NEW ALGORITHMIC AND HARDNESS RESULTS ON GRAPH PARTITIONING PROBLEMS

BY MARCIN JAKUB KAMIŃSKI

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Operations Research

Written under the direction of

Professor Vadim Lozin

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2007

ABSTRACT OF THE DISSERTATION

New algorithmic and hardness results on graph partitioning problems

by Marcin Jakub Kamiński

Dissertation Director: Professor Vadim Lozin

In this thesis we study algorithmic aspects of two graph partitioning problems – GRAPH COLORING and MAXIMUM CUT. This is a summary of main results of the thesis:

- A polynomial-time algorithm for k-VERTEX-COLORABILITY in the class of P_5-free graphs (for any fixed value k).
- A proof of NP-completeness of VERTEX- and EDGE-COLORABILITY in the class of graphs with girth at least g for any value of $g \geq 3$.
- A polynomial-time algorithm for 3-VERTEX-COLORABILITY of $(claw, hourglass)$-free graphs and an extension of that result to an infinitely increasing family of subclasses of *claw*-free graphs.
- An exact algorithm for MAX-CUT running in time $O^*(2^{(1-2/\Delta)n})$ in the class of graphs with maximum degree Δ.
- A proof of NP-completeness of MAX-CUT and MAX-BISECTION on unit disk graphs.

Table of Contents

Chapter 1

Introduction

The topic of this thesis is graph algorithms. In this introductory chapter we are going to define basic graph notions that are common for all the following chapters. We will also list algorithmic problems considered in the text and present a discussion of complexity results related to those problems. In the following sections we assume the knowledge of notions and facts presented here; more specific notions will be defined when needed.

1.1 Graphs

In this thesis we consider undirected, loopless graphs without multiple edges. With the exception of one chapter the graphs under consideration are also unweighted. For graph notions not defined here we refer the reader to [D05].

The *vertex set* of a graph $G = (V, E)$ is denoted by V and its *edge set* by E. Cardinalities of these sets are $n = |V|$ and $m = |E|$, respectively.

$G = (V, E, w)$ is an *edge-weighted graph*, if $G = (V, E)$ is a graph and $w : E \to \mathbb{R}^+ \cup \{0\}$ is a weight function which assigns to each edge ij of G a nonnegative number w_{ij}.

The *neighborhood* of a vertex v, denoted by $N(v)$, is the set of all vertices adjacent to v. The *closed neighborhood* of v, denoted by $N[v]$, is the set $N(v) \cup v$. The number of neighbors of a vertex v is called its *degree* and is denoted by $\deg(v)$. $\Delta = \Delta(G)$ is the maximum of a vertex in graph G.

The *average degree* of a graph is the sum of degrees of all vertices of the graph divided by the number of its vertices. The average degree is denoted by $d = d(G)$; notice that $d = 2m/n$.

If for some graph $\Delta = d$, then the graph is called *regular of degree d*. In particular, regular graphs of degree 3 are called *cubic*.

A subgraph of G is called *induced* by a set of vertices $U \subseteq V$ if it can be obtained from G by deleting the vertices of $V - U$. We denote such a subgraph by $G[U]$.

If $\mathcal{F}$ is a set of graphs then the family of $\mathcal{F}$-free graphs consists of all the graphs that do not contain any graph of $\mathcal{F}$ as an induced subgraphs. We say that graphs from $\mathcal{F}$ are *forbidden*. Such families of graphs are closed under vertex deletion.

The *chordless path* and *cycle* on n vertices are denoted by P_n, C_n, respectively. K_n is the *complete graph* on n vertices and $K_{a,b}$ is the *complete bipartite graph* with parts of size a and b.

A *line graph* of a graph $G = (V, E)$ is the graph $L(G) = (E, E')$ such that $ef \in E'$ if and only if the edges e and f are incident in G.

1.1.1 Graph coloring

A *k-coloring* of a graph is an assignment of numbers from set $\{1, \ldots, k\}$ (called *colors*) to the vertices of the graph in such a way that the endpoints of each edge receive different colors. A graph which admits a k-coloring is called *k-colorable*.

The set of vertices which have been assigned the same color is called a *color class*. Color classes are independent sets and the smallest number of independent sets in which the graph can be partitioned is called its *chromatic number*. A *chromatic coloring* of a graph is a coloring with the least number of colors (with the chromatic number of colors). A chromatic coloring is called *unique* if it is unique up to swapping color classes.

A k-coloring which satisfies an additional restriction – the color assigned to a vertex has to belong to the list of colors admissible for this vertex – is called a *k-list-coloring*.

Similarly, we define a *k-edge-coloring* to be an assignment of numbers from set $\{1, \ldots, k\}$ to the edges of a graph in such a way that two edges sharing an endpoint will receive different colors. Notice that a k-edge-coloring of a graph corresponds to a k-coloring of the vertices of its line graph. A graph which admits a k-edge-coloring is called *k-edge-colorable*. The well-known theorem by Vizing states that every graph is either Δ-edge-colorable or $(\Delta + 1)$-edge-colorable.

Notice that a graph admits a k-coloring (k-edge-coloring) if and only if each of its

connected components does. Also, the chromatic number of a graph is the maximum of the chromatic numbers of its connected components.

1.1.2 Cuts in graphs

A *cut* in a graph is a partition of its vertex set into two disjoint parts. For each cut, we define its *capacity* to be the number of the edges whose endpoints belong to two different parts of the cut. Any cut with maximum cardinality is called a *maximum cut*.

In edge-weighted graphs, the weight $w(C)$ of cut C is the sum of weights of all the edges that have their endpoints in two different parts of the cut. Notice that the unweighted graph can be thought of as a graph whose all edges have weight one.

We will denote the maximum size of a cut in a graph G by $mc(G)$. Notice that a maximum cut in a graph is a union of maximum cuts of its connected components.

A cut in which the number of vertices in two parts differ by at most one is called a *bisection*. A *maximum bisection* is a bisection of maximum capacity.

1.2 Algorithmic problems

It this thesis we consider graph algorithms with a focus on their time complexity. The number of vertices of the input graph, n, will be the measure of the input size of an algorithm. Polynomial/exponential-time/space will mean polynomial/exponential in n. Linear-time algorithms run in time linear in the number of edges of the input graph.

We also assume that the input graph is always connected, as if the graph is not, the partition problems we study in this thesis can be solved on each of the connected components separately and their solutions glued together. This pre- and post-processing can be done very efficiently (in linear time) and does not influence computational complexity of the algorithms.

The reference for all algorithmic terms is [GJ79].

1.2.1 Vertex coloring problems

Many different versions of vertex coloring problems have been introduced and we study some variants of the problem here. The basic coloring problem is VERTEX-COLORABILITY which asks for a coloring of a graph with the least number of color classes.

PROBLEM: VERTEX-COLORABILITY

INPUT: Graph $G = (V, E)$

OUTPUT: Chromatic coloring of G

A related problem is CHROMATIC-NUMBER which consists in computing the chromatic number of a graph, i.e. the number of classes in a chromatic coloring.

PROBLEM: CHROMATIC-NUMBER

INPUT: Graph $G = (V, E)$

OUTPUT: Chromatic number of G

These two are well-known NP-hard problems. We also study the k-VERTEX-COLORABILITY problem which consists in deciding whether the input graph can be colored with k colors, and if so, finding such coloring.

PROBLEM: k-VERTEX-COLORABILITY

INPUT: Graph $G = (V, E)$

OUTPUT: A k-coloring of G, if G is k-colorable; NO otherwise

This problem is known to be NP-hard for any $k \geq 3$.

All vertex colorability problems mentioned above are NP-hard in the class of all graphs. However we may hope that the problems will admit polynomial time solution

when the input graph is required to belong to a particular class of graphs.

An example would be the Δ-VERTEX-COLORING problem in the class of graphs with maximum degree Δ. This is a consequence of Brooks' Theorem ([B41]) that all graphs with maximum degree Δ, except for the clique on $\Delta + 1$ vertices, are Δ-colorable, if $\Delta \geq 3$. A linear-time algorithm was outlined in [L75] and described in detail in [BW01]. On the other hand, VERTEX-COLORABILITY is NP-hard in the class of graphs with maximum vertex degree Δ, if $\Delta \geq 4$.

Other classes of graphs which admit polynomial-time algorithms for VERTEX-COLORABILITY are perfect graphs ([GLS84]), planar graphs, locally connected graphs [K05], and some classes defined by forbidding induced subgraphs [R04, RS04, RS04a, RST02].

There is also a number of classes for which vertex colorability problem remain difficult. Examples are such classes are claw-free, line graphs ([H81]) and triangle-free graphs ([MP96]).

We also study the problem of deciding whether the input graph is k-list-colorable with respect to given color lists, and if so, finding a k-list-coloring.

PROBLEM: k-LIST-COLORING

INPUT: Graph $G = (V, E)$ and a list of admissible colors

$\mathcal{L}(v) \subseteq \{1, \ldots, k\}$ for each vertex $v \in V$

OUTPUT: A k-list-coloring of G with respect to color lists, if G is k-colorable;

NO otherwise

When all colors are available for each vertex, then the k-LIST-COLORING is equivalent with k-VERTEX-COLORABILITY and therefore k-LIST-COLORING is computationally more difficult than k-VERTEX-COLORABILITY.

1.2.2 Edge coloring problems

As we mentioned above a graph with maximum degree Δ is either Δ-edge-colorable or $(\Delta+1)$-edge-colorable. EDGE-COLORABILITY is the algorithmic problem of finding an edge coloring with the least number of colors.

PROBLEM: EDGE-COLORABILITY

INPUT: Graph $G = (V, E)$

OUTPUT: A Δ-edge-coloring of G, if G is Δ-edge-colorable;

$(\Delta+1)$-edge-coloring otherwise

The k-EDGE-COLORABILITY problem consists in determining whether the input graph is k-edge-colorable, and if so, finding a k-edge-coloring.

PROBLEM: k-EDGE-COLORABILITY

INPUT: Graph $G = (V, E)$

OUTPUT: A k-edge-coloring of G, if G is k-edge-colorable; NO otherwise

Holyer proved in [H81] that 3-EDGE-COLORABILITY is NP-hard for cubic graphs and hence also EDGE-COLORABILITY is NP-hard for general graphs. Notice that k-EDGE-COLORABILITY on graph G is equivalent to k-VERTEX-COLORABILITY on its line graph $L(G)$.

1.2.3 Cut problems

The MAX-CUT problem in an edge-weighted graph consists in finding a cut of maximum weight.

PROBLEM: MAX-CUT

INPUT: Graph $G = (V, E, w)$

OUTPUT: A maximum cut of G

If the graph is unweighted (which is algorithmically equivalent with an edge-weighted graph whose all edges have the same, positive weight), the corresponding problem is called SIMPLE-MAX-CUT. If there is no need for distinction or no risk of ambiguity, we will identify both problems and call it simply MAX-CUT.

The MAX-CUT problem is known to be NP-hard in the class of all graphs ([K72]) and it is NP-hard even if the input graph is restricted to be a split or 3-colorable graph ([BJ00]). The problem is also NP-hard in the class of graphs with maximum degree Δ, if $\Delta \geq 3$ ([Y78]).

On the other hand, MAX-CUT can be solved in polynomial time for planar graphs ([H75], [O72]) or graphs with bounded treewidth ([BJ00]).

The MAX-BISECTION problem is to find a maximum bisection of a graph.

PROBLEM: MAX-CUT

INPUT: Graph $G = (V, E, w)$

OUTPUT: A maximum bisection of G

The MAX-BISECTION problem is NP-hard for general graphs ([GJ79]). However, contrary to the MAX-CUT problem, MAX-BISECTION remains NP-hard on planar graphs (result of Jerrum presented in [JKLS05]).

Chapter 2

Coloring vertices of P_5-free graphs

In this chapter we study the problem of coloring vertices of P_5-free graphs. Building on previous work, we develop a polynomial-time algorithm for solving the k-LIST-COLORING problem in the class of P_5-free graphs. The result presented in this section is based on the paper [KL06]. (The most recent version of this paper is [HKLSS06].)

2.1 Introduction

An interesting problem which has been studied in both the graph-theoretical and algorithmic setting is the colorability problem in graphs without long induced paths. Here we are concerned with computational complexity issues.

Given the class of graphs containing all graphs without induced subgraphs isomorphic to the induced path on t vertices (we denote such a path by P_t and call such graphs P_t-free), we want to investigate whether the k-COLORABILITY problem can be solved in this class in polynomial time or can be proved to be NP-hard. A number of results have been obtained in this area for different combinations of parameters k and t. (We give an account of past research below.)

It has been known that the k-COLORABILITY problem can be solved in the class of P_4-free graphs in polynomial time. However, a polynomial-time algorithm for 3-COLORABILITY was the only complete result for the class of P_5-free graphs. Some partial information was obtained for the 4-COLORABILITY in this class, when additional structural restrictions are imposed on graphs. In this section we give a complete solution to the k-COLORABILITY problem in P_5-free graphs for an arbitrary value of k. In fact, our algorithm solves a more general version of the problem, known as LIST COLORING.

2.2 Previous results

The class of P_4-free graphs has been studied extensively and there exist linear-time algorithms for most of algorithmic problems, also for k-COLORABILITY ([CPS84]). A natural question is to investigate whether the result can be extended for the class of graphs containing no induced P_5. A polynomial-time algorithm solving the 3-COLORABILITY problem in that class was proposed in [RST02]. In [RS04a], the authors show that 3-COLORABILITY can in fact be solved in polynomial-time for the class of P_6-free graphs.

Later, the authors of [SW01] presented few algorithmic results concerning coloring of graphs without long paths. They seem to be the first to consider the problem in terms of two parameters – the number of colors k and the number of vertices t of the forbidden path P_t. In their paper they showed, using a different approach than in [RST02], how to 3-color vertices of a P_5-free graph in polynomial time. They also obtained two hardness results showing that 5-COLORABILITY is NP-hard for P_8-free graphs and 4-COLORABILITY is NP-hard for P_{12}-free graphs. The last result was recently improved in [LRS06]; the authors of the note claim that modifying the reduction from [SW01] 4-COLORABILITY can be shown to be NP-hard for P_9-free graphs.

Table 2.1 summarizes previously known results for the k-COLORABILITY problem in the class of P_t-free graphs. (A similar table appears for the first time in [SW01] and is being updated and redrawn ever since in all publications contributing to the area.) When looking at the previous results, two possible research directions seem promising and they were listed as open problems in [RS04]. We restate them here.

Problem 2.1. *Is there a polynomial-time algorithm for the* 4-COLORABILITY *problem in the class of P_5-free graphs?*

Problem 2.2. *Is there a polynomial-time algorithm for the* 3-COLORABILITY *problem in the class of P_7-free graphs?*

As far as we know there has been no progress on the second problem, while for the first some results were obtained. The authors of [LRS06] showed that the 4-COLORABILITY problem can be solved in polynomial time in the class of (P_5, C_5)-free

	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$	...
$k=3$	P	P	P	?	?	?	?	?
$k=4$	P	?	?	?	?	NPc	NPc	NPc
$k=5$	P	?	?	?	NPc	NPc	NPc	NPc
$k=6$	P	?	?	?	NPc	NPc	NPc	NPc
$k=7$	P	?	?	?	NPc	NPc	NPc	NPc
...	P	?	?	?	NPc	NPc	NPc	NPc

Table 2.1: Previously known complexity results for k-colorability of P_t-free graphs.

graphs (where C_5 is the chordless cycle on 5 vertices). Another result was obtained in [HSW06]. In that paper a class of P_5-free graphs that contain a dominating clique on four vertices is studied. (This assumption may seem artificial but follows from the structural properties of P_5-free graphs.) The authors provide a polynomial-time algorithm solving the 4-COLORABILITY problem in that class.

