Graph Coloring with Local and Global Constraints

by

Dániel Marx

PhD Thesis Summary

Under the supervision of
Dr. Katalin Friedl

Department of Computer Science and Information Theory
Budapest University of Technology and Economics

Budapest
2004
1 Introduction

Graph coloring and its variants are useful tools in modeling a wide variety of real-life problems. The basic graph coloring problem is to assign colors to the vertices of a graph in such a way that vertices connected by an edge receive different colors. We want to use as few colors on the vertices as possible. The minimum number of colors required is called the chromatic number of the graph. Another standard problem is edge coloring: here we have to assign colors to the edges of the graph such that two edges having the same end point cannot have the same color. The minimum number of colors required to color the edges is the chromatic index of the graph.

The study of graph coloring was motivated by the celebrated Four Color Conjecture. It was observed in the 19th century that every planar map can be colored with four colors such that countries sharing a boundary have different colors. Many people attempted to prove the conjecture, making graph coloring into a well-studied chapter of graph theory. The conjecture was open for more than a century: it was only in 1977 when Appel and Haken managed to resolve the problem with a tour de force proof requiring heavy use of computers.

Besides map coloring, graph coloring can be used to model several other problems. In the literature we can find applications such as optimizing the register allocations in a compiler [7], assigning frequencies in cellular networks [38], assigning wavelengths in WDM optical networks [14, 34], scheduling interfering jobs [3], assigning aircrafts to flights [6], and many more. The edge coloring version appears for example in timetable design [15] and in the scheduling of biprocessor tasks [31].

However, in real life the problems do not appear in such pure forms that can be directly modeled by graph coloring. There are additional constraints that have to be satisfied, for example, certain vertices cannot receive certain colors. New variants of graph coloring were introduced that allow us to take into account such constraints. In the precoloring extension problem some of the vertices have a preassigned color and we have to extend this precoloring to a proper coloring of the whole graph using the given number of colors. In list coloring each vertex has a list of admissible colors, and our task is to find a coloring where each vertex receives a color from its list. We can also consider the edge coloring versions of precoloring extension and list coloring.

Another variant of coloring is minimum sum coloring. Assume for convenience that the colors are the positive integers. In ordinary vertex coloring our goal is to minimize the number of colors used, that is, we want that the largest color assigned to the vertices is as small as possible. In minimum sum coloring we want to minimize the sum of the colors assigned to the vertices. These two goals lead to two very different problems, an optimum coloring for one goal is not necessary optimal with respect to the other goal. Minimum sum coloring was introduced independently by Supowit [41] (to model a VLSI design problem) and by Kubicka [32] (out of sheer theoretical curiosity). Furthermore,
it turned out that minimum sum coloring appears naturally if we want to minimize the average completion time of interfering jobs [3]. This is a nice example showing that a theoretically interesting problem can appear in multiple applications.

Multicoloring is a natural generalization of coloring where we have to assign to each vertex not only a single color but a set of colors. Now the requirement is that sets of colors assigned to neighbors have to be disjoint. Multicoloring appears naturally in many applications. We consider here the most general setting where each vertex has a demand, which is an integer number saying how many colors are required by the vertex.

2 Research Objectives

The aim of this dissertation is to study these generalized coloring problems from the algorithmic and complexity-theoretic point of view. Intensive research on this subject was started only in the 90s, thus there are still many open questions. Here I resolve some of these questions by giving algorithms and by proving complexity results. I also study some new cases that have not been investigated in the literature.

Most variants of graph coloring are NP-hard, thus we cannot expect a polynomial-time algorithm that solves the problem in general. However, ordinary vertex coloring can be solved efficiently on many classes of graphs, for example on interval graphs, chordal graphs, and partial \( k \)-trees. It is a natural question to ask whether the generalized coloring problems remain easy for these graph classes. My aim was to find special cases that are polynomial-time solvable, or to prove that the problem is NP-hard for a given case. One way to cope with NP-hard optimization problems is to give an approximation algorithm. For some of the problems I investigated the possibility of finding approximation algorithms.

3 Research Methodology

Most of the problems studied in the dissertation were thoroughly studied before in the literature. The graph classes I consider are not chosen arbitrarily: they appeared already in similar contexts, and it follows naturally that they should be investigated for other problems as well.

