Algorithms for Graphs of (Locally) Bounded Treewidth

by

MohammadTaghi Hajiaghayi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, September 2001

Abstract

Many real-life problems can be modeled by graph-theoretic problems. These graph problems are usually NP-hard and hence there is no efficient algorithm for solving them, unless P= NP. One way to overcome this hardness is to solve the problems when restricted to special graphs. Trees are one kind of graph for which several NP-complete problems can be solved in polynomial time. Graphs of bounded treewidth, which generalize trees, show good algorithmic properties similar to those of trees. Using ideas developed for tree algorithms, Arnborg and Proskurowski introduced a general dynamic programming approach which solves many problems such as dominating set, vertex cover and independent set. Others used this approach to solve other NP-hard problems. Matoušek and Thomas applied this approach to solve the subgraph isomorphism problem when the source graph has bounded degree and the host graph has bounded treewidth. In this thesis, we introduce a new property for graphs called *log-bounded fragmentation*, by which we mean after removing any set of at most k vertices the number of connected components is at most $O(k \log n)$, where n is the number of vertices of the graph. We then extend the result of Matoušek and Thomas to the case in which the source graph is a *log*-bounded fragmentation graph and the host graph has bounded treewidth. Besides this result, we demonstrate how bounded fragmentation might be used to measure the reliability of a network.

As the class of graphs of bounded treewidth is of limited size, we need to solve NP-hard problems for wider classes of graphs than this class. Eppstein introduced a new concept which can be considered as a generalization of bounded treewidth. A graph G has *locally bounded treewidth* if for each vertex v of G, the treewidth of the subgraph of G induced on all vertices of distance at most r from v is only a function of r, called *local treewidth.* So far the only graphs determined to have small local treewidth are planar graphs. In this thesis, we prove that the local treewidth of $K_{3,3}$-minor-free or K_5-minor-free graphs is also bounded above by $3r + 4$. Using this result, we extend several polynomial-time approximation algorithms on planar graphs to these graphs. Algorithms on graphs of bounded treewidth also can be extended to graphs of locally bounded treewidth. As an example, we demonstrate how the subgraph isomorphism problem on graphs of locally bounded treewidth can be solved in polynomial time, when the source graph is a *log*-bounded fragmentation graph and has constant diameter.

Key Words. Treewidth, Local treewidth, Minor, Approximation algorithms, Subgraph isomorphism.

Acknowledgements

First, I would like to thank my supervisor, Prof. Naomi Nishimura, for being an enthusiastic and genius supervisor. She introduced me to the concept of treewidth, the basis of this thesis, and provided me a good source of inspiration. She also helped with her useful suggestions on the presentation of the results. This research would have been impossible without her helps and encouragements.

Many thanks to Prof. Dimitrios M. Thilikos, Prof. Prabhakar Ragde and Douglas B. West for their guidance to some new research areas. Thanks go to Prof. Jim Geelen and Prof. Ian Munro, the readers of this thesis, for their thoughtful comments. Also, I would like to thank the staff and faculty of the Department of Computer Science at the University of Waterloo for providing such a nice academic environment.

My friends here at University of Waterloo, namely Saeed Behzadipoor, Mahmood Saadat, Afshin Mategh, Vahid Garousi, Yashar Ganjali, Ali ghodsi, Kamran Sartipi, Zarrin Langari and Amir Chinaei, played a big role in making my life more enjoyable. Especially, I am grateful to Mohammad Mahdian for his encouragement on completing my thesis fast. Also, I would like to extend my appreciation to thank my previous professors in Sharif University of Technology, namely Prof. Ebad Mahmoodian and Prof. Mohammad Ghodsi, and my many other friends in Iran for their warmest friendship. I am blessed to have such excellent companies.

Last but not least, my dear parents and my family receive my heartfelt gratitude for their sweetest support and never-ending love. I always feel the warmth of their love, even now that we are so far away. I wish to thank my brother, Mahdi, and my sisters, Monireh and Mehri, for being the greatest source of love. This thesis is dedicated to all of them.

Contents

List of Figures

Chapter 1

Introduction

Graph-theoretic modeling is one way to solve many real-life problems. For example, we consider this problem from the book of Downey and Fellows on a new type of complexity called fixed parameter complexity [DF99] (Section 2.6). Suppose we gather a set of observations from experiments on a data set such that some pairs of observations are in conflict. We need to find a minimum set of observations such that their removal eliminates all inconsistencies. This problem can be modeled by the famous vertex cover problem in graph theory. There are many other problems which can be modeled similarly.

Unfortunately, it is not known and is not believed that most graph-theoretic problems modeled from real life have efficient algorithms. In other words, these problems are usually NP-hard on general graphs. On the other hand, solving these problems is often essential. Finding algorithms for these problems on special graphs such as trees or planar graphs is one way to attack to these problems. Finding efficient approximation algorithms is another way to cope with the hardness of these problems. We discuss this approach later in this chapter.

Many graph-theoretic problems on trees can be solved in polynomial time, especially using a dynamic programming approach. Many NP-hard problems such as minimum vertex cover, maximum independent set and minimum dominating set have efficient algorithms when restricted to trees [Har69, CGH75]. Nevertheless, for many applications, the underlying graphs are not necessarily trees. Therefore, we consider graphs of *treewidth at most k* which generalize trees. A graph G has *treewidth at most k* if one can construct a tree T

in which each node has an associated subset of at most $k+1$ vertices of the graph, called a *bag*, such that each vertex of the graph appears in at least one bag, end-vertices of each edge appear in at least one common bag, and for each vertex v of G, the vertices of T whose bags contain v form a connected subtree of T. Intuitively, the treewidth of a graph is the measure of its resemblance to a tree, e.g. the treewidth of trees is one. Arnborg and Proskurowski were the first people who applied tree algorithms to solve NP-hard problems on graphs of treewidth at most k. Others extended this approach to many other NP-hard problems such as colorability and Hamiltonicity (Chapter 2). It also appears that many graph problems have practical instances in which the input graphs have small treewidth [Bod98]. Using these facts, progress has been made in development of efficient algorithms.

One NP-hard problem that has been considered for graphs of bounded treewidth is the subgraph isomorphism problem, in which we search for a subgraph of the host graph H *isomorphic* to the source (or pattern) graph G. Here, a graph is *isomorphic* to another graph if there is a one-to-one correspondence between vertices of the two graphs that preserves adjacency. So far this problem has been solved when the source graph is bounded degree and the host graph has bounded treewidth or when both the source graph and the host graph are k-connected and have treewidth at most k [MT92, GN94]. In this thesis we introduce a new concept called *log-bounded fragmentation*. A graph is a *log*-bounded fragmentation graph if removing any set of at most k vertices generates at most $O(k \log n)$ connected components, where n is the number of vertices of the graph. We extend the bounded degree result to the case in which the source graph is a *log*-bounded fragmentation graph and the host graph has bounded treewidth. We also show the class of *log*-bounded fragmentation graphs not only contain the class of bounded degree graphs but also contain others such as the class of Hamiltonian graphs. Introducing the concept of bounded fragmentation as a measure of network reliability is another contribution of the thesis.

The design of practical algorithms on graphs similar to trees is extended to other graphs. An *outerplanar graph* is a planar graph all of whose vertices lie on the outer face. Baker [Bak94], using a decomposition of a planar graph into outerplanar graphs, found efficient approximation algorithms for planar graphs (Section 2.7). A graph has *locally bounded treewidth* if the treewidth of the subgraph induced on all vertices at distance r

from v, for any vertex v of the graph and any $r \in \mathbb{N}$, is bounded by a function $\mathrm{ltw}^G(r)$. Here the function $\mathrm{ltw}^G(r)$, the *local treewidth*, is dependent only on r. Eppstein [Epp99] characterized graphs of locally bounded treewidth. He also proved that Baker's results can be extended to graphs of locally bounded treewidth. In fact, a planar graph G has locally bounded treewidth with $\mathrm{ltw}^G(r) = 3r - 1$ [Bod98]. Eppstein [Epp99] also extended Baker's approach to other problems on graphs of locally bounded treewidth such as the subgraph isomorphism problem for a fixed pattern G. Since, except for planar graphs, the known local treewidth for graphs of locally bounded treewidth is immense, Eppstein's polynomial-time approximation algorithms can not be used for any practical purpose.

We introduce a new class of graphs which have linear local treewidth. A graph G is *H-minor-free* if H can not be obtained from any subgraph of G by contracting edges. A graph is called a *single-crossing graph* if it can be drawn on the plane with at most one crossing. We prove for a single-crossing graph H, the local treewidth of any H-minor-free graph G is bounded by $3r + c_H$ where c_H is a constant dependent on H. We note that planar graphs are both $K_{3,3}$-minor-free and K_5-minor-free, where $K_{3,3}$ and K_5 are both single-crossing graphs. Thus our result is a generalization of the result of Alber et al. [ABFN00] on linear local treewidth of planar graphs. As a consequence, we extend the practical approximation algorithms on planar graphs to this kind of graph. Using our result on the subgraph isomorphism problem, we also extend Eppstein's result to the case in which the pattern graph G is not necessarily fixed.

This thesis is organized as follows. We start with relevant background in Chapter 2. In this chapter, we introduce the terminology used throughout the thesis, and formally define tree decompositions, treewidth, and locally bounded treewidth. Furthermore, we present an overview of the general dynamic programming approach introduced by Arnborg and Proskurowski for solving problems on graphs of bounded treewidth. We also introduce the concepts of fixed parameter complexity and monadic second-order logic and demonstrate how these concepts are related to the thesis. Finally, we survey previous results on subgraph isomorphism and approximation algorithms for graphs of locally bounded treewidth.

Chapter 3 is concerned with approximation algorithms on H-minor-free graphs for single-crossing graphs H. First, we prove these graphs have linear local treewidth. Then we show how Baker's approach on planar graphs can be applied to obtain approximation algorithms for these graphs especially for $K_{3,3}$-minor-free or K_5-minor-free graphs. We also present algorithms for problems on these graphs such as minimum dominating set.

Chapter 4 is devoted to bounded fragmentation. In this chapter, we explain how this property might have applications in network reliability and introduce several classes of bounded fragmentation graphs. Furthermore, we discuss the number of edges of a bounded fragmentation graph as an important issue.

In Chapter 5, we precisely state the bounded degree result of Matoušek and Thomas for the subgraph isomorphism problem and demonstrate how this result can be generalized to graphs with the log-bounded fragmentation property. We present an example that shows how the class of bounded fragmentation graphs contains graphs with maximum degree $n-1$ where n is the number of vertices. We also generalize testing subgraph isomorphism to graphs of locally bounded treewidth.

Finally in Chapter 6, we conclude with a list of open problems and potential extensions for future work.

Chapter 2

Background

Since most areas relevant to the thesis are very broad, we cover a small part of applications of each in this chapter. The reader is referred to the references in each section to obtain more knowledge. In each section, we also introduce results which are improved in this thesis.

After mentioning notation and some basic preliminaries in Section 2.1, in Section 2.2 we introduce the concepts of the representation of a graph as a tree (tree decomposition) and of treewidth, which form the basis of our algorithms. We mainly focus on locally bounded treewidth, which is an extension of bounded treewidth. Background related to this concept is presented in Section 2.3. We present the general dynamic programming approach, which is used to solve several NP-hard problems on graphs of bounded treewidth, in Section 2.4. One of our contributions in this thesis is the improvement of the running times of the algorithms for solving several problems which can be described in logic. Related background is given in Section 2.5. Section 2.6 is devoted to fixed parameter tractability, which is a new way to handle computational intractability. A survey of previous work on approximation algorithms for solving NP-hard problems on graphs of locally bounded treewidth and algorithms for solving subgraph isomorphism is presented in Sections 2.7 and 2.8.

2.1 Preliminaries

We assume the reader is familiar with general concepts of graph theory such as directed graphs, trees and planar graphs. The reader is referred to standard references for appropriate background [BM76].

