ALGORITHMIC DEVELOPMENTS AND COMPLEXITY RESULTS FOR FINDING MAXIMUM AND EXACT INDEPENDENT SETS IN GRAPHS

BY MARTIN MILANIČ

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 V. Lozin

and approved by

_____

_____

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2007

ABSTRACT OF THE DISSERTATION

Algorithmic Developments and Complexity Results for Finding Maximum and Exact Independent Sets in Graphs

by Martin Milanič

Dissertation Director: Professor Vadim V. Lozin

We consider the MAXIMUM INDEPENDENT SET and MAXIMUM WEIGHT INDEPENDENT SET problems in graphs. As these problems are generally NP-hard, we study their complexity in hereditary graph classes, that is, in graph classes defined by a set $\mathcal{F}$ of forbidden induced subgraphs.

We describe a condition on the set $\mathcal{F}$, which guarantees that the MAXIMUM INDEPENDENT SET problem remains NP-hard in the class of $\mathcal{F}$-free graphs. The same hardness result remains valid even under the additional restriction that the graphs are planar and of maximum degree at most three.

We exhibit several cases where the condition is violated, and the problem admits a polynomial-time solution. More specifically, we present polynomial-time algorithms for the MAXIMUM INDEPENDENT SET problem in:

- two graph classes that properly contain claw-free graphs (thus generalizing the classical result for claw-free graphs);
- various subclasses of planar and more general graphs;
- weighted graphs in certain subclasses of graphs of bounded vertex degree.

Our algorithms are based on various approaches. In particular, we develop an extension of the method of finding augmenting graphs. We also make extensive use of some other well-known graph decompositions.

We also introduce and study the exact version of the problem, where, instead of finding an independent set of maximum weight, the goal is to find an independent set of given weight. Determining the computational complexity of this problem for line graphs, or for line graphs of bipartite graphs would resolve long standing open problems. Here, we show that:

- The EXACT WEIGHTED INDEPENDENT SET problem is strongly NP-complete for cubic bipartite graphs.
- The problem is solvable in pseudo-polynomial time for any of the following graph classes:
  mK_2-free graphs, interval graphs and their generalizations k-thin graphs, circle graphs, chordal graphs, AT-free graphs, (claw, net)-free graphs, distance-hereditary graphs, and graphs of bounded tree- or clique-width.

Finally, we show how modular decomposition can be applied to the EXACT WEIGHTED INDEPENDENT set problem. As a corollary, we obtain pseudo-polynomial solutions for the problem in certain subclasses of P_5-free and fork-free graphs.

Acknowledgements

I am deeply indebted to my advisor, Professor Vadim V. Lozin, in numerous ways. He was the person who introduced me to mathematical research. He was an enormous source of ideas, inspiration, encouragement and motivation. This dissertation would certainly not be possible without his great guidance and support.

I would also like to thank my friends at Rutgers, in particular Noam Goldberg, Marcin Kamiński, Tobias Kuna, Dávid Papp, Anupama Reddy, Gábor Rudolf. With them, I have enjoyed many pleasant moments, which will help me remember my stay at Rutgers as a very nice experience.

I thank Professor Peter L. Hammer[1], for having shared his always stimulating research questions.

I would like to thank Professor Jeffry Kahn for being such a great instructor. Thanks to his courses at the Department of Mathematics, I gained knowledge in graph theory and combinatorics, as well as in more advanced topics.

A word of thank definitely goes to the secretaries. Lynn Agre, Terry Hart, Clare Smietana were always there, available and ready to help.

I thank Professors Farid Alizadeh, Endre Boros, Vladimir Gurvich, Jeffry Kahn and Jérôme Monnot for serving on my dissertation committee.

Finally, special thanks go to my parents, for their immense support and understanding during these – for them long – years of my studies abroad.

The results of this thesis are based on joint work with Marcin Kamiński, Vadim V. Lozin, and Jérôme Monnot.

[1]Peter L. Hammer (1936-2006) passed away prematurely on December 27, 2006.

Dedication

To my parents

Table of Contents

Chapter 1

Introduction

Graphs are mathematical structures that model binary relations. As such, they are extremely useful to model problems from such varied areas as computer science, operations research, social networks, biology, etc. In these problems, it is often desirable to find a particular substructure of a graph that is optimal with respect to some optimality criterion. Most of these *graph optimization problems* can be described within the following framework. Given a graph, find a subset of its vertices (or edges) of minimum or maximum cardinality, subject to additional constraints (requirements). For example, the MAXIMUM INDEPENDENT SET problem asks for a maximum subset of pairwise non-adjacent ("independent") vertices.

Another important framework is given by the *exact versions* of graph optimization problems: Given a graph with weights on its vertices (or edges), find a subset of its vertices (or edges) whose total weight equals to some given bound (or determine that there is no such subset, if this is the case). For example, the EXACT PERFECT MATCHING problem asks whether a given edge-weighted graph contains a perfect matching of given weight.

Most of such problems are computationally intractable; they are among the many *NP-hard problems* [55]. There are several ways of dealing with an NP-hard problem. Since solving an optimization problem to optimality may be prohibitive in terms of computational time, we can instead try to obtain an approximate solution in reasonable time: this brings us into the area of *approximation algorithms* [111]. Also, even though a problem may be hard in general, it is often possible to reveal *restrictions on the input instances* under which the problem can be solved efficiently, that is, in *polynomial time*. Of course, what's interesting here is to identify cases for which polynomial-time solutions require new algorithmic ideas.

In this thesis, we focus on four problems related to finding independent sets in graphs: the MAXIMUM INDEPENDENT SET problem (MIS), the MAXIMUM WEIGHT INDEPENDENT

SET problem (MWIS), the EXACT WEIGHTED INDEPENDENT SET problem (EWIS) and the EXACT WEIGHTED MAXIMUM INDEPENDENT SET problem (EWIS$_\alpha$).

1.1 Maximum Independent Sets

The MAXIMUM INDEPENDENT SET problem is a fundamental problem in algorithmic graph theory. Closely related to the MAXIMUM CLIQUE and MINIMUM VERTEX COVER problems, this is one of the central problems of combinatorial optimization and has numerous applications [18]. The problem is well-known to be NP-hard [55], and also hard to approximate [69].[1] The problem remains NP-hard even under substantial restrictions, for instance, for triangle-free graphs [100], $K_{1,4}$-free graphs [90], and planar graphs of degree at most three [54].

More generally, let X be the class of graphs defined by a set $\mathcal{F}$ of forbidden induced subgraphs. Can we draw the line between the polynomial and the NP-hard side, in terms of the set $\mathcal{F}$? That is, under what conditions on $\mathcal{F}$ does the MAXIMUM INDEPENDENT SET problem remain NP-hard in X? On the other hand, when does it become polynomially solvable?

A partial answer to these questions was given in 1982 by Alekseev [1], who proved the following theorem.

Theorem 1.1.1 ([1])**.** *Let X be the class of graphs defined by a set $\mathcal{F}$ of forbidden induced subgraphs. If $\mathcal{F}$ is finite and contains no graph whose each connected component is of the form $S_{i,j,k}$ (see Figure 1.1), then the* MAXIMUM INDEPENDENT SET *problem is NP-hard in X.*

Alekseev's result can only be applied to finite $\mathcal{F}$. After giving the necessary preliminaries, we show in section 1.3.3 that the result of Garey and Johnson [54] who showed NP-hardness of the MIS problem for *planar graphs of vertex degree at most three*, and Alekseev's result admit a common strengthening (Theorem 1.3.1). More specifically, we describe a condition

[1] As the reader will witness in Chapter 2, an indication of the inherent intractability of the problem is provided by the great variety of techniques that have been designed in order to develop polynomial-time solutions to the problem in particular graph classes.

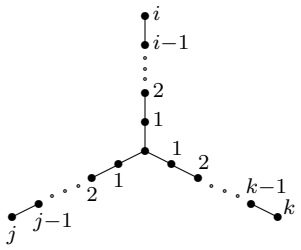

Figure 1.1: Graph $S_{i,j,k}$

on the set $\mathcal{F}$ (which can be infinite) such that the MAXIMUM INDEPENDENT SET problem remains NP-hard in the class of $\mathcal{F}$-free graphs that are planar and of maximum degree at most 3.

In Chapters 3 and 4 we exhibit several cases where this condition is violated, and the MAXIMUM INDEPENDENT SET problem or its weighted counterpart admit a polynomial-time solution. Our algorithms are based on various approaches. For example, we develop an extension of the method of finding augmenting graphs. We also make extensive use of the well-known graph decompositions such as modular decomposition and decomposition by clique separators, as well as other reductions, and combinations thereof. All these techniques are described in Chapter 2.

With the exception of one approximation algorithm with performance ratio 2, all of our algorithms are exact, that is, they compute optimal solutions. As these polynomial results will be best appreciated in view of the hardness theorem (Theorem 1.3.1 in Section 1.3.3), we postpone their detailed formulation until Section 1.4, where an overview of the main contributions of this thesis is given.

1.2 Perfect Matchings and Independent Sets of Exact Weight

Let us now turn our attention to the exact versions of graph optimization problems. Suppose we have a well-solved optimization problem, such as minimum spanning tree, maximum cut in planar graphs, minimum weight perfect matching, or maximum weight independent set in a bipartite graph. How hard is it to determine whether there exists a solution with a given weight?

As noted by Papadimitriou and Yannakakis in [99], many exact versions of polynomially solvable optimization problems are NP-complete when the weights are encoded in binary. The question is then, what happens if the weights are "small," that is, encoded in unary? Contrary to the binary case, the answer to this question depends on the problem.

- The EXACT SPANNING TREE problem, and more generally, the EXACT ARBORESCENCE problem are solvable in pseudo-polynomial time [10].

- The EXACT CUT problem is solvable in pseudo-polynomial time for planar and toroidal graphs [10].

- The EXACT PERFECT MATCHING problem is solvable in pseudo-polynomial time for planar graphs [10], and more generally, for graphs that have a Pfaffian orientation (provided one is given). Karzanov [75] gives a polynomial-time algorithm for the special case of the exact perfect matching problem, when the graph is either complete or complete bipartite, and the weights are restricted to 0 and 1. Papadimitriou and Yannakakis show in [99] that the problem for general (or bipartite) graphs with weights encoded in unary is polynomially reducible to the one with 0-1 weights. Mulmuley, Vazirani and Vazirani [93] show that the exact perfect matching problem has a randomized pseudo-polynomial-time algorithm.

The exact perfect matching problem is of great practical importance. It has applications in such diverse areas as bus-driver scheduling, statistical mechanics (see [76]), DNA sequencing [12], and robust assignment problems [37]. The problem consists in determining whether a given edge-weighted graph contains a perfect matching of a given weight. (A *matching* in a graph is a set of pairwise disjoint edges. A matching is *perfect* if every vertex of the graph is contained in some edge from the matching.) Despite polynomial results for special cases, the deterministic complexity of the exact perfect matching problem remains unsettled for general graphs, and even for bipartite graphs. Papadimitriou and Yannakakis conjectured that the problem is NP-complete [99].

This open problem motivates us to introduce and study the *exact weighted independent set* problem and a restricted version of it, both closely related to the exact perfect matching

problem. The EXACT WEIGHTED INDEPENDENT SET problem (EWIS) consists in determining whether a given vertex-weighted graph contains an independent set of a given weight. The EXACT WEIGHTED MAXIMUM INDEPENDENT SET problem (EWIS_α) is the restriction of EWIS where we require the independent set to be a maximum independent set of the graph.

The connection between the exact perfect matching problem and the exact weighted independent set problem is best understood through line graphs. The *line graph* $L(G)$ of a graph $G = (V, E)$ is the graph whose vertex set is E, and whose two vertices are adjacent if and only if they share a common vertex as edges of G. Clearly, there is a one-to-one correspondence between the matchings of a graph and the independent sets of its line graph. The EXACT MATCHING problem, that is, the problem of determining whether a given edge-weighted graph contains a matching of a given weight, is then precisely the exact weighted independent set problem, restricted to the class of line graphs. Similarly, under the (polynomially verifiable) assumption that the input graph has a perfect matching, the exact perfect matching problem is precisely the exact weighted maximum independent set problem, restricted to the class of the line graphs of graphs with a perfect matching.

Thus, in terms of independent sets, the above open problems translate to the following two.

Open Problem 1. *Determine the complexity of the* EXACT WEIGHTED MAXIMUM INDEPENDENT SET *problem in line graphs.*

Open Problem 2. *Determine the complexity of the* EXACT WEIGHTED MAXIMUM INDEPENDENT SET *problem in line graphs of bipartite graphs.*

Here, we do not answer these questions. However, we present in Chapter 5 several complexity results for the problems EWIS and EWIS_α, when restricted to particular graph classes. We also show how modular decomposition can be applied to the exact weighted independent set problem. Again, we refer to Section 1.4 for a more detailed formulation of the results.

The remainder of this introductory chapter is organized as follows. In Section 1.3, we

give necessary preliminaries: formal definitions of the problems, a short discussion on graph classes, and a general NP-hardness result for the maximum independent set problem in hereditary graph classes. Section 1.4 presents an overview of the main contributions of the thesis. We conclude the chapter with basic graph-theoretic definitions and terminology.

1.3 Preliminaries

1.3.1 Formal Definitions of Problems

An *independent set* (sometimes called *stable set*) in a graph G is a subset of pairwise non-adjacent vertices. The MAXIMUM INDEPENDENT SET problem is that of finding in a graph a *maximum independent set*, that is, an independent set of maximum cardinality (which is denoted by $\alpha(G)$ and referred to as the *independence number* of the graph). If each vertex of G is assigned a positive integer, the *weight* of the vertex, then we say that G is a *weighted graph*. The MAXIMUM WEIGHT INDEPENDENT SET problem consists in finding in a weighted graph an independent set of maximum total weight. Sometimes, we will refer to the maximum weight independent set problem as the "maximum independent set problem in weighted graphs."

The decision versions of these problems can be formally expressed as follows:

MAXIMUM INDEPENDENT SET (MIS)

Instance: A pair (G, k), where $G = (V, E)$ is a graph, k is an integer.

Question: Is there an independent set I in G such that $|I| \geq k$?

MAXIMUM WEIGHT INDEPENDENT SET (MWIS)

Instance: A triple (G, w, k), where $G = (V, E)$ is a graph, $w : V \to \mathbb{Z}$ and k is an integer.

Question: Is there an independent set I in G such that $\sum_{v \in I} w(v) \geq k$?

The EXACT WEIGHTED INDEPENDENT SET problem (EWIS) consists in determining whether a given weighted graph (G, w) with $G = (V, E)$ and $w : V \to \mathbb{Z}$ contains an independent set whose total weight (i.e., the sum of the weights of its members) equals a given integer b. The EXACT WEIGHTED MAXIMUM INDEPENDENT SET problem (EWIS_α) is the restriction of the EWIS problem where we require the independent set to be a maximum

independent set of the graph. Thus, given a weighted graph (G, w) and an integer b, the EWIS$_\alpha$ problem asks about the existence of an independent set I of G with $w(I) = b$ and $|I| = \alpha(G)$.

Formally:

Exact weighted independent set (EWIS)

Instance: A triple (G, w, b), where $G = (V, E)$ is a graph, $w : V \to \mathbb{Z}$ and b is an integer.

Question: Is there an independent set I in G such that $\sum_{v \in I} w(v) = b$?

Exact weighted maximum independent set (EWIS$_\alpha$)

Instance: A triple (G, w, b), where $G = (V, E)$ is a graph, $w : V \to \mathbb{Z}$ and b is an integer.

Question: Is there a maximum independent set I in G such that $\sum_{v \in I} w(v) = b$?

1.3.2 Graph Classes

We will restrict our attention to graph classes with the following nice property: whenever they contain a graph G, they contain all induced subgraphs of G. Such classes are called *hereditary*. Many classes of theoretical or practical importance are hereditary, which includes, among others,

1. *planar graphs*;
2. *bipartite graphs*;
3. *graphs of bounded vertex degree*;
4. *forests*, i.e., graphs without cycles;
5. *graphs of bounded treewidth*;
6. *graphs of bounded clique-width*;
7. *chordal graphs*, i.e., graphs in which every cycle of length at least four has a chord;
8. *perfect graphs*;
9. *interval graphs*, i.e., intersection graphs of intervals on a real line;
10. *circle graphs*, i.e., intersection graphs of chords on a circle;
11. *distance-hereditary graphs*;

12. *line graphs*, i.e., graphs G such that there is a graph H satisfying $G = L(H)$.

Graph classes 5, 6, and 11 from the list will be defined in Sections 2.5 and 5.3.1. For a comprehensive survey on graph classes, we refer the reader to [24].

A representative family of hereditary classes are those containing with every graph G all subgraphs of G (not necessarily induced). Such classes are called *monotone*. In the above list, only the first five classes are monotone. An important and well studied subfamily of monotone classes are *minor-closed* classes, i.e., those containing with every graph G all minors of G. Among classes listed above only 1, 4 and 5 are minor closed.

An important property of hereditary classes is that these and only these classes admit a uniform description in terms of forbidden induced subgraphs, which provides a systematic way to investigate various problems associated with graph classes. Let $\mathcal{F}$ be a set of graphs. If a graph G does not contain induced subgraphs from $\mathcal{F}$, we say that G is $\mathcal{F}$-free. The set of all $\mathcal{F}$-free graphs will be denoted by $Free(\mathcal{F})$. With this notation the above statement about induced subgraph characterization of hereditary classes can be reformulated as follows: a class of graphs X is hereditary if and only if $X = Free(\mathcal{F})$ for some set $\mathcal{F}$.

1.3.3 A Hardness Result

Let C_i and H_i denote the cycle of length i and the graph in Figure 1.2, respectively.

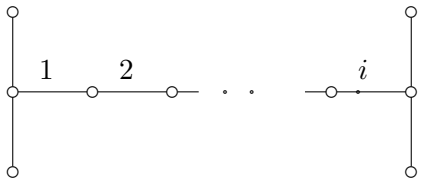

Figure 1.2: Graph H_i

We associate to every graph G a parameter $\kappa(G)$, which is the minimum value of $i \geq 1$ such that G contains an induced copy of either C_i or H_i. If G is an acyclic graph with no induced graphs of the form H_i, we let $\kappa(G) = \infty$. For a (possibly infinite) nonempty set of graphs $\mathcal{F}$, we define

$$\kappa(\mathcal{F}) = \sup \{ \kappa(G) \ : \ G \in \mathcal{F} \} .$$

Finally, for a set of graphs X, let X_3 denote the set of graphs of degree at most 3 in X.

With these definitions in mind, one can use the result of Garey and Johnson about the NP-hardness of the MIS problem in planar graphs of degree at most three [54], and the reduction typically used for the MIS problem (see e.g. [94, 100]), to derive the following hardness result.

Theorem 1.3.1 ([83])**.** *Let $\mathcal{F}$ be a set of graphs and let X be the class of $\mathcal{F}$-free planar graphs. If $\kappa(\mathcal{F}_3) < \infty$, then the* MAXIMUM INDEPENDENT SET *problem is NP-hard in X_3.*

Proof. For $k \geq 3$, let $\mathcal{S}_k$ be the class of all $(C_3, \ldots, C_k, H_1, \ldots, H_k)$-free planar graphs of vertex degree at most 3. To prove the theorem, we first show that for any $k \geq 3$, the MIS problem is NP-hard for graphs in $\mathcal{S}_k$. To this end, we use the operation of *edge subdivision*, i.e., the operation of introducing a new vertex on the edge. It is not difficult to see that the double subdivision of an edge increases the independence number by exactly one, preserves planarity and the maximum vertex degree (except for the trivial case when the maximum degree is one). Applying this operation repeatedly to each edge of the input graph, we eliminate small cycles, as well as small graphs of the form H_i. Therefore, by means of double subdivisions of edges any planar graph of maximum degree at most 3 can be transformed in polynomial time into a graph in the class $\mathcal{S}_k$ for any $k \geq 3$, which proves the NP-hardness of the problem in this class.

To complete the proof, we show that there is a k such that $\mathcal{S}_k \subseteq X_3$. Denote $k := \max\{3, \kappa(\mathcal{F}_3)\}$ and let G belong to $\mathcal{S}_k$. Clearly, G is a planar graph of degree at most 3. Assume that G does not belong to X_3. Then G contains a graph $A \in \mathcal{F}$ as an induced subgraph. Since G is of degree at most 3, $A \in \mathcal{F}_3$. From the choice of G we also know that A belongs to $\mathcal{S}_k$, but then $k < \kappa(A) \leq \kappa(\mathcal{F}_3) \leq k$, a contradiction. Therefore, $G \in X_3$ and hence $\mathcal{S}_k \subseteq X_3$. □

Let us now fix a set of graphs $\mathcal{F}$, and let X denote the class of all (either general, or planar) $\mathcal{F}$-free graphs. Theorem 1.3.1 implies that, unless $P = NP$, one can only hope for a polynomial-time solution to the MAXIMUM INDEPENDENT SET problem in the class X if $\kappa(\mathcal{F}_3) = \infty$. In other words, the problem is polynomially solvable only if at least one of the following conditions is satisfied:

(*i*) $\mathcal{F}$ contains a graph from the class $\mathcal{S}$, the class of all graphs whose each connected component is of the form $S_{i,j,k}$ (see Figure 1.1), or

(*ii*) $\mathcal{F}$ contains graphs of maximum degree at most 3 with arbitrarily large girth (i.e., the size of a smallest cycle), or

(*iii*) $\mathcal{F}$ contains graphs of maximum degree at most 3 with arbitrarily large size of a smallest induced copy of H_i.

1.4 Main Contributions of The Thesis

Besides the general hardness result just described, the main contributions of this thesis are the following.

1.4.1 Extension of the Augmenting Graph Approach

The *method of augmenting graphs* is a general approach to the maximum independent set problem. In Section 2.1, we develop an extension of this method by introducing the notion of a *redundant set* of vertices. Results from Sections 3.2 and 3.4.2 are based on this extension.

1.4.2 Graph Classes With Polynomially Solvable Maximum Independent Set Problem

Chapters 3 and 4 are devoted to polynomial-time solutions to the maximum independent set problem and its weighted version in particular graph classes. Chapter 3 discusses classes of graphs, defined by finitely many forbidden induced subgraphs, while in Chapter 4 the set of forbidden induced subgraphs is infinite.

- In Section 3.1, we describe the first polynomial-time algorithm that solves the problem for weighted *fork-free graphs* (Theorem 3.1.4), thus generalizing a classical result for *claw-free graphs* [90, 95]. The *claw* is the graph $S_{1,1,1}$, and the *fork* is the graph $S_{1,1,2}$.

- In Section 3.2, we show that the method of finding augmenting graphs leads to an efficient solution to the MAXIMUM INDEPENDENT SET problem in the class of

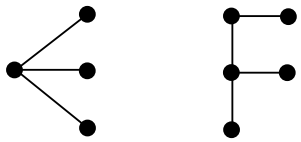

Figure 1.3: The *claw* (left) and the *fork*

$(S_{1,2,5}, \textit{banner})$-free graphs (Theorem 3.2.19), where a *banner* is a graph with vertices a, b, c, d, e and edges ab, bc, cd, de, eb.

- Several results deal with graphs of *bounded vertex degree*, that is, when there is a constant Δ associated to the set of input instances such that the maximum degree of every input graph is bounded above by Δ. In Sections 3.3 and 4.1, we show that for any fixed integer Δ, the MAXIMUM WEIGHT INDEPENDENT SET problem is polynomially solvable for graphs with maximum degree at most Δ that are:
  - $mS_{1,k,k}$-free, for any fixed $m \geq 1$ and $k \geq 1$ (Corollary 3.3.2);
  - $(A_k, A_{k+1}, \ldots)$-free, for any fixed $k \geq 3$, where A_k is the *apple of order* k, that is, the graph obtained from a cycle of length k by introducing a new vertex and joining it by an edge to exactly one vertex of the cycle (Theorem 4.1.1);
  - $(H_k, H_{k+1}, \ldots)$-free, for any fixed $k \geq 1$ (Theorem 4.1.4).
- In Sections 3.4 and 4.2, we describe polynomial-time solutions to the MAXIMUM INDEPENDENT SET problem in subclasses of planar graphs and more generally, subclasses of graphs that exclude a fixed apex graph as a minor. *Under these assumptions*, we show the following results:
  - The MAXIMUM INDEPENDENT SET problem admits a polynomial-time solution for $S_{1,2,k}$-free graphs, for any fixed $k \geq 2$ (Theorem 3.4.8).
  - The MAXIMUM WEIGHT INDEPENDENT SET problem admits a polynomial-time 2-approximation algorithm for $(H_k, H_{k+1}, \ldots)$-free graphs, for any fixed $k \geq 1$ (Theorem 4.2.4).
  - The MAXIMUM WEIGHT INDEPENDENT SET problem admits a linear-time solution

for $(S, L(S'))$-free graphs, where $S, S' \in \mathcal{S}$, and $L(S')$ denotes the line graph of S' (Corollary 3.4.7).

– The MAXIMUM WEIGHT INDEPENDENT SET problem admits a linear-time solution for $(C_k, C_{k+1}, \ldots)$-free graphs, for any fixed $k \geq 3$ (Corollary 4.2.2).

The latter two results are obtained by showing that the treewidth of graphs in such classes is bounded above by a constant, thus giving linear-time solutions to many other optimization problems too [7].

1.4.3 Complexity Results for the Exact Weighted Independent Set Problem

Chapter 5 is devoted to complexity results for the exact weighted independent set problem, when the input graphs are restricted to particular graph classes.

The exact weighted independent set and the exact weighted maximum independent set problems are strongly NP-complete for:

- cubic bipartite graphs (Theorem 5.2.1 and Corollary 5.1.2);
- $\mathcal{F}$-free bipartite graphs of degree at most 3, whenever $\kappa(\mathcal{F}_3) < \infty$ (Theorem 5.2.2).

The exact weighted independent set and the exact weighted maximum independent set problems are solvable in pseudo-polynomial time for any of the following graph classes:

- *mK_2-free graphs* (Theorem 5.3.4),
- *interval graphs* (Theorem 5.3.5) and their generalizations *k-thin graphs* (Theorem 5.3.6),
- *circle graphs* (Theorem 5.3.7),
- *chordal graphs* (Theorem 5.3.8),
- *AT-free graphs* (Theorem 5.3.16),
- *(claw, net)-free graphs* (Corollary 5.3.17),
- *distance-hereditary graphs* (Theorem 5.3.18),

- *graphs of bounded treewidth* (Theorem 5.3.19),
- *graphs of bounded clique-width* (Theorem 5.3.20),
- certain *subclasses of P_5-free and fork-free graphs* (Theorem 5.3.24).

The results of Theorem 5.3.24 are derived by means of modular decomposition. The application of modular decomposition to the exact weighted independent set problem is described in Section 5.3.2 and may be of independent interest.

Note that in view of the relation between the exact perfect matching problem and the exact weighted maximum independent set problem, as pointed out in Section 1.2, each of the above polynomial results also gives a polynomial result to the exact perfect matching problem. Whenever exact weighted maximum independent set problem is (pseudo-)polynomially solvable for a class of graphs X, the exact perfect matching problem is (pseudo-)polynomially solvable for graphs in the set $\{G : L(G) \in X\}$. For example,

- The EXACT PERFECT MATCHING problem is solvable in pseudo-polynomial time for graphs of bounded treewidth (Corollary 5.3.22).

1.5 Notations

All graphs considered are finite, simple and undirected. Unless otherwise stated, n and m will denote the number of vertices and edges, respectively, of the graph considered. For a graph G, we will denote by $V(G)$ and $E(G)$ the vertex-set and the edge-set of G, respectively. An edge $\{u, v\}$ of a graph will be also denoted uv. For a vertex x in a graph G, we denote by $N_G(x)$ the neighborhood of x in G, i.e., the set of vertices adjacent to x, and by $N_G[x]$ the closed neighborhood of x, i.e., the set $N_G(x) \cup \{x\}$. We will write $N(x)$ and $N[x]$ instead of $N_G(x)$ and $N_G[x]$ if no confusion can arise. The degree of v, denoted $deg(v)$, is $|N(v)|$. The *maximum degree* of a graph G is $\Delta(G) = \max\{deg(v) : v \in V(G)\}$. For a subset $U \subset V(G)$, we will denote by $N(U)$ the neighborhood of U, i.e., the set of vertices of G outside U that have at least one neighbor in U. Also, $N_U(v) := N(v) \cap U$, and if W is another subset of $V(G)$ then $N_W(U) := N(U) \cap W$. For a nonnegative integer n

and a graph G, we denote by nG the graph consisting of n disjoint copies of G. P_n and C_n denote the chordless path (also called a *chain*) and the chordless cycle on n vertices.

We say that a graph H is

- an *induced subgraph* of G if H can be obtained from G by deletion of some (possibly none) vertices; the subgraph of G *induced by* $U \subseteq V(G)$ is the graph obtained from G by deleting the vertices from $V(G) \backslash U$ and it will be denoted by $G[U]$;
- a *subgraph* of G if H can be obtained from G by applying a (possibly empty) sequence of vertex and edge deletions;
- a *minor* of G if H can be obtained from G by applying a (possibly empty) sequence of vertex deletions, edge deletions and edge contractions (an edge contraction is the operation of substituting two adjacent vertices u and v by a new vertex adjacent to every vertex in $(N(u) \cup N(v)) \setminus \{u, v\}$);
- a *subdivision* of G if H can be obtained from G by applying a (possibly empty) sequence of edge subdivisions (an edge subdivision is the operation of removing an edge $\{u, v\}$ from a graph G and adding to G a new vertex w and two edges $\{u, w\}, \{w, v\}$).

Let $\mathcal{F}$ be a set of graphs. Similarly as for $\mathcal{F}$-free graphs, *$\mathcal{F}$-minor-free* graphs are defined as those graphs G such that no graph from $\mathcal{F}$ is a minor of G. If a graph H is a minor of G, we also say that G contains H as a minor.

A graph G is *bipartite* if its vertex set admits a bipartition $V(G) = L \cup R$ such that $E(G) \subseteq \{\{u, v\} : u \in L, v \in R\}$. If a bipartite graph G is given together with such a partition, we write $G = (L, R; E)$. (Note that a disconnected bipartite graph may admit several essentially different bipartitions.) By K_n we denote the complete graph on n vertices, and by $K_{s,t}$ the complete bipartite graph with parts of size s and t. A *clique* in a graph is a subset of vertices that induces a complete graph. A *weighted graph* is a pair (G, w), where G is a graph and $w : V(G) \to \mathbb{Z}$. When a weighted graph is part of the input to the maximum weight independent set problem, we will always assume that the weights are positive. (Vertices with nonpositive weights are irrelevant for maximum weight independent sets.)

For a graph G, we denote by co-G (also $\overline{G}$) the edge-complement of G. By *component* we will always mean a connected component. The *distance* between two vertices u and v in a connected graph G is the length (i.e., the number of edges) of a shortest path connecting them. Connected components of $\overline{G}$ will be called *co-components* of G. Given two disjoint subsets $U \subset V(G)$ and $W \subset V(G)$, we will say that U *dominates* W if every vertex of U is adjacent to every vertex of W. A graph is *2-connected* if it remains connected after the removal of any vertex. A *2-connected component* of a graph G is a maximal 2-connected subgraph of G. For graph-theoretical terms not defined here, we refer the reader to [19]. For a subset of vertices $V' \subseteq V$ in a weighted graph (G, w), we let $w(V') = \sum_{v \in V'} w(v)$. For a positive integer k, we write $[k]$ for the set $\{1, \ldots, k\}$.

Chapter 2

Techniques For Finding Maximum Independent Sets: An Overview and Extensions

In this chapter, we provide an overview of techniques and algorithmic tools that have been used in order to tackle the MAXIMUM INDEPENDENT SET problem in particular graph classes:

- In Section 2.1, we describe the *method of augmenting graphs*. We also introduce the notion of a *redundant set* to develop an extension of the existing method.
- Section 2.2 is devoted to *modular decomposition*.
- In Section 2.3, we discuss the *decomposition by clique separators*.
- In Section 2.4, we show that it suffices to solve the problem after the deletion of constantly many vertices from the graph.
- We conclude in Section 2.5 with a brief overview of other techniques.

We remark that the method of augmenting graphs can in general only be applied to unweighted graphs, decomposition by clique separators works for both weighted and unweighted graphs, while modular decomposition is only applicable to weighted graphs.

Most of these algorithmic tools will be exploited in Chapters 3 and 4 where we present polynomial-time solutions to the MAXIMUM INDEPENDENT SET problem in particular graph classes.

2.1 Augmenting Graphs

It is well-known that finding a maximum matching in a given graph can be done in polynomial time. This is due to Berge's idea of augmenting (alternating) chains [11] and the

celebrated algorithm of Edmonds [45] that finds augmenting chains.[1] This result immediately translates into a polynomial solution to the maximum independent set problem in the class of line graphs.

Rephrasing Berge's idea in terms of independent sets, we can say that in a line graph an independent set is maximum if and only if there are no augmenting chains with respect to this set. This idea can be extended to a general approach for finding maximum independent sets, the method of *finding augmenting graphs*.

Let G be a graph and I an independent set in G. We will call the vertices of I *white* and the remaining vertices of G *black*.

Definition 2.1.1. *An* augmenting graph for I in G *is an induced bipartite subgraph* $H = (W, B; E)$ *of* G*, where* $W \cup B$ *is a bipartition of its vertex set and* E *its edge set, such that:*

- $W \subseteq I$,
- $B \subseteq V(G) \backslash I$,
- $|B| > |W|$, *and*
- $N(B) \cap I \subseteq W$.

If a bipartite subgraph H of G is augmenting for I, we also say that I *admits* the augmenting graph. Clearly if $H = (W, B; E)$ is an augmenting graph for I, then I is *not* a maximum independent set in G, since the set $I' = (I - W) \cup B$ is independent and $|I'| > |I|$. We will say that the set I' is obtained from I by H*-augmentation*.

Conversely, if I is not a maximum independent set, and I' is an independent set such that $|I'| > |I|$, then the subgraph of G induced by the set $(I - I') \cup (I' - I)$ is augmenting for I. Therefore, the following key result holds.

Theorem of augmenting graphs. *An independent set* I *in a graph* G *is maximum if and only if there are no augmenting graphs for* I.

---

[1] Lovász and Plummer observed in [77] that Edmonds' solution is "among the most involved of combinatorial algorithms."

This theorem suggests the following general approach to find a maximum independent set in a graph G: begin with any independent set I in G and, as long as I admits an augmenting graph H, apply H-augmentation to I. This approach has proven to be a useful tool to develop approximate solutions to the problem [65], to compute bounds on the independence number [40], and to solve the problem in polynomial time for graphs in special classes [90, 3]. Here, we focus on efficient implementations of the approach for graphs in particular classes. To this end, let us introduce some more definitions.

Definition 2.1.2. *A bipartite graph $H = (W, B; E)$ will be called* augmenting *if there is a graph G and an independent set I in G such that H is augmenting for I in G.*

Clearly not every bipartite graph is augmenting. For instance, a bipartite cycle cannot be augmenting, since it has equally many vertices in both parts. Moreover, without loss of generality we may exclude from our consideration those augmenting graphs which are not minimal.

Definition 2.1.3. *An augmenting graph H for a set I is called* minimal *if no proper induced subgraph of H is augmenting for I.*

Some bipartite graphs that could be augmenting are never minimal augmenting. To give an example, consider the claw $K_{1,3} = S_{1,1,1}$. If it is augmenting for an independent set I, then its subgraph obtained by deleting any vertex of degree 1 is also augmenting for I. The following lemma characterizes minimal augmenting graphs.

Lemma 2.1.4. *An augmenting graph $H = (W, B; E)$ is minimal if and only if*

(i) $|W| = |B| - 1$*;*

(ii) for every nonempty subset $A \subseteq W$*,* $|A| < |N(A)|$*.*

(iii) H is connected.

Proof. Let $H = (W, B; E)$ be a minimal augmenting graph. To show (i), it is enough to observe that if $|W| < |B| - 1$, then the graph induced by $W \cup (B \backslash \{v\})$ for some $v \in B$ is augmenting too, contradicting minimality. To show (ii), assume $|A| \geq |N(A)|$ for some

nonempty subset A of W. Then the vertices in $(W\backslash A)\cup(B\backslash N(A))$ induce a proper subgraph of H which is augmenting too. Condition (iii) easily follows from (i) and (ii).

Conversely, let $H = (W, B; E)$ be an augmenting graph for an independent set I of a graph G, satisfying $(i) - (iii)$. Assume that $H' = (W', B'; E')$ is a proper induced subgraph of H which also is augmenting for I. Then W' is a proper subset of W (since otherwise $|B'| \geq |W| + 1 = |B|$ and $B' = B$, implying $H' = H$). Therefore, the set $A := W\backslash W'$ is nonempty. Moreover, since H' is augmenting, it follows that $N(A) \subseteq B\backslash B'$, which in its turn implies that $|N(A)| \leq |A|$, contradicting (ii). □

For a polynomial-time implementation of the augmenting graph approach in a class of graphs X, one has to

(a) find a complete list of (minimal) augmenting graphs in X,

(b) develop a polynomial-time procedure for detecting augmenting graphs from the list.

For instance, for the class of claw-free graphs, question (a) has a simple answer. Indeed, by definition, augmenting graphs are bipartite, and each vertex in a claw-free bipartite graph clearly has degree at most two. Therefore, every connected claw-free bipartite graph is either an even cycle or a chain. Cycles of even length and chains of odd length are not augmenting. Thus, every connected claw-free augmenting graph is a chain of even length.

In general, augmenting chains are not the only type of augmenting graphs. For instance, Mosca showed in [92] that in the class of (P_6, C_4)-free graphs every augmenting graph is a simple augmenting tree (the graph T_1 represented in Figure 3.4, Section 3.2). Many more types of augmenting graphs have been revealed in [3, 5, 17, 57]. With each of them, one can associate the problem of finding augmenting graphs of the given type. The number of various types of augmenting graphs is generally growing with each extension of the class under review. In order to simplify the problem of finding augmenting graphs, we introduce in the following section the notion of a *redundant set* of vertices, which often allows us to reduce this problem to finding some "basic" types of augmenting graphs.

2.1.1 The Problem of Finding Augmenting Graphs

In its most general form, the problem of finding augmenting graphs can be formulated as follows:

AUGMENTATION

Instance: A graph G, and a maximal independent set I in G.

Problem: Find an augmenting graph for I whenever I admits one.

From NP-hardness of the independent set problem and the Theorem of augmenting graphs we conclude that

Claim 2.1.5. *The problem* AUGMENTATION *is NP-hard.*

Since in its whole generality the problem is intractable, we introduce a hierarchy of subproblems and study the computational complexity of the problems in this hierarchy. For a class $\mathcal{A}$ of augmenting graphs, let us consider the following problem:

AUGMENTATION($\mathcal{A}$)

Instance: A graph G, and a maximal independent set I in G.

Problem: Find an augmenting graph for I whenever I admits an augmenting graph from $\mathcal{A}$.

