus one. Therefore it seems a little unnatural that one of the problems should be considered easy to approximate well (it belongs to APX), while the other is not (it does not belong to APX).

We now define the domination number of a heuristic. Given an instance I of a combinatorial optimisation problem π and a heuristic $\mathcal{A}$, the domination number of $\mathcal{A}$ on input I is $\text{dom}(\mathcal{A}, I) = |\{s \in Sol(I) : c(\mathcal{A}(I)) \leq c(s)\}|$ in the case where π is

a minimisation problem. As before the inequality is reversed if π is a maximisation problem.

The domination number of a heuristic is a function of the size of the input so we need to introduce a measure of input size. Since we will focus only on graph problems we just use $|V|$ as our measure of input size. We use $|I|$ to denote the size of an instance I, that is the number of vertices in the graph described by I. Notice that this definition is very different from that used normally in complexity theory where, for example, the size of edge weights must be taken into account. When we discuss the running time of an algorithm, in particular whether the algorithm runs in polynomial time, we still use the standard notion of the size of the input.

Definition 6.1.1 The domination number of a heuristic $\mathcal{A}$ for a combinatorial optimisation problem π is

$$\mathrm{dom}(\mathcal{A}, n) = \min_{I \in \mathcal{I}(\pi): |I|=n} \mathrm{dom}(\mathcal{A}, I).$$

Since $|Sol(I)|$ may vary for different choices of I with the same size it is often useful to consider the domination ratio of a heuristic $\mathcal{A}$ on instance I, defined by

$$\mathrm{domr}(\mathcal{A}, I) = \frac{\mathrm{dom}(\mathcal{A}, I)}{|Sol(I)|}.$$

Definition 6.1.2 The domination ratio of a heuristic is

$$\mathrm{domr}(\mathcal{A}, n) = \min_{I \in \mathcal{I}(\pi): |I|=n} \mathrm{domr}(\mathcal{A}, I).$$

6.2 Known results

The concept of domination number was first applied to the travelling salesman problem. In 1997 the question was asked in [19] whether there exists a polynomial time algorithm $\mathcal{A}$ for the travelling salesman problem with $\mathrm{dom}(\mathcal{A}, n) \geq n!/p(n)$ for some polynomial $p(n)$. It was conjectured then, that unless $P = NP$, there is no such algorithm [19] . However, Gutin and Yeo [26] proved only a year later that the domination number of the greedy expectation algorithm (GEA) for the travelling

salesman problem satisfies $\text{dom}(\text{GEA}, n) \geq (n-2)!$ for every $n \neq 6$. Their result had previously independently been established in the 1970s in [49] and [50], several years before the introduction of the concept of domination number. Gutin, Yeo and Zverovich [28] showed in 2001, that if $n \geq 2$ the domination number of the greedy algorithm for the travelling salesman problem is 1. Further results on the domination number of TSP heuristics have been obtained in [44] and [46].

Currently all algorithms proved to have domination number at least $\Omega((n-2)!)$, use the following result.

Lemma 6.2.1 [26, 49, 50] Let I be some instance of the asymmetric travelling salesman problem, with n vertices, where $n \neq 6$. Let $\tau(n, c)$ be the average cost of a tour in I. If H is a tour in I, such that $c(H) \leq \tau(n, c)$, then the cost of H is less than or equal to the cost of at least $(n-2)!$ tours.

Using the above lemma the following algorithms have all been shown to have domination number $\Omega((n-2)!)$ for the asymmetric travelling salesman problem.

- *Greedy Expectation Algorithm (GEA).* The GEA will always produce a tour with cost less than the average by always adding an arc to a partial tour, such that the average cost of a tour containing all the arcs we have picked so far never increases. We will discuss the GEA for the frequency assignment problem below.

- *Vertex insertion algorithms (VI algorithms).* A VI algorithm starts off with any cycle of length 2. In each step it adds a new vertex to the existing cycle, by deleting one arc on the existing cycle, and adding two new arcs (adjacent to the new vertex to create a new cycle). The choice of arc to delete is made so as to minimise the cost of the new cycle. Note that the choice of vertex to add could be arbitrary. Various well-known VI algorithms have different rules for the choices of new vertex, for example, Farthest/Closest Insertion. Note that this is clearly a polynomial time algorithm as we only need a linear amount of time to insert each new vertex. In [45] Punnen and Kabadi proved that any

VI algorithm always produces a solution with cost less than the average cost of a tour in our instance.

Other well-known algorithms for the asymmetric travelling salesman problem have been shown to have different domination numbers. See [23] (chapter 6) for more examples. Below we list a few examples with small domination number.

- *The Greedy Algorithm* has domination number 1 (see [28]). The greedy algorithm works by repetitively picking the cheapest possible arc, such that the arcs that are picked can always be extended to a tour.

- *Nearest Neighbour* algorithm has domination number 1 (see [28]). The nearest neighbour algorithm works by starting at some vertex and then picking the cheapest arc out of this vertex. We now have a path of length one. The algorithm now picks the cheapest arc out of the last vertex in the path we have built so far, which goes to a vertex which has not been visited yet. When the path has length $n-1$ we add the last arc between the end-points of the path in order to get a tour.

- *Repetitive Nearest Neighbour* algorithm has domination number between $\frac{n}{2}$ and $n-1$. The repetitive nearest neighbour algorithm works exactly like the nearest neighbour algorithm above, except that we try all n possible starting points for the nearest neighbour algorithm. We then pick the best tour produced by the n iterations of the nearest neighbour algorithm.

In [24] Gutin and Yeo proved that there exists a polynomial algorithm for the asymmetric travelling salesman problem with domination number $\Theta((n-1)!)$. However this prove is based on an unpublished result by Häggkvist. Even if we do believe that the result by Häggkvist is true, it will still not give us a practical algorithm, as it would be much too slow (even though it is polynomial). Therefore it remains an open problem to find a practical algorithm with domination number $\Theta((n-1)!)$.

Almost all results above can be extended to the symmetric TSP (STSP). Note that in the STSP there are only $(n-1)!/2$ possible tours, whereas we had $(n-1)!$ tours in the asymmetric travelling salesman problem. Rublineckii [49] proved that Lemma 6.2.1 also holds for STSP, except that we get $(n-2)!/2$, in the case when n is even. We should note that for the repetitive nearest neighbour algorithm we only have the upper bound 2^{n-3} [28] and not $n-1$ as was the case for asymmetric travelling salesman problem.

Recently domination analysis has been adapted for some other combinatorial optimisation problems (see [2, 6, 7, 25, 27, 35]). We will name a few of the results here. For more detail see the cited papers.

- In [2] the authors show that there exists an algorithm for the partition problem whose domination ratio (which is the domination number divided by the total number of feasible solutions) tends to one as the size of the input goes to infinity. The partition problem is the problem of partitioning a set of numbers into two sets, such that the maximum of the sum of the elements in a set is minimised.