We also want to mention that the CHROMATIC-NUMBER problem is NP-hard in the class of P_5-free graphs ([KKTW01]).

2.3 Background

2.3.1 Coloring vertices

In this section we consider instances of the k-LIST-COLORING problem. An instance $G = (V, E, \mathcal{L})$ consists of the underlying graph $G = (V, E)$ together with a function $\mathcal{L} : V \to 2^{\{1,\ldots,k\}}$ which assigns to a vertex a list of its admissible colors. Recall that we say that G is k-list-colorable if exists a coloring $c : V \to \{1, \ldots, k\}$ such that for each $v \in V$, $c(v) \in \mathcal{L}(v)$.

For a set $W \subseteq V$, we write $\mathcal{L}(W) = \bigcup_{w \in W} \mathcal{L}(w)$. We say that $\mathcal{L}(v)$ (or $\mathcal{L}(W)$) is the *palette* of v (or W). When we want to emphasize the underlying instance H, we write $\mathcal{L}_H$; if the subscript is omitted, we always refer to instance G.

If a vertex is assigned a color, we can exclude that color form the palettes of all its neighbors. We say that an instance $G = (V, E, \mathcal{L}, D)$ is in *simplified form* if there are no adjacent vertices $v, w \in V$ such that $|\mathcal{L}(v)| = 1$ and $\mathcal{L}(v) \subseteq \mathcal{L}(w)$. In this paper

we assume that all instances are in simplified form and that instance simplification in algorithms is done implicity. (It can be easily performed in time linear in the number of edges.)

An instance G is said to be *compatible* with a set $\mathcal{G}$ of instances if G is k-list-colorable if and only if at least one of the instances in $\mathcal{G}$ is k-list-colorable. Notice that if G is compatible with $\mathcal{G}$ and some $H \in \mathcal{G}$ is compatible with $\mathcal{H}$, then G is compatible with $(\mathcal{G} - H) \cup \mathcal{H}$.

2.3.2 Dominating structure

We say that a subset $W \subseteq V$ of the vertex set is a *dominating set* if every vertex in V either belongs to W or has a neighbor in W.

Our algorithm is based on an interesting structural property of P_5-free graphs that has been described by Bascó and Tuza in [BT90a]. (The properties of graphs without long induced paths have been also studied in other papers, see [BT90, BT93, BT02, CK90].) Following their terminology, we say that H is *dominating* in G if G contains a dominating set isomorphic to H. In particular, a dominating clique in G is a dominating set in G which induces a complete graph. Similarly, a dominating P_3 is a dominating set in G which induces a path on 3 vertices.

Theorem 2.3 (Theorem 8 in [BT90a])**.** *In every P_5-free connected graph there is a dominating clique or a dominating P_3.*

We will refer to a dominating set which induces a graph isomorphic to a clique or P_3 as a *dominating structure*. Below we assume that the dominating structure in the input graph G is given and we denote the set by D. Notice that if G is k-list-colorable (for $k \geq 3$), then $|D| \leq k$, and therefore a dominating structure in a P_5-free graph can be found in polynomial time. An efficient algorithm for finding a dominating structure in P_5-free graphs – not necessarily k-list-colorable – was presented in [CK90].

To give an application of Theorem 2.3, let us consider the 3-LIST-COLORING problem in the class of P_5-free graphs. Notice that once a 3-coloring of vertices in D is fixed, then $|\mathcal{L}(v)| \leq 2$ for all $v \in V$. The question whether the coloring of D can be extended

for the whole graph can be stated as a 2-SAT instance and solved in polynomial time. Hence, considering all possible 3-colorings of D and checking extendability of each, we can obtain a polynomial algorithm for the 3-LIST-COLORING problem in the class of P_5 free graphs. (See [RST02] for more details.)

We can partition all vertices in $V - D$ into disjoint classes depending on their neighborhood in the dominating structure D. For $I \subseteq D$, $U_I(G) = \{v \in V - D : N(v) \cap D = I\}$. If G is the underlying instance, we just write U_I. These sets will be referred to as *bags*.

While looking for a coloring in a graph, two adjacent vertices with disjoint palettes can be as well though of as non-adjacent. Essential are only such neighbors that do not have disjoint palettes.

For a vertex $v \in V$, we define the set of its *essential neighbors* $\mathcal{E}(v) = \{w \in N(v) : \mathcal{L}(v) \cap \mathcal{L}(w) \neq \emptyset\}$. The remaining neighbors $N(v) - \mathcal{E}(v)$ are called *non-essential*. Note that the relation of being an essential (non-essential) neighbor is symmetric. Also, assigning a color to a vertex does not change possible color choices for its non-essential neighbors.

Similarly, for a set $W \subseteq V$, we define the set of its essential neighbors $\mathcal{E}(W) = \bigcup_{v \in W} \mathcal{E}(v)$. Also, for sets $U, W \subseteq V$, we say that $\mathcal{E}^U(W) = \mathcal{E}(W) \cap U$ is the *essential part of W with respect to U*. A set $W \subseteq V$ is called *separated* if all essential neighbors of all its vertices belong to W, i.e. $\mathcal{E}(W) \subseteq W$.

Possibly only some vertices from bag U_I have essential neighbors in U_J. For $I, J \subseteq D$ and $I \neq J$, $U_I^J(G) = \mathcal{E}^J(I)$. If G is the underlying instance we simply write U_I^J. Notice that $\mathcal{E}^{U_J}(U_I^J) = U_J^I$.

2.3.3 Structure of the algorithm

The nature of our solution is inductive. Designing the algorithm solving the k-LIST-COLORING problem, we assume there exist polynomial-time algorithms for the p-LIST-COLORING problem for any $p < k$. The problem can be easily solved for $k = 1, 2$ and with a bit more effort for $k = 3$ so below we assume that k is at least 4. Notice that our inductive assumption allows us to find a chromatic coloring of a p-list-colorable

P_5-graph with $p < k$ in polynomial time.

The main idea of our k-coloring algorithm is to use the structural property of P_5-free graphs to create a set of instances compatible with the input instance. These instances are simpler and solving them essentially amounts to finding a $(k-1)$-coloring. For clarity we divided the description of our solution into three parts, each corresponding to one of the following sections.

2.4 Dominating an independent set

In this section we fix the underlying graph (V, E), the dominating set D, and two independent sets S, T belonging to two different bags U_I, U_J. When we speak about the essential part of S or T, we mean it with respect to the other of these two sets. These essential parts of S and T will be denoted by S' and T', respectively. We indicate the instance (if it is different than G) by placing it in the subscript, for example S'_H. Note that S' is empty if and only if T' is empty.

Our first goal is to develop a procedure that given an instance G creates a set of instances $\mathcal{G}$ compatible with G. We also require that for each instance $G_t \in \mathcal{G}$ either S'_{G_t} is empty or the palette of T'_{G_t} has fewer colors than the palette of T'.

Lemma 2.4. *Let $S \subseteq U_I^J$ and $T \subseteq U_J^I$ be independent sets. If $S' \neq \emptyset$, there exists a vertex in S' that is adjacent to all vertices in T'.*

Proof. Let s_1 a vertex from S which has the largest neighborhood in T. If $S' \neq \emptyset$, then $s_1 \in S'$. For contradiction assume that there exists a vertex $t_2 \in T'$ that is not adjacent to s_1. Hence, there must exist a vertex $s_2 \in S$ (different than s_1) adjacent to t_2. Notice that s_2 is not adjacent to some $t_1 \in N(s_1) \cap T$ (by the choice of s_1). Since $I \neq J$, there exists a vertex $v \in (I - J) \cup (J - I)$ but then $G[v, s_1, s_2, t_1, t_2]$ is an induced P_5; a contradiction. □

Notice that from Lemma 2.4 follows that the vertices of S' can be ordered quasi-linearly with respect to the neighborhood containment.

Let $v \in S'$ be a vertex that dominates T' and let us look at the palette of v. We can divide it into two parts – the colors that belong to the palette of T' and the remaining

ones. Notice that assigning to v one of the colors from the palette of T' decreases the size of the palette of T' in the resulting instance. On the other hand, truncating the palette of v so that it contains only the colors not belonging to the palette of T', decreases the size of S' in the resulting instance. The following procedure makes use of this observation.

PROCEDURE $\Pi_{S,T}$

INPUT: Instance $G = (V, E, \mathcal{L}, D)$.

OUTPUT: Set $\mathcal{G}$ of instances compatible with G such that for each $G_t \in \mathcal{G}$, $S'_{G_t} = \emptyset$ or $|\mathcal{L}_{G_t}(T'_{G_t})| < |\mathcal{L}(T')|$.

STEP 1. Let $\mathcal{G} = \emptyset$. If $S' = \emptyset$, then RETURN $\{G\}$.

STEP 2. Find a vertex $v \in S'$ that dominates T'. For every $d \in \mathcal{L}_G(v) \cap \mathcal{L}_G(T')$, ADD TO $\mathcal{G}$ an instance $G' = (V, E, \mathcal{L}_{G'}, D)$ such that $\mathcal{L}_{G'}(v) = d$ and $\mathcal{L}_{G'}(w) = \mathcal{L}_G(w)$ for all vertices $w \in V - \{v\}$.

STEP 3. If $\mathcal{L}_G(v) - \mathcal{L}_G(T') \neq \emptyset$, create an instance $G' = (V, E, \mathcal{L}_{G'}, D)$ such that $\mathcal{L}_{G'}(v) = \mathcal{L}_G(v) - \mathcal{L}_G(T')$ and $\mathcal{L}_{G'}(w) = \mathcal{L}_G(w)$ for all vertices $w \in V - \{v\}$. ADD TO $\mathcal{G}$ the instances returned by $\Pi(G')$.

STEP 4. RETURN $\mathcal{G}$.

Claim 2.5. PROCEDURE $\Pi_{S,T}$ *is correct and runs in polynomial time.*

Proof. Let G be the input instance and $\mathcal{G}$ the output set of instances. To prove that the procedure is correct, we will proceed by induction on $|S'|$. We will prove that $\mathcal{G}$ is (*) compatible with G and (**) for each $G_t \in \mathcal{G}$, $S'_{G_t} = \emptyset$ or $|\mathcal{L}_{G_t}(T'_{G_t})| < |\mathcal{L}(T')|$.

If $|S'| = 0$, then $T' = \emptyset$ and $\mathcal{G}$ consists only of G (STEP 1). Clearly, G is compatible with $\{G\}$ and (**) is also satisfied.

Suppose that $|S'| = i$ and for all instances H with $|S'_H| < i$ the output of the procedure satisfies both conditions (*) and (**). First, let us notice that if one of the instances in $\mathcal{G}$ is k-list-colorable, then so is G because for each instance $G_t \in \mathcal{G}$ and each vertex $v \in V$, $\mathcal{L}_{G_t}(v) \subseteq \mathcal{L}(v)$.

Now suppose that G is k-list-colorable. Since in any k-coloring of G, v receives a color from $\mathcal{L}(v)$, then either one of instances created in STEP 2 or G' is k-list-colorable. If none of the instances created in STEP 2 is k-list-colorable, then G' must be k-list-colorable and – by the induction hypothesis – at least one of the instances created in STEP 3 is k-list-colorable. Hence, (*) is satisfied.

It is easy to see that all instances G_t created in STEP 2 have $|\mathcal{L}_{G_t}(T'_{G_t})| < |\mathcal{L}(T')|$. The set of instances created in STEP 3 comes from a call of the procedure for G' and for these instances the condition is satisfied by the induction hypothesis, since $|S'_{G'}| < |S'|$. Hence, (**) is satisfied.

Now let us show the running time of the procedure. Clearly, identifying sets S', T', finding a dominating vertex v (STEP 2) and creating new instances can be done in polynomial time. Notice that the recursive call in STEP 3 is done for an instance with a smaller essential part of S so the depth of the recursion is at most n. Hence, the running time follows. □

Now we are going to use the procedure to design an algorithm that given G creates a set of instances $\mathcal{G}$ compatible with G. We also want $\mathcal{G}$ to have a polynomial size and we require that for all $G_t \in \mathcal{G}$, the essential part of S in G_t is empty.

Lemma 2.6. *There exists a polynomial-time* ALGORITHM $\Pi'_{S,T}(G)$ *such that given two independent sets $S \subseteq U_I^J$ and $T \subseteq U_J^I$ generates a set of instances $\mathcal{G}$ compatible with G, such that for each $G_t \in \mathcal{G}$, $S'_{G_t} = \emptyset$.*

Proof. Each call of PROCEDURE $\Pi_{S,T}$ produces a set compatible with G that has a polynomial number of members (in fact at most kn). All members have either $S'(G_t) = \emptyset$ or fewer colors in the palette of $T'(G_t)$ than in the palette of T'.

Calling PROCEDURE $\Pi_{S,T}$ recursively until all instances have the property $T'(G_t) = \emptyset$ builds a search tree of bounded depth (at most k) and polynomial degree (at most kn). Hence, the number of instances is polynomial and so is the running time of the algorithm. □

2.5 Dominating color classes

In this section, as in the previous one, we fix the underlying graph (V, E), the dominating set D, and two subsets $I, J \subset D$, $I \neq J$, which induce two bags, U_I, U_J. Procedure $\Theta_{I,J}$ presented in this section is parameterized by I and J.

Our goal is to design an algorithm that given an input graph G creates a set $\mathcal{G}$ of instances compatible with G. We also want $\mathcal{G}$ to have a polynomial number of members and that for each $G_t \in \mathcal{G}$, $U_I^J(G_t)$ is empty or the chromatic number of $U_I^J(G_t)$ is smaller than the chromatic number of U_I^J.

Claim 2.7. Procedure $\Theta_{I,J}$ *is correct and runs in polynomial time.*

Proof. Let G be the input instance and $\mathcal{G}$ the output set of instances. We will show that G is compatible with $\mathcal{G}$ and for each $G_t \in \mathcal{G}$, either $U_I^J(G_t) = \emptyset$ or $\chi(U_I^J(G_t)) < \chi(U_I^J)$.

First let us notice that the set $\mathcal{H}$ is obtained from the set $\mathcal{G}$ by replacing instances $G_t \in \mathcal{G}$ with a set of instances compatible with G_t. Hence, after Step 5, G is compatible with the set $\mathcal{H}$ if and only if G is compatible with the set $\mathcal{G}$. Since G is compatible with $\mathcal{G}$ before the loop (Step 2), it is also compatible after Step 8, and therefore G is compatible with the output set $\mathcal{G}$.