Note that we do not require the output graph to belong to $\mathcal{A}$. If $\mathcal{A}$ is the class of all augmenting graphs, then the problem AUGMENTATION($\mathcal{A}$) coincides with the problem AUGMENTATION and hence is intractable. However, it becomes polynomial-time solvable, for instance, if $\mathcal{A}$ contains only finitely many graphs. Between these two extremes there are infinitely many intermediate classes of augmenting graphs and respective problems.

For example, if the set $\mathcal{A}$ consists of augmenting chains, then, by Edmonds' algorithm, the problem is polynomial-time solvable for line graphs. In 1980, independently Minty [90] and Sbihi [105] extended the solution of Edmonds to claw-free graphs, a class properly containing the line graphs. In conjunction with the fact that in the class of claw-free graphs augmenting chains constitute the only type of augmenting graphs this has led to a

polynomial-time solution to the maximum independent set problem in that class. Recently, the problem of finding augmenting chains has been solved for some extensions of claw-free graphs [58, 71].

The following notion is a helpful tool for establishing reducibility among these problems.

Definition 2.1.6. *In an augmenting graph $H = (W, B; E)$ a subset of vertices U will be called* redundant *if*

- $|U \cap W| = |U \cap B|$,
- *H contains no edges between black vertices of U and vertices of $H - U$.*

Theorem 2.1.7 ([80])**.** *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two classes of augmenting graphs. If there is a constant k such that for every graph $H = (W, B; E) \in \mathcal{A}_2$ there is a redundant subset U of size at most k such that $H - U \in \mathcal{A}_1$, then the problem* Augmentation*$(\mathcal{A}_2)$ is polynomially reducible to the problem* Augmentation*$(\mathcal{A}_1)$.*

Proof. Let $Augment_1(G, I)$ be a procedure that solves the problem Augmentation$(\mathcal{A}_1)$ for a graph G and an independent set I. We assume that the procedure outputs a subset V' of $V(G)$ such that $G[V']$ is augmenting for I whenever I admits an augmenting graph from $\mathcal{A}_1$ (and perhaps even if this is not the case). If no augmenting graph is found, then $V' = \emptyset$.

To prove the theorem we present procedure $Augment_2(G, I)$ that solves the problem Augmentation$(\mathcal{A}_2)$:

Procedure $Augment_2(G, I)$

Input: A graph G and an independent set I in G.

Output: A subset V' of $V(G)$ such that $G[V']$ is augmenting for I whenever I admits an augmenting graph from $\mathcal{A}_2$. If no augmenting graph is found, then $V' = \emptyset$.

begin

for all ($U \subseteq V(G)$ of size at most k such that

$B_0 := U \cap (V(G) \backslash I)$ is an independent set in G,

$|B_0| = |U \cap I|$ and $N_G(B_0) \cap (I \backslash U) = \emptyset$)

do

[*remove the neighbors of* B_0 *in* $V(G) \backslash I$]

let $G' = G - N_G(B_0) \cap (V(G) \backslash I)$;

[*try to solve the problem* AUGMENTATION$(\mathcal{A}_1)$]

let $T = Augment_1(G' - U,\ I \backslash U)$;

if ($T \neq \emptyset$) [*we have an augmenting graph for* I]

then return $U \cup T$;

return $\emptyset$;

end;

Suppose I admits an augmenting graph $H = (W, B; E) \in \mathcal{A}_2$. Then, according to the theorem's assumption, H contains a redundant set U of size at most k such that $H - U \in \mathcal{A}_1$. It is not difficult to see that the graph $H - U$ is augmenting for $I \backslash U$ in $G' - U$. Therefore, procedure $Augment_1$ must output a nonempty set T. Consequently, procedure $Augment_2$ also outputs a nonempty set $U \cup T$. Obviously, $G[U \cup T]$ is a bipartite graph. Moreover, since U is a redundant set, the graph $G[U \cup T]$ is augmenting for I even if $G[T]$ does not coincide with $H - U$. Therefore, whenever I admits an augmenting graph from $\mathcal{A}_2$, procedure $Augment_2$ finds an augmenting graph. To this end, it inspects polynomially many subsets of vertices of the input graph, which results in polynomially many calls of the procedure $Augment_1$. Therefore, the problem AUGMENTATION$(\mathcal{A}_2)$ is polynomially reducible to the problem AUGMENTATION$(\mathcal{A}_1)$. □

Remark. *Note that if in the definition of a redundant set we drop the second condition, then the procedure in the above theorem may fail to work: it may happen that even though T from the procedure induces an augmenting graph for $I \backslash U$ in $G' - U$, the graph induced by $T \cup U$ may not be augmenting for I. In particular, if I' denotes the set of white neighbors of black vertices of U in the graph $G' - U$ and if T is augmenting for $I \backslash U$ in $G' - U$, then $T \cup U$ is augmenting for I if and only if $I' \subseteq V(T)$.*

Applications of Theorem 2.1.7 will be discussed in Sections 3.2 and 3.4.2.

2.2 Modular Decomposition

The idea of *modular decomposition* has been first described in the 1960s by Gallai [53], and also appeared in the literature under various other names such as *prime tree decomposition* [46], *X-join decomposition* [63], or *substitution decomposition* [91]. This technique allows one to reduce many graph problems from arbitrary graphs to so-called *prime* graphs. In this subsection, we show how to apply modular decomposition to the MAXIMUM WEIGHT INDEPENDENT SET problem.

Let $G = (V, E)$ be a graph, U a subset of V and x a vertex of G outside U. We will say that x *distinguishes* U if x has both a neighbor and a non-neighbor in U. A subset $U \subset V(G)$ is called a *module* in G if it is indistinguishable for the vertices outside U. A module U is *nontrivial* if $1 < |U| < |V|$, otherwise it is *trivial*. A graph each module of which is trivial is called *prime*.

An important property of maximal modules is that if G and co-G are both connected, then the maximal modules of G are pairwise disjoint. Moreover, from the above definition it follows that if U and W are maximal modules, then either U dominates W or there are no edges between them. This property provides a reduction of the maximum weight independent set problem (and many other problems) from the graph G to a graph G^0 obtained from G by contracting each maximal module to a single vertex. We formally describe this reduction in the recursive procedure ALPHA(G, w) below.

Algorithm ALPHA(G, w)

Input: a weighted graph (G, w)

Output: an independent set I of maximum weight in G.

1. If $|V(G)| = 1$, set $I = V(G)$ and go to 7.
2. If G is disconnected, partition it into connected components $\mathcal{M}_1, \ldots, \mathcal{M}_k$.
3. If co-G is disconnected, partition G into co-components $\mathcal{M}_1, \ldots, \mathcal{M}_k$.
4. If G and co-G are connected, partition G into maximal modules $\mathcal{M}_1, \ldots, \mathcal{M}_k$.
5. Construct a weighted graph G^0 from G by contracting each $\mathcal{M}_j$ $(j = 1, \ldots, k)$ to a single vertex and assigning to that vertex the weight $w(\text{ALPHA}(G[\mathcal{M}_j]))$.
6. Find in G^0 an independent set I^0 of maximum weight, and set $I = \bigcup_{j \in I^0} \text{ALPHA}(G[\mathcal{M}_j])$.
7. Return I and STOP.

Observe that the graph G^0 constructed in step 5 of the algorithm is either an edgeless graph, a complete graph, or a prime graph. Therefore, the modular decomposition approach reduces the problem from a graph to its prime induced subgraphs. The following theorem answers the question on the complexity of such a reduction.

Theorem 2.2.1. *Let X be a class of graphs and X^* the class of all prime induced subgraphs of the graphs in X. If there is a constant $p \geq 1$ such that the maximum weight independent set problem can be solved for graphs in X^* in time $O(n^p)$, then this problem can be solved for graphs in X in time $O(n^p + m)$.*

Proof. Let G be a graph in X with n vertices and m edges. The recursive decomposition of G produced by *Algorithm* ALPHA can be implemented in time $O(n + m)$ [87]. This decomposition associates with G a tree $T(G)$ whose leaves correspond to the vertices of G, while the internal nodes of $T(G)$ represent induced subgraphs of G with at least two vertices.

Consider an internal node U of $T(G)$, and let G_U denote the induced subgraph of G corresponding to U. Then the children of G_U correspond to the subgraphs $G[\mathcal{M}_1], \ldots, G[\mathcal{M}_k]$, where $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$ is the partition of G_U defined in steps 2–4 of the algorithm. If G_U ($\overline{G}_U$) is disconnected, then G_U^0 is an edgeless (complete) graph, and the problem can be trivially solved for G_U^0 in time $O(|V(G_U^0)|)$. If both G and $\overline{G}$ are connected, then G_U^0 is a prime induced subgraph of G, and the problem can be solved for G_U^0 in time $O(|V(G_U^0)|^p)$ with $p \geq 1$ by our assumption. Summing up over all internal nodes of $T(G)$, we conclude that the total time complexity of the problem on G is bounded by $O(\sum\limits_U |V(G_U^0)|^p)$. It is not difficult to see that the total number of vertices in all graphs G_U^0 corresponding to internal nodes $U \in V(T(G))$ equals to the number of edges of $T(G)$, i.e., $|V(T(G))| - 1$. Since the number of leaves of $T(G)$ is n and the number of internal nodes is at most $n-1$, we conclude that

$$\sum_U |V(G_U^0)|^p \leq (\sum_U |V(G_U^0)|)^p \leq (2n-2)^p = O(n^p).$$

Adding the term $O(n+m)$ needed to obtain the decomposition tree, we obtain the desired time complexity. The theorem is proved. □

An example of a graph class where modular decomposition provides a polynomial-time solution to the maximum weight independent set problem is the class of P_4-free graphs, also known as *cographs*. Every graph in this class is either disconnected, or the complement to a disconnected graph [33]. Therefore, every prime P_4-free graph is the graph on a single vertex.

Application of modular decomposition to the maximum weight independent set problem has been extended from P_4-free graphs to several subclasses of P_5-free and fork-free graphs [15, 27, 49, 59, 72]. In Section 3.1, we will show that modular decomposition leads to a polynomial-time solution to the maximum weight independent set problem in the whole class of fork-free graphs.

2.3 Decomposition by Clique Separators

A *clique separator* in a connected graph G is a subset K of vertices of G which induces a complete graph, such that the graph $G-K$ is disconnected. It is well-known that the MAXIMUM (WEIGHT) INDEPENDENT SET problem can be reduced in polynomial-time to graphs without clique separators. The corresponding divide-and-conquer approach providing such a reduction is known as *decomposition by clique separators*. It was originally developed by Whitesides [113], and adapted for the weighted, and unweighted case of the MAXIMUM INDEPENDENT SET problem by Tarjan [108] and Alekseev [4], respectively.

More specifically, decomposition by clique separators can be used to efficiently solve the MAXIMUM WEIGHT INDEPENDENT SET problem for a class of graphs X, once we know how to solve it on certain subgraphs of the *atoms* (i.e., induced subgraphs of the input graph which contain no clique separators). Here, we recall this recursive method, as described by Tarjan [108].

Procedure ALPHA-DCS(G, w)

Input: a weighted graph (G, w)

Output: an independent set I of maximum weight in G.

Step 1. Let $\{A, B, C\}$ be a vertex partition such that C is a clique, no edge joins a vertex from A and a vertex in B, and $G[A \cup C]$ is an atom. We denote by $w(I)$ the total weight of a vertex set I.

Step 2. For each vertex $v \in C$, determine a maximum-weight independent set $I(v)$ in $G[A - N(v)]$. Determine a maximum-weight independent set I' in $G[A]$.

Step 3. For each vertex $v \in C$, redefine the weight of v to be $w(v) + w(I(v)) - w(I)$. Find a maximum-weight independent set I'' in $G[B \cup C]$ with respect to the new weights.

Step 4. Define

$$I = \begin{cases} I(v) \cup I'', & \text{if } v \in I'' \cap C; \\ I \cup I'', & \text{if } I'' \cap C = \emptyset. \end{cases}$$

Decomposition by clique separators gives a direct solution to the maximum weight independent set problem in *chordal graphs*. As shown by Dirac [41], every chordal graph has a *simplicial vertex*, that is, a vertex whose neighborhood is a clique. Therefore, the only chordal graphs without separating cliques are the complete graphs.

In Section 3.4.2, we will combine the decomposition by clique separators with the augmenting graph approach to solve the maximum independent set problem in $S_{1,2,2}$-free $K_{3,3}$-minor-free graphs.

Recently, Brandstädt and Hoàng [26] combined decomposition by clique separators with modular decomposition into a more general decomposition scheme.

Theorem 2.3.1 ([26])**.** *Let X be a class of graphs. If the* MAXIMUM WEIGHT INDEPENDENT SET *problem can be solved in polynomial time for those induced subgraphs of graphs in X which are prime and have no clique separators, then the* MAXIMUM WEIGHT INDEPENDENT SET *problem is solvable in polynomial time for graphs in X.*

This decomposition scheme has been used in developing polynomial-time solutions to the maximum weight independent set problem in subclasses of P_5-free graphs [26, 28]. We will present another application of this result in Section 4.1.1.

2.4 Removal of Constantly Many Vertices

We continue with a technical lemma that provides a general reduction of the MAXIMUM WEIGHT INDEPENDENT SET problem between graph classes and will be used in Sections 3.3 and 4.1.2.

Theorem 2.4.1. *Let X be a class of graphs such that there is a constant p and a hereditary class of graphs Y such that:*

- MAXIMUM WEIGHT INDEPENDENT SET *problem can be solved in polynomial time for graphs in Y, and*
- *for each $G \in X$, we can find in polynomial time a subset U of its vertex set of cardinality at most p such that $G - U \in Y$.*

Then, the MAXIMUM WEIGHT INDEPENDENT SET *problem can be solved in polynomial time for graphs in X.*

Proof. We describe a polynomial time procedure which solves the MAXIMUM WEIGHT INDEPENDENT SET problem for graphs in X. Given a (weighted) graph $G \in X$, we first find a subset U of its vertex set of cardinality at most p such that $G' = G - U \in Y$. Then, we enumerate in constant time all the independent sets $\{I_1, \ldots, I_N\}$ of $G[U]$. For each $i = 1, \ldots, N$, we find a maximum-weight independent set I'_i in the graph obtained from G' by deleting the neighbors of vertices in I_i. Clearly, the solution to the MAXIMUM WEIGHT INDEPENDENT SET problem for G is then given by a set of the form $I_i \cup I'_i$ of maximum total weight. □

2.5 Other Techniques

Graphs of treewidth at most k, also known as *partial k-trees*, generalize trees and are very important from an algorithmic viewpoint, since many graph problems that are NP-hard for general graphs are solvable in linear time when restricted to graphs of treewidth at most k [7]. In particular, showing that a graph class is of uniformly bounded treewidth implies that the MAXIMUM WEIGHT INDEPENDENT SET problem is solvable in linear time (in the number of vertices) for graphs in such a class. We will make use of this fact in Sections 3.4.1, 3.4.2, 4.1.1 and 4.2.

Treewidth has been introduced by Robertson and Seymour [103, 104] as a graph parameter that roughly measures how tree-like the graph is. For completeness, we give here a definition of treewidth. A *tree decomposition* of a graph $G = (V, E)$ is a pair $(T, \mathcal{X})$ consisting of a tree $T = (\mathcal{I}, F)$ and $\mathcal{X} = \{X_i : i \in \mathcal{I}, X_i \subseteq V\}$ such that

- $\bigcup_{i \in \mathcal{I}} X_i = V$,
- for every $e = \{u, v\} \in E$, there is an $i \in \mathcal{I}$ such that $u, v \in X_i$, and
- for every $v \in V$, the subgraph of T induced by $\{X_i \in \mathcal{X} : v \in X_i\}$ is a tree.

The *width of a tree decomposition* $(T, \mathcal{X})$ is $w((T, \mathcal{X})) = \max\{|X_i| : i \in \mathcal{I}\} - 1$. The *treewidth* of a graph G is $\mathrm{tw}(G) := \min\{w((T, \mathcal{X})) : (T, \mathcal{X}) \text{ is a tree decomposition of } G\}$. Graphs

of treewidth 0 are precisely the edgeless graphs, and graphs of treewidth at most 1 are trees, or more generally, forests. We refer to [14] for an excellent tutorial on treewidth. A generalization of graphs of bounded treewidth is provided by *graphs of bounded clique-width* (cf. Section 5.3.1).

We conclude this chapter by mentioning several other ways of tackling the MAXIMUM (WEIGHT) INDEPENDENT SET problem in particular graph classes:

- In bipartite graphs, the MAXIMUM WEIGHT INDEPENDENT SET problem can be solved by *network flow techniques*.
- In perfect graphs, the MAXIMUM WEIGHT INDEPENDENT SET problem can be solved by *semi-definite programming* [61].
- Techniques based on *Boolean identities* have been developed. *Struction* [6], for example, can be used to solve the MAXIMUM INDEPENDENT SET problem in *circular-arc graphs* [60], and in *subclasses of claw-free graphs* [66, 67].
- Other *graph transformations:*
  - *Clique reduction* has solved the MAXIMUM INDEPENDENT SET problem in *claw-free graphs* [77], *AH-free graphs* [70], and (*bull, fork*)*-free graphs* [36].
  - *Conic reduction* has solved the MAXIMUM INDEPENDENT SET problem in (*fork, parachute, butterfly, kite*)*-free graphs* [79].
  - *Removal of simplicial vertices* has solved the MAXIMUM INDEPENDENT SET problem in (*fork, banner,* $K_{1,4}, P_5$)*-free graphs* [22].
- *Dynamic programming.* Special dynamic programming approaches have been designed for graphs in particular classes, based on their structural properties and characterizations. Example include interval graphs [102], distance-hereditary graphs [9], and AT-free graphs [31]. We will return to these graph classes in Chapter 5, where we will extend these results to pseudo-polynomial time solutions to the EXACT WEIGHTED INDEPENDENT SET problem.

Chapter 3

Polynomial Cases of Finding Maximum Independent Sets: Finitely Many Forbidden Induced Subgraphs

We partition our results about polynomial-time solvable cases of the maximum independent set problem and its weighted version into two types, according to whether the set of forbidden induced subgraphs is finite, or infinite. In this chapter, we deal with the first case. The case where infinitely many subgraphs are forbidden will be the topic of Chapter 4.

Let $\mathcal{F}$ be a finite set of graphs, and let X denote the class of all (either general, or planar) $\mathcal{F}$-free graphs. Recall that Alekseev's result (Theorem 1.1.1) implies that, unless $P = NP$, one can only hope for a polynomial-time solution to the MAXIMUM INDEPENDENT SET problem in the class X if the set $\mathcal{F}$ contains a graph from $\mathcal{S}$. That is, at least one of the forbidden induced subgraphs has to consist of the disjoint union of $S_{i,j,k}$'s (where the values of i, j and k may depend on component).

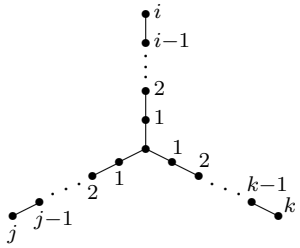

Figure 3.1: Graph $S_{i,j,k}$

A fundamental example of this type where the MAXIMUM INDEPENDENT SET problem admits a polynomial-time solution is provided by the class of claw-free graphs, when $\mathcal{F} = \{claw\} = \{S_{1,1,1}\}$ [90, 105, 77, 95]. We start by presenting two extensions of this classical result:

- In Section 3.1, we describe the first polynomial-time algorithm that solves the problem for weighted fork-free graphs.

- In Section 3.2, we show that the method of finding augmenting graphs leads to an efficient solution to the MAXIMUM INDEPENDENT SET problem in the class of $(S_{1,2,5}, banner)$-free graphs. A *banner* is a graph with vertices a, b, c, d, e and edges ab, bc, cd, de, eb.

In Section 3.3, we show that in the case of bounded maximum degree, forbidding a graph from $\mathcal{S}$ whose each connected component is of the form $S_{i,j,k}$ with $i = 1$ (and arbitrary values of j and k, depending on the component) results in an "easy" class as well. Finally, in Section 3.4 we present subclasses of planar and more general graphs where the problem is solvable in polynomial-time.

3.1 Fork-free Graphs

For a long time, the class of claw-free graphs remained one of the only three maximal graph classes defined by a single forbidden induced subgraph where the maximum weight independent set problem was known to be solvable in polynomial time, the other two being P_4-free graphs [33] and mK_2-free graphs [48]. Recently, Alekseev [3] found a polynomial-time solution for fork-free graphs, extending both claw-free and P_4-free graphs. A *fork* (also called a *chair*) is the graph $S_{1,1,2}$, that is, the graph obtained from a claw by a single subdivision of one of its edges (see Figure 3.2).

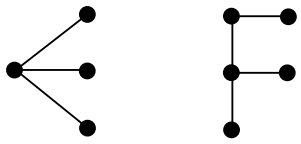

Figure 3.2: The *claw* (left) and the *fork*

Alekseev's solution only works for the unweighted version of the problem. Besides, his algorithm has a high time complexity and uses a sophisticated approach which is difficult

to implement.[1] Here, we propose the first polynomial-time algorithm to solve the problem for weighted fork-free graphs. Our algorithm not only generalizes Alekseev's solution, but also improves on the time complexity.

At the same time, our result extends previously known subclasses of fork-free graphs where the weighted version of the problem was known to be polynomial-time solvable. For example, Brandstädt, Hoàng and Le have developed a polynomial-time solution for weighted $(fork, bull)$-free graphs [23]. To this end, they exploited the idea of modular decomposition, which had led to efficient solutions to the maximum weight independent set problem in some other subclasses of fork-free graphs as well [27, 29]. Modular decompositions will play a key role in our algorithm too.

3.1.1 The Solution

For a graph G and a vertex $x \in V(G)$, let us denote by G_x the graph obtained from G by deleting x and every vertex adjacent to x. Our solution to the maximum weight independent set problem for fork-free graphs is based on two general tools: modular decomposition and the following obvious identity

$$\alpha_w(G) = \max_{x \in V(G)} \{w(x) + \alpha_w(G_x)\}.$$

An immediate consequence of this identity is the following proposition.

Proposition 3.1.1. *If for every vertex $x \in V(G)$, the maximum weight independent set problem can be solved for G_x in time T, then it can be solved for G in time nT.*

Now we prove the main result that leads to an efficient solution of the problem in the class of fork-free graphs.

Theorem 3.1.2. *Let G be a fork-free graph, x an arbitrary vertex of G, and $\tilde{G}$ an induced subgraph of G_x. If both G and $\tilde{G}$ are prime, then $\tilde{G}$ is claw-free.*

Proof. Assume by contradiction that $\tilde{G}$ contains an induced claw. Then it must contain one of the minimal prime extensions of the claw. The complete list of such extensions can

[1] Our rough analysis showed that his solution requires $O(n^{10})$ time.

be found in [25]. It consists of 12 graphs, 7 of which contain a fork, and the remaining 5 are represented in Figure 3.1.1.

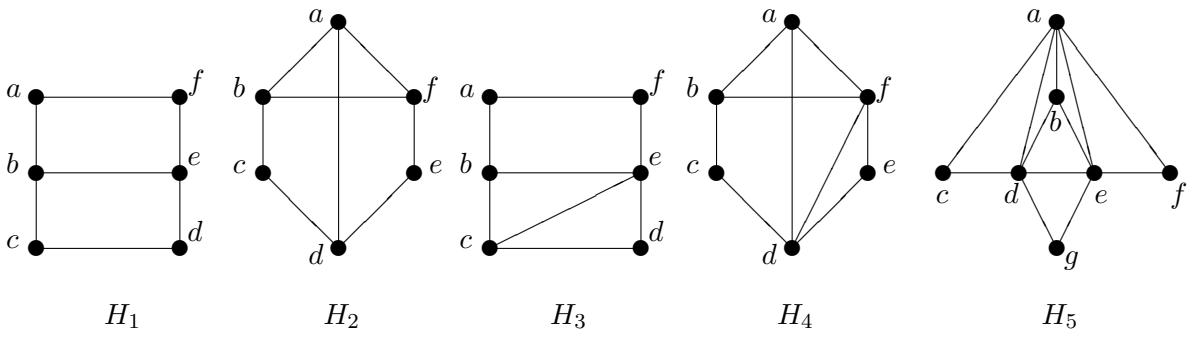

Figure 3.3: Minimal fork-free prime extensions of the claw

Let $H \in \{H_1, \ldots, H_5\}$ denote an induced copy of one of the minimal prime extensions of the claw in the graph $\tilde{G}$.

Claim 3.1.3. *No neighbor of x distinguishes $V(H)$.*

The proof of this claim is a rather tedious case analysis. For the purpose of readability of our current proof, we postpone it to the next section. The statement of the claim allows us to partition the set of neighbors of x into two sets Y and Z such that no vertex in Y has a neighbor in $V(H)$, while Z dominates $V(H)$.

Let W be an (inclusionwise) maximal subset of vertices of G_x with the following properties:

(i) $W \supseteq V(H)$,

(ii) $G[W]$ is connected,

(iii) co-$G[W]$ is connected,

(iv) Z dominates W,

(v) there are no edges between W and Y.

Note that such a set W exists since $V(H)$ satisfies all these properties. By definition, $|W| < |V(G)|$. In addition, $|W| \geq |V(H)| > 1$. Therefore, in order to be prime, G must contain a vertex $u \in V(G)\backslash W$ that distinguishes W. We will now show that the set $W' := W \cup \{u\}$ also enjoys the five stated properties.

Firstly, properties (iv) and (v) for W imply that $u \in V(G_x)$, which yields $W' \subseteq V(G_x)$. W' trivially satisfies property (i).

Since W enjoys properties (ii) and (iii) and since u has both a neighbor and a non-neighbor in W, we conclude that the same two properties still hold for W'.

To see that W' satisfies (iv), i.e., that Z dominates W', assume to the contrary that u has a non-neighbor z in Z. Since u distinguishes W and the complement of $G[W]$ is connected, u distinguishes a pair of non-adjacent vertices $w_1, w_2 \in W$. But now, a fork arises on $\{u, w_1, z, w_2, x\}$, contradicting the fork-freeness of G.

Finally, let us show that there are no edges between W' and Y. Indeed, suppose u is adjacent to a vertex $y \in Y$. Let $P = (v_0, \ldots, v_k)$ be a shortest path connecting $V(H)$ to u in the graph $G[W']$, i.e., $v_0 \in V(H)$ and $v_k = u$. Also, denote $v_{k+1} := y$, $v_{k+2} := x$. Since v_2 has no neighbors in $V(H)$, we conclude by analogy with Claim 3.1.3 that v_1 dominates $V(H)$. But now any two non-adjacent vertices of $V(H)$ together with v_1, v_2 and v_3 induce a fork.

Therefore, the subset W' of $V(G_x)$ satisfies all the above properties, thus contradicting the maximality of W. This completes the proof of Theorem 3.1.2. □

Combining Theorems 2.2.1 and 3.1.2 with Proposition 3.1.1, we conclude that

Theorem 3.1.4 ([81])**.** *The maximum weight independent set problem in the class of fork-free graphs can be solved in polynomial time. In particular, it can be solved in time nT, where T is the time to solve the same problem for claw-free graphs.*

Proof. Let X be the class of fork-free graphs, and X^* the class of prime graphs in X. Also, define $Y := \{G_v : G \in X^* \text{ and } v \in V(G)\}$ and let Y^* denote the class of all prime induced subgraphs of the graphs in Y. By Theorem 3.1.2, every graph in Y^* is claw-free. According to [90, 95], the maximum weight independent set problem in the class of claw-free graphs

can be solved in polynomial time T. Therefore, by Theorem 2.2.1, the problem can be solved for graphs in Y also in time T. This implies an nT solution for graphs in X^* (by Proposition 3.1.1) and an nT solution for graphs in X (again by Theorem 2.2.1). □

The solution for claw-free graphs is based on a reduction to line graphs, where the problem is equivalent to finding a maximum matching in general graphs. We summarize these steps (separately for weighted and unweighted graphs) in the table below, along with corresponding techniques and time complexities. Unfortunately, we can reliably report the complexity only for the line graphs. For the remaining classes, we give rough estimates of the running time based on the information available in the literature.

	Line graphs	Claw-free graphs	Fork-free graphs
unweighted	augmenting graphs $O(\sqrt{n}m)$ [13, 52, 88, 110]	augmenting graphs $O(n^3)$ [90, 105] graph transformations $O(n^4)$ [77]	augmenting graphs $O(n^{10})$ [3]
weighted	augmenting graphs $O(nm + n^2 \log n)$ [51]	augmenting graphs $O(n^7)$ [90, 95]	modular decomposition $O(n^8)$

As seen from the table, in most cases, to implement the reduction to a smaller class, the authors borrowed the idea of augmenting graphs developed for the solution of the problem in the class of line graphs. Here, we used an entirely different approach, and the additional time required for our reduction is not critical. To improve the overall time complexity, one needs a better reduction from claw-free to line graphs.

3.1.2 Proof of Claim 3.1.3

Let y be a neighbor of x. We present the proof for each of the cases $H = H_i$ for $i \in \{1, \ldots, 5\}$. We denote the vertices of H as depicted in Figure 1.

I. $H = H_1$.

Case 1: y has a non-neighbor among the vertices of degree 2 in H. Taking into account the symmetry, we may assume without loss of generality that y is not adjacent to a. We consider two subcases.

1.1: y is adjacent to c. Then

- y is not adjacent to f (otherwise a fork arises on $\{a, f, y, c, x\}$),
- y is not adjacent to e (otherwise a fork arises on $\{f, e, y, c, x\}$),
- y is not adjacent to b (otherwise a fork arises on $\{x, y, b, a, e\}$),
- y is not adjacent to d (otherwise a fork arises on $\{y, d, e, b, f\}$).

But now a fork arises on $\{x, y, c, b, d\}$, a contradiction.

1.2: y is not adjacent to c. Then

- y is not adjacent to b (otherwise a fork arises on $\{x, y, b, a, c\}$),
- y is not adjacent to e (otherwise a fork arises on $\{y, e, b, a, c\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, c, e\}$), and, by symmetry, to f.

Case 2: y is adjacent to every vertex of degree 2 in H. Then y is adjacent to b, since otherwise a fork would arise on $\{x, y, a, d, b\}$. By symmetry, y is adjacent to e.

Therefore, every neighbor of x is adjacent either to all vertices of H (Case 2) or to none of them (Case 1.2), and the claim for the case $H = H_1$ follows.

II. $H = H_2$.

Case 1: y is adjacent to a vertex of degree 2 in H. Taking into account the symmetry, we may assume without loss of generality that y is adjacent to c. We consider two subcases.

1.1: y is adjacent to f. Then

- y is adjacent to d (otherwise a fork arises on $\{d, c, y, f, x\}$,

- y is adjacent to e (otherwise a fork arises on $\{e, f, y, c, x\}$,
- y is adjacent to b (otherwise a fork arises on $\{b, c, y, e, x\}$),
- y is adjacent to a (otherwise a fork arises on $\{a, b, y, e, x\}$).

1.2: y is not adjacent to f. Then

- y is not adjacent to a (otherwise a fork arises on $\{f, a, y, c, x\}$),
- y is not adjacent to e (otherwise a fork arises on $\{f, e, y, c, x\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, a, e\}$).

But now, a fork arises on $\{y, c, d, a, e\}$, a contradiction.

Case 2. y has no neighbor among the vertices of degree 2 in H. Then

- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, c, e\}$),
- y is not adjacent to a (otherwise a fork arises on $\{y, a, d, c, e\}$),
- y is not adjacent to b (otherwise a fork arises on $\{x, y, b, a, c\}$), and, by symmetry, to f.

Therefore, every neighbor of x is adjacent either to all vertices of H (Case 1.1) or to none of them (Case 2), and the claim for the case $H = H_2$ follows.

III. $H = H_3$.

Case 1: y is adjacent to d. We consider two subcases.

1.1: y is adjacent to f. Then

- y is adjacent to a (otherwise a fork arises on $\{a, f, y, d, x\}$),
- y is adjacent to c (otherwise a fork arises on $\{c, d, y, f, x\}$),
- y is adjacent to b (otherwise a fork arises on $\{b, c, y, f, x\}$),
- y is adjacent to e (otherwise a fork arises on $\{e, c, y, a, x\}$).

1.2: y is not adjacent to f. Then

- y is not adjacent to a (otherwise a fork arises on $\{f, a, y, d, x\}$),
- y is not adjacent to b (otherwise a fork arises on $\{a, b, y, d, x\}$),
- y is not adjacent to e (otherwise a fork arises on $\{x, y, e, b, f\}$).

But now, a fork arises on $\{y, d, e, b, f\}$, a contradiction.

Case 2. y is not adjacent to d. We consider two subcases.

2.1: y is adjacent to f. Then

- y is not adjacent to c (otherwise a fork arises on $\{d, c, y, f, x\}$),
- y is not adjacent to b (otherwise a fork arises on $\{c, b, y, f, x\}$).
- y is not adjacent to e (otherwise a fork arises on $\{x, y, e, b, d\}$).

But now, a fork arises on $\{y, f, e, b, d\}$, a contradiction.

2.2: y is not adjacent to f. Then

- y is not adjacent to e (otherwise a fork arises on $\{x, y, e, d, f\}$),
- y is not adjacent to b (otherwise a fork arises on $\{y, b, e, d, f\}$),
- y is not adjacent to a (otherwise a fork arises on $\{x, y, a, b, f\}$),
- y is not adjacent to c (otherwise a fork arises on $\{x, y, c, b, d\}$).

Therefore, every neighbor of x is adjacent either to all vertices of H (Case 1.1) or to none of them (Case 2.2), and the claim for the case $H = H_3$ follows.

IV. $H = H_4$.

Case 1. y is adjacent to c. We consider two subcases.

1.1: y is adjacent to e. Then

- y is adjacent to b (otherwise a fork arises on $\{b, c, y, e, x\}$),

- y is adjacent to f (otherwise a fork arises on $\{f, e, y, c, x\}$),
- y is adjacent to a (otherwise a fork arises on $\{a, b, y, e, x\}$),
- y is adjacent to d (otherwise a fork arises on $\{d, e, y, b, x\}$).

1.2: y is not adjacent to e. Then

- y is not adjacent to f (otherwise a fork arises on $\{e, f, y, c, x\}$),
- y is not adjacent to a (otherwise a fork arises on $\{f, a, y, c, x\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, a, e\}$).

But now, a fork arises on $\{y, c, d, a, e\}$, a contradiction.

Case 2: y is not adjacent to c. We consider two subcases.

2.1: y is adjacent to e. Then

- y is not adjacent to b (otherwise a fork arises on $\{c, b, y, e, x\}$),
- y is not adjacent to a (otherwise a fork arises on $\{b, a, y, e, x\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, a, c\}$).

But now, a fork arises on $\{y, e, d, a, c\}$, a contradiction.

2.2: y is not adjacent to e. Then

- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, c, e\}$),
- y is not adjacent to a (otherwise a fork arises on $\{y, a, d, c, e\}$),
- y is not adjacent to b (otherwise a fork arises on $\{x, y, b, a, c\}$),
- y is not adjacent to f (otherwise a fork arises on $\{x, y, f, a, e\}$).

Therefore, every neighbor of x is adjacent either to all vertices of H (Case 1.1) or to none of them (Case 2.2), and the claim for the case $H = H_4$ follows.

V. $H = H_5$.

Case 1: y is adjacent to c. We consider two subcases.

1.1: y is adjacent to f. Then

- y is adjacent to d (otherwise a fork arises on $\{d, c, y, f, x\}$), and, by symmetry, to e,
- y is adjacent to b (otherwise a fork arises on $\{b, d, y, f, x\}$),
- y is adjacent to g (otherwise a fork arises on $\{g, d, y, f, x\}$),
- y is adjacent to a (otherwise a fork arises on $\{a, b, y, g, x\}$).

1.2: y is not adjacent to f. Then

- y is not adjacent to e (otherwise a fork arises on $\{f, e, y, c, x\}$),
- y is not adjacent to b (otherwise a fork arises on $\{e, b, y, c, x\}$),
- y is not adjacent to g (otherwise a fork arises on $\{e, g, y, c, x\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, b, g\}$).

But now, a fork arises on $\{y, c, d, b, g\}$, a contradiction.

Case 2. y is not adjacent to c. We may assume that y is not adjacent to f, since otherwise we are in a case symmetric to the case *1.2*. Then

- y is not adjacent to a (otherwise a fork arises on $\{x, y, a, c, f\}$),
- y is not adjacent to b (otherwise a fork arises on $\{y, b, a, c, f\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, b, c\}$), and, by symmetry, to e,
- y is not adjacent to g (otherwise a fork arises on $\{y, g, d, b, c\}$).

Therefore, every neighbor of x is adjacent either to all vertices of H (Case 1.1) or to none of them (Case 2), and the claim for the case $H = H_5$ follows. This completes the proof of the claim.

3.2 $(S_{1,2,5}, banner)$-free Graphs

In this section, we show that the maximum independent set problem is polynomial-time solvable for yet another extension of claw-free graphs, the class of $(S_{1,2,5}, banner)$-free graphs. The presentation follows the one in [80].

The *duplication* of a vertex v in a graph G is the operation of adding a new vertex v' to G with $N(v') = N(v)$. The graph obtained from a path on four vertices P_4 by duplicating one of its middle vertices will be called a *banner*. Notice that a banner contains a claw as an induced subgraph. Therefore, banner-free graphs constitute a generalization of claw-free graphs. However, unlike claw-free graph, the class of banner-free graphs is difficult with respect to the independent set problem, which is an immediate corollary of Theorem 1.1.1.

The class of $(S_{1,2,5}, banner)$-free graphs generalizes not only claw-free graphs but also $(S_{1,2,4}, banner)$-free and $(P_8, banner)$-free graphs studied in [57], as well as $(S_{1,2,3}, banner)$-free, $(P_7, banner)$-free graphs, (P_6, C_4)-free and $(P_5, banner)$-free graphs studied earlier in [5, 92, 78].

The key approach in our solution is the method of finding augmenting graphs. In particular, we will apply results developed in Section 2.1.

Let us recall that for a polynomial-time implementation of the augmenting graph approach in a class of graphs X, one has to

(a) find a complete list of (minimal) augmenting graphs in X,

(b) develop a polynomial-time procedure for detecting augmenting graphs from the list.

We have already seen that in some cases (for example in the case of claw-free graphs), question (a) has a simple answer. Extending the class under consideration leads to more complicated structure of augmenting graphs. For instance, it has been shown in [5] that in the class of $(S_{1,2,3}, banner)$-free graphs (a proper extension of claw-free graphs) a minimal augmenting graph is either

- a chain of even length or
- a complete bipartite graph or

- a simple augmenting tree (graph T_1 in Fig. 3.4) or
- an augmenting plant (graph T_3 with $r = 0$ and $s = 1$ in Fig. 3.4).