- In [22] the authors show that a similar result holds for the multiprocessor scheduling problem. In the multiprocessor scheduling problem we are given a set of numbers and want to partition them into a given number of sets. The purpose is still to minimize the maximum sum of a set.

- In [27], the terms $\mathcal{DOM}$-easy and $\mathcal{DOM}$-hard are defined. They partition combinatorial optimisation problems into two classes in the following way. A combinatorial optimisation problem is said to be $\mathcal{DOM}$-hard if there does not exist a polynomial algorithm with domination number $\Omega(|Sol(I)|/n^k)$, for some constant k, where n is the size of the input and $|Sol(I)|$ is the number of feasible solutions given input I. If such an algorithm exists then the problem is said to be $\mathcal{DOM}$-easy. Several problems have been analysed with respect to these classes.

In Section 6.3 we will give bounds on the domination number of some greedy heuristics for both the fixed spectrum and the minimum span frequency assignment problems.

6.3 Domination analysis of greedy heuristics for the fixed spectrum FAP

The results of this section are based on [35].

We study the fixed spectrum version of the frequency assignment problem. Recall, that the given fixed span of frequencies almost always causes some interference. Throughout this section, we apply the model of interference introduced in Section 2.3 as well as the definition of a constraint matrix problem used in chapter 3.

Given an instance I of a fixed spectrum version of the frequency assignment problem,

$$Sol(I) = \{f : V \to \{0, \ldots, \sigma - 1\}\}.$$

Hence $Sol(I)$ consists of all possible frequency assignment with span σ. Thus $|Sol(I)| = \sigma^n$.

The cost of a solution is the sum of the weights of broken constraints or in the unweighted case, the number of broken constraints. The objective now is to minimise the cost over all possible solutions. Given an assignment f we denote its cost (that is the sum of weights of constraints broken) by $c(f)$.

The initial stage of many algorithms that assign frequencies to transmitters consists of a greedy algorithm, consequently it is useful to have a theoretical method of differentiating between the performance of various greedy methods. In this section we compare a standard greedy algorithm with the greedy expectation algorithm by calculating the domination number for each algorithm. First we give a definition for the standard greedy and the greedy expectation algorithm.

Definition 6.3.1 For the *standard greedy algorithm* (SGA) we assume that we are given a fixed ordering $(v_1, \ldots, v_n)$ of the vertices. Initially all vertices are unas-

signed. At each stage of the algorithm we assign a frequency to one vertex. Once an assignment is made, it is not changed. We begin by assigning v_1 with frequency 0. Let w_{ij} be the cost associated with breaking the constraint involving the frequencies assigned to vertices v_i and v_j. In the ith stage of the algorithm, a frequency is assigned to v_i by finding the smallest possible frequency such that $\sum_{j=1}^{i-1} w_{ij}x_{ij}$ is minimised, where $x_{ij} = 1$ if constraint c_{ij} between vertex v_i and vertex v_j is broken, otherwise $x_{ij} = 0$. Clearly in the case when all weights are equal to one, this corresponds to choosing a frequency minimising the number of violated constraints with previously assigned transmitters.

Definition 6.3.2 The *greedy expectation algorithm* is the same as the standard greedy algorithm in that vertices are assigned one after another in the specified order $(v_1, \ldots, v_n)$ and that assignments, once selected, remain fixed. The algorithm differs in the way in which the frequency is selected.

In the ith stage v_i is assigned with the smallest possible frequency such that

$$\left(\sum_{j=1}^{i-1} w_{ij}x_{ij} + \sum_{j=i+1}^{n} w_{ij}p_{ij}\right)$$

is minimised, where p_{ij} is the probability that the constraint between vertex v_i and vertex v_j is broken if vertex v_j is assigned a frequency chosen uniformly at random from $\{0, 1, \ldots, \sigma - 1\}$. Again in the case when all weights are equal to one, this corresponds to minimising the sum of the number of violated constraints with previously assigned transmitters and the expected number of constraints broken when the remaining transmitters are assigned uniformly at random.

In the following subsections we will give bounds on the domination number for the frequency assignment problem of both the greedy expectation algorithm and the standard greedy algorithm.

6.3.1 Domination number of the GEA for the FAP

After having defined the greedy expectation algorithm for the FAP we now give a lower bound for its domination number:

Theorem 6.3.3 The domination number of the greedy expectation algorithm for the fixed spectrum version of the frequency assignment problem is at least $\sigma^{n-\lceil \log_2 n \rceil - 1}$.

Before proving the theorem we first state and prove the following lemma relating the cost of the solution produced by the GEA to the expected cost of a solution generated uniformly at random.

Lemma 6.3.4 Let $\tilde{f}$ be an assignment generated uniformly at random using frequencies from $\{0, ..., \sigma - 1\}$ and let A be the event that $\tilde{f}$ assigns frequencies $x_1, ..., x_{k-1}$ to transmitters $v_1, ..., v_{k-1}$ respectively. Then there exists a frequency j_0 such that

$$E\left(c(\tilde{f})|A, \tilde{f}(v_k) = j_0\right) \leq E\left(c(\tilde{f})|A\right). \tag{6.1}$$

In other words, if we have already assigned frequencies to transmitters $v_1, v_2, \ldots, v_{k-1}$, then we can also assign a frequency to v_k, such that the average cost of a solution, when randomly assigning all unassigned vertices a frequency, does not increase.

Proof: Using the formula of total expectation [21] we obtain

$$E\left(c(\tilde{f})|A\right) = \sum_j E\left(c(\tilde{f})|A, \tilde{f}(v_k) = j\right) P\left(\tilde{f}(v_k) = j|A\right).$$

Let j_0 be the frequency j minimising $E\left(c(\tilde{f})|A, \tilde{f}(v_k) = j\right)$ then

$$\sum_j E\left(c(\tilde{f})|A, \tilde{f}(v_k) = j\right) P\left(\tilde{f}(v_k) = j|A\right) \geq E\left(c(\tilde{f})|A, \tilde{f}(v_k) = j_0\right).$$

□

Proof of Theorem 6.3.3: Let $\tilde{f}$ be an assignment generated uniformly at random. At each step the greedy expectation algorithm finds a frequency j_0 satisfying (6.1). Hence using induction and Lemma 6.3.4 we can show that the algorithm produces a solution with cost at most $E(c(\tilde{f}))$. Thus the domination number of the GEA is at least the number of solutions with cost at least $E(c(\tilde{f}))$. We now compute the number of these solutions.

We regard an assignment as an n-tuple of elements from $\{0,\ldots,\sigma-1\}$. Next we define the addition of two assignments by adding the components modulo σ. This gives the set of assignments the structure of the group $\mathbb{Z}_\sigma^n$.