Notice that after i-th iteration of the loop (Steps 3 – 7), for all $G_t \in \mathcal{G}$ there are no vertices in A that have an essential neighbor in B_i. Therefore at Step 8, for all $G_t \in \mathcal{G}$, no vertex from A has an essential neighbor in U_J^I and clearly either $U_I^J(G_t) = \emptyset$ or $\chi(U_I^J(G_t)) < \chi(U_I^J)$.

Each call of Procedure Π'_{A,B_i} in Step 5 produces a polynomial number of instances with a smaller chromatic number. For each such an instance the procedure Π' is called recursively and since the depth of the recursion is bounded by k, the running time of

the algorithm is polynomial. □

PROCEDURE $\Theta_{I,J}$

INPUT: Instance $G = (V, E, \mathcal{L}, D)$.

OUTPUT: Set $\mathcal{G}$ of instances compatible with G such that for each $G_t \in \mathcal{G}$, $U_I^J(G_t) = \emptyset$ or $\chi(U_I^J(G_t)) < \chi(U_I^J)$.

STEP 1. If $U_I^J = \emptyset$, RETURN $\mathcal{G}$.

STEP 2. Find a chromatic coloring of $G[U_I^J]$ and let A be one of the color classes (non-empty). $(k-1)$–color $G[U_J^I]$ and let $B_1, \ldots, B_{k-1}$ be the color classes of that coloring. Let $\mathcal{G} := \{G\}$ and $\mathcal{H} := \emptyset$.

STEP 3. FOR EACH $i = 1, \ldots, k$ DO

STEP 4. IF $B_i \neq \emptyset$ THEN

STEP 5. FOR EACH $G_t \in \mathcal{G}$,
ADD TO $\mathcal{H}$ the instances returned by $\Pi'_{A,B_i}(G_t)$.

STEP 6. $\mathcal{G} := \mathcal{H}$, $\mathcal{H} := \emptyset$.

STEP 7. END FOR

STEP 8. RETURN $\mathcal{G}$.

Now we will use the procedure to design an algorithm that given G creates a set of instances $\mathcal{G}$ compatible with G. We also require that for each $G_t \in \mathcal{G}$, $U_I^J(G_t)$ is empty.

Lemma 2.8. *There exists a polynomial-time algorithm $\Theta'_{I,J}$ that given two sets $I, J \subset D$ $(I \neq J)$ generates a set of instances $\mathcal{G}$ compatible with G such that for each $G_t \in \mathcal{G}$, $U_I^J(G_t) = \emptyset$.*

Proof. Calling PROCEDURE $\Theta_{I,J}$ recursively until instances have the property $U_I^J(G_t) = \emptyset$ builds a search tree of bounded depth (at most k), since at each step the chromatic number of in $U_I^J(G_t)$ decreases, and a polynomial degree. Hence, the running time of the algorithm is polynomial.

2.6 Main algorithm

In this section we combine techniques described above to construct an algorithm that solves the k-LIST-COLORING problem in the class of P_5-free graphs. Let us notice that we can assume that the input graph is connected, as if it is not, the k-LIST-COLORING problem can be solved on its connected components separately.

We divide the presentation of the main algorithm into three steps. First, we make a simple observation that if in some instance G all bags are separated, than the k-LIST-COLORING problem can be solved easily on G.

Lemma 2.9. *Let $G = (V, E, \mathcal{L}, D)$ be an instance of the k-LIST-COLORING problem such that for each $I \subset D$, $\mathcal{E}(U_I) \subseteq U_I$ and $|\mathcal{L}(v)| = 1$ for each $v \in D$. The k-LIST-COLORING problem can be solved on G in polynomial time.*

Proof. Since vertices of U_I have no essential neighbors outside U_I and vertices of the dominating structure have been already colored, graphs $G[U_I]$ can be colored separately for each $I \subset D$ and solutions can be glued together. Notice that the coloring of each $G[U_I]$ is in fact the $(k-1)$-LIST-COLORING problem. By the inductive assumption this can be solved in polynomial time. □

Second, we want to design a polynomial-time procedure that given an instance G with a dominating set of bounded size creates a set of instances $\mathcal{G}$ compatible with G. We also require that all the bags of each graph in $\mathcal{G}$ are separated.

ALGORITHM Λ

INPUT: Instance $G = (V, E, \mathcal{L}, D)$ such that $|D| \leq k$.

OUTPUT: Set $\mathcal{G}$ of instances compatible with G such that for each $G_t \in \mathcal{G}$ and each $I \subset D$, $\mathcal{E}(U_I(G_t)) \subseteq U_I(G_t)$.

STEP 1. FOR EACH k-coloring of D DO

STEP 2. FOR EACH $I, J \subset D$, $I \neq J$ DO

STEP 3. IF $U_I^J \neq \emptyset$ THEN

STEP 4. FOR EACH $G_t \in \mathcal{G}$,
ADD TO $\mathcal{H}$ the instances returned by $\Theta'_{I,J}(G_t)$;

STEP 5. $\mathcal{G} := \mathcal{H}$, $\mathcal{H} := \emptyset$;

STEP 6. END FOR

STEP 7. END FOR

STEP 8. RETURN $\mathcal{G}$

Lemma 2.10. ALGORITHM Λ *is correct and runs in polynomial time.*

Proof. First let us notice that since the size of D is bounded so is the number of k-colorings of D (STEP 1) and the number of pairs of subsets $I, J \subset D$ (STEP 2). Hence, STEP 3 will be performed at most a constant number of times and STEP 4 takes a polynomial time, so the polynomial running time of the whole algorithm follows.

From Lemma 2.8 it is clear that after STEP 4, G is compatible with $\mathcal{H}$ if and only if G is compatible with G. After STEP 5, all graphs G_t in $\mathcal{G}$ have $U_I^J(G_t) = 0$ for all pairs I, J that have been considered so far. Clearly, at STEP 8 all instances in $\mathcal{G}$ have $U_I^J(G_t) = 0$ for all pairs $I, J \subset D$, $I \neq J$ and, hence, $\mathcal{E}(U_I(G_t)) \subset U_I(G_t)$ for all $I \subset D$. □

Now we are ready to state our main result.

Theorem 2.11. *There exists a polynomial-time algorithm for the* k-LIST-COLORING *problem in the class of* P_5*-free graphs.*

	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$	...
$k=3$	P	P	P	?	?	?	?	?
$k=4$	P	P	?	?	?	NPc	NPc	NPc
$k=5$	P	P	?	?	NPc	NPc	NPc	NPc
$k=6$	P	P	?	?	NPc	NPc	NPc	NPc
$k=7$	P	P	?	?	NPc	NPc	NPc	NPc
...	P	P	?	?	NPc	NPc	NPc	NPc

Table 2.2: Currently known complexity results for k-colorability of P_t-free graphs.

Proof. A k-list-colorable graph cannot contain a clique on $k+1$ vertices as its subgraph. We assume that the input instance G does not contain such a subgraph. (This can be tested in polynomial time, and if G contains a clique on $k+1$ vertices, then the instance is not k-list-colorable.)

According to Theorem 2.3, a connected P_5-free graph contains either a dominating clique or a dominating P_3. Since the size of any clique in G is at most k, the size of the dominating structure is also bounded by k and a dominating set D can be found in polynomial time.

Algorithm Λ is called for such an instance G. It creates a set of instances $\mathcal{G}$ that is compatible with G. Moreover, all bags of every graph in $\mathcal{G}$ are separated and instances of this type can be handled by Lemma 2.9. □

2.7 Conclusion

In this section, we presented the first polynomial-time algorithm to solve the k-colorability problem in the class of P_5-free graphs. Table 2.2 presents the current landscape of complexity results on this problem. We purposely avoid computing the power of the polynomial in our solution since the computational complexity of the algorithm is highly exponential in k. Finding a better algorithm for this problem or possible showing that the problem is fixed parameter tractable is an interesting direction for future research.

Chapter 3

Coloring edges and vertices of graphs without short or long cycles

In this section we show that VERTEX-COLORABILITY and EDGE-COLORABILITY remain difficult even for graphs without short cycles, i.e., without cycles of length at most g for any particular value of g. On the contrary, for graphs without long cycles, both problems can be solved in polynomial time. The content of this section is based on the paper [KL07].

3.1 Introduction

VERTEX-COLORABILITY and EDGE-COLORABILITY are hard algorithmic problems. Showing that 3-EDGE-COLORABILITY is NP-hard for cubic graphs, Holyer proved in [H81] that EDGE-COLORABILITY is NP-hard. A closer look at the proof reveals that his construction is triangle-free which implies that the 3-EDGE-COLORABILITY (and therefore EDGE-COLORABILITY) remain NP-hard for graphs without cycles on three vertices. In this section, we strengthen that result and show that the 3-EDGE-COLORABILITY problem is NP-hard for graphs without cycles of length at most g, for any fixed value of g.

Similarly, Maffray and Preissmann showed in [MP96] that 3-VERTEX-COLORABILITY is NP-hard for triangle-free graphs. We extend their result showing that k-VERTEX-COLORABILITY is NP-hard ($k \geq 3$) on graphs without cycles of length at most g, for any fixed value of g.

Having proved the NP-hardness of VERTEX-COLORABILITY and EDGE-COLORABILITY on graphs without short cycles, we study graphs without long cycles and show that both

problems have polynomial-time solutions for these graphs.

3.2 Graphs without short cycles

The minimum length of a cycle in a graph G is called the *girth* of G. Graphs of large girth have been a subject of intensive investigations with respect to various problems (see e.g. [BKW99, LL03, M92]). In this section, we study computational complexity of VERTEX-COLORABILITY and EDGE-COLORABILITY on graphs of large girth and show that both problems are NP-hard for such graphs.

Theorem 3.1. *For any natural $g \geq 3$, the 3-*EDGE-COLORABILITY *problem is NP-hard in the class of cubic graphs of girth at least g.*

Proof. Let G be a cubic graph. To prove the lemma we will present a polynomial reduction from G to a cubic graph G' with girth at least g such that G is 3-edge-colorable if and only if G' is. Without loss of generality, we can restrict ourselves to even values of g, since graphs of girth at least $g+1$ constitute a subclass of graphs of girth at least g.

For the proof, we shall need a cubic 3-edge-colorable graph of girth at least g. The existence of such a graph follows from a result of Imrich, who proved in [I84] that for any integer $d > 2$, there are infinitely many regular graphs F of degree d whose girth $g(F)$ satisfies the inequality

$$g(F) > \frac{c \log n(F)}{\log(d-1)} - 2,$$

where c is a constant and $n(F)$ is the number of vertices of F. Therefore, for any $g \geq 3$, there is a regular graph F of degree g and of girth at least g. By replacing each vertex of F with a cycle of length g (see Figure 1 for illustration in case of $g = 4$), we obtain a cubic graph H of girth at least g. If g is even, we need only 2 colors to color the edges of each inserted cycle. The third color can be used to color the remaining edges of the graph (i.e. the original edges of G). Hence H is 3-edge-colorable and we can also assume it is connected.

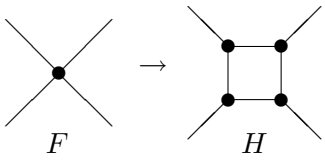

Figure 3.1: Replacement of a vertex of degree 4 with a cycle of length 4

Let ab be an arbitrary edge in H and H_{ab} the graph obtained from H by removing the edge ab. Observe that the distance between a and b in H_{ab} is at least $g-1$ and H_{ab} is connected. Connectedness follows from the fact that every cubic 3-edge-colorable connected graph is bridgeless, i.e. removing an edge does not disconnect the graph (see [D05]).

Now we transform G into G' by replacing each of its edges with a copy of the graph H_{ab} as follows. Given an edge xy in G, we first delete this edge, then incorporate a copy of the graph H_{ab}, and finally, connect x to a and y to b. Since the distance between a and b in each copy of H_{ab} is at least $g-1$, the girth of G' is at least g.

Now we show that G is 3-edge-colorable if and only if G' is.

It is not difficult to see that 3-edge-colorability of G together with 3-edge-colorability of H imply 3-edge-colorability of G'. To prove the converse statement, assume G' is 3-edge-colorable, and let x, y be a pair of vertices of G' that are adjacent in G. Also, let H_{ab}^{xy} be a copy of the graph H_{ab} such that x is adjacent to a and y is adjacent to b in G'. Our goal is to show that in any 3-edge-coloring of G' the edges xa and yb have the same color. Assume the contrary: the color of xa is 1, while the color of yb is 2. In the subgraph of G' induced by the edges of colors 1 and 2, every connected component is a cycle (since this subgraph is 2-regular). The edges xa and yb belong to a same component C of this subgraph, as these edges form a cutset in G'. Let P be the path connecting a to b in C and consisting of edges of the graph H_{ab}^{xy}. According to our assumption, the number of vertices in P is odd. But then the subgraph of H_{ab}^{xy} induced either by the edges of colors 1,3 or by the edges of colors 2,3 is not 2-regular. This contradiction shows that the edges xa and yb have the same color in any 3-edge-coloring of G', which completes the proof of the theorem. □

A natural consequence of the above theorem is the following corollary.

Corollary 3.2. *For any natural $g \geq 3$, the* EDGE-COLORABILITY *problem is NP-hard in the class of graphs of girth at least g.*

In the rest of the section we study vertex colorability.

Theorem 3.3. *For every natural $k, g \geq 3$, k-*VERTEX-COLORABILITY *is NP-hard in the class of graphs of girth at least g.*

Proof. The famous theorem of Erdös ([E59]) states that for each pair of integers g, k ($g \geq 3$, $k \geq 2$) there exists a graph with girth at least g and chromatic number at least k. For a graph of girth at least g and chromatic number at least $k+1$, let H be its edge-minimal $(k+1)$-vertex-colorable subgraph. By definition of H, for any edge ab, the graph H_{ab}, obtained from H be removing ab is k-vertex-colorable and vertices a, b receive the same color in every k-vertex-coloring of H_{ab}. Notice that the distance between a and b in H_{ab} is at least $g-1$.

To prove the theorem, we will show that an arbitrary graph G can be transformed in polynomial time into a graph G' of girth at least g such that G' is k-vertex-colorable if and only if G is. To this end, consider any short cycle C in G and any vertex v on C. Split the neighborhood of v into two parts A, B so that one of the neighbors of v on the cycle is in A, while the other one is in B. Remove vertex v from G and add a copy of the graph H_{ab} defined above along with the edges connecting a to the vertices of A and the edges connecting b to the vertices of B. It is easy to see that the graph obtained in this way is k-vertex-colorable if and only if G is.

Repeating this operation, we can destroy all short cycles in G, thus creating a graph which is k-vertex-colorable if and only if G is. Since the number of short cycles is bounded by a polynomial in the size of the input graph, the overall time required to destroy all cycles shorter than g is bounded by a polynomial. This proves the theorem.