Our graph terminology is as follows. All graphs are finite, simple and undirected, unless indicated otherwise. A graph G is represented by $G = (V, E)$, where V (or $V(G)$) is the set of vertices and E (or $E(G)$) is the set of edges. We denote an edge e in a graph G between u and v by $\{u, v\}$ and a directed edge e in a directed graph G from u to v by (u, v). Here, vertices u and v are called the *end-vertices* of e. We define n to be the number of vertices of a graph when it is clear from context. The maximum degree of G is denoted by $\varDelta(G)$ and the minimum degree of G is denoted by $\delta(G)$. We define the *r-neighborhood* of a set $S \subseteq V(G)$, denoted by $N^r_G(S)$, to be the set of vertices at distance at most r from at least one vertex of $S \subseteq V(G)$; if $S = \{v\}$ we simply use the notation $N^r_G(v)$. The *diameter* of G, denoted by *diam*(G), is the maximum over all distances between pairs of vertices of G. Two disjoint sets S and S' of vertices of undirected (directed) graph G are *adjacent* if and only if there are $u \in S$ and $v \in S'$ such that $\{u, v\} \in E(G)$ ($(u, v) \in E(G)$ or $(v, u) \in E(G)$). The *union* of two disjoint graphs G_1 and G_2, $G_1 \cup G_2$, is a graph G such that $V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$. An *$n$-clique ($K_n$)* is a graph G with n vertices in which every pair of vertices is connected by an edge. A graph G is represented by $K_{n,m}$ if its vertices can be partitioned into sets V_1 and V_2 such that $|V_1| = n$, $|V_2| = m$ and edge $\{u, v\} \in E(G)$ if and only if $u \in V_1$ and $v \in V_2$ or vice versa.

For generalizations of algorithms on undirected graphs to directed graphs, we consider underlying graphs of directed graphs. An *underlying graph* of a directed graph $H = (V, E)$ is an undirected graph $G = (V, E)$ in which $V(G) = V(H)$ and $\{u, v\} \in E(G)$ if and only if $(u, v) \in E(H)$ or $(v, u) \in E(H)$.

A graph $G' = (V', E')$ is a *subgraph of* G if $V' \subseteq V$ and $E' \subseteq E$. A graph $G' = (V', E')$ is an *induced subgraph of* G, denoted by $G[V']$, if $V' \subseteq V$ and E' contains all edges of E which have both end vertices in V'. G is a *supergraph* of G' if G' is a subgraph (not necessarily induced subgraph) of G.

Chapter 5 of this thesis is devoted to solving the subgraph isomorphism problem. An *isomorphism* ϕ from (directed) graph G into (directed) graph H is a one-to-one mapping between vertices of G and H such that for each pair $u, v \in V(G)$, $\{u, v\} \in E(G)$ ($(u, v) \in E(G)$) if and only if $\{\phi(u), \phi(v)\} \in E(H)$ ($(\phi(u), \phi(v)) \in E(H)$). For a set $S \subseteq V(G)$, we define $\phi(S) = \bigcup_{v \in S} \phi(v)$. A (directed) graph G is *isomorphic* to a (directed) graph H if and only if there is an isomorphism ϕ from G into H such that $\phi(G) = V(H)$. A graph G is *subgraph isomorphic* to H if there is a subgraph H' of H which is isomorphic to G. A graph G is *induced subgraph isomorphic* to H if there exists an induced subgraph G' of H isomorphic to G.

One way of describing classes of graphs is by using *minors*, introduced below.

Definition 2.1 Contracting *an edge $e = \{u, v\}$ is the operation of replacing both u and v by a single vertex w whose neighbors are all vertices that were neighbors of u or v, except u and v themselves. A graph G is a* minor *of a graph H if H can be obtained from a subgraph of G by contracting edges. A graph class $\mathcal{C}$ is a* minor-closed *class if any minor of any graph in $\mathcal{C}$ is also a member of $\mathcal{C}$. A minor-closed graph class $\mathcal{C}$ is* H-minor-free *if $H \notin \mathcal{C}$. The* minor containment *problem determines whether a graph is a minor of another graph.*

For example, a planar graph is a graph excluding both $K_{3,3}$ and K_5 as minors.

The set of components of a graph G is represented by $\mathcal{C}(G)$, where each element of $\mathcal{C}(G)$ is a connected graph. For a set $\mathcal{D} \subseteq \mathcal{C}(G)$, we denote the set of vertices which appear in a component of $\mathcal{D}$ by $V(\mathcal{D})$ and the set of edges by $E(\mathcal{D})$. The graph resulting from removal of a set S of vertices and all adjacent edges from G is denoted by $G[V - S]$. A set S is called a *separator* if $|\mathcal{C}(G[V - S])| > 1$. For $k > 0$, graph G is called *k-connected* if every separator has size at least k.

Baker [Bak94] introduced a property of planar graphs useful for designing approximation algorithms, namely a decomposition into *outerplanar graphs* (see Definition 2.2). We consider Baker's approach in more detail in Section 2.7.

Definition 2.2 *[Bak94] Suppose graph G is embedded on the plane without any crossing. A vertex in the embedding is called a* level 1 *vertex if it is on the outer face. If an embedding obtained by removing all vertices in levels 1 to i is denoted by G^i, then the vertices on the outer face of G^i are the* level $i + 1$ *vertices. A crossing-free embedding of a graph G is r-*outerplanar *if it has no vertices of level greater than r. A graph G is called r-outerplanar if it admits an r-outerplanar embedding. The smallest number such that G is r-outerplanar is called the* outerplanarity *number. The terms* outerplanar *and* 1-outerplanar *are equivalent.*

In Chapter 3, we design *approximation algorithms* for several *NP-optimization problems.* Definition 2.3 presents exact descriptions of these terms.

Definition 2.3 *[GJ79] An* NP-optimization problem *is a tuple* $(\mathcal{I}, S, f, opt)$ *such that:*

1. $\mathcal{I}$ *is the set of input instances. We assume that* $\mathcal{I}$ *can be recognized in polynomial time;*

2. $S(x)$ *is a set of feasible solutions associated to each input instance* $x \in \mathcal{I}$*. We assume that each element in* $S(x)$ *has size polynomially bounded in the size of* x*;*

3. f *is an objective function which maps to real numbers each pair* (x, y) *with* $x \in \mathcal{I}$ *and* $y \in S(x)$*. We assume that this function is computable in polynomial time; and*

4. *opt is a goal which belongs to set* $\{min, max\}$*.*

Given an $x \in \mathcal{I}$*, we want to find a* $y \in S(x)$ *such that* $f(x, y) = opt\{f(x, z) | z \in S(x)\}$*. Let* $x \in \mathcal{I}$ *and* $\epsilon > 0$*. A solution* $y \in S(x)$ *for* x *is* ϵ-close *if*

$$(1 - \epsilon)opt(x) \leq f(x, y) \leq (1 + \epsilon)opt(x).$$

A *polynomial time approximation scheme (PTAS)* for $(\mathcal{I}, S, f, opt)$ is a *uniform family* $(A_\epsilon)_{\epsilon \geq 0}$ of approximation algorithms, where A_ϵ is a polynomial time algorithm that, given an $x \in \mathcal{I}$, computes an ϵ-close solution for x in polynomial time. In the above definition, uniformity means that there is an algorithm that, given ϵ, computes A_ϵ. We further state that an optimization problem has a *fully polynomial time approximation scheme* if there exists a PTAS A whose running time is bounded by a polynomial in $|x|$ and $1/\epsilon$.

Among NP-optimization problems, we mainly focus on those problems which are also *hereditary* (see Definition 2.4). In fact, Yannakakis [Yan78] has shown that many natural hereditary problems are NP-complete even when the graphs under consideration are planar graphs.

Definition 2.4 *[Yan78] Property* π *on graphs is called* hereditary *if, whenever* π *holds for* G*,* π *holds for all induced subgraphs of* G*. For hereditary property* π*, the* maximum induced subgraph problem *associated with* π *(MISP(*π*)) is the problem of finding a maximum subset* U *of vertices of a graph* G *such that* $G[U]$ *has property* π*. In the* weighted *case*

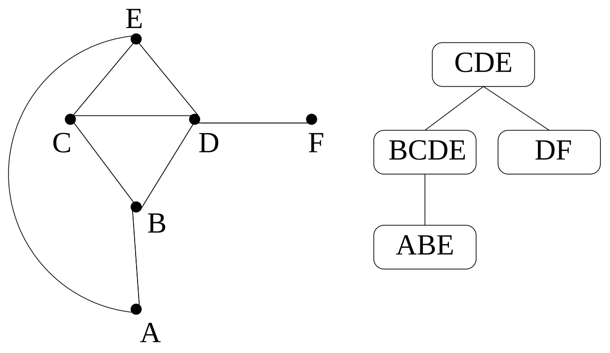

Figure 2.1: A graph and one of its tree decompositions.

(WMISP(π)), each vertex in the given graph $G = (V, E)$ has a nonnegative weight and the problem is to find a maximum weight subset U of V such that $G[U]$ has property π, where the weight of U is the total weight of the vertices in it. We call a problem P associated with the hereditary property π a hereditary maximization problem*.*

Examples of hereditary maximization problems are those in which we search for an induced subgraph of maximum size that is *chordal*, *acyclic*, *without cycles of specified length*, *without edges*, *bounded degree with maximum degree* $r \geq 1$, *bipartite* or *forms a clique* [Yan78].

For exact definitions of various NP-hard problems in this thesis, the reader is referred to Garey and Johnson's book on computers and intractability [GJ79].

2.2 Treewidth

Graphs of bounded treewidth, described in this section, are known for their good algorithmic properties. Many problems which are intractable in the general case can be solved in polynomial time or even linear time on graphs of bounded treewidth [ALS88].

The notion of treewidth was introduced by Robertson and Seymour [RS86] and plays an important role in their fundamental work on graph minors. To define this notion, first we consider the representation of a graph as a tree, which is the basis of our algorithms.

Definition 2.5 *[RS86] A* tree decomposition *of a graph $G = (V, E)$, denoted by $TD(G)$, is a pair (χ, T) in which $T = (I, F)$ is a tree and $\chi = \{\chi_i | i \in I\}$ is a family of subsets of $V(G)$ such that:*

1. *$\bigcup_{i \in I} \chi_i = V$;*
2. *for each edge $e = \{u, v\} \in E$ there exists an $i \in I$ such that both u and v belong to χ_i; and*
3. *for all $v \in V$, the set of nodes $\{i \in I | v \in \chi_i\}$ forms a connected subtree of T.*

To distinguish between vertices of the original graph G and vertices of T in $TD(G)$, we call vertices of T *nodes* and their corresponding χ_i's *bags*. The maximum size of a bag in $TD(G)$ minus one is called the *width* of the tree decomposition. The *treewidth* of a graph G ($tw(G)$) is the minimum width over all possible tree decompositions of G. The reader is referred to Figure 2.1 to see a graph G and a tree decomposition of width 3 for G.

A graph G is called a *k-tree* [Ros74] if either G is a k-clique or G has a vertex u of degree k such that u is adjacent to a k-clique, and the graph obtained by deleting u and all its incident edges is again a k-tree. A graph G is a *partial k-tree* if it is a subgraph of a k-tree.

Lemma 2.1 *(van Leeuwen [Lee90]) G is a partial k-tree if and only if G has treewidth at most k.*

Many graph properties were studied independently for some time, after which it was shown that they were equivalent to treewidth. Bodlaender introduces several equivalent properties of this kind in his paper [Bod98]. There are other related properties such as pathwidth, bandwidth, cutwidth and branchwidth. More discussion of these concepts is presented in Bodlaender's paper [Bod98].

It is interesting to know what kinds of graphs have bounded treewidth. Robertson and Seymour characterized graphs of bounded treewidth.