The main goal is to give polynomial-time algorithms for certain special cases. In order to give such algorithms, we use whatever weapon seems necessary for the problem at hand: dynamic programming, matching theory, network flows, linear algebra, and structural observations. When we fail to give a polynomial-time algorithm for some problem, then we try to show that the problem is NP-hard. The standard way to do is to give a reduction from a known NP-hard problem. In a reduction it is very important to carefully select the problem we are reducing from, as it can make the reduction
much simpler. In some cases we can utilize some subtle connections between seemingly unrelated problems. For example, in Chapter 3.3 I present a reduction from a disjoint paths problem to a precoloring extension problem. In Chapter 4.1, I reduce a precoloring extension problem to a minimum sum coloring problem.

For a hard optimization problem it is standard to investigate whether we can give an algorithm that does not give an optimum solution but only an approximation of the optimum. Approximation algorithms are different from heuristic algorithms: approximation algorithms have a guarantee on the quality of the solution provided. For example, in a minimization problem we try to find a polynomial-time algorithm that always produces a solution with cost at most $\alpha$ times the optimum for some constant $\alpha$. I investigated the approximability of the coloring problems with particular emphasis on whether it is possible to give a polynomial time approximation scheme (PTAS). I also used the theory of APX-hardness to argue that certain problems are unlikely to have a PTAS.

I successfully demonstrated that computers can be used to resolve certain combinatorial questions. For the reduction in Claim 3.3 a graph with certain properties was needed. I found a graph that satisfies these properties by writing a program that enumerates and checks every graph. This shows that in certain cases a reasonable amount of computerized search can settle conjectures.

4 New Results

The new scientific results of the dissertation are summarized in the following claims. The results are grouped according to the different coloring problems.

Claim 1: List coloring

In the list coloring problem each vertex $v$ has a list $L(v)$ of admissible colors, and our task is to find a coloring where each vertex receives a color from its list. The list edge coloring problem is defined analogously: each edge has a list of admissible colors $L(e)$.

Ordinary vertex coloring is a special case of list coloring: every color is allowed on every vertex, hence every vertex has the same list. Therefore if coloring is hard for some class of graphs, then list coloring is also hard for this class. Moreover, there are cases where coloring is easy, but list coloring is hard. For example, edge coloring is polynomial-time solvable for the edges of bipartite graphs, but it follows from [15] that list edge coloring is NP-hard for bipartite graphs. I strengthened this result as follows:

Claim 1.1 List edge coloring is NP-hard for planar 3-regular bipartite graphs (Theorem 2.1.2).
A graph is *outerplanar* if it can be drawn in the plane such that the edges do not cross each other and the vertices are on the outer face. A graph is *series-parallel* if it can be constructed from a single edge by repeatedly subdividing edges and adding parallel edges.

**Claim 1.2** List edge coloring is NP-hard for outerplanar graphs (Theorem 2.1.4) and for series-parallel graphs (Corollary 2.1.5).

The significance of this result comes from the fact that outerplanar graphs and series-parallel graphs have treewidth at most 2. For graphs with treewidth 1 (i.e., for forests) the problem is polynomial-time solvable by a method combining dynamic programming and bipartite matching. Usually when a problem can be solved in polynomial time for trees, then it is expected that the same method can be generalized for bounded treewidth graphs. My results shows that this is not case with list edge coloring. However, I show that list edge coloring is linear time solvable for bounded degree outerplanar and series-parallel graphs.

In the multicoloring version of list edge coloring each edge has a demand \( x(e) \). Now the goal is to select \( x(e) \) colors from \( L(e) \) in such way that the colors selected for edges sharing an end point are disjoint. That is, a color can be selected on at most one of the edges incident to the same vertex. Marcotte and Seymour [33] gave a characterization theorem for the list edge multicoloring of trees. Although they do not state it explicitly, by using standard techniques this theorem can be turned into a polynomial-time algorithm for the list edge multicoloring of trees. I present an algorithm that works for a slightly larger class of graphs.

**Claim 1.3** List edge multicoloring can be solved in \( O(C^2 n^2) \) time for graphs that do not have even cycles and have at most one odd cycle \((C\) is the number of colors, \(n\) is the number of vertices). Moreover, for every fixed \(k\) there is a randomized polynomial-time algorithm (with bounded error probability) that solves the problem for graphs having at most \(k\) cycles.

The algorithms in Claim 1.3 are based on the following idea. I introduce a new variant of list edge multicoloring where instead of exactly prescribing how many colors have to be assigned to each edge, we prescribe how many colors have to appear on the edges incident to each vertex \(v\). It turns out that this problem is polynomial-time solvable for every graph by a reduction to a matching problem. Moreover, using some simple linear algebra I show that in certain cases this new variant is the same as the original list edge multicoloring problem, hence we can solve the original problem by the same reduction to matching.