Now we consider a collection of bipartitions of the vertex set V into (X_i, Y_i) for $i = 1,\ldots,\lceil \log_2 n \rceil$, such that for every edge e there is some i such that e joins a vertex in X_i to a vertex in Y_i. In addition we require that $\bigcap_{i=1}^{n} Y_i \neq \emptyset$. Given vertex set $V = \{v_1,\ldots,v_n\}$, one way to do this is as follows. Define X_i by letting $v_k \in X_i$ if and only if the i-th least significant bit in the binary representation of $k-1$, is equal to 1. Let $Y_i = V \setminus X_i$.

We consider the multiset $\hat{H}$ of all assignments f such that $f(v_k) = \left(\left(\sum_{i:v_k \in X_i} a_i\right) + b\right) \bmod \sigma$ as $a_1,\ldots,a_{\lceil \log_2 n \rceil}$ and b run through all possible combinations of values from $\{0,\ldots,\sigma-1\}$. Thus $|\hat{H}| = \sigma^{\lceil \log_2 n \rceil + 1}$. We claim that the mean cost of an assignment in $\hat{H}$ equals $E(c(\tilde{f}))$. To see this first observe that choosing an assignment uniformly at random from $\hat{H}$ is the same procedure as selecting an assignment f by setting $f(v_k) = \left(\sum_{i:v_k \in X_i} A_i + B\right) \bmod \sigma$ where $A_i,\ldots,A_{\lceil \log_2 n \rceil}$ and B are independent random variables taking the values $0,\ldots,\sigma-1$ uniformly at random.

We now consider the joint distribution of the channels assigned to the endpoints u, v of an edge. Suppose without loss of generality that $u \in X_1$, $v \in Y_1$. Then

$$
\begin{aligned}
&Pr\left((f(u),f(v)) = (i,j)\right) \\
&= \sum_{a_2,\ldots,a_{\lceil \log_2 n \rceil}} Pr\left((f(u),f(v)) = (i,j), A_2 = a_2,\ldots,A_{\lceil \log_2 n \rceil} = a_{\lceil \log_2 n \rceil}\right) \\
&= \sum_{a_2,\ldots,a_{\lceil \log_2 n \rceil}} Pr\left((f(u),f(v)) = (i,j) \,|\, A_2 = a_2,\ldots,A_{\lceil \log_2 n \rceil} = a_{\lceil \log_2 n \rceil}\right) \\
&\quad \cdot Pr\left(A_2 = a_2,\ldots,A_{\lceil \log_2 n \rceil} = a_{\lceil \log_2 n \rceil}\right).
\end{aligned}
$$

Given the values of $A_2,\ldots,A_{\lceil \log_2 n \rceil}$ there is one possible choice for A_1 and B, giving $f(u) = i$, $f(v) = j$. Hence the conditional probability in the sum is $1/\sigma^2$ and consequently when we carry out the summation we also obtain $1/\sigma^2$.

Therefore the probability that the constraint corresponding to edge e is broken is the same as in an assignment where all the channels are chosen uniformly at

random. Thus by linearity of expectation the mean cost of an assignment in $\hat{H}$ equals $E(c(\tilde{f}))$.

Let H be obtained by removing duplicates from $\hat{H}$. Now H is a subgroup of $\mathbb{Z}_\sigma^n$, because of the following. If $u, v \in \hat{H}$, then $u + v \in \hat{H}$, as we can sum the a_i's and b used to generate u with the a_i's and b used to generate v, in order to get the a_i's and b used to generate $u + v$.

Let $C_1, ..., C_k$ be the cosets of H. Since $|H| \leq \sigma^{\lceil \log_2 n \rceil + 1}$, this implies that $k \geq \sigma^{n - \lceil \log_2 n \rceil - 1}$.

Let g_i be the assignment in $\mathbb{Z}_\sigma^n$, such that $C_i = g_i + H$. Now let $\hat{C}_i$ be the multiset $g_i + \hat{H}$. Given an edge e, if we choose an element g of $\hat{C}_i$ uniformly at random, the probability that e is broken in g is the same as the probability that e is broken in $\tilde{f}$. Therefore the mean cost of an assignment in $\hat{C}_i$ is $E(c(\tilde{f}))$, which implies that at least one assignment has cost greater than or equal to $E(c(\tilde{f}))$. Since there are at least $\sigma^{n - \lceil \log_2 n \rceil - 1}$ cosets (which by definition are disjoint), at least $\sigma^{n - \lceil \log_2 n \rceil - 1}$ assignments have cost greater than or equal to $E(c(\tilde{f}))$. □

6.3.2 Domination number of the SGA for the FAP

In this section we give an upper bound on the domination number of the standard greedy algorithm for the frequency assignment problem. The basic idea is to construct a graph on which the standard greedy algorithm works very badly and then use Chernoff type bounds (see Theorem 6.3.5 below) to show that the probability that an assignment chosen uniformly at random, performs as badly as the assignment produced by the standard greedy algorithm, decreases exponentially with the size of the graph.

We will use the following theorem, originally due to Hoeffding [31]

Theorem 6.3.5 Let the random variables $X_1, \ldots, X_N$ be independent, with $a_k \leq X_k \leq b_k$ for each k, for suitable constants a_k, b_k. Let $S = \sum_{k=1}^{N} X_k$ and let $\mu = E[S]$. Then for any $t \geq 0$,

$$P\left(S - \mu \geq t\right) \leq e^{-2t^2 / \sum_{k=1}^{N} (b_k - a_k)^2}.$$

Using this result we are able to show

Theorem 6.3.6 The domination number of the standard greedy algorithm for the frequency assignment problem is at most $\sigma^n e^{-\frac{n}{3600}}$.

Before proving the main theorem we establish the following lemma.

Lemma 6.3.7 Let $\sigma \geq 3$ be fixed and e be an edge in a constraint graph having constraint

$$c = \begin{cases} \frac{\sigma+1}{2} & \text{if } \sigma \text{ is odd,} \\ \frac{\sigma+2}{2} & \text{if } \sigma \text{ is even.} \end{cases}$$

Suppose the endpoints of e are labelled by choosing labels independently and uniformly at random from $\{0, 1, \ldots, \sigma - 1\}$. Then the probability that the constraint, corresponding to edge e, is broken is at most $\frac{7}{8}$.

Proof: Case 1: σ is odd, $\sigma \geq 3$.

Out of all possible assignments the endpoints of e can receive, the number of assignments when the constraint is not broken is

$$2\sum_{i=1}^{\frac{\sigma-1}{2}} i = \left(\frac{\sigma-1}{2}\right)\left(\frac{\sigma+1}{2}\right).$$

From this it follows that the probability that the constraint corresponding to edge e is broken is

$$1 - \frac{\sigma^2 - 1}{4\sigma^2} = 1 - \frac{1}{4} + \frac{1}{4\sigma^2} \leq \frac{7}{8}.$$

Case 2: σ is even, $\sigma > 3$.

