On Sum Coloring of Graphs

by

Mohammadreza Salavatipour

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

On Sum Coloring of Graphs

Mohammadreza Salavatipour
Master of Science
Graduate Department of Computer Science
University of Toronto
2000

The sum coloring problem asks to find a vertex coloring of a given graph G, using natural numbers, such that the total sum of the colors of vertices is minimized amongst all proper vertex colorings of G. This minimum total sum is the chromatic sum of the graph, $\Sigma(G)$, and a coloring which achieves this total sum is called an optimum coloring. There are some graphs for which the optimum coloring needs more colors than indicated by the chromatic number. The minimum number of colors needed in any optimum coloring of a graph is called the strength of the graph, which we denote by $s(G)$. Trivially $\chi(G) \leq s(G)$. In this thesis we present various results about the sum coloring problem. We prove the NP-Hardness of finding the vertex strength for graphs with $\Delta = 6$ and also give some logarithmic upper bounds for the vertex strength of graphs with small chromatic number. We also prove that the sum coloring problem is NP-complete for split graphs. Polynomial time algorithms are presented for the sum coloring of k-split graphs, P_4-reducible graphs, chain bipartite graphs, and cobipartite graphs.

We can extend the idea of sum coloring to edge coloring and define the edge chromatic sum and the edge strength of a graph similarly. We prove that the edge sum coloring and the edge strength problems are both NP-complete for cubic graphs. Also we give a polynomial time algorithm to solve the edge sum coloring problem on trees, and show that using the Monadic Second Order Logic we can solve this problem on partial k-trees with bounded degree in linear time.

Acknowledgements

I do not see the completion of my M.Sc program as a personal achievement; there are many people who have generously helped me along the way. I would like to thank the people who have supported and encouraged me.

First and foremost, I would like to thank my advisor, Derek Corneil. He is an excellent advisor and a great researcher. His deep experience, sharpness, and insight in different areas of graph theory provided me a great source of inspiration. He has helped me with his great suggestions, good questions, and new ideas whenever I was distracted. This thesis would not have been possible without his supervision. I also thank Mike Molloy, the second reader of this thesis, for his helpful comments.

From the very beginning years of my study, I have received great support from my parents. After being far from my family, and from my wife for a while, I realized that I love them more than I could imagine it before. I don't know how to thank my family, and in particular my mother for her unconditional love. And about my wife, I have no words to thank her who is the primary reason for my happy life.

For many useful discussions, I would like to thank my fellow graduate students Mohammad Mahdian, Kiumars Kaveh, and Babak Farzad. I can not forget my primary teachers in university, who directed me to this area of science, Dr. Ebad Mahmoodian and Dr. Mohammad Ghodsi.

Finally, I thank the members of the department who made the department a great place to work and study. I also thank the University of Toronto for financial assistance.

تقدیم به‌همسر عزیزم، مادر مهربانم.

Contents

Chapter 1

Introduction

1.1 Statement of the problem

A broad and rich area of graph theory, which has received much attention in the last few decades, is graph coloring. As evidence of this attention we mention the book by Jensen and Toft [25], in which more than 200 open coloring problems are presented. Generally, coloring of a graph is an assignment of natural numbers to the elements of a graph, such as vertices, such that any two adjacent elements have different colors. Finding a coloring for the vertices of a given graph, using the minimum number of colors, is usually referred to as vertex coloring, or coloring for short, and this minimum number of colors, denoted by $\chi(G)$, is called the *chromatic number* of the graph. There are many other variations of graph coloring. We consider the following kind of coloring, which is called the *sum coloring* problem:

> *For a given graph G, find a proper vertex coloring of G, using natural numbers, such that the total sum of the colors of vertices is minimized amongst all proper colorings of G .*

This minimum total sum of colors is called the *chromatic sum* of G, and is denoted by $\Sigma(G)$. We refer to a vertex coloring whose total sum is $\Sigma(G)$ as an *optimum vertex coloring.*

The notion of a coloring in which we want to minimize the total sum of colors first appeared in 1987 from two different sources. In theoretical graph theory, Kubika [27] in her Ph.D thesis introduced the chromatic sum of a graph with the above notation, as a variation of ordinary vertex coloring. The second source of vertex sum coloring arose from its application in VLSI design. Supowit [36] introduced the *optimum cost chromatic partition* (OCCP) problem, which is very similar to the sum coloring problem.

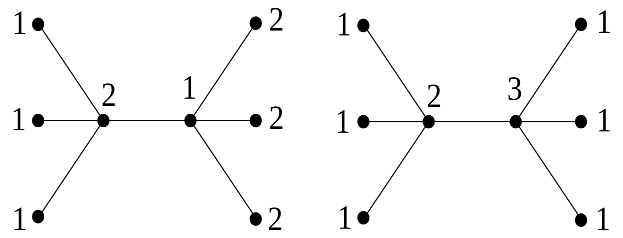

Figure 1.1: A coloring of a tree with total sum 12 using 2 colors, and a coloring with total sum 11 using 3 colors

The OCCP problem asks to find a proper coloring of a graph, using a given set of colors $\{c_1, c_2, \ldots, c_k\}$, such that the total sum of colors is minimized. In other words, this is the same as the definition of sum coloring with the modification that the set of colors is a specific given set, rather than the set of natural numbers.

It is interesting to note that these two source articles have each motivated bodies of research that seem to be unaware of the results arising from the other source. As a consequence there has been some duplication of results in the literature.

Assume that P is an optimum vertex coloring of a graph G with k colors. Let's call the set of vertices having color i, C_i, for $1 \leq i \leq k$. Therefore, by definition, $\Sigma(G) = \sum_{i=1}^{k} i|C_i|$ and it follows immediately that $|C_i| \geq |C_{i+1}|$, for $1 \leq i < k$. Also, it's clear that $k \geq \chi(G)$. One might think that we can obtain an optimum vertex coloring by finding a proper coloring with $\chi(G)$ colors and then assigning color 1 to the largest color class, color 2 to the next largest color class, and so on. However, this does not always yield an optimum vertex coloring, even for trees. For example any vertex 2-coloring of the tree in figure 1.1 gives a total sum of 12 whereas the coloring shown with 3 colors gives a total sum of 11. Therefore, it is not necessarily true that $k = \chi(G)$. So what is the minimum number of colors needed to obtain the chromatic sum of a graph? This is the parameter we call the *strength* of G, and will denote it by $s(G)$. It is trivial that $s(G) \geq \chi(G)$.

As an interesting result, it has been proved in [29] that for any fixed k, almost all trees have strength at least k. This shows that the strength of a graph can be far from its chromatic number, and as a consequence, the optimum vertex coloring might be far from the sum coloring achieved using any vertex $\chi(G)$-coloring.

In edge coloring problems, we are coloring the edges of a graph, rather than the vertices, and two edges that share a vertex must have different colors. Similar to the vertex sum coloring problem we can define the *edge sum coloring* problem, to be an edge coloring, using natural numbers, such that the total sum of the colors of the edges of

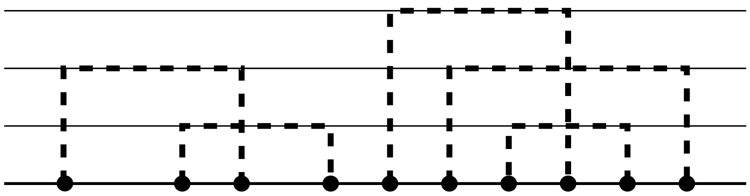

Figure 1.2: An example of the over-the-cell routing problem

the graph is minimized. We call this minimum total sum, the *edge chromatic sum*, and denote it by $\Sigma'(G)$. An edge coloring is called an *optimum edge coloring*, if its total sum of colors is $\Sigma'(G)$. Similar to the vertex strength, the *edge strength* of a graph is the minimum number of colors needed to achieve an optimum edge coloring of the graph, and is denoted by $s'(G)$. Note that there are some graphs for which the edge strength and the chromatic index are not equal, i.e $s'(G) > \chi'(G)$. See [17] for such an example. Edge sum coloring is a more recent notion and, not surprisingly, there are not many results about it in the literature.

1.2 Applications and motivation

The application mentioned by Nicolosco et al.[31] that motivated their study of the sum coloring problem is a problem in VLSI design, known as the over-the-cell routing problem. This is exactly the same problem that lead Supowit [36] to the notion of the OCCP problem. We are given a set of two-terminal nets that should be connected electrically. We have a base line on which the nets lie and some parallel horizontal tracks equally spaced, at distances $d = 1, 2, 3, \ldots$ from the base line. The connection of two net terminals, whose positions are given, must be composed of two vertical segments and one horizontal segment where the horizontal segment must lie on one of the tracks. Note that no overlapping nets can be routed through the same track (figure 1.2). The goal is to minimize the total length of wires needed to connect these nets. The total length of horizontal wires is fixed and is equal to the total sum of distances of two terminals of the nets. Therefore we have to minimize the sum of the lengths of the vertical wires. It is easy to see that this is equivalent to the sum coloring problem restricted to interval graphs.

Another application is given by Kroon et al. [30]. The OCCP problem for interval graphs is equivalent to the Fixed Interval Scheduling Problem (FISP) with machine dependent processing costs. In this scheduling problem each job $j \in J$ must be executed during a given time interval (s_j, f_j). We assume that a sufficient number of machines are available and that each job must be executed by one of the machines. However, the processing costs are machine-dependent. That is, if job j is executed by machine i, then the associated processing cost is c_i. The objective is to find a feasible non-preemptive schedule for all jobs with minimum total processing costs. Other variants of FISP have been considered in the literature (see references No 1, 11-14, 22, and 23 of [22]).

Bar-noy et al. [2] considered the application of the sum coloring problem to the resource allocation problem with constraints imposed by conflicting resource requirements. Assume that we have a distributed resource allocation system in which the constraints are given as a conflict graph G, whose nodes represent processors, and the edges indicate competition on resources. In other words, two nodes are adjacent if the corresponding processors can not run their jobs simultaneously. The allocation of the resources must satisfy the following conditions:

- *Mutual exclusion:* No two conflicting jobs are executed simultaneously.
- *No infinite wait:* The request of any processor is eventually granted.

The goal is to minimize the average response time. This is equivalent to minimizing the sum of the job completion times. Assuming some fixed execution time for jobs, this is exactly the sum coloring problem for the given conflict graph.

For some resource allocation problems, such as the classic *dining philosophers*, efficient solutions require an edge coloring of the conflict graph. For this kind of problem, finding an optimum solution is equivalent to solving the edge sum coloring problem [2].

1.3 Notation and definitions

Our basic notation and terminology reference is [38]. All the graphs we consider are *simple undirected loopless*, unless specified otherwise. We denote a graph G with vertex set V and edge set E by $G(V, E)$. A graph is a *multigraph* if E is a family, rather than a set. The edge containing two vertices u and v is denoted by uv. By *size* of a graph, we mean the number of vertices of that graph, and is denoted by $|G|$. Two vertices u and v are *adjacent* or *neighbors* if $uv \in E$. We call the number of neighbors of vertex v, the *degree* of v, and denote it by $deg(v)$. The maximum and minimum degrees of a graph are usually denoted by Δ and δ respectively. A graph is called *regular* if all the vertices have the same degree, and is called k*-regular* if this degree is k. 3-regular graphs are sometimes called *cubic* graphs. A graph $G'(V', E')$ is a subgraph of $G(V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. G' is an *induced subgraph* of G if it is a subgraph of G and it contains all the edges uv, such that $u, v \in V'$, and $uv \in E$.

Two graphs $G(V,E)$ and $G'(V',E')$ are *isomorphic* if there exists a one-to-one and onto function $f : V \longrightarrow V'$ such that $uv \in E$ if and only if $f(u)f(v) \in E'$. The *cartesian product* of graphs G and H, written $G \times H$, is the graph with vertex set $V(G) \times V(H)$ specified by putting (u,v) adjacent to (u',v') if and only if (1) $u = u'$ and $vv' \in E(H)$, or (2) $v = v'$ and $uu' \in E(G)$.

A graph is a *complete graph* or *clique* if for any pair u, v of vertices, $uv \in E$. A graph is *empty* or an *independent set* if it has no edges. We denote the complete graph with n vertices, the path with n vertices, and the cycle with n vertices by K_n, P_n, and C_n respectively. By $\omega(G)$ we mean the size of the maximum clique of the graph G. By the complement of a graph $G(V,E)$, we mean the graph $\overline{G}(V,E')$ where $uv \in E'$ if and only if $uv \notin E$. A vertex having degree one, in a tree, is called a *leaf.* A vertex of a tree which is not a leaf is an *internal vertex* or *internal node.*

A graph $G(V,E)$ is *chordal* if it has no induced cycle of size greater than 3. A graph is a *split* graph if there is a partition of its vertices into a clique and an independent set. A set of intervals on the real line can be represented by a graph whose vertex set corresponds to the set of intervals where two vertices are adjacent if and only if the corresponding intervals overlap. Such a graph is called an *interval graph.* If all the intervals have the unit size then the associated interval graph is called *unit interval graph.* Clearly both split graphs and interval graphs are chordal. A *matching* is a set of edges such that no two edges in that set share a vertex. A matching is called a *maximum matching* if its size is the largest size amongst all matchings of the graph.

A *coloring* of a graph $G(V,E)$ is a function $f : V \longrightarrow N$, where $f(v)$ is called the color of vertex v, for every vertex $v \in V$. A coloring is called *proper* if no two adjacent vertices have the same color. Graph G is *k-colorable* if there exists a proper coloring of G using at most k colors. A 2-colorable graph is called a *bipartite graph.* Similarly, a *k-partite graph* is a graph which is k-colorable. Therefore the vertex set of any k-partite graph can be partitioned into k independent sets. A *complete bipartite* graph, is a bipartite graph in which the edge set contains all possible edges between the two parts of the graph. The *chromatic number* of graph G is the smallest number k such that G is k-colorable, and is denoted by $\chi(G)$.

Similarly, an *edge coloring* is a function $f : E \longrightarrow N$, where $f(e)$ is the color of edge e. A *proper edge coloring* is an edge coloring such that no two edges that share a vertex have the same color. A Graph G is *k-edge-colorable* if there exists a proper edge coloring of G using k colors. The *chromatic index* of a graph is the minimum number k such that G is k-edge-colorable and is denoted by $\chi'(G)$. The *line graph* of a graph $G(V,E)$ is the graph $G'(V',E')$, such that for every $e \in E$ there is a node $v \in V'$, and $uv \in E'$ if and

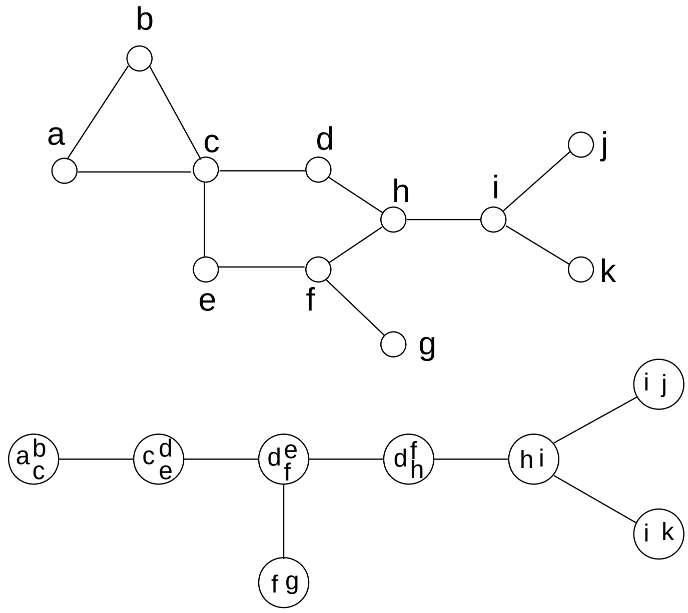

Figure 1.3: A graph and its tree decomposition with width 2

only if the corresponding edges in E share a vertex. Clearly the edge coloring of a graph is equivalent to the vertex coloring of the corresponding line graph. The *coloring number* of a graph G, denoted by $col(G)$, is defined to be the smallest number d such that there exists a linear ordering $<$ of the vertex set where the back degree of every vertex u, which is $|\{v \, : \, v < u, \; uv \in E\}|$, is strictly less than d.

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

- $\bigcup_{i \in I} X_i = V$.
- for each edge $uv \in E$, there exists an $i \in I$, such that $v \in X_i$ and $u \in X_i$.
- for all $i, j, k \in I$, if j is a vertex in T on the path between i and k then $X_i \cap X_k \subseteq X_j$.

The *width* of a tree decomposition (X, T) is defined as $\max_{i \in I} |X_i| - 1$. The *tree-width* of a graph G, is the minimum width of (X, T), over all tree decompositions of G.

A *k-tree* is defined recursively as follows:

- A clique of size k is a k-tree.
- If $T(V, E)$ is a k-tree and C is a complete subgraph of T on k vertices, then the graph $G' = (V \cup \{x\}, E \cup \{xv | v \in C\})$, where $x \notin V$ is also a k-tree.