List edge multicoloring can be applied to model the scheduling of file transfers between processors. Let the vertices be the processors, if there is an edge \(e\) with demand
Table 1: Results for the list (multi)coloring problems

<table>
<thead>
<tr>
<th>Graph</th>
<th>Vertex coloring</th>
<th>Edge coloring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>list coloring</td>
<td>list coloring</td>
</tr>
<tr>
<td></td>
<td>list multicoloring</td>
<td>list multicoloring</td>
</tr>
<tr>
<td>Trees</td>
<td>Polynomial</td>
<td>NP-hard</td>
</tr>
<tr>
<td></td>
<td>[27]</td>
<td>(Theorem 2.2.2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partial $k$-trees</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td></td>
<td>[27]</td>
<td>(Theorem 2.2.2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cycles</td>
<td>Polynomial</td>
<td>(Randomized) Poly.</td>
</tr>
<tr>
<td></td>
<td>[27]</td>
<td>(Corollary 2.3.11)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bipartite graphs</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td></td>
<td>[27]</td>
<td>[27]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planar regular</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
<tr>
<td>bipartite graphs</td>
<td>[29]</td>
<td>[29]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$x(e)$ between two vertices, then this means that there is a direct connection between the two processors and a file has to be transfered on this connection. The file transfer represented by the edge $e$ requires $x(e)$ time slots. The colors correspond to the time slots, thus by assigning color sets to the edges we can schedule the file transfers. The requirement that every color appears at most once at a vertex ensures that a processor performs at most one transfer at the same time. (This requirement can be weakened: we can consider the variant of the problem where a processor can participate in at most $f$ file transfers simultaneously, the method works in that case as well.) Furthermore, by setting the lists of the edges appropriately, we can express additional constraints. For example, we can require that a transfer is not started before a certain time, or it is finished before a certain deadline. Claim 1.3 gives efficient algorithms for this problem if the topology of the network is almost a tree.

The vertex coloring version of list coloring can be solved in linear time for trees [27]. However, unlike in the edge coloring case, the list multicoloring problem is NP-hard for trees.

Claim 1.4 List multicoloring is NP-hard for binary trees (Theorem 2.2.1).

Table 1 summarizes the (old and new) results for list coloring.

Claim 2: Precoloring extension

In the precoloring extension problem a graph is given with some of the vertices having a preassigned color. This precoloring has to be extended to the whole graph using at most
$k$ colors. Biró, Hujter and Tuza [5, 24, 25] started a systematic survey of precoloring extension on different graph classes. In particular, they have shown in [5] that for interval graphs precoloring extension is NP-hard, but becomes polynomial-time solvable if every color appears at most once in the precoloring (this special case is abbreviated 1-PrExt). They raised two open questions on the possible generalizations of these results:

- Is it possible to solve 1-PrExt in polynomial time for the more general class of chordal graphs (a graph is chordal if it does not contain induced cycles of length greater than 3)?
- Does precoloring extension remain NP-hard for unit interval graphs?

It turns out that the answer is positive for both questions:

**Claim 2.1** Precoloring extension can be solved in $O(n^4)$ time for an $n$ vertex chordal graph if every color appears at most once in the precoloring (Theorem 3.1.5). This answers an open question of Hujter and Tuza [25].

**Claim 2.2** Precoloring extension is NP-hard for unit interval graphs (Theorem 3.3.1). This answers an open question of Hujter and Tuza [25].

In the algorithm of Theorem 3.1.5 a network is constructed and the precoloring extension problem is reduced to finding a maximum flow in this network. I define a new matroid that plays an important role in the analysis of the algorithm.

A possible application area for Claim 2.1 is the design and configuration of Wavelength Division Multiplexing (WDM) optical networks. WDM technology allows us to establish several data channels in a single optical fiber using the different wavelengths. Moreover, all-optical switches can route the different channels of an incoming fiber to different outgoing fibers. Therefore by configuring the switches appropriately, we can create direct optical connections between distant nodes of the network. However, in order to configure the network, we have to assign a wavelength to each connection such that connections that use the same fiber receive different wavelengths. This wavelength assignment problem is a coloring problem. The conflict graph of the network has one vertex for each connection, and two vertices are neighbors if the corresponding two connections share a fiber. If the number of wavelengths in a fiber is $k$, then the wavelength assignment problem can be solved if and only if the vertices of the conflict graph can be colored with $k$ colors.