□

Corollary 3.4. *For any natural $g \geq 3$, the* VERTEX-COLORABILITY *problem is NP-hard in the class of graphs with girth at least g.*

3.3 Graphs without long cycles

In this section we study graphs without long cycles. Unlike graphs without short cycles, here we have to distinguish between graphs containing no *long* cycles and graphs without *long induced* cycles. The maximum length of a cycle in a graph is called its *circumference*, while the *chordality* of a graph is the maximum length of a induced cycle. Clearly, graphs of circumference at most c constitute a subclass of graphs of chordality at most c. If $c < 3$, these two classes are identical and coincide with the class of forests, i.e., graphs every connected component of which is a tree. It is easy to see that restricted to trees both VERTEX-COLORABILITY and EDGE-COLORABILITY can be solved in polynomial time. A related result deals with the notion of partial k-trees, or equivalently, graphs of tree-width at most k. It has been shown in [B90, ZNN96] (resp. [TP97]) that EDGE-COLORABILITY (resp. VERTEX-COLORABILITY) of graphs of bounded tree-width is a polynomially solvable task. We use this result to show that both problems have polynomial solutions for graphs of bounded circumference.

Theorem 3.5. *For any natural c, there exists a constant k such that the tree-width of graphs of circumference at most c is at most k.*

Proof. If $c < 3$, then $k = 1$, as forests are exactly graphs of tree-width at most 1. For $c \geq 3$, we use the induction on c and the following two observation (the proof of which can be found, for instance, in [LR04]): first, the tree-width of a graph cannot be larger than the tree width of any of its blocks (maximal 2-connected subgraphs), and second, the addition of j vertices to a graph increases its tree-width by at most j.

Let G be a graph of circumference at most c and let H be a block in G. To prove the theorem, we will show that by deleting at most c vertices from H we can obtain a graph of circumference at most $c-1$. This is obvious in the case when H contains at most one cycle of length c. Now let C^1 and C^2 be two cycles of length c in H. Assume they are vertex disjoint. Consider two edges $e_1 \in C^1$ and $e_2 \in C^2$. Since H is 2-connected, there is a cycle in H containing both e_1 and e_2. In this cycle, one can distinguish two disjoint paths P' and P'', each of which contains the endpoints in C^1 and C^2, and the remaining vertices outside the cycles. The endpoints of the paths P' and P'' partition each of the

cycles C^1 and C^2 into two parts. The larger parts in both cycles together with paths P' and P'' form a cycle of length at least $c+2$, contradicting the initial assumption. This contradiction shows that any two cycles of length c in H have a vertex in common. Therefore, removing the vertices of any cycle of length c from H results in a graph of circumference at most $c-1$, as required. □

Corollary 3.6. *For any natural c, the* VERTEX-COLORABILITY *and* EDGE-COLORABILITY *problems can be solved for graphs of circumference at most c in polynomial time.*

We complete the chapter by discussing complexity of the problems on graphs of bounded chordality. For VERTEX-COLORABILITY, such restrictions do not generally lead to an efficient solution: indeed, 4-VERTEX-COLORABILITY remains NP-hard for graphs of chordality at most 8 (since it is NP-hard for graphs containing no path on 8 vertices as an induced subgraph [LRS06]) and VERTEX 5-VERTEX-COLORABILITY remains NP-hard for graphs of chordality at most 8 (since it is NP-hard for graphs containing no path on 8 vertices as an induced subgraph [SW01]). However, for graphs of chordality at most 3 (chordal graphs) the problem of VERTEX COLORABILITY is known to be solvable in polynomial time (see [G04]).

The authors are not aware of the status of the EDGE COLORABILITY problem on graphs of bounded chordality. However, polynomial-time solvability of its decision version can be easily derived from some known results.

Theorem 3.7. *For any natural k and c, the* EDGE k-COLORABILITY *problem on graphs of chordality at most c can be solved in polynomial time.*

Proof. Since graphs of maximum vertex degree $k+1$ are not k-edge-colorable, the problem can be restricted to graphs of degree at most k. It has been shown by Bodlaender and Thilikos [BT97] that if a graph has chordality at most c and maximum degree at most k, then its treewidth is at most $k\,(k-1)^{c-3}$. As we mentioned before, coloring the edges of graphs of bounded tree-width is a polynomially solvable task [ZNN96]. □

Chapter 4

Vertex 3-colorability of claw-free graphs

The 3-VERTEX-COLORABILITY problem is NP-hard in the class of *claw*-free graphs and it remains hard in many of its subclasses obtained by forbidding additional subgraphs. (Line graphs and *claw*-free graphs of vertex degree at most four provide two examples.)

In this section we study the computational complexity of the 3-VERTEX-COLORABILITY problem in subclasses of *claw*-free graphs defined by finitely many forbidden subgraphs. We prove a necessary condition for polynomial-time solvability of the problem in such classes and propose a linear-time algorithm for an infinitely increasing hierarchy of classes of graphs. The algorithm is based on a generalization of the notion of locally connected graphs. The results presented in this section come from the paper [KL07a].

4.1 Introduction

A *claw* is a complete bipartite graphs $K_{1,3}$ with parts of size one and three. A graph is called *claw*-free if it does not contain an induced subgraph isomorphic to the claw.

In this section we study the 3-VERTEX-COLORABILITY problem restricted to the class of *claw*-free graphs. Finding a 3-edge-coloring in a graph G is equivalent to finding a 3-vertex-coloring in the line graph of G. 3-EDGE-COLORABILITY can be thought of as a subproblem of 3-VERTEX-COLORABILITY on *claw*-free graphs since line graphs are *claw*-free. (In fact line graphs form a proper subclass of *claw*-free graphs, see [H69].)

In this chapter we study computational complexity of the problem in other subclasses of *claw*-free graphs defined by finitely many forbidden induced subgraphs. First we prove a necessary condition for polynomial-time solvability of the problem in such classes, and then for an infinitely increasing hierarchy of classes that satisfy the condition, we propose a linear-time solution. To develop such a solution for the basis of this hierarchy,

we generalize the notion of locally connected graphs that has been recently studied in the context of the 3-VERTEX-COLORABILITY problem.

A *wheel* on n vertices, denoted by W_n, is obtained from the cycle C_n by adding a vertex adjacent to every vertex of the cycle. A *diamond* is the graph obtained from K_4 by removing one edge; we denote it by K_4-e. A *gem* is the graph obtained from P_4 by adding a vertex adjacent to all vertices of the path.

In this section we will assume that the minimum degree of the input graph is at least three. Notice that if a graph G contains a vertex v of degree one or two, then G is 3-colorable if and only if $G-v$ is.

Every graph with five vertices contains either a triangle or its complement or a C_5. Therefore, every graph with a vertex of degree five or more contains either a claw or K_4 or W_5. Since K_4 and W_5 are not 3-colorable, we conclude that every 3-colorable *claw*-free graph, has maximum vertex degree at most 4. Therefore below we will also assume that the the maximum degree of the input graph is at most four.

4.2 NP-hardness

In this section we establish several results on the NP-hardness of the 3-VERTEX-COLORABILITY problem in subclasses of *claw*-free graphs. We start by recalling the following known fact.

Lemma 4.1. *The 3-*VERTEX-COLORABILITY *problem is NP-hard in the class of* (*claw*, *diamond*)*-free of maximum vertex degree four.*

The result follows by a reduction from 3-EDGE-COLORABILITY of triangle-free cubic graphs, which is an NP-hard problem ([H81]). It is not difficult to verify that if G is a triangle-free cubic graph, then $L(G)$ is a (claw, K_4-e)-free regular graph of degree four.

To establish more results, let us introduce more definitions and notation. First, we introduce the following three operations:

- replacement of an edge by a diamond (Figure 4.1);
- implantation of a diamond at a vertex (Figure 4.2);
- implantation of a triangle into a triangle (Figure 4.3)

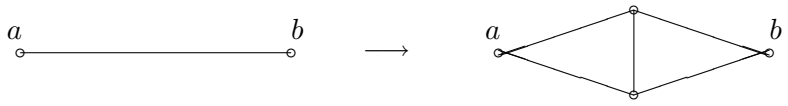

Figure 4.1: Replacement of an edge by a diamond

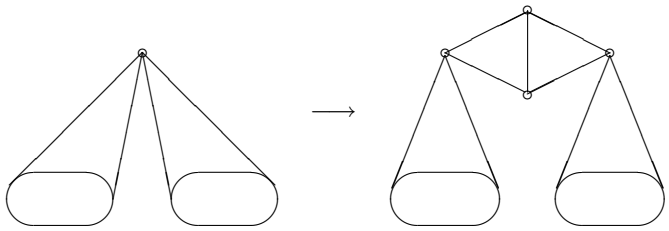

Figure 4.2: Diamond implantation

Observe that a graph obtained from a graph G by diamond or triangle implantation is 3-colorable if and only if G is. Denote by

$T_{i,j,k}$ the graph represented in Figure 4.4;

$T^1_{i,j,k}$ the graph obtained from $T_{i,j,k}$ by replacing each edge, which is not in the triangle, by a diamond;

$T^2_{i,j,k}$ the graph obtained from $T^1_{i,j,k}$ by implanting into its central triangle a new triangle.

Finally, denote by

$\mathcal{T}$ the class of graphs every connected component of which is an induced subgraph of a graph of the form $T_{i,j,k}$;

$\mathcal{T}^1$ the class of graphs every connected component of which is an induced subgraph of a graph of the from $T^1_{i,j,k}$;

$\mathcal{T}^2$ the class of graphs every connected component of which is an induced subgraph of a graph of the from $T^2_{i,j,k}$.

Notice none of the classes $\mathcal{T}^1$ and $\mathcal{T}^2$ contains the other. Indeed, $\mathcal{T}^2 \setminus \mathcal{T}^1$ contains a gem, while $\mathcal{T}^1 \setminus \mathcal{T}^2$ contains the graph $T^{\Delta}_{0,0,0}$ (see Figure 4.5 for the definition of $T^{\Delta}_{i,j,k}$). On the other hand, both $\mathcal{T}^1$ and $\mathcal{T}^2$ include the class $\mathcal{T}$.

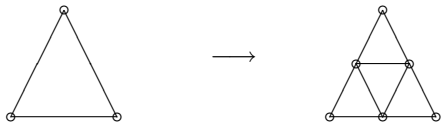

Figure 4.3: Triangle implantation

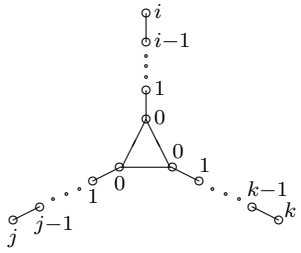

Figure 4.4: The graph $T_{i,j,k}$

Theorem 4.2. *Let X be a subclass of claw-free graphs defined by a finite set M of forbidden induced subgraphs. If $M \cap \mathcal{T}^1 = \emptyset$ or $M \cap \mathcal{T}^2 = \emptyset$, then the 3-*VERTEX-COLORABILITY *problem is NP-hard for graphs in the class X.*

Proof. We prove the theorem by a reduction from the class of $(claw, gem, W_4)$-free graphs of vertex degree at most 4, where the problem is NP-hard, since this class is an extension of $(claw, diamond)$-free graphs of maximum degree 4.

Let G be a $(claw, gem, W_4)$-free graph of vertex degree at most 4. Without loss of generality we can also assume that G is K_4-free (since otherwise G is not 3-colorable) and every vertex of G has degree at least 3. We will show that if $M \cap \mathcal{T}^1 = \emptyset$ or $M \cap \mathcal{T}^2 = \emptyset$, then G can be transformed in polynomial time into a graph in X, which is 3-colorable if and only if G is. Assume first that $M \cap \mathcal{T}^1 = \emptyset$.

Let us call a triangle in G *private* if it is not contained in any diamond. Also, we shall call a vertex x *splittable* if the neighborhood of x can be partitioned into two disjoint cliques X_1, X_2 with no edges between them. In particular, in a $(claw, gem, W_4, K_4)$-free graph every vertex of a private triangle is splittable. Also, it is not difficult to verify that every chordless cycle of length at least 4 contains a splittable vertex.

Given a splittable vertex x with cliques X_1, X_2 in its neighborhood, apply the diamond implantation k times, i.e., replace x with two new vertices x_1 and x_2, connect x_i to every vertex in X_i for $i = 1, 2$, and connect x_1 to x_2 be a chain of k diamonds. Obviously the graph obtained in this was is 3-colorable if and only if G is. We apply this operation to every splittable vertex of G and denote the resulting graph by $G(k)$. Observe that $G(k)$ is $(C_4, \ldots, C_k)$-free and the distance between any two private triangles is at least k.

Let us show that if k is larger than the size of any graph in M, then $G(k)$ belongs to X. Assume by contradiction that $G(k)$ does not belong to X, then it must contain an induced subgraph $A \in M$. We know that A cannot contain chordless cycles $C_4, \ldots, C_k$. Moreover, it cannot contain cycles of length greater than k, since A has at most k vertices. For the same reason, each connected component of A contains at most one private triangle. But then $A \in \mathcal{T}^1$, contradicting our assumption that $M \cap \mathcal{T}^1 = \emptyset$.

Now assume that $M \cap \mathcal{T}^2 = \emptyset$. In this case, we first transform G into $G(k)$ and then implant a new triangle into each private triangle of G. By analogy with the above case, we conclude that the graph obtained in this way belongs to X, thus completing the necessary reduction. □

The above theorem not only proves NP-hardness of the problem in certain classes, but also suggests what classes can have the potential for accepting a polynomial-time solution. In particular, for subclasses of *claw*-free graphs defined by a single additional induced subgraph we obtain the following corollary of Theorem 4.2 and Lemma 1.

Corollary 4.3. *If X is the class of $(claw, H)$-free graphs, then the 3-*VERTEX-COLORABILITY *problem can be solved in polynomial time in X only if $H \in T^{\Delta}$, where T^{Δ} is the class of graphs every connected component of which is an induced subgraph of a graph of the form $T^{\Delta}_{i,j,k}$ with at least two non-zero indices (see Figure 4.5) .*

In the next section, we analyze some of $(claw, H)$-free graphs with $H \in \mathcal{T}^{\Delta}$ and derive polynomial-time solutions for them.

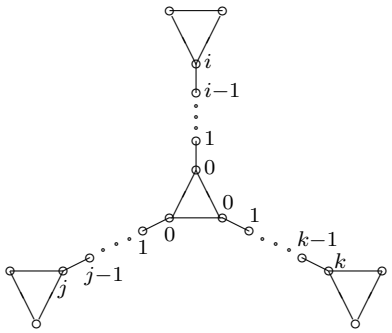

Figure 4.5: The graph $T^{\Delta}_{i,j,k}$

4.3 Polynomial-time results

We start by reporting several results that are known from the literature. It has been shown in [RST02] that the 3-VERTEX-COLORABILITY problem can be solved in polynomial time for $(claw, T_{0,0,2})$-free graphs. More generally, we can show that

Theorem 4.4. *The 3-*VERTEX-COLORABILITY *problem can be solved in polynomial time in the class of $(claw, T_{i,j,k})$-free graphs for any i, j, k.*

Proof. First, observe that 3-VERTEX-COLORABILITY is a problem solvable in polynomial time on graphs of bounded clique-width [CMR00]. From the results in [LR04] it follows that $(claw, T_{i,j,k})$-free graphs of bounded vertex degree have bounded clique-width for any i, j, k. In the introduction, it was observed that *claw*-free graphs containing a vertex of degree 5 or more are not 3-colorable, which provides the desired conclusion. □

Another solvable case described in [RST02] deals with $(claw, hourglass)$-free graphs, where an hourglass is the graph consisting of a vertex of degree 4 and a couple of disjoint edges in its neighborhood. Unfortunately, the authors of [RST02] do not claim any time complexity for their solution. In Section 4.3.1 we show that this case can be solved in linear time by generalizing the notion of locally connected graphs studied recently in connection with the 3-VERTEX-COLORABILITY problem. Then in Section 4.3.2 we extend this result to an infinitely increasing hierarchy of subclasses of *claw*-free graphs.