Note that 1-trees are exactly standard trees. Any subgraph of a k-tree is called a *partial k-tree*. It is proved that partial k-trees are exactly the graphs with tree-width at most k [34].

A well known class of graphs is the class of perfect graphs. A graph G is *perfect* if and only if for every induced subgraph H of G, we have $\chi(H) = \omega(H)$. An important subclass of perfect graphs is cographs. G is a *cograph* if and only if it doesn't have an induced P_4.

The following results are immediate consequences of the definition of optimum vertex coloring.

Fact 1.1 *Let C be an optimum vertex coloring of G which uses k colors. Let's call the set of vertices having color i in C, C_i, for $1 \leq i \leq k$. Then:*

1. $|C_1| \geq |C_2| \geq \ldots \geq |C_k| \geq 1$.
2. *C_i is a maximal independent set in the subgraph $G - \bigcup_{j=1}^{i-1} C_j$.*
3. $s(G) \leq \Delta + 1$.
4. $\Sigma(G - C_1) = \Sigma(G) - |V|$.
5. $s(G - C_1) = s(G) - 1$.
6. $\Delta(G - C_1) \leq \Delta(G) - 1$.

1.4 Previous work

As we mentioned before, the chromatic sum problem is introduced by Kubika in her Ph.D dissertation. In [29], Kubika and Schwenk prove the NP-completeness of the chromatic sum problem for general graphs. On the other hand, they give a polynomial time algorithm to find the chromatic sum of trees. Also, for any integer k, they show how to construct the smallest tree T_k, whose vertex strength is at least k.

Theorem 1.2 *[29] For any integer k, there is a tree of size*

$$|T_k| = \frac{1}{\sqrt{2}}[(2+\sqrt{2})^{k-1} - (2-\sqrt{2})^{k-1}]$$

which needs at least k colors in any optimal vertex coloring of it.

As a consequence, they show:

Corollary 1.3 *[29] For any integer k, almost every tree requires at least k colors in any optimum vertex coloring of it.*

In [11] Erdos et al. continue the study of graphs that require many colors in their optimum vertex colorings and give some other constructions to make such graphs.

Erdos et. al [37] give some interesting tight bounds on the chromatic sum of a graph in terms of the number of vertices and edges of the graph. They prove that $\Sigma(G) \leq |V|+|E|$. Also, they show that:

Theorem 1.4 *[37] For any connected graph G:*

$$\lceil\sqrt{8|E|}\rceil \leq \Sigma(G) \leq \lfloor\frac{3}{2}(|E|+1)\rfloor.$$

These bounds are tight as they show there exist graphs that attain these bounds.

In [17] Hajiabolhassan et al. consider optimum vertex colorings of graphs which use the minimum number of colors, i.e optimum vertex colorings with $s(G)$ colors. They prove a theorem similar to Brooks' theorem, by showing that $s(G) \leq \Delta$ if G is neither a complete graph nor an odd cycle. Furthermore, they improve this bound and show that:

Theorem 1.5 *[17] For any graph G:*

$$s(G) \leq \lceil\frac{col(G)+\Delta(G)}{2}\rceil.$$

As far as we know, this is the best bound for the strength of a graph. In that article, they conjecture:

Conjecture 1.6 *[17] For every graph G:*

$$s(G) \leq \lceil\frac{\chi(G)+\Delta(G)}{2}\rceil.$$

The conjecture is affirmed for trees and is the best possible bound as proved by Jiang and West [26], where they show that for every integer k there exists a tree with $\Delta = 2k-2$ whose strength is k.

In [31] the sum coloring problem restricted to the family of interval graphs is studied. The sum coloring problem is NP-complete for interval graphs if the sizes of intervals are at least 4 (see also reference 16 of [31]). They give an approximation algorithm that can be used to obtain a lower bound for $\Sigma(G)$, where G is an interval graph. The idea behind this algorithm is as follows: Let P be any optimum vertex coloring of G, k be the number of colors used in P, and C_i be the set of vertices of G having color i, $1 \leq i \leq k$. Also, let A_1 be a maximum independent set in G. Clearly, it is impossible to color more than

A_1 vertices with color 1, in any optimum vertex coloring of G. Therefore $|C_1| \leq |A_1|$. We can call A_1, the largest 1-colorable subgraph of G, as well. In general, let A_i be a subgraph of G which is i-colorable, and has the maximum number of vertices among i-colorable subgraphs of G, for $1 \leq i \leq \chi(G)$. So we have:

$$\begin{aligned}
|C_1| &\leq |A_1| \\
|C_1| + |C_2| &\leq |A_2| \\
&\vdots \\
|C_1| + |C_2| + \ldots + |C_{\chi(G)}| &\leq |V| \\
&\vdots \\
|C_1| + |C_2| + \ldots + |C_k| &\leq |V|
\end{aligned}$$

Therefore, the best case (the case with the minimum sum of colors) is when we can have $|A_i|$ nodes in the first i sets, $1 \leq i \leq \chi(G)$. In other words, when $|C_i| = |A_i| - |A_{i-1}|$. This represents the following lower bound on the chromatic sum of a graph:

$$\Sigma(G) \geq |A_1| + \sum_{i=1}^{\chi(G)} i(|A_i| - |A_{i-1}|).$$

Their algorithm computes the values of $|A_i|$, for an interval graph G, using a greedy method and a property of interval graphs, called the consecutive 1's property. They also show that this algorithm gives the exact value of $\Sigma(G)$ if G is a proper interval graph, or more generally, if the size of each interval is at most 3.

On the other hand, there are some similar results on the OCCP problem. Kroon et al. [30] study this problem for interval graphs and trees. They give a linear time algorithm for this problem restricted to trees. Also, they show that this problem is NP-complete for interval graphs, even if there are four different values for color costs, and it is solvable in polynomial time if there are at most two different values for color costs. Finally, they give an integer linear program (ILP) formulation of this problem, and prove that the zero-one matrix corresponding to the constraints of the ILP is perfect, if and only if $G \times K_{|V|}$ does not contain an induced odd cycle of size 7 or more.

Jansen [22] has studied the OCCP problem for several classes of graphs. He proves that the OCCP problem for cographs and for graphs with tree-width at most k can be solved in time $O(|V| + |E|)$ and $O(|V| \log^{k+1} |V|)$, respectively. The algorithm he developed for cographs consecutively finds a maximum independent set in the remainder of the graph, assigns the cheapest color among the set of unused colors, and removes the independent set from the graph. This algorithm uses the cotree representation of the cographs, and produces $\chi(G)$ sets. By Corneil et al. [8] we can find a cotree of a cograph in linear time.

For the case of graphs with bounded tree-width, he uses a dynamic programming method based on a tree decomposition of the graph. Finding the tree decomposition of a graph with bounded tree-width can be solved in linear time, as proved by Bodlaender [7]. Also, he uses the following lemma:

Lemma 1.7 *[22] For a graph $G(V,E)$ with constant tree-width, there exists an optimum vertex coloring f such that at most $O(\log|V|)$ colors are used.*

Let $T(I,F)$ be a tree decomposition of G, and for each node $i \in I$, let Y_i be the set of all vertices in a set X_j, with $j = i$ or j is a descendant of i in the rooted tree T. The algorithm computes a table $minc_i$, for each node $i \in I$. If m is the number of allowed colors, then for each coloring $f : X_i \longrightarrow \{1,2,\ldots,m\}$, there is an entry in the table $minc_i$, fulfilling:

$$minc_i(f) = \min_{\overline{f}:Y_i \longrightarrow \{1,2,\ldots,m\}, \overline{f}(x)=f(x) \forall x \in X_i} \sum_{j=1}^{m} c_j |\{y | y \in Y_i, \overline{f}(y) = j\}|.$$

In other words, for each coloring f of X_i, $minc_i(f)$ denotes the minimum sum over all colorings $\overline{f}$ of Y_i, where $\overline{f}$ and f have the same color for each vertex $x \in X_i$. Using the above lemma, he shows how to compute the entries of the table for each node in time $O(m^{k+1}k^2)$, and therefore the algorithm runs in time $O((\log|V|)^{k+1}|V|)$.

In the same article, he considers the ILP formulation of the problem, given by Sen et al. [35] and shows that the corresponding polyhedron contains only integral 0/1 extrema if and only if the graph G is a diamond $(K_4 - e)$ free chordal graph. On one hand he shows the NP-completeness of this problem on bipartite graphs and also permutation graphs, and on the other hand he proves that the OCCP problem can be solved for the complements of bipartite graphs in polynomial time using bipartite matching.

Approximation algorithms for the sum coloring problem are studied in [28], [23],[2], [3]. In [28] it is shown that approximating the chromatic sum problem within an additive constant factor is NP-hard. Then, Bar-Noy et al. in [2] prove that the sum coloring problem can not be approximated within a factor $n^{1-\epsilon}$, for any $\epsilon > 0$, unless $NP = ZPP$. Also, they prove that finding consecutive maximum independent sets and assigning the first available color to each gives a 4-approximation to the sum coloring problem. As a consequence of this theorem, we have a 4γ-approximation algorithm for the sum coloring problem for a family of graphs, whenever we have a γ-approximation algorithm for the maximum independent set problem for that family. Also, they show that their bound is tight within a factor 2. In other words, they show that there exists family of graphs for which the above algorithm is at least a 2-approximation algorithm. For bipartite graphs, they prove that the following simple algorithm is a $\frac{9}{8}$-approximation for the chromatic sum: Consider the following two colorings and take the minimum of them. One is a 2-coloring of it. The other coloring colors a maximum independent set with color 1, and then 2-colors the remaining vertices. The maximum independent set can be found in bipartite graphs using a matching algorithm.

In [3] Bar-Noy et al. prove the hardness of approximating of the sum coloring problem for bipartite graphs. They prove that there exists an $\epsilon > 0$ such that there is no $(1+\epsilon)$-approximation algorithm for the sum coloring problem for bipartite graphs, unless P=NP. Also, they improve the previous ratio for bipartite graphs and give a $\frac{10}{9}$-approximation algorithm for bipartite graphs.

Jansen [23] has many approximation results for the OCCP problem. He proves that there exists no approximation algorithm with ratio $O(|V|^{0.5-\epsilon})$ for the OCCP problem restricted to bipartite graphs and interval graphs, unless $P = NP$. On the other hand, he gives algorithms for bipartite graphs and interval graphs that approximate the OCCP problem with ratio $O(|V|^{0.5})$. Therefore, these are the best possible approximation algorithms for the OCCP problem for these classes of graphs. Finally he proves that there is no algorithm to approximate the OCCP problem with ratio $O(|V|^{1-\epsilon})$ for permutation graphs, split graphs, and therefore chordal graphs, unless $P = NP$.

The only known results about the edge sum coloring problem appear in [17] and [2]. The edge sum coloring problem is introduced independently in both of these articles as the vertex sum coloring problem restricted to line graphs. Hajiabolhassan et al. in [17] introduce the notion of edge strength, and similar to Vizing's theorem for chromatic index, they prove:

Theorem 1.8 *[17] For every graph G, $s'(G) \leq \Delta + 1$.*

Bar-noy et al. [2] prove that the edge sum coloring problem is NP-hard for multigraphs. They consider a special kind of coloring, called *compact coloring* in which every edge e with color i has neighboring edges with all colors $1, 2, \ldots, i-1$. They prove that any compact coloring is a 2-approximation of the edge sum coloring problem.

1.5 Overview

The main results of this thesis appear in chapters 2, 3 and 4. In chapter 2 we give some general results on the sum coloring problem. This includes the proof of NP-completeness of finding the vertex strength of graphs, which appears in section 2.1. We use the same technique used by Kubika and Schwenk [29] to prove the NP-completeness of chromatic sum. In the next section, we give some upper bounds on the strength of graphs with small chromatic number in terms of the size of the graph. By theorem 1.2, the strength of a bipartite graph, in particular a tree, can be arbitrary large. We show that it can not be larger than the logarithm of the size of the graph. Finally, in the last section, we extend the Kubika and Schwenk [29] result about the NP-completeness of the sum coloring problem and prove that this problem is NP-complete for the class of split graphs.

Chapter 3 provides some algorithms for the sum coloring problem for some classes of graphs. As an affirmative result, with respect to the NP-completeness result of the last section of chapter 2, we prove that the sum coloring problem can be solved in polynomial time if we bound by a constant the degrees of vertices of either part of a split graph. In the second section, we extend the result of Jansen for OCCP of cographs by giving an algorithm to solve this problem for a more general class of graphs, which contain cographs, called P_4-reducible graphs. Finally, in the last section we show that the sum coloring problem can be solved efficiently for chain bipartite graphs and cobipartite graphs.

Chapter 4 deals with the edge sum coloring problem. In the first section, we prove the NP-completeness of this problem and also the edge strength problem for cubic graphs. In section 4.2, we give a polynomial time algorithm to find the edge chromatic sum and an optimum edge sum coloring of a given tree. This algorithm can be extended to find an optimum edge sum coloring of weighted trees. In the next section, using Monadic Second Order Logic, we show that there exists a linear time algorithm to solve the edge sum coloring problem for partial k-trees with bounded degree, for fixed k.

Finally, in the last chapter, we give our concluding remarks and some problems that are still left open. These problems can be a starting point for future work in this area.

Chapter 2

General results on sum coloring

In sum coloring of graphs, there are two parameters that we are interested in. One is the total sum of the costs of colors (the chromatic sum $\Sigma(G)$ of the graph), which we try to minimize. The other one is the minimum number of colors used in an optimum vertex coloring, which is the vertex strength, $s(G)$. We can study these parameters from two points of view: one is from the mathematical aspect, in which we can give the exact value or at least some bounds for these parameters. The other is from the algorithmic aspect, in which we analyze the complexity of computing these parameters.

In this chapter, we look at these parameters from both of these points of view. The first section proves that there exists no polynomial time algorithm that finds $s(G)$ for graphs with $\Delta \leq 6$, unless $P = NP$. In the second section, we show that although the vertex strength of graphs with small chromatic number might be much larger than their chromatic number, there are some logarithmic bounds in terms of the size of the graph for this parameter. Finally, we improve the NP-completeness result of Kubika and Schwenk [29] for computing the chromatic sum, by proving that it is NP-complete even for the restricted class of split graphs.

2.1 Complexity of finding the vertex strength

Since the sum coloring problem seems no easier than the chromatic number problem, one can expect that finding the vertex strength is NP-Hard. We prove that in fact it is NP-Hard to find the vertex strength even for a graph with $\Delta = 6$.

Our proof is very similar to the proof of Kubika and Schwenk [29] for proving the NP-completeness of finding the chromatic sum. We give a reduction from the vertex 3-coloring problem restricted to the graphs with maximum degree 4.

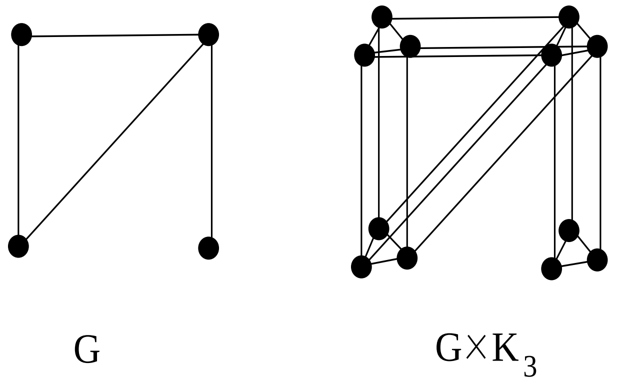

Figure 2.1: A graph G and the cartesian product $G \times K_3$

The instance and the question of vertex 3-coloring problem is stated as:

Instance: A graph G with maximum degree 4.

Question: Is G 3-colorable?

It is well known that the above restricted version of vertex coloring problem is NP-complete even for the class of planar graphs [15]. We state the vertex strength problem restricted to the class of graphs with maximum degree 6 as:

Instance: A graph G with maximum degree 6.

Question: Is $s(G) \leq 3$?

Theorem 2.1 *The vertex strength problem restricted to the class of graphs with maximum degree 6 is NP-Hard.*

Proof: We are going to reduce the vertex 3-coloring of graphs with maximum degree 4 problem to the vertex strength of graphs with maximum degree 6 problem. Consider an instance of the vertex coloring problem. Let $G(V, E)$ be the graph of this instance. We want to know if $\chi(G) \leq 3$.

Construct the graph $G'(V', E')$ as follows: Let G_1, G_2, G_3 be 3 copies of the graph G, and let v_i be the vertex of graph G_i which corresponds to vertex v of the graph G. Let $V' = \bigcup_{i=1}^{3} V_i$, where V_i is the vertex set of G_i. Put all the edges of each G_i in E' $(1 \leq i \leq 3)$. Also, for each vertex $v \in G$, add the edges $\{v_1v_2, v_1v_3, v_2v_3\}$ to E'. In other words, G' is the cartesian product $G \times K_3$. See figure 2.1.