Here, the number of assignments leading to an unbroken constraint on edge e is

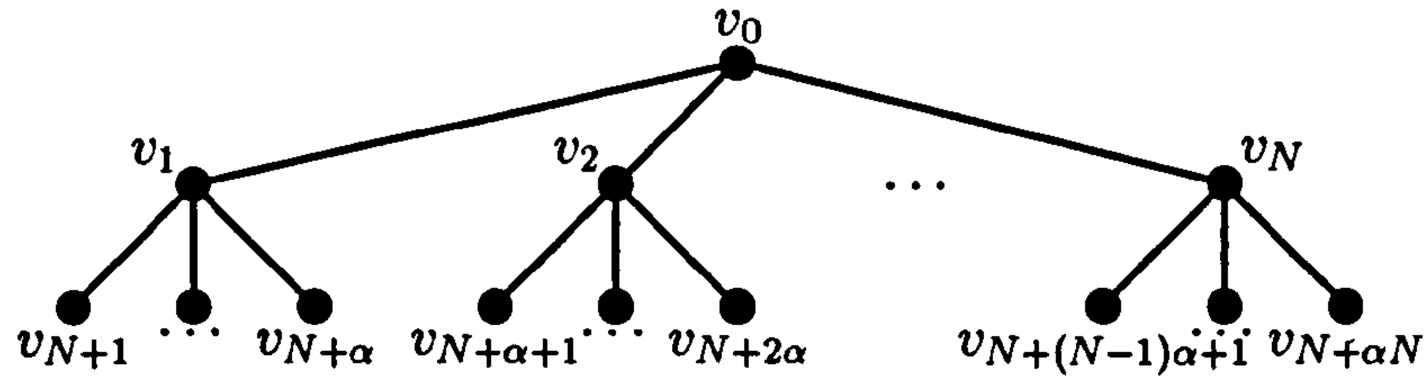
$$2\sum_{i=1}^{\frac{\sigma-2}{2}} i = \left(\frac{\sigma-2}{2}\right)\left(\frac{\sigma}{2}\right).$$

The probability that edge e is broken is

$$1 - \frac{\sigma^2 - 2\sigma}{4\sigma^2} = 1 - \frac{1}{4} + \frac{1}{2\sigma} \leq \frac{7}{8}.$$

□

Figure 6.1: The graph G_n



Proof of Theorem 6.3.6: Let σ be fixed and for each n consider the graph, $G_n = (V, E)$, where $V = \{v_0, \ldots, v_{n-1}\}$. Choose $\alpha \in \mathbb{N}$ so that $8 \leq \alpha \leq n-2$, let $N = \lfloor \frac{n-1}{\alpha+1} \rfloor$, and now let

$$E = \left\{ v_0 v_{1+k}, v_{1+k} v_{N+k\alpha+1}, \ldots, v_{1+k} v_{N+(k+1)\alpha} : k = 0, \ldots, N-1 \right\}.$$

The constraint on any edge incident with v_0 is

$$c_1 = \begin{cases} \frac{\sigma-1}{2} & \text{if } \sigma \text{ is odd,} \\ \frac{\sigma}{2} & \text{if } \sigma \text{ is even.} \end{cases}$$

The constraint on all the other edges is

$$c_2 = \begin{cases} \frac{\sigma+1}{2} & \text{if } \sigma \text{ is odd,} \\ \frac{\sigma+2}{2} & \text{if } \sigma \text{ is even.} \end{cases}$$

G_n is shown in Figure 6.1. Note that if $n-1$ is not a multiple of $\alpha+1$ then $N + \alpha N < n-1$, and so the vertices $v_{N+\alpha N+1}, v_{N+\alpha N+1}, \ldots, v_{n-1}$ will be isolated vertices.

If we apply the standard greedy algorithm to G_n, labelling vertices in the order $v_0, \ldots, v_n$ then v_0 receives label 0, and $v_1, \ldots, v_N$ receive label c_1. This means that the other constraints cannot be satisfied and so αN constraints are broken.

Now consider an assignment $\tilde{f}$ where all the labels are chosen independently and uniformly at random from $\{0, \ldots, \sigma-1\}$. For $i = 1, \ldots, N$ and $j = 1, \ldots, \alpha$ let $X_{i,j} = 1$ if the constraint corresponding to $v_i v_{N+(i-1)\alpha+j}$ is broken and 0 otherwise.

For $i = 1, \ldots, N$, let $X_i = \sum_{j=1}^{\alpha} X_{i,j}$ and let $Y_i = 1$ if the constraint corresponding to $v_0 v_i$ is broken and 0 otherwise. Since, for $r \neq s$, there are no edges between

vertices that are endpoints of edges that may contribute to X_r and X_s it follows that $\{X_1, \ldots, X_N\}$ forms a collection of independent identically distributed random variables. Let $S = \sum_{i=1}^{N} X_i$ and $T = \sum_{i=1}^{N} Y_i$. From 6.3.7 we get $E(S) \leq \frac{7}{8}\alpha N$.

The probability that $\tilde{f}$ breaks at least as many constraints as the standard greedy algorithm is at most

$$P(S+T \geq \alpha N) \leq P(S \geq (\alpha-1)N) = P\left(S - \frac{7}{8}\alpha N \geq (\alpha-1)N - \frac{7}{8}\alpha N\right).$$

Applying Theorem 6.3.5, providing $\alpha \geq 8$ and therefore $(\alpha-1)N - \frac{7}{8}\alpha N \geq 0$, we get

$$\begin{aligned} P\left(S - \frac{7}{8}\alpha N \geq (\alpha-1)N - \frac{7}{8}\alpha N\right) &\leq P\left(S - E(S) \geq (\alpha-1)N - \frac{7}{8}\alpha N\right) \\ &\leq e^{-2\frac{(\alpha/8-1)^2N^2}{\alpha^2 N}}. \end{aligned}$$

Hence the domination number of the standard greedy algorithm is at most $\sigma^n e^{-2(\frac{1}{8}-\frac{1}{\alpha})^2 N}$. Recall from the definition of N that $\sigma^n e^{-2(\frac{1}{8}-\frac{1}{\alpha})^2 N} = \sigma^n e^{-2(\frac{1}{8}-\frac{1}{\alpha})^2 \lfloor \frac{n-1}{\alpha+1} \rfloor}$.

We claim that this is at most $\sigma^n e^{-(\frac{1}{8}-\frac{1}{\alpha})^2 \frac{n}{\alpha+1}}$. To see this write $n-1 = q(\alpha+1)+r$, where $q, r \in \mathbb{Z}$ such that $0 \leq r \leq \alpha$. Note that since $n \geq \alpha+2$ we have $q \geq 1$.