The WDM network configuration problem was studied by several papers in the literature, with a particular emphasis on tree and tree-like networks [13, 12, 14, 11, 1]. However, the model becomes slightly different if we consider another type of optical switches. The simplest optical switch is the passive star that selects one incoming fiber.
for each wavelength, and transmits the data on this wavelength to every outgoing fiber. If we have this type of switches, then connections having the same wavelength cannot go through the same switch. That is, the connections with the same wavelength have to be vertex disjoint, not only edge disjoint. We can take into account this restriction by modifying the definition of the conflict graph: connect those vertices where the corresponding connections go through the same switch. It is well-known [19] that the intersection graph of paths in a tree is always chordal, hence the conflict graph in this problem will be a chordal graph. Vertex coloring is polynomial-time solvable for chordal graphs, hence we can determine in polynomial time whether the wavelength assignment problem can be solved. Moreover, the algorithm in Claim 2.1 gives a method for solving the slightly more general problem where some of the connections already have a wavelength, but every wavelength is assigned to at most one connection. For example, this is the case if all the connections going through a particular switch are already assigned a wavelength, but we are free to assign any wavelength to the remaining connections.

In the proof of Theorem 3.3.1 (Claim 2.2) I reduce a disjoint paths problem to pre-coloring extension. The NP-hardness of this disjoint paths problem is proved separately in Section 3.2. This result on the disjoint paths problem is of independent interest, thus I present it in a separate claim:

Claim 2.3 The disjoint paths problem on rectangle grid graphs is NP-hard even if the input satisfies Eulerian condition: for every vertex $v$, the number of edges incident to $v$ plus the number of demands involving $v$ gives an even number (Corollary 3.2.6). An analogous result holds in the directed case (Corollary 3.2.4). These results settle opens question posed by Vygen [44].

The Eulerian condition seems to be a very technical and arbitrary condition. However, it plays a central role in the theory of disjoint paths. There are many nice theoretical results for the cases when the Eulerian condition and some other condition hold. For example, the theorem of Okamura and Seymour [39] gives a good characterization for the disjoint paths problem if the network is planar, the Eulerian condition holds, and the terminals of the demands are on the outer face. Corollary 3.2.6 says that we cannot strengthen the Okamura-Seymour Theorem by dropping the requirement that the terminals have to be on the outer face (unless $P = NP$). Therefore Claim 2.3 nicely complements the existing positive results, showing a limit to possible generalizations.

Fiala [16] and independently Easton and Parker [9] have shown that edge precoloring extension is NP-hard for bipartite graphs with maximum degree at most 3. I present here a stronger result:

Claim 2.4 Edge precoloring extension is NP-hard for planar 3-regular bipartite graphs (Theorem 3.4.1). This answers an open question of Fiala [16].
The reason for proving NP-hardness for this particular graph class was that it allows us to prove quite easily the NP-hardness of minimum sum edge coloring on planar graphs (Theorem 4.2.1). In a similar reduction, de Werra et. al [8] also used Theorem 3.4.1 to prove the NP-hardness of a different edge coloring problem.

As a slight strengthening of Claim 1.2, I extended the hardness result for list edge coloring to the more restricted problem of precoloring extension:

**Claim 2.5** Edge precoloring extension is NP-hard for outerplanar graphs (Theorem 3.4.2) and for series-parallel graphs (Corollary 3.4.3).

**Claim 3: Minimum sum coloring**

In the minimum sum coloring problem we have to assign colors (positive integers) to the vertices of the graph, and the goal is to minimize the sum of the colors assigned. The problem has applications in VLSI design [42] and scheduling [3].

Giaro and Kubale have shown that minimum sum edge coloring is NP-hard for bipartite graphs [18]. I proved the following stronger complexity results:

**Claim 3.1** Minimum sum edge coloring is NP-hard for planar bipartite graphs having maximum degree 3 (Theorem 4.2.1), for planar 3-regular (non-bipartite) graphs (Theorem 4.2.2), and for graphs with treewidth at most 2 (Theorem 4.3.6). Moreover, minimum sum edge coloring is APX-hard for (non-planar) bipartite graphs (Theorem 4.2.3).