Since the degree of each vertex in G is at most 4, and each $v_i \in G'$ is connected to two more vertices, therefore the maximum degree of G' is at most 6. It's trivial that we can construct G' in polynomial time. Now we claim that:

$$\chi(G) \leq 3 \Longleftrightarrow s(G') = 3.$$

First suppose that $s(G') = 3$. It means that there is an optimum vertex coloring of G' in which just 3 colors are used. Since G is a subgraph of G', this coloring induces also a proper 3-coloring for G. Therefore $\chi(G) \leq 3$.

Now assume that $\chi(G) \leq 3$. Therefore we can color G_1 with 3 colors, independently. To obtain a proper 3-coloring of G', we use the same partition of vertices of G_1 for G_2 and G_3, with the modification that the color of the j'th class of G_i, is $(i + j - 2 \; mod \; 3) + 1$, instead of j. Thus each G_i $(1 \leq i \leq 3)$ is colored with colors $1, 2, 3$, and also this is a proper coloring of G'. It's not difficult to see that this is an optimum vertex coloring for G'. This follows since each complete subgraph of size 3 of G', which contains the corresponding vertices of copies of G, requires at least 3 colors, in any proper coloring of G'. In this coloring each of these complete subgraphs are colored with colors $1, 2, 3$, which clearly gives the least possible sum of colors. Thus this is an optimum vertex coloring of G', and clearly uses the least possible number of colors. Therefore $s(G') = 3$. ■

Using this method and by reduction from k-color problem, for $k > 3$, we can prove that:

Corollary 2.2 *For a given graph G, it is NP-Hard to determine if $s(G) \leq k$, for any fixed $k \geq 3$.*

We don't know the time complexity of deciding if the strength of a given graph is equal to 2, but we expect it to be NP-Hard.

2.2 Some bounds on the strength of graphs with small chromatic number

The notion of vertex strength of graphs has been studied by Erdos [11], Kubika and Schwenk [29], and Hajiabolhassan et al. [17]. It's not difficult to see that $s(G) \leq \Delta + 1$. Hajiabolhassan et al. [17] give an analogous theorem to Brooks' theorem, for the vertex strength of a graph:

Theorem 2.3 *[17] Let G be a connected graph. Then $s(G) = \Delta + 1$ if and only if G is the complete graph or an odd cycle.*

This theorem implies Brooks' theorem. Also, they improved the above bound to $\lceil \frac{col(G)+\Delta(G)}{2} \rceil$, where $col(G)$ is the coloring number of G. (theorem 1.5).

It is well known that aside from regular graphs, $\chi(G) \leq col(G) \leq \Delta$. They conjectured that we can replace the coloring number in the above theorem by the chromatic number. This conjecture is true for trees and follows immediately from the above theorem and the fact that the coloring number of every tree is 2. This bound is sharp for trees as proved by Jiang and West [26]. However, the conjecture is still open even for the class of bipartite graphs.

Here we prove that for the class of graphs with chromatic number at most 4, the vertex strength is in $O(\log n)$. In particular we prove that, for bipartite graphs $s(G) < \log_2 n$, for tripartite graphs $s(G) \leq 2\log_2(\alpha n + 1)$ where $\alpha = \sqrt{2} - 1$, and for graphs with chromatic number at most 4, $s(G) \leq 4\log_2(\beta n + 2)$, where $\beta = 2^{\frac{1}{4}} - 1$. These bounds are interesting since in many cases, such as complete bipartite graphs and regular graphs with high degree, they are better than the only known bounds. Remember that the chromatic sum problem restricted to bipartite graphs is NP-complete [3]. Moreover, we prove that our bounds are sharp. In particular, we prove that for any k, there exists a tree whose strength is k and has at most α^k vertices, where $\alpha \leq 2 + \sqrt{2}$. This means that the order of the strength of trees captures the upper bound for the strength of bipartite graphs.

Theorem 2.4 *If G is a bipartite graph, then:*

$$s(G) < \log_2 n.$$

Proof: Assume that G is a bipartite graph and $s(G) = s$. It's trivial that $s \geq 2$. If $s = 2$ then there is nothing to prove. So assume that $s > 2$. Let P be any optimum vertex coloring of G using s colors. Call the set of vertices having color i in this coloring C_i, $1 \leq i \leq s$. It is clear that $|C_i| \geq |C_{i+1}| \geq 1$, $1 \leq i < s$. We use the following two lemmas in our proof:

Lemma 2.5 *For any three consecutive class of colors C_i, C_{i+1}, C_{i+2} of coloring P, where $1 \leq i \leq s-2$, we have:*

$$|C_i| \geq |C_{i+1}| + 3|C_{i+2}| + 2.$$

Proof: Let G' be the induced subgraph of G on the vertex set $C_i \cup C_{i+1} \cup C_{i+2}$. Note that G' is also bipartite. Thus we can recolor it with two colors i and $i+1$. The sum of colors of the vertices of G' in this coloring is at most

$$i\lceil \frac{|C_i| + |C_{i+1}| + |C_{i+2}|}{2} \rceil + (i+1)\lfloor \frac{|C_i| + |C_{i+1}| + |C_{i+2}|}{2} \rfloor.$$

Note that by this recoloring we obtain another proper coloring of G using $s-1$ colors. Because by this recoloring we can reduce the number of colors used in the coloring of G, and since s is the minimum number of colors in an optimum coloring of G, we must have:

$$i|C_i| + (i+1)|C_{i+1}| + (i+2)|C_{i+2}| < i(|C_i| + |C_{i+1}| + |C_{i+2}|) + \lfloor \frac{|C_i| + |C_{i+1}| + |C_{i+2}|}{2} \rfloor$$
$$\Longrightarrow |C_{i+1}| + 2|C_{i+2}| < \lfloor \frac{|C_i| + |C_{i+1}| + |C_{i+2}|}{2} \rfloor$$
$$\Longrightarrow |C_{i+1}| + 3|C_{i+2}| + 2 \leq |C_i|.$$

■

Lemma 2.6 *For every color class i of the coloring P, such that $i < s-1$, we have:*

$$|C_i| > 2^{s-i}.$$

Proof: We have at least one vertex of each color. Therefore $|C_s| \geq 1$ and $|C_{s-1}| \geq 1$. It follows from the previous lemma that $|C_{s-2}| \geq 6$ and $|C_{s-3}| \geq 11$. Now we use simple backward induction. The base is for $s-2$ and $s-3$, and the lemma is correct. Assume that $i \leq s-4$, and $C_{i+1} \geq 2^{s-i-1}$ and $C_{i+2} \geq 2^{s-i-2}$. So:

$$|C_i| \geq |C_{i+1}| + 3|C_{i+2}| + 2 \geq 2^{s-i-1} + 3 \times 2^{s-i-2} + 2 > 2^{s-i}.$$

■

The total number of vertices in G is equal to the sum of the number of vertices of C_i, for $1 \leq i \leq s$. Therefore:

$$n = \sum_{i=1}^{s} |C_i| > \sum_{i=1}^{s-3} 2^{s-i} + 6 + 1 + 1 \geq 1 + \sum_{i=1}^{s} 2^{s-i} = 2^s$$
$$\Longrightarrow 2^s < n \Longrightarrow s < \log_2 n.$$

■

Note that we can do better in lemma 2.5 and prove that:

$$|C_i| \geq |C_{i+1}| + 3|C_{i+2}| + 1 + 2 \sum_{j=i+3}^{s} |C_j|.$$

Also we can find better lower bounds for the size of C_i by solving the recursive relation of lemma 2.5, but this doesn't change the final bound for s more than a small constant factor, and therefore, the strength will be in $O(\log n)$.

For the case of graphs with chromatic number 3 or 4, we can find similar bounds. Let G be a graph whose strength is greater than its chromatic number. Using the technique of lemma 2.5, we can easily prove the following inequalities for the size of color class C_i:

Lemma 2.7 *For the graph G, where $s(G) > \chi(G)$:*

- *If $\chi(G) = 3$ then : $|C_i| \geq |C_{i+2}| + 2|C_{i+3}| + 1$.*
- *if $\chi(G) = 4$ then : $|C_i| \geq -\frac{1}{3}|C_{i+1}| + \frac{1}{3}|C_{i+2}| + |C_{i+3}| + \frac{5}{3}|C_{i+4}| + 1$.*

We know that $|C_{s-j}| \geq 1$, $0 \leq j \leq 3$. By solving the recursive relation for $|C_i|$ in lemma 2.7, we obtain the following bounds:

Lemma 2.8 *For the graph G, where $s(G) > \chi(G)$:*

- *If $\chi(G) = 3$ then: $|C_i| \geq 2^{\frac{s-i}{2}}$, for $i < s-3$.*
- *If $\chi(G) = 4$ then: $|C_i| \geq 2^{\frac{s-i}{4}}$, for $i < s-4$.*

Therefore, for the case $\chi(G) = 3$ we have:

$$n = \sum_{i=1}^{s} |C_i| \geq 7 + \sum_{i=1}^{s-4} 2^{\frac{s-i}{2}} \geq (3\sqrt{2} - 4) + \sum_{i=1}^{s} 2^{\frac{s-i}{2}} \geq (3\sqrt{2} - 4) + \frac{2^{\frac{s}{2}} - 1}{\sqrt{2} - 1}.$$

Let $\alpha = \sqrt{2} - 1$. Since $\alpha(3\sqrt{2} - 4) < 1$, therefore:

$$\alpha n + 1 > 2^{\frac{s}{2}} \Longrightarrow \log_2(\alpha n + 1) > \frac{s}{2}.$$

Hence:

Theorem 2.9 *For a graph G, where $\chi(G) = 3$, we have:*

$$s(G) \leq 2\log_2(\alpha n + 1)$$

where $\alpha = \sqrt{2} - 1$.

Similarly, if $\chi(G) = 4$ then by lemma 2.8:

$$n = \sum_{i=1}^{s} |C_i| \geq 7 + \sum_{i=1}^{s-5} 2^{\frac{s-i}{4}} \geq \sum_{i=1}^{s} 2^{\frac{s-i}{4}} - 1 = \frac{2^{\frac{s}{4}} - 1}{2^{\frac{1}{4}} - 1} - 1.$$

Therefore:

Theorem 2.10 *For the graph G, where $\chi(G) = 4$, we have:*

$$s(G) \leq 4\log_2(\beta n + 2)$$

where $\beta = 2^{\frac{1}{4}} - 1$.

Unfortunately, we couldn't use this method to obtain a general bound in terms of n for the strength of k-partite graphs, for fixed k, but we guess that there exists a similar bound for the case of k-partite graphs.

Conjecture 2.11 *If G is a graph where $\chi(G) \leq k$, for some fixed k, then $s(G) \in O(\log n)$.*

Now we prove that the bounds we have given are tight. To do so we show that for any fixed k, there exists a tree whose strength is k, such that $k \in \Theta(\log n)$, where n is the number of vertices. We use the same family of trees that Kubika and Schwenk introduced in [29]. They showed how to construct a tree T_k which is the smallest tree whose strength is at least k, $k \geq 3$. They proved that the number of vertices of this tree is:

$$|T_k| = \frac{1}{\sqrt{2}}[(2+\sqrt{2})^{k-1} - (2-\sqrt{2})^{k-1}].$$

Let $\alpha = 2 + \sqrt{2}$. Therefore:

$$n = |T_k| \leq \frac{\alpha^{k-1}}{\sqrt{2}} \Longrightarrow \sqrt{2}n \leq \alpha^{k-1} \Longrightarrow \frac{\log n + \frac{1}{2}}{\log \alpha} \leq k - 1.$$

Since $\frac{1}{\log \alpha} \geq 0.56$, we can say:

$$k \geq 0.56 \log n + 1.25 \Longrightarrow k \in \Theta(\log n).$$

Hence, we've proved that:

Theorem 2.12 *For any given k, there exist a tree whose strength is k and whose number of vertices is in $O(2^k)$.*

2.3 Complexity of vertex sum coloring

Finding the chromatic sum or finding an optimum vertex coloring of a graph, seems to be no easier than the ordinary vertex coloring. Kubika and Schwenk [29] proved that the vertex sum coloring problem is NP-complete for arbitrary graphs. On the other hand, there are some classes of graphs where the chromatic number can be found in polynomial time, and sometimes very easily, whereas finding their chromatic sum is NP-complete. Interval graphs are such a family [31], [16].

In this section we prove that the vertex sum coloring problem is NP-complete for the class of split graphs, a subclass of chordal graphs. Therefore, our result proves the NP-completeness of the problem for the class of chordal graphs, as well. Also, since the OCCP formulation is a generalization of the one we are using, our NP-completeness result implies Jansen's result, for the class of split graphs. Note that the ordinary vertex coloring problem can be solved in polynomial time for the class of chordal graphs. This is another example of a class of graphs, where vertex coloring is in P, whereas vertex sum coloring is NP-complete.

Recall the definition of split graphs from section 1.3. Split graphs are both chordal and cochordal, as follows from the following theorem:

Theorem 2.13 *[13] A graph G is split if and only if G and $\overline{G}$ are chordal.*

We prove the NP-completeness of the sum coloring problem for split graphs by reduction from the *exact cover by 3-sets* problem. The instance and the question of this problem can be stated as follows:

Instance: Set X with $3q$ elements, and a collection C of 3-element subsets of X.

Question: Does C contain an exact cover for X? In other words, is there a subset $C' \subseteq C$, such that every element of X occurs in exactly one member of C' ?

This problem is known to be NP-complete [15]. To prove the main result of this section, first we state the instance and the question of the chromatic sum problem:

Instance: A graph G, and a positive integer k.

Question: Is it true that $\Sigma(G) \leq k$?

Note that if there exists a polynomial time algorithm for the above problem, then there exists a polynomial time algorithm, which can find the exact value of $\Sigma(G)$.

First we prove a general result for split graphs.

Lemma 2.14 *If $G(C \cup I, E)$ is a split graph, where C is the complete part and I is the independent part, and $|C| = n_C$ and $|I| = n_I$, then $s(G) \leq n_C + 1$.*

Proof: Consider an optimum vertex coloring of G. Since C is a clique, all vertices in C must have different colors. Let i be the smallest positive number such that none of the vertices of C have color i. In this case, no vertex in I can have a color greater than i, otherwise we can simply change the color of that vertex to i. So the only colors (possibly) used in I are $1, 2, \ldots, i$. It follows that there can't be any color greater than $n_C + 1$ in C, otherwise there is some color $i + x$ $(1 \leq x \leq n_C - i + 1)$ that is not used in C, and we can simply change the color greater than $n_C + 1$ to $i + x$. ∎

Corollary 2.15 *There is an optimum vertex coloring of G such that the colors used in that coloring are from the set $\{1, 2, \ldots, n_C + 1\}$.*

We state the main result of this section in the following theorem.

Theorem 2.16 *The chromatic sum problem is NP-complete for the class of split graphs.*

Proof: It is not hard to see that the chromatic sum problem is in NP, since we can easily verify in polynomial time whether a given coloring is proper or not, and if its total sum of colors is less than k.

To prove the completeness, suppose that we are given an instance of the *Exact Cover by 3-sets* problem. We have the set $X = \{x_1, x_2, \ldots, x_{3q}\}$, and collection $C = \{c_1, c_2, \ldots, c_m\}$, and we want to know if there exists a collection C', such that each element $x_i \in X$, occurs exactly once in C'. Construct the split graph $G(V, E)$ as follows: for each x_i $(1 \leq i \leq 3q)$ create vertex v_{x_i}, and for each c_j $(1 \leq i \leq m)$ create vertex v_{c_j} in G. Therefore $V = \{v_{x_1}, v_{x_2}, \ldots, v_{x_{3q}}, v_{c_1}, v_{c_2}, \ldots, v_{c_m}\}$. For any two non-equal $i, j \in \{1, 2, \ldots, m\}$, put an edge between v_{c_i} and v_{c_j}. Also, put an edge between v_{x_i} and v_{c_j}, if and only if $x_i \notin c_j$. Denote the set of vertices $\{v_{x_1}, v_{x_2}, \ldots, v_{x_{3q}}\}$ by V_X, and the set of vertices $\{v_{c_1}, v_{c_2}, \ldots, v_{c_m}\}$, by V_C.

We assume that $m > \frac{q(3q+5)}{2}$, otherwise we can add some dummy vertices to the part V_C, and connect them to all the vertices in V_C, and in V_X. We claim that:

$$\Sigma(G) = \frac{1}{2}[m(m+1) + 3q(q+1)] \iff \text{there exists such a } C'.$$

It follows from the construction of G, that it is a split graph, since V_C is a clique, and V_X is an independent set.

Assume that there exists such a C'. Therefore, there are q vertices in V_C, $v_1, v_2, \ldots, v_q$, such that v_i is not adjacent to v_{i1}, v_{i2}, v_{i3} in V_X, and $v_{ij} \neq v_{kl}$ if $i \neq k$ and $1 \leq j, l \leq 3$. So we can color G as follows: assign color i to the vertex v_i $(1 \leq i \leq q)$, and to the vertices in V_X that are not adjacent to it, which are v_{i1}, v_{i2}, v_{i3}. Assign colors $q+1, \ldots, m$ to the rest of the vertices in V_C. Call this coloring A. Let $S = \frac{1}{2}[m(m+1) + 3q(q+1)]$. First of all, it is clear that A is a proper coloring and the sum of this coloring is equal to S. We show that there can't be any coloring with a total sum less than S.