Further extension to the class of $(S_{1,2,4}, \mathit{banner})$-free graphs adds only finitely many new minimal augmenting graphs to this list [57]. From the point of view of existence of a polynomial-time algorithm to find augmenting graphs, any finite collection of augmenting graphs can be neglected. Moreover, we do not even need any description of such a collection. As an example exploiting this observation we prove Theorem 3.2.1 below.

We introduce two families of graphs generalizing paths and cycles. Recall that the *duplication* of a vertex v of a graph G results in a graph obtained from G by introducing a new vertex v' with $N(v') = N(v)$. Let us call a *strip* any finite graph obtained from a path by repeatedly performing the duplication of vertices, and a *bracelet* any finite graph obtained in the same manner from a cycle.

Theorem 3.2.1. *For any positive integers k and Δ, there are only finitely many $S_{1,2,k}$-free connected bipartite graphs of maximum degree at most Δ, different from strips and bracelets.*

Proof. Let $l = (d + 1)(n + 2)$. There are only finitely many connected graphs of vertex degree at most d which are P_l-free. Therefore, we assume that a connected bipartite graph G of degree at most d contains a longest induced path $P = (v_1, \ldots, v_r)$ with $r \geq l$.

If $G = P$, then G is a strip. If G is different from P, it must contain a vertex v outside P, which has a neighbor on P.

First, suppose that v has at least three neighbors on P. Since the degree of v is at most d, the neighbors of v divide P into at most $d + 1$ edge-disjoint paths, at least one of which has many edges. Then an induced $S_{1,2,n}$ can be easily found.

Second, suppose that every vertex outside P has at most one neighbor on P. If $v \notin V(P)$ has a neighbor v_i on P, then either $i = 2$ or $i = r - 1$, since otherwise either P is not a longest path or G contains an induced $S_{1,2,n}$. Suppose that $i = 2$. To avoid an induced $S_{1,2,n}$, we conclude that v_2 is the only neighbor of v. By symmetry, the same argument holds for neighbors of v_{r-1}. Therefore, G is a strip in this case.

Third, assume that v has two neighbors on P, say v_i, v_j with $i < j$. Then either $|i-j| = 2$ or $i = 1$ and $j = r$, since otherwise (similarly as above) an induced $S_{1,2,n}$ arises.

The above discussion allows us to conclude that every vertex of G outside P has a neighbor on P, since otherwise one can find an induced $S_{1,2,n}$ in G. To complete the proof, we distinguish between the two following cases.

Case 1. A vertex $v \notin V(P)$ is adjacent to v_1 and v_r, i.e., P together with v induce a cycle C. To see that G must be a bracelet, consider an induced bracelet Q in G that contains C and has as many vertices as possible. For a vertex z of the bracelet, let us denote by $Q(z)$ the set of all vertices w of Q satisfying $N_Q(w) = N_Q(z)$. If $G \neq Q$, then there is a vertex u of G outside Q that has a neighbor in Q, say x. Let C' be a cycle in Q through x such that $|V(C')| = |V(C)|$. Let y be a neighbor of x on C', and let z be the neighbor of y on C', different from x. From our previous observations, we can conclude without loss of generality that the only two neighbors of u on C' are x and z, and further that $N_Q(u) \subseteq Q(x) \cup Q(z)$.

The vertex u is adjacent to all vertices of $Q(x)$, since otherwise G would contain an induced $S_{1,2,n}$, centered at z (and containing u, y and $x' \in Q(x) \backslash N(u)$). By the same token, u is adjacent to all vertices of $Q(z)$. Therefore, $N_Q(u) = Q(x) \cup Q(z)$. However, the set $V(Q) \cup \{u\}$ then induces a bracelet Q' with $|V(Q')| > |V(Q)|$, contradicting our choice of Q.

Case 2. Every vertex v of G outside P is adjacent either to v_2 or to v_{r-1} or to two vertices at distance 2 in P. In other words, v is a duplicate of a vertex of P. To see that G must be a strip, consider an induced strip Q in G that contains P and has as many vertices as possible. Similarly as above, for a vertex z of Q we denote by $Q(z)$ the set of all vertices w of Q satisfying $N_Q(w) = N_Q(z)$. If $G \neq Q$, then Q has a neighbor $u \in V(G) \backslash V(Q)$. Clearly, all the longest paths in Q appear symmetrically in Q. We may therefore assume that the neighbors of u in Q appear only among duplicates of neighbors of u on P (or we are in one of the previously considered cases). That is, $N_Q(u) \subseteq \cup_{v \in N_P(u)} Q(v)$. It follows that $N_Q(u) = \cup_{v \in N_P(u)} Q(v)$, since otherwise it is easy to find an induced $S_{1,2,n}$, in each of the possible cases: $N_P(u) = \{v_2\}$, $N_P(u) = \{v_{r-1}\}$ or $N_P(u) = \{v_i, v_{i+2}\}$ for some i. But now $V(Q) \cup \{u\}$ induces a strip Q' with $|V(Q')| > |V(Q)|$, contradicting the choice of Q. This contradiction completes the proof of the theorem. □

The following simple lemma can be found in [5].

Lemma 3.2.2. *A connected bipartite banner-free graph containing a C_4 is complete bipartite.*

According to this lemma, the problem of finding augmenting graphs in the class under consideration splits into two subproblems:

(A) finding $(S_{1,2,5}, C_4)$-free augmenting graphs;

(B) finding complete bipartite augmenting graphs.

A solution to problem (B) in the class of *banner*-free graphs has been proposed in [5]. In the rest of this section we analyze problem (A). To this end, we further decompose it into two subproblems:

(A.1) finding $(S_{1,2,5}, C_4)$-free augmenting graphs of bounded vertex degree;

(A.2) finding $(S_{1,2,5}, C_4)$-free augmenting graphs containing a vertex of high degree.

From Theorem 3.2.1 we derive the following conclusion.

Corollary 3.2.3. *In the class of $(S_{1,2,5}, C_4)$-free graphs there are finitely many minimal augmenting graphs of bounded vertex degree different from chains.*

Proof. Let H be an $(S_{1,2,5}, C_4)$-free minimal augmenting graph. According to Theorem 3.2.1, we can assume without loss of generality that H is either a strip or a bracelet. Notice that a cycle cannot be an augmenting graph and the duplication of any vertex of a cycle leads to an induced C_4. Therefore, H is a strip. We assume that H is obtained from a path P by duplicating some (possibly none) vertices of P. As before, no vertex of degree 2 on P can be duplicated, since otherwise a C_4 would arise. And if an endpoint of P was duplicated, then H is not a minimal augmenting graph. Therefore, $H = P$ is an augmenting chain. □

Finding augmenting chains in $(S_{1,2,k}, banner)$-free graphs is a polynomially solvable task for any fixed k [71]. Therefore, we proceed to subproblem (A.2). First of all, let us show that without loss of generality we may restrict ourselves to augmenting graphs containing a *black* vertex of high degree.

Let us point out that in order to check that a bipartite C_4-free graph induced by vertices $a, b, c, d, e, f, g, h, i$ with edges $(a, b), (b, c), (c, d), (d, e), (e, f), (f, g), (g, h), (f, i)$ is an $S_{1,2,5}$, one only needs to check that $(a, f), (a, h), (b, g), (b, i), (c, h)$ are non-edges.

Lemma 3.2.4. *If a minimal augmenting $(S_{1,2,5}, C_4)$-free graph H contains no black vertex of degree more than k, then the degree of each white vertex is at most $2k + 1$.*

Proof. Assume that H contains a white vertex a of degree more than $2k + 1$. Denote by A_j the set of vertices of H at distance j from a. Since H is minimal, at most one vertex of A_1 has no neighbors in A_2, and because of C_4-freeness, every vertex of A_2 has exactly one neighbor in A_1. Therefore $|A_2| \geq 2k+1$. Again by the minimality of H, every vertex of A_2 has a neighbor in A_3. If H contains no black vertex of degree more than k, then $|A_3| \geq 3$.

Suppose A_4 contains a (white) vertex x and let y be its neighbor in A_3. Due to the minimality of H, x has at least one more black neighbor, say z. If $deg(y), deg(z) \leq k$, then A_2 contains a vertex non-adjacent both to y and z, and hence there is an induced $S_{1,2,5}$ in H. Therefore, A_4 is empty.

Let $x \in A_3$. By Lemma 2.1.4 and Hall's theorem [64] we know that the subgraph $H - x$ has a perfect matching M. For a subset $U \subset V(H - x)$ of vertices of the same color, we denote by $m(U)$ the set of vertices of the opposite color matched with vertices of U with respect to M. Denote $A := A_1$, $B := m(A)$, $C := A_2 - B$, $D := m(C)$. If $deg(x) \leq k$, B contains at least $k + 1$ vertices each of which has a neighbor in D.

Consider a vertex $d_1 \in D$ such that $m(a_1) \neq a$, where a_1 is the only neighbor of $m(d_1)$ in A. If $deg(d_1) \leq k$, there is a vertex $b \in B$ such that b is not adjacent to d_1, b has a neighbor d_2 in D, and $m(b)$ is not adjacent to $m(d_1)$.

Assume now that there is an edge $m(d_2)d_1$. Then obviously $m(d_1)$ is not adjacent to d_2. If $deg(d_1), deg(d_2) \leq k$, then vertices d_1, d_2 have at most $2k-2$ neighbors in B, while vertices $m(d_1), m(d_2)$ have at most 2 neighbors in A. Thus, there is a couple of vertices $a_2, a_3 \in A$ such that $m(a_2) \neq a$, $m(a_2)$ is non-adjacent to d_1, d_2 and there are no edges between a_2, a_3 and $m(d_1), m(d_2)$. But now the vertices $d_2, m(d_2), d_1, m(d_1), a_1, a, a_2, m(a_2), a_3$ induce an $S_{1,2,5}$ in H. This contradiction shows that $m(d_2)$ cannot be adjacent to d_1.

An analogous argument shows that $m(d_1)$ is not adjacent to d_2. But now, the vertices

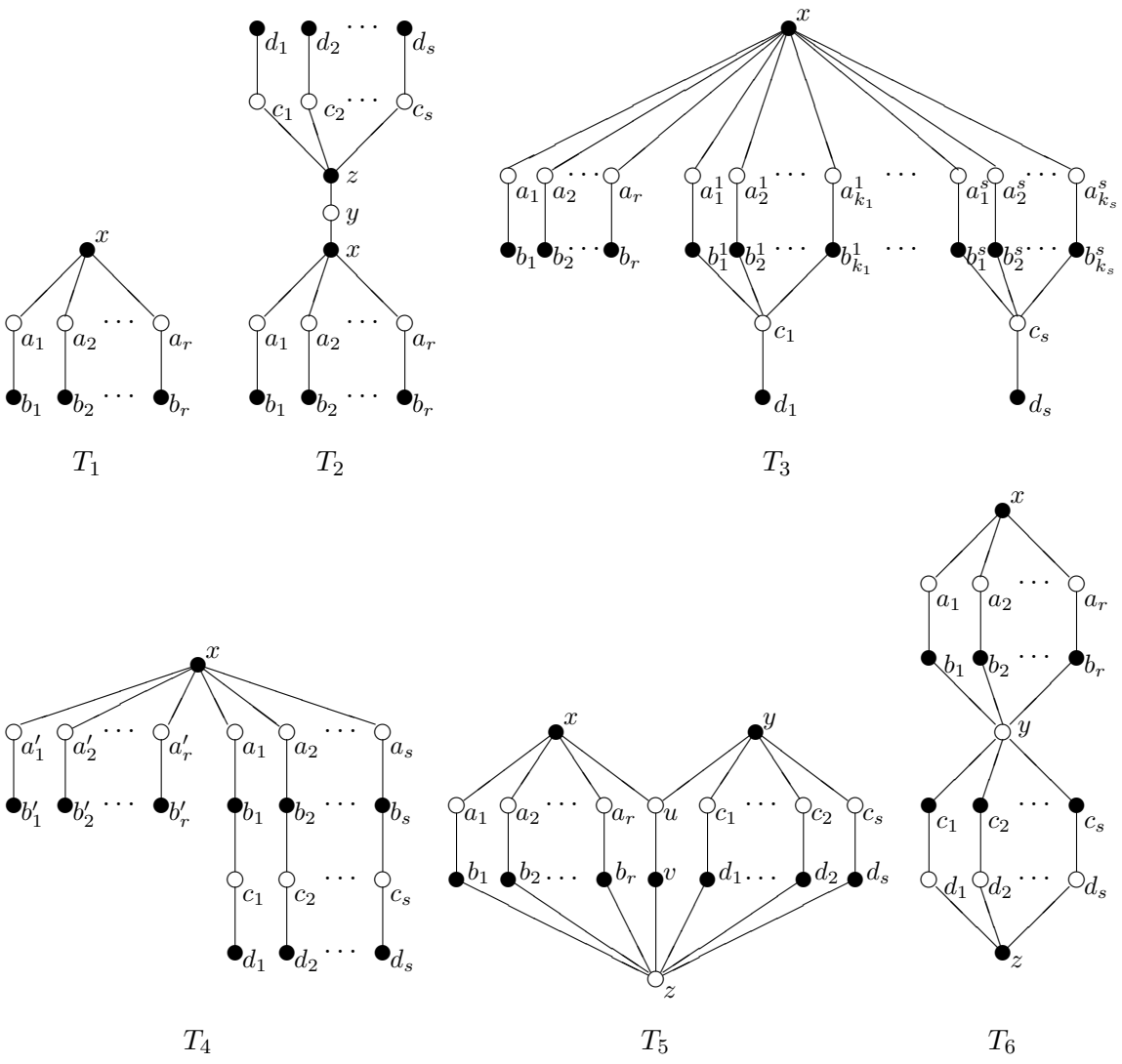

Figure 3.4: "Basic" families of augmenting $(S_{1,2,5}, banner)$-free graphs

$m(d_2), d_2, b, m(b), a, a_1, m(d_1), d_1, m(a_1)$ induce an $S_{1,2,5}$. This contradiction completes the proof of the lemma. □

The above lemma permits us to restrict ourselves to augmenting graphs containing a black vertex x of "sufficiently large" degree k. Figure 3.4 represents all "basic" families of augmenting graphs of this type. The meaning of the word "basic" in the above sentence is specified in the following theorem.

Theorem 3.2.5. *Let G be an $(S_{1,2,5}, banner)$-free graph. The problem of finding in G a*

minimal augmenting $(S_{1,2,5}, C_4)$-free graph with a black vertex of degree at least 6 can be reduced in polynomial time to the problem of finding in G one of the augmenting graphs $T_1, \ldots, T_6$ represented in Figure 3.4.

3.2.1 Proof of Theorem 3.2.5

Throughout the section we shall denote by H a minimal augmenting $(S_{1,2,5}, C_4)$-free graph with a black vertex x of degree $k \geq 6$, by $A = \{a_1, \ldots, a_k\}$ the neighborhood of x and by C the remaining white vertices of H, i.e., those that are not in A. Note that, due to the C_4-freeness of H, every neighbor of A except x has exactly one neighbor in A.

According to Lemma 2.1.4 and Hall's theorem [64], the subgraph $H - x$ has a perfect matching. For a subset of vertices $U \subseteq V(H - x)$ of the same color, we shall denote by $m(U)$ the set of vertices of the opposite color matched with the vertices in U. In particular, $B := m(A)$, and $D := m(C)$. Also, let C_1 denote the set of vertices in C that have at least one and at most $k-1$ neighbors in B, and C_0 the set of vertices in C that have no neighbors in B. Finally, $D_0 := m(C_0)$ and $D_1 := m(C_1)$. Since H is C_4-free, we know that

(0) $C - (C_0 \cup C_1)$ contains at most one vertex, and any vertex of D is adjacent to at most one vertex of A.

Lemma 3.2.6. *If H contains a vertex $y \in C$ which is adjacent to every vertex of B, then either $H = T_5$ or $H = T_6$ or H contains a redundant set U of size at most 10 such that either $H - U = T_1$, $H - U = T_4$ or $H - U = T_6$.*

Proof. Assume first that $C_1 \neq \emptyset$. Since H is C_4-free we know that

(1) y has no neighbors in D_1, and $m(y)$ has no neighbors in $A \cup C_1$;

(2) every vertex of C_1 has exactly one neighbor in B.

Also, from $S_{1,2,5}$-freeness of H we can derive that

(3) Any two vertices of C_1 have different neighbors in B. Indeed, $c_1, c_2 \in C_1$ both are adjacent to $b \in B$, then for any $a_1, a_2 \in A$ different from $m(b)$ and non-adjacent to $m(c_1)$, the subgraph induced by vertices $a_1, x, a_2, m(a_2), y, b, c_1, m(c_1), c_2$ is isomorphic to $S_{1,2,5}$.

(4) $H[C_1 \cup D_1]$ is an induced matching. Indeed, if for a couple of vertices $c_1, c_2 \in C_1$ there is an edge $c_1 m(c_2)$, then for the neighbor $b_1 \in B$ of c_1 and a vertex $b_2 \in B$ such that b_2 is non-adjacent to c_1, c_2 and $m(b_2)$ is non-adjacent to $m(c_1), m(c_2)$, the subgraph induced by vertices $b_2, m(b_2), x, m(b_1), b_1, c_1, m(c_1), c_2, m(c_2)$ is isomorphic to $S_{1,2,5}$.

(5) No vertex of C_0 has a neighbor in D_1. Indeed, if $c \in C_0$ is adjacent to $d \in D_1$, then for the vertex $b \in B$ adjacent to $m(d)$ and any two vertices $a_1, a_2 \in A$ different from $m(b)$ and non-adjacent to d, the subgraph induced by vertices $c, d, m(d), b, m(b), x, a_1$, $m(a_1), a_2$ is isomorphic to $S_{1,2,5}$.

(6) No vertex of C_1 has a neighbor in D_0. Indeed, if $c \in C_1$ is adjacent to $d \in D_0$, then for the vertex $b \in B$ adjacent to c and any two vertices $a_1, a_2 \in A$ different from $m(b)$ and non-adjacent to d, the subgraph induced by vertices $m(d), d, c, b, m(b), x, a_1, m(a_1), a_2$ is isomorphic to $S_{1,2,5}$.

(7) $|C_0| \leq 1$. Indeed, assume $|C_0| \geq 2$. Due to the minimality of H and (5), C_0 must have a vertex z adjacent to $m(y)$. Then no other vertex of C_0 is adjacent to $m(y)$ by analogy with (3).

Since $C_1 \neq \emptyset$, we may consider a vertex $c_1 \in C_1$ and its neighbor $b \in B$.

If z has a neighbor $d \in D_0$ different from $m(z)$, then H contains an $S_{1,2,5}$ induced by vertices $m(c_1), c_1, b, y, m(y), z, d, m(d), m(z)$. Therefore, $m(z)$ is the only neighbor of z in D_0.

If no vertex of C_0 other than z is adjacent to $m(z)$, then $|C_0 - z| = |N(C_0 - z)|$, which contradicts the minimality of H. Therefore, there is a vertex $c_0 \in C_0$ different from z which is adjacent to $m(z)$.

If $m(z)$ is not adjacent to $m(b)$, then H contains an $S_{1,2,5}$ induced by vertices $c_0, m(z)$, $z, m(y), y, b, c_1, m(c_1), m(b)$. Therefore, $m(z)$ is adjacent to $m(b)$, which, together with the C_4-freeness of H implies that $|C_1| \leq 1$.

If $m(c_0)$ is not adjacent to y, then H contains an $S_{1,2,5}$ induced by vertices $m(c_0), c_0$, $m(z), z, m(y), y, b', m(b'), b$ where $b' \in B$ such that $m(b')$ is non-adjacent to $m(z)$ and $m(c_0)$. Therefore, $m(c_0)$ is adjacent to y.

If $m(c_0)$ has a neighbor $c' \in C_0$ different from c_0, then H contains an $S_{1,2,5}$ induced by vertices $a, x, m(b'), b', y, m(c_0), c_0, m(z), c'$, where $b' \in B$ such that $m(b')$ is non-adjacent to $m(z)$ and $m(c_0)$, and $a \in A$ is different from $m(b')$. Therefore, c_0 is the only neighbor of $m(c_0)$ in C_0.

These observations, together with the minimality of H, imply that $m(z)$ is adjacent to all vertices of C_0. Indeed, if the set $C_0' := C_0 - N(m(z))$ is nonempty, then $N(C_0') = m(C_0')$, and consequently $|N(C_0')| = |C_0'|$, contradicting the minimality of H.

If $m(c_1)$ has a neighbor in A, say a, then H contains an $S_{1,2,5}$ induced by vertices $m(z), z, m(y), y, m(a), a, m(c_1), c_1, x$. Therefore, $m(c_1)$ has no neighbors in A. It is now not hard to verify that the set $U := \{c_1, m(c_1)\}$ is a redundant subset of size 2 such that $H - U = T_6$.

(8) If $|C_1| \geq 2$, then no vertex of D_1 has a neighbor in A. To the contrary, assume a vertex $d \in D_1$ has a neighbor $a \in A$, and let $c \in C_1$ be a vertex different from $m(d)$. Also, denote by $b_1 \in B$ the neighbor of $m(d)$ and by $b_2 \in B$ the neighbor of c. Then vertices $x, a, d, m(d), b_1, y, b_2, c, m(y)$ induce an $S_{1,2,5}$ in H.

From the above list of claims we conclude that if $|C_1| \geq 2$, then $U_1 := \{y, m(y)\} \cup C_0 \cup D_0 \cup N_A(D_0) \cup m(N_A(D_0)) \cup N_{C_1}(m(N_A(D_0))) \cup m(N_{C_1}(m(N_A(D_0))))$ is a redundant subset of size at most 8 such that $H - U_1 = T_4$. If $|C_1| \leq 1$, then $U_2 := U_1 \cup C_1 \cup D_1 \cup N_A(D_1) \cup m(N_A(D_1))$ is a redundant subset of size at most 10 such that $H - U_2 = T_1$.

Now assume that $C_1 = \emptyset$. If in addition $C_0 = \emptyset$, then $H - \{y, m(y)\} = T_1$.

If $C_0 \neq \emptyset$, then due to the minimality of H there must exist a vertex $z \in C_0$ adjacent to $m(y)$. Then

- no other vertex of C_0 is adjacent to $m(y)$ by analogy with (3).
- y is not adjacent to $m(z)$.

Denote by C_0' the set of vertices of $C_0 - \{z\}$ adjacent to $m(z)$ and let $C_0'' := C_0 - (C_0' \cup \{z\})$. Then y is adjacent to every vertex in $m(C_0')$, since otherwise for any vertex $d \in m(C_0')$ non-adjacent to y, any vertex $a \in A$ non-adjacent both to $m(z)$ and d, and any vertex $b \in B$

different from $m(a)$, we have $H[d, m(d), m(z), z, m(y), y, m(a), a, b] = S_{1,2,5}$. Consequently, no vertex of $m(C_0')$ has a neighbor in A, since otherwise a C_4 arises.

If $C_0'' \neq \emptyset$, then it must contain a vertex c that has a neighbor $d \in m(C_0')$, since otherwise $|N(C_0'')| = |C_0''|$. But then for any two vertices $a_1, a_2 \in A$ non-adjacent to d and $m(c)$, we have $H[a_1, x, a_2, m(a_2), y, d, c, m(c), m(d)] = S_{1,2,5}$. Therefore, $C_0'' = \emptyset$. Now if $m(z)$ has no neighbors in A then $H = T_6$, and if $m(z)$ has a neighbor in A then $H = T_5$. □

From now on, we assume that $C = C_0 \cup C_1$, i.e., every vertex of C has a non-neighbor in B. This implies

Claim 3.2.7. *Every vertex of D_1 has at most one neighbor in C_0.*

Proof. Assume a vertex $d_1 \in D_1$ has at least two neighbors $c_0', c_0'' \in C_0$. Denote $c_1 := m(d_1)$, $B' := N(c_1) \cap B$ and $B'' := B - B'$. By definition, $B' \neq \emptyset$ and $B'' \neq \emptyset$. Let $a_1 \in m(B')$ and a_2, a_3 be any two other vertices of A non-adjacent to d_1. Without loss of generality assume that a_2 is not adjacent to $m(c_0')$. If in addition a_1 is not adjacent to $m(c_0')$, then $H[a_2, x, a_1, m(a_1), c_1, d_1, c_0', m(c_0'), c_0''] = S_{1,2,5}$. If a_1 is adjacent to $m(c_0')$, then $H[c_0'', d_1, c_0', m(c_0'), a_1, x, a_2, m(a_2), a_3] = S_{1,2,5}$. This contradiction shows that every vertex of D_1 has at most one neighbor in C_0. □

Claim 3.2.8. *Every vertex of C_0 has at least one neighbor in D_1.*

Proof. Denote by C_0' the vertices of C_0 that have neighbors in D_1 and $C_0'' := C_0 - C_0'$. If C_0'' is not empty, it must contain a vertex c_0'' that has a neighbor $d_0' \in m(C_0')$, since otherwise $|C_0''| = |N(C_0'')|$ contradicting the minimality of H. Denote $c_0' := m(d_0')$, $d_1 \in D_1$ a neighbor of c_0', $c_1 := m(d_1)$, $B' := N(c_1) \cap B$ and $B'' := B - B'$. If d_1 has a neighbor $a_1 \in A$, then for any two vertices $a_2, a_3 \in A$ different from a_1 and non-adjacent to d_0', $H[c_0'', d_0', c_0', d_1, a_1, x, a_2, m(a_2), a_3] = S_{1,2,5}$. If d_1 has no neighbors in A, then for any $a_1 \in m(B')$, $a_2 \in m(B'')$ and $a_3 \in A$ different from a_1, $H[c_0', d_1, c_1, m(a_1), a_1, x, a_2, m(a_2), a_3] = S_{1,2,5}$. Therefore, $C'' = \emptyset$ and the claim is proved. □

A natural consequence of the two preceding claims is the following corollary.

Corollary 3.2.9. $|C_0| \leq |C_1|$.

Lemma 3.2.10. *If $|C_1| \leq 3$, then H contains a redundant set U of size at most 24 such that $H - U = T_1$.*

Proof. Let $|C_1| \leq 3$. The above corollary then implies $|C| \leq 6$, and therefore also $|D| \leq 6$. Due to the C_4-freeness of H, every vertex of D has at most one neighbor in A, so that $|N_A(D)| \leq 6$. Now it is easy to see that the set $U := C \cup D \cup N_A(D) \cup m(N_A(D))$ is a redundant set of size at most 24 such that $H - U = T_1$. □

From now on we assume that $|C_1| \geq 4$.

Lemma 3.2.11. *Let $|C_1| \geq 4$ and $C_0 = \emptyset$.*

(a) If there are vertices $y, z \in C_1$ such that $N_B(y) \cap N_B(z) \neq \emptyset$, then $H = T_2$ or $H = T_5$ or H contains a redundant set U of size at most 2 such that $H - U = T_2$.

(b) If for any two vertices $y, z \in C_1$, $N_B(y) \cap N_B(z) = \emptyset$, then $H = T_3$ or H contains a redundant set U of size at most 4 such that $H - U = T_3$.

Proof. To prove (a), denote by b_1 the common neighbor of y and z in B.

Case 1. Assume first that y has one more neighbor in B, say b_2. Suppose B contains a vertex b_3 non-adjacent both to y and z. Then $m(b_3)$ is adjacent to $m(y)$, since otherwise $H[b_3, m(b_3), x, m(b_2), y, b_1, z, m(y)] = S_{1,2,5}$. This implies that $m(b_3)$ is adjacent to $m(z)$, since otherwise $H[m(z), z, b_1, y, m(y), m(b_3), x, a, b_3] = S_{1,2,5}$, where $a \in A$ is a vertex non-adjacent to $b_1, m(y), m(z)$. Therefore, B contains at most one vertex adjacent neither to y nor to z, since otherwise a C_4 arises. From this and the fact that $|B| \geq 6$ we conclude without loss of generality that y has at least three neighbors in B. But now the vertices $b_3, m(b_3), m(z), z, b_1, y, b_2, m(b_2), b_4$ induce an $S_{1,2,5}$ in H, where b_4 is one more neighbor of y. This contradiction shows that $N_B(y) \cup N_B(z) = B$.

Let $c \in C_1 \backslash \{y, z\}$. Then c is not adjacent to b_1. Suppose the converse, and let $b \in B$ be a vertex adjacent to y but not to z such that $m(b)$ is not adjacent to $m(c)$. Then, H contains an $S_{1,2,5}$ induced by vertices $a, x, m(b), b, y, b_1, c, m(c), z$ where $a \in A$ is a vertex non-adjacent to $m(c)$ and different from $m(b), m(b_1)$.

By symmetry, we may assume that c is adjacent to a vertex $b \in B$ that is adjacent to y but not to z. For every vertex $b' \in N_B(z) - N_B(y)$ that is not adjacent to c, $m(y)$ must be adjacent to $m(b')$, since otherwise $H[b', m(b'), x, m(b_1), b_1, y, b, c, m(y)] = S_{1,2,5}$. Due to the C_4-freeness of H, this implies that c is non-adjacent to at most one vertex in $N_B(z) - N_B(y)$. The C_4-freeness of H also implies that c is adjacent to at most one vertex in $N_B(z) - N_B(y)$. Therefore, $|N_B(z) - N_B(y)| \leq 2$, and consequently $|N_B(y) - N_B(z)| \geq 3$.

Let $b' \in B$ be a non-neighbor of y. Then $b' \in N_B(z) - N_B(y)$. Let $b'' \in B$ denote a neighbor of y different from b and b_1 such that $m(b'')$ is non-adjacent to $m(z)$. (Such a vertex exists since $|N_B(y) - N_B(z)| \geq 3$ and since $m(z)$ is adjacent to at most one neighbor of x.) If b' is adjacent to c, then $H[b'', m(b''), x, m(b_1), b_1, z, b', c, m(z)] = S_{1,2,5}$, and if b' is non-adjacent to c, then $H[z, b', m(b'), x, m(b), b, y, b'', c] = S_{1,2,5}$. This contradiction implies that y is adjacent to all vertices of B, which in turn contradicts the fact that $y \in C_1$.

Case 2. Now we assume that $N_B(c) = \{b_1\}$ for any vertex $c \in C_1$ adjacent to b_1. Then for any such a vertex, $m(c)$ has no neighbors in A. Indeed, if $a_1 \in A$ is a neighbor of $m(c)$, then for any vertices $a_2, a_3 \in A$ different from $m(b_1)$ and non-adjacent to $m(c)$, we have $H[c', b_1, c, m(c), a_1, x, a_3, m(a_3), a_2] = S_{1,2,5}$, where c' is another vertex of C_1 adjacent to b_1. Therefore, if every vertex of C_1 is adjacent to b_1, then $H = T_2$.

Suppose now that C_1 has a vertex c' non-adjacent to b_1, and let b_2 denote a neighbor of c' in B. Then c' is adjacent to $m(y)$ (and similarly to $m(c)$ for every vertex $c \in C_1$ adjacent to b_1), since otherwise $H[c', b_2, m(b_2), x, m(b_1), b_1, y, m(y), z] = S_{1,2,5}$. Consequently, c' is adjacent to every vertex $b \in B$ different from b_1, since otherwise $H[b, m(b), x, m(b_2), b_2, c', m(y), y, m(z)] = S_{1,2,5}$. Together with C_4-freeness of H this implies that c' is the only vertex of C_1 non-adjacent to b_1. If in addition $m(c')$ has no neighbors in A, then $\{c', m(c')\}$ is a redundant set in H and $H - \{c', m(c')\} = T_2$. If $m(c')$ has a neighbor in A, then this neighbor must be $m(b_1)$, in which case $H = T_5$.

Now we proceed to the proof of (b). If for each vertex $d \in D_1$, the vertex $m(d)$ is the only neighbor of d, then $H = T_3$. Therefore, we assume that a vertex $d \in D_1$ has a neighbor $y \neq m(d)$.

Let first y belong to C_1. Since $|C_1| \geq 4$, there is a vertex $a \in A$ such that a is

not adjacent to d, and $m(a)$ is adjacent neither to $m(d)$ nor to y. But then for any neighbor $b \in B$ of $m(d)$ and any vertex $a' \in A - \{a, m(b)\}$ non-adjacent to d, we have $H[y, d, m(d), b, m(b), x, a, m(a), a'] = S_{1,2,5}$.

Now assume that $y \in A$. Let first y be the only neighbor of $m(y)$, i.e., let $\{y, m(y)\}$ be a redundant set. If there is a vertex $d' \in D_1$ non-adjacent to y, then H contains an $S_{1,2,5}$ induced by vertices $d', m(d'), b, m(b), x, y, d, m(d), m(y)$, where $b \in B$ is a neighbor of $m(d')$. If every vertex of D_1 is adjacent to y, then they have no other neighbors in A and hence $H - \{y, m(y)\} = T_3$.

Now suppose that $m(y)$ is adjacent to a vertex $c \in C_1$. Then every vertex d' of D_1 (different from $m(c)$) also is adjacent to y, since otherwise for any neighbor $b \in B$ of $m(d')$, we have $H[d', m(d'), b, m(b), x, y, d, m(d), m(y)] = S_{1,2,5}$. Therefore, if $\{y, m(y), c, m(c)\}$ is a redundant set in H, then $H - \{y, m(y), c, m(c)\} = T_3$. In order to show that the set $\{y, m(y), c, m(c)\}$ is redundant, assume by contradiction that $m(c)$ has a neighbor $a \in A$. Without loss of generality let $m(a)$ be non-adjacent to $m(d)$. But then $H[m(a), a, m(c), c, m(y), y, d, m(d), d'] = S_{1,2,5}$, where d' is any vertex of D_1 different from d and $m(c)$. □

Lemma 3.2.12. *Let $|C_1| \geq 4$ and $C_0 \neq \emptyset$. Then, the set $U := C_0 \cup D_0$ is a redundant set of size 2 such that $H - U = T_5$.*

Proof. Assume that $C_0 \neq \emptyset$. Then it must contain a vertex c which has a neighbor d in D_1, since otherwise $|N(C_0)| = |C_0|$. Let $b \in B$ be a non-neighbor and $b' \in B$ a neighbor of $m(d)$. If d is not adjacent to $m(b)$, then for any vertex $a \in A$ different from $m(b), m(b')$ and non-adjacent to d, we have $H[c, d, m(d), b', m(b'), x, m(b), b, a] = S_{1,2,5}$.

Now assume that for any non-neighbor $b \in B$ of $m(d)$, the vertex d is adjacent to $m(b)$. Since d may have at most one neighbor in A, we conclude that $m(d)$ is adjacent to all but one vertex b in B.

This implies that $|C_0| \leq 1$. Indeed, suppose C_0 contains one more vertex, say c'. Then due to the minimality of H, either c' has a neighbor in D_1 or we can assume without loss of generality that c' is adjacent to $m(c)$. If c' has a neighbor $d' \in D_1$ different from d, then analogously $m(d')$ is adjacent to all but one vertex of B, which leads to an induced C_4, since $|B| \geq 6$. If c' is adjacent to d, then for any two vertices $a_1, a_2 \in A$ non-adjacent to d and $m(c)$

such that $m(a_2)$ is adjacent to $m(d)$, we have $H[a_1, x, a_2, m(a_2), m(d), d, c, m(c), c'] = S_{1,2,5}$. Finally, if c' is adjacent to $m(c)$, then for a vertex $a_1 \in A$ non-adjacent to $m(c), m(c')$ and such that $m(a_1)$ is adjacent to $m(d)$, we have $H[m(c'), c', m(c), c, d, m(d), m(a_1), a_1, b'] = S_{1,2,5}$, where $b' \in B$ is adjacent to $m(d)$. This contradiction shows that $C_0 = \{c\}$.

Assume $m(c)$ has a neighbor $c_1 \in C_1$. Obviously $c_1 \neq m(d)$ and c_1 is not adjacent to d. In addition, c_1 has a neighbor $y \neq m(c)$ non-adjacent to $m(d)$. Indeed, if c_1 is adjacent to b then $y = b$, and if c_1 is not adjacent to b then it has a neighbor in $B - \{b\}$, in which case $y = m(c_1)$. But then $H[y, c_1, m(c), c, d, m(d), b', m(b'), b''] = S_{1,2,5}$, where b' and b'' are two vertices in $B - \{b\}$ non-adjacent to c_1 (observe that c_1 may have at most one neighbor in $B - \{b\}$) such that $m(b')$ is not adjacent to y and $m(c)$. This contradiction shows that $m(c)$ has no neighbors in C_1.

Assume now that $m(c)$ is adjacent to a vertex $a \in A$ different from $m(b)$. Since $|C_1| \geq 4$, we may consider three vertices $c_1, c_2, c_3 \in C_1$ different from $m(d)$. Remember that c_j $(j = 1, 2, 3)$ must have a neighbor in B, but it cannot have more than one neighbor in $B - \{b\}$. To avoid an induced C_4 we conclude that $m(c_j)$ cannot be adjacent both to c and a. If $m(c_j)$ is adjacent to a, then for any vertex $b'' \in B - \{b\}$ non-adjacent to c_j we have $H[b'', m(d), d, c, m(c), a, m(c_j), c_j, x] = S_{1,2,5}$. If $m(c_j)$ is adjacent to c, then for any vertex $a' \in A - \{a, m(b)\}$ such that a' is non-adjacent to $m(c_j)$ and $m(a')$ is not adjacent to c_j we have $H[m(a'), a', x, a, m(c), c, m(c_j), c_j, d] = S_{1,2,5}$. Therefore, we conclude that $m(c_j)$ is adjacent neither to a nor to c. If in addition c_j has a neighbor $b' \in B - \{b, m(a)\}$, then $H[c, m(c), a, m(a), m(d), b', c_j, m(c_j), m(b')] = S_{1,2,5}$. If two vertices c_i and c_j are both adjacent to b, then for any vertex $a' \in A - \{m(b)\}$ such that $m(a')$ is not adjacent to c_i, c_j and a' is not adjacent to $m(c_i)$ we have $H[m(d), m(a'), a', x, m(b), b, c_i, m(c_i), c_j] = S_{1,2,5}$. The only case is left is when two vertices c_i and c_j are both adjacent to $m(a)$. Then for any two vertices $a_1, a_2 \in A - \{a, m(b)\}$ non-adjacent to $m(c_i)$ we have $H[a_1, x, a_2, m(a_2), m(d), m(a), c_i, m(c_i), c_j] = S_{1,2,5}$. This contradiction shows that the only possible neighbor of $m(c)$ in A is $m(b)$.

Due to the C_4-freeness of H, $m(c)$ cannot be adjacent to $m(b)$. Therefore, we conclude that c is the only neighbor of $m(c)$.