Let us put these negative results in context. Theorem 4.2.1 and Theorem 4.2.2 say that we cannot expect a polynomial-time algorithm for the minimum sum edge coloring of planar graphs, even if we assume that the graph is bipartite, or the graph is regular. However, an easy argument shows that if the planar graph is both bipartite and regular, then minimum sum edge coloring is easy to solve. Moreover, minimum sum edge coloring on planar graphs can be well approximated: in [C8] I present a polynomial-time approximation scheme (PTAS) for the more general problem of minimum sum edge *multicoloring* on planar graphs. (A PTAS means that for every \( \epsilon > 0 \) there is a polynomial-time algorithm to find a solution with cost at most \((1 + \epsilon)\) times the optimum.) This result is not included in the dissertation, but in Chapter 5.4 a weaker result is presented: a PTAS for the minimum sum edge multicoloring of trees. By Theorem 4.2.1 and 4.2.2 we cannot expect that the PTAS for planar graphs can be improved to an exact polynomial-time algorithm. Furthermore, Theorem 4.2.3 shows that minimum sum edge coloring is APX-hard for bipartite graphs, thus it is unlikely that the PTAS for trees can be extended to bipartite graphs. In [C8] I present a PTAS also for graphs with bounded treewidth. By Theorem 4.3.6 it is unlikely that this PTAS could be improved to a polynomial-time exact algorithm. Therefore the negative results in Claim 3.1 nicely complement the positive results (see Table 2).
Table 2: Results for the minimum sum edge (multi)coloring problems

<table>
<thead>
<tr>
<th>Graph</th>
<th>Minimum sum edge coloring</th>
<th>Minimum sum edge multicoloring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algorithm</td>
<td>Hardness</td>
</tr>
<tr>
<td>Paths</td>
<td>Polynomial [18, 40]</td>
<td>Pseudopolynomial</td>
</tr>
<tr>
<td>Trees</td>
<td>Polynomial [18, 40]</td>
<td>PTAS [28]</td>
</tr>
<tr>
<td>Bipartite graphs</td>
<td>1,796-approx. [23]</td>
<td>APX-hard (Theorem 4.2.4)</td>
</tr>
<tr>
<td></td>
<td>PTAS [C8]</td>
<td>2-approx. [4]</td>
</tr>
<tr>
<td></td>
<td>PTAS [C8]</td>
<td>APX-hard (Theorem 4.2.4)</td>
</tr>
<tr>
<td>Planar graphs</td>
<td>PTAS [C8]</td>
<td>NP-hard (Theorem 4.2.1)</td>
</tr>
<tr>
<td>Partial k-trees</td>
<td>PTAS [C8]</td>
<td>NP-hard (Theorem 4.3.6)</td>
</tr>
</tbody>
</table>

Minimizing the number of colors used in the coloring and the sum of the colors assigned are two very different problems. There are graphs where we have to use many colors to minimize the sum of the colors. For example, every tree can be colored with two colors, but for every \( k \geq 2 \) there is such a tree where we need \( k \) colors if we want to minimize the sum of the colors assigned. Denote by \( s(G) \) the minimum number of colors that is required by a minimum sum coloring of \( G \) (this parameter is also called the chromatic strength of \( G \)). Similarly, the minimum number of colors required for a minimum sum edge coloring is denoted by \( s'(G) \).

Salavatipour [40] has shown that for every \( k \geq 3 \) it is NP-hard to decide whether \( s(G) \leq k \) holds for the given graph \( G \). The complexity of the problem for \( k = 2 \) remained an open question. I resolved the complexity of the case \( k = 2 \), and determined the exact complexity of the problem for \( k \geq 3 \):

**Claim 3.2** It is coNP-hard to decide whether \( s(G) \leq 2 \) holds (Theorem 4.4.3). This answers an open question from [40]. Moreover, for every \( k \geq 3 \), deciding whether \( s(G) \geq k \) holds is not only NP-hard but \( \Theta_p^P \)-complete (Corollary 4.4.13).