Let P be an optimum vertex coloring of G, such that color i $(1 \leq i \leq m+1)$ is not used by any vertex in V_C. By corollary 2.15, the set of colors used by V_C is $\{1, 2, \ldots, i-1, i+1, \ldots, m+1\}$.

Lemma 2.17 *If $i \leq q$, then the total cost of the coloring P, is at least:*

$$\frac{m(m+1)}{2} + m - i + 1 + \frac{3i(i-1)}{2} + 3i(q - i + 1).$$

Proof: The cost of colors used by vertices in V_C is clearly $\frac{m(m+1)}{2}+m-i+1$. Consider the arrangement of vertices of V_C, in the order of their color value $v_1, v_2, \ldots, v_m$. The only vertices in V_X that can have color 1, are those three vertices that are not connected to v_1. Similarly, for any color $j < i$, there are only three vertices in V_X that can have color j. Thus for any color j, $(1 \leq j \leq i-1)$, there are at most three vertices in V_X that can have that color, and the rest of the vertices in V_X must be colored with color i. The total sum of colors is obtained by a simple calculation. ■

Let $L = m - i + 1 + \frac{3i(i-1)}{2} + 3i(q - i + 1)$. Consider the coloring P, and assume that $i \leq q$. Since $m > \frac{q(3q+5)}{2}$, therefore $L > \frac{3q(q+1)}{2}$. Thus, the sum of the coloring P is more than S. Now, let's assume that $i > q$. So all colors $1, 2, \ldots, q$ are used by vertices in V_C. In the best case, for the coloring of V_X, there are three vertices with color j, for each $1 \leq j \leq q$, and for the coloring of V_C, just colors $1, 2, \ldots, m$ are used. The sum of this coloring is exactly S and can be obtained only if the vertices of V_X can be colored in this way, equivalently, there is a collection $V_C' \subset V_C$, such that $|V_C'| = q$, and V_C' is a vertex cover for V_X. This proves that the sum of any coloring other than A is more than S, and we can obtain A only if there exists such a covering.

■

Chapter 3

Algorithms for vertex sum coloring

The last chapter provided some theoretical results on the strength and chromatic sum of a graph. In this chapter we give some algorithms to solve the sum coloring problem, for some special classes of graphs. In contrast with the result of the last section of the previous chapter, in the first section we give an algorithm to solve the sum coloring problem of split graphs with some degree bounds. Also, we extend the result of Jansen on cographs, and give a polynomial time algorithm for a more general class of graphs, called P_4-reducible graphs, which contains the class of cographs. Recall that the sum coloring problem is NP-complete for the important class of bipartite graphs. Therefore, it is interesting to consider restrictions on bipartite graphs for which we can obtain a polynomial time algorithm for the sum coloring problem. In the last section, we consider chain bipartite graphs and cobipartite graphs and give polynomial algorithms for each class to find an optimum coloring.

3.1 Vertex sum coloring of k-split graphs

In the last section of the previous chapter, we showed that for any split graph $G(C \cup I, E)$, we have $s(G) \leq n_C + 1$, where n_C is the size of the clique part of G. Since C is a clique, it is trivial that $s(G) \geq n_C$. The following lemma which holds for any split graph will be used later in the design of algorithms for sum coloring of some subclasses of split graphs.

Lemma 3.1 *For split graph $G(C \cup I, E)$, where $|C| = n_C$ and $|I| = n_I$, we have:*
(i) $\Sigma(G) \leq \frac{n_C(n_C+1)}{2} + n_C + n_I$.
(ii) If $s(G) = n_C + 1$ *then* $\Sigma(G) = \frac{n_C(n_C+1)}{2} + n_C + n_I$.

Proof: **(i)** Consider the vertex sum coloring of G in which all of the vertices of I have color 1, and the vertices of C are colored by colors $2, 3, \ldots, n_C + 1$. It is trivial that this is a proper coloring and the total sum of colors is

$$\frac{n_C(n_C+1)}{2} + n_C + n_I.$$

(ii) If $s(G) = n_C + 1$ then there are $n_C + 1$ vertices such that the set of colors of these vertices is exactly $\{1, 2, \ldots, n_C + 1\}$, and the total sum of the colors of the other vertices is at least $n_I - 1$. Therefore:

$$\Sigma(G) \geq \frac{(n_C+1)(n_C+2)}{2} + n_I - 1 = \frac{n_C(n_C+1)}{2} + n_C + n_I.$$

By part (i) $\Sigma(G) \leq \frac{n_C(n_C+1)}{2} + n_C + n_I$. Hence: $\Sigma(G) = \frac{n_C(n_C+1)}{2} + n_C + n_I$ as required. ∎

By theorem 2.16 we see that the vertex sum coloring problem is NP-complete for the class of split graphs, and as a consequence, for the class of chordal graphs. The natural question that comes to mind is: for which classes of chordal graphs can we solve the vertex sum coloring problem efficiently ? As we mentioned in section 1.4 the vertex sum coloring problem is in P for proper interval graphs. Here we study some restrictions of split graphs where the degree of each vertex in the independent set or in the clique part is bounded.

Definition 3.2 *A split graph $G(C \cup I, E)$, where C is a complete subgraph and I an independent set, is a k-split graph if the degree of each vertex is bounded by k. It is called a k_I-split graph if the degree of each vertex of I is at most k. It is a k_C-split graph if the degree of each vertex of C is at most k.*

From the definition, it follows that a graph is k-split if and only if it is k_I-split and also k_C-split. Here, we give polynomial time algorithms to find the chromatic sum and also an optimum vertex sum coloring of k_I-split and k_C-split graphs, for fixed k.

3.1.1 Algorithm for k_I-split graphs

Let $G(C \cup I, E)$ be a k_I-split graph where $|C| = n_C$, $|I| = n_I$. By lemma 2.14 and the fact that C is a clique of size n_C, we have: $n_C \leq s(G) \leq n_C + 1$. Lemma 3.1 gives the exact value of $\Sigma(G)$ and its proof gives an optimum vertex coloring for the case $s(G) = n_C + 1$.

Now suppose that $s(G) = n_C$. We denote the set of neighbors of a vertex $c \in C$ that are in set I by $N_I(c)$. Let P be an optimum coloring of G with n_C colors. Clearly C is colored with colors $1, 2, \ldots, n_C$. Let v_{c_i} be the vertex of C that has color i in P. The set of vertices in I that have color 1 are exactly those vertices that are not connected to v_{c_1}, i.e the color of vertices in $N_I(v_{c_1})$ is at least 2. Among them, those that are not connected to v_{c_2} have color 2, and the remainder have color at least 3. In general, the number of vertices having color greater than i in I, is:

$$|N_I(v_{c_1}) \cap N_I(v_{c_2}) \ldots \cap N_I(v_{c_i})|.$$

Note that since the degree of each vertex in I is bounded by k, no vertex in I has a color greater than $k+1$ in any optimum vertex coloring of G. It can be verified that the total sum of colors in I is:

$$\sum_{i=1}^{k+1} |\{v \in I | c(v) \geq i\}|.$$

There are n_I vertices in I that have color greater than or equal to 1. Let:

$$S = |N_I(v_{c_1})| + |N_I(v_{c_1}) \cap N_I(v_{c_2})| + \ldots + |\bigcap_{i=1}^{k} N_I(v_{c_i})|.$$

Therefore, the total sum of colors in I is $n_I + S$ and the total sum of colors of coloring P is:

$$\frac{n_C(n_C+1)}{2} + n_I + S.$$

Therefore, to find an optimum vertex coloring of G, using n_C colors, we must minimize the term S. In other words, the vertices $v_{c_1}, v_{c_2}, \ldots, v_{c_k}$ should be selected in such a way that the sum S is minimized amongst all possible assignment of colors $1, 2, \ldots, n_C$ to the vertices of C. We can simply consider all permutations π with k elements, such that each element is a vertex of C, and compute the value of S for each permutation by assigning color i to the vertex $v_{c_{\pi(i)}}$ of C, and then taking the minimum over all values of S. The number of such permutations is $O({n_C}^k)$. The chromatic sum of G is the minimum between this value and the sum of the optimum vertex coloring of G using $n_C + 1$ colors. Therefore, we have proved the following theorem:

Theorem 3.3 *Let $G(C \cup I, E)$ be a k_I-split graph, for fixed k. Then the chromatic sum and also an optimum vertex coloring of G can be computed in $O({n_C}^k)$, where $n_C = |C|$.*

3.1.2 Algorithm for k_C-split graphs

Let $G(C \cup I, E)$ be a k_C-split graph, where $|C| = n_C$ and $|I| = n_I$. Again, if $s(G) = n_C + 1$ then we know the exact value of $\Sigma(G)$ by lemma 3.1. Assume that $s(G) = n_C$ and consider an optimum vertex coloring of G, called P. Let v_{c_1} be the vertex in C that has color 1 in P. Clearly every vertex in $I - N_I(v_{c_1})$ also has color 1. We show that no vertex in $N_I(v_{c_1})$ can have a color greater than $k+1$. Otherwise, let v_x be a vertex in $N_I(v_{c_1})$ with color

x, such that $k+1 < x \leq n_C$. Therefore there exists a color y such that $y \leq k+1$ and y has not appeared on any vertex in $N_I(v_{c_1})$. Let v_{c_x} and v_{c_y} be the vertices of C having colors x and y, respectively. By exchanging the colors of v_{c_x} and v_{c_y} we can assign color y to v_x. This exchange reduces the total sum of the colors of P, which is a contradiction.

So, to find an optimum coloring of G, using n_C colors, we select one of the vertices of C, call v_{c_1}, and assign color 1 to it and to all vertices in $I - N_I(v_{c_1})$. Then we consider all possible assignments of colors $2, 3, \ldots, k+1$ to the vertices of $N_I(v_{c_1})$. Note that the degree of v_{c_1} is at most k. Since k is fixed, there are a constant number of such assignments. Also, for each assignment of colors to the vertices in $N_I(v_{c_1})$, we have to find a coloring for the uncolored vertices of C that is feasible with coloring of $N_I(v_{c_1})$. To do so we construct a bipartite graph $G'(X \cup Y, E')$, such that $X = \{x_1, x_2, \ldots, x_{n_C - 1}\}$, $Y = \{y_2, y_3, \ldots, y_{k+1}\}$, and $x_i y_j \in E'$ if and only if there is no edge between the ith uncolored vertex of C and the vertex in $N_I(v_{c_1})$ with color j. It's not difficult to see that using a bipartite matching in G' that covers all the vertices in Y (if there exists such a matching), we can find those vertices of C that will have colors $2, 3, \ldots, k+1$, and then color the other vertices of C with colors $k+2, \ldots, n_C$ arbitrarily. By taking the minimum between the total sum of the colors of each of these colorings, we find the sum of the optimum coloring using n_C colors. Finally, we have to take the minimum between this amount and the total sum of the coloring using $n_C + 1$ colors.

Finding a maximum matching in G' can be done in time $O(n_C)$, by applying the augmenting path algorithm k times. Therefore, for each assignment of colors to the vertices in $N_I(v_{c_1})$ we spend $O(n_C)$ time to complete the coloring. Also, at the first stage, there are n_C choices for selecting the vertex v_{c_1}. Thus, overall we spend $O(n_C^2)$ to find the minimum sum of colors between all colorings using n_C colors.

Theorem 3.4 *If $G(C \cup I, E)$ is a k_C-split graph, for fixed k, then the chromatic sum and also an optimum vertex coloring of G can be computed in $O({n_C}^2)$, where $|C| = n_C$.*

Corollary 3.5 *The chromatic sum of k-split graphs can be computed in time $O({n_C}^k)$.*

Proof: Since any k-split graph is a k_I-split and a k_C-split graph, the result follows from theorems 3.3 and 3.4. ■

3.2 Vertex sum coloring of P_4-reducible graphs

Cographs were rediscovered several times and under different names and definitions, which are all equivalent. Some of these names are: *D^*-graphs, Hereditary Dacey graphs,* and *2-parity graphs*. Many problems that are NP-complete for arbitrary graphs have polynomial time solutions, when restricted to the class of cographs. Most of these algorithms use the following recursive definition of cographs:

Theorem 3.6 *Let $G(V, E)$ be a graph*

1. *If $|V| = 1$, then G is a cograph.*

2. *If $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are cographs, then $G(V_1 \cup V_2, E_1 \cup E_2)$ is a cograph.*

3. *If $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are cographs, then $G = (V_1 \cup V_2, E_1 \cup E_2 \cup \{xy | x \in V_1, y \in V_2\})$ is a cograph.*

The operation for making G from G_1 and G_2 in the second part of the theorem, is called *union*, and the operation in the third part, is called *join*. We can also replace the third operation in this theorem with a *complement* operation.

So a cograph is a graph that can be obtained from single vertices by a finite sequence of union and join operations.

Jansen [22] gives an algorithm for solving the OCCP problem for cographs. Clearly, we can solve the sum coloring problem for cographs using this algorithm. We extend this result by giving a polynomial time algorithm for finding the optimum vertex coloring of the class of P_4-reducible graphs, which is a superclass of cographs. P_4-reducible graphs were introduced by Jamison and Olariu [19] as a generalization of cographs:

Definition 3.7 *A graph G is P_4-reducible if every vertex belongs to at most one P_4.*

Our result is interesting since P_4-reducible graphs form a proper superset of cographs and a proper subset of permutation graphs, and by Jansen [22] the OCCP problem is NP-complete for permutation graphs. As far as we know, the time complexity of the sum coloring problem is not known for permutation graphs, but we expect it to be NP-complete.

Similar to cographs, P_4-reducible graphs have also a recursive definition. Before giving this definition as a theorem, we define another type of operation on graphs.

Definition 3.8 *Let $G(V, E)$ be a graph such that two adjacent vertices b and c in V are each adjacent to all other vertices in V. We define the* add tail *operation on G, by connecting an additional vertex a to b and an additional vertex d to c. The resulting graph is*

$$G' = (V \cup \{a, d\}, E \cup \{ab, cd\}).$$

Jamison and Olariu [19] proved the following theorem :

Theorem 3.9 *G is a P_4-reducible graph if and only if it can be obtained from a single vertex by a finite sequence of union, join and add tail operations.*

For every class of graphs that has a recursive definition using some predefined operations, (like cographs and P_4-reducible), there is a natural way of associating a tree, whose nodes correspond to the operations used in constructing the graph, and whose leaves are precisely the vertices of the graph. We call the corresponding tree of cographs, a *cotree*, and the corresponding tree of P_4-reducible graphs, a *pr-tree*. If v is an internal node of a pr-tree, we call the subgraph induced by the leaves of the subtree with root v, $G_{T(v)}$. For the pr-tree T of graph G, an internal node v is labeled union, join or add tail according to the following rule:

$$\text{label}(v) = \begin{cases} union & \text{iff } \mathrm{G_{T(v)}} \text{ is disconnected} \\ join & \text{iff } \overline{\mathrm{G}}_{\mathrm{T(v)}} \text{ is disconnected} \\ addtail & \text{otherwise.} \end{cases}$$

Corneil, Perl, Stewart [8] give a linear time recognition algorithm for cographs. If G is a cograph, this algorithm also gives the cotree of G. Also a linear time recognition algorithm for P_4-reducible graphs is in Jamison and Olariu [20] which also produces the corresponding pr-tree.

For our problem, we assume that a P_4-reducible graph G with its pr-tree is given and we want to find an optimum vertex coloring.

3.2.1 Max-IS Algorithm

The algorithm we present, solves the OCCP problem, which is a more general problem and can be used to solve the sum coloring problem by letting the color costs be $1, 2, \ldots, n$.

A *maximum independent set partitioning* (Max-ISP) is a partitioning of the vertices into a sequence of independent sets such that each independent set is a maximum independent set in the remainder of the graph. That is, if we call the ith independent set in the sequence b_i, then b_i is a maximum independent set of vertices of $G[V - \{b_1 \cup b_2 \cup \ldots, b_{i-1}\}]$. A *maximal independent set partitioning* is defined similarly, where each independent set is a maximal independent set in the remainder of the graph. The partitioning of vertices that the algorithm finds is a maximum independent set partitioning (Max-ISP), and we call the algorithm *Max-IS*. We will show that it is enough to find a Max-ISP for G and assign color c_i to the vertices of the class b_i.

The algorithm works recursively on the pr-tree T of G. It starts from the root of T, and goes down the tree. The general step of the algorithm has the following input and output:

Input: A node v of T, and $T(v)$, the subtree rooted at v.

Output: A Max-ISP of $G_{T(v)}$.