4.3.1 Almost locally connected graphs

A graph G is *locally connected* if for every vertex $v \in V$, the graph $G[N(v)]$ induced by the neighborhood of v is connected. The class of locally connected graphs has been studied in [K05] in the context of the 3-VERTEX-COLORABILITY problem.

A notion, which is closely related to 3-coloring, is 3-*clique ordering*. In a connected graph G, an ordering $(v_1, \ldots, v_n)$ of vertices is called a 3-*clique ordering* if v_2 is adjacent to v_1, and for each $i = 3, \ldots, n$, the vertex v_i forms a triangle with two vertices preceding v_i in the ordering.

It is not difficult to see that for a graph with a 3-clique-ordering, the 3-VERTEX-COLORABILITY problem is solvable in time linear in the number of edges. (We will refer to such running time of an algorithm as *linear time*.) In [K05], it has been proved that if G is connected and locally connected, then G admits a 3-clique-ordering and it can be found in linear time. Therefore, the 3-VERTEX-COLORABILITY problem in the class of locally connected graphs can be solved in linear time.

Now we introduce a slightly broader class and show that the 3-VERTEX-COLORABILITY of graphs of vertex degree at most 4 in the new class can be decided in linear time. This will imply, in particular, a linear-time solution for $(claw, hourglass)$-free graphs.

We say that a graph G is *almost locally connected* if the neighborhood of each vertex either induces a connected graph or is isomorphic to $K_1 + K_2$ (disjoint union of an edge and a vertex). In other words, the neighborhoods of all vertices are connected, or, if the degree of a vertex is 3, we allow the neighborhood to be disconnected, provided it consists of two connected components, one of which is a single vertex. If v is a vertex of degree 3 and w is an isolated vertex in the neighborhood of v, then we call the edge vw a *pendant edge*.

A maximal (with respect to set inclusion) subset of vertices that induces a 3-clique orderable graph will be called *3-clique orderable component*. Since a pendant edge belongs to no triangle in an almost locally connected graph, the endpoints of the pendant edge form a 3-clique orderable component, in which case we call it *trivial*. For the non-trivial 3-clique orderable components we prove the following helpful claim.

Claim 4.5. *Let G be an almost locally connected graph with $\Delta(G) \leq 4$ and with at least 3 vertices.*

(1) Every non-trivial 3-clique orderable component of G contains at least 3 vertices.

(2) Any two non-trivial 3-clique orderable components are disjoint.

(3) Any edge of G connecting two vertices in different non-trivial 3-clique orderable components is pendant.

Proof. To see (1), observe that in an almost locally connected graph with at least 3 vertices every non-pendant edge belongs to a triangle.

To prove (2), suppose G contains two non-trivial 3-clique orderable components M_1 and M_2 sharing a vertex v. Without loss of generality let v have a neighbor w in $M_1 - M_2$. Since M_1 is not trivial, the edge vw cannot be pendant, and hence there is a vertex $u \in M_1$ adjacent both to v and w. Notice that u cannot belong to M_2, since otherwise $M_2 \cup \{w\}$ induces a 3-clique orderable graph contradicting maximality of M_2. On the other hand, M_2 must have a triangle containing vertex v, say v, x, y. If neither u nor w has a neighbor in $\{x, y\}$, then G is not locally connected. If there is an edge between $\{u, w\}$ and $\{x, y\}$, then M_2 is not a maximal set inducing a 3-clique orderable graph. This contradiction proves (2), which in its turn implies (3) in an obvious way. □

An obvious corollary from the above claim is that an almost locally connected graph admits a unique partition into 3-clique orderable components and such a partition can be found in linear time.

Now we present a linear-time algorithm that, given a connected, almost locally connected graph G, decides 3-VERTEX-COLORABILITY of G and finds a 3-coloring, if one exists. In the algorithm, the operation of *contraction* of a set of vertices A means substitution of A by a new vertex adjacent to every neighbor of the set A.

ALGORITHM $\mathcal{A}$

Step 1. Find the unique partition $V_1, \ldots, V_k$ of G into 3-clique orderable components.

Step 2. If one of the graphs $G[V_i]$ is not 3-colorable, return the answer G IS NOT 3-COLORABLE. Otherwise, 3-color each 3-clique orderable component.

Step 3. Create an auxiliary graph H, contracting color classes in each 3-clique orderable component V_i and then removing vertices of degree 2.

Step 4. If H is isomorphic to K_4, return the answer G IS NOT 3-COLORABLE. Otherwise, 3-color H.

Step 5. Expand the coloring of H to a coloring of G and return it.

To show that the algorithm is correct and runs in linear time, we need to prove two properties of the auxiliary graph built in Step 3 of the algorithm.

Claim 4.6. *Let H be the auxiliary graph built in Step 3 of the algorithm. G is 3-colorable if and only if H is.*

Proof. If G is 3-colorable, then each 3-clique orderable component has a unique coloring. Therefore, the graph obtained by contracting color classes of each 3-clique orderable component (and, possibly, removing vertices of degree 2) is 3-colorable.

If H is 3-colorable, then vertices of degree 2 that were deleted in Step 3 can be restored and receive colors that are not assigned to their neighbors. Let a be a vertex of a triangle in H corresponding to a 3-clique orderable component in G. The edges that a is incident to (but not the triangle edges) correspond to the pendant edges of this component. Each is incident to two vertices in different color classes in H and will be in G. □

From the definition of 3-clique-ordering one can derive the following simple observation.

Claim 4.7. *If G has a 3-clique-ordering, then $m \geq 2n - 3$.*

Claim 4.8. *Let G be a connected and almost locally connected graph of vertex degree at most 4, and H be the auxiliary graph built in Step 3 of the algorithm. Then, $\Delta(H) \leq 3$.*

Proof. Let T be a 3-clique orderable component of G. Denote by n_2, n_3, n_4 the number of vertices of degree $2, 3$ and 4 in $G[T]$, respectively. Also, let m' be the number of edges in $G[T]$. Then, $2m' = 2n_2 + 3n_3 + 4n_4$. In addition, $m' \geq 2(n_2 + n_3 + n_4) - 3$ (by Claim 4.7). Therefore, $n_2 \leq 3$. Notice that any pendant edge in G is adjacent to

two vertices of degree 3 that are of degree 2 in the graphs induced by their 3-clique orderable components. Therefore, the number of pendant edges of a 3-clique orderable component is at most 3.

If a, b, c are vertices of a triangle corresponding to a locally connected component, and three pendant edges are incident to a, then b and c are removed in Step 4 of the algorithm, as both have degree 2, and a has degree 3 in H. If a is incident to two pendant edges, and b to one, then c is removed (as has degree 2) and both b and c have degree at most 3 in H. If each of vertices a, b, c is incident to one pendant edge only, then the degree of each is at most 3. □

Theorem 4.9. *Algorithm $\mathcal{A}$ decides 3-*VERTEX-COLORABILITY *of a connected, almost locally connected graph G with maximum vertex degree at most 4 and finds a 3-coloring of G, if one exists, in linear time.*

Proof. The correctness of the algorithm follows from Claims 4.6, 4.8 and the observation that by the Brooks theorem the only graph of maximum degree 3 that is not 3-colorable is K_4.

The first step of the algorithm can be implemented as a breadth first search and performed in linear time. 3-coloring each of the locally connected components in linear time can be done by applying the algorithm presented in [K05]. Also, building the auxiliary graph and testing if it is isomorphic to K_4 are linear time tasks. As was proved in [L75], Step 4 may be performed in linear time as well, and Step 5 is also clearly linear. □

Corollary 4.10. *If G is $(claw, hourglass)$-free, then there exists a linear time algorithm to decide 3-*VERTEX-COLORABILITY *G, and find a 3-coloring, if one exists.*

Proof. If $\Delta(G) \geq 5$, then G is not 3-colorable and it can be checked in linear time. Notice that the only disconnected neighborhoods that are allowed in a 3-colorable, *claw*-free graphs are $2K_2$ and $K_1 + K_2$. Since G is H-free, none of the neighborhoods is isomorphic to $2K_2$; otherwise, the graph induced by a vertex together with its neighborhood would be isomorphic to H. Hence, all neighborhoods in G are either connected or isomorphic

to $K_1 + K_2$. Therefore, the graph is almost locally connected and by Lemma 4.9, the 3-VERTEX-COLORABILITY problem can be solved for it in linear time. □

4.3.2 More general classes

In this section we extend the result of the previous one to an infinitely increasing hierarchy of subclasses of *claw*-free graphs. The basis of this hierarchy is the class of $(claw, hourglass)$-free graphs. Now let us denote by H_k the graph obtained from a copy of *hourglass* H and a copy of P_k by identifying a vertex of degree 2 of H and a vertex of degree 1 of P_k. In particular, $H_1 = H$.

Theorem 4.11. *For any $k \geq 1$, there is a linear time algorithm to decide 3-*VERTEX-COLORABILITY *of a $(K_{1,3}, H_k)$-free graph G, and to find a 3-coloring, if one exists.*

Proof. Without loss of generality we consider only connected K_4-free graphs of vertex degree at most 4 in the class under consideration. For those graphs that are *hourglass*-free, the problem is linear-time solvable by Corollary 4.10. Assume now that a $(claw, H_k)$-free graph G contains an induced hourglass H (which implies in particular that $k > 1$) and let v denote the center of H. There are only finitely many connected graphs of bounded vertex degree that do not have vertices of distance $k+1$ from v. Therefore, without loss of generality, we may suppose that G contains a vertex x_{k+1} of distance $k+1$ from v, and let $x_{k+1}, x_k, \ldots, x_1, v$ be a shortest path connecting x_{k+1} to v. In particular, x_1 is a neighbor of v. Since G is H_k-free, x_2 has to be adjacent to at least one more vertex, say u, in the neighborhood of v. If u is not adjacent to x_1, then u, x_1, x_2, x_3 induce a claw. Thus, u is adjacent to x_1, while x_2 has no neighbors in $N(v)$ other than x_1 and u.

Let us show that u has degree 3 in G. To the contrary, assume z is the fourth neighbor of u different from v, x_1, x_2. To avoid the induced claw $G[u, v, x_2, z]$, z has to be adjacent to x_2. This implies z is not adjacent to x_1 (else a $K_4 = G[x_1, u, x_2, z]$ arises), and z is adjacent to x_3 (else x_2, x_1, z, x_3 induce a claw). But now the path $x_k, \ldots, x_3, z, u$ together with v and its neighbors induce an H_k; a contradiction. By symmetry, x_1 also has degree 3 in G.

Notice that in any 3-coloring of G, vertices v and x_2 receive the same color. Therefore, by identifying these two vertices (more formally, by replacing them with a new vertex adjacent to every neighbor of v and x_2) and deleting u and x_1, we obtain a new graph G' which is 3-colorable if and only if G is. Moreover, it is not hard to see that the new graph is again $(claw, H_k)$-free. By applying this transformation repeatedly we reduce the initial graph to a graph G" which is either $(claw, hourgalss)$-free or contains no vertices of distance $k+1$ from the center of an hourglass. In both cases the problem is linear-time solvable, which completes the proof. □

Chapter 5

An exact algorithm for solving Max-Cut on graphs with bounded maximum degree

Analyzing a simple ENUMERATE & SOLVE approach, we obtain new algorithmic results for the MAX-CUT problem. In particular, we propose the fastest known polynomial-space algorithm for MAX-CUT in the class of graphs with bounded maximum degree Δ (for $\Delta > 7$). Our algorithm runs in time $O^*(2^{(1-2/\Delta)n})$. The results presented in this section are based on the paper [DKP07].

5.1 Introduction

The MAX-CUT was one of the first problems whose **NP**-hardness was established. The problem also seems challenging in the context of exact algorithms since it is not known whether there exists a polynomial-space algorithm solving this problem in time $O^*(c^n)$ for $c < 2$. (We say that a function f belongs to the set $O^*(g(n))$ if and only if f is in the set $O(poly(n) \cdot f(n))$ for some polynomial $poly(n)$.)

The MAX-CUT-EXTENSION is a version of the problem in which for some vertices it has already been decided to which part they belong and the goal is to find a maximal cut subject to the prepartition. We prove that if the graph induced by the vertices that have not been assigned to any part is bipartite, then the MAX-CUT-EXTENSION problem can be solved in polynomial time.

This result allows us to apply the ENUMERATE & SOLVE technique. Having found a large bipartite subgraph B of the input graph G, we enumerate all assignments of vertices in $G - B$ to two different parts of the cut and then solve an instance of MAX-CUT-EXTENSION for each such preassignment. This approach yields a fast exponential algorithm in the class of graphs with bounded maximum degree.

5.2 Definitions

Recall that a *cut* $C = (V_0, V_1)$ in a graph is a partition of its vertex set V into two disjoint subsets V_0 and V_1. Notice that the characteristic vector of one of the parts, say V_0, uniquely determines the partition.

For the purpose of this section, we will think of a partition as an assignment of $0-1$ values to the vertices of the graph. Let x_i be a Boolean variable which takes value 0, if $v_i \in V_0$, and 1, if $v_i \in V_1$. The weight of a cut in a graph $G = (V, E, w)$ can be expressed as a pseudo-boolean function,

$$w(C) = \sum_{ij \in E} w_{ij} \left(x_i \overline{x_j} + \overline{x_i} x_j\right) = \sum_{i \in V} w_i x_i - 2 \sum_{ij \in E} w_{ij} x_i x_j, \tag{5.1}$$

where $w_i = \sum_{ij \in E} w_{ij}$.

Given a graph G as the input, the MAX-CUT problem asks to compute a cut in G that maximizes (5.1).

It is easy to see that if the weights are restricted to be nonnegative real numbers, the MAX-CUT problem can be solved in polynomial time for the class of bipartite graphs.

5.3 Previous work

The worst-case complexity of the maximum cut problem has been studied in a few papers, some of them dealing with the weighted and some with the unweighted case. The problem of finding a maximum cut can be modeled as an instance of the constraint satisfaction problem with two variables per clause (2-CSP) or an instance of maximum satisfiability with the two variables per clause (MAX-2-SAT). Fast algorithms for any of these two problems yield efficient algorithms for SIMPLE-MAX-CUT.

The fastest algorithm for SIMPLE-MAX-CUT in arbitrary graphs was proposed by Williams in [W05]. In fact, the algorithm computes the number of solutions to an instance of a 2-CSP problem and employs interesting, non-standard techniques. Used as a SIMPLE-MAX-CUT solver, the algorithm runs in time $O^*(2^{\omega n/3})$ but, unfortunately, requires exponential space of $O^*(2^{\omega n/3})$, where n is the number of vertices of the input graph, $\omega < 2.376$ is the matrix multiplication exponent (the product of two $k \times k$ matrices

can be computed in time $O(k^\omega)$). As mentioned before, it is not known whether there exists a polynomial-space algorithm that computes SIMPLE-MAX-CUT and runs faster than the naive one of time complexity $O^*(2^n)$. This is one of open questions listed in [W04].