Now,

$$\begin{aligned} 2\lfloor \frac{n-1}{\alpha+1} \rfloor - \frac{n}{\alpha+1} &= 2\lfloor \frac{q(\alpha+1)+r}{\alpha+1} \rfloor - \frac{q(\alpha+1)+r+1}{\alpha+1} \\ &= q - \frac{r+1}{\alpha+1} \\ &\geq q-1 \\ &\geq 0. \end{aligned}$$

Hence,

$$\sigma^n e^{-2(\frac{1}{8}-\frac{1}{\alpha})^2 N} \leq \sigma^n e^{-(\frac{1}{8}-\frac{1}{\alpha})^2 \frac{n}{\alpha+1}}. \tag{6.2}$$

Since the above is true for all α such that $8 \leq \alpha \leq n-2$ we may choose α to minimise the right hand side of 6.2.

This is equivalent to finding the maximum of $(\frac{1}{8} - \frac{1}{\alpha})^2 \frac{n}{\alpha+1}$. This occurs for some $\alpha \in (24, 25)$.

Setting $\alpha = 24$ shows that providing $n \geq 26$, the domination number of the standard greedy algorithm is at most $\sigma^n e^{-\frac{n}{3600}}$. □

Corollary: For fixed σ and large enough n the domination number of the SGA is strictly less than the domination number of the GEA.

6.4 Domination analysis of greedy heuristics for the minimum span FAP

6.4.1 Introduction and definition

In Section 6.3 we studied the domination number of greedy heuristics for the fixed spectrum version of the frequency assignment problem. Now, looking at the minimum span version of the frequency assignment problem, we have to find a different approach to the problem.

Whereas the fixed spectrum version of the frequency assignment problem minimises the number of constraints that are broken between transmitters, the minimum span problem does not allow any constraints to be broken and therefore minimises the spectrum of frequencies needed in order to achieve this.

We are given a constraint matrix problem with constraint matrix $C = (c_{ij})$ and underlying graph G.

We first consider how to define $Sol(I)$ for the minimum span frequency assignment problem. It is not possible to let $Sol(I)$ contain all feasible frequency assignments since the number of these is unbounded. Clearly many of these are extremely poor solutions since they use frequencies that are larger than the sum of all the constraints. Our definition of $Sol(I)$ is motivated by the observation from Chapter 3.2 which we repeat here for convenience, [29].

Theorem 6.4.1 (as stated earlier) (Based on the theorem of Gallai, Roy and Vitaver [54]) The *minimum span* of a constraint matrix problem with underlying graph G is the minimum over all acyclic orientations, ω, of the length of the longest directed path in (G,ω).

Therefore we can define $Sol(I)$ to be the set of all orderings ω of the vertices of an instance I or equivalently an acyclic orientation of the edges of the complete graph on the vertices of I. This definition of $Sol(I)$ ensures that $|Sol(I)|$ is the same for all instances with the same number of vertices. The cost of a solution is the length of the longest path in (G,ω) or equivalently the minimum span of a feasible frequency assignment respecting ω (in the sense of Theorem 3.2.1).

We will now introduce an intuitively reasonable greedy algorithm that returns an orientation ω of the given initial graph G. Unfortunately this algorithm will turn out to have quite a small domination number for the minimum span frequency assignment problem.

6.4.2 The algorithm

In this section we present a heuristic that uses Theorem 3.2.1 in a very straightforward manner and has domination number at most $2(n-1)$.

The algorithm we are about to describe returns an orientation ω of the edges of K_n. We are basically only asking two things from the algorithm:

- it may not create any cycles;
- at each step the algorithm has to minimise the cost of the longest path.

The algorithm takes as input a constraint matrix problem with constraint matrix $C = (c_{ij})$. Let G denote the underlying graph of C. Our algorithm computes an acyclic orientation on a complete graph with vertices $\{v_1,\ldots,v_n\}$ by first orienting all the edges that are present in G and then those that are not in G. The first part of the algorithm consists of $n-1$ stages. In the ith stage edges joining v_{i+1} to vertices in $\{v_1,\ldots,v_i\}$ are oriented one at a time in such a way that no cycles are created

and so that whenever there is a choice of which orientation to give to an arc, the one minimising the length of the longest path in the digraph constructed so far is chosen.

A precise description of the first part of the algorithm is stated in Algorithm 1.

Algorithm 1 Output: Acyclic orientation

Input $C = (c_{ij})$.
Let G be the underlying graph of C.
Let Q be the empty digraph on vertices $\{v_1, \ldots, v_n\}$.
for $j = 2$ to n **do**
  for $i = 1$ to $j - 1$ **do**
    if $\{v_i, v_j\}$ is an edge of G **then**
      if Q contains a directed path from v_i to v_j **then**
        add the arc (v_i, v_j) to Q.
      else if Q contains a directed path from v_j to v_i **then**
        add the arc (v_j, v_i) to Q.
      else if $LP(Q \cup (v_i, v_j)) \leq LP(Q \cup (v_j, v_i))$ **then**
        add the arc (v_i, v_j) to Q.
      else
        add the arc (v_j, v_i) to Q.
      end if
    end if
  end for
end for

After that the algorithm puts in directed edges of weight zero still without creating cycles in order to make the graph complete. We need this property to comply with our definition of the domination number.

The next two results show that the solution produced by the algorithm is not made worse by requiring it to compute a total ordering of the vertices rather than

just an acyclic orientation of the edges of G.

Proposition 6.4.1 Suppose D is a weighted acyclic digraph and $v_i, v_j \in V(D)$ such that neither (v_i, v_j) nor (v_j, v_i) is an arc of D. Then it is possible to add one of the arcs of zero length joining either v_i to v_j or v_j to v_i without increasing the length of the longest path in D.

The following corollary follows by applying Proposition 6.4.1 several times.

Corollary 6.4.2 Given a weighted acyclic digraph D, it is possible to add arcs of length zero to form a complete acyclic digraph D' in such a way that the longest path in D' is no longer than the longest path in D.

To prove the proposition it suffices to show that if v_1, v_2 are vertices that are not joined by an edge then it is possible to insert an arc of weight zero between them, without increasing the length of a longest path.

Proof of Proposition 6.4.1: Consider two vertices v_1 and v_2. Let α be the length of the longest path ending in v_1, β be the length of the longest path starting at v_1. Similarly, let γ be the length of the longest path ending in v_2 and δ be the length of the longest path starting at v_2.

Case 1: There is a path in D from v_1 to v_2 going through at least one other vertex. Inserting the arc (v_1, v_2) with zero length will not change the length of the longest path as it will not appear in the longest path. The same applies if there is a path from v_2 to v_1.