Assume now that there is a vertex $c_1 \in C_1$ such that c_1 and $m(d)$ have a common neighbor in B, say b_1. Then c_1 is not adjacent to b, for otherwise the vertex set $\{m(c), c, d, m(b), b, c_1, b_1, m(b_1), m(c_1)\}$ would induce an $S_{1,2,5}$ in H. Also, $m(c_1)$ is not adjacent to $m(b)$, since otherwise $H[b, m(b), m(c_1), c_1, b_1, m(d), b_2, m(b_2), b_3] = S_{1,2,5}$ for any two distinct vertices b_2, $b_3 \in B - \{b, b_1\}$ such that $m(b_2)$ is not adjacent to $m(c_1)$. But now, an $S_{1,2,5}$ arises on the vertex set $\{m(c_1), c_1, b_1, m(d), d, m(b), x, a_1, b\}$, where $a_1 \in A$ is any vertex different from $m(b_1)$ and non-adjacent to $m(c_1)$. This contradiction shows that $N_B(c_1) = \{b\}$ for every vertex $c_1 \in C_1 - \{m(d)\}$.

Now, observe that for every vertex $c_1 \in C_1 - \{m(d)\}$, $m(c_1)$ is adjacent to $m(d)$. Indeed, if this is not the case, we have $H[m(d), m(a), a, x, m(b), b, c_1, m(c_1), c_2] = S_{1,2,5}$ for any vertex $a \in A$ different than $m(b)$ and non-adjacent to $m(c_1)$ and any $c_2 \in C_1 - \{m(d), c_1\}$. Therefore, we conclude that $H - \{c, m(c)\} = T_5$. □

3.2.2 Finding Augmenting Graphs $T_1, \ldots, T_6$

Now we present polynomial-time algorithms for finding augmenting graphs from the six basic families represented in Figure 3.4. To this end, we first check whether G contains a certain small induced subgraph (a so-called *initial structure*) which is contained in every large enough graph from a family T_i under consideration, and then try to extend it to the whole augmenting graph. For clarity of the proofs, we shall use the labeling of vertices of augmenting graphs $T_1, \ldots, T_6$ as represented in Figure 3.4.

Given a black vertex b, we will denote by $W(b) = N(b) \cap S$ the set of white neighbors of b. For a nonnegative integer i, we denote by B^i the set of all black vertices having exactly i white neighbors. The independence number of G (i.e., the size of a maximum independent set in G) is denoted $\alpha(G)$.

Lemma 3.2.13. *If G contains no augmenting P_3, then a simple augmenting tree T_1 (if any) can be found in polynomial time.*

Proof. If G contains no augmenting P_3 but contains an augmenting T_1, then $r \geq 2$, where r is the number of leaves in T_1 (see Figure 3.4). Therefore, we first have to check if G contains an induced $P_5 = (b_1, a_1, x, a_2, b_2)$ with $\{b_1, b_2\} \subseteq B^1$. If G contains no such an

initial structure, then it contains no augmenting T_1. If such a structure exists, then we proceed as follows.

Let us denote $A = W(x)\backslash\{a_1, a_2\}$, and for $a \in A$, let $K(a)$ denote the set of black neighbors of a which are in B^1, and which are not adjacent to any of $\{x, b_1, b_2\}$. Notice that a desired simple augmenting tree exists only if $K(a) \neq \emptyset$ for all a in A. Finally, let $V' = \cup_{a \in A} K(a)$.

Consider any vertex a in A. If $K(a)$ contains two non-adjacent vertices b and b', then b, a and b' induce an augmenting P_3 in G, a contradiction. Hence, each $K(a)$ induces a clique in G. It follows that a desired simple augmenting tree exists if and only if $\alpha(G[V']) = |A|$.

It is easy to see that $G[V']$ must be P_5-free. Indeed, consider an induced $P_4 = (p_1, p_2, p_3, p_4)$ in $G[V']$, and let $a \in A$ be such that $p_1 \in K(a)$. None of the vertices p_3 and p_4 is adjacent to a, since $K(a)$ is a clique. But now, $p_2 \in K(a)$, since otherwise the vertex set $\{p_4, p_3, p_2, p_1, a, x, a_2, b_2, a_1\}$ induces an $S_{1,2,5}$ in G. Hence, if $G[V']$ contains a $P_4 = (p_1, p_2, p_3, p_4)$, then p_1 and p_2 have a common white neighbor, while p_2 and p_3 do not have a common white neighbor. This implies that $G[V']$ is P_5-free.

Since the independence number of a (P_5, banner)-free graph can be computed in polynomial time (see e.g. [78, 57]), we can efficiently compute $\alpha = \alpha(G[V'])$. If $\alpha < |A|$, we conclude that G contains no simple augmenting tree containing the above initial structure. Otherwise, we may choose one vertex from each clique $K(a)$ to obtain a simple augmenting tree. □

Lemma 3.2.14. *If G contains no augmenting P_3 or P_7, then an augmenting T_2 (if any) can be found in polynomial time.*

Proof. We may restrict ourselves to finding a T_2 with $r, s \geq 2$, since any T_2 with, say, $r = 1$ either equals to P_7 or contains a redundant subset $U = \{a_1, b_1\}$ such that $T_2 - U = T_1$.

As an initial structure, consider the subgraph of T_2 (see Figure 3.4) induced by vertices $a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2, x, y, z$ such that $\{b_1, b_2, d_1, d_2\} \subseteq B^1$.

Let us denote $A = (W(x) \cup W(z))\backslash\{a_1, a_2, c_1, c_2, y\}$, and for $a \in A$, let $K(a)$ denote the set of black neighbors of a which are in B^1, and which are not adjacent to any of $\{x, z, b_1, b_2, d_1, d_2\}$. Note that due to the C_4-freeness of the augmenting graph, $(W(x) \cap$

$A) \cap (W(z) \cap A) = \emptyset$, and that a desired augmenting T_2 exists only if $K(a) \neq \emptyset$ for all a in A. Finally, let $V' = \cup_{a \in A} K(a)$.

Consider any vertex a in A. If $K(a)$ contains two non-adjacent vertices b and b', then b, a and b' induce an augmenting P_3 in G, a contradiction. Hence, each $K(a)$ induces a clique in G. It follows that a desired augmenting T_2 exists if and only if $\alpha(G[V']) = |A|$.

We now show that $G[V']$ is P_3-free. Suppose, to the contrary, that (p_1, p_2, p_3) is an induced P_3 in $G[V']$. Let $a \in A$ be such that $p_1 \in K(a)$. Since $K(a)$ is a clique, p_3 is not adjacent to a. This implies that a_2 is not adjacent to a, since otherwise p_2 and p_3 should have a common white neighbor different from a, which is impossible since $p_2 \in B^1$. Without loss of generality we may assume that a is adjacent to x, but not to z. But now the vertex set $\{p_2, p_1, a, x, y, z, a_2, b_2, a_1\}$ induces an $S_{1,2,5}$ in G, a contradiction.

Hence, $G[V']$ is a disjoint union of cliques, and the independence number $\alpha = \alpha(G[V'])$ can be trivially computed. If $\alpha < |A|$, we conclude that G contains no augmenting T_2 containing the above initial structure. Otherwise, we may choose one vertex from each clique $K(a)$ to obtain an augmenting T_2. □

Lemma 3.2.15. *If G contains no augmenting P_3, then an augmenting T_3 (if any) can be found in polynomial time.*

Proof. We may restrict ourselves to finding a T_3 with $s \geq 2$, since any T_3 with $s \in \{0, 1\}$ is either a simple augmenting tree T_1 or contains a redundant subset U of size 2 such that $T_3 - U = T_1$.

As an initial structure, consider the subgraph of T_3 (see Figure 3.4) induced by vertices $d_1, c_1, b_1^1, a_1^1, x, a_1^2, b_1^2, c_2, d_2$ such that $\{b_1^1, b_1^2\} \subseteq B^2$ and $\{d_1, d_2\} \subseteq B^1$.

Let us denote $A = W(x) \backslash \{a_1^1, a_1^2\}$, and for $a \in A$, let $K(a)$ denote the set of black neighbors b of a which are in $B^1 \cup B^2$, which are not adjacent to any of $\{x, b_1^1, b_1^2, d_1, d_2\}$, and such that if $b \in B^2$ then G contains a pair of adjacent vertices $c(b)$ and $d(b)$ such that $W(b) = \{a, c(b)\}$, $d(b) \in B^1$, and $d(b)$ is not adjacent to any of $\{x, b_1^1, b_1^2, d_1, d_2, b\}$.

Note that due to the C_4-freeness of the augmenting graph, the sets $K(a_1)$ and $K(a_2)$ are disjoint for any two distinct $a_1, a_2 \in A$, and that a desired augmenting T_3 exists only if $K(a) \neq \emptyset$ for all a in A. Finally, let $V' = \cup_{a \in A} K(a)$.

Consider any vertex a in A. Suppose $K(a)$ contains two non-adjacent vertices b and b'. If $b, b' \in B^1$, then b, a and b' induce an augmenting P_3 in G, a contradiction. Now assume $b \in B^2$, and let $\{c(b), d(b)\}$ be a pair of adjacent vertices such that $W(b) = \{a, c(b)\}$, $d(b) \in B^1$, and $d(b)$ is not adjacent to any of $\{x, b_1^1, b_1^2, d_1, d_2, b\}$. Since $b \in B^2$, it cannot be adjacent to both c_1 and c_2; without loss of generality we may assume that b is not adjacent to c_1, i.e., $c(b) \neq c_1$. But now the vertex set $\{d_1, c_1, b_1^1, a_1^1, x, a, b, c(b), b'\}$ induces an $S_{1,2,5}$ in G, a contradiction.

Therefore, each $K(a)$ induces a clique in G. It follows that $\alpha(G[V']) = |A|$ is a necessary condition for the existence of an augmenting T_3 extending the initial structure.

Let us now show that $G[V']$ must be P_5-free. Indeed, consider an induced $P_4 = (p_1, p_2, p_3, p_4)$ in $G[V']$, and let $a \in A$ be such that $p_1 \in K(a)$. None of the vertices p_3 and p_4 is adjacent to a, since $K(a)$ is a clique. But now, $p_2 \in K(a)$, since otherwise the vertex set $\{p_4, p_3, p_2, p_1, a, x, a_1^2, b_1^2, a_1^1\}$ induces an $S_{1,2,5}$ in G. Hence, if $G[V']$ contains a $P_4 = (p_1, p_2, p_3, p_4)$, then p_1 and p_2 have a common white neighbor, while p_2 and p_3 do not have a common white neighbor. This implies that $G[V']$ is P_5-free.

Since the stability number of a (P_5, banner)-free graph can be computed in polynomial time, we can efficiently compute $\alpha = \alpha(G[V'])$. If $\alpha < |A|$, we conclude that G contains no augmenting T_3 containing the above initial structure.

If $\alpha = |A|$, let B denote a maximum independent set in $G[V']$ (in particular, B contains precisely one vertex b from each clique $K(a)$), and let $C = \{c(b) : b \in B \cap B^2\}$ and $D = \{d(b) : b \in B \cap B^2\}$. Consider the induced subgraph H of G, obtained by adding to the initial structure all the vertices from $A \cup B \cup C \cup D$. Now, to see that H is an augmenting T_3 we only need to show that $B \cup D$ is an independent set. The set B was chosen to form an independent set in G. By definition of $d(b)$, no $b \in B \cap B^2$ is adjacent to $d(b)$. Suppose there is an edge connecting a vertex b from B to a $d(b')$ for some $b' \in B \cap B^2$. Let a' denote the unique common white neighbor of x and b'. Now, the vertex set $\{b, d(b'), c(b'), b', a', x, a_1^1, b_1^1, a_1^2\}$ induces an $S_{1,2,5}$ in G, a contradiction. Similarly, if there is an edge connecting vertices $d, d' \in D$, then the vertex set $\{d, d', c(b'), b', a', x, a_1^1, b_1^1, a_1^2\}$ (where $d' = d(b')$ and a' is the unique common white neighbor of x and b') induces an $S_{1,2,5}$ in G. This contradiction completes the proof of the lemma. $\square$

Lemma 3.2.16. *If G contains no augmenting P_3, then an augmenting T_4 (if any) can be found in polynomial time.*

Proof. This follows immediately from the fact that every T_4 is a special case of a T_3, and from the proof of Lemma 3.2.15. □

Lemma 3.2.17. *An augmenting T_5 (if any) can be found in polynomial time.*

Proof. We may restrict ourselves to finding a T_5 with $r, s > 0$ and $r \geq 2$, since a T_5 with $r = 0$ contains a redundant set U of size 4 such that $T_5 - U = T_1$, and a T_5 with $r = s = 1$ can be found in polynomial time.

As an initial structure, consider the subgraph of T_5 (see Figure 3.4) induced by vertices $a_1, a_2, b_1, b_2, c_1, d_1, u, v, x, y, z$ such that $\{b_1, b_2, v, d_1\} \subseteq B^2$.

Let us denote $A_x = W(x)\backslash\{a_1, a_2, u\}$, $A_y = W(y)\backslash\{u, c_1\}$, and for $a \in A := A_x \cup A_y$, let $K(a)$ denote the set of common neighbors of a and z which are in B^2, and which are not adjacent to any of $\{x, y, b_1, b_2, v, d_1\}$.

Note that it follows from the C_4-freeness of the augmenting graph that the sets A_x and A_y are disjoint, and that for every $a \in A$, $K(a)$ is a clique. Finally, let $V'_x = \cup_{a \in A_x} K(a)$, $V'_y = \cup_{a \in A_y} K(a)$, and $V' = V'_x \cup V'_y$. From the definition of the sets $K(a)$ it follows that $V'_x \cap V'_y = \emptyset$. Moreover, a desired augmenting T_5 exists if and only if $\alpha(G[V']) = |A|$.

Let us show that $G[V']$ is a disjoint union of cliques. First, we observe that each of $G[V'_x]$ and $G[V'_y]$ is a disjoint union of cliques. Indeed, suppose that (p_1, p_2, p_3) is an induced P_3 in $G[V'_x]$. Let $a \in A_x$ be such that $p_1 \in K(a)$. Since $K(a)$ is a clique, p_3 is not adjacent to a. This implies that p_2 is not adjacent to a, since otherwise p_2 and p_3 should have a common white neighbor different from a, which is impossible since $p_2 \in B^2$ and $W(p_2) = \{a, z\}$. But now the vertex set $\{p_3, p_2, p_1, a, x, u, v, y, c_1\}$ induces an $S_{1,2,5}$ in G, a contradiction. Also, there are no edges between V'_x and V'_y, for if there are vertices $a \in A_x$ and $a' \in A_y$ with $bb' \in E$ for some $b \in K(a)$ and $b' \in K(a')$, then the vertex set $\{y, a', b', b, a, x, a_1, b_1, a_2\}$ induces an $S_{1,2,5}$ in G.

Hence, the independence number $\alpha = \alpha(G[V'])$ can be trivially computed. If $\alpha < |A|$, we conclude that G contains no augmenting T_5 containing the above initial structure.

Otherwise, we may choose one vertex from each clique $K(a)$ to obtain an augmenting T_5. □

Lemma 3.2.18. *If G contains no augmenting P_3 or P_7, then an augmenting T_6 (if any) can be found in polynomial time.*

Proof. We may restrict ourselves to finding a T_6 with $r \geq 2$, since a T_6 with $r = s = 1$ is a P_7.

As an initial structure, consider the subgraph of T_6 (see Figure 3.4) induced by vertices $a_1, a_2, b_1, b_2, c_1, d_1, x, y, z$ such that $\{b_1, b_2, c_1\} \subseteq B^2$ and such that x and z have no common white neighbors.

Let us denote $A_x = W(x)\backslash\{a_1, a_2\}$, $A_z = W(z)\backslash\{d_1\}$ and for $a \in A := A_x \cup A_z$, let $K(a)$ denote the set of common neighbors of a and y which are in B^2, and which are not adjacent to any of $\{x, b_1, b_2, c_1, z\}$.

Note that $A_x \cap A_z = \emptyset$ by assumption. Also, due to the C_4-freeness of the augmenting graph, each $K(a)$ for $a \in A$ is a clique. Finally, let $V'_x = \cup_{a \in A_x} K(a)$, $V'_z = \cup_{a \in A_z} K(a)$, and $V' = V'_x \cup V'_z$. It follows that a desired augmenting T_6 exists only if $\alpha(G[V']) = |A|$.

Let us show that $G[V']$ is a $S_{1,1,2}$-free graph. Indeed, suppose that $\{p_1, p_2, p_3, p_4, p_5\}$ induces an $S_{1,1,2}$ in $G[V']$ (with a P_4 on $\{p_1, p_2, p_3, p_4\}$ and an additional edge p_2p_5), and let $a \in A$ be such that $p_1 \in K(a)$. Since $K(a)$ is a clique, none of p_3, p_4, p_5 is adjacent to a. Now, p_2 must be adjacent to a, or an $S_{1,2,5}$ arises on the vertex set $\{b_1, a_1, x, a, p_1, p_2, p_3, p_4, p_5\}$ (if $a \in A_x$) or on the vertex set $\{d_1, c_1, z, a, p_1, p_2, p_3, p_4, p_5\}$ (if $a \in A_z$). By symmetry, p_2 and p_5 also share a common white neighbor $a' \in A$ different from a. This means that p_2 has at least three white neighbors y, a, a', contradicting the assumption that p_2 belongs to B^2.

Since the independence number of an $S_{1,1,2}$-free graph can be computed in polynomial time (see e.g. [3]), we can efficiently compute $\alpha = \alpha(G[V'])$. If $\alpha < |A|$, we conclude that G contains no augmenting T_6 containing the above initial structure. Otherwise, we may choose one vertex from each clique $K(a)$ to obtain an augmenting T_6. □

As a consequence of the above lemmas and the results from Section 2.1.1, we obtain the following conclusion.

Theorem 3.2.19 ([80]). *The maximum independent set problem can be solved in polynomial time in the class of* $(S_{1,2,5}, banner)$*-free graphs.*

3.3 $mS_{1,k,k}$-free Graphs of Bounded Vertex Degree

In this section, we assume that the input graphs are of maximum degree at most Δ, for some fixed $\Delta \geq 3$. Forbidding what graphs of the form $S_{i,j,k}$ will result in an "easy" class? One of the simplest examples of graphs of the form $S_{i,j,k}$ is a chordless path. Trivially, in a class of graphs of bounded vertex degree excluding a path P_k (for any natural k), the MAXIMUM WEIGHT INDEPENDENT SET problem is solvable in polynomial time, as every such class contains only finitely many connected graphs. Moreover, from Theorem 4.1.1 from Section 4.1.1 we can conclude that the problem is solvable in polynomial time also for $S_{1,k,k}$-free graphs of bounded vertex degree, since $S_{1,k,k}$ is contained in every apple of order more than $2k + 1$.

The case of P_k-free graphs can be easily generalized to mP_k-free graphs (for any natural m), as mP_k is an induced subgraph of P_l with $l \geq m(k + 1) - 1$. However, a generalization of $S_{1,k,k}$-free graphs to $mS_{1,k,k}$-free graphs is not so obvious. More generally, we are not aware of any argument showing that whenever the MAXIMUM WEIGHT INDEPENDENT SET problem is polynomial-time solvable for F-free graphs, where F is a graph different from a chordless path, the same conclusion holds for mF-free graphs. For graphs of bounded vertex degree, however, this is always the case.

Theorem 3.3.1 ([82]). *Let F be a graph and let Δ, m be positive integers. If the* MAXIMUM WEIGHT INDEPENDENT SET *problem is solvable in polynomial time for F-free graphs of vertex degree at most Δ, then it is also solvable in polynomial time for mF-free graphs of vertex degree at most Δ.*

Proof. We prove the theorem by induction on m. The basis of the induction is obvious. Let $m > 1$ and G be an mF-free graph of vertex degree at most Δ. If G is F-free, then we are done. Otherwise, let G contain an induced copy of F. By deleting the vertices of F together with their neighbors, we obtain an induced subgraph G' of G which is $(m - 1)F$-free. Since the vertex degree in G is at most Δ, the number of deleted vertices does not

exceed $|V(F)|(\Delta+1)$. By the induction hypothesis, the MAXIMUM WEIGHT INDEPENDENT SET problem can be solved in polynomial time for $(m-1)F$-free graphs. Therefore, by Theorem 2.4.1, the problem admits a polynomial-time solution for G too. □

Therefore, the following corollary holds.

Corollary 3.3.2 ([82]). *For every fixed k, m and Δ, the* MAXIMUM WEIGHT INDEPENDENT SET *problem is solvable in polynomial time for $mS_{1,k,k}$-free graphs of maximum degree at most Δ.*

3.4 Planar and More General Graphs

In this section, we turn our attention to the MAXIMUM INDEPENDENT SET problem in planar graphs and more generally, graphs that exclude a fixed apex graph as a minor. An *apex graph* is a graph that becomes planar after deleting one vertex. By Wagner's Theorem [112], planar graphs are characterized by the absence of two apex graphs $K_{3,3}$ or K_5 as minors. Therefore, for every nonplanar apex graph H, the class of H-minor-free graphs strictly contains planar graphs.

3.4.1 Forbidding a Graph from $\mathcal{S}$ and a Line Graph of a Graph from $\mathcal{S}$

Let $\mathcal{T} = \{L(G) : G \in \mathcal{S}\}$ denote the set of all line graphs of graphs in $\mathcal{S}$. The goal of this section is to show Theorem 3.4.6, the analogue of the following theorem from [85] obtained by replacing the requirement on bounded maximum degree with the property "excludes H as a minor," where H a fixed apex graph.

Theorem 3.4.1 ([85]). *For any positive integer Δ and any two graphs $H_1 \in \mathcal{S}$ and $H_2 \in \mathcal{T}$ there is a number N such that every graph of vertex degree at most Δ with no induced subgraphs isomorphic to H_1 or H_2 has treewidth at most N.*

In fact, many results valid for graphs of bounded degree remain true for planar graphs. Consider, for instance, the following obvious observation: connected graphs of bounded degree and bounded diameter have bounded treewidth (as there are only finitely many such

graphs).[2] The same is true for planar graphs: planar graphs of bounded diameter have bounded treewidth.

A class of graphs is said to have the *diameter-treewidth property* if there is a function associated with the class such that the treewidth of any graph from the class is bounded above by the function of its diameter. Clearly, for any such class, bounded diameter implies bounded treewidth. It has been shown by Eppstein [47] that among minor-closed graph classes, the diameter-treewidth property is enjoyed precisely by the classes excluding an apex graph.

Theorem 3.4.2 ([47], [38])**.** *Let X be a minor-closed graph family. Then X has the diameter-treewidth property if and only if X does not contain all apex graphs.*

The proof found by Eppstein has been substantially simplified by Demaine and Hajiaghayi in [38]. The key tool for this simplification is Lemma 3.4.3 below. An $n \times n$ grid G_n is the graph with the vertex set $\{1, \ldots, n\} \times \{1, \ldots, n\}$ such that (i, j) and (k, l) are adjacent if and only if $|i - k| + |j - l| = 1$. An *augmented grid* is a grid G_n augmented with additional edges (and no additional vertices). Vertices (i, j) with $\{i, j\} \cap \{1, n\} \neq \emptyset$ are *boundary vertices* of the grid; the other ones are *nonboundary*.

Lemma 3.4.3. *Let H be an apex graph, let G be H-minor-free graph, let $r = 14|V(H)|-22$, and let $m > 2r$ be the largest integer such that $\mathrm{tw}(G) \geq m^{4|V(H)|^2(m+2)}$. Then G be can be contracted into an augmented grid R, that is, an $(m - 2r) \times (m - 2r)$ grid augmented with additional edges (and no additional vertices) such that each vertex $v \in V(R)$ is adjacent to less than $(r + 1)^6$ nonboundary vertices of the grid.*

From this lemma we can conclude that if H is an apex graph, and a subclass X of H-minor-free graphs is closed under edge contractions and vertex deletions, then the treewidth of graphs in X is bounded if and only if it is bounded for graphs of bounded vertex degree in X. More formally:

Corollary 3.4.4 ([74])**.** *Let H be an apex graph and X a subclass of H-minor-free graphs. Denote by Y the class of all graphs that can be obtained from graphs in X by applying*

[2]The *diameter* of a connected graph G is the maximum distance between two vertices in G. The *distance* between two vertices u and v in G is the length of a shortest u-v path in G.

a sequence of (zero or more) edge contractions and vertex deletions. Suppose that the treewidth of graphs in Y is bounded above by a function of their maximum degree: there exists a function f such that

$$\mathrm{tw}(G) \leq f(\Delta(G)), \quad \textit{for all} \quad G \in Y\,. \tag{3.1}$$

Then, there is a number N such that every graph in X has treewidth at most N.

Proof. For $\Delta \geq 0$, let $Y_\Delta := \{G \in Y : \Delta(G) \leq \Delta\}$. Let $r = 14|V(H)| - 22$, and consider a graph $G \in X$ with $\mathrm{tw}(G) \geq (2r+1)^{4|V(H)|^2(2r+3)}$. (If there is no such graph, then the treewidth of graphs in X is bounded and we are done.)

Lemma 3.4.3 implies that G can be contracted into an $(m-2r) \times (m-2r)$ augmented grid R such that each vertex $v \in V(R)$ is adjacent to less than $(r+1)^6$ nonboundary vertices of the grid, where $m > 2r$ is the largest integer such that $\mathrm{tw}(G) \geq m^{4|V(H)|^2(m+2)}$. Let R_0 denote the subgraph of R induced by the nonboundary vertices of R. Since $R_0 \in Y_{(r+1)^6}$, it follows from (3.1) that the treewidth of R_0 is at most $f((r+1)^6)$. As the tree-width of a minor of a graph never exceeds the treewidth of the graph, and since the treewidth of an $n \times n$ grid is n, we conclude that $m - 2r - 1 \leq \mathrm{tw}(R_0) \leq f((r+1)^6)$, implying $m+1 \leq f((r+1)^6) + 2r + 2 =: C$. This inequality and the choice of m then imply that the treewidth of G is bounded above by the constant $C^{4|V(H)|^2(C+2)}$. □

In order to obtain an analog of Theorem 3.4.1, we have to prove that for any $H_1 \in \mathcal{S}$ and $H_2 \in \mathcal{T}$, the class of (H_1, H_2)-free graphs is closed under edge contractions. Unfortunately, this is not generally true. However, we have the following result, where by nT we denote the graph consisting of n disjoint copies of a graph T.

Lemma 3.4.5. *Let G be an $(nS_{k,k,k}, nT_{k,k,k})$-free graph, and let G' be a graph, obtained from G by a sequence of (zero or more) edge contractions or vertex deletions. Then, G' is $((2n-1)S_{k+1,k+1,k+1}, (2n-1)T_{k+1,k+1,k+1})$-free.*

Proof. If we apply a sequence of edge contractions or vertex deletions to a graph, the isomorphism type of the obtained graph does not depend on the order of operations. Moreover, the class of $((2n-1)S_{k+1,k+1,k+1}, (2n-1)T_{k+1,k+1,k+1})$-free graphs is closed under vertex

deletions. Therefore it suffices to show that the graphs obtained from an $(nS_{k,k,k}, nT_{k,k,k})$-free graph by applying a sequence of edge contractions are $((2n-1)S_{k+1,k+1,k+1}, (2n-1)T_{k+1,k+1,k+1})$-free.

Suppose not. Let $\{G_i\}_{i=0}^{m}$ be a shortest sequence of graphs with $G_0 \in Free(nS_{k,k,k}, nT_{k,k,k})$ such that for each $i \in \{1,\dots,m\}$, G_i is obtained from G_{i-1} by contracting an edge, and $G_m \notin Free((2n-1)S_{k+1,k+1,k+1}, (2n-1)T_{k+1,k+1,k+1})$.

Suppose that G_m contains an induced copy S of $(2n-1)S_{k+1,k+1,k+1}$. Since G_m was obtained from G_0 by a sequence of edge contractions, there exist pairwise disjoint subsets $\{V_x \subseteq V(G_0) : x \in V(S)\}$ of $V(G_0)$ such that each V_x induces a connected graph in G_0, and, for all pairs of distinct vertices $x, x' \in V(S)$, there is an edge in G_0 joining a vertex from V_x and a vertex from $V_{x'}$ if and only if $\{x, x'\}$ is an edge of S.

For each $i \in \{1,\dots,2n-1\}$, let $S^{(i)}$ denote the i-th copy of $S_{k+1,k+1,k+1}$ in S, let $x^{(i)}$ be the vertex of degree 3 in $S^{(i)}$, let $R^{(i)} = V(S^{(i)})\backslash\{x^{(i)}\}$, and finally let $R = \cup_{i=0}^{2n-1} R^{(i)}$. Then, $|V_x| = 1$ for all $x \in R$. Indeed, assume that $|V_x| \geq 2$ for some $x \in R$. As G_m is (up to isomorphism) independent of the order of edge contractions, we may assume that a contraction corresponding to x occurred in the last step. Let $\{x', x''\}$ be the edge of G_{m-1} which was contracted into x. Then, as $N_{G_{m-1}}(\{x', x''\}) = N_{G_m}(x)$, a simple consideration shows that G_{m-1} contains an induced copy of $(2n-1)S_{k+1,k+1,k+1}$. But this contradicts the minimality of m.

So we can simply assume that $R \subseteq V(G_0)$. Consider now an arbitrary copy S' of a $S_{k+1,k+1,k+1}$ in S, say $S' = S^{(1)}$. Let $\{a, b, c\} = N_{S'}(x^{(1)})$, let P be an induced b–c path in $G_0[\{b, c\} \cup V_{x^{(1)}}]$, and let Q be a shortest a–P path in $G_0[\{a\} \cup V_{x^{(1)}}]$. Note that such paths exist since $G_m[\{b, x^{(1)}, c\}]$ contains a b–c path $P' = (b, x^{(1)}, c)$, and $G_m[\{a, x^{(1)}\}]$ contains an a–P' path $Q' = (a, x^{(1)})$.

Let $Q = (v_0, v_1, \dots, v_{r_1})$ with $v_0 = a$ and $v_{r_1} \in V(P)$. By minimality of $|Q|$, the only vertex of Q which has a neighbor on P is v_{r_1-1}. If v_{r_1} is the only neighbor of v_{r-1} on P, then $G_0[V_{x^{(1)}} \cup R]$ contains an induced $S_{k,k,k}$, centered at v_{r_1}. If v_{r_1} has two nonadjacent neighbors on P, then $G_0[V_{x^{(1)}} \cup R]$ contains an induced $S_{k,k,k}$, centered at v_{r_1-1}. Finally, if v_{r_1-1} has precisely two neighbors on P, which are adjacent in G_0, then $G_0[V_{x^{(1)}} \cup R]$ contains an induced $T_{k,k,k}$.

The above arguments apply to $S^{(i)}$ for $i > 1$ as well. Therefore, each $G_0[V_{x^{(i)}} \cup R^{(i)}]$ contains either an induced $S_{k,k,k}$ or an induced $T_{k,k,k}$. As there are no edges between $V_{x^{(i)}} \cup R^{(i)}$ and $V_{x^{(j)}} \cup R^{(j)}$ for $i \neq j$, this implies that G_0 contains either an induced $nS_{k,k,k}$ or an induced $nT_{k,k,k}$. This contradiction shows that G_m is $(2n-1)S_{k+1,k+1,k+1}$-free.

Therefore, G_m must contain an induced copy T of $(2n-1)T_{k+1,k+1,k+1}$. Similarly as before, there exist pairwise disjoint subsets $\{V_x \subseteq V(G_0) : x \in V(T)\}$ of $V(G_0)$ such that each V_x induces a connected graph in G_0, and, for all pairs of distinct vertices $x, x' \in V(T)$, there is an edge in G_0 joining a vertex from V_x and a vertex from $V_{x'}$ if and only if $\{x, x'\}$ is an edge of T.

For each $i \in \{1, \ldots, 2n-1\}$, let $T^{(i)}$ denote the i-th copy of $T_{k+1,k+1,k+1}$ in T, let $x^{(i)}, y^{(i)}, z^{(i)}$ be the vertices of degree 3 in $T^{(i)}$, let $R^{(i)} = V(T^{(i)}) \backslash \{x^{(i)}, y^{(i)}, z^{(i)}\}$, and finally let $R = \cup_{i=0}^{2n-1} R^{(i)}$.

As before, $|V_x| = 1$ for all $x \in R$ (for otherwise G_{m-1} would contain an induced copy of $(2n-1)T_{k+1,k+1,k+1}$, contradicting the minimality of m).

So we can simply assume that $R \subseteq V(G_0)$. Again, assume for the moment that $i = 1$. Let a, b, c be the respective neighbors of degree 2 of $x^{(1)}, y^{(1)}, z^{(1)}$ in $T^{(1)}$, let P be an induced b–c path in $G_0[\{b, c\} \cup V_{y^{(1)}} \cup V_{z^{(1)}}]$, and let Q be a shortest a–P path in $G_0[\{a\} \cup V_{x^{(1)}} \cup V_{y^{(1)}} \cup V_{z^{(1)}}]$. Note that such paths exist since $G_m[\{b, y^{(1)}, z^{(1)}, c\}]$ contains a b–c path $P' = (b, y^{(1)}, z^{(1)}, c)$, and $G_m[\{a, x^{(1)}, y^{(1)}, z^{(1)}\}]$ contains an a–P' path $Q' = (a, x^{(1)}, y^{(1)})$.

Let $Q = (v_0, v_1, \ldots, v_{r_1})$ with $v_0 = a$ and $v_{r_1} \in V(P)$. By minimality of $|Q|$, the only vertex of Q which has a neighbor on P is v_{r_1-1}. If v_{r_1} is the only neighbor of v_{r-1} on P, then $G_0[V_x \cup R]$ contains an induced $S_{k,k,k}$, centered at v_{r_1}. If v_{r_1} has two nonadjacent neighbors on P, then $G_0[V_x \cup R]$ contains an induced $S_{k,k,k}$, centered at v_{r_1-1}. Finally, if v_{r_1-1} has precisely two neighbors on P, which are adjacent in G_0, then $G_0[V_x \cup R]$ contains an induced $T_{k,k,k}$.

Therefore, each $G_0[V_{x^{(i)}} \cup R^{(i)}]$ contains either an induced $S_{k,k,k}$ or an induced $T_{k,k,k}$. As there are no edges between $V_{x^{(i)}} \cup R^{(i)}$ and $V_{x^{(j)}} \cup R^{(j)}$ for $i \neq j$, this implies that G_0 contains either an induced $nS_{k,k,k}$ or an induced $nT_{k,k,k}$. This contradiction completes the proof of the lemma. □

Obviously, every graph $H_1 \in \mathcal{S}$ is an induced subgraph of $nS_{k,k,k}$ for some n and k, and every graph $H_2 \in \mathcal{T}$ is an induced subgraph of $nT_{k,k,k}$ for some n and k. Therefore, summarizing the above discussion we conclude that

Theorem 3.4.6 ([74])**.** *For any apex graph H and any two graphs $H_1 \in \mathcal{S}$ and $H_2 \in \mathcal{T}$ there is a number N such that every H-minor-free graph with no induced subgraphs isomorphic to H_1 or H_2 has treewidth at most N.*

Clearly, this implies the following corollary.

Corollary 3.4.7. *For any apex graph H and any two graphs $H_1 \in \mathcal{S}$ and $H_2 \in \mathcal{T}$, the* MAXIMUM WEIGHT INDEPENDENT SET *problem admits a linear-time solution in the class of H-minor-free graphs with no induced subgraphs isomorphic to H_1 or H_2.*

3.4.2 $S_{1,2,k}$-free $K_{3,3}$-minor-free Graphs

In this section, we provide an increasing family of subclasses of planar graphs where the maximum independent set problem admits a polynomial-time solution. More generally, for every fixed value of k, we show that the maximum independent set problem is polynomially solvable in the class of $K_{3,3}$-minor-free graphs with no induced subgraphs isomorphic to $S_{1,2,k}$.

Theorem 3.4.8 ([83])**.** *For any positive integer k, the maximum independent set problem is polynomially solvable for $S_{1,2,k}$-free $K_{3,3}$-minor-free graphs.*

The rest of this section is devoted to the proof of this result. The solution combines the technique of finding augmenting graphs, reduction to 2-connected components, and boundedness of the treewidth. First, we reduce the problem in polynomial time from the class under consideration to the smaller class of $S_{1,2,2}$-free graphs excluding $K_{3,3}$ as a minor. Then we develop a polynomial-time algorithm for the maximum independent set problem in $S_{1,2,2}$-free $K_{3,3}$-minor-free graphs.

Reduction to $S_{1,2,2}$-free $K_{3,3}$-minor-free Graphs

Recall from Section 2.3 that the maximum (weight) independent set problem can be reduced in polynomial-time to graphs without clique separators. In the special case when the graph

is decomposed by cliques of size 1, the problem reduces to 2-connected graphs. Thus, without loss of generality we consider only 2-connected graphs in our class. The following auxiliary result will prove useful for our reduction.

Lemma 3.4.9. *For any integer $k \geq 2$, there exists a constant D_k such that for every 2-connected $S_{1,2,k}$-free $K_{3,3}$-minor-free graph G that contains an induced copy of $S_{1,2,2}$, one has*

$$diam(G) \leq D_k\,.$$

Proof. Consider an induced copy H of $S_{1,2,2}$ in a 2-connected $S_{1,2,k}$-free $K_{3,3}$-minor-free graph G. Let $V(H) = \{a, b, c, d, e, f\}$ and $E(H) = \{ab, ac, ad, ce, df\}$. We will show that no vertex in G has distance greater than k from $V(H)$. Assume, by contradiction, that there is a vertex v at distance $k+1$ from $V(H)$, and let P be a shortest path connecting v to H. If one of the following three cases occurs

- the vertex of P closest to H is not adjacent to b,
- the vertex of P closest to H is not adjacent to c and e,
- the vertex of P closest to H is not adjacent to d and f,

then by simple inspection one can easily find an $S_{1,2,k}$ induced by some vertices of H and P. If none of these cases occurs, then due to 2-connectivity of G we may consider a path P' from v to H avoiding the vertex of P closest to H. Again, if one of the above three cases occurs with respect to P', then G contains an induced $S_{1,2,k}$. Otherwise, the vertices of H together with two vertices of P and P' closest to H induce a subgraph containing $K_{3,3}$ as a minor.

This contradiction shows that the eccentricity of a, i.e., of the vertex of degree 3 in H, is bounded:

$$ecc(a) = \max_{v \in V(G)} d(v, a) \leq k + 2\,.$$

By triangle inequality, this implies $diam(G) \leq 2 \cdot ecc(a) \leq 2k + 4$. □

As the treewidth of a $K_{3,3}$-minor-free graph is bounded above by a function of its diameter (c.f. Section 3.4.1), it follows from Lemma 3.4.9 that the treewidth of 2-connected

$S_{1,2,k}$-free $K_{3,3}$-minor-free graphs that contain an induced copy of $S_{1,2,2}$ is bounded above by some constant c_k. Therefore, the maximum independent set problem can be solved efficiently for such graphs. These observations immediately imply the following conclusion.

Corollary 3.4.10. *For any $k \geq 2$, the* MAXIMUM INDEPENDENT SET *problem for $S_{1,2,k}$-free $K_{3,3}$-minor-free graphs is polynomially equivalent to the* MAXIMUM INDEPENDENT SET *problem for $S_{1,2,2}$-free $K_{3,3}$-minor-free graphs.*

$S_{1,2,2}$-free $K_{3,3}$-minor-free Graphs

In this subsection, we present a polynomial-time solution for the maximum independent set problem in the class of $S_{1,2,2}$-free $K_{3,3}$-minor-free graphs. We start with an informal description of the solution.