Theorem 2.1 *[RS86] For every fixed k, the set $\{G | G$ is a graph with treewidth at most $k\}$ can be characterized by finite sets of forbidden minors.* □

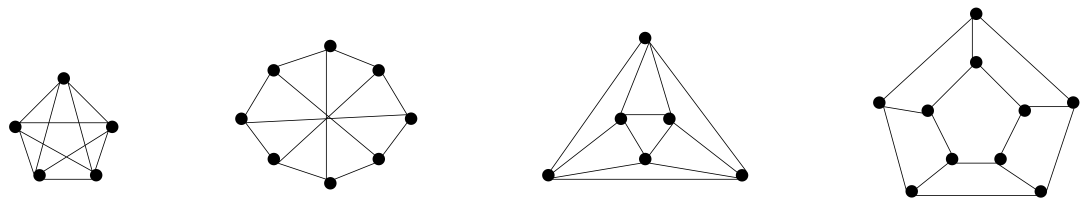

Figure 2.2: The forbidden minors for graphs of treewidth at most three.

A number of characterizations of Theorem 2.1 for small treewidth are as follows:

Theorem 2.2 *[APC90]*

1. *A graph G has treewidth at most 1 if and only if G does not contain K_3 as a minor.*
2. *A graph G has treewidth at most 2 if and only if G does not contain K_4 as a minor.*
3. *A graph G has treewidth at most 3 if and only if it does not contain any of the four graphs shown in Figure 2.2 as a minor.*

$\square$

By Theorem 2.2, forests are the only graphs of treewidth at most 1.

Theorem 2.3 *[Bod98] A graph G has treewidth at most 2 if and only if every biconnected component of G is a series-parallel graph.* $\square$

We often use the following property of tree decompositions, especially for designing polynomial-time algorithms on graphs of bounded treewidth.

Lemma 2.2 *[Bod98] Let $T_1, T_2, \cdots, T_p$ be subtrees of a tree decomposition of G formed by removing a node z from the tree decomposition and let V_i, $1 \leq i \leq p$, be the sets of vertices of G appear in bags of nodes of T_i except those appear in χ_z. The set χ_z is a separator for G. More precisely, after removal of χ_z from G, there is no edge between V_i and V_j, $i \neq j$.*

The reader is referred to Bodlaender's paper for further properties and classes of graphs of bounded treewidth.

An important related problem is determining, when given a graph G and an integer k, whether the treewidth of G is at most k. This problem is NP-complete [ACP87] even for graphs of maximum degree at most nine, bipartite graphs and cocomparability graphs [BT97]. This problem has been solved for several classes of graphs such as chordal graphs, permutation graphs [BKK95], circular arc graphs [SSR94], circle graphs [Klo93] and distance hereditary graphs [BDK00]. Bodlaender et al. [BGHK95] gave an approximation algorithm with performance ratio $O(\log n)$ for this problem on general graphs. Solving the problem for the case in which the parameter k is fixed is also interesting. The first polynomial-time algorithm for this problem was presented by Arnborg, Corneil and Proskurowski [ACP87]. The running time of this algorithm is $O(n^{k+2})$. Using fundamental results on graph minors, Robertson and Seymour gave a non-constructive proof of the existence of a decision algorithm with running time $O(n^2)$. An algorithm with a faster running time was developed by Lagergren [Lag96] and Bodlaender and Kloks [BK96]. Finally, Bodlaender [Bod96] found a constructive linear-time algorithm for the problem. All these algorithms have a hidden constant factor that is at least exponential in k, and hence they are impractical in general. In the cases $k = 2, 3, 4$, practical linear-time algorithms exist [AP86, MT92, San96]. Alber et al. [ABFN00] presented a constructive efficient algorithm for finding a tree decomposition of an r-outerplanar graph G in time $O(r|V(G)|)$ (the treewidth of an r-outerplanar graph is bounded by $3r-1$ [ABFN00]). Using this algorithm, we will design an efficient linear-time algorithm for constructing tree decompositions of $K_{3,3}$-minor-free or K_5-minor-free bounded diameter graphs (see Chapter 3).

2.3 Locally bounded treewidth

As mentioned in Section 2.2, many NP-complete problems can be solved in polynomial time when restricted to graphs of bounded treewidth. The class of graphs of bounded treewidth is of limited size; we would like to solve NP-complete problems for wider classes of graphs.

Baker [Bak94] developed several approximation algorithms to solve NP-complete problems for planar graphs. One of the bases of her work was the fact that the treewidth of a planar graph G is bounded by $O(diam(G))$. We discuss this approach in more detail in Section 2.7. To extend these algorithms to other graph families, Eppstein [Epp00] introduced the notion of bounded local treewidth, defined formally below, which is a generalization of the notion of treewidth. Intuitively, a graph has bounded local treewidth (or locally bounded treewidth) if the treewidth of an r-neighborhood of each vertex $v \in V(G)$ is a function of r, $r \in \mathbb{N}$, and not $|V(G)|$.

Definition 2.6 *The* local treewidth *of a graph G is the function* $\mathrm{ltw}^G : \mathbb{N} \to \mathbb{N}$ *that associates with every $r \in \mathbb{N}$ the maximum treewidth of an r-neighborhood in G. We set* $\mathrm{ltw}^G(r) = \max_{v \in V(G)}\{\mathrm{tw}(G[N_G^r(v)])\}$, *and we say that a graph class $\mathcal{C}$ has* bounded local treewidth (or locally bounded treewidth) *when there is a function $f : \mathbb{N} \to \mathbb{N}$ such that for all $G \in \mathcal{C}$ and $r \in \mathbb{N}$,* $\mathrm{ltw}^G(r) \leq f(r)$. *A class $\mathcal{C}$ has* linear local treewidth *if there is a constant $c \in \mathbb{R}$ such that* $\mathrm{ltw}^G(r) \leq cr$ *for all $G \in \mathcal{C}$, $r \in \mathbb{N}$.*

As mentioned in Section 2.2, Robertson and Seymour characterized the graphs of bounded treewidth by minor-closed families of graphs. Eppstein [Epp00] extended this characterization to graphs of locally bounded treewidth.

A graph is called an *apex* graph if the deletion of a vertex produces a planar graph. Intuitively, apex graphs are nearly planar graphs. The most important aspect of apex graphs is that they are examples of graphs without locally bounded treewidth. An $n \times n$ *planar grid* is a graph consisting of vertices (i, j), $1 \leq i, j \leq n$, such that vertex (i, j) is adjacent to vertices $(i, j-1)$, $(i, j+1)$, $(i-1, j)$ and $(i+1, j)$ (if they exist). A graph G constructed from an $n \times n$ planar grid by connecting a new vertex v to all of the vertices in the grid is an example of an apex graph. Here, the treewidth of $G[N_G^1(v)] = G$ is $n+1$ [Epp00] which is dependent on the number of vertices, *i.e.* n^2+1. This example demonstrates a family of apex graphs that do not have locally bounded treewidth. Using a complicated proof, Eppstein generalized this result to all graphs of locally bounded treewidth.

Theorem 2.4 *[Epp00] Let $\mathcal{F}$ be a minor-closed family of graphs. Then $\mathcal{F}$ has locally bounded treewidth if and only if $\mathcal{F}$ does not contain all apex graphs.* □

Example 2.1 *If G is a graph of treewidth at most k, then $ltw^G(r) \leq k$ for all $r \in N$.*

Example 2.2 *If $\Delta(G) = d$, where d is a constant, then $ltw^G(r) \leq d(d-1)^{r-1}$ for all $r \in N$.*

Examples 2.1 and 2.2 provide simple instances of graphs of locally bounded treewidth. A further example, obtained from Theorem 2.4, yields the class of graphs which do not contain the graph $K_{3,n}$ as a minor. We note that $K_{3,n}$ is an apex graph for all $n \geq 1$.

It is worth mentioning that the local treewidth in the proof of Theorem 2.4 is very large. Eppstein [Epp00] proved that for any surface S of (orientable or non-orientable) genus equal to γ, there exists a constant c such that for all graphs G embeddable in S and for all $r \geq 0$, $\mathrm{ltw}^G(r) \leq c \cdot \gamma \cdot r$. Grohe [Gro] proved the class of graphs almost embeddable in a surface S has linear local treewidth. Unfortunately, the constants involved in both results are immense. As we will see in Section 2.7, since local treewidth plays an important role in the running times of approximation algorithms on these graphs, we need a smaller function. So far, the only graphs determined to have local treewidth small enough to give practical approximation algorithms are planar graphs; Bodlaender [Bod98] proved that for planar G, $\mathrm{ltw}^G(r) \leq 3r - 1$. In Chapter 3, we will show that other classes of graphs, including $K_{3,3}$-minor-free or K_5-minor-free graphs, also have small local treewidth. We will prove for any $K_{3,3}$-minor-free or K_5-minor-free graph G, $\mathrm{ltw}^G(r) \leq 3r + 4$.

As all graphs of locally bounded treewidth studied so far have linear local treewidth, Grohe [Gro] raised an interesting open problem of whether there is a minor-closed family of graphs of locally bounded treewidth that does not have linear or polynomial local treewidth. We note that the known local treewidth for general graphs of locally bounded treewidth is very large.

2.4 General dynamic programming approach for graphs of bounded treewidth

Many NP-complete problems have linear-time or polynomial-time algorithms when they are restricted to graphs of bounded treewidth. There are a few techniques for obtaining such algorithms. The main technique is called *computing tables of characterizations of partial solutions.* This technique is a dynamic programming approach, first introduced by

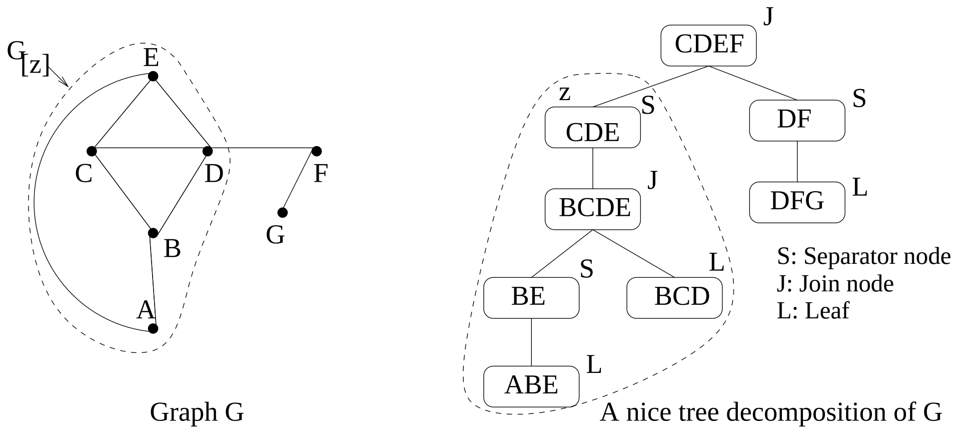

Figure 2.3: Graph G, a nice tree decomposition for G and terminal subgraph $G_{[z]}$.

Arnborg and Proskurowski [AP89]. This technique also appeared in a paper written by Bern et al. [BLW87]. Bodlaender [Bod97] described a better presentation of this technique. Other approaches applicable for solving problems on graphs of bounded treewidth are *graph reduction* [ACPS93, BdF96] and *describing the problems in logic* (see Section 2.5).

In this section, we sketch Bodlaender's general description of Arnborg and Proskurowski's method and give an example which can be solved by this approach. The reader is referred to the original paper [Bod97] for more detail or to other papers [ABFN00, AP89] for more examples. Using this approach, we also solve the subgraph isomorphism problem in Chapter 5.

Without loss of generality, we can assume that the given tree decomposition of G ($TD(G)$) is a *nice tree decomposition* [Bod98], which is a rooted binary tree and has three types of nodes:

1. a **leaf** node which does not have any children;

2. a **separator** node which has one child, and whose bag is a subset of its child's bag; and

3. a **join** node which has two children, and whose bag is the union of its children's bags.

A graph G and a nice tree decomposition for G are presented in Figure 2.3.

Bodlaender proved that for every graph G of treewidth at most k, if a tree decomposition of G of width k is given, we can transform it into a nice tree decomposition of the same width with $O(k \cdot |V(G)|)$ nodes in linear time [Bod98]. Because of this linear-time transformation, we assume that the given tree decomposition of G is a nice tree decomposition throughout this thesis.