Case 2: There is neither a path from v_1 to v_2 nor a path from v_2 to v_1 in D.

If adding the arc (v_1, v_2) with zero length would increase the length of the longest path in D, we must have

$$\alpha + \delta > \max\{\alpha + \beta, \gamma + \delta\}.$$

Similarly, if adding the arc (v_2, v_1) with zero length would increase the length of the longest path in D, we must have

$$\gamma + \beta > \max\{\alpha + \beta, \gamma + \delta\}.$$

Therefore

$$\alpha + \beta + \gamma + \delta > 2\max\{\alpha + \beta, \gamma + \delta\} \geq \alpha + \beta + \gamma + \delta.$$

This is a contradiction. □

Now we are given a complete digraph D', which was built by orienting and adding (with weight zero) all possible edges to G_n. Note that D' yields a hamiltonian path since it is complete and acyclic.

6.4.3 Domination number

Theorem 6.4.3 The domination number of the greedy algorithm for the minimum span frequency assignment problem on graphs with n vertices is at most $2(n-1)$.

Proof: Consider the constraint matrix problem with

$$\begin{aligned} c_{i+1,i} = c_{i,i+1} &= 2^{i-1} \quad \text{for } i = 1, \ldots, n-1, \\ c_{i+2,i} = c_{i,i+2} &= 2^{i-1} \quad \text{for } i = 1, \ldots, n-2, \\ c_{ij} &= 0 \qquad \text{if } |i-j| \geq 3. \end{aligned}$$

We claim that if $i < j$ and $c_{ij} > 0$ then the greedy algorithm will add the arc (v_i, v_j). To prove this we use induction on i. First note that the greedy algorithm begins by adding the arc (v_1, v_2). Next consider the edge $\{v_1, v_3\}$. If this edge is oriented from v_3 to v_1 then there is a path v_3, v_1, v_2 of length $2^0 + 2^0 = 2$, but if we orient the edge from v_1 to v_3 the length of the longest path is only $2^0 = 1$. So we add the arc (v_1, v_3).

Suppose then that the arcs (v_1, v_2), (v_1, v_3), (v_2, v_3), ..., (v_{i-1}, v_i), (v_{i-1}, v_{i+1}) are added by the greedy algorithm.

The algorithm next considers the edge corresponding to the constraint $c_{i,i+1}$. If the arc (v_i, v_{i+1}) is added, then the longest path is $v_1 v_2 \ldots v_{i-1} v_i v_{i+1}$ with length $2^0 + 2^1 + \ldots + 2^{i-2} + 2^{i-1} = 2^i - 1$.

On the other hand if the arc (v_{i+1}, v_i) is added, then the longest path is $v_1 v_2 \ldots v_{i-1}$ $v_{i+1} v_i$ with length $2^0 + 2^1 + \ldots + 2^{i-2} + 2^{i-1} = 2^i - 1$.

Since the length of the longest path is equal, regardless of how we orient the edge $\{v_i, v_{i+1}\}$, the algorithm will add the arc (v_i, v_{i+1}).

The next edge we consider is the edge $\{v_i, v_{i+2}\}$. Suppose the arc (v_i, v_{i+2}) is added. We will then get the path $v_1 v_2 \dots v_{i-1} v_i v_{i+2}$ with length $2^0 + 2^1 + \dots + 2^{i-2} + 2^{i-1} = 2^i - 1$.

Suppose now that the arc (v_{i+2}, v_i) is added. We will then get the path $v_{i+2} v_i v_{i+1}$ with length $2^{i-1} + 2^{i-1} = 2^i$.

Therefore the edge will be oriented from v_i to v_{i+2} and hence the claim is true by induction.

The algorithm constructs an acyclic orientation in which the longest path is $v_1 v_2 \dots v_n$ which has length $2^{n-1} - 1$. Denote this path by Q.

To calculate the domination number of the algorithm on this instance we must find the number of acyclic orientations that include a path P with length at least $2^{n-1} - 1$.

To do this we first consider which (undirected) edges must be present in such a path. We claim that

1. P must contain the edge $\{v_{n-1}, v_n\}$ and
2. for $i = 1, \dots, n-2$ P must contain exactly one of the edges $\{v_i, v_{i+1}\}$ and $\{v_i, v_{i+2}\}$.

To see that P must contain the edge $\{v_{n-1}, v_n\}$, note that the total length of all the other edges is $2(2^{n-2} - 1) = 2^{n-1} - 2$ which is strictly less than the length of Q.

In order to prove the second part of the claim we proceed by induction. Suppose that P contains $\{v_{n-1}, v_n\}$ and for $i = k, \dots, n-2$, P contains exactly one of the edges $\{v_i, v_{i+1}\}$ and $\{v_i, v_{i+2}\}$.

If P contains both $\{v_{k-1}, v_k\}$ and $\{v_{k-1}, v_{k+1}\}$ then at this point we have shown that P definitely contains $n - k + 2$ edges, but the endpoints of these edges are members of $\{v_{k-1}, \dots, v_n\}$ so the edges must include a cycle.

On the other hand if P contains neither $\{v_{k-1}, v_k\}$ nor $\{v_{k-1}, v_{k+1}\}$ then the total length of all the edges definitely in P together with all the edges with at least one endpoint in $\{v_1, \ldots, v_{k-2}\}$ is $2^{n-2} + \ldots + 2^{k-1} + 2(2^{k-3} + \ldots + 1) = 2^{n-1} - 2$. This again is strictly less than the length of Q. Hence, the claim follows by induction.

We have shown that if we have a path P with $c(P) \geq c(Q)$, P has to include the edge $\{v_{n-1}, v_n\}$ as well as exactly one of the edges $\{v_i, v_{i+1}\}$ or $\{v_i, v_{i+2}\}$.

This leaves us to show that there exist $2(n-1)$ paths with the properties of P.

Such a path P satisfying properties (1) and (2) must include $n-1$ edges. Hence P completely determines an ordering of the vertices of G.

Therefore the domination number of the algorithm is equal to the number of directed paths P in G satisfying (1) and (2).

Let $A = \{i : v_iv_{i+1} \in P \text{ or } v_{i+1}v_i \in P, i < n-1\}$. If A is non-empty, let $k = \max\{i : i \in A\}$, if A is empty, let $k = n-1$.

Then $1 \leq k \leq n-1$. We claim that for each value of k there are two paths satisfying (1) and (2).

If $k = n-1$ then if n is even the only two paths are $v_1v_3v_5 \ldots v_{n-1}v_nv_{n-2}v_{n-4} \ldots v_4v_2$ and the reverse of this path. If n is odd then we get the path $v_1v_3v_5 \ldots v_nv_{n-1}v_{n-3}$ $v_{n-5} \ldots v_4v_2$ and the reverse of this path.