The main tool in our solution is the technique of augmenting graphs. In order to apply this technique, we characterize minimal augmenting graphs in our class. To this end, we first show that minimal augmenting graphs in this class cannot contain vertices of arbitrarily high degree (Lemma 3.4.11). Then, with the exception of finitely many graphs, we provide a complete characterization of minimal augmenting graphs in the class: these are graphs that arise from paths and cycles by duplication of some vertices (Theorem 3.2.1 from Section 3.2 and Lemma 3.4.15). Next, we show that all such graphs, except for so-called *minimal augmenting strips with an inner twin*, contain a redundant set of constant size whose deletion results in a path (Lemma 3.4.17). Together with the polynomial-time algorithm for finding augmenting paths in $S_{1,2,3}$-free graphs [58], this allows to solve the problem by means of augmentation, unless the input graph contains a minimal augmenting strip with an inner twin.

To get rid of minimal augmenting strips with inner twins, we apply a double transformation of the input graph G as follows. First, we transform G into a weighted graph G' by identifying vertices that share the same neighborhood, and so map the problem to that of finding a maximum weighted independent set in G'. Next, we decompose G' into 2-connected components, and then apply the inverse transformation to the resulting components. It turns out that each 2-connected component of G' transforms in this way into a graph that does not contain minimal augmenting strips with an inner twin. Moreover, as

this graph is isomorphic to an induced subgraph of the original graph, it is again $S_{1,2,2}$-free and does not contain $K_{3,3}$ as a minor.

Now we turn to a formal description of the above procedure. We begin by showing that minimal augmenting graphs in our class cannot contain vertices of arbitrarily high degree.

Lemma 3.4.11. *The maximum degree of minimal augmenting $S_{1,2,2}$-free $K_{3,3}$-minor-free graphs is bounded by a constant.*

Proof. Let $H = (W, B; E)$ be a minimal augmenting $S_{1,2,2}$-free $K_{3,3}$-minor-free graph with the set of white vertices W and the set of black vertices B. The proof consists of two parts.

1. *No black vertex of H has degree more than 9.*

Assume that H contains a black vertex x of degree at least 10. By Lemma 2.1.4 and Hall's Theorem [64], we know that the subgraph $H - x$ has a perfect matching M. For a vertex $v \in V(H - x)$, we denote by $m(v)$ the unique vertex such that $\{v, m(v)\}$ is an edge in M. Also, let A denote the set of neighbors of x, and $A' = \{m(v) : v \in A\}$.

Since H does not contain a $K_{3,3}$ as a subgraph, we have

$$|N_A(u) \cap N_A(v)| \leq 2 \tag{3.2}$$

for all pairs of distinct vertices $u, v \in A'$.

Assume that A' contains two distinct vertices u', v' each of which has at least 5 neighbors in A. Then, according to (3.2), the set $N_A(u')\backslash(N_A(v') \cup \{m(u')\})$ contains at least one vertex, say u; the set $N_A(v')\backslash(N_A(u') \cup N_A(m(u)))$ contains at least one vertex, say v; the set $N_A(u')\backslash(N_A(v') \cup N_A(m(u)))$ contains at least one vertex, say w. But now an $S_{1,2,2}$ arises on $\{x, w, u, m(u), v, v'\}$, contradicting our assumption.

Therefore, A' contains a subset A'' of at least 9 vertices, each of which has at most 4 neighbors in A. Let $u, v \in A''$. Clearly A contains a vertex nonadjacent both to u and v. To avoid an induced $S_{1,2,2}$, we conclude that either $N_A(u) \subseteq N_A(v)$ or $N_A(v) \subseteq N_A(u)$. Therefore, the vertices of A'' admit an ordering $u_1, u_2, \ldots, u_{|A''|}$ such that $N_A(u_{i+1}) \subseteq N_A(u_i)$ for each i. But then $N_A(u_1) \cap N_A(u_2) \supseteq \{m(u_2), m(u_3), m(u_4)\}$, which leads to an induced $K_{3,3}$ in H. This contradiction completes the first part of the proof.

2. *If H contains no black vertex of degree more than $k \geq 2$, then the degree of each white vertex is at most $4k-3$.*

Assume that H contains a white vertex x of degree more than $4k-3$, while no black vertex of H has degree more than $k \geq 2$. Fix an arbitrary neighbor b of x. As before, the subgraph $H-b$ has a perfect matching M. For a subset $U \subset V(H-b)$ of vertices of the same color, we denote by $m(U)$ the set of vertices of the opposite color matched with vertices of U with respect to M. For a vertex $a \in V(H-b)$, let $m(a) := m(\{a\})$. Denote $A_1 := N(x) \backslash \{b, m(x)\}$ and $A_2 := m(A_1) \backslash N(m(x))$.

Since $m(x)$ has at most $k-1$ neighbors in the set $m(A_1)$, it follows that $|A_2| \geq 3k-3$. Now, fix an arbitrary vertex $a \in A_2$, and let $A_3 = A_2 \backslash N(m(a))$. We see that $|A_3| \geq 2k-2$.

Note that a is adjacent to all vertices in $m(A_3)$, since otherwise the vertices $x, m(x), m(a), a$ together with any vertex $v \in m(A_3)$ non-adjacent to a and its neighbor $m(v)$ induce an $S_{1,2,2}$ in H.

Since H does not contain an induced $K_{3,3}$, every vertex of A_3 has at most two neighbors in $m(A_3)$. Now, fix an arbitrary vertex $a' \in A_3$. In particular, given $|A_3| \geq 2k-2$ and the bound on the degree of black vertices, this implies that there is a vertex $a'' \in A_3$ which shares no neighbor with a' in the set $m(A_3)$. But now an $S_{1,2,2}$ arises on the vertex set $\{x, m(x), m(a'), a', m(a''), a''\}$. This contradiction completes the proof of the lemma. □

Now we proceed to characterizing of minimal augmenting graphs in our class that are of bounded vertex degree. Recall from Section 3.2 that a *strip* is any finite graph, obtained from a path by repeatedly performing the duplication of vertices (zero or more times), and a *bracelet* any finite graph obtained in the same manner from a cycle.

Remark. *To avoid confusion in the definitions we will introduce later on, let us agree that we consider graphs of the form $K_{2,n}$ (for $n \geq 2$) as bracelets, but not as strips.*

It follows from Theorem 3.2.1 that for any positive integer Δ, there are only finitely many $S_{1,2,2}$-free connected bipartite graphs of maximum degree at most Δ different from strips and bracelets. Together with Lemma 2.1.4, this leads to the following observation.

Corollary 3.4.12. *There are only finitely many minimal augmenting $S_{1,2,2}$-free $K_{3,3}$-minor-free graphs different from strips and bracelets.*

Thus, the problem of finding augmenting graphs in the class under consideration reduces to finding augmenting strips and bracelets. Now we provide a further specification of the structure of augmenting graphs in our class. To this end, let us introduce some more notations and definitions.

Definition 3.4.13. *Let us call two vertices in a graph G* similar *if they have the same neighborhood in G. A couple of similar vertices will be also called a* twin *in G.*

Clearly, similarity is an equivalence relation. Note that every equivalence class is an independent set. For a vertex $v \in V(G)$, we denote by C_v the equivalence class containing v.

Definition 3.4.14. *Given a graph G and a vertex $v \in V(G)$,*

- *the* thickness of v *is the cardinality of C_v;*
- *the* thickness *of G is the maximum thickness of any vertex of G.*

The following lemma specifies the structure of minimal augmenting strips and bracelets in our class in terms of their thickness.

Lemma 3.4.15. *Let $H = (W, B; E)$ be a minimal $K_{3,3}$-minor-free augmenting strip or bracelet. Then H is either*

- *a strip of thickness at most 2, or*
- *a bracelet obtained from an even cycle by the duplication of exactly one vertex.*

Proof. If $H = K_{2,3}$ (in which case we consider H as a bracelet), the lemma is trivially true. Assume now that $H \neq K_{2,3}$.

Suppose that the thickness of H is at least 3. By Lemma 2.1.4, no subset of white vertices A with the same neighborhood can have cardinality more than 2 (else $A \cup N(A)$ would induce a $K_{3,4}$). Therefore, there is a subset B' of black vertices with the same neighborhood such that $|B'| \geq 3$. By the $K_{3,3}$-minor-freeness and connectedness, we have $1 \leq |N(B')| \leq 2$. Denote $A = W \backslash N(B')$. If A is nonempty, then $|A| \geq |W| - 2$ and $|N(A)| \leq |B| - 3$, which together with Lemma 2.1.4 implies $|A| \geq |N(A)|$, contradicting the

minimality of H. If A is empty, then $H = K_{1,3}$, contradicting the minimality of H again. Thus, we conclude that thickness of H is at most two, which proves the lemma in case when H is a strip.

Assume now that H is a bracelet. Since no cycles are augmenting, H must contain a vertex x of thickness 2. Since H has no minor isomorphic to $K_{3,3}$, no neighbor of x has thickness 2 (else H contain a subdivision of $K_{3,3}$). Therefore, x has exactly 2 neighbors, both of thickness 1. Next, observe that x must be black, since otherwise $A := C(x)$ would violate the inequality $|N(A)| > |A|$. Hence, all white vertices have thickness 1, and since $|B| = |W| + 1$, there can only be one black vertex of thickness more than 1. The lemma follows. □

Our next step is to show that some of the augmenting graphs revealed in the above lemma can be neglected, as they contain redundant sets. Again, we start with definitions.

Definition 3.4.16. *In a strip H,*

- *an* endpoint *is a vertex that belongs to a longest induced path P in H and has degree 1 in P;*
- *an* inner twin *is a pair of similar vertices at distance at least 4 from every endpoint of H.*

Lemma 3.4.17. *Let H be a minimal augmenting $K_{3,3}$-minor-free strip or bracelet satisfying $|V(H)| \geq 19$. Then either H is a strip containing an inner twin, or H contains a redundant set $U \subseteq V(H)$ of size at most 18 such that $H - U$ is an augmenting chain.*

Proof. If H is a bracelet, then it follows from Lemma 3.4.15 that H contains a redundant set $U \subseteq V(H)$ of size 4 such that $H - U$ is an augmenting chain.

Now let H be a strip and let $P = (v_1, \ldots, v_l)$ be a longest induced path in H. Also, let a_i denote the thickness of v_i, for $i \in \{1, \ldots, l\}$. By Lemma 3.4.15, $a_i \leq 2$ for any i. Thus, if $l \leq 9$ then $|V(H)| \leq 18$, and therefore, in what follows we assume that $l \geq 10$.

If $a_i = 2$ for some $i \in \{5, \ldots, l-4\}$, the H contains an inner twin. Now assume $a_i = 1$ for $5 \leq i \leq l-4$. Denote by $x = v_i$ the black vertex satisfying $i = \min\{i' : 1 \leq i' \leq 6, a_{i'} =$

$a_{i'+1} = \cdots = a_6 = 1\}$. Note that such a vertex exists since $l \geq 10$. Symmetrically, let $y = v_j$ denote the black vertex satisfying $j = \max\{j' : l-5 \leq j' \leq l \,:\, a_{l-5} = a_{l-4} = \cdots = a_{j'} = 1\}$. Also, denote by H' the path connecting x to y in H, and by U the remaining vertices of H. It is not difficult to see that U is a redundant set of size at most 18 and $H - U$ is an augmenting chain. □

Combining the results of Corollary 3.4.12, Theorem 2.1.7 and Lemma 3.4.17, we conclude that the problem of finding minimal augmenting graphs in an $S_{1,2,2}$-free $K_{3,3}$-minor-free graph polynomially reduces to that of finding augmenting chains and minimal augmenting strips containing an inner twin. According to a recent result of Gerber, Hertz and Lozin [58], the problem of finding augmenting chains is polynomially solvable in the class of $S_{1,2,3}$-free graphs. Therefore:

Lemma 3.4.18. *A maximum independent set in an $S_{1,2,2}$-free $K_{3,3}$-minor-free graph that contains no minimal augmenting strips with inner twins can be computed in polynomial time.*

Hence, the only unsolved case in our quest for a polynomial-time solution to the maximum independent set problem for $S_{1,2,2}$-free $K_{3,3}$-minor-free graphs is the case when the input graph contains a strip with an inner twin. As mentioned at the beginning of this subsection, this case can be reduced to the previously studied case by means of a double transformation of the input graph. Informally, this transformation can be descibed as follows. First, in a graph G we shrink every class of similar vertices C to a single vertex and assign to this vertex the weight equal $|C|$, obtaining in this way a weighted graph G'. Obviously, a maximum independent set in G corresponds to a maximum-weight independent set in G' and vice versa. To solve the problem for G', we first decompose it into 2-connected components, and then for each 2-connected component of G' we implement a reverse transformation by expanding every vertex with weight w to a class of similar vertices of cardinality w. It turns out that every 2-connected graph transforms in this way into an unweighted graph without stips with inner twins.

We now describe these transformations in detail. For the input graph G, we denote by $\mathcal{C}$ the set of all similarity classes, i.e., classes of vertices with the same neighborhood. For

each similarity class $C \in \mathcal{C}$, we fix an arbitrary member of C and denote it by v_C.

Transformation 1 (From unweighted to weighted)**.**

$$\phi_1 : \bar{G} \mapsto (\hat{G}, \hat{w})$$

INPUT: *An induced subgraph $\bar{G}$ of G.*

OUTPUT: *The ordered pair $(\hat{G}, \hat{w})$, where*

- *$\hat{G}$ is the subgraph of G, induced by the set $\{v_C : C \in \mathcal{C},\ C \cap V(\bar{G}) \neq \emptyset\}$, and*
- *$\hat{w}$ is the collection of vertex weights of $\hat{G}$, given by $\hat{w}(v_C) = |C \cap V(\bar{G})|$ for all $v_C \in V(\hat{G})$.*

Transformation 2 (From weighted to unweighted)**.**

$$\phi_2 : (\hat{G}, \hat{w}) \mapsto \bar{G}$$

INPUT: *An ordered pair $(\hat{G}, \hat{w})$, where*

- *$\hat{G}$ is an induced subgraph of G of the form $\hat{G} = G[\{v_C : C \in \mathcal{C}'\}]$ for some nonempty subset of equivalence classes $\mathcal{C}' \subseteq \mathcal{C}$, and*
- *$\hat{w}$ is the collection of integer vertex weights of $\hat{G}$ satisfying $1 \leq \hat{w}(v_C) \leq |C|$ for all $C \in \mathcal{C}'$.*

OUTPUT: *The subgraph $\bar{G}$ of G, induced by the vertex set $F = \bigcup_{C \in \mathcal{C}'} F_C$ where, for each $C \in \mathcal{C}'$, F_C is an arbitrary subset of C of cardinality $\hat{w}(v_C)$. (For definiteness, we do the following for each equivalence class $C \in \mathcal{C}$: we fix, once and for all, a numbering of vertices of C, and then put into F_C the first $\hat{w}(v_C)$ vertices of C.)*

Up to isomorphism, these two transformations are inverse to each other. More specifically:

Lemma 3.4.19. *(i) For every input $\bar{G}$ to ϕ_1, the graph $\phi_2(\phi_1(\bar{G}))$ is isomorphic to $\bar{G}$.*

(ii) $\phi_1 \circ \phi_2 = id$, i.e., for every input $(\hat{G}, \hat{w})$ to ϕ_2, we have $\phi_1(\phi_2(\hat{G}, \hat{w})) = (\hat{G}, \hat{w})$.

The following result will be of crucial importance for our solution.

Lemma 3.4.20. *Let $\bar{G}$ be an induced subgraph of G that contains a minimal augmenting strip with an inner twin, and let $(\hat{G}, \hat{w}) = \phi_1(\bar{G})$. Then $\hat{G}$ contains a cut-vertex.*[3]

Proof. Let $H = (W, B; E)$ be a minimal augmenting strip with an inner twin $\{u, u'\}$ in $\bar{G}$. By definition, H is an induced subgraph of $\bar{G}$ and hence of G.

First, we notice that u has a neighbor of thickness 2 in H. If not, then, according to Lemma 2.1.4, we conclude that $u, u' \in B$. Deleting the vertices $\{u, u'\}$ from H results in two disjoint strips, say $H_i = (W_i, B_i, E_i)$ for $i = 1, 2$. Since $\{u, u'\}$ is an inner twin of H, the sets $A_i := W_i \backslash N(u)$ (for $i = 1, 2$) are nonempty. But now, it follows from Lemma 2.1.4 that

$$|B| = |N(A_1)| + |N(A_2)| + 2 \geq (|A_1| + 1) + (|A_2| + 1) + 2 = |W_1| + |W_2| + 2 = |W| + 2 = |B| + 1\,,$$

a contradiction.

Therefore, there is a twin $\{v, v'\}$ in H such that $uv, uv', v'u, v'u' \in E(G)$. Consider the 4-cycle C induced by the vertices $\{v, v', u, u'\}$. We claim that C is a separating set of G. Indeed, since $\{u, u'\}$ is an inner twin in H, we may consider two vertices $x \in V(H) \cap (N(u) \backslash N(v))$ and $y \in V(H) \cap (N(v) \backslash N(u))$. Then, C separates x from y: if x and y belonged to the same connected component of $G - C$, the graph G would contain a subdivision of $K_{3,3}$, contradicting the assumption.

Next, we show that $\{u, u'\}$ is a twin in G. Assume there is a vertex $a \in N(u) \backslash N(u')$. Let C_x, C_y denote the connected components of $G - C$ containing x and y, respectively, and let x' and x'' denote vertices in $V(H) \cap V(C_x)$ at distance 1 and 2 from x, respectively. Similarly we define y', y''. To avoid an induced $S_{1,2,2}$ on $\{x, u', u, a, x', x''\}$, we see that a has a neighbor in $\{x, x', x''\}$. Therefore, $a \in C_x$. Next, we observe that a is adjacent to v, since otherwise an $S_{1,2,2}$ arises on the vertex set $\{v, u', u, a, y, y'\}$. By symmetry, we conclude that a is adjacent to v'. However, this leads to a contradictory $K_{3,3}$-minor contained in the

[3] A *cut-vertex* is a vertex of a graph such that removal of the vertex causes an increase in the number of connected components. In particular, if the graph is connected, then a cut-vertex is a vertex whose removal disconnects the graph.

vertex set $\{a, u, u', v, v', x, x', x''\}$. A symmetric argument shows that $\{v, v'\}$ is also a twin in G.

Now, we show that $\{u, u'\}$ separates x from v in G. Assume that there is a path $P = (v_1, \ldots, v_r)$ in $G - \{u, u'\}$ from x to v (with $v_1 = x$ and $v_r = v$). Then $r \geq 3$ and since $\{v, v'\}$ is a twin in G, v_{r-1} is adjacent to v' too. But now, a subdivision of $K_{3,3}$ arises on $V(P) \cup V(C)$, a contradiction.

Therefore, $\{u, u'\}$ is a twin in G that separates a pair of vertices of H with different neighborhoods in G. As can be seen from the above proof, $\{u, u'\}$ separates x from v in $\bar{G}$ as well. Since x and v belong to different equivalence classes of $\mathcal{C}$, the vertex v_{C_u} separates v_{C_y} from v_{C_v} in $\hat{G}$. Thus, v_{C_u} is a cut-vertex of $\hat{G}$ and the proof is complete. □

We now summarize all the above arguments in the following procedure that finds a maximum independent set in an $S_{1,2,2}$-free $K_{3,3}$-minor-free graph G.

Procedure ALPHA

Input: An $S_{1,2,2}$-free $K_{3,3}$-minor-free graph G.

Output: An independent set I of G of maximum cardinality.

Step 0. (Preprocessing)

0.1. Determine the connected components $C_1, \ldots, C_r$ of G. If $r = 1$, go to Step 0.2. Else return $I = \cup_{i=1}^{r} \text{ALPHA}(C_i)$ and halt.

0.2. Compute $\mathcal{C}$, the partition of $V(G)$ into classes of vertices with the same neighborhood.

Step 1. Compute $(G', w) = \phi_1(G)$.

Step 2. Compute a maximum-weight independent set I' of G'. To this end, first reduce the problem to the 2-connected components of G (e.g., by applying decomposition by clique separators, see Section 2.3). To compute a maximum-weight independent set of a 2-connected component $\hat{G}$, with vertex weights $\hat{w}$, perform the following steps:

2.0. Remove vertices of $\hat{G}$ with non-positive weights.

2.1. Compute $\bar{G} = \phi_2(\hat{G}, \hat{w})$.

2.2. Compute a maximum independent set $\bar{I}$ of $\bar{G}$ (by augmentation).

2.3. Compute $\hat{I} = \{v_C : C \in \mathcal{C},\ C \cap \bar{I} \neq \emptyset\}$, a maximum-weight independent set of $\hat{G}$.

Step 3. Return $I := \cup_{v \in I'} C_v$, a maximum independent set in G, and halt.

Based on this algorithm, we can now prove the main result of this subsection.

Theorem 3.4.21. *The maximum independent set problem is polynomially solvable for $S_{1,2,2}$-free $K_{3,3}$-minor-free graphs.*

Proof. The correctness of the algorithm follows from the following observations:

In step 2.1, *the weighted graph $(\hat{G}, \hat{w})$ is a legitimate input to ϕ_2.* To see this, we remark that as the vertex weights are redefined in the decomposition into 2-connected components, their integrality is preserved and the new weights are never larger than the previous ones (cf. Section 2.3). After the removal of vertices with non-positive weights (they can never appear in a maximum-weight independent set!), the graph $\hat{G}$ becomes an induced subgraph of G' with integer vertex weights $\hat{w}$ satisfying $1 \leq \hat{w}(v) \leq w(v)$ for all $v \in V(\hat{G})$. Since (G', w) is an input to ϕ_2 (by Lemma 3.4.19(i)), so is $(\hat{G}, \hat{w})$.

$\bar{G}$ is an induced subgraph of G. Indeed, as observed above, $\hat{G}$ is an induced subgraph of G' with integer vertex weights $\hat{w}$ satisfying $1 \leq \hat{w}(v) \leq w(v)$ for all $v \in V(\hat{G})$. Clearly, the graph $\bar{G} = \phi_2(\hat{G}, \hat{w})$ is an induced subgraph of $\phi_2(G', w) = \phi_2(\phi_1(G)) = G$.

The set $\hat{I}$ in step 2.3 *is a maximum-weight independent set of $\hat{G}$. The output set I is a maximum independent set of G.* For the first statement, note that by Lemma 3.4.19(ii), we have $(\hat{G}, \hat{w}) = \phi_1(\bar{G})$. Both statements then easily follow from the following observations:

- Every maximal independent set in an arbitrary induced subgraph $\tilde{G}$ of G is a union of equivalence classes $C \in \mathcal{C}$.
- Let $\phi_1(\tilde{G}) = (\tilde{G}', \tilde{w})$. If I is maximal independent set in $\tilde{G}$, then the set $I' := \{v_C : C \in \mathcal{C}, C \cap I \neq \emptyset\}$ is a independent set in $\tilde{G}'$ with $\tilde{w}(I') = |I|$. Conversely, if $I' \subseteq V(\tilde{G}')$

is a independent set in $\tilde{G}'$, then the set $I := \cup_{v \in I'} C_v$ is an independent set in $\tilde{G}$ with $|I| = \tilde{w}(I')$.

To determine the time complexity of the algorithm, observe that Steps 0, 1 and 3 can be implemented to run in polynomial time. For Step 2, we conclude from Lemma 3.4.20 that the graph $\bar{G} = \phi_2(\hat{G}, \hat{w})$ contains no minimal augmenting strips with an inner twin. Since $\bar{G}$ is an induced subgraph of G, it is also an $S_{1,2,2}$-free $K_{3,3}$-minor-free graph. By Lemma 3.4.18, a maximum independent set can be computed efficiently in $\bar{G}$. A polynomial-time bound for Step 2 and hence polynomial-time complexity of the algorithm ensue. □

Theorem 3.4.21 and Corollary 3.4.10 together provide a proof of Theorem 3.4.8.

Chapter 4

Polynomial Cases of Finding Maximum Independent Sets: Infinitely Many Forbidden Induced Subgraphs

In this chapter, we discuss polynomial-time solution to the MAXIMUM INDEPENDENT SET problem in classes defined by forbidding infinitely many induced subgraphs.

Recall that our hardness result from Section 1.3.3 implies that, unless $P = NP$, one can only hope for a polynomial-time solution to the MAXIMUM INDEPENDENT SET problem in the class of $\mathcal{F}$-free graphs if at least one of the following conditions is satisfied:

(i) $\mathcal{F}$ contains a graph from the class $\mathcal{S}$, or

(ii) $\mathcal{F}$ contains graphs of maximum degree at most 3 with arbitrarily large girth (i.e., the size of a smallest cycle), or

(iii) $\mathcal{F}$ contains graphs of maximum degree at most 3 with arbitrarily large size of a smallest induced copy of H_i.

We will say that a graph class $Free(\mathcal{F})$ is of *type* (i) (*type* (ii), *type* (iii), respectively), if the set $\mathcal{F}$ satisfies condition (i) (condition (ii), (iii), respectively). Graph classes of type (i) have been the subject of Chapter 3. Here, we turn our attention to the other two cases.

The literature contains many examples of classes of type (ii) for which MAXIMUM WEIGHT INDEPENDENT SET problem is solvable in polynomial time. Perhaps the most remarkable example of this type is the class of perfect graphs [61]. The list of minimal forbidden subgraphs for this class contains odd cycles of length at least 5. A particular subclass of perfect graphs is the class of chordal, or triangulated, graphs [56]. This is precisely the class of graphs of chordality at most 3. *Chordality* of a graph is the length of a longest induced cycle. Therefore, graphs of chordality at most k are precisely $\mathcal{F}$-free graphs, where

$\mathcal{F} = \{C_{k+1}, C_{k+2}, \ldots\}$.

For any $k > 3$, the complexity of the MAXIMUM INDEPENDENT SET problem in graphs of chordality at most k is unknown. However, with some additional restrictions, the MAXIMUM WEIGHT INDEPENDENT SET problem can be solved for graphs of bounded chordality in polynomial time. This is the case for AT-free graphs [31] (a subclass of graphs of chordality at most 5), and graphs where both the chordality and the maximum degree are bounded. For the latter class, polynomial-time solvability of the MAXIMUM WEIGHT INDEPENDENT SET problem follows readily from the following two facts:

- the MAXIMUM WEIGHT INDEPENDENT SET problem is solvable in linear time for graphs of bounded treewidth [7];
- graphs of bounded degree and bounded chordality have bounded treewidth [16].

More formally:

Theorem 4.0.22 ([16]). *For any positive integers k and Δ there is a number N such that every graph of maximum degree at most Δ and chordality at most k has treewidth at most N.*

Classes of type (*iii*) seem to have appeared less frequently in the literature. We are aware of only one example of this type, which, however, also belongs to the second type. This example is due to Hertz and de Werra [70] and is known under the name of AH-free graphs.

By analogy of graphs of bounded chordality, a natural example of classes of type (*iii*) is obtained by forbidding all large H_i's, that is, $\mathcal{F} = \{H_k, H_{k+1}, \ldots\}$ for some $k \geq 1$. For every $k \geq 1$, the class of $(H_k, H_{k+1}, \ldots)$-free graphs properly contains fork-free graphs. However, while the MAXIMUM (WEIGHT) INDEPENDENT SET problem is polynomially solvable for fork-free graphs (cf. Section 3.1), the complexity status of the MAXIMUM INDEPENDENT SET problem in $(H_k, H_{k+1}, \ldots)$-free graphs is still unsettled (for every $k \geq 1$).

In Section 4.1, we extend polynomial-time solvability of the MAXIMUM WEIGHT INDEPENDENT SET problem for graphs of bounded degree and bounded chordality to graphs of

bounded degree without so-called large induced apples. We also show that the MAXIMUM WEIGHT INDEPENDENT SET problem is solvable in polynomial time for $(H_k, H_{k+1}, \ldots)$-free graphs of bounded degree.

Section 4.2 is devoted to planar graphs and their extensions. We show that the treewidth of planar graphs and, more generally, graphs excluding a fixed apex graph as a minor, is bounded above by their chordality, which results in an analogue of Theorem 4.0.22. We also provide a simple 2-approximation algorithm for MAXIMUM WEIGHT INDEPENDENT SET problem in $(H_k, H_{k+1}, \ldots)$-free graphs excluding a fixed apex graph as a minor, for every value of $k \geq 1$.

4.1 Graphs of Bounded Vertex Degree

In this section, we assume that there is a constant upper bound $\Delta \geq 3$ on the maximum vertex degree of input graphs. We start with graphs that do not contain *large induced apples*.

4.1.1 Graphs Without Large Apples

An *apple* of order k, denoted by A_k, is a graph obtained from a chordless cycle of length k by adding a new vertex and connecting it to exactly one vertex of the cycle. Clearly, graphs without induced apples of order more than k generalize graphs of chordality at most k. The goal of this section is to show the following result.

Theorem 4.1.1 ([82]). *For every fixed k and Δ, the maximum weight independent set problem is solvable in polynomial time for $(A_k, A_{k+1}, \ldots)$-free graphs of maximum vertex degree at most Δ.*

Let X denote the class of $(A_k, A_{k+1}, \ldots)$-free graphs of maximum vertex degree at most Δ. By Theorem 2.3.1 from Section 2.3, it is enough to show that MAXIMUM WEIGHT INDEPENDENT SET problem is solvable in polynomial time for those induced subgraphs of graphs in X which are prime and have no clique separators.

The following lemma provides the key structural result toward a proof of Theorem 4.1.1.

Lemma 4.1.2 ([82]). *Let k and Δ be positive integers and G a prime $(A_k, A_{k+1}, \ldots)$-free graph without clique separators of maximum vertex degree at most Δ. Then there is a constant $f(k, \Delta)$ such that G is either $(C_{f(k,\Delta)}, C_{f(k,\Delta)+1}, \ldots)$-free or claw-free.*

Assuming the validity of this lemma, we can now easily prove Theorem 4.1.1.

Proof of Theorem 4.1.1. Let G be an $(A_k, A_{k+1}, \ldots)$-free graph of maximum degree at most Δ which is prime and have no clique separators. By Lemma 4.1.2, there is a constant $f(k, \Delta)$ such that G is either $(C_{f(k,\Delta)}, C_{f(k,\Delta)+1}, \ldots)$-free or claw-free. One can verify in polynomial time if G is claw-free. If this is the case, then an independent set of maximum weight can be found in polynomial time. Otherwise, G is a graph of bounded chordality and of maximum degree at most Δ. According to [16], its treewidth is bounded by a constant (depending only on Δ and $f(k, \Delta)$), and an independent set of maximum weight can again be found in polynomial time.

Combining these observations with Theorem 2.3.1 proves the theorem. □

The rest of this subsection is devoted to the proof of Lemma 4.1.2.

Proof of Lemma 4.1.2. The proof is based on case-by-case analysis and involves quite a few technical details.

Suppose for contradiction that G contains an induced claw $K = (a; b, c, d)$, as well as a long induced cycle C. Assume that $|C| \geq k\Delta$ and let v be a vertex outside C. Clearly, v cannot be adjacent to exactly one vertex on C. Also, it is not difficult to see that v cannot have more than four neighbors on C, since otherwise an induced apple of order at least k arises. Therefore, all vertices outside C can be partitioned into four types according to the number of neighbors on the cycle (type i standing for the vertices with exactly i neighbors on C). In order to avoid big apples, we also must conclude that

- every vertex of type 2 has two consecutive neighbors on C,
- every vertex of type 3 has three consecutive neighbors on C,
- every vertex of type 4 has two pairs of consecutive neighbors on C,
- no vertex of type 0 can be adjacent to a vertex of type 3 or 4.

All of the above statements can be checked by direct inspection, except for the case when a vertex v of type 2 is adjacent to vertices $i-1$ and $i+1$ on the cycle. In order to prove that this is not possible, let us consider the set A of all vertices of G adjacent to exactly two vertices of $C-\{i\}$, namely to $i-1$ and to $i+1$. In particular, $i, v \in A$. Let B be the connected component of the complement of $G[A]$ containing i and v, and let z be a vertex of G distinguishing B. Without loss of generality, we may assume that z is adjacent to v and non-adjacent to i (obviously, B contains two non-adjacent vertices distinguished by z). To avoid a big induced apple, we must conclude that z is adjacent to $i-1$ and $i+1$ and to no other vertex of C. But then z belongs to A and consequently to B, which contradicts the choice of z.

From the above discussion, we know that if a vertex v has a neighbor i on C, the v must also be adjacent to $i-1$ or $i+1$. In particular, no claw in G can have more than two vertices on C. The rest of the proof is partitioned into several cases according to the size of the intersection of a claw and a long cycle. In these cases, we assume that the length of C is at least $ck\Delta$ for some constant c depending on the case considered.

Case 1: $K \cap C = \{a, d\}$. Let y be the other neighbor of a on C. To avoid a claw with three vertices on C, we conclude that y is adjacent both to b and to c. Denote by z a vertex adjacent to b, but not to c. Without loss of generality we may assume that $N(z) \cap C \neq \{a, y\}$ (otherwise, as before, denote $A := \{x \ : \ N(x) \cap C = \{a, y\}\}$, B the connected component of $\overline{G[A]}$ containing a and y, and z a vertex of G distinguishing B).

If z belongs to C, then the reader can easily find an induced apple of order at least $|C|/4$. From now on, we may assume that no vertex on C distinguishes b and c, i.e., that $N(b) \cap C = N(c) \cap C$. If z has a neighbor on C, then we may assume without loss of generality that $N(z) \cap C \neq N(b) \cap C$, and an induced apple of order at least $|C|/4$ can easily be found. If z does not belong to C and has no neighbor on C, then z is of type 0 and hence b and c are of type 2. Let U denote the vertex set of that connected component of the complement of $G[\{v \in V : N(v) \cap C = \{a, y\}\}]$ which contains b and c, and let K_0 denote a maximal clique of G satisfying $\{a, y\} \subseteq K_0 \subseteq V \backslash U$. Suppose that there is a z-C path avoiding $K_0 \cup U$, and let P be a shortest such path. By minimality, every vertex of P except the last one (which has a neighbor on C) is of type 0. Also, the vertex c must have

a neighbor on P, since otherwise a long apple arises. But now, a long apple can be formed with the portion of P from the last neighbor of c on P till the end, a part of C, and one of the edges connecting b to C.

We may now assume that every z-C path avoiding K_0 intersects U. Let P be a shortest such path, say $P = (v_0, \ldots, v_r)$ with $v_0 = z$, and let x be the last vertex on P with $N(x) \cap C = \{a, y\}$, say $x = v_i$. If x belongs to U, then x_{i+1} dominates U (otherwise we could replace z by x_{i+1}), and no vertex of P following x_{i+1} has a neighbor in U. By minimality of P, we may assume that $z = x_{i-1}$, and a long induced apple in G can easily be found. If $x \notin U$, then either x has no neighbors in U, or x dominates U. But now, a long apple can be formed that includes either the last vertex of P on U (in the former case) or a non-neighbor of x in K_0 (in the latter case).

Case 2: $K \cap C = \{b, c\}$. Then a is of type 3 or 4, and hence d has a neighbor on C. Analyzing all possible types of neighborhoods of vertices a and d on the cycle, we conclude that G contains an apple of order at least $|C|/4$.

Case 3: $K \cap C = \{a\}$. Then each of the vertices $\{b, c, d\}$ has at least one more neighbor on C. Assuming that no claw in G has more than one vertex on C, we conclude that no common neighbor of b and c on C is adjacent to a common non-neighbor of b and c on C. Analyzing the possible neighborhoods of b, c and d on C, we can either find a long induced apple, or a claw intersecting a long cycle in more than one vertex.

Case 4: $K \cap C = \{d\}$. As in Case 3, we see that no common neighbor of b and c on C can be adjacent to a common non-neighbor of b and c on C. Analyzing the possible types of vertices a, b and c, we conclude that the only case that does not immediately reduce to finding a long apple or one of the already proved cases, is the case when the vertex a is of type 2, while b and c are of type 0. In this case, we proceed similarly as in Case 1. Letting y be the other neighbor of a on C, we define

- U as the vertex set of that connected component of the complement of $G[N(a) \backslash (C \cup N(C))]$ which contains b and c,
- K_0 as a maximal clique of G satisfying $\{a, d, y\} \subseteq K_0 \subseteq W$, where W is the set of vertices that are either have no neighbors in U or dominate U,

- z as a vertex outside U that is adjacent to b, but not to c.

Note that no vertex of K_0 is adjacent to both d' and y', the respective neighbors of d and y on C (since otherwise a long apple containing a would arise). Suppose that there is a z-C path avoiding $K_0 \cup U$, and let z and P be chosen so that P is a shortest such path, say $P = (v_0, \ldots, v_r)$ with $v_0 = z$. If every vertex of P except the last one (which has a neighbor on C) is of type 0, then a long apple arises. So let x be the last vertex of P adjacent to $\{d, y\}$, say $x = v_i$. Then x is necessarily adjacent to precisely one vertex among $\{d', y'\}$. (If it is adjacent to none of them, the case reduces to Case 1, forming a claw $(d; d', x, w)$ (say), where w is a non-neighbor of x in K_0. If it is adjacent to both of them, then G contains a long apple through the vertices $x = v_i$, d', y' and v_{i-1}.) We conclude, without loss of generality, that $N(x) \cap C = \{d', d, y\}$, and $N(w) \cap C = \{d, y, y'\}$, where w is a non-neighbor of x in K_0. To avoid a long induced apple, we conclude that x is adjacent to a, and moreover that x is not adjacent to b (and hence, by minimality of P, x has no neighbors in U). However, the case now reduces to Case 1, with the claw $(a; b, c, x)$, and the cycle through the vertices $\{d', x, a, w, y'\}$.

We may now assume that every z-C path avoiding K_0 intersects U. Let P be a shortest such path, say $P = (v_0, \ldots, v_r)$ with $v_0 = z$, and let x be the last vertex on P that belongs to U, say $x = v_i$. Then v_{i+1} dominates U, and, similarly as above, we conclude that no internal vertex of P following v_i is adjacent to $\{d, y\}$. Also, it is easy to see that $P \cap U = \{x\}$, since otherwise v_{i+1} would distinguish U and we could replace z by v_{i+1}. By minimality of P, we may assume that $z = v_{i-1}$. Now, either a long induced apple in G can be found (if v_{i+1} is adjacent to $\{d, y\}$), or there is a long induced cycle C' and a vertex outside C' with precisely two neighbors on C' at distance 2 (if v_{i+1} is not adjacent to any of $\{d, y, a\}$), or the case reduces to Case 3. Contradiction.