The base case of the algorithm is when v is a leaf, and the Max-ISP of a leaf is the leaf itself. Now we define the rules of the algorithm when v is an internal node or the root. Assume that the number of children of v is m ($m \geq 2$). Call its children $w_1, w_2, \ldots w_m$. We have three cases for the type of the node v:

- **v is a union node:**

  Therefore the subgraph $G_{T(v)}$ is disconnected. Find the Max-ISP of each of the subgraphs $G_{T(w_1)}, G_{T(w_2)}, \ldots, G_{T(w_m)}$ recursively. Call the j'th independent set of the Max-ISP of $G_{T(w_i)}$, b_{ij}, and let s_i denote the number of independent sets in $G_{T(w_i)}$. Then the j'th independent set of the Max-ISP of $G_{T(v)}$, denoted by b_j, will be $\bigcup_{i=1}^{m} b_{ij}$. Note that b_{ij} will be empty, if $j > s_i$. So by this definition, the number of maximum independent sets found for $G_{T(v)}$, is equal to $max\{s_i\}_{1 \leq i \leq m}$.

- **v is a join node:**

  Thus $G_{T(v)}$ is the join of the subgraphs $G_{T(w_1)}, G_{T(w_2)}, \ldots, G_{T(w_m)}$. Again, find the Max-ISP of each of these subgraphs recursively. As before, call the j'th independent set of the Max-ISP of $G_{T(w_i)}$, b_{ij}, and let s_i denote the number of independent sets in $G_{T(w_i)}$. The total number of independent sets over all the subgraphs $G_{T(w_1)}, G_{T(w_2)}, \ldots, G_{T(w_m)}$, will be $\sum_{i=1}^{m} s_i = S$. Sort all of these independent sets by their increasing size and return the sorted list.

- **v is an add tail node:**

  In this case, v has two children, w_1 and w_2, one corresponds to a graph consisting of two non-adjacent vertices a and d, and the other one corresponds to a P_4-reducible graph. Let's assume, without loss of generality, that $G_{T(w_1)}$ is the one having a and d, and let the vertices of $G_{T(w_2)}$ that are connected to a and d, be b and c respectively. Find the Max-ISP of $G_{T(w_2)}$ recursively. Since b and c are each adjacent to all other vertices of $G_{T(w_2)}$, each of them is itself a maximum independent set in this partitioning, and of course has size one. So we can reorder (to be proven later) the sequence of Max-ISP of $G_{T(w_2)}$ such that these two independent sets are at the end of this sequence. In other words, we can change these two independent sets with the last two in the sequence of Max-ISP, and it will be a Max-ISP as well.

Now, to find a Max-ISP for $G_{T(v)}$, we continue in the following two cases:

– If the number of independent sets of $G_{T(w_2)}$ is greater than two, then add a and d to the first independent set class and return.

– If the number of independent sets of $G_{T(w_2)}$ is equal to two, then add a to the set containing vertex c, and add d to the set containing vertex b, and return.

After finding the Max-ISP for the graph G, the algorithm assigns color c_1, which has the least cost value, to the first independent set in the sequence, c_2 to the second one, and so on. We claim that this coloring is an optimum vertex coloring of G with cost values $c_1, c_2, \ldots, c_n$.

3.2.2 Correctness of the Max-IS algorithm

In this section we prove the correctness of the algorithm. To do so, we first prove some lemmas, which are used in the main theorem of this subsection. The second one proves the correctness of the main part of the algorithm, which is finding a Max-ISP for G.

Lemma 3.10 *Let G be an arbitrary graph. Then:*

- *If G is disconnected, then its chromatic number is equal to the maximum of the chromatic numbers of its components.*

- *If G is the join of some smaller subgraphs, then its chromatic number is equal to the sum of the chromatic numbers of those subgraphs.*

Proof: The first part is trivial. If G is the join of some subgraphs, say $G_1, G_2, \ldots, G_k$, then no two vertices from two distinct G_i's can have the same color in any proper coloring of G. Therefore the class of colors used in each subgraph G_i is different than the colors used for any other G_j $(i \neq j)$. Thus we need at least $\sum_{i=1}^{k} \chi(G_i)$ colors for a proper coloring of G. ∎

Lemma 3.11 *If G is a P_4-reducible graph and T is the corresponding pr-tree, then the algorithm finds a Max-ISP for G.*

Proof: We prove this lemma by induction on the size of G. The base case is when $|G| = 1$, and clearly the algorithm works correctly. Now suppose that the algorithm finds a Max-ISP for all P_4-reducible graphs of size smaller than n, and let $|G| = n$. Consider node r, the root of tree T, and its children $w_1, w_2, \ldots, w_m$. We have three cases for the type of r:

- **r is a union node:**

  So G is the union of the subgraphs $G_{T(w_1)}, G_{T(w_2)}, \ldots, G_{T(w_m)}$. The algorithm finds a Max-ISP for each of these subgraphs recursively, which is correct by the induction hypothesis, because all of them have smaller size than $|G|$. Recall from the algorithm that the j'th independent set of graph $G_{T(w_i)}$ is called b_{ij}. Because G is disconnected, any maximum independent set of G restricted to a component of it induces a maximum independent set. Thus $\bigcup_{i=1}^{m} b_{i1}$ is a maximum independent set of G. In general, $\bigcup_{i=1}^{m} b_{ij}$ is a maximum independent set of $G - \{b_1, b_2, \ldots, b_{j-1}\}$ and hence the algorithm finds a Max-ISP for G in this case.

- **r is a join node:** So G is the join of the subgraphs $G_{T(w_1)}, G_{T(w_2)}, \ldots, G_{T(w_m)}$. Again, by the induction hypothesis the algorithm finds a Max-ISP for each of these subgraphs recursively. Since G is the join of $G_{T(w_1)}, G_{T(w_2)}, \ldots, G_{T(w_m)}$, any independent set, and in particular any maximum independent set of G, will be contained completely in one of these subgraphs. Therefore, each independent set in any Max-ISP of G is a maximum independent set in the remainder of one of its components. Thus, each b_{ij} will be an independent set in a Max-ISP of G. So it's enough to sort all of them by their size and the resulting sequence is a Max-ISP for G.

- **r is an add tail node:** Recall the part of the Max-IS algorithm where the internal node is an add tail node. So r has two children, w_1 and w_2, where $G_{T(w_1)}$ is just two non-adjacent vertices. By the induction hypothesis, the algorithm finds a Max-ISP for $G_{T(w_2)}$. Since b and c are adjacent to all other vertices in $G_{T(w_2)}$, each of them will be a maximum independent set in the Max-ISP of $G_{T(w_2)}$.

  If there is any vertex other than b and c in $G_{T(w_2)}$, then we can put the independent sets of b and c at the end of the Max-ISP of $G_{T(w_2)}$. Also, a and d are not adjacent to any vertex, other than b and c. Thus the maximum independent set of G will contain the maximum independent set of $G_{T(w_2)}$, which is neither $\{b\}$ nor $\{c\}$, union $\{a, d\}$.

  If there is no vertex, other than b and c in $G_{T(w_2)}$, then the maximum independent sets of G are just $\{a, c\}$ and $\{b, d\}$. So in both cases, the algorithm finds a Max-ISP for G.

Therefore, in all of these three cases for the type of root of T, the Max-IS algorithm finds a Max-ISP for G. ■

Lemma 3.12 *Let G be a P_4-reducible graph. Then the number of independent sets found by the Max-IS algorithm for G, is equal to $\chi(G)$.*

Proof: We prove it by induction on the size of graph. The base case, where $|G| = 1$ is trivial. Now suppose that the statement is valid for all graphs of size smaller than n, and let $|G| = n$. Let T be the pr-tree of graph G, and consider type of the node r, the root of T:

- r **is a union node:** So G is disconnected. Call its components $G_1, G_2, \ldots, G_m$. Since the size of each G_i is smaller than n, we can apply the induction hypothesis to it. So the algorithm partitions each G_i into $\chi(G_i)$ independent sets. As we mentioned in part one of the algorithm, in this case the number of independent sets found for G, is equal to the maximum of number of independent sets of its components. Therefore the number of cells in the partition of G is the maximum of the chromatic numbers of its components. By lemma 3.10, this is equal to the chromatic number of G.

- r **is a join node:** So G is the join of some smaller subgraphs, say $G_1, G_2, \ldots, G_m$. The size of each G_i is smaller than n, and again we can apply the induction hypothesis to them. Therefore the algorithm partitions each G_i into $\chi(G_i)$ independent sets. According to part two of the algorithm, the number of cells in the partition of G is the sum of the number of independent sets of all of these subgraphs, which is equal to $\sum_{i=1}^{m} \chi(G_i)$. By lemma 3.10, this number is equal to $\chi(G)$.

- r **is an add tail node:** Let w_1 and w_2 be the two children of r, where $G_{T(w_1)}$ is the two non-adjacent vertices a and d. We have $|G_{T(w_2)}| = n - 2$, and so we can use the induction hypothesis. Therefore the number of maximum independent sets of $G_{T(w_2)}$, found by the Max-IS algorithm, is equal to $\chi(G_{T(w_2)})$. Also $G_{T(w_2)}$ is a subgraph of G and therefore $\chi(G) \geq \chi(G_{T(w_2)})$. The Max-IS algorithm partitions G into the same number of independent sets as it does for $G_{T(w_2)}$. Therefore, G is partitioned into $\chi(G_{T(w_2)})$ independent sets, which is certainly not more than $\chi(G)$.

■

Now we are ready to prove the main theorem of this section.

Theorem 3.13 *If $b_1, b_2, \ldots, b_k$ is a Max-ISP for a P_4-reducible graph G of size n, and $c_1 \leq c_2 \leq \ldots \leq c_n$ are the given color cost values, then assigning the color with cost c_i to the vertices of class b_i, gives an optimum vertex coloring for G.*

Proof: We prove this by induction on n. The base case, where $|G| = 1$, is trivial. Now suppose that the theorem is correct for all graphs of size smaller than n. Let P be an optimum vertex coloring of G, and let $p_1, p_2, \ldots, p_l$ be the sequence of independent sets of P. From the observations at the end of section 1.3, this is a maximal independent set partitioning. Consider the pr-tree T of graph G. Let r be the root of T, which has m children $w_1, w_2, \ldots, w_m$. We have three cases for the type of r:

- r **is a union node:** In this case, G is disconnected and is the union of the subgraphs $G_{T(w_1)}, G_{T(w_2)}, \ldots, G_{T(w_m)}$. Let p_{ij} be the subset of the independent set p_j, restricted to the subgraph $G_{T(w_i)}$. So $p_{i1}, p_{i2}, \ldots, p_{it_i}$ are independent sets of $G_{T(w_i)}$, where t_i is the number of them. In any optimum vertex coloring, the vertices of p_j, and therefore the vertices of p_{ij}, all have color c_j. Since the size of each $G_{T(w_i)}$ is smaller than n, we can use the induction hypothesis for it. The algorithm finds a Max-ISP for $G_{T(w_i)}$, which is $b_{i1}, b_{i2}, \ldots, b_{is_i}$, and assigns color c_j to the vertices of the class b_{ij}. By lemma 3.12, we have $s_i = \chi(G_{T(w_i)}) \leq t_i$. Now using the induction hypothesis, this is an optimum vertex sum coloring, for the subgraph $G_{T(w_i)}$. So for the subgraph $G_{T(w_i)}$ we have:

  $$\sum_{j=1}^{s_i} c_j |b_{ij}| \leq \sum_{j=1}^{t_i} c_j |p_{ij}|.$$

  But we know that:

  $$b_j = \bigcup_{i=1}^{m} b_{ij} \quad \text{and} \quad p_j = \bigcup_{i=1}^{m} p_{ij}.$$

  Thus we have:

  $$\sum_{j=1}^{k} c_j b_j = \sum_{j=1}^{k} c_j |\bigcup_{i=1}^{m} b_{ij}| = \sum_{j=1}^{k} c_j \sum_{i=1}^{m} |b_{ij}| =$$

  $$\sum_{j=1}^{k} \sum_{i=1}^{m} c_j |b_{ij}| = \sum_{i=1}^{m} \sum_{j=1}^{s_i} c_j |b_{ij}| \leq \sum_{i=1}^{m} \sum_{j=1}^{t_i} c_j |p_{ij}| =$$

  $$\sum_{j=1}^{l} \sum_{i=1}^{m} c_j |p_{ij}| = \sum_{j=1}^{l} c_j \sum_{i=1}^{m} |p_{ij}| = \sum_{j=1}^{l} c_j |\bigcup_{i=1}^{m} p_{ij}| = \sum_{j=1}^{l} c_j p_j.$$

  Therefore the cost of the coloring of the Max-IS algorithm is not more than any optimum vertex coloring.

- r **is a join node:** So G is the join of the subgraphs $G_{T(w_1)}, G_{T(w_2)}, \ldots, G_{T(w_m)}$. Each p_i must be contained completely in one of these subgraphs. Let's call the subsequence of $p_1, p_2, \ldots, p_l$ that are subsets of $G_{T(w_i)}$, $p_{i1}, p_{i2}, \ldots, p_{it_i}$. So $p_{i1}, p_{i2}, \ldots, p_{it_i}$ is a Max-ISP of $G_{T(w_i)}$. Also call the subsequence of $c_1, c_2, \ldots, c_n$ that is used by the coloring P for the vertices of $G_{T(w_i)}$, $c_{i1}, c_{i2}, \ldots, c_{it_i}$. Note that p_{ij} (for all i, j) is equal to p_x for some $1 \leq x \leq l$, and c_{ij} is equal to c_x. In the Max-IS algorithm, first we find the Max-ISP of each $G_{T(w_i)}$, which is $b_{i1}, b_{i2}, \ldots, b_{is_i}$. Now we use the induction hypothesis for each $G_{T(w_i)}$: $|G_{T(w_i)}| < n$ and the sequence of $b_{i1}, b_{i2}, \ldots, b_{is_i}$ gives a Max-ISP for $G_{T(w_i)}$. Also, $s_i \leq t_i$ by lemma 3.12. Thus assigning color c_{ij} to the vertices of b_{ij}, is an optimum vertex coloring and therefore its cost is not more than the cost of the coloring P, which is:

$$\sum_{j=1}^{t_i} c_{ij} p_{ij}.$$

  This assignment of colors to the sets b_{ij} gives a coloring for G, whose cost is not more than the cost of the optimum vertex coloring P. But we have to prove that if we sort all b_{ij}'s by their size and assign color c_i to the i'th set, it gives a coloring which is not worse than this one. Actually, this is not difficult to show. We prove by contradiction. Assume that for some i, j and some i', j', we have:

$$|b_{ij}| > |b_{i'j'}| \quad \text{and} \quad |c_{ij}| > |c_{i'j'}|.$$

  It can be seen easily that if we exchange the color of the vertices of sets b_{ij} and $b_{i'j'}$, then the resulting coloring is not worse than the former one. Therefore, if we assign the smaller color to the larger set of b_{ij}'s (in the same way that the Max-IS algorithm works), it gives a coloring which is not worse than the coloring in which the vertices of b_{ij} have color c_{ij}. This completes the proof of this case.

- r **is an add tail node:** Let w_1 be the child of r where $G_{T(w_1)}$ is just two non-adjacent vertices a and d, and w_2 be the other child. In any optimum vertex sum coloring of G, vertices a and d have at least color c_1. Thus:

$$\Sigma(G) \geq \Sigma(G - \{a, d\}) + 2c_1.$$

  If $G = P_4$, then trivially $\Sigma(G) = 2c_1 + 2c_2$, and the coloring that the Max-IS algorithm finds has the same cost. Otherwise, there is some vertex, other than b and c, in $G - \{a, d\}$. By the induction hypothesis, the algorithm finds an optimum vertex sum coloring for $G - \{a, d\}$, and we know that none of b and c have color c_1, and we assign c_1 to a and d. Thus the cost of the coloring of G by the Max-IS algorithm will be:

$$\Sigma(G - \{a, d\}) + 2c_1 \leq \Sigma(G).$$

Hence the algorithm finds an optimum vertex coloring of G in this case.

■

3.2.3 Analysis of the Algorithm

In this section, we analyze the time complexity of the algorithm. As we mentioned before, the pr-tree of a P_4-reducible graph can be constructed in linear time. The main phase of the algorithm is to find a Max-ISP of the given graph. Then it just sorts the color costs, and assigns them to the independent sets found in the previous phase. Sorting the cost values, can be done in $O(n \log n)$. So, what lefts is the analysis of the main phase.

Suppose that $G(V, E)$ is a P_4-reducible graph, and T is its corresponding pr-tree. Each of the union, join and add tail operations needs at least two subgraphs to operate on. In other words, each internal node of T, has at least two children. Also, we know that the number of leaves of T, is equal to $|V|$. So the height of T is at most in $O(n)$. It is easy to see that the number of nodes of T is bounded by $2n - 1$.

Consider the node v of tree T, and the step of the algorithm where we want to find the Max-ISP of $G_{T(v)}$ using the Max-ISP of its children. We claim that computing $b_1, b_2, \ldots, b_k$, the Max-ISP of $G_{T(v)}$, takes $O(|G_{T(v)}|)$ time, assuming that the Max-ISP of its children are computed. If v is a leaf then it takes constant time to do so. Assume that v is an internal node, with children $w_1, w_2, \ldots, w_m$. Suppose that we have computed the Max-ISP of them. Recall from the algorithm that $b_{i1}, b_{i2}, \ldots, b_{is_i}$ is the Max-ISP of the i'th child of v. According to the algorithm:

- v **is a union node:** To compute b_j, the j'th independent set of $G_{T(v)}$, we have to take the union over all b_{ij} $(1 \leq i \leq m)$, which takes $O(|b_j|)$ time. Thus, this induction step takes $\sum_{i=1}^{k} |b_i| \leq O(|G_{T(v)}|)$ time.