Mitchem et al. [36] and independently Hajiabolhassan et al. [20] have proved an analog of Vizing’s Theorem: they have shown that \( \Delta(G) \leq \chi'(G) \leq s'(G) \leq \Delta(G) + 1 \) for every simple graph \( G \) (\( \Delta(G) \) is the maximum degree, \( \chi'(G) \) is the chromatic index). Harary and Plaxtholt conjectured (see [45]) that the second inequality is in fact an equality: \( s'(G) = \chi'(G) \) holds for every simple graph \( G \). However, in [36, 20] it is shown that for \( k = 4 \) and for every \( k = 5, 7, 9, 11, 13, \ldots \) there is a graph \( G_k \) with \( \chi'(G_k) = k \) and
Claim 3.3 For every \( k \geq 3 \) there is a graph \( G_k \) with \( \chi'(G_k) = k \) and \( s'(G_k) = k + 1 \) (Proposition 4.4.6). For every \( k \geq 3 \) it is \( \Theta_2^p \)-complete to decide whether \( s'(G) \leq k \) holds (Corollary 4.4.12). Furthermore, it is \( \Theta_2^p \)-complete to decide whether \( s'(G) = \Delta(G) \) holds (Corollary 4.4.14) or to decide whether \( s'(G) = \chi'(G) \) holds (Corollary 4.4.15).

For \( k > 3 \) the graph \( G_k \) is based on a construction of Izbicki [26]. In order to find the graph \( G_3 \), I wrote a program that tries every graph up to a given size, and checks whether one of them have the required property. I used the program nauty of McKay [35] to efficiently generate all the non-isomorphic graphs.

Hajiabolhassan et al. [20] asks as an open question how the graphs with \( s'(G) = \Delta(G) \) can be characterized. In light of the fact that deciding this property is \( \Theta_2^p \)-complete, we cannot hope to have an efficient characterization. This explains why it was worth obtaining this completeness result. If we only know that the problem is NP-hard, then this does not rule out the possibility of having some structure theorem that gives an NP characterization. However, by showing that the problem is \( \Theta_2^p \)-complete, I have shown that graphs with \( s'(G) = \Delta(G) \) do not have a nontrivial characterization (modulo some complexity-theoretic assumptions).

Claim 4: Minimum sum multicoloring

Minimum sum multicoloring is a generalization of minimum sum coloring: in this problem the vertices receive color sets instead of single colors. Each vertex \( v \) has a demand \( x(v) \) that says how many colors are needed by the vertex. We have to assign color sets to the vertices such that neighbors receive disjoint sets of colors. The natural generalization of minimum sum coloring would be trying to minimize the total sum of all the colors assigned to the vertices. However, under the name minimum sum multicoloring a different problem is studied in the literature, which problem is motivated by applications in scheduling. The finish time of a vertex in a coloring is defined to be the highest color assigned to the vertex. The sum of the coloring is the sum of the finish times. In minimum sum multicoloring our goal is to minimize the sum of the coloring. This problem is related to the scheduling of interfering jobs if the goal is to minimize the average completion time [4, 3].

Minimum sum multicoloring was considered in several papers recently [4, 3, 22, 21, 17, 28]. In [22] a polynomial-time approximation scheme (PTAS) is given for trees. That is, the authors show that for every fixed \( \epsilon > 0 \) there is a polynomial-time algorithm that always produces a solution with cost at most \( (1 + \epsilon) \) times the optimum. Two open questions were raised in the paper:
• Is minimum sum multicoloring NP-hard for trees?

• If the problem is NP-hard for trees, then is it possible to solve the problem in polynomial for paths?

I answered the first question:

**Claim 4.1** Minimum sum multicoloring is NP-hard for binary trees, even if every demand is polynomially bounded (Theorem 5.2.5). This answers an open question from [22].

Kovács [28] gave a partial answer to the second question by giving a pseudopolynomial-time algorithm for the problem on paths. Pseudopolynomial means that the running time of the algorithm is a polynomial of the length of the path and the maximum demand size. Therefore the algorithm is polynomial only if the demands are polynomially bounded. I proved a result on the structure of optimum colorings for paths:

**Claim 4.2** Denote by $p$ the maximum demand size. If the graph is a path, then minimum sum multicoloring has an optimum solution where every color set is the union of $O(\log p)$ continuous intervals (Theorem 5.1.14). If the graph is bipartite (resp. perfect), then there is an optimum solution where every color set is the union of $n$ (resp. $n^2$) intervals (Corollary 5.1.18).

This result can be useful in designing algorithms, and might lead to a fully polynomial-time algorithm for the problem. Some of the approximation algorithms are based on the idea of reducing the search space by considering only solutions where the color sets have simple structure. Knowing that in certain cases simple color sets are sufficient to find the optimum can lead to improved algorithms.