Case 5: $K \cap C = \emptyset$ and a vertex of K has a neighbor on C. If at least two vertices of K have a neighbor on C, then it is easy to find either a big chordless cycle C' intersecting K, or an induced claw intersecting C. If exactly one vertex of K has a neighbor on C, then, to avoid a claw intersecting C, we conclude that this vertex is not the center of K, say it is b. Then b is of type 2 and a is of type 0. Let us denote the neighborhood of b on C by $\{x, y\}$. We proceed similarly as in Cases 1 and 4. We define

- U as the vertex set of that connected component of the complement of $G[N(a)\backslash(C \cup N(C))]$ which contains c and d,
- K_0 as a maximal clique of G satisfying $\{b, x, y\} \subseteq K_0 \subseteq W$, where W is the set of vertices that are either have no neighbors in U or dominate U,
- z as a vertex outside U that is adjacent to c, but not to d.

To avoid Case 4 and a large induced apple, we conclude that z is either of type 0 or $N(z) \cap C = \{x, y\}$.

If $N(z) \cap C = \{x, y\}$, then, in order to avoid Case 1, we conclude that z is adjacent to b. Again, to avoid Case 1, or a long induced apple, we conclude that z dominates K_0.

Suppose that there is a z-C path avoiding $K_0 \cup U \cup \{a\}$, and let z and P be chosen so that P is a shortest such path, say $P = (v_0, \ldots, v_r)$ with $v_0 = z$. Observe that all internal vertices of P are of type 0 (by definition of K_0 and to avoid previous cases), which implies that no internal vertex of P is adjacent to b. Next, note that c must be adjacent to v_1 (or we are in Case 4), and that, in order to avoid a long apple, the last neighbor of a on $P - \{v_1\}$ (if any) coincides with the last neighbor of c on $P - \{v_1\}$. Now, in each of the cases corresponding to the position of the last neighbor of a on P (if any), we can find a long chordless cycle and an induced claw intersecting it.

We may now assume that every z-C path avoiding K_0 intersects $U \cup \{a\}$. Let P be a shortest such path. Then P is disjoint from U (else we can apply a similar analysis as the one in Case 4 and either conclude that G contains a long apple, or the case reduces to Case 4 or the subcase of the previous paragraph). Assuming that a belongs to P, we again conclude that all the internal vertices of P following a are of type 0, and a long apple arises.

The case when z is of type 0 (and then also non-adjacent to b) can be handled in a similar manner.

Case 6: $K \cap C = \emptyset$ and no vertex of K has a neighbor on C. Since G is connected, there must exist a chordless path $X = (x_1, \ldots, x_s)$ such that x_1 is the only vertex of X adjacent to a, while x_s is the only vertex of X that has a neighbor on C. The set of all such paths connecting a to C will be denoted by $\mathcal{P}$. Observe in every path $X \in \mathcal{P}$ the vertex with maximum index is of type 2, while the remaining vertices of X are of type 0. Given a vertex

x_j on $X \in \mathcal{P}$, we define X-distance from x_j to a or to C to be the number of edges of X connecting x_j to a or to C, respectively. In what follows, all distances will be measured according to this definition.

Since G is 2-connected, $\mathcal{P}$ must contain a pair of paths $X = (x_1, x_2, \ldots, x_s)$ and $Y = (y_1, y_2, \ldots, y_t)$ such that $x_i \neq y_j$ for all i, j. If additionally $N(x_s) \cap C \neq N(y_t) \cap C$, we call XY a disjoint pair. If XY is not disjoint, then $\mathcal{P}$ must contain one more path Z which avoids the clique $N(x_s) \cap C = N(y_t) \cap C$, since otherwise this clique is separating. It is not difficult to see that Z can be chosen in such a way that

$(*)$ either XZ or YZ is a disjoint pair.

If XY is a disjoint pair, any edge of the form $x_i y_j$ will be called a *XY-chord*. If no such an edge exists, the pair will be called *chordless*. If G contains a chordless pair, then it obviously contains a big chordless cycle intersecting the claw. Therefore, we assume in the rest of the proof that

$(**)$ for every claw and every large cycle satisfying conditions of Case 6, any disjoint pair connecting the claw to the cycle contains a chord.

If XY is a disjoint pair with a chord $x_i y_j$, we denote by $X x_i y_j Y$ the path $(x_1, \ldots, x_i, y_j, \ldots, y_t)$. Let $\hat{i}$ be the largest index such that $x_{\hat{i}}$ has a neighbor on Y. Similarly, let $\hat{j}$ be the largest index such that $y_{\hat{j}}$ has a neighbor on X. If $x_{\hat{i}}$ is adjacent to $y_{\hat{j}}$, we call XY a *good pair* and $x_{\hat{i}} y_{\hat{j}}$ the lowest XY-chord. If XY is not good, we can associate to it a good pair as follows. Let $x_{\bar{i}} \in X$ be the neighbor of $y_{\hat{j}}$ closest to C. Without loss of generality we may assume that $x_{\bar{i}}$ is not adjacent to $x_{\hat{i}}$, i.e., $\bar{i} < \hat{i} - 1$, since otherwise the claw induced by $x_{\bar{i}}, x_{\bar{i}-1}, x_{\hat{i}}, y_{\hat{j}}$ does not satisfy condition $(**)$. Denote $X' = X x_{\bar{i}} y_{\hat{j}} Y$. Clearly, $X' \in \mathcal{P}$. Symmetrically, we can define a path Y'. Then $X'Y'$ is a disjoint pair with fewer chords than XY. Repeated applications of this construction result in a good pair. This observation allows us to assume in the rest of the proof that

$(***)$ every disjoint pair is good.

Let XY be a good pair with the lowest chord $x_{\hat{i}} y_{\hat{j}}$. The neighbors of x_s and y_t on C partition C into two parts the largest of which together with the vertices $x_{\hat{i}}, x_{\hat{i}+1}, \ldots, x_s$

and $y_{\hat{j}}, y_{\hat{j}+1}, \ldots, y_t$ create a new chordless cycle C' of length at least $|C|/2$. To avoid Case 5 with respect to C', we conclude that a is not adjacent to $x_{\hat{i}}$ and $y_{\hat{j}}$. Also, to avoid a claw intersecting C', we conclude that the vertices $x_{\hat{i}}, y_{\hat{j}}, x_{\hat{i}-1}y_{\hat{j}-1}$ create a clique. Denote $X^0 = (x_1, \ldots, x_{\hat{i}-1})$, $X^1 = (x_{\hat{i}}, \ldots, x_s)$, and $Y^0 = (y_1, \ldots, y_{\hat{j}-1})$, $Y_1 = (y_{\hat{j}}, \ldots, y_t)$. Also, $X_Y = X^0 \cup Y^1$ and $Y_X = Y^0 \cup X^1$. Observe that the pair $X_Y Y_X$ is also good and its lowest chord coincides with $x_{\hat{i}}y_{\hat{j}}$.

Since the clique $x_{\hat{i}}, y_{\hat{j}}, x_{\hat{i}-1}y_{\hat{j}-1}$ is not separating, there must exist a path $Z = (z_1, \ldots, z_u)$ avoiding this clique. Moreover, it is not difficult to see that the path Z can be chosen in such a way that is creates a disjoint (and hence good) pair with one of the four paths X, Y, X_Y, Y_X. Without loss of generality we will assume that XZ is good pair and will denote its lowest chord by $x_{\bar{i}}z_{\bar{k}}$. Let us show that the path Z can be chosen so that

(****) $x_{\bar{i}} \in X^0$, i.e., $\bar{i} < \hat{i}$.

To this end, observe that the pair X^0Y^0 is not disjoint with respect to C', as $N(x_{\hat{i}-1}) \cap C' = N(y_{\hat{j}-1}) \cap C'$. According to $(*)$ and $(***)$, there must exist a path Z' such that either X^0Z' or Y^0Z' is a good pair with respect to C'. Without loss of generality, we assume that X^0Z' is a good pair with respect to C'. If Z' meets C' at some vertices belonging to C, then XZ' is a good pair with respect to C, in which case defining $Z := Z'$ gives us a pair XZ satisfying (****). If Z' meets C' at some vertices of Y, we can extend Z' by means of vertices of Y to a path Z'' that meets C, in which case defining $Z := Z''$ gives us a pair XZ satisfying (****). Finally, if Z' meets C' at some vertices of X, then we extend Z' by means of vertices of X to a path Z'' that meets C, in which case defining $X := X_Y$ and $Z := Z''$ gives us a pair XZ satisfying (****). Therefore, in what follows, without loss of generality we will assume that the path Z avoiding the clique $x_{\hat{i}}, y_{\hat{j}}, x_{\hat{i}-1}y_{\hat{j}-1}$ is chosen so that XZ is a good pair with the lowest chord $x_{\bar{i}}z_{\bar{k}}$ such that $x_{\bar{i}} \in X^0$. We denote $Z^0 = (z_1, \ldots, z_{\bar{k}-1})$, $Z^1 = (z_{\bar{k}}, \ldots, z_u)$.

With the above notations in mind, assume now that XY is a good pair such that its lowest chord $x_{\hat{i}}y_{\hat{j}}$ is as close to a as possible. If $Y^0 \cap Z^1 = \emptyset$, we consider a neighbor $x \in X^0$ of $z_{\bar{k}}$ closest to a and a neighbor $y \in Y^0$ of $x_{\hat{i}}$ closest to a and two paths $X^* = Xxz_{\bar{k}}Z$ and $Y^* = Yyx_{\hat{i}}X$. It is not difficult to see that X^*Y^* is a good pair and the lowest X^*Y^*-chord

is closer to a than $x_{\hat{i}}y_{\hat{j}}$, which contradicts the choice of XY. If Y^0 ans Z^1 have a non-empty intersection, we consider any chordless path Y^* starting at y_1 and ending at z_u, all vertices of which belong to $Y^0 \cup Z^1$. Clearly XY^* is a good pair and the lowest XY^*-chord is closer to a than $x_{\hat{i}}y_{\hat{j}}$. This contradiction completes the proof of the lemma. □

4.1.2 Graphs Without Large Induced H_i's

In this section, we deal with $(H_k, H_{k+1}, \ldots)$-free graphs, a class of type (iii). Notice that $(H_k, H_{k+1}, \ldots)$-free graphs generalize claw-free graphs. We will show that graphs of bounded vertex degree in this class are not too different from claw-free graphs.

Lemma 4.1.3 ([82]). *For every fixed positive integers k and Δ, there is a constant $\rho = \rho(k, \Delta)$ such that any connected $(H_k, H_{k+1}, \ldots)$-free graph G of maximum vertex degree at most Δ contains an induced claw-free subgraph with at least $|V(G)| - \rho$ vertices.*

Proof. To prove the lemma, we will show that for any two induced copies K and K' of a claw in G, the distance between them does not exceed $k+1$. Suppose by contradiction that a shortest path P joining a claw $K = (x; a, b, c)$ to another claw $K' = (x'; a', b'c')$ consists of $r \geq k+2$ edges. Let us write $P = (v_0, v_1, \ldots, v_{r-1}, v_r)$ where $v_0 \in V(K), v_r \in V(K')$, and the only edges of P are $v_i v_{i+1}$ for $0 \leq i \leq r-1$.

Observe that vertex v_1 may belong to another claw induced by some vertices of $V(K) \cup \{v_1, v_2\}$, in which case we denote this claw by $\widetilde{K}$; otherwise let $\widetilde{K} := K$. Analogously, by $\widetilde{K}'$ we denote either a claw containing vertex v_{r-1} and induced by some vertices of $V(K') \cup \{v_{r-1}, v_{r-2}\}$ (if such a claw exists) or K' otherwise. But now the two claws $\widetilde{K}$ and $\widetilde{K}'$ together with the vertices of P connecting them induce a graph H_l with $l \geq k$. Contradiction.

To conclude the proof, assume that G contains an induced claw K. According to the above discussion, the distance from the center of K to the center of any other claw in G (if any) is at most $k+3$. Since G is a connected graph of maximum degree at most Δ, there is a constant $\rho = \rho(k, \Delta)$ bounding the number of vertices of G of distance at most $k+3$ from the center of K. Deletion of all these vertices leaves a subgraph of G which is claw-free. □

It follows from the proof of Lemma 4.1.3 that the class X of all connected $(H_k, H_{k+1}, \ldots)$-free graph G of maximum vertex degree at most Δ satisfies the hypotheses of Theorem 2.4.1 from Section 2.4. Indeed, for a graph $G \in X$, the set U of centers of induced claws in G can clearly be computed in polynomial time. Moreover, a polynomial-time solution to the MAXIMUM WEIGHT INDEPENDENT SET problem for connected graphs in some class implies a solution to the problem for the whole class.

Theorem 4.1.4 ([82])**.** *For every fixed positive integers k and Δ, the* MAXIMUM WEIGHT INDEPENDENT SET *problem is solvable in polynomial time for $(H_k, H_{k+1}, \ldots)$-free graphs of maximum vertex degree at most Δ.*

4.2 Planar and More General Graphs

Recall Corollary 3.4.4 from Section 3.4.

Corollary. *Let H be an apex graph and X a subclass of H-minor-free graphs. Denote by Y the class of all graphs that can be obtained from graphs in X by applying a sequence of (zero or more) edge contractions and vertex deletions. Suppose that the treewidth of graphs in Y is bounded above by a function of their maximum degree: that is, there exists a function f such that $\mathrm{tw}(G) \leq f(\Delta(G))$, for all $G \in Y$. Then, the treewidth of graphs in X is bounded.*

Trivially, any class of bounded chordality is closed under edge contractions and vertex deletions. Together with Theorem 4.0.22, this implies the following result.

Theorem 4.2.1. *For any apex graph H and any positive integer k, there is a number N such that every H-minor-free graph of chordality at most k has treewidth at most N.*

Corollary 4.2.2. *For any apex graph H and any positive integer $k \geq 3$, the* MAXIMUM WEIGHT INDEPENDENT SET *problem is solvable in linear time for $(C_k, C_{k+1}, \ldots)$-free H-minor-free graphs.*

Let us now consider $(H_k, H_{k+1}, \ldots)$-free graphs excluding a fixed apex graph as a minor. We will show that any graph in such a class can be partitioned into a claw-free graph, and a graph of bounded treewidth. In particular, this will give a polynomial-time approximation

algorithm for MAXIMUM WEIGHT INDEPENDENT SET problem with performance ratio 2. The existence of such an approximation algorithm also follows from a PTAS for the MWIS problem in any minor-closed family [39]. Nevertheless, we believe that it is worth mentioning our approach, mainly because it is so simple: one part of the bipartition in the lemma below is given by $V_1 = \{v \in V(G) : v \text{ is the center of an induced claw in } G\}$.

Lemma 4.2.3 ([83]). *For any positive integer k and any apex graph H, there is a constant $\rho = \rho(k, H)$ such that the vertex set V of every connected $(H_k, H_{k+1}, \ldots)$-free graph G with no minor isomorphic to H can be partitioned (in polynomial time) into sets V_1 and V_2 such that:*

(i) the treewidth of $G[V_1]$ is at most ρ, and

(ii) the graph $G[V_2]$ is claw-free.

Proof. To prove the lemma, we will show that for any two induced copies K and K' of a claw in G, the distance between them does not exceed $k+1$. Suppose by contradiction that there is a (shortest) path $P = (v_0, v_1, \ldots, v_{r-1}, v_r)$ with $v_0 \in V(K), v_r \in V(K')$ and $r \geq k+2$.

Observe that vertex v_1 may belong to another claw induced by some vertices of $V(K) \cup \{v_1, v_2\}$, in which case we denote this claw by $\widetilde{K}$; otherwise let $\widetilde{K} := K$. Analogously, by $\widetilde{K}'$ we denote either a claw containing vertex v_{r-1} and induced by some vertices of $V(K') \cup \{v_{r-1}, v_{r-2}\}$ (if such a claw exists) or K' otherwise. But now the two claws $\widetilde{K}$ and $\widetilde{K}'$ together with the vertices of P connecting them induce a graph H_l with $l \geq k$. Contradiction.

To conclude the proof, fix the center x of an induced claw in G (if G is claw-free, then $V_1 = \emptyset$ and $V_2 = V$ provides a desired partition). According to the above discussion, the distance from x to the center of any other claw in G (if any) is at most $k+3$. Let V_1 denote the set of all vertices of G that are at distance at most $k+3$ from x. Then, the diameter of V_1 is at most $2k+6$. Therefore, by the diameter-treewidth property of H-minor-free graphs (cf. Theorem 3.4.2), the treewidth of $G[V_1]$ is bounded. Also, if we let $V_2 = V \backslash V_1$, we see that the graph $G[V_2]$ is claw-free. □

Using the partition $V = V_1 \cup V_2$ of the vertex set of G as in the Lemma 4.2.3, we

get a 2-approximation algorithm for the MAXIMUM WEIGHT INDEPENDENT SET problem as follows. We solve the problem exactly in each of the parts $G[V_i]$, for $i = 1, 2$, and output the better of the two solutions. Clearly, this can be done in polynomial time, and the output solution is at least half as heavy as a maximum-weight independent set of G. So, we have a polynomial-time approximation algorithm with performance ratio 2.

Theorem 4.2.4 ([83])**.** *For any apex graph H and any positive integer k, there is a 2-approximation polynomial-time algorithm to solve the* MAXIMUM WEIGHED INDEPENDENT SET *problem in the class of $(H_k, H_{k+1}, \ldots)$-free graphs with no minor isomorphic to H.*

Chapter 5

The Exact Weighted Independent Set Problem

In this chapter, we address complexity issues for the exact versions of the MAXIMUM WEIGHT INDEPENDENT SET problem: the EXACT WEIGHTED INDEPENDENT SET (EWIS) problem, and the EXACT WEIGHTED MAXIMUM INDEPENDENT SET (EWIS_α) problem. Recall that the EXACT WEIGHTED (MAXIMUM) INDEPENDENT SET problem is the problem of determining whether a given weighted graph contains a (maximum) independent set of a given weight. The results in this chapter are from [89].

We first discuss in Section 5.1 some relations between the complexities of the problems MWIS, EWIS and EWIS_α. In Section 5.2, we present the first hardness results. Finally, Section 5.3 is devoted to pseudo-polynomial time solutions. The algorithms mostly resemble those for the MWIS problem in respective graph classes, and are based either on a dynamic programming approach (Section 5.3.1), or on modular decomposition, whose application to the exact weighted independent set problem is developed in Section 5.3.2.

The triple (G, w, b) will always represent an instance of EWIS (or EWIS_α), i.e., $G = (V, E)$ is a graph, $w : V \to \mathbb{Z}$ are vertex weights, and $b \in \mathbb{Z}$ is the target weight. If H is an induced subgraph of G, we will also consider triples of the form (H, w, b) as instances of EWIS, with the weights w representing the restriction of w to $V(H)$. We will denote by $\text{EWIS}(G, w, b)$ the solution to the instance (G, w, b) of EWIS, that is, $\text{EWIS}(G, w, b)$ is *yes* if there is an independent set I in G with $w(I) = b$, and *no* otherwise. Similarly, $\text{EWIS}_\alpha(G, w, b)$ is *yes* if there is a maximum independent set I in G with $w(I) = b$, and *no* otherwise.

5.1 Preliminary Observations

The exact weighted independent set problem is (weakly) NP-complete for any class of graphs containing the edgeless graphs $\{nK_1 : n \geq 0\}$. There is a direct equivalence between the exact weighted independent set problem on $\{nK_1 : n \geq 0\}$ and the subset sum problem, a well-known NP-complete problem [55]. The *subset sum* problem is the following: given n integers $a_1, \ldots, a_n$ and a bound b, determine whether there is a subset $J \subseteq [n]$ such that $\sum_{j \in J} a_j = b$.

Therefore, for a given class of graphs X, the question of interest is whether the EWIS problem is strongly NP-complete for graphs in X, or is it solvable in pseudo-polynomial time.

First, it is easy to see that we can assume without loss of generality that all weights are positive:

Remark. *The* EWIS *problem with arbitrary integer weights is polynomially equivalent to the* EWIS *problem, restricted to instances* (G, w, b) *such that* $b \leq w(V(G))$ *and* $1 \leq w(v) \leq b$*, for all* $v \in V$*. The same equivalence holds true for the* EWIS_α *problem.*

Proof. Solving the EWIS problem for any particular instance reduces to solving n problems EWIS_k, in which the independent sets are restricted to be of size k, for all $k \in [n]$ (unless $b = 0$, in which case the solution is trivial, as the empty set is an independent set of weight 0). The weights in EWIS_k can be assumed to be positive: otherwise, we can add a suitably large constant N to each vertex weight and replace b by $b + kN$ to get an equivalent EWIS_k problem with positive weights only. Finally, applying the same transformation again with $N = w(V) + 1$ reduces the problem EWIS_k to a single EWIS problem with positive weights. Repeating this for all values of $k \in [n]$, the result follows.

Finally, if all vertex weights are positive, we can delete from the graph all vertices whose weight exceeds b, as they will never appear in a solution. The solution is trivial if $b > w(V)$.

The same assumption on vertex weights as for the EWIS problem can also be made for the instances (G, w, b) of its restricted counterpart EWIS_α. Again, if some of the weights are negative, we can modify the weights and the target value as we did above for EWIS_k.

Now we only do it for $k = \alpha(G)$. Note that we can compute $\alpha(G)$ as that only $p \in [n]$ such that $\text{EWIS}_\alpha(G, \mathbf{1}, p)$ is *yes*, where $\mathbf{1}$ denotes the unit vertex weights. □

We now discuss some relations between the complexities of the problems MWIS, EWIS and EWIS_α, when restricted to particular graph classes.

Lemma 5.1.1. *Let X be a class of graphs. The following statements are true.*
(i) If the EWIS_α *problem is solvable in pseudo-polynomial time for graphs in X, then so is the* MWIS *problem.*
(ii) If the EWIS *problem is solvable in pseudo-polynomial time for graphs in X, then so is the* EWIS_α *problem.*
(iii) Let $X' = \{G' : G \in X\}$ where $G' = (V', E')$ is the graph, obtained from a graph $G = (V, E) \in X$, by adding pendant vertices, as follows: $V' = V \cup \{v' : v \in V\}$, $E' = E \cup \{\{v, v'\} : v \in V\}$. If the EWIS_α *problem is solvable in pseudo-polynomial time for graphs in X', then the* EWIS *problem is solvable in pseudo-polynomial time for graphs in X.*

Proof. (i) Let (G, w, k) be an instance of the decision version of the weighted independent set problem. As we can assume positive weights, G contains an independent set of total weight at least k if and only if G contains a maximum independent set of total weight at least k. By testing values for b from $w(V)$ down to k and using an algorithm for the EWIS_α problem on the instance (G, w, b), we can decide whether G contains a maximum independent set of total weight at least k.
(ii) Let (G, w, b) be an instance of the EWIS_α problem. It is easy to see that the following algorithm solves EWIS_α.
Step 1. Compute $\alpha(G)$, which is equal to the maximum $k \in [n]$ such that $\text{EWIS}(G, \mathbf{1}, k)$ is *yes*, where $\mathbf{1}$ denotes the unit vertex weights.
Step 2. Let $N = w(V) + 1$. For every vertex $v \in V(G)$, let $w'(v) = w(v) + N$. Let $b' = b + \alpha(G)N$. Then it is easy to verify that $\text{EWIS}_\alpha(G, w, b) = \text{EWIS}(G, w', b')$.
(iii) Let (G, w, b) with $G = (V, E) \in X$ be an instance of EWIS. Let G' be the graph, defined as in the lemma. Let $n = |V|$ and let $w'(v) = (n + 1)w(v)$ for all $v \in V$ and

$w'(v) = 1$ for $v \in V'$. Then, it is easy to verify that EWIS(G, w, b) is *yes* if and only if EWIS$_\alpha(G', w', b')$ is *yes* for some value b' in the set $\{(n+1)b, \ldots, (n+1)b + n - 1\}$. □

The problem EWIS is clearly in NP, and so is EWIS$_\alpha$ for any class of graphs X where MIS is polynomially solvable. Therefore, Lemma 5.1.1 implies the following result.

Corollary 5.1.2. *Let X be a class of graphs. The following statements are true.*
(i) If the MWIS *problem is strongly NP-complete for graphs in X, then the* EWIS$_\alpha$ *problem is strongly NP-hard for graphs in X. If, in addition, the* MIS *problem is polynomial for graphs in X, then* EWIS$_\alpha$ *is strongly NP-complete for graphs in X.*
(ii) If the EWIS$_\alpha$ *problem is strongly NP-hard for graphs in X, then the* EWIS *problem is strongly NP-complete for graphs in X.*
(iii) Let X' be as in Lemma 5.1.1. If the EWIS *problem is strongly NP-complete for graphs in X, then the* EWIS$_\alpha$ *problem is strongly NP-hard for graphs in X'. If, in addition, the* MIS *problem is polynomial for graphs in X', then* EWIS$_\alpha$ *is strongly NP-complete for graphs in X'.*

Therefore, we are mainly interested in determining the complexity (strong NP-complete or pseudo-polynomial results) of the EXACT WEIGHTED INDEPENDENT SET problem in those classes of graphs where the MAXIMUM WEIGHT INDEPENDENT SET problem is solvable in pseudo-polynomial time. Moreover, combining parts (*ii*) and (*iii*) of the lemma shows that when $X \in$ {forests, bipartite graphs, chordal graphs}, the problems EWIS and EWIS$_\alpha$ are equivalent (in the sense that, when restricted to the graphs in X, they are either both solvable in pseudo-polynomial time, or they are both strongly NP-complete).

We conclude this subsection by showing that a similar equivalence remains valid for the class of line graphs. More precisely, if L, $L(Bip)$, $L(K_{2n})$ and $L(K_{n,n})$ denote the classes of line graphs, line graphs of bipartite graphs, line graphs of complete graphs with an even number of vertices, and line graphs of complete balanced bipartite graphs, respectively, we have the following result.

Lemma 5.1.3. *The* EWIS *problem is strongly NP-complete for graphs in L (resp., $L(Bip)$) if and only if the* EWIS$_\alpha$ *problem is strongly NP-complete for graphs in $L(K_{2n})$ (resp.,*

$L(K_{n,n})$).

Proof. The backward implication is given by part (ii) of Lemma 5.1.1. The forward implication follows from a reduction of the exact matching problem to the exact perfect matching problem which we show now. Given an instance $G = (V, E)$ with edge weights w and a target b for the exact matching problem, construct an instance $(K_{n'}, w', b')$ for the exact perfect matching problem as follows. If $n = |V|$ is odd, we add a new vertex and we complete the graph G. For an edge e of G, let $w'(e) = Nw(e)$ where $N = w(E) + 1$, for an edge $e \notin E$ let $w'(e) = 1$. The transformation is clearly polynomial, and G has a matching of weight b if and only if $K_{n'}$ has a perfect matching of weight $NM + k$ for some value of $k \in \{0, \ldots, n-1\}$. Also, it is easy to see that in the case of bipartite graphs $G = (L, R; E)$ with $|L| \leq |R|$, we can add $|R \setminus L|$ vertices to L to balance the bipartition. □

5.2 Hardness Results

The maximum weight independent set problem is solvable in polynomial time for bipartite graphs by network flow techniques. However, as we show in this section, the exact version of the problem is strongly NP-complete even for cubic bipartite graphs.

The strong NP-completeness of the EWIS problem in bipartite graphs follows from a straightforward reduction from the balanced biclique problem which is known to be NP-complete [55]. The *balanced biclique* problem consists in determining whether, given a bipartite graph $G = (L, R; E)$ where $L \cup R$ is a bipartition of its vertex set, and an integer k, there exist subsets $L' \subseteq L$ and $R' \subseteq R$ with $|L'| = |R'| = k$ such that the subgraph induced by $L' \cup R'$ is a complete bipartite subgraph (also called *biclique*) of size k. From an instance G and k of balanced biclique, we introduce weight 1 on each vertex of L, weight $B = \max\{|L|, |R|\} + 1$ on each vertex of R, and we set $b = k + Bk$. It is clear that there exist an independent set in $(L, R; (L \times R) \setminus E)$ with weight b if and only if there exists a balanced biclique in $(L, R; E)$ of size k.

We now strengthen this result by proving that the EWIS$_\alpha$ problem is strongly NP-complete even for cubic bipartite graphs. By contrast, for graphs of maximum degree 2,

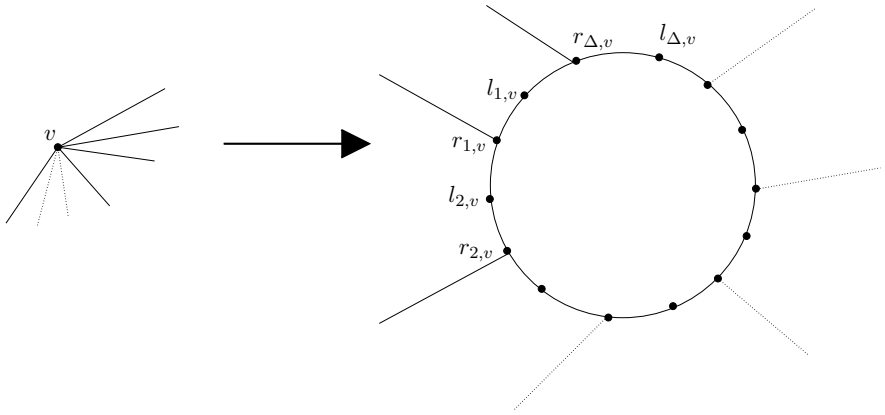

Figure 5.1: The gadget $H(v)$.

EWIS and EWIS$_\alpha$ are pseudo-polynomially solvable problems.[1]

Theorem 5.2.1. *The* EXACT WEIGHTED MAXIMUM INDEPENDENT SET *problem is strongly NP-complete in cubic bipartite graphs.*

Proof. The problem is clearly in NP, as the maximum independent set problem is solvable in polynomial time for bipartite graphs. The hardness reduction is made from the decision version of the clique problem in regular graphs which is known to be NP-complete, see [55]. Let (G, k) be an input to the clique problem, where $G = (V, E)$ is a Δ-regular graph on n vertices and k is an integer. Without loss of generality, assume that $0 < k < \Delta < n - 1$, since otherwise the problem is easy. We build the instance $I = (G', w)$ of EWIS$_\alpha$ where $G' = (L, R; E')$ is a bipartite graph as follows:

- For each vertex $v \in V$, we construct a gadget $H(v)$ which is a cycle of length 2Δ. Thus, it is a bipartite graph where the left set is $L_v = \{l_{1,v}, \ldots, l_{\Delta,v}\}$ and the right set is $R_v = \{r_{1,v}, \ldots, r_{\Delta,v}\}$. The weights are $w(l_{i,v}) = 1$ and $w(r_{i,v}) = n\Delta(\frac{2+n\Delta}{2})$ for $i \in [\Delta]$. The gadget $H(v)$ is illustrated in Fig. 5.1.

- For each edge $e \in E$, we construct a gadget $H(e)$ which is also a cycle of length 4. Thus, it is a bipartite graph where the left set is $L_e = \{l_{1,e}, l_{2,e}\}$ and the right set

[1]Every connected graph in this class is either a cycle or a path, and the treewidth of such graphs is at most 2. By Corollary 5.3.1 and Theorem 5.3.19 from Section 5.3, the problem is solvable in pseudo-polynomial time in this class.

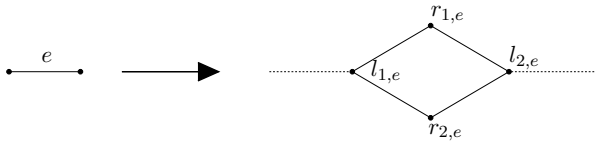

Figure 5.2: The gadget $H(e)$.

is $R_e = \{r_{1,e}, r_{2,e}\}$. The weights are $w(l_{i,e}) = \frac{(n\Delta)}{2}$ and $w(r_{i,e}) = \frac{(n\Delta)^2}{2}(\frac{2+n\Delta}{2})$ for $i = 1, 2$. The gadget $H(e)$ is illustrated in Fig. 5.2.

- We interconnect these gadgets by iteratively applying the following procedure. For each edge $e = \{u, v\} \in E$, we add two edges $\{r_{i,u}, l_{1,e}\}$ and $\{l_{i,u}, r_{1,e}\}$ where $l_{i,u}$ is a neighbor of $r_{i,u}$ in $H(u)$ between gadgets $H(u)$, $H(e)$ and two edges $\{r_{j,v}, l_{2,e}\}$ and $\{l_{j,v}, r_{2,e}\}$ where $l_{j,v}$ is a neighbor of $r_{j,v}$ in $H(v)$ between gadgets $H(v)$, $H(e)$ such that the vertices $r_{i,u}$, $l_{i,u}$, $r_{j,v}$ and $l_{j,v}$ have degree 3.

It is clear that G' is bipartite and the weights are polynomially bounded. Moreover, since G is a Δ-regular graph, we conclude that G' is 3-regular.

We claim that there exist a clique V^* of G with size at least k if and only if $\text{EWIS}_\alpha(G', w, b)$ is *yes* with

$$b = k\Delta + n\Delta\frac{k(k-1)}{2} + n\Delta(\frac{2+n\Delta}{2})\left((n-k)\Delta + (\frac{n\Delta}{2} - \frac{k(k-1)}{2})n\Delta\right).$$

Let I be a maximum independent set of G' with $w(I) = b$. Since G' is cubic and bipartite, G' has a perfect matching (for instance, take a perfect matching in each gadget $H(v)$ and $H(e)$), and we conclude that $\alpha(G) = |I| = |R| = |L|$. This implies in particular that for any vertex $v \in V$, either L_v or R_v is a subset of I. Moreover, the same property holds for any $e \in E$ (i.e., either L_e or R_e is a subset of I). Moreover, by construction of the weights, the quantity $k\Delta$ of b must come from vertices $l_{i,v}$, $r_{i,v}$ or $l_{i,e}$. Since $k < n$, this quantity cannot come from $r_{i,v}$. Moreover, since $l_{i,e} \in I$ if and only if $L_e \subseteq I$, the contribution of L_e in I is $n\Delta$. In this case, the contribution of $k\Delta$ must come from $l_{i,v}$. Thus, we obtain:

$$|I \cap L_V| = k\Delta\,, \qquad |I \cap R_V| = (n-k)\Delta. \tag{5.1}$$

where $L_V = \cup_{v \in V} L_v$ and $R_V = \cup_{v \in V} R_v$. Thus, using (5.1) we must obtain:

$$w\left(I \cap (L_E \cup R_E)\right) = n\Delta\frac{k(k-1)}{2} + n\Delta(\frac{2+n\Delta}{2})(\frac{n\Delta}{2} - \frac{k(k-1)}{2})n\Delta. \tag{5.2}$$

where $L_E = \cup_{e \in E} L_e$ and $R_E = \cup_{e \in E} R_e$. Now, we prove that there are exactly $k(\frac{k-1}{2})$ gadgets $H(e)$ with $L_e \subseteq I$. Assume the reverse; then, $|I \cap L_E| = k(k-1) - 2p$ and $|I \cap R_E| = n\Delta - k(k-1) + 2p$ for some $p \neq 0$ (p can be negative). Combining these equalities with equality (5.2), we deduce that $p = 0$, contradiction.

Thus, if we set $V^* = \{v \in V : L_v \subseteq I\}$, we deduce from previously $|V^*| = k$ and we will have necessarily that V^* is a clique of G.

Conversely, let V' be a clique of G with $|V'| \geq k$ and consider a subclique $V^* \subseteq V'$ of size exactly k. We set $S = S_L \cup S_R$ with $S_L = \cup_{v \in V^*} L_v \cup_{e \in E(V^*)} L_e$ and $S_R = \cup_{v \in V \setminus V^*} R_v \cup_{e \in E \setminus E(V^*)} R_e$. One can easily verify that $w(I) = b$ and that I is a maximum independent set of G'. Indeed, let us assume the converse; thus, there exist $r_{i,v} \in I$ (and thus $R_v \subseteq I$), $l_{j,e} \in I$ (with $j = 1, 2$) and $\{r_{i,v}, l_{j,e}\} \in E'$. By construction of I, we deduce that $e = \{u, v\} \in E(V^*)$ and then $L_v \subseteq I$, contradiction. The proof is complete. □

As corollary of Theorem 5.2.1, we can derive that the biclique problem remains NP-complete when the minimum degree of $G = (L, R; E)$ is $n - 3$ where $|L| = |R| = n$. In this case, we replace any gadget $H(e)$ of Theorem 5.2.1 by a cycle of length $2n\Delta$ and we delete edges $\{l_{i,u}, r_{1,e}\}$ and $\{l_{j,v}, r_{2,e}\}$.

We also remark that Theorem 5.2.1 implies the strong NP-completeness of EWIS_α for perfect graphs, a well-known class where MAXIMUM WEIGHT INDEPENDENT SET problem is solvable in polynomial time.

Let us now strengthen the result of Theorem 5.2.1 to a more general setting: for hereditary subclasses of bipartite graphs of maximum degree 3. By analogy with Theorem 1.3.1, we obtain the following result.

Theorem 5.2.2. *Let X be the class of $\mathcal{F}$-free bipartite graphs. If $\kappa(\mathcal{F}_3) < \infty$, then the* EXACT WEIGHTED MAXIMUM INDEPENDENT SET *problem is strongly NP-complete in the class X_3.*

Proof. The problem is clearly in NP. We show completeness in two steps. First, for $k \geq 3$, let $\mathcal{S}_k$ be the class of all bipartite $(C_3, \ldots, C_k, H_1, \ldots, H_k)$-free graphs of vertex degree at most 3, and let us show that for any fixed k, the problem is strongly NP-complete for graphs in $\mathcal{S}_k$. Let (G, w, b) be an instance of the EWIS$_\alpha$ problem where G is a bipartite graph of maximum degree at most 3.

We can transform the graph G in polynomial time to a weighted graph G', as follows. Let $k' = \lceil \frac{k}{2} \rceil$. We replace each edge e of G by a path $P(e)$ on $2k' + 2$ vertices. Let $N = w(V) + 1$. We set the weights w' of the endpoints of $P(e)$ equal to the weights of the corresponding endpoints of e, while each internal vertex of $P(e)$ gets weight N. It is easy to verify that G' belongs to $\mathcal{S}_k$.

We claim that EWIS$_\alpha(G, w, b)$ is *yes* if and only if EWIS$_\alpha(G', w', b + mk'N)$ is *yes*, where $m = |E(G)|$.

One direction is immediate, as each maximum independent set of G can be extended to a maximum independent set of G', by simply adding k' internal vertices of each newly added path. Doing so, the weight increases by $mk'N$.

Suppose now that EWIS$_\alpha(G', w', b + mk'N)$ is *yes*. Let I' be a maximum independent set of G' of weight $b + mk'N$. Since I' is independent, it can contain at most k' internal vertices of each newly added path. Therefore, for each $e \in E(G)$, the set I' must contain *exactly* k' internal vertices of $P(e)$ – otherwise its weight would be at most $W + (mk' - 1)N$, contradicting our choice of N.