Using a nice tree decomposition, the rest of the algorithm is as follows. We compute for each node z of $TD(G)$ a certain table. To compute this table for node z, we only use the tables already constructed for its children (if they exist) and the structure of G restricted to the bag of z (χ_z). We perform this computation in a bottom-up fashion. To solve the original problem, we inspect the table of the root r of $TD(G)$.

To explain what kind of table must be computed, we need further definitions. We define the *terminal subgraph* $G_{[z]}$ for a node z of $TD(G)$ to be the induced subgraph of G over vertices of χ_z and bags of descendants of z in $TD(G)$. For example, the terminal subgraph $G_{[z]}$ for a node z is depicted in Figure 2.3. The important property of a terminal subgraph $G_{[z]}$ is that since χ_z is a separator for G (Lemma 2.2) there is no edge between $V(G) - V(G_{[z]})$ and $V(G_{[z]}) - \chi_z$.

We are now ready to present more precisely the rest of the algorithm described in Bodlaender's paper [Bod97]. We assume a graph-theoretic problem P is given. We also demonstrate the meaning of each step when P is the 3-colorability problem, in which we determine whether it is possible to color the vertices of a graph G with three colors such that end-vertices of each edge of G have different colors.

Step 1: Define a general form of the *solution* to the problem P. For example, in 3-colorability, the solution is a mapping $f : V(G) \to \{1, 2, 3\}$, such that $\forall\{u, v\} \in E(G) : f(u) \neq f(v)$.

Step 2: Define a *partial solution.* A partial solution is a restriction of a solution to a terminal subgraph $G_{[z]}$. Intuitively, the partial solution describes the possible structure of a solution on G, when we consider only what happens on $G_{[z]}$. In 3-colorability, the partial solution is a 3-coloring of the vertices of the terminal subgraph $G_{[z]}$, i.e. a mapping $f' : V(G_{[z]}) \to \{1, 2, 3\}$, such that $\forall\{u, v\} \in E(G_{[z]}) : f'(u) \neq f'(v)$. For other problems such as minor containment or subgraph isomorphism, defining a partial solution may require more work (see Chapter 5 for further detail).

Step 3: Define an *extension of a partial solution.* We demonstrate the relationship between a partial solution and a solution. This step is often very simple. For instance, in 3-colorability, a mapping f of G is an extension of a partial solution f' for terminal subgraph $G_{[z]}$ if and only if f' is the restriction of f to $V(G_{[z]})$.

Step 4: Define a *characteristic* of a partial solution. A characteristic is a crucial part of a partial solution needed to determine whether or not a partial solution can be extended to a solution. If two partial solutions have the same characteristic then one can be extended to a solution if and only if the other can be extended to a solution. In 3-colorability, the characteristic of a partial solution $f' : V(G_{[z]}) \rightarrow \{1, 2, 3\}$ of terminal subgraph $G_{[z]}$ is the restriction of f' to χ_z. The assignment of colors to vertices of χ_z is all we need to determine how a partial solution of the terminal subgraph $G_{[z]}$ can be extended to a solution. The correctness of this characteristic follows from the fact that since χ_z is a separator in G, there are no edges of $E(G)$ between a vertex in $V(G_{[z]}) - \chi_z$ and a vertex in $V(G) - V(G_{[z]})$.

Step 5: Show that for each of the three types of nodes of $TD(G)$, we can efficiently build in polynomial time the *full set of characteristics* for terminal subgraph $G_{[z]}$ from the full sets of characteristics for its children (if they exist). The *full set of characteristics* for a terminal subgraph $G_{[z]}$ is the set of all characteristics of partial solutions on G. Intuitively, the full set of characteristics for terminal subgraph $G_{[z]}$ is all we need to know about $G_{[z]}$ when we solve the problem P. An important task in this step is to show that the cardinality of the full set of characteristics for each terminal subgraph is polynomial. In 3-colorability of a partial k-tree G, for each node z, $|\chi_z| \leq k+1$ and hence the full sets of characteristics have at most 3^{k+1} elements, which is a constant when k is constant. To construct the full set of characteristics for a separator node z we have this lemma:

Lemma 2.3 *[Bod97] Let z be a separator node and z' be its child. A characteristic mapping $f : \chi_z \rightarrow \{1, 2, 3\}$ belongs to the full set of characteristics for z if and only if there exists a characteristic mapping $f' : \chi_{z'} \rightarrow \{1, 2, 3\}$ belonging to the full set of characteristics for z' such that f is the restriction of f' to χ_z.*

The proof follows from the fact that because z is a separator node, $\chi_z \subseteq \chi_{z'}$ and thus $G_{[z]} = G_{[z']}$. Using this lemma, one can construct the full set of characteristics for a separator node from the full set of characteristics for its children. Similar lemmas exist for other kinds of nodes [Bod97]. Thus we can build the full set of characteristics for any node in $TD(G)$.

Step 6: Show that using the full set of characteristics for root r of $TD(G)$, we can solve the problem efficiently. For instance, in 3-colorability particularly, G is 3-colorable if and only if the full set of characteristics for the root is not empty (the partial solution corresponding to this characteristic is in fact a solution). This condition can be checked efficiently.

Approaches similar to that for 3-colorability can be used for many graph-theoretic problems, e.g. maximum independent set, minimum dominating set and minimum vertex cover [Arn85, Bod96, Bod97]. The most important steps of these algorithms are finding the right choices of characteristics and showing that the cardinality of each full set of characteristics is polynomial. Although it is usually possible to state lemmas for constructing the full set of characteristics for a node from the full sets of characteristics for its children, detailed work is usually required to prove the lemmas.

We use this dynamic programming technique to solve the subgraph isomorphism problem in Chapter 5. In this problem, we need to find a mapping from vertices of a host graph G to vertices of another graph H instead of to the set $\{1, 2, 3\}$.

2.5 Logical description of graph-theoretic problems

As mentioned in Section 2.4, describing problems in logic is an applicable approach for solving graph-theoretic problems on graphs of bounded treewidth. A general framework for describing several graph-theoretic properties in logic is monadic second order logic, defined formally below.

Monadic second order logic (MSOL) is a language for expressing properties, especially graph-theoretic ones, in logic. It has variables for vertices, edges, sets of edges and sets of vertices. Its logical connectives are **and**, **or** and **not**. Quantifiers $\forall$ and $\exists$ can be applied to the variables. In addition, it has four binary relations: set membership ($s \in S$), adjacency test for vertices ($adj(u, v)$), incidency test for vertices and edges ($inc(v, e)$) and equality for variables ($a = b$).

Arnborg et al. [ALS88] extend MSOL to extended monadic second order logic (EMSOL) to have counting or summing evaluations over sets. This feature allows us to define minimization or maximization problems in monadic second order logic. They show that problems definable in EMSOL have linear-time or polynomial-time algorithms for graphs of bounded treewidth. The reader is referred to this paper for a list of some famous NP-optimization problems definable in EMSOL.

For example, we show how the maximum independent set problem for a graph $G = (V, E)$ can be described in EMSOL.

$$max_{V' \subseteq V} V' : \forall u \forall v \exists e (u \in V' \wedge v \in V' \wedge inc(u, e) \wedge inc(v, e)) \Rightarrow \neg(e \in E)$$

In the above formula, in a graph $G = (V, E)$ we search for a set $V' \subseteq V$ of maximum size such that there exists no edge in E both of whose end-vertices are in V'. This is the exact definition of the maximum independent set problem.

Courcelle [Cou90] related MSOL to the notion of treewidth.

Theorem 2.5 *[Cou90] Let w be a fixed constant and ϕ be a property of graphs that is definable in monadic second order logic. Then ϕ can be decided in linear time on graphs of treewidth at most w.* $\square$

Because of the large hidden constant in the complexity of linear-time algorithm of Theorem 2.5, this theorem does not provide practical algorithms. It still provides a simple way to determine if a property is linear-time decidable on partial k-trees. Abrahamson and Fellows [AF93] gave a more straightforward automata-theoretic proof of Courcelle's theorem.

Unfortunately, the analogue of Courcelle's theorem does not hold for NP-complete problems which have a monadic second order definition on graphs of locally bounded treewidth. Instead, there is a similar theorem for a somewhat limited class of NP-complete problems which can be defined using *first-order logic*, a restricted form of monadic second order logic, in which we do not have variables for sets and operations for set membership.

Theorem 2.6 *[FG99] Let C be a class of graphs of locally bounded treewidth and let ϕ be a property definable in first-order logic. Then for every $k \geq 1$, there is an algorithm which in time $O(n^{1+(1/k)})$ decides whether a given graph $G \in C$ has property ϕ, where n is the number of vertices of the graph G.* □

Hamiltonicity and 3-colorability are examples of properties which have monadic second order logic descriptions but not first-order logic descriptions [DF99]. Examples of first-order definable problems are the k-dominating set problem and the k-independent set problem for fixed k [DF99]. In the former problem, one searches for a set of k vertices of a graph such that each of the rest of the vertices has at least one neighbor in the set, and in the latter problem, one searches for a set of k vertices of a graph such that there exists no edge of the graph both of whose end-vertices are in the set.

Frick and Grohe also improved the running time of the algorithm mentioned in Theorem 2.6 for minor-closed families of graphs of locally bounded treewidth.

Theorem 2.7 *[FG99] Let C be a minor-closed class of graphs that have locally bounded treewidth and ϕ be a property definable in first-order logic. Then there is a linear-time algorithm deciding whether a given graph $G \in C$ has property ϕ.* □

The hidden constant in the complexity of the linear-time algorithm of Theorem 2.7, similar to that of Courcelle's theorem, is very large. In Chapter 3, we show how this large hidden constant can be improved for special graphs such as $K_{3,3}$-minor-free graphs and K_5-minor-free graphs.

Using Theorem 2.7 and the fact that for fixed k, k-dominating set and k-independent set are first-order expressible properties on graphs, we have linear-time algorithms deciding whether a given graph G has these properties. Frick and Grohe [FG99] also generalized Theorem 2.7 to structures other than graphs. For example, consider the (k, d)-circuit satisfiability problem, for $d \geq 1$, in which one decides whether a given boolean circuit of depth at most d has a satisfying assignment such that at most k input gates are set to **true**. They proved this problem can be solved in linear time for circuits whose underlying graphs are in a minor-closed family of graphs of locally bounded treewidth. Another example is evaluating a (boolean) database query against a relational database expressed in the relational calculus. As relational calculus is contained in first-order logic, they showed that Boolean relational calculus queries can be evaluated in linear time for a database whose underlying graph is in a minor-closed family of graphs of locally bounded treewidth.

Seese [See96] presented a theorem similar to Theorem 2.7 for bounded degree graphs.

Theorem 2.8 *[See96] For every first-order definable property of graphs there is a linear-time algorithm that decides whether a given graph of constant degree has this property.* $\square$

2.6 Fixed parameter tractability

Developing practical algorithms for NP-hard problems is an important issue. Two common methods for dealing with NP-hard problems are heuristic and approximation methods. In heuristic methods, we are often unable to analyze the methods theoretically and thus we focus on approximation methods in theoretical computer science. Nevertheless, there are many NP-complete problems such as dominating set which are not believed to have constant factor approximations (see Garey and Johnson's standard book [GJ79]). Therefore, restrictions of these problems to certain graphs, e.g. planar graphs, have been considered. Recently, Downey and Fellows [DF99] introduced another concept to handle NP-hardness, namely *fixed parameter tractability*. For many NP-complete problems, the inherent combinatorial explosion is often due to a small part of a problem, namely a *parameter*. The parameter is often an integer and small in practice. The running times of simple algorithms may be exponential in the parameter but polynomial in the problem size. For example, for the k-vertex cover problem, in which we search for a vertex cover of size k (k is a parameter), a simple algorithm is to check all subsets of size at most k. The running time of this algorithm is $O(n^{k+1})$, where the exponent is a function of k. In fixed parameter tractability, we search for algorithms with running time $O(f(k)n^{O(1)})$. Such algorithms may be practical for small values of k. For instance, it has been shown that k-vertex cover has an algorithm with running time $O(kn + 1.3^k)$ [NR99] and hence this problem is fixed parameter tractable.