More algorithms have been developed for sparse graphs. The upper bounds on their running times are given as linear functions of the number of edges in the input graph. (The number of edges of the input graph is denoted by m.) It makes them faster than the algorithms whose running time is bounded by a linear function of the number of vertices (like [W05] or the naive algorithm) only if m is linearly bounded by n.

In [GHNR03] an algorithm solving SIMPLE-MAX-CUT (via MAX-2-SAT) in time $O^*(2^{m/3})$ was proposed by Gramm et al. The bound was then improved to $O^*(2^{m/4})$ by Fedin and Kulikov in [FK06]. Their algorithm solves the maximum cut problem in a graph with integer weights on its edges. In a paper by Scott and Sorkin ([SS03]; see also [SS04]) a faster algorithm for MAX-CUT, running in time $O^*(2^{\min((m-n)/2,m/5)})$, was described. A recent paper by Kneis and Rossmanith ([KR05]) offers a SIMPLE-MAX-CUT algorithm with running time $O^*(2^{m/5.217})$. All of those algorithms use polynomial space.

5.4 Our contribution

In this section we apply the ENUMERATE & SOLVE technique that seems to be a new approach to the MAX-CUT problem. The method consists in enumerating cuts in a subgraph B of G and then extending them in an optimal way to cuts in G. The technique is applied to graphs with bounded maximum degree and to general graphs. In both cases, we obtain an exponential-time algorithm that uses polynomial space.

For some classes of graphs our algorithms offer the best running time known. In particular, we obtain the fastest known algorithm solving the MAX-CUT problem in the class of graphs with bounded maximum degree Δ, if $\Delta = 8, 9$. Our results also yield a MAX-CUT algorithm and a polynomial-space SIMPLE-MAX-CUT algorithm, that are the fastest known in the class of graphs with bounded maximum degree Δ, for $\Delta \geq 8$.

For weighted graphs with bounded maximum degree Δ, we present an algorithmic

scheme that computes a maximum cut. For fixed Δ, the algorithm runs in time $O^*(2^{(1-(2/\Delta))n})$ and polynomial space. For $\Delta \geq 8$, our algorithm is faster than the MAX-CUT algorithm from [SS04] and the SIMPLE-MAX-CUT algorithm from [KR05]. It is slower than the exponential-space SIMPLE-MAX-CUT algorithm from [W05] for $\Delta \geq 10$.

For general weighted graphs, we obtain an algorithm that computes a maximum cut and runs in time $2^{mn/(m+n)}$. Our algorithm is faster than the MAX-CUT algorithm from [SS04] for $m > 4n$ and faster than the SIMPLE-MAX-CUT algorithm from [KR05] for $m > 4.217n$. It is slower than the SIMPLE-MAX-CUT exponential-space [W05] for $m > \omega n/(3-\omega) > 3.808n$.

5.5 Extending a partial partition of vertices

In this section we consider a modification of the MAX-CUT problem. Suppose some of the vertices have already been partitioned into two subsets and now the problem is to find an optimal cut in the graph with respect to that pre-partition. We prove that if the graph induced by the vertices that have not yet been partitioned is bipartite, then the problem of finding an optimal extension of the partial partition can be solved in polynomial time. The algorithms presented in the following sections are based on this result.

Let $U \subseteq V$ be a subset of vertices of G such that the subgraph $G' = G[U']$ induced by the vertices in $U' = V \setminus U$ is bipartite. Also, let (U_0, U_1) be a partition of U into two subsets. Consider the problem of finding a partition (V_0, V_1) of V with $U_0 \subseteq V_0$ and $U_1 \subseteq V_1$ that maximizes (5.1). We refer to this problem as MAX-CUT-EXTENSION.

The vertices in U have already been assigned to some parts of the cut, thus variables x_i, for $i \in U$, have their values fixed. There are four possible types of edges in the cut: edges with both endpoints in U, from U_0 to U', from U_1 to U', and with both endpoints in U'. The problem of finding an optimal extension of the pre-partition is now equivalent to maximizing the following pseudo-boolean function,

$$\sum_{\substack{i \in U_0 \\ j \in U_1}} w_{ij} + \sum_{\substack{i \in U_0 \\ j \in U'}} w_{ij} x_j + \sum_{\substack{i \in U_1 \\ j \in U'}} w_{ij} \overline{x_j} + \sum_{\substack{i \in U' \\ j \in U'}} w_{ij} \left(x_i \overline{x_j} + \overline{x_i} x_j \right) \tag{5.2}$$

where all sums are taken over edges $ij \in E$ of the graph G. Putting,

$$c_j = \sum_{i \in U_0} w_{ij} - \sum_{i \in U_1} w_{ij} + \sum_{i \in U'} w_{ij}$$

where all sums are again taken over edges $ij \in E$, and omitting the constant term, the problem is equivalent to finding a maximum of the function,

$$\sum_{j \in U'} c_j x_j - 2 \sum_{ij \in E'} w_{ij} x_i x_j \tag{5.3}$$

where E' is the edge set of the bipartite graph G'. In other words, the problem of finding an optimal extension of the pre-partition can be stated as the following integer quadratic program:

$$\begin{aligned} \max \quad & \sum_{j \in U'} c_j x_j - 2 \sum_{ij \in E'} w_{ij} x_i x_j \\ \text{s.t.} \quad & x_i \in \{0, 1\} \end{aligned} \tag{5.4}$$

The standard linearization technique applied to (5.4) by introducing $y_{ij} = x_i x_j$, yields the following integer linear program:

$$\begin{aligned} \max \quad & \sum_{j \in U'} c_j x_j - 2 \sum_{ij \in E'} w_{ij} y_{ij} \\ \text{s.t.} \quad & y_{ij} \geq x_i + x_j - 1 \\ & x_i \in \{0, 1\} \\ & y_{ij} \in \{0, 1\} \end{aligned} \tag{5.5}$$

It is easy to see that (5.4) and (5.5) are equivalent. They have the same optimal value and there is an easy correspondence between their optimal solutions, namely $y_{ij} = x_i x_j$.

Having modelled the original quadratic problem (5.4) as an integer linear program, let us study the continuous relaxation of (5.5):

$$\begin{aligned} \max \quad & \sum_{j \in U'} c_j x_j - 2 \sum_{ij \in E'} w_{ij} y_{ij} \\ \text{s.t.} \quad & y_{ij} \geq x_i + x_j - 1 \\ & x_i \geq 0 \\ & x_j \leq 1 \\ & y_{ij} \geq 0 \\ & y_{ij} \leq 1 \end{aligned} \tag{5.6}$$

Lemma 5.1. *The constraint matrix of the linear program (5.6) is totally unimodular, i.e., the determinant of every square submatrix of it equals* 0 *or* ± 1.

Proof. Let A be the constraint matrix of (5.6). It has $|U'|+|E'|$ columns and $2|U'|+3|E'|$ rows and all its entries are either 0 or ± 1. Let B be an edge-vertex incidence matrix of G', with rows corresponding to edges and columns corresponding to vertices. Notice that B is a submatrix of A. Moreover, any submatrix of A that has two non-zero entries in every row and every column has to be a submatrix of B.

Take any square $k \times k$ submatrix of A. We will prove the lemma by induction on k. Clearly, the result holds for $k = 1$.

Now assume that all $(k-1) \times (k-1)$ submatrices of A are totally unimodular and consider a matrix M which is a $k \times k$ submatrix of A.

If all entries of any row or column of M are 0, then $\det(M) = 0$ and M is totally unimodular. If any row or column of M has a single non-zero element (± 1), then using the expansion method for calculating determinants and the induction hypothesis, it is easy to see that $\det(M)$ is either 0 or ± 1, and A is totally unimodular.

Suppose that each row and each column of M has at least two non-zero entries. Hence, M must be a submatrix of B but, since B is an incidence matrix of a bipartite graph, so is M. It is possible to partition the columns of M into two parts, according to the partition of vertices of bipartite graph. The sum of the columns in each part yields a unit vector (each edge of the bipartite subgraph has one endpoint in each part) and that implies linear dependence of M, therefore $\det(M) = 0$ and M is totally unimodular. □

Theorem 5.2. *Let $U \subseteq V$ be such that the subgraph $G' = G[U']$ induced by the vertices in $U' = V \backslash U$ is bipartite and (U_0, U_1) be a partition of U into two subsets, then the problem of finding a partition (V_0, V_1) of V with $U_0 \subseteq V_0$ and $U_1 \subseteq V_1$ that maximizes (5.1) is polynomial-time solvable.*

Proof. The problem of finding a partition (V_0, V_1) of V with $U_0 \subseteq V_0$ and $U_1 \subseteq V_1$ that maximizes (5.1) can be modelled as the integer quadratic program (5.4) which is equivalent to (5.5). Total unimodularity of the constraint matrix of (5.6) (by Lemma 5.1) implies the existence of an optimal $0-1$ solution of (5.6), and such a solution can be

found in polynomial time (see for example [S99]). Since the relaxation (5.6) of (5.5) has an optimal $0-1$ solution, therefore (5.4) can be solved in polynomial time. □

Before we proceed to the next section, let us briefly describe the algorithmic technique we are going to apply. Given an induced bipartite subgraph $G[B]$ of G, one can enumerate all partitions of $V \setminus B$ and find an optimal extension of each in polynomial time (by Theorem 5.2). The complexity of such a technique is $O^*(2^{|V \setminus B|})$ and it strongly depends on the size of the bipartite subgraph that has to be constructed.

5.6 Algorithm for graphs with bounded maximum degree

In this section we present and analyze an algorithmic scheme $\mathtt{A}(\Delta)$. For a fixed integer Δ ($\Delta \geq 3$), the scheme yields an algorithm whose input is a weighted graph $G = (V, E, w)$ of maximum degree Δ and whose output is a maximum cut in G with respect to the weight function w.

`Step 1.` If G is isomorphic to the complete graph on $\Delta + 1$ vertices, then let B be any pair of vertices and go to `Step 3`.

`Step 2.` Δ-color G. Let B be the union of 2 largest color classes of the coloring.

`Step 3.` Enumerate all partitions of elements of $V \setminus B$ into two subsets (all $0-1$ assignments) and for each find an optimal extension of the partial partition.

`Step 4.` Find a cut C that has the largest weight among all checked in `Step 3`. Return the cut C.

Theorem 5.3. *For a fixed integer Δ ($\Delta \geq 3$), Algorithm $\mathtt{A}(\Delta)$ computes* MAX-CUT *in a graph G in time $O^*(2^{(1-(2/\Delta))n})$ and polynomial space.*

Proof. Let us first notice that the algorithm indeed finds a maximum cut. It is clear that the induced subgraph $G[B]$ is bipartite. Therefore, any partition of $V \setminus B$ into two subsets can be extended to an optimal partition of V in polynomial time by Theorem 5.2. Clearly, by enumerating all partitions of $V \setminus B$ and then extending each in an optimal way, one finds a maximum cut in G.

The enumeration of partitions in `Step 3` is the bottleneck of the algorithm; it needs exponential time $O^*(2^{|V\setminus B|})$. Other steps can be performed in linear time. It is clear for `Steps 1` and `4`, and the linear time algorithm for `Step 2` is given in [L75]. Notice, that the algorithm can be implemented in such a way that each step uses only polynomial space. In particular, in `Step 3` we need to store only currently best solution.

Suppose that the input graph is isomorphic to the complete graph on $\Delta+1$ vertices. The number of partitions that are enumerated in `Step 3` is 2^{n-2} but since $\Delta = O(n)$ the claimed running time follows.

Now suppose that the input graph G is not isomorphic to the complete graph on $\Delta+1$ vertices. Then, by Brooks' Theorem G is Δ colorable ([L75]). Clearly, the union of two largest color classes has size at least $2n/\Delta$ and $|V \setminus B| \leq n(1-(2/\Delta))$. The number of partitions that are enumerated in `Step 3` is $O^*(2^{(1-(2/\Delta))n})$ and the claimed running time follows. □

5.7 Algorithm for general graphs

Let us notice that in the algorithm presented in the previous section, the assumption of bounded maximum degree is needed only to obtain an induced bipartite graph. Now we relax this assumption and study the complexity of the method in general graphs. Let us formalize that as Algorithm `B`. The input of `B` is a weighted graph $G = (V, E, w)$ and the output is a maximum cut in G with respect to the weight function w.

`Step 1.` Find a maximal independent set I_0 in G.

`Step 2.` Find a maximal independent set I_1 in $G[V \setminus I_0]$. Let B be the union of I_0 and I_1.

`Step 3.` Enumerate all partitions of elements of $V \setminus B$ into two subsets (all $0-1$ assignments) and for each find an optimal extension of the partial partition.

`Step 4.` Find a cut C that has the largest weight among all checked in `Step 3`. Return the cut C.

To complete the description of the algorithm, we need to provide a procedure that

finds an induced bipartite subgraph in `Steps 1` and `2`.

From Turan's theorem follows that the size of a maximum independent set is at least $n/(d+1)$, and as shown in [STY03], there is a linear-time algorithm that constructs an independent set of at least that size. As the time complexity of `B` depends on $|B|$, we need to give a lower bound on the size of the bipartite subgraph B.

Claim 5.4. *The set B of vertices constructed in `Step 2` of Algorithm `B` has size at least $2/(d+2)$.*

Proof. Let $i = |I_0|$ and m' be the number of edges of the subgraph $G[I_0 \cup I_1]$. If $i \geq 2n/(d+2)$, then $|B| \geq 2n/(d+2)$ and the claim follows. Suppose $i < 2n/(d+2)$. The average degree d' in the graph $G[V-I_0]$ is $d' = 2(m-m')/(n-i)$. Notice that $m' \geq n-i$, since I_0 is an independent set. Hence, $d' \leq 2n/(n-i) - 2$ and since $i < 2n/(d+2)$, we have $d' < d$. It follows that $|B| = i + (n-i)/(d'+1) \geq 2n/(d+2)$. □

Having established the lower-bound on the size of B, we can claim the running time of Algorithm `B`. Notice that $2/(d+2) = n/(n+m)$ and $n - |B| \leq mn/(m+n)$.

Theorem 5.5. *Algorithm `B` computes* MAX-CUT *in a graph G with n vertices and m edges in time $O^*(2^{mn/(m+n)})$, and polynomial space.*

The proof of this theorem is similar to the proof of Theorem 5.3 and will be omitted.

Chapter 6

Max-Cut and Max-Bisection are NP-hard on unit disk graphs

In this section we prove that the MAX-CUT and MAX-BISECTION problems are NP-hard on unit disk graphs. We also show that λ-precision graphs are planar for $\lambda > 1/\sqrt{2}$ and give a dichotomy theorem for the computational complexity of MAX-CUT on λ-precision unit disk graphs. The results presented here are based on the paper [DK07].