The edge coloring version of minimum sum multicoloring is defined analogously: each edge has a demand $x(e)$, and the goal is to find a coloring that minimizes the sum of the finish times of the edges. Minimum sum edge coloring is the special case where every demand is 1, hence the complexity results in Claim 3.1 hold also for minimum sum edge multicoloring. Minimum sum edge coloring is polynomial-time solvable for trees [18, 40], but the algorithms do not generalize for the multicoloring case:

**Claim 4.3** Minimum sum edge multicoloring is NP-hard for trees even if every demand is 1 or 2 (Theorem 5.3.1).

I presented a polynomial-time approximation scheme for the minimum sum edge multicoloring of trees. The PTAS uses some of the ideas from the approximation schemes in [22, 21]. However, some significant new ideas are also required, since there are great differences between the vertex coloring and the edge coloring versions of the problem.
Claim 4.4 For every fixed $\epsilon > 0$, there is a linear-time algorithm that solves minimum sum edge multicoloring for trees, and produces an output with sum at most $1 + \epsilon$ times the optimum.

Let us recall the file transfer application presented after Claim 1.3. Minimum sum edge multicoloring can be used to model a similar problem. However, now every color can be used on every edge, hence in this case we cannot model requirements like a fixed deadline. Another difference is that here the question is not whether the transfers can be scheduled using the given set of time slots, but our goal is to minimize the average completion time. The PTAS gives an approximate result arbitrarily close to the optimum in polynomial time if the network is a tree. In [C8] I extended this result to partial $k$-trees, hence we have an approximation algorithm not only for trees but for “tree-like” graphs as well.

Claim 5: Clique coloring

In clique coloring we have to satisfy weaker requirements than in ordinary vertex coloring. Instead of requiring that the two end points of each edge have two different colors, we only require that every inclusionwise maximal (nonextendable) clique contains at least two different colors. It is possible that a graph is $k$-clique-colorable, but its chromatic number is greater than $k$. For example, a clique of size $n$ is 2-clique-colorable, but its chromatic number is $n$.

In Chapter 6 we prove complexity results for clique coloring and related problems. Clique coloring is harder than ordinary vertex coloring: it is coNP-complete even to check whether a 2-clique-coloring is valid [2]. The complexity of 2-clique-colorability is investigated in [30], where they show that it is NP-hard to decide whether a perfect graph is 2-clique-colorable. However, it is not clear whether this problem belongs to NP. A valid 2-clique-coloring is not a good certificate, since we cannot verify it in polynomial time: as mentioned above, it is coNP-complete to check whether a 2-clique-coloring is valid. I determined the exact complexity of the problem:

Claim 5.1 $k$-clique-coloring is $\Sigma_2^p$-complete for every $k \geq 2$ (Corollary 6.2.2).

A graph is $k$-clique-choosable if whenever a list of $k$ colors are assigned to each vertex (the lists of the different vertices do not have to be the same), then the graph has a clique coloring where the color of each vertex is taken from its list. This notion is an adaptation of choosability introduced for graphs independently by Erdős, Rubin, and Taylor [10] and by Vizing [43]. In [37] it is shown that every planar or projective planar graph is 4-clique-choosable. I determined the complexity of clique-choosability:

Claim 5.2 $k$-clique-choosability is $\Pi_3^p$-complete for every $k \geq 2$ (Corollary 6.3.3).
Clique coloring is not a hereditary property: a $k$-clique-colorable graph can contain an induced subgraph that is not $k$-clique-colorable. Therefore it is natural to investigate graphs that are hereditary $k$-clique-colorable: graphs where every induced subgraph is $k$-clique-colorable.

Claim 5.3 Recognizing hereditary $k$-clique-colorable graphs is $\Pi_2^P$-complete for every $k \geq 3$ (Corollary 6.4.6).

5 Application of the Results

In the dissertation I present numerous examples from the literature where a variant of graph coloring is used to model some real-life problem. The applications are from such diverse areas as scheduling, frequency assignment, VLSI design, and compiler optimization. The algorithms developed in the dissertation can be used for these problems, either directly or as part of some more complex heuristic method. The results are presented as abstract graph coloring algorithms and not as a method for a particular application. Since a coloring problem can be useful in more than one application, the coloring algorithms can be used in more than one setting, including in situations not yet considered.

By proving complexity results, I show that in certain cases we cannot expect to find polynomial-time algorithms. If a problem is NP-hard, then we should concentrate on finding a good approximate algorithm or trying some other approach. Thus the complexity results give us useful guidance in finding promising research areas.