Definition 2.7 *[DF99] A parameterized problem $L \subset \Sigma^* \times N$ is* fixed parameter tractable (FPT) *if there is an algorithm that correctly decides, for input $(x, k) \in \Sigma^* \times N$, whether $(x, k) \in L$ in time $f(k)n^c$, where n is the size of the main part of the input x, $|x| = n$, k is* *a parameter (usually an integer), c is a constant independent of k, and f is an arbitrary function.*

One of the interesting and important properties of fixed parameter tractability is that the definition is *unchanged* if we replace time $f(k)n^c$ by time $f'(k) + n^{c'}$ in the above definition.

Downey and Fellows [DF99] have also introduced a hierarchy of parameterized complexity classes $FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P]$ and the concepts of reduction and completeness for these classes. Here we mean by complexity class FPT the class of all problems which are FPT (we use term FPT as a class when we mention it explicitly). It is conjectured that the hierarchy is proper. Thus, $W[i]$-complete problems, $i \geq 1$, are likely not to be fixed parameter tractable. For instance, independent set is complete for $W[1]$ and dominating set is complete for $W[2]$ [DF99]. Both problems are FPT when restricted to planar graphs, e.g. Alber et al. [ABFN00] have shown that planar k-dominating set can be solved in time $O(c^{\sqrt{k}}n)$, where $c = 3^{6\sqrt{34}}$. In Chapter 3, we show that these problems are FPT when restricted to $K_{3,3}$-minor-free or K_5-minor-free graphs.

Another property of FPT is its relation to approximation algorithms for NP-optimization problems. First, for some NP-optimization problems, Garey and Johnson [GJ79] proved that there is no good approximation algorithm, even assuming $P \neq NP$. Arora et al. [ALM+98] extended the result to many other problems. FPT provides another tool for the extension of the result to many problems not covered by the results of Arora et al.

Theorem 2.9 *[DF99] An NP-optimization problem has a fully polynomial-time approximation scheme if and only if it is fixed parameter tractable.* □

Assuming $W[1] \neq FPT$, the NP-optimization problems that are $W[1]$-hard do not have fully polynomial-time approximation schemes. This result can be extended to PTASs, as well.

Definition 2.8 *An approximation algorithm for an optimization problem is* efficient *if it computes a solution within a factor $(1+1/k)$ of the optimal in time $O(f(k)n^c)$ for a function f and a constant c.*

All approximation algorithms introduced in Baker's paper [Bak94] and Chapter 3 are efficient.

Theorem 2.10 *[DF99] Unless $FPT = W[1]$, an NP-optimization problem is fixed parameter intractable if and only if it has no efficient PTAS.* □

Theorem 2.10 provides a powerful means for proving that a problem does not have an efficient PTAS.

Methods for designing FPT algorithms include a variety of techniques such as well-quasi-ordering, bounded treewidth, color coding (hashing) and elementary methods. The reader is referred to the book of Downey and Fellows [DF99] as a good reference for these methods.

2.7 Approximation algorithms for graphs of locally bounded treewidth

Many results design PTASs restricted to certain special graphs, especially graphs of bounded or locally bounded treewidth. Lipton and Tarjan [LT80] were the first who proved various NP-optimization problems have PTASs over planar graphs. Unfortunately, Chiba et al. have shown to reach a performance ratio half of the optimal in Lipton and Tarjan's work, the graph must have at least $2^{2^{400}}$ vertices and so their approach is known to be impractical [CNS82]. Using a different approach, Baker [Bak94] gave practical PTASs for the problems considered by Lipton and Tarjan. Alon *et al.* [AST90] generalized Lipton and Tarjan's ideas to graphs without a fixed minor. Like Lipton and Tarjan's PTASs, their PTASs were impractical too.

By partitioning a graph into three forests, Chen and He [CH95, Che95] obtained efficient approximation algorithms of ratio 3 for many NP-hard hereditary maximization problems on planar, $K_{3,3}$-minor-free graphs and K_5-minor-free graphs (these graphs have locally bounded treewidth by Theorem 2.4). After that, Chen [Che98], using Baker's approach, found approximation algorithms of ratio $1 + 1/\log n$ for NP-hard hereditary maximization problems on $K_{3,3}$-minor-free graphs and K_5-minor-free graphs. His approach was a non-trivial generalization of Baker's approach for these types of graphs. Baker's technique decomposes a planar embedding by successively deleting outer faces; vertices are given level numbers corresponding to the iterations in which they are deleted. Removing only those vertices with level number congruent to i mod k results in a k-outerplanar graph; there are k choices of i, and every vertex is in exactly $k-1$ of the resulting k-outerplanar graphs. Many NP-complete problems (such as maximum independent set, minimum dominating set, and minimum vertex cover) can be solved exactly on k-outerplanar graphs by dynamic programming. Suppose s_i, $1 \leq i \leq k$, is the optimal solution for the ith k-outerplanar graph. Baker [Bak94] shows that by taking the best among $s_1, \cdots, s_k$ as a (nearly optimal) solution for the original graph, we have a solution within a factor of $(1+1/k)$ of the optimal. Chen's approach differs from Baker's mainly in construction of layers. Because of his special construction of layers, his approach only applies to inherent maximization problems such as the maximum independent set problem.

Eppstein [Epp00] showed that Baker's technique can be extended by replacing bounded outerplanarity with bounded local treewidth. As with k-outerplanar graphs, a wide range of NP-complete problems can be solved in linear time on graphs of bounded treewidth (see Section 2.4). The decomposition by deleting every kth face is replaced by deleting every kth level of a breadth-first tree of G, provided that the treewidth of the resulting graphs is a function of k. In Chapter 3, we consider this approach in more detail, when we solve many NP-optimization problems on $K_{3,3}$-minor-free graphs and K_5-minor-free graphs in linear and quadratic time.

2.8 Subgraph isomorphism

We consider the subgraph isomorphism problem, in which we search for a subgraph of host graph H isomorphic to the source (or pattern) graph G, in Chapter 5. This problem has many applications in different areas such as biology and organic chemistry [ABG$^+$92].

Because the subgraph isomorphism problem is NP-complete [GJ79], much attention has focused on solving this problem by adding restrictions to the source or host graph. Specific classes of graphs for which there are polynomial-time algorithms are as follows: trees [Mat78], two-connected outerplanar graphs [Lin89], and two-connected series-parallel graphs [LS88]. These are all graphs of bounded treewidth. Nevertheless, the subgraph isomorphism problem remains NP-complete for general graphs of bounded treewidth [Sys82].

As mentioned in Section 2.5, graphs of bounded treewidth allow us to use a dynamic programming approach for solving problems, especially those which can be described in the language of extended monadic second order logic (EMSOL). Except the case in which the source graph is fixed, the subgraph isomorphism problem can not be expressed in EMSOL, and the general approach of Arnborg et al. [ALS88] can not be used. We need to add more restrictions to solve this problem in polynomial time. Matoušek and Thomas [MT92] have shown the subgraph isomorphism problem has an $O(n^{k+4.5})$ time algorithm for k-connected partial k-trees and an $O(n^{k+2})$ time algorithm for bounded degree partial k-trees. They have proved that the problem remains NP-complete when the source graph is a tree, and the host graph is a partial 2-tree which has at most one node of degree greater than three. Gupta and Nishimura [GN94], using a different approach, derive polynomial-time algorithms with the same asymptotic complexity for the k-connected partial k-tree case and other embeddings. Dessmark et al. [DLP00a] improve the running time of Gupta and Nishimura's algorithm from $O(n^{k+4.5})$ to $O(n^{k+2})$ for the case of k-connected partial k-trees. Gupta and Nishimura have proved the subgraph isomorphism problem on partial k-trees remains NP-complete when the source graph is not k-connected [GN96b]. This problem has been considered for weaker classes of graphs such as partial k-paths, and it has been shown that for k-connected partial k-paths there is an $O(n^3)$ time algorithm for solving the subgraph isomorphism problem; here the exponent is independent of k [GN96a].

Matoušek and Thomas's approach mainly uses the general ideas of dynamic programming introduced by Arnborg and Proskurowski (see Section 2.4); however they have presented their algorithm in a very complicated manner. Their approach has been stated for the minor containment problem and the solution for subgraph isomorphism can be obtained as an special case of this approach. The main idea is a tricky definition of the partial solution and its characteristic. This is a good example of how Arnborg and Proskurowski's method can be used for problems which are not definable in monadic second order logic. In Chapter 5, we extend the bounded degree result of Matoušek and Thomas to handle a more general property, namely bounded fragmentation. A graph is a *log-bounded fragmentation* graph if, after removing any set of at most k vertices, the number of connected components is at most $O(k \log n)$, where n is the number of vertices of the graph. The class of bounded fragmentation graphs contains the class of bounded degree graphs and other classes of graphs such as the class of Hamiltonian graphs (see Chapter 4 for the proof).

Gupta and Nishimura use a different approach. First, they introduce the concept of *normalized tree decomposition* of a graph. Roughly speaking, a *normalized tree decomposition of a partial k-tree H (NTD(H))* is a tree decomposition of H in which nodes of $NTD(H)$ are partitioned into the set of *separator nodes*, whose bags have size k, and the set of *clique nodes*, whose bags have size $k+1$. The root of $NTD(H)$ is a separator node, the leaves of $NTD(H)$ are clique nodes, children of each separator node are clique nodes and children of non-leaf clique nodes are separator nodes. Gupta and Nishimura prove a normalized tree decomposition for a partial k-tree G can be constructed in $O(n^2)$ time. In addition, they introduce another concept called a *tree decomposition graph* of a graph G (TDG(G)). Intuitively, if G is isomorphic to a subgraph of H, $TDG(G)$ is a directed graph which contains all appearances of normalized tree decompositions G in $NTD(H)$. A node in $TDG(G)$ corresponds to a potential node in $NTD(H)$ and an edge corresponds to a potential edge in $NTD(H)$. Each node in $TDG(G)$ is also either a separator node or a clique node. Using these two concepts, Gupta and Nishimura generalize Matula's algorithm [Mat78] for subtree isomorphism to subgraph isomorphism. They process the nodes of $NTD(H)$ bottom-up and at each node z of $NTD(H)$, they determine whether each node of $TDG(G)$ can be mapped to z. To this end, they use a bipartite matching between children of z and successors of each node of $TDG(G)$. The polynomial running time of the algorithm follows from the polynomial size of $NTD(H)$ and $TDG(G)$ and the polynomial running time of the bipartite matching algorithm.

Gupta and Nishimura's method for solving subgraph isomorphism has also been used to solve other related problems such as embedding problems [GN94], finding the largest common subgraph [Bra01] and finding a maximum packing in which we search for the maximum number of disjoint copies of a source graph in a host graph [DLP00b].

Generalizations of isomorphism include homomorphism and minor containment. In the former generalization, the edges of the source graph are mapped to edge disjoint paths in the host graph and in the latter generalization, the vertices of the source graph are mapped to connected subgraphs of the host graph. Naturally, both of these generalizations are NP-complete in the general case, but the former generalization supports polynomial-time algorithms for the cases of k-connected [GN94] and bounded degree [MT92] partial k-trees and the latter generalization supports a polynomial-time algorithm only for the bounded degree case [MT92]. Gupta et al. [GNPR00] present a more efficient algorithm for partial k-paths.

Using Baker's approach (see Section 2.7), Eppstein solves the subgraph isomorphism from a fixed pattern G into a graph H of locally bounded treewidth. The idea is as follows. First, for graph H, we construct the layers introduced in Baker's approach. If a fixed pattern G appears in H, it must appear in $diam(G)$ consecutive layers of G. We can prove each subgraph induced on a constant number of layers has bounded treewidth, and thus we can solve subgraph isomorphism by the general dynamic programming approach. By considering all choices of $diam(G)$ consecutive layers of H, we can solve subgraph isomorphism in linear time. In Chapter 5, we give more detail of this approach when we present an algorithm for testing subgraph isomorphism from a bounded fragmentation graph G (not a fixed pattern G) into a graph H of locally bounded treewidth.