6.1 Introduction

Unit disk graphs are intersection graphs of unit diameter disks in the plane. Place n disks of diameter one in the plane so that the centers of disks do not coincide. An undirected graph is said to be a *unit disk graph* if there exists a one-to-one correspondence between its vertices and disks in such a way that two vertices are adjacent if and only if the corresponding disks intersect. (We assume tangent disks do intersect, however the classes of unit disk graphs with open or closed disks coincide [LL06].) Each configuration of disks that defines a unit disk graph is called its *intersection model*. This can be easily translated into a *proximity model* which is a collection of distinct points on the plane in one-to-one correspondence with the vertices of the graph in such a way that two vertices are adjacent if and only if two points are at distance at most one. Notice that unit disk graphs are simple and loopless. Often it is convenient to identify points in the proximity model and vertices of the graph and we will do so.

In recent years there has been an increasing interest in the study of unit disk graphs and their randomized version, the random geometric graphs, due to their use as models of wireless communication networks (see for example [ASSC02]).

Sometimes an additional assumption on the proximity model is imposed and points

are required to be at distance greater or equal λ from each other. Graphs which have such a proximity model are called λ-*precision* unit disk graphs. Notice that the class of λ_1-precision unit disk graphs is contained in the class of λ_2-precision unit disk graphs, for every $\lambda_1 \geq \lambda_2$.

In [CC90], many algorithmic problems – CHROMATIC-NUMBER, MAXIMUM-INDEPENDENT-SET, MINIMUM-DOMINATING-SET – have been proved to be NP-hard on unit disk graphs. The authors of that paper also provide a polynomial-time algorithm for MAX-CLIQUE. They conclude the paper pointing out that the computational complexity of all considered problems is the same for planar and unit disk graphs.

The computational complexity of the MAX-CUT and MAX-BISECTION problem on unit disk graphs has not been known, even though several authors offered approximation algorithms for MAX-CUT and MAX-BISECTION. In [HMRRRS98], among other approximation results, the authors present a polynomial-time approximation scheme for λ-precision unit disk graphs. A polynomial-time approximation scheme for the MAX-BISECTION problem on unit disk graphs was developed in [JKLS05]. It is also worth mentioning that the problem of recognizing unit disk graphs is NP-hard ([BK98]) so approximation algorithms require providing an intersection model for the input graph.

In this section, we prove that the MAX-CUT and MAX-BISECTION problems are NP-hard for unit disk graphs, which to the knowledge of the authors were open problems (see [S03, JKLS05]). Also, it turns out that MAX-CUT is the first problem known whose computational complexities on planar and unit disk graphs do not coincide. In the last section, we show that the λ-precision unit disk graphs are planar for $\lambda > 1/\sqrt{2}$. An interesting open problem is to investigate the computational complexity of MIN-BISECTION on unit disk graphs ([DPPS01]).

6.2 Mesh drawings

A *drawing of a graph* $G = (V, E)$ is a mapping f which assigns to each vertex of G a distinct point in the plane and to each edge uv a continuous arc between $f(u)$ and $f(v)$, not passing through the image of any other vertex. We also allow interiors of images of

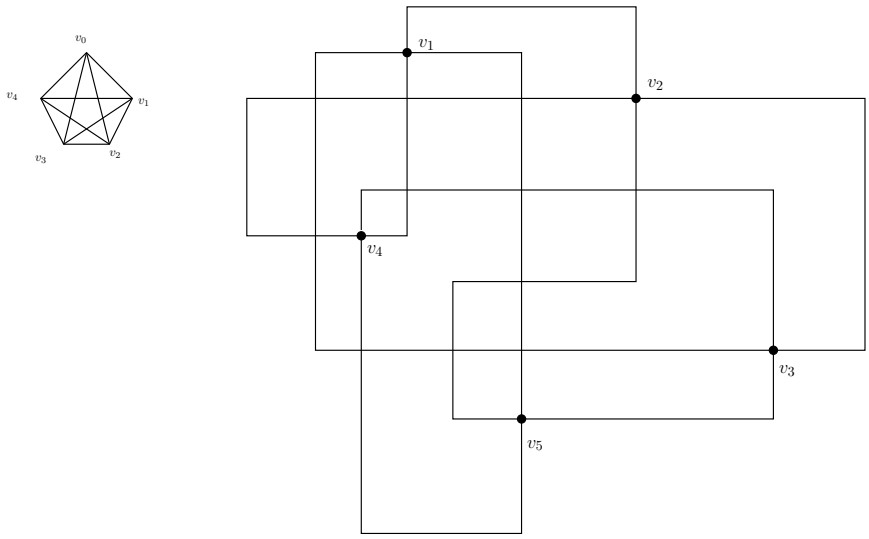

Figure 6.1: K_5 and its mesh drawing.

two different edges to intersect only at a finite number of points. Each such intersection is called a *crossing point*. A graph which can be drawn in the plane without any crossing points is called a *planar graph*. Below, if it does not lead to misunderstanding, we often do not distinguish between vertex/edge and its image.

The *mesh* $\mathcal{M}$ is the set of points in the plane which have at least one integral coordinate. These points of the mesh that have two integral coordinates are called *mesh crosses*. The distance between two points of $v = (x_1, y_1)$ and $w = (x_2, y_2)$ in $\mathcal{M}$ is the Manhattan distance $d(v, w) = \{|x_1 - x_2| + |y_1 - y_2|\}$.

A *mesh drawing* of a graph is a drawing in which the images of all vertices are mesh crosses and the images of edges belong to the mesh. Notice that in a mesh drawing only two edges can intersect at a crossing point and a crossing point always occurs at a mesh cross.

A necessary condition for a graph to have a mesh drawing is to be of maximum degree 4. In fact that condition is also sufficient, as shown by the following argument. Let G be any graph of maximum degree 4. Place all vertices of G at distinct mesh crosses in such a way that the distance between any pair of vertices is at least 5. For a vertex of G whose corresponding mesh cross has coordinates (a, b) we define four "corridors":

$x = a - 1$ or $y = b + 2$; $x = a - 2$ or $y = b + 1$; $x = a + 1$ or $y = b - 2$; $x = a + 2$ or $y = b - 1$.

Then an edge between any pair of vertices in $\mathcal{M}$ can be drawn entirely within the corridors of its endpoints. Hence, we can construct a mesh drawing of G and the construction can be done in polynomial time.

Let us say that a mesh drawing is *standard* if the distance between any two crossing points is at least 10, the distance between any two vertices is at least 10, the distance between any vertex and any crossing point is at least 10, and if interior points of two edges belong to two different parallel vertical or horizontal lines, the distance between these two lines is at least 10.

Given a mesh drawing of a graph, we can create its standard mesh drawing. Suppose that the distance between two vertices v, w is less than 10. Pick a line which separates these two vertices (i.e. at most one of v, w belongs to the line). Without loss of generality, we can assume it is a horizontal line $y = k$ for some $k \in \mathbb{Z}$ and v lies above that line. Now all the points of the drawing whose y-coordinate is at most k will be moved down by some constant c. The constant should be such that after w is moved down by c the distance between v and w is at least 10. All the edges that were broken by that operation should be extended (by adding a vertical segment of length c) in a way that makes a new graph isomorphic to the original one. This way we decreased the number of pairs of vertices that were at distance at most 10 from each other. Similar techniques can be used to satisfy all the conditions of the standard mesh drawing. Moreover, given a graph G, its standard mesh drawing can be found in polynomial time.

Lemma 6.1. *Every graph of maximum degree 4 has a standard mesh drawing and the drawing can be found in time polynomial in the number of vertices of the graph.*

Before we proceed to the next section, we want to note a useful fact. Given a graph G, let G' be the graph obtained from G by subdividing one of the edges of G twice. Then,

$$mc(G') = mc(G) + 2. \tag{6.1}$$

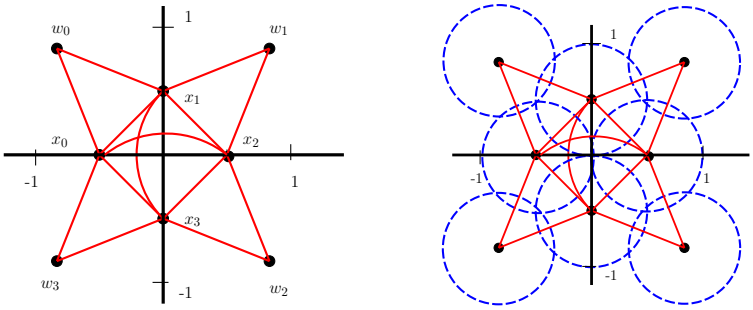

Figure 6.2: Gadget H and its intersection model.

6.3 Reduction

In this section we give a polynomial-time reduction from the class of graphs with maximum degree $\Delta \leq 4$ to the class of unit disk graphs. First we define a gadget that will be used in the reduction.

Let H be the graph on vertices: $x_0, x_1, x_2, x_3, w_0, w_1, w_2, w_3$ such that $H[x_0, x_1, x_2, x_3]$ is a K_4, $H[w_i, x_i, x_{i+1}]$ is a K_3, for $i = 0, 1, 2, 3$, where $i + 1$ is taken modulo 4, and H has no other edges than those needed to satisfy the two previous conditions.

Lemma 6.2. *H is a $(1/\sqrt{2})$-precision unit disk graph.*

Proof. Let us consider the following proximity model for H. Place vertices x_0, x_1, x_2, x_3 at points $(1/2, 0)$, $(0, 1/2)$, $(-1/2, 0)$ and $(0, -1/2)$, respectively. Vertices w_0, w_1, w_2, w_3 should be put at points $(4/5, 4/5)$, $(-4/5, 4/5)$, $(-4/5, -4/5)$ and $(4/5, -4/5)$, respectively. Now it is easy to verify that this is indeed a proximity model of H and all distances are at least $1/\sqrt{2}$. □

Given a graph G and the gadget H just defined, let (x_0x_2), (x_1x_3) be two edges in G not incident to each other. We define the graph G^* to be the graph obtained from G by adding the new vertices w_0, w_1, w_2, w_3 and edges necessary to make $G^*[x_0, x_1, x_2, x_3, w_0, w_1, w_2, w_3]$ isomorphic to H. We say that *H was constructed in G* on (x_0x_2), (x_1x_3).

The following lemma is straightforward by considering that the contribution of each triangle w_i, v_i, v_{i+1} to the maximum cut in G^* is exactly 2.

Lemma 6.3. *Let G^* be the graph obtained from G by constructing H on two non-incident edges of G. Then, $mc(G^*) = mc(G) + 8$*

Let us state and prove the main theorem.

Theorem 6.4. MAX-CUT *is NP-hard on unit disk graphs.*

Proof. To prove that MAX-CUT is NP-hard on unit disk graphs we are going to present a polynomial-time procedure that takes an arbitrary graph G of maximum degree 4 and produces a unit disk graph G'. Moreover, knowing $mc(G')$ we are able compute $mc(G)$ in polynomial time.

STEP 1 Let G be a graph of maximum degree 4. Let us consider its standard mesh drawing. According to Lemma 6.1, it does exist and can be found in polynomial time.

STEP 2 Subdivide edges of G putting new vertices at mesh crosses that are not crossing points. Now we have two types of vertices – those that were created in this step and those that correspond to original vertices of G.

STEP 3 For each crossing point (x, y), subdivide only the edge between $(x, y-1)$ and $(x, y+1)$ (the vertical one). We remove two vertices placed at $(x-1, y)$ and $(x+1, y)$ and consequently all the edges they were incident to. We place new vertices at coordinates $(x-1.5, y+0.5)$, $(x-0.5, y+0.5)$, $(x+0.5, y+0.5)$, $(x+1.5, y+0.5)$ and create edges between pairs of vertices: $(x-2, y)$ and $(x-1.5, y+0.5)$, $(x-1.5, y+0.5)$ and $(x-0.5, y+0.5)$, $(x-0.5, y+0.5)$ and $(x+0.5, y+0.5)$, $(x+0.5, y+0.5)$ and $(x+1.5, y+0.5)$, $(x+1.5, y+0.5)$ and $(x+2, y)$. Notice that the drawing is not a mesh drawing anymore.

STEP 4 For each crossing point, construct a copy of graph H on the crossing edges. Place new vertices in the way described in the proof of Lemma 6.2 and create straight line edges in each copy of H.

STEP 5 If along one of the original edges of G there is an odd number of vertices, we need to subdivide one of the new edges once more. Pick an edge between points (x, y)

and $(x+1, y)$ whose both endpoints and the neighbors of the endpoints are of degree at most 2 and the endpoints belong to the original edge of G. Move vertex at (x, y) to a new position $(x - 1/4, y)$ and vertex at $(x + 1, y)$ to $(x + 5/4, y)$. Create a new vertex at $(x + 1/2, y + 1/2)$ and edges between pairs of vertices: $(x - 1/4, y)$ and $(x + 1/2, y)$, $(x + 5/4, y)$ and $(x + 1/2, y)$.

Let $U(G)$ be the graph whose drawing was constructed above. We will show that the graph is in fact a unit disk graph.

Claim 6.5. *$U(G)$ is a $(1/\sqrt{2})$-precision unit disk graph.*

Proof of Claim. To prove that $U(G)$ is a unit disk graph we will show that placing vertices in the plane at the same coordinates as in the construction above gives a proximity model of $U(G)$.

Notice that after Step 2 all adjacent vertices, except the endpoints of crossing edges, are at distance exactly 1. Once crossing edges have been replaced by a construction described in Step 3, all the neighbors of a vertex are within distance 1 from it.

Notice that after constructing a copy of H at a crossing point (Step 4), "w" vertices are farther than a unit distance from all the vertices that they are not adjacent to. Also, the new edges created between endpoints belonging to two crossing edges are now connected. This together with Lemma 6.2 yields that the graph obtained after Step 4 is a unit disk graph.

Observe that if we start with a standard drawing, an edge whose both endpoints and the neighbors of the endpoints are of degree at most 2 can always be found. It is easy to see, that the construction described in Step 5 leaves a unit disk graph. □

Let k be the number of crossing points in the standard mesh drawing of G from Step 1, and let t be the total number of subdivisions of all the original edges of G created at the end of executing Step 5. Notice that t must be even.

Claim 6.6. $mc(U(G)) = mc(G) + 8k + t$

Proof of Claim. A copy of H is constructed on each pair of crossing edges and each copy of H increases the value of the maximum cut by 8 (Lemma 6.3). Also, each double

subdivision of an edge increases the value of maximum cut by 2 (Equation 6.1). Hence, $mc(U(G)) = mc(G) + 8k + t$. □

We have shown that any graph G of maximum degree 4 can be transformed in a unit disk graph G', and this reduction can be implemented in polynomial time. If there exist a polynomial-time algorithm solving MAX-CUT on unit disk graphs, then knowing the construction and $mc(G')$, the value of maximum cut of G' can be also computed in polynomial time. But as mentioned in section 1, MAX-CUT is NP-hard on graphs with maximum degree Δ, if $\Delta \geq 3$, and therefore it is also NP-hard on unit disk graphs. □

Taking two disjoint copies of a unit disk graph G creates a unit disk graph whose maximum bisection is twice the value of maximum cut of G. The following fact is therefore a simple corollary of Theorem 6.4.

Fact 6.7. MAX-BISECTION *is NP-hard on unit disk graphs.*

6.4 Precision and planarity

In this section we study the relation between λ-precision and planarity.