Let I denote the set, obtained from I' by deleting the internal vertices of newly added paths. Then, I is an independent set of G. Indeed, if $e = \{u, v\} \in E(G)$ for some $u, v \in I$, then I' can contain at most $k' - 1$ internal vertices of $P(e)$, contradicting the above observation. Also, it is easy to see that I is a maximum independent set of G. Finally, as the weight of I is exactly b, we conclude that EWIS$_\alpha(G, w, b)$ is *yes*.

This shows that the EWIS$_\alpha$ problem is strongly NP-complete in the class $\mathcal{S}_k$. To prove

strong NP-completeness of the problem in the class X, we now show that the class X contains all graphs in $\mathcal{S}_k$, for $k := \max\{3, \kappa(\mathcal{F}_3)\}$. Let G be a graph from $\mathcal{S}_k$. Assume that G does not belong to X. Then G contains a graph $A \in \mathcal{F}_3$ as an induced subgraph. From the choice of G we know that A belongs to $\mathcal{S}_k$, but then $k < \kappa(A) \leq \kappa(\mathcal{F}_3) \leq k$, a contradiction. Therefore, $G \in X_3$ and the theorem is proved. □

5.3 Polynomial Results

In this section, we present pseudo-polynomial solutions to the exact weighted independent set problem, when the input graphs are restricted to particular classes. Our algorithms mostly resemble those for the MWIS problem in respective graph classes, and are based either on a dynamic programming approach (Section 5.3.1), or on modular decomposition (Section 5.3.2).

First, we observe that when developing polynomial-time solutions to the EWIS problem, we may restrict our attention to connected graphs.

Lemma 5.3.1. *Let (G, w, b) be an instance of the* EWIS *problem, and let $C_1, \ldots, C_r$ be the connected components of G. Suppose that for each $i \in [r]$, the set of solutions $(\text{EWIS}(C_i, w, k) : k \in [b])$ for C_i is given. Then, we can compute the set of solutions $(\text{EWIS}(G, w, k) : k \in [b])$ for G in time $O(rb^2)$.*

In order to show Lemma 5.3.1, we consider the following generalization of the subset sum problem.

Generalized Subset Sum (GSS)

Instance: Nonempty sets of positive integers $A_1, \ldots, A_n$ and a positive integer b.

Question: Is there a nonempty subset J of $[n]$ and a mapping $a : J \to \cup_{j \in J} A_j$ such that $a(j) \in A_j$ for all $j \in J$, and $\sum_{j \in J} a(j) = b$?

By generalizing the dynamic programming solution to the subset sum problem, it is easy to show the following.

Lemma 5.3.2. *Generalized subset sum can be solved in time $O(nb^2)$ by dynamic programming.*

In fact, in the stated time, not only we can verify if there is a $J \subseteq [n]$ and a mapping a as above such that $\sum_{j \in J} a(j) = b$ for the given b, but we can answer this question for *all* values $b' \in [b]$.

Proof. Let B denote the set of *all* values $b' \in [b]$ such that there a nonempty subset S of $[n]$ and a mapping $a : S \to \cup_{i \in S} A_i$ such that $a(i) \in A_i$ for all $i \in S$, and $\sum_{i \in S} a(i) = b'$.

Let us show by induction on n that we can generate B in time $O(nb^2)$. The statement is trivial for $n = 1$.

Suppose now that $n > 1$. Let $I = (A_1, \ldots, A_n; b)$ be an instance of the GSS problem. Let B' be the inductively constructed set of all possible values of $b' \in [b]$ such that the solution to the GSS problem on the instance $(A_1, \ldots, A_{n-1}; b')$ is *yes*. By induction, the set B' was constructed in time $O((n-1)b^2)$.

Let $\beta \in [b]$. Then, β will belong to B, i.e., the solution to the GSS problem, given $(A_1, \ldots, A_n; \beta)$, will be *yes*, if and only if either $\beta \in B'$, or we can write β as $\beta = b' + a_n$ for some $b' \in B'$ and $a_n \in A_n$. In other words, $B = B' \cup B''$, where B'' denotes the set of all such sums: $B'' = \{b' + a_n : b' \in B', a_n \in A_n, b' + a_n \leq b\}$.

The set B'' can be constructed in time $O(b^2)$. Adding this time complexity to the time $O((n-1)b^2)$ needed to construct B' proves the above statement and hence the lemma. □

Lemma 5.3.1 now follows immediately.

Lemma 5.3.1. It is enough to observe that for every $k \in [b]$, EWIS(G, w, k) is *yes* if and only if the solution to the GSS problem on the instance $(A_1, \ldots, A_r; k)$ is *yes*, where A_i denotes the set of all values $k' \in [b]$ such that EWIS(C_i, w, k') is *yes*. □

5.3.1 Dynamic Programming Solutions

We can summarize the results of this subsection in the following theorem.

Theorem 5.3.3. *The* EXACT WEIGHTED INDEPENDENT SET *problem and the* EXACT WEIGHTED MAXIMUM INDEPENDENT SET *problems admit pseudo-polynomial-time solutions in each of*

the following graph classes: mK_2-free graphs, interval graphs and their generalizations k-thin graphs, circle graphs, chordal graphs, AT-free graphs, $(claw, net)$-free graphs, distance-hereditary graphs, graphs of treewidth at most k, and graphs of clique-width at most k.

The rest of this subsection is devoted to proving this result. By part (ii) of Lemma 5.1.1, it suffices to develop pseudo-polynomial solutions for the EXACT WEIGHTED INDEPENDENT SET problem. The algorithms mostly resemble those for the MAXIMUM WEIGHT INDEPENDENT SET problem and exploit the special structure of graphs in the classes.

mK_2-free Graphs

Our first example deals with graphs with no large induced matchings. Recall that K_2 denotes the graph consisting of two adjacent vertices. The disjoint union of m copies of K_2 is denoted by mK_2. Thus, graphs whose largest induced matching consists of less than m edges are precisely the mK_2-free graphs.

Theorem 5.3.4. *For every positive integer m, the* EXACT WEIGHTED INDEPENDENT SET *problem admits a pseudo-polynomial algorithm for mK_2-free graphs.*

Proof. All maximal independent sets $I_1, \ldots, I_N$ in an mK_2-free graph can be found in polynomial time [2, 8, 101, 109]. Since every independent set is contained in a maximal one, EWIS(G, w, k) will take the value *yes* if and only if there is an $i \in [N]$ such that EWIS$(G[I_i], w, k)$ is *yes*. Thus, the EWIS problem in mK_2-free graphs reduces to solving polynomially many instances of the subset sum problem. □

Interval Graphs

Interval graphs are one of the most natural and well-understood classes of intersection graphs. They are intersection graphs of intervals on the real line, and many optimization problems can be solved by dynamic programming on these graphs.

Formally, given a collection $\mathcal{I} = ([a_i, b_i] : i \in I)$ of intervals on the real line, its *intersection graph* $G(\mathcal{I})$ is defined by $V(G(\mathcal{I})) = \mathcal{I}$, and there is an edge connecting $[a_i, b_i]$ and $[a_j, b_j]$ if and only if $[a_i, b_i] \cap [a_j, b_j] \neq \emptyset$. The collection $\mathcal{I}$ is said to be an *interval model* of

$G(\mathcal{I})$. A graph G is said to be an *interval graph* if it admits an interval model, i.e., if there is a collection $\mathcal{I}$ of intervals on the real line such that $G = G(\mathcal{I})$.

A representation of interval graphs that is particularly suitable for the EWIS problem is the following. It was shown by Ramalingam and Pandu Rangan [102] that a graph $G = (V, E)$ is interval if and only if it admits a vertex ordering $(v_1, \ldots, v_n)$ such that for all triples (r, s, t) with $1 \leq r < s < t \leq n$, the following implication is true:

$$\text{if } \{v_r, v_t\} \in E \text{ then } \{v_s, v_t\} \in E\,.$$

Moreover, such an ordering of an interval graph can be found in time $O(n + m)$. Based on this ordering, we can prove the following statement.

Theorem 5.3.5. *The* EXACT WEIGHTED INDEPENDENT SET *problem admits an $O(bn+m)$ algorithm for interval graphs.*

Proof. Let $(v_1, \ldots, v_n)$ be a vertex ordering such that $\{v_s, v_t\} \in E$, whenever $\{v_r, v_t\} \in E$, for all triples (r, s, t) with $1 \leq r < s < t \leq n$.

For every $i \in [n]$, let G_i denote the subgraph of G induced by $\{v_1, \ldots, v_i\}$ (also, let G_0 be the empty graph). Then, for every $i \in [n]$, either there is a $j = j(i)$ such that $N_{G_i}(v_i) = \{j, j+1, \ldots, i-1\}$, or $N_{G_i}(v_i) = \emptyset$ (in which case let us define $j(i) = i$). Now, if I is an independent set of G_i, then either $v_i \in I$ (in which case $I \backslash \{v_i\}$ is an independent set of $G_{j(i)-1}$), or $v_i \notin I$ (in which case I is an independent set of G_{i-1}). This observation is the key to the following simple $O(bn + m)$ dynamic programming solution to the EWIS problem on interval graphs.

Step 1. Find a vertex ordering $(v_1, \ldots, v_n)$ as above.

Step 2. Set EWIS(G_0, w, k) to *no* for all $k \in [b]$.

Step 3. For $i = 1, \ldots, n$, do the following:

3.1. Find $j \in [i]$ such that $N_{G_i}(v_i) = \{j, j+1, \ldots, i-1\}$.

3.2. For $k \in [b]$, do the following:

If $k = w(v_i)$, set EWIS(G_i, w, k) to *yes*.

If $k < w(v_i)$, set EWIS(G_i, w, k) to EWIS(G_{i-1}, w, k).

If $k > w(v_i)$, set EWIS(G_i, w, k) to *yes* if at least one of the solutions to EWIS$(G_{j(i)-1}, w, k - w(v_i))$ and EWIS(G_{i-1}, w, k) is *yes*, and to *no* otherwise.

Step 4. Output EWIS(G_n, w, b). □

k-thin Graphs

The property used in the above characterization of interval graphs has been generalized by Mannino *et al.* in [86], where they define the class of k-thin graphs. A graph $G = (V, E)$ is said to be *k-thin* if there exist an ordering $(v_1, \ldots, v_n)$ of V and a partition of V into k classes such that, for each triple (r, s, t) with $1 \leq r < s < t \leq n$, if v_r, v_s belong to the same class and $\{v_r, v_t\} \in E$, then $\{v_s, v_t\} \in E$.

Let us mention at this point that finding a feasible frequency assignment of a given cost can be modeled as the EWIS problem on a k-thin graph, where the parameter k depends on the input to the frequency assignment problem. For further details, we refer the reader to the paper [86].

Based on the same idea as for interval graphs, a dynamic programming solution for k-thin graphs can be obtained, provided we are given an ordering and a partition of the vertex set.

Theorem 5.3.6. *Suppose that for a k-thin graph $G = (V, E)$, $k \geq 2$, an ordering $(v_1, \ldots, v_n)$ of V and a partition of V into k classes are given such that, for each triple (r, s, t) with $1 \leq r < s < t \leq n$, if v_r, v_s belong to the same class and $\{v_r, v_t\} \in E$, then $\{v_s, v_t\} \in E$. Then, the* EXACT WEIGHTED INDEPENDENT SET *problem admits an $O(bn^k)$ algorithm for G.*

Proof. The proof is a straightforward extension of the proof of Theorem 5.3.5. Let $V_1, \ldots, V_k$ be the classes of the partition. Instead of the graphs G_i, induced by the first i vertices, we now consider all graphs $G(i_1, \ldots, i_k)$, induced by the "first" i_r vertices of each class (according to the ordering on V restricted to the class), for all $r \in \{1, \ldots, k\}$, and for all $O(n^k)$ choices of such k-tuples $(i_1, \ldots, i_k) \in \{1, \ldots, |V_1|\} \times \ldots \times \{1, \ldots, |V_k|\}$. □

Circle Graphs

Besides intervals on the real line, chords on a circle provide another popular intersection model. The intersection graphs of chords on a circle are called *circle graphs*. In this subsection, we present a $O(b^2n^2)$ dynamic-programming algorithm for the EWIS problem in circle graphs. Our algorithm for EWIS on circle graphs is based on the dynamic programming solution for the MAXIMUM INDEPENDENT SET problem, developed by Supowit in [107].

Theorem 5.3.7. *The* EXACT WEIGHTED INDEPENDENT SET *problem admits an* $O(b^2n^2)$ *algorithm for circle graphs.*

Proof. Consider a finite set of N chords on a circle. We may assume without loss of generality that no two chords share an endpoint. Number the endpoints of the chords from 1 to $2N$ in the order as they appear as we move clockwise around the circle (from an arbitrary but fixed starting point).

The idea is simple. For $1 \leq i < j \leq 2N$, let $G(i,j)$ denote the subgraph of G induced by chords whose both endpoints belong to the set $\{i, i+1, \ldots, j\}$. Obviously $G = G(1, 2N)$.

Let $1 \leq i < j \leq 2N$. If $j = i+1$ then EWIS$(G(i,j), w, k)$ is *yes* if and only if either $k = 0$, or $(i, i+1)$ is a chord and $k = w((i, i+1))$.

Otherwise, let r be the other endpoint of the chord whose one endpoint is j. If $r < i$ or $r > j$, then no independent set of the graph $G(i,j)$ contains the chord (r,j), so EWIS$(G(i,j), w, k)$ is *yes* if and only if EWIS$(G(i,j-1), w, k)$ is *yes*. Suppose now that $i \leq r \leq j-1$ and let I be an independent set of $G(i,j)$. The set I may or may not contain the chord (r,j). If I does not contain (r,j), then I is an independent set of of $G(i,j-1)$ as well. If I contains (r,j), then no other chord in I can intersect the chord (r,j). In particular, this implies that I is of the form $I = \{(r,j)\} \cup I_1 \cup I_2$ where I_1 is an independent set of $G(i, r-1)$ and I_2 is an independent set of $G(r+1, j-1)$.

Therefore, EWIS$(G(i,j), w, k)$ is *yes* if and only if either EWIS$(G(i,j-1), w, k)$ is *yes*, or EWIS(G', w, k) is *yes*, where G' is the graph whose connected components are $G[\{(r,j)\}]$, $G(i, r-1)$ and $G(r+1, j-1)$. Assuming that the solutions for $G(i, r-1)$ and $G(r+1, j-1)$ have already been obtained recursively, we can apply Corollary 5.3.1 in this case.

The above discussion implies an obvious $O(b^2n^2)$ algorithm that correctly solves the problem. □

Chordal Graphs

Chordal (or triangulated) graphs are graphs in which every cycle of length at least four has a chord. They strictly generalize interval graphs and provide another class where the MWIS problem is polynomially solvable. Unfortunately for our purpose, the usual approaches for the MWIS problem in chordal graphs ([50, 108]) heavily rely on the maximization nature of the problem, and generally do not preserve the overall structure of independent sets. As such, they do not seem to be directly extendable to the exact version of the problem. Instead, we develop a pseudo-polynomial time solution to the EWIS problem in chordal graphs by using one of the many characterizations of chordal graphs: their *clique tree representation*.

Theorem 5.3.8. *The* EXACT WEIGHTED INDEPENDENT SET *problem admits an* $O(b^2n(n+m))$ *algorithm for chordal graphs.*

Proof. Given a chordal graph G, we first compute a *clique tree* of G. This can be done in time $O(n+m)$ [73]. A clique tree of a chordal graph G is a tree T whose nodes are the maximal cliques of G, such that for every vertex v of G, the subgraph T_v of T induced by the maximal cliques containing v is a tree. Furthermore, we fix an arbitrary node K_r in the clique tree in order to obtain a *rooted clique tree*. For a maximal clique K, we denote by $G(K)$ the subgraph of G induced by the vertices of K and all vertices contained in some descendant of K in T.

The algorithm is based on a set of identities developed by Okamoto, Uno and Uehara in [97], where a clique tree representation was used to develop linear-time algorithms to count independent sets in a chordal graph. Let $\mathcal{IS}(G)$ be the family of independent sets in G. For a vertex v, let $\mathcal{IS}(G,v)$ be the family of independent sets in G that contain v. For a vertex set U, let $\overline{\mathcal{IS}}(G,U)$ be the family of independent sets in G that contain no vertex of U. Consider a maximal clique K of G, and let $K_1,\ldots,K_l$ be the children of K in T. (If K is a leaf of the clique tree, we set $l=0$.) Then, as shown in [97], for every distinct

$i, j \in [l]$, the sets $V(G(K_i))\backslash K$ and $V(G(K_j))\backslash K$ are disjoint. Moreover, if $\sqcup$ denotes the disjoint union, the following relations hold:

$$\begin{aligned}
\mathcal{IS}(G(K)) &= \overline{\mathcal{IS}}(G(K), K) \sqcup \bigsqcup_{v \in K} \mathcal{IS}(G(K), v)\,; \\
\mathcal{IS}(G(K), v) &= \left\{ I \cup \{v\} \;\middle|\; I = \bigcup_{i=1}^{l} I_i,\ I_i \in \begin{cases} \mathcal{IS}(G(K_i), v), & \text{if } v \in K_i; \\ \overline{\mathcal{IS}}(G(K_i), K \cap K_i), & \text{otherwise.} \end{cases} \right\}; \\
\overline{\mathcal{IS}}(G(K), K) &= \left\{ I \;\middle|\; I = \bigsqcup_{i=1}^{l} I_i,\ I_i \in \overline{\mathcal{IS}}(G(K_i), K \cap K_i) \right\}; \\
\overline{\mathcal{IS}}(G(K_i), K \cap K_i) &= \overline{\mathcal{IS}}(G(K_i), K_i)) \sqcup \bigsqcup_{u \in K_i \backslash K} \mathcal{IS}(G(K_i), u) \quad \text{for each } i \in [l]\,.
\end{aligned}$$

We extend our usual Boolean predicate $\mathrm{EWIS}(H, w, k)$ to the following two: for a vertex v of a weighted graph (H, w) and in integer k, let $\mathrm{EWIS}(H, w, k, v)$ denote the Boolean predicate that is *yes* if and only if in H there is an independent set I of total weight k that contains v. Also, for a set of vertices U let $\overline{\mathrm{EWIS}}(H, w, k, U)$ take the value *yes* if and only if in H there is an independent set of total weight k that contains no vertex from U. Based on the above equations, we can develop the following recursive relations for EWIS:

$$\mathrm{EWIS}(G(K), w, k) = \overline{\mathrm{EWIS}}(G(K), w, k, K) \vee \bigvee_{v \in K : w(v) \leq k} \mathrm{EWIS}(G(K), w, k, v) \tag{5.3}$$

where $\vee$ denotes the usual *Boolean OR function* (with the obvious identification *yes* $\leftrightarrow 1$, *no* $\leftrightarrow 0$). That is, its value is *yes* if at least one of its arguments is *yes*, and *no* otherwise.

$$\mathrm{EWIS}(G(K), w, k, v) = \mathbf{GSS}(A_1, \ldots, A_l, k - w(v)) \tag{5.4}$$

where $\mathbf{GSS}(A_1, \ldots, A_l, k)$ denotes the solution to the generalized subset sum problem on

the input instance $(A_1, \ldots, A_l, k)$, where the sets A_i for $i \in [l]$ are given by

$$A_i = \begin{cases} \{k' - w(v) \ : \ w(v) \leq k' \leq k, \ \text{EWIS}(G(K_i), w, k', v) = yes\}, & \text{if } v \in K_i; \\ \{k' \ : \ 1 \leq k' \leq k, \ \overline{\text{EWIS}}(G(K_i), w, k', K \cap K_i) = yes\}, & \text{otherwise.} \end{cases}$$

Note that if $I_i \in \mathcal{IS}(G(K_i), v)$ and $I_j \in \mathcal{IS}(G(K_j), v)$ for some distinct indices $i, j \in [l]$, then we have $I_i \cap I_j = \{v\}$. Moreover, since this is the only possible nonempty intersection of two independent sets from $\bigcup_{i=1}^{l} I_i$ in the equation for $\mathcal{IS}(G(K), v)$, it follows that the sum of the weights of the sets $I_i \backslash \{v\}$ (over all $i \in [l]$) equals to the weight of $\left(\bigcup_{i=1}^{l} I_i\right) \backslash \{v\}$, thus justifying Equation (5.4).

Similarly, we have

$$\overline{\text{EWIS}}(G(K), w, k, K) = \mathbf{GSS}(A_1, \ldots, A_l, k) \tag{5.5}$$

where, for each $i \in [l]$, the set A_i is given by

$$A_i = \{k' : 1 \leq k' \leq k, \ \overline{\text{EWIS}}(G(K_i), w, k', K \cap K_i) = yes\},$$

and, finally, for each $i \in [l]$, we have:

$$\overline{\text{EWIS}}(G(K_i), w, k, K \cap K_i) = \overline{\text{EWIS}}(G(K_i, w, k, K_i)) \vee \bigvee_{u \in K_i \backslash K} \text{EWIS}(G(K_i), w, k, u). \tag{5.6}$$

Given the above equations, it is now easy to develop a pseudo-polynomial dynamic programming algorithm. Having constructed a rooted tree T of G, we traverse it in a bottom-up manner. For a leaf K, we have

$$\overline{\text{EWIS}}(G(K), w, k, K) = \begin{cases} yes, & \text{if } k = 0; \\ no, & \text{otherwise.} \end{cases}$$

and

$$\text{EWIS}(G(K), w, k, v) = \begin{cases} yes, & \text{if } w(v) = k; \\ no, & \text{otherwise.} \end{cases}.$$

For every other node K, we compute the values of $\overline{\text{EWIS}}(G(K), w, k, K)$ and $\text{EWIS}(G(K), w, k, v)$ by referring to the recursive relations (5.6), (5.5) and (5.4) in this order. Finally, the value of $\text{EWIS}(G, w, k)$ is given by $\text{EWIS}(G(K_r), w, k)$, which can be computed using Equation (5.3).

The correctness of the procedure follows immediately from the above discussion. To justify the time complexity, observe that in a node K of the tree with children $K_1, \ldots, K_l$, the number of operations performed is $O(\sum_{i=1}^{l} |K_i| + lM^2 + |K|lM^2)$. Summing up over all the nodes of the clique tree, and using the fact that a chordal graph has at most n maximal cliques, which satisfy $\sum_{K \in V(T)} |K| = O(n + m)$ [97], the claimed complexity bound follows. □

AT-free Graphs

The class of *AT-free graphs* is another class that contains the interval graphs. Moreover, AT-free graphs contain other well-known subclasses of perfect graphs, for instance *permutation graphs* and their superclass, the class of *co-comparability graphs*.

A triple $\{x, y, z\}$ of pairwise non-adjacent vertices in a graph G is an *asteroidal triple* if for every two of these vertices there is a path between them avoiding the closed neighborhood of the third. Formally, x and y are in the same component of $G - N[z]$, x and z are in the same component of $G - N[y]$, and y and z are in the same component of $G - N[y]$. A graph is called *AT-free* if it has no asteroidal triples.

Our dynamic programming algorithm that solves EWIS for AT-free graphs is based on the dynamic programming approach to the MWIS problem in AT-free graphs, developed by Broersma, Kloks, Kratsch and Msüller in [31]. Let us start with a definition.

Definition 5.3.9. *Let x and y be two distinct nonadjacent vertices of an AT-free graph G. The interval $I(x, y)$ is the set of all vertices z of $V(G)\backslash\{x, y\}$ such that x and z are in one component of $G - N[y]$, and z and y are in one component of $G - N[x]$.*

Now, we recall some structural results from [31].

Theorem 5.3.10 ([31])**.** *Let $I = I(x, y)$ be a nonempty interval of an AT-free graph G, and let $s \in I$. Then there exist components $C_1^s, \ldots, C_t^s$ of $G - N[s]$ such that the components of*

$I \backslash N[s]$ are precisely $I(x,s)$, $I(s,y)$, and $C_1^s, \ldots, C_t^s$.

Theorem 5.3.11 ([31])**.** *Let G be an AT-free graph, let C be a component of $G - N[x]$, let $y \in C$, and let D be a component of the graph $C - N[y]$. Then $N[D] \cap (N[x] \backslash N[y]) = \emptyset$ if and only if D is a component of $G - N[y]$.*

Theorem 5.3.12 ([31])**.** *Let G be an AT-free graph, let C be a component of $G - N[x]$, let $y \in C$, and let C' be the component of $G - N[y]$ that contains x. Let $B_1, \ldots, B_l$ denote the components of the graph $C - N[y]$ that are contained in C'. Then $I(x,y) = \cup_{i=1}^{l} B_i$.*

We will also need the following simple observation.

Proposition 5.3.13. *Let (G,w) be a weighted graph. Then,* EWIS(G,w,k) *is yes if and only if there is a vertex $x \in V(G)$ such that* EWIS$(G - N(x), w, k)$ *is yes.*

Combining Proposition 5.3.13 with Theorems 5.3.11 and 5.3.12, we obtain the following lemma.

Lemma 5.3.14. *Let (G,w) be a weighted AT-free graph, $G = (V,E)$. Let $x \in V$ and let C be a component of $G - N[x]$. For a vertex y of C, let C_y denote the subgraph of G induced by $C - N(y)$. Then,* EWIS(C,w,k) *is yes if and only if there is a vertex $y \in C$ such that* EWIS(C_y,w,k) *is yes. Moreover, the connected components of such a C_y are precisely $\{y\}$, $I(x,y)$, and the components of $G - N[y]$ contained in C.*

Similarly, using Theorem 5.3.10 we obtain the following conclusion.

Lemma 5.3.15. *Let (G,w) be a weighted AT-free graph, $G = (V,E)$. Let $I = I(x,y)$ be an interval of G. If $I = \emptyset$, then* EWIS$(G[I],w,k)$ *is yes if and only if $k = 0$. Otherwise, let us denote by I_s the subgraph of G induced by $I - N(s)$, for all $s \in I$. Then,* EWIS(I,w,k) *is yes if and only if there is a vertex $s \in I$ such that* EWIS(I_s,w,k) *is yes. Moreover, the connected components of such an I_s are precisely $\{s\}$, $I(x,s)$, $I(s,y)$, and the components of $G - N[s]$ contained in I.*

Theorem 5.3.16. *The* EXACT WEIGHTED INDEPENDENT SET *problem admits a pseudo-polynomial algorithm for AT-free graphs.*

Proof. It follows from the above discussion that the following pseudo-polynomial algorithm correctly solves the problem.

Step 1. For every $x \in V$ compute all components of $G - N[x]$.

Step 2. For every pair of nonadjacent vertices $x, y \in V(G)$ compute the interval $I(x, y)$.

Step 3. Sort all the components and intervals according to nonincreasing number of vertices.

Step 4. In the order of Step 3, compute the solutions to EWIS(C, w, k), for each component C (for all $k \in \{0, 1, \ldots, w(C)\}$) and the solutions to EWIS(I, w, k) for each interval I (for all $k \in \{0, 1, \ldots, w(I)\}$). To compute the solutions to EWIS(C, w, k) for a component C, first compute the solutions to EWIS$(C - N(y), w, k)$, for all $y \in C$, by applying Lemma 5.3.14 and Corollary 5.3.1. Similarly, to compute the solutions to EWIS(I, w, k) for an interval I, first compute the solutions to EWIS$(I - N(s), w, k)$, for all $s \in I$, by applying Lemma 5.3.15 and Corollary 5.3.1.

Step 5. Compute EWIS(G, w, k) using Observation 5.3.13 and Corollary 5.3.1. □

In [21], it is shown that for every vertex v of a $(claw, net)$-free graph G, the non-neighborhood of v in G is AT-free.[2] Thus, Theorem 5.3.16 immediately implies the following result.

Corollary 5.3.17. *The* EXACT WEIGHTED INDEPENDENT SET *problem admits a pseudo-polynomial algorithm for* $(claw, net)$*-free graphs.*

Distance Hereditary Graphs

A graph is *distance-hereditary* if the distance between any two connected vertices (that is, vertices in the same connected component) is the same in every induced subgraph in which they remain connected. Bandelt and Mulder provided in [9] a *pruning sequence* characterization of distance-hereditary graphs: whenever a graph contains a vertex of degree one, or a vertex with a twin (another vertex sharing the same neighbors), remove such a vertex. A graph is distance-hereditary if and only if it the application of such vertex removals results in a single-vertex graph.

[2] A *net* is the graph obtained from a triangle by attaching one pendant edge to each vertex.

More formally, a *pruning sequence* of a distance-hereditary graph G is a sequence of the form $\sigma = (x_1R_1y_1, x_2R_2y_2, \ldots, x_{n-1}R_{n-1}y_{n-1})$ where $(x_1, \ldots, x_n)$ is a total ordering of $V(G)$ such that for all $i \in \{1, \ldots, n-1\}$, the following holds:

- $R_i \in \{P, T, F\}$.
- If we denote by G_i the subgraph of G induced by $\{x_i, \ldots, x_n\}$, then:
  - If $R_i = P$ then x_i is a *pendant vertex*, that is, a vertex of degree one in the graph G_i, with $N_{G_i}(x_i) = \{y_i\}$.
  - If $R_i = T$ then x_i and y_i are *true twins* in G_i, that is, $N_{G_i}[x_i] = N_{G_i}[y_i]$.
  - If $R_i = F$ then x_i and y_i are *false twins* in G_i, that is, $N_{G_i}(x_i) = N_{G_i}(y_i)$.

A pruning sequence of a distance-hereditary graph can be computed in linear time [35] and can be useful for algorithmic developments on distance-hereditary graphs. A solution to the MWIS problem in distance-hereditary graphs based on the pruning sequence characterization has been developed by Cogis and Thierry in [32]. It turns out that their approach can be generalized in order to solve the exact version of the problem.

Theorem 5.3.18. *The* EXACT WEIGHTED INDEPENDENT SET *problem admits an* $O(b^2n + m)$ *algorithm for distance-hereditary graphs.*

Proof. We first define an auxiliary problem:

P1(G, b, p, q)

Instance: A graph G, a positive integer b, and two functions

$$p, q : V \times \{0, 1, \ldots, b\} \to \{0, 1\}.$$

Question: Is there an independent set I of G, and a mapping $w : V \to \{0, 1, \ldots, b\}$ such that the following holds:

- $\sum_{x \in V} w(x) = b$,
- $p(x, w(x)) = 1$ whenever $x \in I$,
- $q(x, w(x)) = 1$ whenever $x \notin I$?

Let us show that the EWIS problem is polynomially reducible to **P1**. Let (G, w, b) be an instance to the EWIS problem. Define $p, q, : V(G) \times \{0, 1, \ldots, b\} \to \{0, 1\}$ as follows. For each $x \in V(G)$ and each $k \in \{0, 1, \ldots, b\}$, let

$$p(x,k) = \begin{cases} 1, & \text{if } k = w(x); \\ 0, & \text{otherwise,} \end{cases}$$

and

$$q(x,k) = \begin{cases} 1, & \text{if } k = 0; \\ 0, & \text{otherwise.} \end{cases}$$

Then, it is easy to see that EWIS(G, w, b) is *yes* if and only if **P1**(G, b, p, q) is *yes*.

In what follows, we will present an $O(b^2 n + m)$ to solve the problem **P1** on an instance (G, b, p, q), if G is a distance-hereditary graph. For two functions $f, g : \{0, 1, \ldots, N\} \to \{0, 1\}$, we denote their *convolution* $f * g$ as the function $f * g : \{0, 1, \ldots, N\} \to \{0, 1\}$, given by the following rule: for every $k \in \{0, 1, \ldots, N\}$, we have

$$(f * g)(k) = \begin{cases} 1, & \text{if there is a } k' \in \{0, 1, \ldots, k\} \text{ such that } p(k') = q(k - k') = 1; \\ 0, & \text{otherwise.} \end{cases}$$

Procedure P1-DH

Input: A distance-hereditary graph G, a positive integer b, and two functions $p, q : V \times \{0, 1, \ldots, b\} \to \{0, 1\}$.

Output: The answer to the question in **P1**(G, b, p, q).

Step 1. Compute the pruning sequence $\sigma = (x_1 R_1 y_1, x_2 R_2 y_2, \ldots, x_{n-1} R_{n-1} y_{n-1})$ for G. To each vertex $x \in V(G)$, associate a pair of functions $p^x, q^x : \{0, 1, \ldots, b\} \to \{0, 1\}$, by $p^x(\cdot) = p(x, \cdot)$ and $q^x(\cdot) = q(x, \cdot)$.

Step 2. Check if the pruning sequence is empty. If yes, there is only one vertex x left. If $\max\{p^x(b), q^x(b)\} = 1$, then output *yes*. Else, output no.

Else, let xRy be the head of the pruning sequence. Update the pruning sequence by removing xRy from it. Update p^y and q^y as follows.

- If $R = P$ then let

$$\begin{aligned} p^y(k) &\leftarrow (p^y * q^x)(k)\,, \\ q^y(k) &\leftarrow \max\{(p^x * q^y)(k), (q^x * q^y)(k)\}\,, \end{aligned}$$

  for each $k \in \{0, 1, \ldots, b\}$.

- If $R = T$ then let

$$\begin{aligned} p^y(k) &\leftarrow \max\{(p^y * q^x)(k), (p^x * q^y)(k)\}\,, \\ q^y(k) &\leftarrow (q^x * q^y)(k)\,, \end{aligned}$$

  for each $k \in \{0, 1, \ldots, b\}$.

- If $R = F$ then let

$$\begin{aligned} p^y(k) &\leftarrow \max\{(p^x * q^y)(k), (p^x * p^y)(k), (q^x * p^y)(k)\}\,, \\ q^y(k) &\leftarrow (q^x * q^y)(k)\,, \end{aligned}$$

  for each $k \in \{0, 1, \ldots, b\}$.

Go to *Step 2*.

The correctness of the algorithm can be easily proved by induction on n. We leave this routine proof to the reader. Clearly, the algorithm can be implemented so that it runs in time $O(b^2 n + m)$. □

Graphs of Treewidth at most k

It is easy to see that on trees, the EWIS problem admits a simple dynamic programming solution. With some care, the same approach can be generalized to graphs of bounded treewidth.

Theorem 5.3.19. *For every fixed k, the* EXACT WEIGHTED INDEPENDENT SET *problem admits an $O(b^2 n)$ algorithm for graphs of treewidth at most k.*

A detailed description of the algorithm and its analysis can be found in [89].

Graphs of Clique-width at most k

The *clique-width* of a graph G is defined as the minimum number of labels needed to construct G, using the following four graph operations:

(i) Create a new vertex v with label i (denoted by $i(v)$).

(ii) Take the disjoint union of two labeled graphs G and H (denoted by $G \oplus H$).

(iii) Join by an edge each vertex with label i to each vertex with label j ($i \neq j$, denoted by $\eta_{i,j}$).

(iv) Rename label i to j (denoted by $\rho_{i \to j}$).

An expression built from the above four operations is called a *clique-width expression*. A clique-width expression using k labels is called a *k-expression*. Each k-expression t uniquely defines a labeled graph $lab(t)$, where the labels are integers $\{1, \ldots, k\}$ associated with the vertices and each vertex has exactly one label. We say that a k-expression t defines a graph G if G is equal to the graph obtained from the labeled graph $lab(t)$ after removing its labels. The clique-width of a graph G is equal to the minimum k such that there exists a k-expression defining G.

The clique-width of a graph of treewidth k is bounded above by $3 \cdot 2^{k-1}$ [34]. This implies that a class of graphs with uniformly bounded treewidth is also of bounded clique-width. The complete graphs show that converse is generally not true. In this sense, showing that a problem can be efficiently solved for graphs of bounded clique-width is more general than showing the same statement for graphs of bounded treewidth.

Many graph problems that are NP-hard for general graphs are solvable in linear time when restricted to graphs of clique-width at most k, if a k-expression is given as part of the input.[3] The EWIS problem is no exception.

Theorem 5.3.20. *For every fixed k, the* EXACT WEIGHTED INDEPENDENT SET *problem admits an $O(2^k b^2 l)$ algorithm for graphs of clique-width at most k, where l is the number of operations in a given k-expression for G.*

[3]If only a graph G of clique-width at most k is given, then an $O(2^{6k})$-expression defining G can be computed in $O(n^3)$ time, as shown by Oum in [98].

Proof. Suppose that the labels are integers $\{1, \ldots, k\} = [k]$. For every subset of labels $S \subseteq [k]$, let $\text{EWIS}(G, w, S, m)$ denote the answer to the following question: "Is there an independent set of G with total weight m that contains exactly the labels from S?"

Given a k-expression t defining the input graph G, we can solve $\text{EWIS}(G, w, b)$ by first computing all the values for $\text{EWIS}(G, w, S, m)$, for every subset of labels $S \subseteq [k]$, and every $m \in [b]$. It is easy to see that this can be performed in time $O(b^2 l)$ by the following dynamic programming algorithm.

If $|V| = 1$ then let $v \in V$. For all $S \subseteq [k]$, and for all $m \in [b]$, let

$$\text{EWIS}(G, w, S, m) = \begin{cases} yes, & \text{if } S = \{label(v)\} \text{ and } m = w(v); \\ no, & \text{otherwise.} \end{cases}$$

If $G = G_1 \oplus G_2$ then let for all $S \subseteq [k]$, and for all $m \in [b]$:

$$\text{EWIS}(G, w, S, m) = \begin{cases} yes, & \text{if EWIS}(G_1, w, S, m) = yes; \\ yes, & \text{if EWIS}(G_2, w, S, m) = yes; \\ yes, & \text{if there is an } m' \in [m-1] \text{ such that} \\ & \quad \text{EWIS}(G_1, w, S, m') = \text{EWIS}(G_2, w, S, m - m') = yes; \\ no, & \text{otherwise.} \end{cases}$$

This can be computed in time $O(b^2)$, similarly as in Corollary 5.3.1.

If $G = \eta_{i,j}(G_1)$ then let for all $S \subseteq [k]$, and for all $m \in [b]$:

$$\text{EWIS}(G, w, S, m) = \begin{cases} \text{EWIS}(G_1, w, S, m), & \text{if } \{i, j\} \not\subseteq S; \\ no, & \text{otherwise.} \end{cases}$$

If $G = \rho_{i \to j}(G_1)$ then let for all $S \subseteq [k]$, and for all $m \in [b]$:

$$\text{EWIS}(G, w, S, m) = \begin{cases} \text{EWIS}(G_1, w, S, m), & \text{if } S \cap \{i, j\} = \emptyset; \\ \text{EWIS}(G_1, w, S \cup \{i\}, m), & \text{if } S \cap \{i, j\} = \{j\}; \\ no, & \text{otherwise.} \end{cases}$$

Having computed all the values $\mathrm{EWIS}(G, w, S, m)$, the value of $\mathrm{EWIS}(G, w, b)$ is clearly given by

$$\mathrm{EWIS}(G, w, b) = \begin{cases} \textit{yes}, & \text{if there is an } S \subseteq [k] \text{ such that } \mathrm{EWIS}(G, w, S, b) = \textit{yes}; \\ \textit{no}, & \text{otherwise.} \end{cases}$$

□

Note that the same algorithm runs in pseudo-polynomial time whenever the clique-width of the input graph is of the order $O(\log n)$.