In this chapter, we introduced preliminary definitions, related topics and the survey of the current results about designing PTASs on graphs of locally bounded treewidth and the subgraph isomorphism problem. New results on these concepts are presented in Chapters 3 and 5.

Chapter 3

Algorithms for $K_{3,3}$-minor-free or K_5-minor-free graphs

In this chapter, we extend Baker's approach for designing PTASs for NP-optimization problems on planar graphs (see Section 2.7). As mentioned in Section 2.1, a planar graph is both $K_{3,3}$-minor-free and K_5-minor-free. We generalize Baker's ideas to $K_{3,3}$-minor-free graphs and K_5-minor-free graphs. The reader is referred to Section 2.7 for discussion of previous work in this area.

We introduced the concept of local treewidth in Section 2.3. We mentioned that Eppstein [Epp00] characterized graphs of locally bounded treewidth and showed how Baker's ideas [Bak94] for designing PTASs could be generalized to graphs of locally bounded treewidth. Unfortunately, the hidden constant involved in Eppstein's work is large, and it causes the resulting PTASs to be impractical.

As mentioned in Section 2.3, Bodlaender [Bod98] showed that for a planar graph G, $\mathrm{ltw}^G(k) \leq 3k - 1$. In this chapter, we generalize Bodlaender's result demonstrating linear local treewidth of planar graphs to $K_{3,3}$-minor-free graphs and K_5-minor-free graphs. In fact, we present a more general theorem: we prove that if a graph H is a single-crossing graph (can be drawn on the plane with at most one crossing) then the local treewidth of any H-minor-free graph is bounded above by $3k + c_H$ where c_H is a constant depending only on H.

The results of this chapter, especially Section 3.1, are mainly obtained from joint work

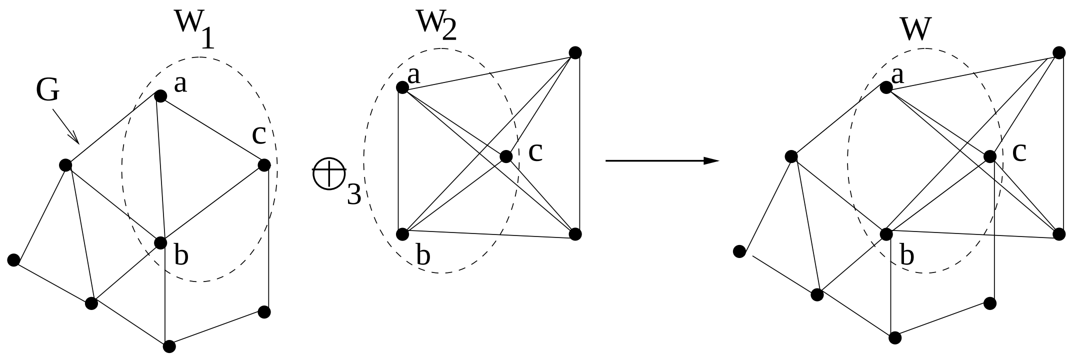

Figure 3.1: Graph summation operation: identifying sets W_1 and W_2 and deleting edge $\{a, b\}$

with Nishimura, Ragde and Thilikos [HNRT01].

The rest of this chapter is organized as follows. First, we present our main results on local treewidth and construction of tree decompositions of $K_{3,3}$-minor-free or K_5-minor-free graphs (Section 3.1). Then, we show how our results can be applied to find algorithms and practical PTASs for these graphs (Sections 3.2 and 3.3).

3.1 Local treewidth of clique-sum graphs

In this section, first we show H-minor-free graphs, where H is a single-crossing graph, have linear local treewidth. Then we introduce the concept of layers for these graphs and present a practical algorithm for construction of a tree decomposition of any subgraph induced on a constant number of consecutive layers.

The graph summation operation plays an important role in our results. Suppose G_1 and G_2 are graphs with disjoint vertex-sets and $k \geq 0$ is an integer. For $i = 1, 2$, let $W_i \subseteq V(G_i)$ form a clique of size k and let G'_i $(i = 1, 2)$ be obtained from G_i by deleting some (possibly no) edges from $G_i[W_i]$ with both endpoints in W_i. Consider a bijection $h : W_1 \to W_2$. We define a *k-sum* G of G_1 and G_2, denoted by $G = G_1 \oplus_k G_2$ or simply by $G = G_1 \oplus G_2$, to be the graph obtained from the union of G'_1 and G'_2 by identifying w with $h(w)$ for all $w \in W_1$. The images of the vertices of W_1 and W_2 in $G_1 \oplus_k G_2$ form the *join set*. In the rest of this section, when we refer to a vertex v of G in G_1 or G_2, we mean the corresponding vertex of v in G_1 or G_2 (or both). The reader is referred to Figure 3.1 to see an example in which a 3-sum of a graph G and K_5 is depicted.

We use the following three simple lemmas to obtain our main result.

Lemma 3.1 *For any graph G and subgraph G' of G, $\mathrm{ltw}^{G'}(k) \le \mathrm{ltw}^{G}(k)$, for any $k \ge 0$.*

Proof: It is enough to observe that for any $v \in G'$ and $k \ge 0$, $N^k_{G'}(v) \subseteq N^k_G(v)$. Thus the removal of vertices of $N^k_G(v) \setminus N^k_{G'}(v)$ from bags of a tree decomposition of $N^k_G(v)$ results in a tree decomposition of $N^k_{G'}(v)$ with the same local treewidth or less. □

Lemma 3.2 *For any two graphs G and H, $\mathrm{tw}(G \oplus H) \le \max\{\mathrm{tw}(G), \mathrm{tw}(H)\}$.*

Proof: Let W be the set of vertices of G and H identified during the $\oplus$ operation. Since W is a clique in G, in every tree decomposition of G, there exists a node α such that W is a subset of χ_α [BM93]. Similarly, it is true for W and a node α' of each tree decomposition of H. Hence, we can construct a tree decomposition of G and a tree decomposition of H and add an edge between α and α'. In fact, every vertex (edge) of $G \oplus H$ is a vertex (an edge) in G or H and thus appears in a bag of the tree decomposition of $G \oplus H$ (see Properties (1) and (2) of tree decompositions). The set W is the only common set of vertices of G and H in $G \oplus H$. Nodes whose bags contain a vertex $w \in W$ form connected subtrees in tree decompositions of G and H. Hence, by adding an edge between α and α', the nodes also form a connected subtree in the tree decomposition of $G \oplus H$ (see Property (3) of tree decompositions). □

Lemma 3.3 *For any graph G, any clique R of G, any $v \in R$, and any $k \ge 0$,* $\mathrm{tw}(G[N^k_G(R)]) \le \mathrm{tw}(G[N^{k+1}_G(v)])$.

Proof: We note that all vertices in $R - v$ are at distance 1 from v. Therefore $N^k_{G_1}(R) \subseteq N^{k+1}_{G_1}(v)$, and the result follows from Lemma 3.1. □

Lemma 3.4 shows how the local treewidth changes when we apply a graph summation operation.

Lemma 3.4 *If G_1 and G_2 are graphs where $\mathrm{ltw}^{G_i}(r) \le f(r)$, $f(r) \ge 0$ for all $r \in \mathbb{N}$, and $G = G_1 \oplus_k G_2$, then $\mathrm{ltw}^{G}(r) \le f(r)$.*

Proof: To show $\mathrm{ltw}^G(r) \leq f(r)$, we prove for any $v \in V(G)$ and for all $r \geq 0$, $\mathrm{tw}(G[N_G^r(v)]) \leq f(r)$. Since $f(r) \geq 0$, the claim is clear for $r = 0$. Thus we assume $r > 0$ in the rest of the proof. Let W be the join set of $G_1 \oplus_k G_2$. Without loss of generality, we can assume v is from G_1. If $N_G^r(v)$ contains only vertices originally from G_1, the result follows from our initial assumption about G_1, i.e. $\mathrm{ltw}^{G_1}(r) \leq f(r)$.

We now assume $N_G^r(v)$ contains vertices from G_2. If $v \in W$, then $N_G^r(v) \subseteq N_{G_1}^r(v) \cup N_{G_2}^r(v)$. In addition, since $r \geq 1$ and vertices of W form a clique in G_i for $i = 1, 2$, $W \subseteq N_{G_i}^r(v)$. Using these two facts, $G[N_G^r(v)]$ is a subgraph of $G_1[N_{G_1}^r(v)] \oplus G_2[N_{G_2}^r(v)]$ over the join set W. Thus, by Lemmas 3.1 and 3.2, we know

$$tw(G[N_G^r(v)]) \leq max\{tw(G_1[N_{G_1}^r(v)]), tw(G_2[N_{G_2}^r(v)])\} \leq f(r).$$

We now consider the case in which $v \notin W$ and there exists a vertex $u \in N_G^r(v) - W$ which is from G_2. Since W is the only common set of vertices of G_1 and G_2 in G, at least one vertex of W is on the shortest path from v to u in G and hence is at distance of at most $r-1$ from v. Therefore, $W \cap N_{G_1}^{r-1}(v) \neq \emptyset$. Let $w \in W \cap N_{G_1}^{r-1}(v)$ be the vertex with minimum distance p from v where $1 \leq p \leq r-1$. We observe that each vertex u with the aforementioned property is at distance at most $r-p$ from at least one vertex of W. Thus $N_G^r(v) \subseteq N_{G_1}^r(v) \cup N_{G_2}^{r-p}(W)$. As one vertex of W is at distance $p \leq r-1$ from v and vertices of W form a clique in G_1, each vertex of W is at distance at most r from v in G_1, *i.e.* $W \subseteq N_{G_1}^r(v)$. Also, $W \subseteq N_{G_2}^{r-p}(W)$. Thus we can obtain $G_1[N_{G_1}^r(v)] \oplus G_2[N_{G_2}^{r-p}(W)]$ over the join set W. As mentioned above $N_G^r(v) \subseteq N_{G_1}^r(v) \cup N_{G_2}^{r-p}(W)$, and thus $G[N_G^r(v)]$ is a subgraph of $G_1[N_{G_1}^r(v)] \oplus G_2[N_{G_2}^{r-p}(W)]$. Hence, by Lemma 3.2,

$$tw(G[N_G^r(v)]) \leq max\{tw(G_1[N_{G_1}^r(v)]), tw(G_2[N_{G_2}^{r-p}(W)])\}. \tag{3.1}$$

By Lemma 3.1, since $G_2[N_{G_2}^{r-p}(W)]$, $p \geq 1$, is a subgraph of $G_2[N_{G_2}^{r-1}(W)]$,

$$tw(G_2[N_{G_2}^{r-p}(W)]) \leq tw(G_2[N_{G_2}^{r-1}(W)]). \tag{3.2}$$

Thus by 3.1 and 3.2 and the fact that $tw(G_1[N_{G_1}^r(v)]) \leq f(r)$ (our assumption about G_1),

$$tw(G[N_G^r(v)]) \leq max\{f(r), tw(G_2[N_{G_2}^{r-1}(W)])\}. \tag{3.3}$$

Since W is a clique in G_2, by Lemma 3.3,

$$tw(G_2[N_{G_2}^{r-1}(W)]) \leq tw(G_2[N_{G_2}^{r}(w)]) \leq f(r). \tag{3.4}$$

Finally using 3.3 and 3.4, we conclude that $tw(G[N_G^r(v)]) \leq f(r)$. □

It is known that any H-minor-free graph G, for single-crossing graph H (see introduction of this chapter), can be obtained from planar graphs and graphs of treewidth at most c_H by means of a series of k-sums, $0 \leq k \leq 3$, where c_H is a constant dependent only on the single-crossing graph H [RS93]. Because of this property, we call H-minor-free graphs *clique-sum graphs* when H is a single crossing graph. A series of k-sums (not necessarily unique) which generate a clique-sum graph G are called *a set of clique-sum operations* of G. Lemma 3.5 follows from this definition of clique-sum graphs:

Lemma 3.5 *For any clique-sum graph G which excludes a single crossing graph H as a minor, any minor G' of G is also a clique-sum graph which excludes the same graph H as a minor.*

Proof: The proof follows from the fact that if G' is a minor of G and G is H-minor-free, then G' is H-minor-free too. □

Theorem 3.1 demonstrates our main result on the local treewidth of clique-sum graphs.