So now consider the case when $k < n-1$. Assume that $(v_k, v_{k+1}) \in P$. Then by the definition of k, (v_{k+1}, v_{k+3}), (v_{k+2}, v_{k+4}), ..., (v_{n-2}, v_n), $(v_{n-1}, v_n) \in P$. We will now show by induction that $\{v_i, v_{i+1}\} \in P$ for all $i = 1, 2, \ldots, k$. This is clearly true for $i = k$, by the above, so now assume that it is true for all $i = j, j+1, \ldots, k$, where $1 < j \leq k$. If $(v_{j-1}, v_j) \notin P$, then by (2) $(v_{j-1}, v_{j+1}) \in P$. However then v_{j+1} would be the endpoint of three edges of P, a contradiction. This completes the induction proof, which implies that $P = v_1v_2v_3 \ldots v_kv_{k+1}v_{k+3}v_{k+5} \ldots v_{n-1}v_nv_{n-2}v_{n-4} \ldots v_{k+4}v_{k+2}$ when $k+n$ is even and $P = v_1v_2v_3 \ldots v_kv_{k+1}v_{k+3}v_{k+5} \ldots v_nv_{n-1}v_{n-3} \ldots v_{k+4}v_{k+2}$ when $k+n$ is odd. If we had assumed that $(v_{k+1}, v_k) \in P$ above then we would have got the reverse of the paths we have just found.

So for all values of k there are exactly two paths which satisfy (1) and (2). This completes the proof. □

Next it would be interesting to know a good algorithm to give a lower bound on the domination number for the minimum span frequency assignment problem. However we did not have enough time to consider this within this thesis and state it instead as an open problem.

Chapter 7

Conclusion

We close by summarising the main results of this thesis and more importantly, outlining some open questions.

In Chapter 3 we discussed the constraint matrix problem and described a dynamic programming approach that enabled us to solve the constraint matrix problem for a variety of classes of underlying graphs. It seems very likely that this approach may be used for other classes of graph such as chains of cycles of arbitrary size and perhaps graphs of pathwidth two. However the status of the following seems very unclear.

Open Problem 1 What is the complexity of the constraint matrix problem when restricted to the case where the underlying graph has treewidth at most two?

In Chapter 4 we introduced the one-close neighbour problem, where we wish to find a frequency assignment f such that for each vertex v_i,

$$|f(v_i) - f(v_j)|_\sigma \geq p$$

for all $v_j \in N(v_i)$, and

$$|f(v_i) - f(v_j)|_\sigma \geq p + q$$

for all but one $v_j \in N(v_i)$. We found optimal solutions for a number of classes of graphs. It is easy to see that the one-close-neighbour problem is NP-hard in the case when $p = 1$ and $q = 0$, as this would correspond to vertex colouring. However

we do not know anything about the complexity in less obvious situations such as the following.

Open Problem 2 What is the complexity of the one-close-neighbour problem in the case that $p = q = 1$?

As we mentioned in Chapter 5, the $L(2,1)$-labelling problem has been very well-studied. The next problem is well-known and particularly frustrating.

Open Problem 3 Is there a polynomial time algorithm to compute the $L(2,1)$-span of a series-parallel network?

This question formed the motivation for the most important results of the thesis, namely determining bounds on the $L(2,1)$-span for outerplanar graphs and finding a polynomial time algorithm to compute the $L(2,1)$-span for outerplanar graphs. At this stage it is unclear whether the methods used here can be modified to deal with series-parallel networks. One other problem, which we have not considered and is well-known but seems worth mentioning is the following.

Open Problem 4 What is the complexity of computing the $L(p,q)$ span for trees?

Our final chapter discusses the domination number of an algorithm and gives bounds on the domination number of two greedy heuristics for the fixed spectrum problem. The situation with the minimum span problem seems less clear.

Open Problem 5 Find a heuristic for the minimum span frequency assignment problem that has a large domination number.

Bibliography

[1] K.I. Aardal, S.P.M. van Hoesel, A.M.C.A. Koster, C. Mannino and A. Sassano, Models and Solution Techniques for Frequency Assignment Problems, Konrad-Zuse-Zentrum für Informationstechnik Berlin, ZIB-Report 01-40 (2001).

[2] N. Alon, G. Gutin and M. Krivelevich, Algorithms with large domination ratio, *submitted* (2003).

[3] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees, *Discrete Appl. Math.* 23 (1989) 11-24.

[4] S. Arnborg, D.G. Corneil and A. Proskurowski, Complexitiy of finding embeddings in a k-tree, *SIAM J. Alg. Disc. Meth.* 8 (1987) 277-284.

[5] J. Bang-Jensen and G. Gutin, *Digraphs: Theory, Algorithms and Applications*, Springer, London, 2000.

[6] D. Ben-Arieh, G. Gutin, M. Penn, A. Yeo and A. Zverovitch, Transformations of Generalized ATSP into ATSP: experimental and theoretical study, *To appear in Oper. Res. Letters.*

[7] D. Berend and S.S. Skiena, Combinatorial dominance guarantees for heuristic algorithms, Manuscript, 2002.

[8] H.L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, *SIAM J. Comput.* 25 (1996) 1305-1317.

[9] H.L. Bodlaender, A tourist guide through treewidth, *Acta Cybernet.* 11 (1993) 1-21.

[10] H.L. Bodlaender, Treewidth: Algorithmic techniques and results, *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1295 (1997) 29-36.

[11] H.L. Bodlaender, T. Kloks, B. Tan and J. van Leeuwen, λ-Coloring of Graphs, *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1770 (2000) 395-406.

[12] R. Borndörfer, A. Eisenblätter, M. Grötschel and A. Martin, Frequency Assignment in Cellular Phone Networks, *Ann. Oper. Res.* 76 (1998) 73-93.

[13] T. Calamoneri and R. Petreschi, The $L(2,1)$-Labeling of Planar Graphs, *Proc. of 5th ACM DIAL-M* (2001) 28-33.

[14] G.J. Chang and D. Kuo, The L(2,1)-labeling problem on graphs, *SIAM J. Discrete Math.* 9 (1996) 317-338.

[15] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms* (The MIT, Massachusetts, 1992).

[16] R. Diestel, *Graph Theory* (Springer-Verlag, New York, 2000).

[17] A. Eisenblätter, Frequency Assignment in GSM Networks: Models, Heuristics and Lower Bounds, Ph.D. Thesis, Technische Universität Berlin, 2001.

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

[19] F. Glover and A. Punnen, The traveling salesman problem: New solvable cases and linkages with the development of approximation algorithms, *J. Oper. Res. Soc.* 48 (1997) 502-510.