Theorem 6.8. *A λ-precision unit disk graph is planar, for every $\lambda > 1/\sqrt{2}$.*

Proof. Let G be a λ-precision unit disk graph, for some $\lambda > 1/\sqrt{2}$. Consider a drawing of G defined by its proximity model: put vertices in the plane at the same positions as in the proximity model and connect two of them by a straight line segment if and only if the distance between them is at most 1.

Let us consider an edge e of G of length x. The set of points at distance more than $1/\sqrt{2}$ from both endpoints of e consists of two disjoint regions; one on each side of the straight line containing e. It is easy to verify that the distance between these two regions is strictly greater than $2\sqrt{1/2 - (x/2)^2}$ and therefore always strictly greater than 1. Hence, there is no edge crossing e and the planarity of G follows. □

Finally, we can state a dichotomy result on MAX-CUT for λ-precision unit disk graphs which is a consequence of Theorem 4 and Theorem 8.

Theorem 6.9. *The* MAX-CUT *problem is NP-hard in the class of λ-precision unit disk graphs if $\lambda \leq 1/\sqrt{2}$ and can be solved in polynomial time if $\lambda > 1/\sqrt{2}$*

Proof. Notice $U(G)$ constructed in the proof of Theorem 6.4 is a $(1/\sqrt{2})$-precision unit disk. Hence, MAX-CUT is NP-hard in the class of λ-precision unit disk graphs for $\lambda \leq 1/\sqrt{2}$.

Since λ-precision unit disk graphs are planar for $\lambda > 1/\sqrt{2}$ and, as we mentioned, MAX-CUT can be solved in polynomial time on planar graphs, then the MAX-CUT problem can be solved on these graphs in polynomial time. □

References

[ASSC02] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, Wireless sensor networks: a survey, *Computer Networks*, 38, 393 – 422, 2002

[B41] R. L. Brooks, On Colouring the Nodes of a Network, *Proc. Cambridge Philos. Soc.* 37, 194-197, 1941

[B90] H. L. Bodlaender, Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, *J. Algorithms* 11, 631–643, 1990

[BJ00] H. L. Bodlaender and K. Jansen, On the complexity of the maximum cut problem, *Nordic Journal of Computing*, 7(1), 14–31, 2000

[BK98] H. Breu and D.G. Kirkpatrick, Unit disk recognition is NP-hard, *Computational Geometry*, 9, 3–24, 1998

[BKW99] O.V. Borodin and A.V. Kostochka and D.R. Woodall, Acyclic colourings of planar graphs with large girth, *J. London Math. Soc. (2)* 60, 344–352, 1999

[BT90] G. Bascó and Zs. Tuza, A characterization of graphs without long induced paths, *Journal of Graph Theory*, 14(4), 155 – 464, 1990

[BT90a] G. Bascó and Zs. Tuza, Dominating cliques in P_5-free graphs, *Periodica Mathematica Hungarica*, 21(4), 303 – 308, 1990

[BT93] G. Bascó and Zs. Tuza, Dominating properties and induced subgraphs, *Discrete Mathematics*, 111, 37 – 40, 1993

[BT97] H. L. Bodlaender and D. M. Thilikos, Treewidth for graphs with small chordality, *Discrete Applied Mathematics*, 79, 45–61, 1997.

[BT02] G. Bascó and Zs. Tuza, Structural domination of graphs, *Ars Combinatoria*, 63, 235 – 256, 2002

[BW01] B. Baetz and D. R. Wood, Brooks' Vertex-Colouring Theorem in Linear Time, *Technical Report CSAAG -2001-05*, Basser Department of Computer Science, The University of Sydney, 2001

[C89] K. Cameron, Induced matchings, *Discrete Appl. Math.* 24, 97–102, 1999

[CC90] N.B. Clark, C.J. Colbourn, and D.S. Johnson, Unit disk graphs, *Discrete Mathematics*, 86, 165–177, 1990

[CK90] M. B. Cozzens and L. L. Kelleher, Dominating cliques in graphs, *Discrete Math.* 86, 101–116, 1990

[CMR00] B. Courcelle, J.A. Makowsky and U. Rotics, Linear time solvable optimization problems on graphs of bounded clique-width, *Theory Comput. Systems*, 33, 125–150, 2000

[CPS84] D. G. Corneil, Y. Perl and L.K. Stewart, Cographs: recogniton, applications, and algorithms, *Congressus Numer.* 43, 249 – 258, 1984

[D05] R. Diestel, Graph Theory, 3rd Edition, *Springer-Verlag*, 2005

[DK07] J. Díaz and M. Kamiński, Max-Cut and Max-Bisection are NP-hard on unit disk graphs, *Theoretical Computer Science*, to appear (http://dx.doi.org/10.1016/j.tcs.2007.02.013)

[DKP07] F. Della Croce, M. Kamiński and V. Paschos, An exact algorithm for MAX-CUT in sparse graphs, *Operations Research Letters*, to appear (http://dx.doi.org/10.1016/j.orl.2006.04.001)

[DPPS01] J. Díaz, M. Penrose, J. Petit, and M. Serna, Approximating layout problems on random geometric graphs, *Journal of Algorithms*, 39, 78–116, 2001

[E59] P. Erdös, Graph theory and probability, *Canad. J. Math.* 11, 34–38, 1959

[FK06] S. S. Fedin and A. S. Kulikov, A $2^{|E|/4}$-time algorithm for max-cut, *Journal of Mathematical Sciences*, to appear

[G04] Martin Golumbic, Algorithmic Graph Theory and Perfect Graphs, *North Holland*, 2004

[GHNR03] J. Gramm, E. A. Hirsch, R. Niedermaier and P. Rossmanith, Worst-case upper bounds for Max2Sat with an application to MaxCut, *Discrete Appl. Math.*, 130, 139–155, 2003

[GJ79] M. R. Garey, D. S. Johnson, Computers and Intractability, W. H. Freeman, San Francisco, 1979

[GLS84] M. Grötschel, L. Lovász, A. Schrijver, Polynomial algorithms for perfect graphs, *Ann. Discrete Math.* 21, 325–356, 1984

[H69] F. Harary, Graph Theory, Addison-Wesley, Reading, MA, 1969

[H75] F. Hadlock, Finding a maximum cut of a planar graph in polynomial time, *SIAM J. Comput.* 4(3), 221–225, 1975

[H81] Ian Holyer, The NP-completness of edge-coloring, *SIAM J. Computing* 10, 718–720, 1981

[HKLSS06] Ch. Hoàng, M. Kamiński, V. Lozin, J. Sawada and X. Shu, Deciding k-colourability of P_5-free graphs in polynomial time, *submitted*

[HMRRRS98] H.B. Hunt, M.V. Marathe, V. Radhakrishnan, S. Ravi, D. Rosenkratz and R.E. Stearns, NC-approximation schemes for NP- and PSPACE-Hard problems for geometric graphs, *J. Algorithms*, 26, 238–274, 1998

[HSW06] C. Hoang, J. Sawada, Z. Wang, Colorability of P_5-free graphs (http://www.cis.uoguelph.ca/ sawada/pub.html)

[I84] W. Imrich, Explicit construction of regular graphs without small cycles, *Combinatorica* 4, 53–59, 1984

[K72] R.M. Karp, Reducibility among combinatorial problems, In *Complexity of Computer Computations*, Plenum Press, 85–104, 1972

[K05] M. Kochol, 3-coloring and 3-clique-ordering of locally connected graphs, *Journal of Algorithms*, 54, 122–125, 2005

[KKTW01] D. Kral, J. Kratochvil, Z. Tuza and G. J. Woeginger, Complexity of colouring graphs without forbidden induced subgraphs, In *WG* 2001, volume 2204 of *Lecture Notes in Computer Science*, 54–262, 2001

[KL06] M. Kamiński, V. Lozin, Coloring vertices of P_5-free graphs in polynomial time, *Rutcor Research Report*, RRR 21 – 2006, September 2006

[KL07] M. Kamiński, V. Lozin, Coloring edges and vertices of graphs without short or long cycles, *Contributions to Discrete Mathematics*, 2, 61 – 66, 2007

[KL07a] M. Kamiński, V. Lozin, On 3-colorability of claw-free graphs, *Algorithmic Operations Research*, 2, 15 – 21, 2007

[KLB03] M. Kochol, V. Lozin, B. Randerath, The 3-colorability problem on graphs with maximum degree 4, *SIAM J. Computing* 32, 1128–1139, 2003

[KR05] J. Kneis and P. Rossmanith, A new satisfiablity algorithm with applications to max-cut, *Technical Report AIB-2005-08*, Department of Computer Science , RWTH Aachen

[JKLS05] K. Jansen, M. Karpinski, A. Lingas, and E. Seidel, Polynomial-time approximation schemes for Max-bisection on planar and geometric graphs, *SIAM J. Computing*, 35(1), 110–119, 2005

[L75] L. Lovász, Three short proofs in graph theory, *Journal of Combinatorial Theory* (B), 19, 269–271, 1975

[LL03] X. Li and R. Luo, Edge coloring of embedded graphs with large girth, *Graphs Combin.* 19, 393–401, 2003

[LL06] E. J. van Leeuwen and J. van Leeuwen, On the representation of disk graphs, *Technical Report UU-CS-2006-037*, Utrecht University, 2006

[LR04] V. Lozin and D. Rautenbach, On the band-, tree- and clique-width of graphs with bounded vertex degree, *SIAM J. Discrete Mathematics*, 18, 195–206, 2004

[LRS06] V. Bang Le, B. Randerath and I. Schiermeyer, Two remarks on coloring graphs without long induced paths, in Report No. 7/2006 (Algorithmic Graph Theory), Mathematisches Forschungsinstitut Oberwolfach

[M92] O.J. Murphy, Computing independent sets in graphs with large girth, *Discrete Appl. Math.* 35, 167–170, 1992

[MP96] F. Maffray and M. Preissmann, On the NP-completeness of the k-colorability problem for triangle-free graphs, *Discrete Mathematics* 162, 313–317, 1996

[O72] G.I. Orlova and Y.G. Dorfman, Finding the maximal cut in a graph, *Engineering Cybernetics*, 10, 502–506, 1972

[PT95] S. Poljak and Z. Tuza, Maximum cuts and large bipartite subgraphs, In *Combinatorial Optimization. Papers from the DIMACS Special Year*, AMS, 181–224, 1995

[R04] B. Randerath, 3-colorability and forbidden subgraphs. I. Characterizing pairs, *Discrete Mathematics* 276, 313–325, 2004

[RS04] B. Randerath and I. Schiermeyer, Vertex coloring and forbidden subgraphs – a survey, *G*raphs and Combinatorics, 20(1), 1 – 40, 2004

[RS04a] B. Randerath and I. Schiermeyer, 3-colorability $\in \mathcal{P}$ for P_6-free graphs, *Discrete Applied Mathematics* 136, 299 – 313, 2004

[RST02] B. Randerath, I. Schiermeyer and M. Tewes, Three-colourability and forbidden subgraphs. II: polynomial algorithms, *Discrete Mathematics* 251, 137 – 153, 2002

[S99] A. Schrijver, Theory of linear and integer programming, Wiley, 1999

[S03] J. Spinrad, Efficient graphs representations, Fields Institute Monographs 19, AMS, Providence, 2003

[SS03] A. D. Scott and G. B. Sorkin, Faster algorithms for MAX CUT and MAX CSP with polynomial expected time for sparse instances, In *Proc. RANDOM'03*, volume 2764 of *Lecture Notes in Computer Science*, Springer-Verlag, 382–395, 2003

[SS04] A. D. Scott and G. B. Sorkin, Solving sparse semi-random instances of Max-Cut and Max-CSP in linear expected time, Research Report 23417 (W0411-056), IBM Research division, Thomas J. Watson Research Center, 2004

[STY03] S. Sakai, M. Togasaki, and K. Yamazaki, A note on greedy algorithms for the maximum weighted independent set problem, *Discrete Appl. Math.*, 126, 313–322, 2003

[SV82] L.J. Stockmeyer and V.V. Vazirani, NP-completeness of some generalizations of the maximum matching problem, *Inform. Process. Lett.* 15, 14–19, 1982

[SW01] J. Sgall and G. J. Woeginger, The complexity of coloring graphs without long induced paths, *Acta Cybernetica* 15(1), 107–117, 2001

[TP97] J.A. Telle and A. Proskurowski, Algorithms for vertex partitioning problems on partial k-trees, *SIAM J. Discrete Math.* 10, 529–550, 1997

[W04] G. J. Woeginger, Space and time complexity of exact algorithms: some open problems, In *Proc. IWPEC'04*, volume 3162 of *Lecture Notes in Computer Science*, Springer-Verlag, 281–290, 2004

[W05] R. Williams, A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348 (2–3), 357–365, 2005

[Y78] M. Yannakakis, Node- and edge-deletion NP-complete problems, In *IEEE Symposium on Theory of Computing (STOC)*, 253–264, 1978

[ZNN96] X. Zhou, S.-I. Nakano and T. Nishizeki, Edge-coloring partial k-trees, *J. Algorithms* 21, 598–617, 1996

Vita

Marcin Jakub Kamiński

May 2007	Ph.D. in Operations Research, Rutgers University, New Brunswick, NJ, USA
2002 – 2007	Teaching Assistant and Instructor, Department of Mathematics, Rutgers University, New Brunswick, NJ, USA
2002 – 2007	Ph.D. Student, RUTCOR, Rutgers University, New Brunswick, NJ, USA
2002	Software Developer, SynaptiCAD, Inc., Blacksburg, VA, USA
June 2001	M.Sc. in Computer Science, Department of Computer Science, Gdańsk University of Technology, Gdańsk, Poland

Publications

2006	J. Díaz, M. Kamiński, *Max-Cut and Max-Bisection are NP-hard on unit disk graphs*, Theoretical Computer Science, to appear
	J. Díaz, M. Kamiński, D. Thilikos, *Recognizing partial* $2 \times \infty$ *grids in linear time*, submitted
	Ch. Hoàng, M. Kamiński, V. Lozin, J. Sawada, X. Shu, *Deciding* k*-colourability of* P_5*-free graphs in polynomial time*, submitted
	D. Cardoso, M. Kamiński, V. Lozin, *Maximum* k*-regular induced subgraphs*, Journal of Combinatorial Optimization, to appear
	M. Kamiński, V. Lozin, M. Milanič, *Recent developments on graphs of bounded clique-width*, Discrete Applied Mathematics, accepted
2005	M. Kamiński, V. Lozin, *Coloring edges and vertices of graphs without short or long cycles*, Contributions to Discrete Mathematics, Volume 2 (2007), pp. 61 – 66
	F. Della Croce, M. Kamiński, V. Paschos, *An exact algorithm for the Max-Cut problem in sparse graphs*, Operations Research Letters, to appear

M. Kamiński, V. Lozin, *On 3-colorability of claw-free graphs*, Algorithmic Operations Research, Volume 2 (2007), pp. 15 – 21

2003 K. Giaro, M. Kamiński, *Introduction to quantum computing* (in Polish), Warsaw, EXIT 2003 (a survey book based on my master's thesis)