Theorem 3.1 *For any clique-sum graph G excluding a single-crossing graph H as a minor and for all $r \geq 0$, $\mathrm{ltw}^G(r) \leq 3r + c_H$.*

Proof: By the definition of clique-sum graphs, we can assume $G = G_1 \oplus G_2 \oplus \cdots \oplus G_m$ where each G_i, $1 \leq i \leq m$, is either a planar graph or a graph of treewidth at most c_H. We use induction on m, the number of G_i's. For $m = 1$, we wish to show that G_1 is either a planar graph whose local treewidth is $3r - 1$ or a graph of treewidth at most c_H. In the former case $\mathrm{ltw}^G(r) = \mathrm{ltw}^{G_1}(r) = 3r - 1 \leq 3r + c_H$, $c_H \geq 0$, and in the latter case $\mathrm{ltw}^G(r) = \mathrm{ltw}^{G_1}(r) = c_H \leq 3r + c_H$, $r \geq 0$. Thus the basis of induction is true for both cases. We assume the induction hypothesis is true for $m = h$, and we prove the hypothesis for $m = h + 1$. Let $G' = G_1 \oplus G_2 \oplus \cdots \oplus G_h$ and $G'' = G_{h+1}$. Thus $G = G' \oplus G''$. By the induction hypothesis, $ltw^{G'}(r) \leq 3r + c_H$ and $ltw^{G''}(r) \leq 3r + c_H$. The proof, for $m = h+1$, follows from this fact and Lemma 3.4. □

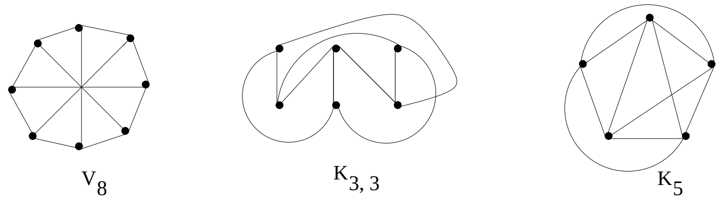

Figure 3.2: Graph V_8 and single-crossing embeddings of $K_{3,3}$ and K_5

Using the fact that K_5 and $K_{3,3}$ are single-crossing graphs (Figure 3.2), we observe that K_5-minor-free graphs and $K_{3,3}$-minor-free graphs are clique-sum graphs. Wagner [Wag37] gave a better characterization for these graphs. He proved that a graph has no minor isomorphic to $K_{3,3}$ if and only if it can be obtained from planar graphs and K_5 by 0-,1-, and 2-sums. He also showed that a graph has no minor isomorphic to K_5 if and only if it can be obtained from planar graphs and V_8, shown in Figure 3.2, by 0-,1-,2-, and 3-sums. Since both K_5 and V_8 have treewidth four, the value of constant c_H in the proof of Theorem 3.1 is four, and we have:

Corollary 3.1 *If G is a K_5-minor-free or $K_{3,3}$-minor-free graph then* $\mathrm{ltw}^G(k) \leq 3k + 4$.

$\square$

As mentioned in Section 2.7, the concept of the kth outer face in planar graphs can be replaced by the concept of the kth layer (or level) in graphs of locally bounded treewidth. The kth layer (L_k) of a graph G consists of all vertices at distance k from an arbitrary fixed vertex v of $V(G)$. We denote *consecutive layers from i to j* by $L[i, j] = \bigcup_{i \leq k \leq j} L_k$.

Theorem 3.2 *For any clique-sum graph G, the treewidth of $G[L[i, j]]$ is bounded above by $3(j - i + 1) + c_H$.*

Proof: By contracting the connected subgraph $G[L[0, i - 1]]$ to a vertex v' and applying Lemma 3.5, we obtain another clique-sum graph G'. As all vertices at distance d, $i \leq d \leq j$, from v in G are at distance d', $1 \leq d' \leq j - i + 1$, from v' in G' and all vertices at distance more than j from v in G are at distance more than $j - i + 1$ from v' in G', we have $G[L[i, j]] = G'[L[1, j - i + 1]]$. Thus $tw(G[L[i, j]]) = tw(G'[L[1, j - i + 1]])$. Since all vertices of $L[1, j-i+1]$ in G' are in the $j-i+1$-neighborhood of v', $tw(G'[L[1, j-i+1]]) \leq tw(G'[N_{G'}^{j-i+1}(v')])$. By the definition of local treewidth, $tw(G'[N_{G'}^{j-i+1}(v')]) \leq ltw^{G'}(j-i+1)$. Finally by Theorem 3.1, we have $ltw^{G'}(j-i+1) \leq 3(j-i+1)+c_H$. Using these facts, $tw(G[L[i,j]]) \leq 3(j-i+1)+c_H$, as desired. □

Theorem 3.2 gives an upper bound on the treewidth of consecutive layers from i to j, but it does not provide a constructive algorithm to obtain a tree decomposition of this width. As mentioned in Section 2.2, using Bodlaender's algorithm [Bod96], we can construct a tree decomposition of this width in linear time, but the hidden constant factor is huge, and the algorithm is impractical. Below we give a practical algorithm which constructs a tree decomposition of width $3(j-i+1)+c_H$ for consecutive layers from i to j in $K_{3,3}$-minor-free or K_5-minor-free graphs. First we determine a set of clique-sum operations of $K_{3,3}$-minor-free or K_5-minor-free graphs.

Theorem 3.3 *[KM92] A set of clique-sum operations of a K_5-minor-free graph $G = G_1 \oplus G_2 \oplus \cdots \oplus G_m$ can be found in $O(n^2)$ time such that $\sum_{i=1}^{m} |V(G_i)| = O(|V(G)|)$.* □

Asano [Asa85] presented an $O(n)$ time algorithm for finding a set of clique-sum operations of a graph with no subgraph homomorphic to $K_{3,3}$. As for a cubic graph H (degree of each vertex is at most three), H is a minor of G if and only if G contains a subgraph homeomorphic to H, we have:

Theorem 3.4 *A set of clique-sum operations of a $K_{3,3}$-minor-free graph $G = G_1 \oplus G_2 \oplus \cdots \oplus G_m$ can be found in $O(n)$ time such that $\sum_{i=1}^{m} |V(G_i)| = O(|V(G)|)$.* □

Before stating the main theorem on construction of a tree decomposition of consecutive layers, we present a simple lemma.

Lemma 3.6 *Let $G = G_1 \oplus G_2 \oplus \cdots \oplus G_m$ be a clique-sum graph. If there exists a vertex $v \in V(G)$ such that each vertex of G is at distance at most r from v, then in each G_i, $1 \leq i \leq m$, there exists a vertex v_i such that each vertex of G_i is at distance at most r from v_i.*

Proof: We use induction on m, the number of G_i's. If $m = 1$, the basis of induction is clearly true. We assume the induction hypothesis is true for $m \leq h$, and we prove the

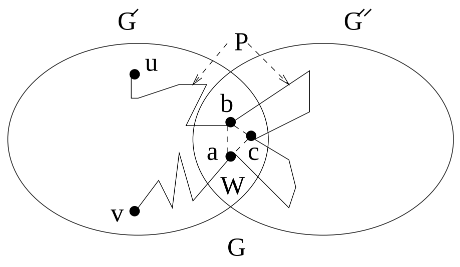

Figure 3.3: The replacement of the part of path P between a and b by edge $\{a, b\}$

hypothesis for $m = h + 1$. We suppose $G = G' \oplus G''$ where $G' = G_1 \oplus G_2 \oplus \cdots \oplus G_h$ and $G'' = G_{h+1}$. We let W be the join set of $G' \oplus G''$.

First we consider the case in which vertex v defined in the statement of Lemma is in W. We show that v has a path of length at most r in G' (G'') to each vertex u in G' (G''). Without loss of generality, we take a vertex $u \in V(G')$. If a part of a shortest path P in G from v to u goes through vertices in $V(G'') - V(G')$, this part goes through vertices of W. Let a be the first vertex in W and b be the last vertex in W on path P from v to u (see Figure 3.3). We note that a and b can be the same vertex. Since vertices of W form a clique in G', edge $\{a, b\}$ is present in G'. We can replace the part of path P which goes through vertices in $V(G'')$ (and has length at least one) by edge $\{a, b\}$ and obtain a path from v to u in G' with length less than the length of P. Thus, there is a path of length at most r in G' from v to each vertex u in G'. Using the induction hypothesis for G' and G'', we obtain the result for G.

We now consider the case in which $v \in V(G) - W$. Without loss of generality, we assume $v \in V(G') - W$. A shortest path in G from v to a vertex u of G'' goes through a vertex in W whose distance is at least one from v. Hence each vertex of G'' is at distance at most $r - 1$ from a vertex in W. Since vertices of W form a clique in G'', each vertex of G'' is at distance at most r from each vertex w of W. We now show that v has a path of length at most r in G' to each vertex u in G'. If a part of a shortest path P in G from v to u goes through vertices in $V(G'') - V(G')$, this part (which has length at least one) can be replaced in G' by an edge between vertices of W without increasing the length of the path (see the proof of the previous case). Applying the induction hypothesis for G' and G'' obtains the desired result for G. $\square$

We are ready to present our algorithm for construction of a tree decomposition for a constant number of consecutive layers.

Theorem 3.5 *For $K_{3,3}$-minor-free (K_5-minor-free) graph G, we can construct a tree decomposition for $G[L[i,j]]$ of treewidth $3(j-i+1)+c_H$ in $O((j-i+1)^3 \cdot n)$ ($O((j-i+1)^3 \cdot n+n^2)$) time.*

Proof: As in the proof of Theorem 3.2, we contract the connected subgraph $G[L[0,i-1]]$ to a vertex v' and obtain another clique-sum graph G' such that $G[L[i,j]] = G'[L[1,j-i+1]]$. By Lemma 3.5, graph $G'' = G'[L[0,j-i+1]]$ is a $K_{3,3}$-minor-free (K_5-minor-free) graph and by the definition of layers each vertex in G'' is at distance at most $j-i+1$ from v'. By Theorem 3.3 (3.4), we can determine a set of clique-sum operations of graph G'' in $O(n)$ ($O(n^2)$) time.

After determining a set of clique-sum operations of $G'' = G_1 \oplus G_2 \oplus \cdots \oplus G_m$, we construct a tree decomposition for each G_i, $1 \leq i \leq m$. If G_i is a K_5 (V_8), we can easily construct a tree decomposition of width four in constant time. We now consider the case in which G_i is a planar graph. By Lemma 3.6, in each G_i, there exists a vertex v_i such that each vertex in G_i is at distance at most $j-i+1$ from v_i. It is known that if a planar graph G has a rooted spanning tree T in which the longest path has length d, then a tree decomposition of G with width at most $3d$ can be found in time $O(dn)$ [Bak94, Epp99]. Since each vertex in G_i is at distance at most $j-i+1$ from v_i, by breadth first search, we can construct a spanning tree rooted at v_i with the longest path of length at most $j-i+1$. Hence we can construct a tree decomposition for G_i of treewidth $3(j-i+1)$ in time $O((j-i+1)\cdot|V(G_i)|)$.

Having tree decompositions of G_i's, $1 \leq i \leq m$, in the rest of the algorithm, we glue together the tree decompositions of G_i's using the construction given in the proof of Lemma 3.2. To this end, we introduce an array $Nodes$ indexed by all subsets of $V(G)$ of size at most three. In this array, for each subset whose elements form a clique, we specify a node of the tree decomposition which contains this subset. We note that for each clique C in G_i, there exists a node z of $TD(G)$ such that all vertices of C appear in the bag of z [BM93]. This array is initialized as part of a preprocessing stage of the algorithm. Now, for the $\oplus$ operation between $G_1 \oplus \cdots \oplus G_h$ and G_{h+1} over the join set W, using array

$Nodes$, we find a node α in the tree decomposition of $G_1 \oplus \cdots \oplus G_h$ whose bag contains W. Since we have the tree decomposition of G_{h+1}, we can find the node α' of the tree decomposition whose bag contains W by brute force over all subsets of size at most three of bags. Simultaneously, we update array $Nodes$ by subsets of $V(G)$ which form a clique and appear in bags of the tree decomposition of G_{h+1}. Then we add an edge between α and α'. As the number of nodes in a tree decomposition of G_{h+1} is in $O(|V(G_{h+1})|)$ and each bag has size at most $3(j-i+1)$ (and thus there are at most $27(j-i+1)^3$ choices for a subset of size at most three), this operation takes $O((j-i+1)^3 \cdot |V(G_{h+1})|)$ time for G_{h+1}.