[20] J.R. Griggs and R.K. Yeh, Labeling graphs with a condition at distance 2, *SIAM J. Discrete Math.* 5 (1992) 586-595.

[21] G.R. Grimmett and D.R. Stirzaker, *Probability and Random Processes* (Clarendon Press, Oxford, 1992).

[22] G. Gutin, T. Jensen and A. Yeo, Domination analysis for minimum multiprocessor scheduling, *submitted* (2003).

[23] G. Gutin and A.P. Punnen, *Travelling Salesman Problem and Its Variations* (Kluwer Academic Publishers, Dordrecht, 2002).

[24] G. Gutin and A. Yeo, TSP tour domination and Hamilton cycle decomposition of regular digraphs, *Oper. Res. Lett.*, 28 (2001) 107-111.

[25] G. Gutin and A. Yeo, Anti-matroids, *Oper. Res. Lett.* 30 (2002) 97–99.

[26] G. Gutin and A. Yeo, Polynomial algorithms for the TSP and the QAP with a factorial domination number, *Discrete Appl. Math.* 119 (2002) 107-116.

[27] G. Gutin, A. Vainshtein and A. Yeo, Domination Analysis of Combinatorial Optimization Problems, *Discrete Applied Mathematics* 129 (2003) 513-520.

[28] G. Gutin, A. Yeo and A. Zverovich, Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP, *Discrete Appl. Math.* 117 (2002) 81-86.

[29] J. van den Heuvel, personal communication.

[30] J. van den Heuvel, R.A. Leese and M.A. Shepherd, Graph labelling and radio channel assignment, *J. Graph Theory* 29 (1998) 263-284.

[31] W.J. Hoeffding, Probability inequalities for sums of bounded random variables, *J. Amer. Statist. Assoc.* 58 (1963) 713-721.

[32] S. Hurley, D.H. Smith and S.U. Thiel, FASoft: A system for discrete channel frequency assignment, *Radio Sci.* 32 (1997) 1921-1939.

[33] K. Jonas, Graph colouring analogues with a condition at distance two: L(2,1)-labelings and list λ-labelings, Ph.D. Thesis, Dept. of Math., University of South Carolina, Columbia, SC, 1993.

[34] A. W. J. Kolen, A genetic algorithm for frequency assignment, Tech. report, Universiteit Maastricht, 1999.

[35] A.E. Koller and S.D. Noble, Domination analysis of greedy heuristics for the frequency assignment problem, *Discrete Math.* 275 (2004) 331-338.

[36] A.M.C.A. Koster, Frequency Assignment - Models and Algorithms, Ph.D. Thesis, Maastricht University, 1999.

[37] R.A. Leese and S.D. Noble, Cyclic labellings with constraints at two distances, *Electron. J. Combin.* 11 (2004) R16.

[38] C.J.H. McDiarmid, On the span in channel assignment problems: bounds, computing and counting, *Discrete Math.*, 266 (2003) 387-397.

[39] C. J. H. McDiarmid, Discrete mathematics and radio channel assignment, *Recent advances in algorithms and combinatorics*, Springer-Verlag, Berlin, 2002.

[40] C.J.H. McDiarmid, A doubly cyclic channel assignment problem, *Discrete Appl. Math.* 80 (1997) 263-268.

[41] C.J.H. McDiarmid and B.A. Reed, Channel assignment on graphs of bounded treewidth, *Discrete Math.* 273 (2003) 183-192.

[42] R. Montemanni, D.H. Smith and S.M. Allen, Lower bounds for fixed spectrum frequency assignment, *Ann. Oper. Res.* 107 (2001) 237-250 (2002).

[43] C.H. Papadimitriou and K. Steiglitz *Combinatorial Optimization: Algorithms and Complexity* (Dover Publications Inc., Mineola, 1998).

[44] A.P. Punnen, The traveling salesman problem: new polynomial approximation algorithms and domination analysis, *J. Inform. Optim.* 22 (2001) 191-206.

[45] A.P. Punnen and S.N. Kabadi, Domination ananlysis of some heuristics for the asymmetric traveling salesman problem, *Discrete Appl. Math.*, 119 (2002) 117-128

[46] A.P. Punnen, F. Margot and S.N. Kabadi, TSP heuristics: domination analysis and complexity, *Algorithmica* 32 2 (2002) 111-127.

[47] N. Robertson and P.D. Seymour, Graph minors. I. Excluding a forest, *J. Comb. Theory Series B* (1983) 35:39-61.

[48] N. Robertson and P.D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, *J. Comb. Theory Series B* (1986) 7:309-322.

[49] V.I. Rublineckii, Estimates of the accuracy of procedures in the travelling salesman problem, *Numer. Math. Comput. Technol.* IV (1973) 18-23 (Russian).

[50] V.I. Sarvanov, The mean value of the functional of the assignment problem, *Vestsi. Akad. Navuk Belarusi Ser. Fiz.-Mat. Navuk* 143 2 (1976) 111-114.

[51] G. Szekeres and H.S. Wilf, An inequality for the chromatic number of a graph, *J. Comb. Theory* 4 (1968) 1-3.

[52] A. Vince, Star chromatic number, *J. Graph Theory* 12 (1988) 551-559.

[53] R.J. Wilson, *Introduction to Graph Theory* (Addison Wesley Longman Limited, Harlow, 1996).

[54] D.B. West, *Introduction to Graph Theory* (Prentice Hall Inc., Upper Saddle River, 1996).

[55] R.K. Yeh, Labeling graphs with a condition at distance 2, Ph.D. Thesis, Dept. of Math., University of South Carolina, 1990.

Appendix A

Notation

Notation	Meaning
G	a graph
(V, E)	vertex set V and edge set E of G
n	number of vertices in the graph, $\|V\|$
$\{v_i, v_j\}$	vertices v_i and v_j are adjacent, edge between v_i and v_j
(v_i, v_j)	an arc (oriented edge) from vertex v_i to vertex v_j
$N(v)$	neighbourhood of v
$N[v]$	$N(v) \cup v$, closed neighbourhood of v
$d(v)$	vertex degree
$\delta(G)$	minimum vertex degree in G
$\Delta(G)$	maximum vertex degree in G
$\chi(G)$	chromatic number
$\sigma(G)$	span

Notation	Meaning
$\sigma_c(G)$	span, using cyclic channel metric
v_i	ith vertex in an ordered graph
C	constraint matrix
c_{ij}	constraint matrix element
c_{max}	maximum constraint in G
f_i	frequency assigned to transmitter i
d_{ij}	distance between transmitters i and j
f	assignment
$c(f)$	cost of an assignment
w_{ij}	weight put on constraint c_{ij}
$\mathrm{dw}(v_i)$	$\sum_{v_j:\{v_i,v_j\}\in E} c_{ij}$, weighted degree of vertex v_i
ω	acyclic orientation in G