Due to the unknown complexity of the exact perfect matching problem, the problem of determining the complexity of the EWIS problem is of particular interest for line graphs of bipartite graphs, and their subclasses and superclasses. Line graphs of bipartite graphs form a hereditary class of graphs. Their characterization in terms of forbidden induced subgraphs has been obtained in [106], as follows. A graph G is the line graph of a bipartite graph if and only G is $\mathcal{F}$-free, where $\mathcal{F} = \{\textit{claw}, \textit{diamond}, C_5, C_7, \ldots\}$. A *diamond* is the graph obtained by deleting a single edge from a complete graph on 4 vertices.

Keeping in mind this characterization of line graphs of bipartite graphs, it is interesting to consider the following immediate consequence of Theorem 5.3.20.

Corollary 5.3.21. *The* EXACT WEIGHTED INDEPENDENT SET *problem admits a pseudo-polynomial solution in each of the following graph classes:*

- *(claw, co-claw)-free graphs,*
- *(gem, fork, co-P)-free graphs (see Figure 5.3 in Section 5.3.2) and their subclass (claw, diamond, co-P)-free graphs,*
- *(P_5, diamond)-free graphs.*

Proof. Each of the above subclasses is of bounded clique-width (see [30, 29, 20]). □

Also, we can derive from Theorem 5.3.20 a particular complexity result for the exact perfect matching problem.

Corollary 5.3.22. *For every fixed k, the* EXACT PERFECT MATCHING *problem admits a pseudo-polynomial algorithm for graphs of treewidth at most k.*

Proof. As shown by Gurski and Wanke [62], a set X of graphs has bounded treewidth if and only if $L(X) := \{L(G) : G \in X\}$ has bounded clique-width. Since the exact perfect matching problem in X is polynomially equivalent to the problem EWIS_α in the set $L(X)$, the statement follows from Theorem 5.3.20 and part (ii) of Lemma 5.1.1. □

5.3.2 Modular Decomposition

Recall that in Section 2.2 we have seen how modular decomposition can be applied to the MAXIMUM WEIGHT INDEPENDENT SET problem. In this subsection, we show how to apply modular decomposition to the EWIS problem.

We formally describe this reduction for the EWIS problem in the recursive procedure MODULAR_EWIS(G, W, b) below. It turns out that in order to apply this decomposition to the EWIS problem, we need to relax the problem so that each vertex of the input graph is equipped with a nonempty set of possible weights (instead of just a single one). For simplicity, we still name this problem EWIS. When all sets are singletons, the problem coincides with the original EWIS problem.

EXACT WEIGHTED INDEPENDENT SET (EWIS)

Instance: An ordered triple (G, W, b), where $G = (V, E)$ is a graph, b is a positive integer and $W = (W_v : v \in V)$ with $W_v \subseteq [b]$ for all $v \in V$ is the collection of possible weights for each vertex of G.

Question: Is there an independent set I of G and a mapping $w : I \to [b]$ such that $w(v) \in W_v$ for all $v \in I$, and $\sum_{v \in I} w(v) = b$?

In graph classes that are closed under duplicating vertices, this extended version is pseudo-polynomially equivalent to the original one: given an input (G, W, b) to the extended version, we can construct a weighted graph (G', w') from (G, W) by replacing each vertex

v of G with a clique K_v on $|W_v|$ vertices, assigning different weights from W_v to different vertices of K_v, and joining a vertex from K_u with a vertex from K_v by an edge if and only if $\{u, v\}$ was an edge of G. Then, it is clear that EWIS$(G, W, b) = \textit{yes}$ if and only if EWIS$(G', w', b) = \textit{yes}$. However, working with the extended version enables us to apply modular decomposition to *arbitrary* graph classes.

Algorithm MODULAR_EWIS(G, W, b)

Input: An ordered triple (G, W, b), where $G = (V, E)$ is a graph, b is a positive integer and $W = (W_v : v \in V)$ with $W_v \subseteq [b]$ for all $v \in V$ is the collection of possible weights for each vertex of G.

Output: $(\text{EWIS}(G, W, k) : k \in [b])$

1. If $|V| = 1$, say $V = \{v\}$, set, for each $k \in [b]$,

$$\text{EWIS}(G, W, k) = \begin{cases} yes, & \text{if } k \in W_v; \\ no, & \text{otherwise} \end{cases}$$

   and stop.

2. If G is disconnected, partition it into connected components $\mathcal{M}_1, \ldots, \mathcal{M}_r$, and go to step 5.

3. If co-G is disconnected, partition G into co-components $\mathcal{M}_1, \ldots, \mathcal{M}_r$, and go to step 5.

4. If G and co-G are connected, partition G into maximal modules $\mathcal{M}_1, \ldots, \mathcal{M}_r$.

5. For all $j \in [r]$, let

$$(\text{EWIS}(G[\mathcal{M}_j], W, k) : k \in [b]) = \text{MODULAR\_EWIS}(G[\mathcal{M}_j], W, b)\,.$$

   Construct a graph G^0 from G by contracting each $\mathcal{M}_j$ (for $j \in [r]$) to a single vertex, and assign to that vertex the set of weights

$$W_{\mathcal{M}_j} = \{k \in [b] : \text{EWIS}(G[\mathcal{M}_j], W, k) = yes\}\,.$$

6. For each $k \in [b]$, let

$$\text{EWIS}(G, W, k) = \text{EWIS}(G^0, (W_{\mathcal{M}_j} : j \in [r]), k)$$

   and stop.

We remark that for each input graph, at most one of the steps 2-4 is performed. (At most one among $\{G$, co-$G\}$ is disconnected; moreover, if G and co-G are both connected, then the maximal modules of G are pairwise disjoint.) Observe that the graph G^0 constructed in step 5 of the algorithm is either an edgeless graph, a complete graph, or a prime graph. Therefore, the modular decomposition approach reduces the problem from a graph to its prime induced subgraphs.

The correctness of the procedure is straightforward: every independent set I of G consists of pairwise disjoint independent sets in the subgraphs of G induced by $\mathcal{M}_1, \ldots, \mathcal{M}_r$; moreover, those $\mathcal{M}_i$'s that contain a vertex from I form an independent set in G^0. And conversely, for every independent set I^0 in G^0 and every choice of independent sets $\{I_j : j \in I^0\}$ with I_j independent in $G[\mathcal{M}_j]$, the set $\cup_{j\in[r]} I_j$ is independent in G.

The following theorem answers the question on the complexity of such a reduction.

Theorem 5.3.23. *Let X be a class of graphs and X^* the class of all prime induced subgraphs of the graphs in X. If there is a $p \geq 1$ and a $q \geq 2$ such that the* EXACT WEIGHTED INDEPENDENT SET *problem can be solved for graphs in X^* in time $O(b^q n^p)$, then the* EXACT WEIGHTED INDEPENDENT SET *problem can be solved for graphs in X in time $O(b^q n^p + m)$.*

The proof of this theorem is a slight modification of the proof of Theorem 2.2.1, taking into account that:

- If G_U is disconnected, then G_U^0 is an edgeless graph, and the problem can be solved for G_U^0 in time $O(b^2|V(G_U^0)|)$, since it is a generalized subset sum problem (cf. Lemma 5.3.2).
- If G_U is disconnected, then G_U^0 is a complete graph, and the problem can be solved trivially for G_U^0 in time $O(b|V(G_U^0)|)$.

Just like for the weighted independent set problem, modular decomposition is the key to pseudo-polynomial-time solutions to the EWIS problem in several subclasses of P_5-free and *fork*-free graphs. The results are summarized in the following theorem; all graphs mentioned in the theorem or its proof are depicted in Figure 5.3.

Theorem 5.3.24. *The* EXACT WEIGHTED INDEPENDENT SET *problem is solvable in pseudo-polynomial time for each of the following classes:*

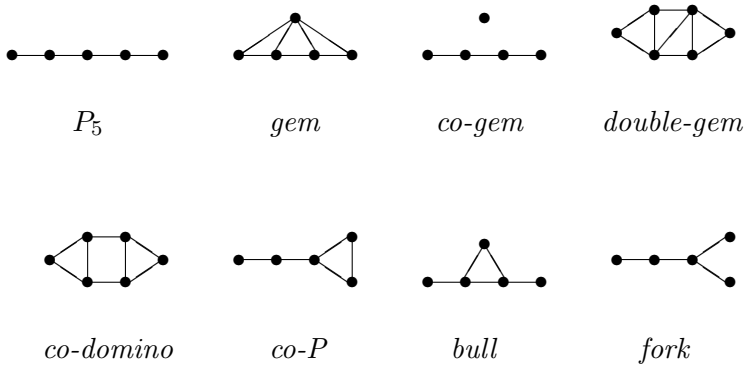

Figure 5.3: Some 5- and 6-vertex graphs

- (P_5, *double-gem, co-domino*)*-free graphs (and their subclass,* (P_5, *co-P*)*-free graphs),*
- (*bull, fork*)*-free graphs,*
- (*co-P, fork*)*-free graphs,*
- (P_5, *fork*)*-free graphs.*

Proof. This theorem essentially follows from Theorem 5.3.23 and the results in [27] and [72]. We briefly summarize the main ideas.

Every prime (P_5, *double-gem, co-domino*)-free graph is $2K_2$-free (the complementary version of this statement is proved in [72]). Since we can easily extend Theorem 5.3.4 to the extended version of EWIS, this implies the result for (P_5, *double-gem, co-domino*)-free graphs.

The (extended) EWIS problem can be solved in pseudo-polynomial time for *co-gem*-free graphs. Indeed, for every vertex v of a *co-gem*-free graph G, the non-neighborhood of v in G is P_4-free. So the problem reduces to solving $O(nb)$ subproblems in P_4-free graphs, which can be done by modular decomposition. Every P_4-free graph is either disconnected, or its complement is disconnected. Thus, the only prime P_4-free graph is the graph on a single vertex.

In [27], it is shown that prime graphs that contain a *co-gem* and are either ($bull, fork$)-free, (*co-P*, $fork$)-free or (P_5, $fork$)-free have a very simple structure. The (extended) EWIS

problem can be solved in pseudo-polynomial time for such graphs. Together with the above observation about *co-gem*-free graphs and Theorem 5.3.23, this concludes the proof. □

5.4 A Final Remark

As we saw in the introduction, motivation for studying the exact weighted independent set problem comes from the fact that the complexity of the exact perfect matching problem is still unknown, even for bipartite graphs. Hence, the problem of determining the complexity of the EWIS$_\alpha$ problem is of particular interest for line graphs of bipartite graphs, and their subclasses and superclasses. We will now show that the class $L(Bip)$ of line graphs of bipartite graphs is sandwiched between two graph classes for which the complexity of the EWIS$_\alpha$ problem is known, and whose (infinite) sets of forbidden induced subgraphs differ only in two graphs.

Recall that the line graphs of bipartite graphs are precisely the $(claw, diamond, C_5, C_7, \ldots)$-free graphs. Replacing the diamond in the above characterization by its subgraph C_3 results in a smaller class of $(claw, C_3, C_5, C_7, \ldots)$-free graphs. It is easy to see that this is precisely the class of bipartite graphs of maximum degree 2. Every connected graph in this class is either an even cycle or a path, and the treewidth of such graphs is at most 2. By Corollary 5.3.1 and Theorem 5.3.19, the problem is solvable in pseudo-polynomial time in this class.

On the other hand, if we replace the $claw = K_{1,3}$ with $K_{1,4}$ in the above characterization of $L(Bip)$, we obtain a class of graphs that properly contains line graphs of bipartite graphs. This class of $(K_{1,4}, diamond, C_5, C_7, \ldots)$-free graphs contains the class of $(K_{1,4}, C_3, C_5, C_7, \ldots)$-free graphs, which is precisely the class of bipartite graphs of maximum degree at most 3. The results of Section 5.2 imply that the problem is strongly NP-complete for this class, and hence also for the larger class of $(K_{1,4}, diamond, C_5, C_7, \ldots)$-free graphs.

To summarize, the class $L(Bip)$ of line graphs of bipartite graphs is sandwiched between two graph classes for which the complexity of the EWIS$_\alpha$ problem is known, as the following

diagram shows.

$$\begin{array}{ccccc} \mathit{Free}(\{\mathit{claw}, C_3, C_5, C_7, \ldots\}) & \subset & L(\mathit{Bip}) & \subset & \mathit{Free}(\{K_{1,4}, \mathit{diamond}, C_5, C_7, \ldots\}) \\ \text{pseudo-polynomial} & & ??? & & \text{strongly NP-complete} \end{array}$$

Chapter 6

Conclusion

In this thesis, we have presented several complexity results for the interrelated problems of finding independent sets of maximum cardinality, maximum weight, or of given weight in a graph. The common natural assumption was that the input graphs belong to a hereditary class of graphs.

Several open-ended questions and challenges are left for future research.

6.1 Open Complexity Questions

First, let us informally observe how widely open remains the gap between the polynomial and the NP-hard side of the MAXIMUM INDEPENDENT SET problem in hereditary graph classes. The areas of unknown complexity status of the problem in $\mathcal{F}$-free graphs occur already when $\mathcal{F}$ consists of a single graph on 5, 6, or 7 vertices. More specifically, there are four minimal classes defined by a single forbidden induced subgraph for which the complexity status of the MAXIMUM INDEPENDENT SET problem is unknown. These are the P_5-free graphs (P_5 is the unique minimal *connected* graph for which this question is open), $P_4 + P_2$-free graphs, $P_3 + P_3$-free graphs, and $P_3 + 2P_2$-free graphs.[1] The complexity of the problem is unknown even for (P_5, C_5)-free graphs.

For hereditary classes defined by infinitely many forbidden induced subgraphs, let us emphasize that the complexity of the MAXIMUM INDEPENDENT SET problem is unknown for $(C_k, C_{k+1}, C_{k+2}, \ldots)$-free graphs, for every $k \geq 5$, as well as for $(H_k, H_{k+1}, H_{k+2}, \ldots)$-free graphs, for every $k \geq 1$ (cf. Chapter 4).

On a more general note, recall that Alekseev's result (Theorem 1.1.1) provides sufficient

[1]This observation follows from Theorem 1.1.1 and polynomial-time solvability of the problem in the classes of P_4-free graphs [33], mK_2-free graphs [48] and $S_{1,1,1} + K_2$-free graphs [84].

conditions for the set $\mathcal{F}$ which guarantee that the MAXIMUM INDEPENDENT SET problem remains NP-hard for $\mathcal{F}$-free graphs. Interestingly, it is not known whether the converse of Alekseev's theorem holds true: to the best of our knowledge, no graph $S \in \mathcal{S}$ is known such that the MAXIMUM INDEPENDENT SET problem is NP-hard in the class of S-free graphs.[2]

Regarding the exact version of the problem, we recall that the complexity status of the EXACT WEIGHTED (MAXIMUM) INDEPENDENT SET problem remains unsettled for line graphs of bipartite graphs, and for its superclasses line graphs, claw-free graphs, and fork-free graphs.

6.2 The Augmenting Graph Method

As described in Section 2.1, the method of augmenting graphs can generally be applied only to unweighted graphs. However, this approach has proved useful in designing polynomial-time algorithms to some weighted cases as well (for instance for claw-free graphs [90, 95]). More generally, we ask: To what extent can the method of augmenting graphs be applied to *weighted graphs* (either to the *maximization*, or to the *exact* versions of the problem)?

By analogy with the work done for the weighted matching problem [43, 44], a natural question related to augmenting graphs is the following. Let X be a hereditary class of graphs where the MAXIMUM (WEIGHT) INDEPENDENT SET problem is polynomially solvable. Can the method of augmenting graphs help in designing *fast* (say, linear-time) *approximation algorithms* for the MAXIMUM (WEIGHT) INDEPENDENT SET problem for graphs in X?

6.3 Further Algorithmic Improvements

The results from Chapters 3 and 4 can be summarized in the following table:

[2] By contrast, it is easy to see that the converse of the extension to the infinite case (Theorem 1.3.1) is false. Indeed, the MAXIMUM INDEPENDENT SET problem is NP-hard in the class of $\mathcal{F}$-free planar graphs of degree at most 3, where $\mathcal{F} = \{C_{3k+1} : k = 1, 2, \ldots\}$. This follows from the NP-hardness of the MAXIMUM INDEPENDENT SET problem in planar graphs of degree at most 3: by performing the double subdivision of each edge of the input graph, we obtain a graph that can only contain induced cycles of orders that are multiples of 3.

Polynomial results for the MAXIMUM WEIGHT INDEPENDENT SET **problem in $\mathcal{F}$-free graphs**

$\mathcal{F}$	additional restriction	method(s)
$\{S_{1,1,2}\} = \{\text{fork}\}$		*MD*
$\{S_{1,2,5}$, banner$\}$	unweighted	*AG*
$\{S_{1,2,k}\}$	$K_{3,3}$-minor-free, unweighted	*AG*, *DCS*, *BT*
$\{S, L(S')\}$ with $S, S' \in \mathcal{S}$	H-minor-free, H apex	*BT*
$\{mS_{1,k,k}\}$	bounded degree	other
$\{A_k, A_{k+1}, \ldots\}$	bounded degree	*MD*, *DCS*
$\{H_k, H_{k+1}, \ldots\}$	bounded degree	other
$\{C_k, C_{k+1}, \ldots\}$	H-minor-free (H apex)	*BT*
$\{H_k, H_{k+1}, \ldots\}$	H-minor-free (H apex; 2-approx.)	other

AG = augmenting graphs, *MD* = modular decomposition, *DCS* = decomposition by clique separators, *BT* = bounded treewidth

Except for the first two results in the above table, all of them depend on a set of parameters, either integers (m, k, Δ), or graphs (S, S', H). Let K denote the set of parameters to the problem. An algorithm is said to be *fixed-parameter tractable* (FPT) if it runs in time $O(f(|K|)n^{O(1)})$ for a function f [42, 96] (rather than in time $O(n^{g(|K|)})$ for some function g). It can be easily verified that, except for the case of $mS_{1,k,k}$-free graphs of bounded degree, all of the algorithms from the above table are FPT. We leave the development of an *FPT algorithm* for the MAXIMUM WEIGHT INDEPENDENT SET problem in $mS_{1,k,k}$-free graphs of maximum vertex degree at most Δ as an open problem. We remark that it would suffice to develop an FPT algorithm for the problem of finding an induced copy of $S_{1,k,k}$ in graphs from this class.

Another area where further improvement should be possible is the *reduction of the* MAXIMUM WEIGHT INDEPENDENT SET *problem from claw-free graphs to line graphs.* (Recall from Section 3.1 that the currently fastest algorithm for the MAXIMUM WEIGHT INDEPENDENT

SET problem in line graphs is of time complexity $O(nm+n^2 \log n)$, while for claw-free graphs this time complexity is $O(n^7)$.) The resolution of this challenging research problem would result in better complexities of the MAXIMUM WEIGHT INDEPENDENT SET problem for the claw-free and fork-free graphs.

We believe that modular decomposition may be useful for such a reduction, since most of the forbidden graphs characterizing the class of line graphs are not prime (see e.g. [68] for the complete list of minimal non-line graphs). Moreover, many of them contain a clique separator. This suggests the idea of combining the modular decomposition technique with finding clique separators. If a reduction based on this approach is possible for claw-free graphs, it would probably improve the time complexity greatly.

References

[1] V.E. Alekseev, The effect of local constraints on the complexity of determination of the graph independence number. *Combinatorial-algebraic methods in applied mathematics*, Gorkiy University Press, Gorky (1982) 3-13 (in Russian).

[2] V.E. Alekseev, On the number of maximal independent sets in graphs from hereditary classes, *Combinatorial-algebraic methods in discrete optimization*, University of Nizhny Novgorod (1991) 5–8 (in Russian).

[3] V.E. Alekseev, Polynomial algorithm for finding the largest independent sets in graphs without forks, *Discrete Appl. Math.* 135 (2004) 3–16.

[4] V.E. Alekseev, On easy and hard hereditary classes of graphs with respect to the independent set problem. Stability in graphs and related topics, *Discrete Appl. Math.* 132 (2003) 17–26.

[5] V.E. Alekseev and V.V. Lozin, Augmenting graphs for independent sets, *Dicsrete Appl. Math.* 145 (2004) 3–10.

[6] G. Alexe, P.L. Hammer, V.V. Lozin and D. de Werra, Struction revisited, *Discrete Appl. Math.* 132 (2003) 27–46.

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

[8] E. Balas and C.S. Yu, On graphs with polynomially solvable maximum-weight clique problem, *Networks* 19 (1989) 247-253.

[9] H.-J. Bandelt and H.M. Mulder, Distance-hereditary graphs, *J. Combin. Theory Ser. B* 41 (1986) 182–208.

[10] F. Barahona and W.R. Pulleyblank, Exact arborescences, matchings, and cycles, *Discrete Appl. Math.* 16 (1987) 91-99.

[11] C. Berge, Two theorems in graph theory, *Proc. Nat. Acad. Sci. USA* 43 (1957) 842–844.

[12] J. Błażewicz, P. Formanowicz, M. Kasprzak, P. Schuurman and G. Woeginger, A polynomial time equivalence between DNA sequencing and the exact perfect matching problem, *Discrete Optim.* (2006) doi:10.1016/j.disopt.2006.07.004

[13] N. Blum, A new approach to maximum matching in general graphs, *Lecture Notes in Computer Science* 443 (1990) 586-597.

[14] H.L. Bodlaender, A partial k-arboretum of graphs with bounded treewidth, *Theoret. Comput. Sci.* 209 (1998) 1–45.

[15] H.L. Bodlaender, A. Brandstädt, D. Kratsch, M. Rao and J. Spinrad, On algorithms for (P_5, gem)-free graphs, *Theoret. Comput. Sci.* 349 (2005) 2–21.

[16] H.L. Bodlaender and D.M. Thilikos, Treewidth for graphs with small chordality, *Discrete Appl. Math.* 79 (1997) 45–61.

[17] R. Boliac and V.V. Lozin, An augmenting graph approach to the stable set problem in P_5-free graphs, *Discrete Appl. Math.* 131 (2003) 567-575.

[18] I.M. Bomze, M. Budinich, P.M. Pardalos and M. Pelillo, The maximum clique problem. Handbook of combinatorial optimization, Supplement Vol. A, 1–74, Kluwer Acad. Publ., Dordrecht, 1999.

[19] J.A. Bondy and U.S.R. Murty (1976). Graph theory with applications, *American Elsevier Publishing Co., Inc., New York.*

[20] A. Brandstädt, (P_5, diamond)-free graphs revisited: structure and linear time optimization, *Discrete Appl. Math.* 138 (2004) 13–27.

[21] A. Brandstädt and F.F. Dragan, On linear and circular structure of (claw, net)-free graphs, *Discrete Appl. Math.* 129 (2003) 285–303.

[22] A. Brandstädt and P.L. Hammer. On the stability number of claw-free P_5-free and more general graphs. Proceedings of the Conference on Optimal Discrete Structures and Algorithms—ODSA '97 (Rostock), *Discrete Appl. Math.* 95 (1999) 163–167.

[23] A. Brandstädt, T.C. Hoàng and V.B. Le, Stability number of bull- and chair-free graphs revisited, *Discrete Appl. Math.* 131 (2003) 39–50.

[24] A. Brandstädt, V.B. Le and J. Spinrad, *Graph classes: a survey.* SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, PA, 1999.

[25] A. Brandstädt, T.C. Hoàng and J.-M. Vanherpe, On minimal prime extensions of a four-vertex graph in a prime graph, *Discrete Math.* 288 (2004) 9–17.

[26] A. Brandstädt and C. Hoàng, On Clique Separators, Nearly Chordal Graphs, and the Maximum Weight Stable Set Problem, *Proceedings IPCO 2005*, 265–275, *Lecture Notes in Comput. Sci.* 3509, Springer, Berlin, 2005.

[27] A. Brandstädt, V.B. Le and N.H. de Ridder, Efficient robust algorithms for the maximum weight stable set problem in chair-free graph classes, *Inform. Process. Lett.* 89 (2004) 165–173.

[28] A. Brandstädt, V.B. Le and S. Mahfud, New Applications of Clique Separator Decomposition for the Maximum Weight Stable Set Problem, extended abstract in: *Proceedings FCT 2005*, Lübeck, LNCS 3623 (2005) 505–516.

[29] A. Brandstädt, H.-O. Le and J.-M. Vanherpe, Structure and stability number of chair-, co-P- and gem-free graphs revisited, *Inform. Process. Lett.* 86 (2003) 161–167.

[30] A. Brandstädt and S. Mahfud, Maximum weight stable set on graphs without claw and co-claw (and similar graph classes) can be solved in linear time, *Inform. Process. Lett.* 84 (2002) 251–259.

[31] H. Broersma, T. Kloks, D. Kratsch and H. Müller, Independent sets in asteroidal triple-free graphs, *SIAM J. Discrete Math.* 12 (1999) 276–287.

[32] O. Cogis and E. Thierry, Computing maximum stable sets for distance-hereditary graphs, *Discrete Optim.* 2 (2005) 185–188.

[33] D.G. Corneil, H. Lerchs and L. Stewart-Burlingham, Complement reducible graphs, *Discrete Appl. Math.* 3 (1981) 163–174.

[34] D.G. Corneil and U. Rotics, On the relationship between clique-width and treewidth, *SIAM J. Comput.* 34 (2005) 825–847.

[35] G. Damiand, M. Habib and C. Paul, A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. Combinatorics and computer science (Palaiseau, 1997), *Theoret. Comput. Sci.* 263 (2001) 99–111.

[36] C. De Simone and A. Sassano, Stability number of bull- and chair-free graphs, *Discrete Appl. Math.* 41 (1993) 121–129.

[37] V.G. Deĭneko and G.J. Woeginger, On the robust assignment problem under a fixed number of cost scenarios, *Oper. Res. Lett.* 34 (2006) 175–179.

[38] E. D. Demaine and M. T. Hajiaghayi, Diameter and treewidth in minor-closed graph families, revisited, *Algorithmica* 40 (2004) no. 3, 211–215.

[39] E.D. Demaine, M.T. Hajiaghayi and K.-i. Kawarabayashi, Algorithmic Graph Minor Theory: Decomposition, Approximation, and Coloring, *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS) 2005, Pittsburgh, PA, October 23-25, 2005*, 637–646.

[40] T. Denley, The independence number of graphs with large odd girth, *Electron. J. Combinatorics*, 1:Research Paper 9, 12 pp, 1994.

[41] G.A. Dirac, On rigid circuit graphs, *Abh. Math. Semin. Univ. Hamburg* 25 (1961) 71-76.

[42] R.G. Downey and M.R. Fellows, Parametrized Complexity, Monographs in Computer Science. Springer-Verlag, New York, 1999.

[43] D.E. Drake and S. Hougardy, Improved linear time approximation algorithms for weighted matchings, Approximation, randomization, and combinatorial optimization, 14–23, *Lecture Notes in Comput. Sci.* 2764, Springer, Berlin, 2003.

[44] D.E. Drake and S. Hougardy, A simple approximation algorithm for the weighted matching problem, *Inform. Process. Lett.* 85 (2003) 211–213.

[45] J. Edmonds, Path, trees, and flowers, *Canadian J. Math.* 17 (1965) 449–467.

[46] A. Ehrenfeucht and G. Rozenberg, Primitivity is hereditary for 2-structures, *Theoret. Comput. Sci.* 70 (1990) 343–358.

[47] D. Eppstein, Diameter and treewidth in minor-closed graph families, *Algorithmica* 27 (2000) 275–291.

[48] M. Farber, M. Hujter and Zs. Tuza, An upper bound on the number of cliques in a graph, *Networks* 23 (1993) 207–210.

[49] J.-L. Fouquet and V. Giakoumakis, On semi-P_4-sparse graphs, Graphs and combinatorics (Marseille, 1995). *Discrete Math.* 165/166 (1997) 277–300.

[50] A. Frank, Some polynomial algorithms for certain graphs and hypergraphs, *Proc. of the 5th Brit. Comb. Conf., Aberdeen 1975*, Congr. Numer. XV, 211–226 (1976).

[51] H.N. Gabow, Data structures for weighted matching and nearest common ancestors with linking, *Proc. SODA '90*, 1990, 434-443.

[52] H.N. Gabow and R.E. Tarjan, Faster scaling algorithms for general graph matching problems, *J. ACM* 38 (1991) 815-853.

[53] T. Gallai, Transitiv orientierbare graphen, *Acta Math. Acad. Sci. Hungar.* 18 (1967) 25–66.

[54] M.R. Garey and D.S. Johnson, The rectilinear Steiner tree problem is NP-complete, *SIAM J. Appl. Math.* 32 (1977) 826–834.

[55] M.R. Garey and D.S. Johnson (1979). Computers and intractability. A guide to the theory of NP-completeness, *CA, Freeman*.

[56] F. Gavril, Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph, *SIAM J. Comput.* 1 (1972) 180–187.

[57] M.U. Gerber, A. Hertz and V.V. Lozin, Stable sets in two subclasses of banner-free graphs, *Discrete Appl. Math.* 132 (2004) 121–136.

[58] M.U. Gerber, A. Hertz and V.V. Lozin, Augmenting chains in graphs without a skew star, *J. Combin. Theory Ser. B* 96 (2006) 352–366.

[59] V. Giakoumakis and I. Rusu, Weighted parameters in $(P_5, \overline{P}_5)$-free graphs, *Discrete Appl. Math.* 80 (1997) 255–261.

[60] M.C. Golumbic and P.L. Hammer, Stability in circular arc graphs, *J. Algorithms* 9 (1988) 314–320.

[61] M. Grötschel, L. Lovász and A. Schrijver, Polynomial algorithms for perfect graphs. Topics on perfect graphs, 325–356, North-Holland Math. Stud., 88, North-Holland, Amsterdam, 1984.

[62] F. Gurski and E. Wanke, Line graphs of bounded clique-width, *Discrete Appl. Math.*, to appear, 2007.

[63] M. Habib and M.C. Maurer, On the X-join decomposition for undirected graphs, *Discrete Appl. Math.* 1 (1979) 201–207.

[64] P. Hall, On representatives of subsets, *J. London Math. Soc.* 10 (1935) 26–30.

[65] M.M. Halldórsson, Approximating discrete collections via local improvements, *Proceedings of the Sixth SAIM-ACM Symposium on Discrete Algorithms (San Francisco, CA, 1995)*, 160–169, 1995.

[66] P.L. Hammer, N.V.R. Mahadev and D. de Werra, Stability in CAN-free graphs, *J. Combin. Theory Ser. B* 38 (1985) 23–30.

[67] P.L. Hammer, N.V.R. Mahadev and D. de Werra, The struction of a graph: application to CN-free graphs, *Combinatorica* 5 (1985) 141–147.

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

[69] J. Håstad, Clique is hard to approximate within $n^{1-\varepsilon}$, *Acta Mathematica* 182 (1999) 105–142.

[70] A. Hertz and D. de Werra, On the stability number of AH-free graphs, *J. Graph Theory* 17 (1993) 53–63.

[71] A. Hertz, V. Lozin and D. Schindl, Finding augmenting chains in extensions of claw-free graphs, *Inform. Process. Lett.* 86 (2003) 311–316.

[72] C.T. Hoáng and B. Reed, Some classes of perfectly orderable graphs, *J. Graph Theory* 13 (1989) 445-463.

[73] W.-L. Hsu and T.-H. Ma, Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs, *SIAM J. Comput.* 28 (1999) 1004-1020.

[74] M. Kamiński, V. Lozin and M. Milanič, Recent Developments on Graphs of Bounded Clique-width, submitted. (Available online as RUTCOR Research Report 6-2007, at `http://rutcor.rutgers.edu/pub/rrr/reports2007/6_2007.pdf` .)

[75] A.V. Karzanov, Maximum matching of given weight in complete and complete bipartite graphs, *Cybernetics* 23 (1987) 8-13; translation from *Kibernetika* 1 (1987) 7-11.

[76] M. Leclerc, Polynomial time algorithms for exact matching problems, Masters Thesis, University of Waterloo, Waterloo, 1986.

[77] L. Lovász and M.D. Plummer, Matching Theory. *Ann. Discrete Math*, 29. North-Holland Publishing Co., Amsterdam; Akadmiai Kiad (Publishing House of the Hungarian Academy of Sciences) Budapest, 1986.

[78] V.V. Lozin, Stability in P_5- and banner-free graphs, *European J. Oper. Research*, 125 (2000) 292–297.

[79] V.V. Lozin, Conic reduction of graphs for the stable set problem, *Discrete Math.* 222 (2000) 199–211.

[80] V.V. Lozin and M. Milanič, On finding augmenting graphs, submitted. (Available online as RUTCOR Research Report 38-2005, at `http://rutcor.rutgers.edu/pub/rrr/reports2005/38_2005.pdf` .)

[81] V.V. Lozin and M. Milanič, A polynomial algorithm to find an independent set of maximum weight in a fork-free graph, *Proceedings of the Seventeenth Annual ACM–SIAM Symposium on Discrete Algorithms (Miami, FL, 2006),* 26–30, ACM, New York, 2006.

[82] V. Lozin and M. Milanič, Maximum Independent Sets in Graphs of Low Degree, *Proceedings of the Eighteenth Annual ACM–SIAM Symposium on Discrete Algorithms (New Orleans, LA, 2007),* 874–880. ACM, New York, SIAM, Philadelphia.

[83] V. Lozin and M. Milanič, On the maximum independent set problem in subclasses of planar and more general graphs. In preparation.

[84] V.V. Lozin and R. Mosca, Independent sets in extensions of $2K_2$-free graphs, *Discrete Appl. Math.* 146 (2005) 74–80.

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

[86] C. Mannino, G. Oriolo, F. Ricci and S. Chandran. The stable set problem and the thinness of a graph, *Oper. Res. Lett.* 35 (2007) 1–9.

[87] R.M. McConnell and J. P. Spinrad, Modular decomposition and transitive orientation, *Discrete Math.* 201 (1999) 199–241.

[88] S. Micali and V.V. Vazirani, An $O(\sqrt{|V|}|e|)$ algorithm for finding maximum matching in general graphs, *Proceedings of the Twenty-First Annual IEEE Symposium on Foundations of Computer Science*, (1980) 17-27.

[89] M. Milanič and J. Monnot, On the complexity of the exact weighted independent set problem, submitted. (A preliminary version is available online as DIMACS Technical Report 2006-17, at `ftp://dimacs.rutgers.edu/pub/dimacs/TechnicalReports/TechReports/2006/2006-17.ps.gz`.)

[90] G.J. Minty, On maximal independent sets of vertices in claw-free graphs. *J. Combin. Theory Ser. B* 28 (1980) 284–304.

[91] R.H. Möhring, Algorithmic aspects of comparability graphs and interval graphs, in: I. Rival (Ed.) *Graphs and Orders*, D. Reidel, Boston, 1985, 41–101.

[92] R. Mosca, Independent sets in certain P_6-free graphs, *Discrete Appl. Math.* 92 (1999) 177–191.

[93] K. Mulmuley, U. Vazirani and V.V. Vazirani, Matching is as easy as matrix inversion, *Combinatorica* 7 (1987) 105-113.

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

[95] D. Nakamura and A. Tamura, A revision of Minty's algorithm for finding a maximum weight stable set of a claw-free graph, *J. Oper. Res. Soc. Japan* 44 (2001) 194–204.

[96] R. Niedermaier, Invitation to fixed-parameter algorithms. Oxford Lecture Series in Mathematics and its Applications, 31. Oxford University Press, Oxford, 2006.

[97] Y. Okamoto, T. Uno and R. Uehara, Linear-time counting algorithms for independent sets in chordal graphs, Graph-theoretic concepts in computer science, 433–444, *Lecture Notes in Comput. Sci.* 3787, Springer, Berlin, 2005.

[98] S.-i. Oum, Approximating rank-width and clique-width quickly, Graph-theoretic concepts in computer science, 49–58, *Lecture Notes in Comput. Sci.* 3787, Springer, Berlin, 2005.

[99] C.H. Papadimitriou and M. Yannakakis, The complexity of restricted spanning tree problems, *J. ACM* 29 (1982) 285-309.

[100] S. Poljak, A note on stable sets and coloring of graphs, *Comment. Math. Univ. Carolinae* 15 (1974) 307–309.

[101] E. Prisner, Graphs with few cliques, Graph Theory, Combinatorics, and Algorithms, Vol. 1, 2 (Kalamazoo, MI, 1992), 945-956, Wiley-Interscience Publ., Wiley, New York, 1995.

[102] J. Ramalingam and C. Pandu Rangan, A unified approach to domination problems on interval graphs, *Inform. Process. Lett.* 27 (1988) 271-274.

[103] N. Robertson and P.D. Seymour, Graph minors. I. Excluding a forest, *J. Combin. Theory Ser. B* 35 (1983) 39–61.

[104] N. Robertson and P.D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, *J. Algorithms* 7 (1986) 309–322.

[105] N. Sbihi, Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile, *Discrete Math.* 29 (1980) 53–76.

[106] W. Staton and G.C. Wingard, On line graphs of bipartite graphs, *Util. Math.* 53 (1998) 183–187.

[107] K.J. Supowit, Finding a maximum planar subset of a set of nets in a channel, *IEEE Trans. on CAD of ICAS*, CAD-6 (1987) 93–94.

[108] R.E. Tarjan, Decomposition by clique separators, *Discrete Math.* 55 (1985) 221–232.

[109] S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, A new algorithm for generating all the maximal independent sets, *SIAM J. Comput.* 6 (1977) 505-517.

[110] V.V. Vazirani, A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph maximum matching algorithm, *Combinatorica* 14 (1994) 71–109.

[111] V. Vazirani (2001). Approximation Algorithms, *Springer Verlag, Berlin, Heidelberg, New York*.

[112] K. Wagner, Über eine Eigenschaft der ebenen Komplexe, *Math. Ann.* 114 (1937) 570-590.

[113] S.H. Whitesides, An algorithm for finding clique cut-sets, *Inform. Process. Lett.* 12 (1981) 31–32.

Vita

Martin Milanič

2003-2007 Rutgers University, New Brunswick, NJ, USA

Ph.D., Operations Research, May 2007

M.S., Operations Research, January 2006

1998-2003 University of Ljubljana, Faculty of Mathematics and Physics, Department of Mathematics, Ljubljana, Slovenia

B.S., Mathematics, July 2003

Conference Publications

V. V. Lozin and M. Milanič, A polynomial algorithm to find an independent set of maximum weight in a fork-free graph, Proceedings of the Seventeenth Annual ACM–SIAM Symposium on Discrete Algorithms. Miami, FL, 22–24 January, 2006, ACM, New York, 2006, pp 26–30.

V. Lozin and M. Milanič, Maximum Independent Sets in Graphs of Low Degree, Proceedings of the Eighteenth Annual ACM–SIAM Symposium on Discrete Algorithms. New Orleans, LA, 7–9 January, 2007. ACM, New York, SIAM, Philadelphia, pp 874–880.