- v **is a join node:** All we have to do is to merge the m pre-sorted lists of Max-ISP of the children of v. Clearly the time this takes is:

$$\sum_{i=1}^{m} \sum_{j=1}^{s_i} |b_{ij}| \leq \sum_{i=1}^{m} |G_{T(w_i)}| = |G_{T(v)}|.$$

- **v is an add tail node:** In this case, it takes constant time to convert the Max-ISP of the subgraphs induced by children of v to a Max-ISP of $G_{T(v)}$.

So if v is the root of the subtree, in any of the above three cases, it takes $O(|G_{T(v)}|)$ time to compute the Max-ISP of $G_{T(v)}$, using the Max-ISP of children of v. Since the maximum height of T is in $O(n)$, each leaf participates in the calculation, at most $O(n)$ times. We have proved the following theorem:

Theorem 3.14 *For a given P_4-reducible graph G of size n, the time complexity of the Max-IS algorithm is $O(n^2)$.*

3.3 Vertex sum coloring of chain bipartite and co-bipartite graphs

As we mentioned before, vertex sum coloring seems to be harder than standard vertex coloring. The NP-completeness of the vertex sum coloring problem for the class of bipartite graphs is proved by Bar-noy et al. [3]. So it's natural to consider the complexity of this problem for subclasses of bipartite graphs.

In this section we consider two families of graphs, related to bipartite graphs, and give polynomial time solutions for finding an optimum vertex coloring for each family. The first family is the family of chain bipartite graphs. The notion of chain graphs was introduced by Yannakakis [39, 40]. Then we look at the family of cobipartite graphs and use the matching technique to find their chromatic sum.

Definition 3.15 *A bipartite graph $G(X \cup Y, E)$ is called a chain graph if for every two vertices $x_i, x_j \in X$, we have either $N(x_i) \subseteq N(x_j)$ or $N(x_j) \subseteq N(x_i)$.*

In other words, there is an ordering of the vertices of X, $x_{\pi_1}, x_{\pi_2} \ldots, x_{\pi_n}$, where n is the size of X, such that $N(x_{\pi_i}) \subseteq N(x_{\pi_{i+1}})$, $1 \leq i < n$. We call this property, the *chain* property.

It's not difficult to see that if we have an ordering of the vertices of part X having the chain property, we can find an ordering of the vertices of Y with this property, as well. This shows that the definition of chain bipartite graphs is unambiguous. Now we describe the algorithm for finding the chromatic sum of a chain bipartite graph.

Let $G(X \cup Y, E)$ be a chain bipartite graph. Without loss of generality, we assume that the graph is connected. Also, let $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_m$ ($|X| = n$ and

$|Y| = m$) be the orderings of X and Y respectively, having the chain property. By G_{ij} we mean the induced subgraph of G on vertices $\{x_1, x_2, \ldots, x_i\} \cup \{y_1, y_2, \ldots, y_j\}$.

For a pair (i, j) where G_{ij} has no edges, consider the following vertex sum coloring of G: Assign color 1 to all of the vertices of G_{ij}. If $n - i \geq m - j$ then assign color 2 to $x_{i+1}, \ldots, x_n$ and assign color 3 to $y_{j+1}, \ldots, y_m$. Otherwise, if $n - i < m - j$ then assign color 3 to $x_{i+1}, \ldots, x_n$ and assign color 2 to $y_{j+1}, \ldots, y_m$. Let S_{ij} be the total sum of this coloring. We call a pair (i, j) a *proper pair* if the set of vertices of G_{ij} is a maximal independent set of G. Let S_{min} be the minimum of S_{ij} for all proper pairs (i, j). The algorithm computes S_{min} and returns the minimum of $\{S_{min}, 2n_x + n_y, n_x + 2n_y\}$ as the chromatic sum of G. The last two values are the total cost of the colorings in which the vertices of one part all have color 1 and the vertices of the other part all have color 2. We refer to figure 1.1 to see why assigning color 1 to the vertices of one part and assigning color 2 to the vertices of the other part does not necessarily give an optimum vertex coloring of a chain bipartite graph.

Theorem 3.16 *The chromatic sum of G is equal to the minimum of $\{S_{min}, 2n_x+n_y, n_x+2n_y\}$.*

Proof: Let C be an optimum vertex coloring of G. We denote the color of vertex v by $c(v)$. If no vertex in X has color 1, then all of the vertices in Y must have color 1 and so all the vertices in X must have color 2. Thus the total cost of C will be $2n_x + n_y$. Similarly if no vertex in Y has color 1, then all of the vertices in X must have color 1, and all the vertices in Y must have color 2, and the total cost of C will be $n_x + 2n_y$.

Now assume that there is at least one vertex in X and at least one vertex in Y such that both of them have color 1. Let i be the largest number such that $c(x_i) = 1$, and let j be the largest number such that $c(y_j) = 1$. Since $c(x_i) = 1$ no vertex in $N(x_i)$ can have color 1. Also, we know that C is an optimum vertex coloring, and $N(x_k) \subseteq N(x_i)$, for $k < i$. Therefore, all the vertices $x_1, x_2, \ldots, x_{i-1}$ must have color 1, too. Similarly all the vertices $y_1, y_2, \ldots, y_{j-1}$ must have color 1. Thus the vertex set of G_{ij} is an independent set.

It follows from the definition of i and j that the color of the vertices $x_{i+1}, \ldots, x_{n_x}$ and $y_{j+1}, \ldots, y_{n_y}$ must be greater than 1. So each of them must be connected to a vertex having color 1, otherwise we could simply change its color to 1, which reduces the total sum of colors. In particular, x_{i+1} is connected to y_a for some $a \leq j$, and y_{j+1} is connected to x_b for some $b \leq i$. Because of the chain property it follows that x_i is connected to all of the vertices $y_{j+1}, \ldots, y_{n_y}$, and y_j is connected to all of the vertices $x_{i+1}, \ldots, x_{n_x}$, and the subgraph $G - G_{ij}$ is a complete bipartite subgraph. Therefore G_{ij} is a maximal graph such that its vertex set is an independent set. Thus (i, j) is a proper pair.

Note that if $k > i$ and $l > j$, then no two vertices x_k and y_l, can have the same color. Now it's clear that the vertices of the larger of the two sets $\{x_{i+1}, \dots, x_{n_x}\}$ and $\{y_{j+1}, \dots, y_{n_y}\}$ must be colored with color 2, and the vertices of the smaller one with color 3. This kind of coloring is the same as the one we use in the algorithm when we select a proper pair. Therefore by considering all proper pairs, we will eventually consider the proper pair (i, j) and compute the cost of coloring C, which is equal to the chromatic sum of G.

■

To find a proper pair (i, l) for a fixed i, first we find the largest number j such that $x_i y_j \notin E$. If such a number does not exists then it's clear that there is no proper pair having i as the first element.

We now show that (i, j) is a proper pair and that it is the only proper pair having i as the first element. This follows from the fact that, if $x_a y_b \in E$, $1 \leq a \leq i$ and $1 \leq b \leq j$, then because of the chain property, $x_i y_b \in E$. Since $b \leq j$, this implies that $x_i y_j \in E$, which is a contradiction.

Therefore, to find a proper pair with (i, j) for a fixed i, it takes at most $O(deg(x_i))$ time [1]. So overall, the time complexity of finding all proper pairs is $O(|E|)$. We have proved the following theorem:

Theorem 3.17 *We can find the chromatic sum of chain bipartite graphs in $O(|E|)$.*

Another family of graphs for which the chromatic sum (and also an optimum vertex coloring) can be computed efficiently is the class of cobipartite graphs. A graph G is cobipartite if it is isomorphic to the complement of a bipartite graph, i.e its vertex set can be partitioned into two disjoint sets, each inducing a complete subgraph. We explain how to derive the chromatic sum of such a graph using bipartite matching.

Assume that we are given a cobipartite graph $G(A \cup B, E)$ where the subgraphs induced on A and B, denoted by G_A and G_B respectively, are both complete. It follows that the number of vertices of each color class in any optimum vertex coloring of G is at most two. Otherwise, there are at least two vertices of the same color in part A or B, which is a contradiction.

So we have some color classes of size one, and some color classes of size two such that each of them has one vertex x in A and one vertex y in B, such that $xy \notin E$. It is clear

[1] Depending on how the graph is given, we may be able to find the pair (i, j) in constant time, but the total time of reading the graph from input is at least $O(|E|)$

that to get a minimum total sum, we must have as many classes of size two as possible and assign the smallest colors to them.

Let $G'(A \cup B, E')$ be the complement of G. So G' is bipartite. Let $M = \{v_1u_1, v_2u_2, \ldots, v_ku_k\}$ be a maximum matching in G', where $|M| = k$. It is trivial that $u_iv_i \notin E$, $1 \leq i \leq k$, and there are at most k such pairs in E. Therefore to obtain an optimum vertex coloring of G, we assign color i to vertices u_i and v_i, $1 \leq i \leq k$, and color arbitrarily each of the remaining vertices with one of the colors $k+1, k+2, \ldots, |V| - 2k$. Computing the maximum bipartite matching can be done in $O(|E||V|^{0.5})$ as follows from Even and Tarjan [12]. We summarize the above arguments in the following theorem:

Theorem 3.18 *The vertex sum coloring problem can be solved on cobipartite graphs in $O(|E||V|^{0.5})$ time.*

Note that Jansen [22] independently has proved this theorem using the same technique.

Chapter 4

Edge sum coloring

It is quite natural to try to extend the notion of vertex sum coloring to other kinds of graph coloring, such as edge coloring. The edge sum coloring problem on graphs asks to find a proper edge coloring, such that if E_i is the class of edges having color i, then the total sum $\sum_{i \geq 1} i|E_i|$ is minimized. We call such a coloring, an *optimum edge coloring*, and call the sum of colors the *edge chromatic sum*.

As we mentioned in section 1.4, this problem was first introduced in [17] and [2] as the vertex sum coloring of the line graph of a given graph. We know from theorem 1.8 that $\Delta \leq s'(G) \leq \Delta + 1$, and that the edge sum coloring is NP-hard for multigraphs. These are almost the only known results for edge sum coloring.

In this chapter, we give some more results on this problem. In the first section, we prove that finding the edge chromatic sum and also finding the edge strength of a simple graph are both NP-complete. In fact we prove that these problems are NP-complete even for the class of 3-regular graphs. In the second section we present a polynomial time algorithm, which uses weighted matching in bipartite graphs, to find the edge chromatic sum of trees. The algorithm can easily be modified to find an optimum edge coloring, as well. This algorithm can also be used for the case that the tree is a weighted tree, i.e a value is given for each edge of the tree, as the weight of that edge. Finally, in the third section we show how to use *Extended Monadic Second Order Logic* to find a linear time algorithm for finding the edge chromatic sum of partial k-trees with bounded degree, for fixed k.

4.1 Complexity of the edge chromatic sum and the edge strength problems

In sections 2.1 and 2.3 we saw that the vertex strength problem and the chromatic sum problem for split graphs are NP-complete.

In [2] Bar-noy et al. prove that the edge sum coloring problem is NP-complete for general multigraphs, but the complexity of this problem was left open for simple graphs. In this section, we prove that the edge sum coloring problem is NP-complete for simple graphs. In particular, we show that finding the edge chromatic sum and finding the edge strength of a cubic graph are both NP-complete. A cubic graph is a graph whose vertices all have degree three. We use the reduction from the chromatic index problem restricted to cubic graphs:

Instance: A cubic graph G.

Question: Is $\chi'(G) = 3$?

Holyer [18] proves that this problem is NP-complete. By Vizing's theorem we know that $\Delta \leq \chi'(G) \leq \Delta + 1$. So any cubic graph is 4-edge colorable. Also, by theorem 1.8 $\Delta \leq s'(G) \leq \Delta + 1$. So the edge strength of a cubic graph is also either 3 or 4. We prove that deciding whether $s'(G) = 3$ for a cubic graph is NP-complete. First we show that finding the edge chromatic sum of a cubic graph is NP-complete:

Instance: A cubic graph G of size n.

Question: Is $\Sigma'(G) = 3n$?

Note that since the degree of each vertex of a cubic graph is three, in any edge coloring of G the total sum of the colors of the edges of a vertex is at least $1 + 2 + 3$. Therefore, $\Sigma'(G) \geq 3n$.

Theorem 4.1 *The edge chromatic sum problem is NP-complete for cubic graphs.*

Proof: First of all, it is trivial that this problem belongs to NP, since we can easily verify in polynomial time whether a given edge coloring is a proper one or not, and if its total sum is equal to $3n$ or not.

To prove the NP-completeness we use a reduction from the chromatic index problem. We prove that for the cubic graph G:

$$\Sigma'(G) = 3n \Longleftrightarrow \chi'(G) = 3.$$

First, assume that $\chi'(G) = 3$. This means that there exists a 3-edge coloring of G, called C. Since the degree of each vertex in G is three, all numbers $1, 2, 3$ must appear on the edges incident with each vertex. So the total sum of the colors in C is equal to $3n$. Therefore $\Sigma'(G) = 3n$.

Now, suppose that $\Sigma'(G) = 3n$. It suffices to prove that any optimum edge coloring of G is a 3-edge coloring of G. Assume, by way of contradiction, that C is an optimum edge coloring of G with 4 colors. So there exists at least one vertex such that color 4 is the color of one of its incident edges. Therefore the total sum of the colors of the edges incident with that vertex is more than 6. We have the lower bound 6 for the sum of the colors of the edges of every other vertex. Therefore the total sum of colors of the edges of the graph will be strictly greater than $3n$, which is a contradiction. Therefore, any optimum edge sum coloring of G is also a 3-edge coloring of G, and so $\chi'(G) = 3$.

■

Corollary 4.2 *For a cubic graph G we have:*

$$s'(G) = 3 \Longleftrightarrow \chi'(G) = 3$$

Proof: If $s'(G) = 3$ then trivially $\chi'(G) = 3$. Now assume that $\chi'(G) = 3$. From the arguments we had in the proof of theorem 4.1 it follows that if $s'(G) > 3$ then $\Sigma'(G) > 3n$. Also we know that the sum of any 3-edge coloring of G is $3n$. This proves that in this case $s'(G) = 3$. ■

Since finding the chromatic index is NP-complete for the class of cubic graphs, therefore:

Theorem 4.3 *Finding the edge strength of cubic graphs is NP-complete.*

4.2 Edge sum coloring of trees

In this section, we give a polynomial time algorithm that finds the edge chromatic sum of trees. This algorithm uses the dynamic programming method. We can find an optimum edge coloring as well, by storing some extra information in the data tables. Before that, we give an upper bound for the edge strength of bipartite graphs.

We know the upper bound $\Delta+1$ for the edge strength from theorem 1.8. We prove, as a lemma, that for bipartite graphs the value of the edge strength is equal to the maximum degree of the graph, and will use this fact in our algorithm.

Lemma 4.4 *If G is a bipartite graph with maximum degree Δ, then $s'(G) = \Delta$.*

Proof: Suppose that the lemma is not true, and let G be a minimal counter example, with respect to the number of edges. That is $s'(G) \geq \Delta + 1$. First of all, we claim that there exists an optimum edge coloring of G, in which there is just one edge with color s'. To prove this, let uv be an edge with the color s'. Because G is a minimal counter example, if we remove any edge from G, in particular uv, the resulting graph has edge strength of at most Δ. Thus we can consider an optimum edge coloring of $G - uv$, which uses Δ colors, and assign the color s' to uv.

Now, consider the edge uv, which has color s'. For each of u and v, there is at least one color that does not appear on its adjacent edges. Let i be the color which does not appear at u, and j be the color that does not appear at v. We know that $i < s'$ and $j < s'$. Clearly i must appear at v, say on the edge vy, otherwise we can simply change the color of uv to i, which removes the color s', and also decrease the sum of colors, both are contradictions. Similarly j appears at u, say on edge ux.

Let G_{ij} be the subgraph of G, which contains just the edges with color i or j. Consider the component of G_{ij} which contains the edge ux. Since the degree of each vertex in G_{ij} is at most two, this component is a path that starts from ux. But this path does not contain the edge vy. Otherwise, this path together with the edge uv makes an odd cycle in G, contradicting G being bipartite. Therefore the component of G_{ij} which contains ux is different from the one that contains vy. Thus we can simply exchange the colors i and j in the component containing ux, and let the color of uv be j. It can be easily seen that this exchange does not increase the total sum of colors, and removes the color s' which is a contradiction. ■

Assume that we are given tree T of size n, with a Breath First Search ordering, rooted at a vertex with the largest degree. For vertex v of T, we denote the subtree rooted at v by T_v. By *lower edges* of v, we mean the set of edges that connect v to its children. We denote the degree of vertex v by $deg(v)$. Define the *best set of k for vertex v*, to be the first $deg(v) - 1$ natural numbers, excluding the number k. Therefore, the best set of k for v is:

$$\begin{cases} \{2, 3, \ldots, deg(v)\} & k = 1 \\ \{1, 2, \ldots, deg(v) - 1\} & k \geq deg(v) \\ \{1, 2, \ldots, k-1, k+1, \ldots, deg(v)\} & 1 < k < deg(v). \end{cases}$$

If C is an edge coloring of T, the set of colors used for the lower edges of v, is denoted by L_v. The following lemma shows the existence of an optimum edge coloring with a specific structure for the lower edges of each vertex.