I presented an application where WDM network configuration can be modeled with graph coloring. I investigated a version of the problem that is polynomial-time solvable for tree networks. My algorithm in Claim 2.1 can solve a somewhat more general problem, where the wavelengths are already assigned to connections going through a distinguished switch.

An application of edge coloring is to model the scheduling of file transfers or some other tasks involving two dedicated processors. In Claim 1.3 I consider the version of the problem when there are additional constraints prescribing in which time slots the transfers can be performed. I gave efficient algorithms for the case when the network is almost a tree, previous algorithms worked only for acyclic networks. In Claim 4.4 a similar problem is considered, but here we want to minimize the average completion time of the file transfers. It turns out that the problem is NP-hard on trees, but has a polynomial-time approximation scheme. Previously only constant factor approximation algorithms were known for minimum sum edge multicoloring, this is the first time that a PTAS is found for some graph class.
6 Acknowledgments

I’m grateful to my supervisor Katalin Friedl for her kind guidance throughout the years. She gave tons of advice on my manuscripts, teaching me how to write papers. Judit Csima and Lajos Rónyai provided valuable comments on some of my papers.

The most important years of my mathematical education were my high school years in Szent István Gimnázium, Budapest, where László Lackó was my teacher. He taught me what mathematics is, and his classes had a lasting effect on my career.

I would like to thank my colleagues at the Department of Computer Science and Information Theory of the Budapest University of Technology and Economics, and especially András Recski for providing a pleasant working environment. The financial support of the OTKA grants 30122, 44733, 42559, and 42706 of the Hungarian National Science Fund are gratefully acknowledged.

Last but not least I would like to thank the love and support of my family, without which this project would not have been possible.

References


Publications

Journal papers


Conference papers


Minimum sum multicoloring on the edges of planar graphs and partial $k$-trees. 2nd Workshop on Approximation and Online Algorithms (WAOA), 2004 (Bergen), To appear.

Manuscripts

Precoloring extension on unit interval graphs, submitted.
The complexity of chromatic strength and chromatic edge strength, submitted.
Complexity results for minimum sum edge coloring, manuscript.
Precoloring extension on chordal graphs, manuscript.
Complexity of clique coloring and related problems, manuscript.

Citations

NP-completeness of list coloring and precoloring extension on the edges of planar graphs, Journal of Graph Theory, To appear.


We shall reduce PrExt edge coloring in bipartite cubic planar graphs to our problem. Given a bipartite cubic planar graph $BP$ and 3 pairwise disjoint matchings $E_i$, the question of PrExt edge coloring is to determine if it is possible to extend the edge precoloring $E_1$, $E_2$, $E_3$ to a proper 3-edge coloring of $G$. Very recently, this problem has been shown NP-complete in Marx [15]


Marx [M02] has recently shown that pSMC is NP-hard on trees, answering a question posed in an earlier version of this paper [HKP+99]. His result holds for even binary trees when the weight are polynomially bounded.


Marx proved the hardness of the pSMC problem on trees in [5]. He has shown that pSMC is NP-hard even on binary trees, even when p is polynomially bounded. Thus, the SMC problem on trees turned out to be one of the few scheduling-type of problems in which the preemptive version is essentially harder than the non-preemptive version.


Preemptive sum multicoloring has been shown to be strongly NP-hard for trees, even binary trees with polynomially bounded weights [38].

... There are hardness results specific to sum multicoloring; the case to date is a recent NP-hardness result of Marx [38] of pSMC on trees.


Beside the chromatic sum, various notions concerning coloring with costs have been motivated by scheduling problems. In some of them, given number of colors have to be assigned to the vertices. For results of this type, see, e.g., [Ma02].


Table 1, row “Line graphs of trees”, column “pSMC”: PTAS [39]


MaxEDP defined in rectilinear grids where any vertex can be a terminal is also NP-hard (see [3]).


Moreover, Marx shows in [12] that it remains NP-complete even in eulerian grids.

Precoloring extension on unit interval graphs, submitted.


On unit interval graphs, the unrestricted PrExt problem is known to be NP-complete [Mar02a].

Precoloring extension on chordal graphs, manuscript.


1-PrExt is polynomial-time solvable on chordal graphs ([Mar03]) but NP-complete on permutation graphs ([Ja97]).

Complexity of clique coloring and related problems, manuscript.


[Mar02b] For every $k \geq 2$, on the clique hypergraphs of graphs it is $\Sigma^P_2$-complete to test $k$-colorability.