Lemma 4.5 *Let v be any vertex of T, and $k \in \{1, 2, \ldots, n\}$. Suppose that C is an optimum edge coloring of T, such that the color k appears on the edge joining v to its* *father. Then there exists an optimum edge coloring C', in which the colors of the edges in $E(T) - E(T_v)$ are unchanged, and L_v is equal to the best set of k for vertex v.*

Proof: Assume, by way of contradiction, that there is no such optimum edge coloring. Consider an optimum edge coloring C'', such that the colors of the edges in $T - T_v$ are the same as in C, and the first i smallest numbers in L_v, are the same as in the best set of k for vertex v, and i is the maximum possible number, between all optimum edge colorings of T.

Let j be the first color that is in the best set of k for v, and $j \notin L_v$. So instead of j, a number greater than j belongs to L_v, call it l. Assume that l is the color of the edge vu_1. Since $j \notin L_v$, j must be the color of one of the lower edges of u_1, otherwise we can simply change the color of vu_1 to j and reduce the sum of colors. Therefore $j \in L_{u_1}$. Suppose that j is the color of the edge u_1u_2. If $l \notin L_{u_2}$, then we can exchange the color of vu_1 and u_1u_2, which doesn't increase the sum of the colors of T_v, but we get an edge coloring in which the number of colors that are in both L_v and the best set of k for v is more than i, which is a contradiction. Therefore $l \in L_{u_2}$.

Using the same argument, there is a chain of edges, starting from vu_1, going down in T, such that the colors of the edges in the chain are j and l, alternatively. This chain finishes somewhere, because T is finite. The number of edges having color j is not more than the number of edges having color l, because the chain starts with the edge vu_1, which has color l. Therefore if we exchange the colors j and l in this chain, the total sum of colors won't increase. But by this exchange, we get another optimum coloring, in which the number of common colors between L_v and the best set of k for v, is more than i. This contradiction proves that our assumption is wrong. Therefore the coloring C' exists. ∎

Now we explain the algorithm which uses the dynamic programming method. Let the maximum degree of T be Δ. We have a $n \times (\Delta + 1)$ table, called S, such that:

$$S[v, j] = \text{The cost of an optimum edge coloring of } T_v \text{ such that } j \notin L_v.$$

For $1 \leq j \leq \Delta + 1$, the initial values of S are filled as:

$$\begin{cases} S[x, j] = 0 & x \text{ is a leaf} \\ S[x, j] = \infty & \text{otherwise.} \end{cases}$$

The algorithm computes the values of this table in a bottom-up way, from the leaves of the tree up to the root. It computes the value $S[v, j]$ for each internal node v, after it has computed the values for the children of v.

From the definition of best set, it follows that for all values of $m \geq deg(v)$, the best sets of m for vertex v, are all equal. So, when we have computed $S[v, deg(v)]$, we can set the values of $S[v, m]$ to $S[v, deg(v)]$, for $m > deg(v)$.

Suppose that $u_1, u_2, \ldots, u_k$ are the children of internal node v. Assume that $S[u_i, j]$ is computed, for $1 \leq i \leq k$ and $1 \leq j \leq \Delta + 1$, and we want to compute the value of $S[v, m]$ $(1 \leq m \leq \Delta+1)$, the cost of an optimum edge coloring for T_v, such that $m \notin L_v$.

Construct the complete weighted bipartite graph $G_v = (A \cup B, E')$, as follows:

$$A = \{a_1, a_2, \ldots, a_k\} \qquad \text{and} \qquad B = \{b_j | j \in \text{the best set of } m \text{ for } v\}$$
$$w(a_i b_j) = S[u_i, j] + j.$$

By $w(a_i b_j)$ we mean the weight of the edge $a_i b_j$. Now find a min-weighted maximum matching in G_v, and call it M. This matching covers all the vertices of A. We assign the colors of the lower edges of v, according to the following rule:

The color of vu_i is c_i, if edge $a_i b_{c_i}$ is in M.

It's trivial that by this assignment the value of $S[v, m]$ is equal to the sum of the weights of M, which is:

$$\sum_{e \in M} w(e) = \sum_{a_i b_{c_i} \in M} S[u_i, c_i] + c_i.$$

Knowing how to compute the value of $S[v, m]$, from the computed values of children of v, the algorithm starts from the leaves of T, and fills in the table, from bottom to up, until it computes the value of $S[r, \Delta + 1]$, where r is the root of T. This is equal to the chromatic sum of T.

Theorem 4.6 *The above algorithm computes the chromatic sum of T.*

Proof: The correctness of the algorithm follows easily from the following lemma:

Lemma 4.7 *If the values of $S[u_i, j]$ are computed for $1 \leq i \leq k$ and $1 \leq j \leq \Delta + 1$, then the above algorithm computes the value $S[v, m]$ correctly.*

Proof: Suppose that we want to compute the sum of an optimum edge coloring of T, in which $m \notin L_v$ for the internal node v. By lemma 4.5 there exists such an optimum edge coloring in which L_v is equal to the best set of m for v. Therefore, to find such an optimum edge coloring and its cost, we have to find a proper permutation of the values of the best set of m for v, and assign the i'th element of it as the color of the i'th lower edge of v.

Since we are looking for the cost of an optimum edge coloring, if the color of the edge vu_i is c_i, then the sum of the colors of the edges in T_{u_i}, is $S[u_i, c_i]$. So all we have to do, is to find the values c_i, for $1 \leq i \leq k$, such that:

$$\begin{cases} c_i \in \text{the best set of } m \text{ for } v \\ c_i \neq c_j \qquad \text{for } i \neq j \\ \sum_{i=1}^{k} S[u_i, c_i] + \sum_{i=1}^{k} c_i \ \text{ is minimized.} \end{cases}$$

Consider the bipartite graph $G_v = (A \cup B, E')$ and the min-weighted maximum matching of it, M. According to the rule of the algorithm, we select the optimum edge coloring of T_{u_i}, with the condition that $l \notin L_{u_i}$, and assign the color l to the edge vu_i. Since M is a matching, color l is different from the color of any other lower edge of v. Thus, this is a proper coloring.

We set $S[v, m]$ to the total sum of this coloring which is:

$$\sum_{e \in M} w(e) = \sum_{a_i b_{c_i} \in M} S[u_i, c_i] + c_i.$$

Since M is a min-weighted matching, the above sum is minimized. Equivalently, the cost we find is the cost of an optimum edge coloring of T_v, given that $m \notin L_v$. This proves that we compute $S[v, m]$ correctly.

■

By the previous lemma the algorithm computes the value $S[r, \Delta + 1]$, where r is the root of T, correctly. Since T is bipartite, by lemma 4.4 it needs Δ colors for an optimum edge coloring of it. So the value of $S[r, \Delta + 1]$ is equal to the edge chromatic sum of T.

■

To find an optimum edge coloring for T, we only need to keep track of the colors of the lower edges of each vertex v, when we compute $S[v, m]$. This can be easily done by storing this extra information for each entry of the table S.

The most time consuming step of the algorithm is to find the min-weighted matching. If the vertex v has k children, then the size of the bipartite graph G_v, is of $O(k)$. Note that we don't make G_v for the values of m where $m > deg(v)$. The fastest min-weighted maximum matching algorithm works in time $O(|E||V| \log |V|)$ and is due to Galil et al. [14]. Using this algorithm, for each vertex v, and each value $m \leq \deg(v)$, we spend $O(deg(v)^3 \log deg(v))$ time. Thus the total amount of time for computing the entries of the row v of the table S is $O(deg(v)^4 \log deg(v))$. Summing up these values for all v, we have the bound $O(n^4 \log n)$ for the time complexity of this algorithm. Therefore, we can say:

Theorem 4.8 *We can find an optimum edge sum coloring of a tree, and therefore its edge chromatic sum in time* $O(n^4 \log n)$.

Weighted Trees: The above algorithm can also be used for finding an optimum edge coloring of weighted trees. By a weighted tree, we mean a tree T, with a weight $w(e)$ for each edge e of T. The cost of an edge coloring $f : E \longrightarrow N$ is defined as:

$$\sum_{e \in E} w(e) f(e)$$

and the goal is to minimize the above sum. We use the same algorithm as for the regular trees. We have the table S, whose $S[v,i]$ entry gives the cost of the optimum edge coloring for the subtree T_v, with the extra condition that $i \notin L_v$. To compute the value of $S[v,i]$, we construct the bipartite graph $G_v(A \cup B, E')$ in the same way, but the the value of the weights of its edges are a bit different. If the color of the edge vu_i is j, then its contribution to the total sum is $j \times w(vu_i)$, rather than just j. Therefore, the weight of the edge $a_i b_j$ in G_v, is:

$$S[u_i, j] + j \times w(vu_i).$$

Now we follow the same steps. It's not difficult to see that the value of $S[root, \Delta + 1]$ is the cost of an optimum edge coloring of T.

4.3 Edge sum coloring of partial k-trees with bounded degree

In a long series of papers, Robertson and Seymour showed many deep results on graph minors. There, they introduced the notion of the tree-width and path-width of a graph [32, 33]. This area has been studied extensively by many others. Bodlaender [5, 6] has good surveys on this topic.

We saw in section 1.4 that the vertex sum coloring problem is polynomially solvable for graphs with bounded tree-width, i.e partial k-trees for fixed k. In this section we show how to use *Monadic Second Order Logic* to solve the edge sum coloring problem for partial k-trees with bounded degree.

Monadic second order logic (MS) is a powerful language which contains, like first order logic, propositional logic operators ($\wedge$, $\vee$, $\neg$, $\Longrightarrow$, and $\Longleftrightarrow$), individual variables (which are denoted by small letters $x, y, z, \ldots$), existential ($\exists$) and universal ($\forall$) quantifiers, and predicates. Moreover, it contains set variables $X, Y, Z, \ldots$, and the membership ($\in$) symbol. It allows existential and universal quantifiers over set variables.

We can use this language to define problems on finite graphs. Then, solving a problem for a given instance is equivalent to deciding whether the formula expressing the problem in MS, is satisfiable or not. For example, the property that a subgraph of a graph G, induced by set Z, is connected, can be stated as:

$$partition(U, V, Z) \equiv (Z = U \cup V) \wedge (U \cap V = \emptyset) \wedge (U \neq Z) \wedge (V \neq Z)$$
$$adjacent(U, V) \equiv \exists u \exists v (v \in V \wedge u \in U \wedge adj(u, v))$$
$$connected(Z) \equiv \forall U \forall V \ \ partition(U, V, Z) \Longrightarrow adjacent(U, V)$$

where adj is the adjacency relation of the vertices of the graph. Although we don't have the equality operator in the definition of MS, we used it freely, since it can be written as

$$(Z = Y) \equiv \forall x (x \in Z \Longleftrightarrow x \in Y).$$

We say a graphic problem has the MS property if it can be formulated in the MS formulation. Courcelle [9, 10] introduced the use of MS to solve problems restricted to partial k-trees. He proved that any problem that can be formulated by a MS formula has a linear time algorithm, when restricted to graphs with bounded tree-width and if a tree decomposition of the graph is given.

Later, Arnborg et al. [1] extended the definition of MS, to *Extended Monadic Second Order Logic* (EMS). An EMS formula, is a MS formula that also contains some evaluation expressions that contain cardinality of set variables, weight of elements of graphs (e.g weights of edges), arithmetic operators $(+, -, \times)$, and comparative operators. Using this extension, we can express, not only the decision problems, but also the optimization problems for graphs, such as the maximum independent set problem. An EMS property is called a *linear extremum EMS property* if the evaluation term is linear in the quantities of cardinalities of the sets. For a detailed definition of MS and EMS properties we refer to [1]. They showed how to transform a problem on partial k-trees having a MS or EMS property to a problem on binary trees. Since we can make the tree decomposition of a partial k-tree in linear time, we can decide MS or EMS properties on such graphs if we can decide MS properties on labeled binary trees.

Theorem 4.9 *[1] For the classes of graphs with bounded tree-width all problems having the MS property or the linear extremum EMS property can be decided in linear time, if the graph is given together with a tree decomposition of it.*

For example, we know that 3-color problem, which asks whether a given graph G is 3-colorable or not, is NP-complete. This problem can be formulated in MS as follows:

$$\exists X \subseteq V\ \exists Y \subseteq V\ \exists Z \subseteq V : (X \cap Y = \emptyset) \wedge (X \cap Z = \emptyset) \wedge (Y \cap Z = \emptyset) \wedge$$
$$\forall v \in V (v \in X \vee v \in Y \vee v \in Z) \wedge \forall v \in V\ \forall u \in V : uv \in E$$
$$\Longrightarrow (\neg(v \in X \wedge u \in X) \wedge \neg(v \in Y \wedge u \in Y) \wedge \neg(v \in Z \wedge u \in Z))$$

So the chromatic number problem which is NP-complete in general, is solvable in linear time for partial k-trees, for any fixed k.

To prove that the edge sum coloring problem is solvable in linear time for any partial k-tree with bounded degree and any fixed k it suffices to give a linear extremum EMS formula expressing this problem. The property that graph G can be edge colored with c colors, can be expressed as:

$$Edgecoloring(G, c) \equiv \exists E_1 \subseteq E\ \exists E_2 \subseteq E\ \ldots\ \exists E_c \subseteq E :$$
$$(E_1 \cap E_2 = \emptyset) \wedge \ldots \wedge (E_1 \cap E_c = \emptyset) \wedge \ldots \wedge (E_{c-1} \cap E_c = \emptyset) \wedge$$
$$\forall e\, (e \in E \Longleftrightarrow e \in E_1 \vee e \in E_2 \vee \ldots \vee e \in E_c) \wedge$$
$$(e_1 \in E_1 \wedge e_2 \in E_1 \Longrightarrow \neg adj(e_1, e_2)) \wedge \ldots \wedge (e_1 \in E_c \wedge e_2 \in E_c \Longrightarrow \neg adj(e_1, e_2))$$

By theorem 1.8 we know that if the maximum degree of a graph is Δ then the number of colors used in an optimum edge sum coloring of the graph is at most $\Delta + 1$. So if the degrees of the vertices of a class of graphs is bounded by some constant C_Δ then any of the graphs in that class use at most $C_\Delta + 1$ colors in any optimum edge sum coloring of it. Therefore, we can state the edge sum coloring problem for the class of graphs with maximum degree bounded by C_Δ as finding the minimum of the following evaluation term under the constrain $Edgecoloring(G, C_\Delta + 1)$:

$$\sum_{i=1}^{C_\Delta+1} i|E_i|.$$

By theorem 4.9, the edge sum coloring problem for the class of graphs with maximum degree bounded by C_Δ is directly solvable in linear time for partial k-trees from the above minimization problem.

Chapter 5

Concluding remarks

5.1 Conclusion

In this thesis we studied the sum coloring problem. We proved that finding the vertex strength of a graph with maximum degree 6 is NP-complete. By theorem 1.2 we knew that the vertex strength of a graph, in particular a tree, might be far from its chromatic number. Here we showed that for the graphs with small chromatic number ($\chi(G) \leq 4$) the vertex strength is in $O(\log n)$. To show that the sum coloring problem may be much harder (assuming $P \neq NP$) than the standard vertex coloring, we proved the NP-completeness of it for split graphs. As a consequence this problem is NP-complete for chordal graphs, whereas the standard coloring problem is in P for perfect graphs and therefore for chordal graphs.

By adding more restrictions to split graphs, we got the class of k-split graphs. We showed that although the problem is NP-complete for split graphs, if we bound the degree of the vertices of one part of a split graph, we have a polynomial time algorithm for this problem. Also, going further up in the hierarchy diagram of graphs, we gave an algorithm for this problem for the class of P_4-reducible graphs, a superclass of cographs and a subclass of permutation graphs. Efficient algorithms for cobipartite and chain bipartite graphs were presented at the end of chapter 3.

For the edge sum coloring problem, we proved the NP-completeness of this problem for cubic graphs. Also, we provided an algorithm, using the dynamic programming method and weighted matching in bipartite graphs, to find the edge chromatic sum of trees. Finally, the existence of a linear time algorithm for this problem for partial k-trees with bounded degree, for fixed k, was proved using the Monadic Second Order Logic tool.

5.2 Open problems

One of the most interesting open questions in the area of chromatic sum is conjecture 1.6 presented by Hajiabolhassan et al. [17] which states that the strength of a graph is at most $\lceil \frac{\chi(G)+\Delta}{2} \rceil$. We believe that this conjecture is true for bipartite graphs. In that case we would have the tight upper bound $1 + \lceil \frac{\Delta}{2} \rceil$ for the strength of bipartite graphs, and interestingly, trees capture this upper bound. If in fact the conjecture is true for bipartite graphs it would be interesting to see if the proof could be generalized to k-colorable graphs, for $k \geq 3$.

In the proof of the NP-completeness of the vertex strength problem, we think that the bound of 6 for the maximum degree of the graph can be improved to 4. In other words, we expect that the vertex strength problem is NP-complete for the graphs with $\Delta = 4$. Also, we don't know the complexity of deciding if the strength of a given graph is 2 or not. Since $\chi(G) \leq s(G)$ we have to consider only bipartite graphs. So for a given bipartite graph G, can we check in polynomial time if $s(G) = 2$? It doesn't seem to be solvable in polynomial time since the sum coloring problem is NP-complete for bipartite graphs.

In section 2.2 we proved that for graphs with $\chi(G) \leq 4$, $s(G) \in O(\log n)$. Although we couldn't generalize the method we used in section 2.2 to give logarithmic bounds for the vertex strength of graphs with bounded higher chromatic number, we expect that the same logarithmic bound holds for such graphs.

It is interesting to consider the time complexity of the sum coloring problem for other classes of graphs. Since the OCCP problem is NP-complete for permutation graphs [22], it doesn't seem to be hard to prove the NP-completeness of the sum coloring problem for permutation graphs. If this is the case, then some other interesting questions come to mind, such as: what is the time complexity of this problem restricted to the class of graphs that are both interval and cointerval? It is known that these graphs are the intersection of permutation and split graphs.

In appendix A we extend the notion of sum coloring to list sum coloring and give an algorithm to solve this problem for graphs with bounded tree-width. We give a direct application of this problem, so it might be interesting to study this problem on other restricted families of graphs.

We believe that there are classes of graphs for which the edge sum coloring problem is harder than vertex sum coloring. We tried to extend the results of sections 4.2 and 4.3 to graphs with bounded tree-width, using the dynamic programming method based on the tree decomposition of the graph. This method fails at a stage which needs to solve a 3-dimensional matching. It might be the case that this problem is NP-complete for partial k-trees, which would be a really interesting result.

Bibliography

[1] S. Arnborg and J. Lagergren "Easy problems for tree decomposable graphs", *J. of Algorithms*, **12**, 308-340 (1991).

[2] A. Bar-Noy, M. Bellare, M. M. Halldorsson, H. Shachnai, T. Tamir "On chromatic sums and distributed resource allocation", *Information and computing*, **140**, 183-202, (1998).

[3] A. Bar-Noy, G. Kortsarz "The minimum color sum of bipartite graphs", In URL: *http://www.eng.tau.ac.il/amotz/publications.html*, (1997).

[4] H. Bodlaender "Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees", *J. of algorithms*, **11**, 631-643 (1990).

[5] H. Bodlaender "A tourist guide through tree-width", *Acta cybernetica*, **1**, 1-23 (1993).

[6] H. Bodlaender " Tree-width: Algorithmic techniques and results", *Proceedings 22nd International Symposium on Mathematical Foundations of Computer Science, MFCS'97*, LNCS **1295**, 29-36 (1997).

[7] H.L.Bodlaender "A linear time algorithm for finding tree decompositions of small tree-width", *SIAM J. Comp*, **25**, 1305-1317, (1996).

[8] D.G. Corneil, Y. Perl, and L.K. Stewart "A linear recognition algorithm for cographs", *SIAM J. Comp.*, Vol. **14** No 4, 926-934 (1985).

[9] B.Courcelle "The monadic Second order Logic of graphs I: Recognizable sets of finite graphs", *Information and Computation*, **85**, 12-75 (1990).

[10] B.Courcelle "The Monadic Second order Logic of graphs III: Treewidth, forbidden minors and complexity issues", *Informatique Theorique* **26**, 257-286 (1992).

[11] P. Erdos, E. Kubika, A. Schwenk "Graphs that require many colors to achieve their chromatic sum", *Congressus Numerantium*, **71**, 17-28 (1990).

[12] S. Even and R.E. Tarjan "Network flow and testing graph connectivity" *SIAM J. Comp.*, **4**, 507-518 (1975).

[13] S. Foldes, P.L. Hammer "Split graphs", *8th South-Eastern Conf. on Combinatorics, Graph Theory and Computing, Congressus Numerantium*, **19**, 311-315 (1977).

[14] Z. Galil, S. Micali, H. Gabow "An $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs" *SIAM J. Comp.*, **15** 1, 120-130, (1986).

[15] M.R. Garey and D.S. Johnson , "Computers and Intractability", *Freeman, San Francisco* 1979.

[16] M. Grtschel, L. Lovasz, A. Schrijver "Polynomial algorithms for perfect graphs" *Topics on perfect graphs North-Holland Math. Stud.*, **88**, 325-356 (1984).

[17] H. Hajiabolhassan, M.L. Mehrabadi, R. Tusserkani "Minimal coloring and strength of graphs", *to appear in Disc. Math.*

[18] I. Holyer "The NP-completeness of edge coloring", *SIAM J. Comp.*, **10**, 718-720 (1981).

[19] B. Jamison, S. Olariu "P_4-reducible graphs - a class of uniquely tree-representable graphs", *Disc. Math.*, **51**, 35-39 (1984).

[20] B. Jamison, S. Olariu "A linear time recognition algorithm for P_4-reducible graphs", *Proc. 9th Conf. on Found. of Software Technol. and Theor. Comp. Sci*, LNCS **405**, 1-19 (1989).

[21] K. Jansen "The optimum cost chromatic partition problem", *Algorithms and complexity*, LNCS **1203**, 25-36 (1997).

[22] K. Jansen "Complexity results for the optimum cost chromatic partition problem", *Preprint.*

[23] K. Jansen "Approximation results for the optimum cost chromatic partition problem", *Automata, languages and programming*, LNCS **1256**. 727-737 (1997).

[24] K. Jansen "Approximation results for the optimum cost chromatic partition problem", *Preprint.*

[25] T.R Jensen and B. Toft "Graph coloring problem", *Wiely-Interscience series in discrete mathematics and optimization*, (1995).

[26] T. Jiang and D. West "Coloring of trees with minimum sum of colors", J. of graph theory, to appear.

[27] E. Kubika "The chromatic sum of a graph", *Ph.D dissertation, Western Michigan University*, (1989).

[28] K. Kubika, G. Kubika, and D. Kountanis "Approximation algorithms for the chromatic sum", In *Proc. of the First Great Lakes Computer Science Conf.*, LNCS **507**, 15-21, (1989).

[29] E. Kubika, A.J. Schwenk "An introduction to chromatic sums", *Proc. of the seventeenth Annual ACM Comp. Sci. Conf. ACM press*, 39-45, (1989).

[30] L.G. Kroon, A. Sen, H. Deng, A. Roy "The optimum cost chromatic partition problem for trees and interval graphs", *Workshop on graph theoretical concepts in computer science*, LNCS **1197** (1996).

[31] S. Nicolosco, M. Sarrafzadeh, X. Song "On the sum coloring problem on interval graphs", *Algorithmica*, **23**, 109-126 (1999).

[32] N. Robertson and P.D Seymour "Graph minors, I. Excluding a forest" *J. of combinatorial theory*, Series **B 35**, 39-61 (1983).

[33] N. Robertson and P.D Seymour "Graph minors, V. Excluding a planar graph" *J. of combinatorial theory*, Series **B 41**, 92-114 (1986).

[34] P.Scheffler "The graphs of tree-width k are exactly partial k-trees", manuscript (1986).

[35] A. Sen, H. Deng, and S. Guha "On a graph partition problem with an application to VLSI layout", *Information Processing Letters*, **43**, 87-94 (1992).

[36] K.J. Supowit "Finding a maximum planar subset of nets in a channel", *IEEE Trans. on Computer Aided Design*, CAD 6, **1**, 93-94 (1987).

[37] C. Thomassen, P. Erdos, Y. Alavi, P.J. Malde, A.J. Schwenk "Tight bounds on the chromatic sum of a connected graph", *J. of graph theory*, Vol **13**, No 3, 353-357 (1989).

[38] D. West "Introduction to Graph Theory", Prentice-Hall Inc. (1996).

[39] M. Yannakakis "Computing the minimum fill-in is NP-complete", *SIAM J. Alg. Disc. Meth.*, Vol **2**, No. 1, 77-79 (1981).

[40] M. Yannakakis "Node-deletion problems on bipartite graphs", *SIAM J. Comp.*, Vol **10**, No. 2, 310-327 (1981).

Appendix A

List sum coloring of partial k-trees

Consider the following scheduling problem: we have n machines, called $M_1, M_2, \ldots, M_n$, and some jobs such that each of them needs some process to be done by each of the machines. We may assume that the process of a job on any machine takes unit time. There are some constraints for each job, which is given as a conflict graph G of size n, in which the nodes represent the machines, and two nodes are connected if the job can not be executed on the corresponding machines simultaneously. Also we have a list for each machine indicating in which intervals of time we can use that machine for this particular job. Our goal is to find a schedule for each job in such a way that the sum of the job completion time is minimized.

It's not difficult to see that the above problem is equivalent to the following variation of the sum coloring problem, which is called *list sum coloring*: A graph G with a list of colors for each vertex $v \in G$ is given. We want to find a proper coloring of G such that the color of each vertex is one of the colors of its list and the total sum of the colors is minimized. If such a coloring does not exist then the total sum is defined to be infinity.

In the list sum coloring problem we are looking for a *list coloring* in which the total sum of the colors is minimized. The list coloring problem is a well known problem in which we want to find a proper coloring of graph G with a given list of colors for each vertex, such that the color of each vertex is one of the colors of its list.

We can easily reduce the vertex sum coloring problem to the list sum coloring problem, by letting each vertex list be $\{1, 2, \ldots, n\}$. So:

Theorem A.1 *If the vertex sum coloring problem is NP-complete for the class* Π *of graphs, then the list sum coloring problem is also NP-complete for the class* Π.

So list sum coloring is NP-complete for bipartite graphs, chordal graphs, and interval graphs. Also the list coloring problem can be reduced to the list sum coloring problem.

This follows from the fact that if we can find an optimum list sum coloring for graph G with the total sum less than infinity then there exists a list coloring of G with the given list of colors. Thus:

Theorem A.2 *If the list coloring problem is NP-complete for the class* Π *of graphs, then the list sum coloring problem, is also NP-complete for the class* Π *of graphs.*

So the list sum coloring problem is not easier than either the list coloring or the vertex sum coloring problems. But there are some classes of graphs for which the list sum coloring problem can be solved efficiently. In this appendix we present an algorithm to solve this problem for the class of graphs with bounded tree-width, i.e partial k-trees, in time $O(l^{k+1}|V|)$, where l is the size of the lists of the vertices, and $|V|$ is the number of vertices.

Recall the definition of a tree decomposition of a graph from section 1.3. A tree decomposition (X, T) of width k is called *smooth* if for all $i \in I$, $|X_i| = k+1$ and for all $ij \in F$, $|X_i \cap X_j| = k$. Any tree decomposition of a graph G can be transformed to a smooth tree decomposition of G with the same width, by applying the following operations as many times as possible [7]:

(i) If for $ij \in F$, $X_i \subseteq X_j$, then contract the edge ij in T and take as the new node $X_{j'} = X_j$.

(ii) If for $ij \in F$, $X_i \not\subseteq X_j$ and $|X_j| < k+1$, then choose a vertex $v \in X_i - X_j$ and add v to X_j.

(iii) If for $ij \in F$, $|X_i| = |X_j| = k+1$ and $|X_i - X_j| > 1$, then subdivide the edge ij in T, let i' be the new node, choose a vertex $v \in X_i - X_j$ and a vertex $w \in X_j - X_i$, and let $X_{i'} = X_i - v \cup w$.

The *contract* operation removes two adjacent vertices v and w and replaces them with one new vertex that is made adjacent to all vertices that were adjacent to v and w. Bodlaender [7] shows that for a constant k and for the graph G with tree-width at most k, we can find a tree decomposition of G in linear time. Using the following lemma, which is proved by him, it can be seen that we can find a smooth tree decomposition of such a graph in linear time:

Lemma A.3 *[7] If* (X, T) *is a smooth tree decomposition of* $G(V, E)$ *with tree-width* k*, then* $|I| = |V| - k$*.*

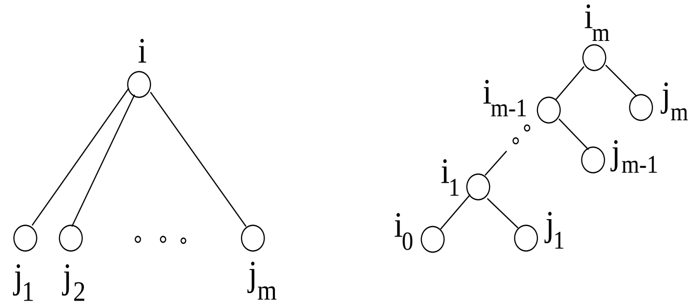

Figure A.1: Transforming a smooth tree decomposition to a smooth binary tree decomposition

We can transform any smooth tree decomposition to a smooth binary tree decomposition with the same width, such that each vertex i has either no child, or two children j_1 and j_2, where $X_{j_1} = X_i$, using the following transformation: for the vertex $i \in I$ with m children $j_1, j_2, \ldots, j_m$ replace the node i with $m+1$ copies of it, called $i_0, i_2, \ldots, i_m$, where i_x $(1 \leq x \leq m)$ has two children, the left one is i_{x-1} and the right one is j_x [6] (see figure A.1).

One can easily see that this transformation produces a smooth binary tree decomposition and we can transform any smooth tree decomposition to a smooth binary one in time linear in the size of the tree.

Let $G(V, E)$ be a graph of size n and tree-width of at most k. We denote the list assigned to the vertex v by $l(v)$. Suppose that the size of each list is bounded by l. We assume that a smooth binary tree decomposition of G, called (X, T), is given as well. For index $i \in I$, let G_i be the subgraph whose vertex set is the set of all vertices in a set X_j, with $j = i$ or j is a descendant of i in the rooted tree T.

We have a $|I| \times l^{k+1}$ table called Opt. Suppose the vertices of X_i are $v_{i_1}, v_{i_2}, \ldots, v_{i_{k+1}}$. The value of $Opt[i, c_1, c_2, \ldots, c_{k+1}]$, where $c_j \in l(v_{i_j})$ for $1 \leq j \leq k+1$, is the cost of an optimum list sum coloring of the graph G_i, such that the color of vertex v_{i_j} is c_j. The initial values of the entries of the table are all infinity. The table is filled in a bottom-up manner, i.e the algorithm starts by computing the values for the leaves of T, and always computes the value of the table for an internal node when it has computed the values of its child or its children. In the following lemma, we show how to compute the values of this table.

Lemma A.4 *Let i be a node of the tree T and assume that $X_i = v_{i_1}, v_{i_2}, \ldots, v_{i_{k+1}}$.*

(i) If i is a leaf then:

$$Opt[i, c_1, c_2, \ldots, c_{k+1}] = \sum_{j=1}^{k+1} c_j$$

where $c_x \neq c_y$ if $v_{i_x} v_{i_y} \in E$, $1 \leq x, y \leq k+1$.

(ii) If i is an internal node with children j_1 and j_2, where $X_i = X_{j_1}$ and $X_{j_2} = X_i \cup \{v'\} - \{v_{i_x}\}$, then:

$$\begin{aligned} Opt[i, c_1, c_2, \ldots, c_{k+1}] = \; & \min_{c_{v'}} \{Opt[j_2, c_1, \ldots, c_{x-1}, c_{v'}, c_{x+1} \ldots, c_{k+1}] + \\ & Opt[j_1, c_1, c_2, \ldots, c_{k+1}] - \textstyle\sum_{j \neq x} c_j\} \end{aligned}$$

where $c_{v'} \in l(v')$, and if $v_{i_x} v' \in E$ then $c_x \neq c_{v'}$.

Proof: The proof is almost straight forward. The only point is the subtraction in case (ii), which is the sum of the colors of the vertices of X_i which are computed in both X_{j_1} and in X_{j_2}, and is redundant. ■

The number of entries of the table for each node $i \in I$ is at most l^{k+1}, and to compute the value of each entry we compare at most l values, one for each color of the list of the vertex v'. Therefore the total number of calculations is in $O(l^{k+1}|I|)$. By lemma A.3 we know that $|I| \in O(|V|)$. Therefore the complexity of the algorithm is of $O(l^{k+1}|V|)$. Note that since k is a fixed constant, if the size of each of the lists is bounded, then the time complexity of the algorithm will be linear. We can conclude from the above arguments that:

Theorem A.5 *The list sum coloring problem is solvable in time $O(l^{k+1}|V|)$ for the class of graphs with tree-width at most k, where l is the maximum size of the lists of the vertices.*

Since all trees, series-parallel graphs, outerplanar graphs, almost k-trees, and Helin graphs have bounded tree-width, we can say:

Corollary A.6 *We can solve the list sum coloring problem on trees, series-parallel graphs, outerplanar graphs, almost k-trees and Helin graphs in polynomial time.*