The claimed running time follows from the time required to determine a set of clique-sum operations, the time required to construct tree decompositions, the time needed for gluing tree decompositions together and the fact that $\sum_{i=1}^{m} |V(G_i)| = O(|V(G)|)$. Here we note that the only difference between the running time of the algorithm for $K_{3,3}$-minor-free graphs and that for K_5-minor-free is the time required to determine a set of clique-sum operations ($O(n)$ time for the former graphs and $O(n^2)$ time for the latter graphs). The rest of the algorithm requires linear time for both graphs.

Finally, we prove that the width of the constructed tree decomposition of G is $3(j-i+1)+4$. We use induction on m, the number of G_i's, where $G = G_1 \oplus G_2 \oplus \cdots \oplus G_m$. For $m = 1$, G_1 is either a planar graph with treewidth $3(j-i+1)$ or a graph of treewidth at most 4. In both cases the basis of induction is true. We assume the induction hypothesis is true for $m = h$, and we prove the hypothesis for $m = h+1$. Let $G' = G_1 \oplus G_2 \oplus \cdots \oplus G_h$ and $G'' = G_{h+1}$. Thus $G = G' \oplus G''$. By the induction hypothesis, treewidth of both G' and G'' is at most $3(j-i+1)+4$. The proof, for $m = h+1$, follows from this fact and Lemma 3.2. □

In the rest of this chapter, we show how the results of this section can be applied to find algorithms for clique-sum graphs, especially $K_{3,3}$-minor-free graphs and K_5-minor-free graphs.

3.2 Fixed parameter algorithms

As discussed in Section 2.5, Frick and Grohe proved that if C is a minor-closed class of graphs that has locally bounded treewidth and ϕ is a property definable in first-order logic, then there is a linear-time algorithm deciding whether a given graph $G \in C$ has property ϕ (Theorem 2.7). Since by Lemma 3.5, clique-sum graphs are closed under taking of minors and by Theorem 3.1 they have locally bounded treewidth, we conclude:

Corollary 3.2 *Any first-order logic property ϕ can be decided in linear time over clique-sum graphs.* □

Frick and Grohe's linear-time algorithm has an immense hidden constant resulting from several factors including the cost of computing tree decompositions. Bodlaender's linear-time algorithm for constructing a tree decomposition, used in Frick and Grohe's algorithm, is only of theoretical interest due to very large constants involved in the algorithm. In contrast, our practical algorithm for construction of tree decompositions helps to improve the constants for $K_{3,3}$-minor-free graphs and K_5-minor-free graphs. Therefore, algorithms for k-dominating set, k-independent set, (k,d)-circuit satisfiability and evaluating a (boolean) database query against a relational database expressed in the relational calculus have better running times when the graphs or underlying graphs under consideration are $K_{3,3}$-minor-free graphs and K_5-minor-free graphs.

The hidden constant in the linear-time algorithm of Theorem 2.7 is still large. Alber et al. [ABFN00] designed a fixed parameter algorithm for finding a k-dominating set (dominating set of size k) in planar graphs. Here, we extend their result to $K_{3,3}$-minor-free or K_5-minor-free graphs. The constant involved in this algorithm (Theorem 3.8) is very much smaller than that in the linear-time algorithm, mentioned above, and it is practical for small values of k. First, we present two preliminary theorems.

Theorem 3.6 *[ABFN00] If a tree decomposition of width w of a graph is known, then a minimum dominating set can be determined in time $O(3^w \cdot n)$, where n is the number of vertices.* □

The proof of Theorem 3.6 mainly follows the general dynamic programming approach introduced in Section 2.4. Suppose we formed layers of vertices of a graph G (see Section

3.1). The next theorem relates the number of vertices of a dominating set to the number of layers.

Theorem 3.7 *If a graph $G = (V, E)$ has a k-dominating set, then the number of layers in layering of vertices of G from any vertex $v \in V(G)$ is at most $3k$.*

Proof: The idea of the proof follows from an idea of Alber et al. [ABFN00]. We note that each vertex in the dominating set can dominate vertices from the previous, the next, or its own layer only. Hence, each vertex in the dominating set can contribute to at most three layers and hence the number of layers is at most $3k$. $\square$

Theorem 3.8 *For $K_{3,3}$-minor-free (K_5-minor-free) graphs, the problem of k-dominating set for fixed k can be solved in $O(3^{9k}n)$ ($O(3^{9k}n + n^2)$) time. Thus this problem is FPT on these graphs.*

Proof: If a graph G has a k-dominating set, the number of layers is at most $3k$ by Theorem 3.7. We can construct a tree decomposition of $G[L[0, 3k]] = G$ of width $3(3k+1)+4 = 9k+7$ in $O(n)$ ($O(n^2)$) time by Theorem 3.5. Finally, using this tree decomposition, we can solve the problem in $O(3^{9k} \cdot n)$ time by Theorem 3.6. Thus the overall running time is $O(3^{9k}n)$ ($O(3^{9k}n + n^2)$). $\square$

Alber et al. [ABFN00] proved that if a tree decomposition of width w of a graph is known, then a solution to each of variants of dominating set such as independent dominating set, total dominating set, perfect dominating set, perfect independent dominating set and total perfect dominating set can be determined in at most $O(4^w \cdot n)$ time. In addition, since a solution to each of these problems still is a dominating set for the graph, a theorem similar to Theorem 3.7 holds for each of them, *i.e.* if a graph has a solution of size k to each of these problems, then the number of layers of the graph is at most $3k$. Using these two facts we can solve these problems on $K_{3,3}$-minor-free (K_5-minor-free) graphs in $O(4^{9k}n)$ ($O(4^{9k}n + n^2)$) time (the proof is the same as the proof of Theorem 3.8).

3.3 Approximation algorithms

In this section, using Baker's approach on planar graphs, we will derive several PTASs for graph-theoretic optimization problems on $K_{3,3}$-minor-free graphs and K_5-minor-free graphs. Most problems considered in this section are hereditary maximization problems. Yannakakis has shown that for many natural hereditary properties π (see Section 2.1 for the definition), MIPS(π) is NP-complete even when the graphs under consideration are planar graphs [Yan78]. This result provides motivation to find approximation algorithms for such problems. Here we show how our results in Section 3.1 can be applied to obtain approximation algorithms for both maximization and minimization problems such as the maximum independent set problem, the minimum vertex cover problem and the minimum dominating set problem on $K_{3,3}$-minor-free graphs and K_5-minor-free graphs. In the rest of this section, parenthesized parts pertain to K_5-minor-free graphs when it is clear from context. Also by superscripts on equalities and inequalities, we mean the facts from which the equalities and inequalities are obtained.

Theorem 3.9 *Let G be a non-negative vertex-weighted $K_{3,3}$-minor-free (K_5-minor-free) graph and let $k \geq 1$ be an integer. The maximization problem WMISP(π) for a hereditary property π over G has a PTAS of ratio $1 + 1/k$ of the optimal with worst-case running time in $O(k|V| + kTime_\pi(3(k-1)+4, |V|))$ ($O(k|V|^2 + kTime_\pi(3(k-1)+4, |V|))$), where $Time_\pi(w, n)$ is the worst-case running time of WMISP(π) over an n-vertex partial w-tree whose tree decomposition is given. $Time_\pi(w, n)$ is nondecreasing as n increases.*

Proof: First we decompose graph G into several induced subgraphs, each of which having bounded treewidth, and mention some properties of these induced subgraphs. For $1 \leq i \leq k$ and $j \geq 0$, we define $L_{ij} = L[(j-1)k+i, jk+i-2]$. Here we assume a layer is empty when its level number is not between zero and the total number of layers, e.g. consider $j = 0$. We note that there is no edge between L_{ij} and $L_{i(j+1)}$. Let $\mathcal{L}_i = \bigcup_{j \geq 0} L_{ij}$ and $G_i = G[\mathcal{L}_i]$. Here every vertex appears in exactly $k-1$ of the $\mathcal{L}_i$'s or G_i's (vertices in layer L_h only do not appear in $\mathcal{L}_i$ where i is congruent to $h+1$ mod k). We label this fact by [**Fact a**].

Then, we construct a tree decomposition of width $3(k-1)+4$ for each G_i as follows. By Theorem 3.5, we can construct a tree decomposition of width $3(k-1)+4$ for $G[L_{ij}]$ in linear (quadratic) time. Since $G_i = \bigcup_{j \geq 0} G[L_{ij}]$, a tree decomposition of width $3(k-1)+4$ for G_i can be constructed by gluing tree decompositions of $G[L_{ij}]$'s together (adding edges to become one tree) in $O(|V|)$ ($O(|V|^2)$) time (note that $G[L_{ij}]$'s are disjoint).

Next, we solve the WMISP(π) on each G_i, $1 \leq i \leq k$. Since $|V(G_i)| \leq |V(G)|$, Opt_i, the maximum weighted solution of WMISP(π) over G_i, can be constructed in $Time_\pi(3(k-$

$1) + 4, |V(G)|)$.

Finally, we take Opt_m the solution with maximum weight among $Opt_1, Opt_2, \cdots, Opt_k$ as our solution for graph G, and show that it has a ratio $1 + 1/k$ of the optimal. Suppose Opt is the maximum weighted solution on graph G. We prove $\frac{weight(Opt)}{weight(Opt_m)} \leq \frac{k}{k-1}$. Because of the hereditary property of WMISP(π), we have:

$$weight(Opt \cap \mathcal{L}_i) \leq weight(Opt_i) \tag{3.5}$$

Using 3.5, we have:

$$k \cdot weight(Opt_m) \geq \sum_{i=1}^{k} weight(Opt_i) \geq^{(3.5)} \sum_{i=1}^{k} weight(Opt \cap \mathcal{L}_i) =^{[Fact\ a]} (k-1) \cdot weight(Opt).$$

The claimed running time follows immediately from the running time of constructing the tree decomposition and solving WMISP(π) for each G_i, and the number of G_i's. □

Corollary 3.3 *For $K_{3,3}$-minor-free (K_5-minor-free) graphs, there exist a PTAS of ratio $1+1/k$ of the optimal with running time $O(k \cdot 4^{3k} \cdot n)$ ($O(k \cdot 4^{3k} \cdot n + k \cdot n^2)$), for maximum independent set.*

Proof: Using dynamic programming on a tree decomposition, this problem can be solved in $O(4^w \cdot n)$ time, over each n-vertex partial w-tree whose tree decomposition is given [AP89]. Thus $Time_\pi(w, n)= O(4^w \cdot n)$ and the result follows from Theorem 3.9. □

Below we give examples that show how our result can be applied to NP-minimization problems, e.g. the minimum vertex cover problem and the minimum dominating set problem. The ideas of the proofs of Theorems 3.10 and 3.11 follow ideas of Grohe [Gro] for general graphs of locally bounded treewidth, which are in fact Baker's ideas for planar graphs. We note that the constants involved in Grohe's work are immense in contrast to those in our work.

Theorem 3.10 *For any integer $k \geq 1$, the minimum weighted vertex cover problem on $K_{3,3}$-minor-free (K_5-minor-free) graphs has a PTAS of ratio $1 + 1/k$ of the optimal with worst-case running time $O(k \cdot 8^{3k} n)$ ($O(k \cdot 8^{3k} n + k \cdot n^2)$).*

Proof: As in the proof of Theorem 3.9, we first decompose graph G into several induced subgraphs each has bounded treewidth. For $1 \leq i \leq k$ and $j \geq 0$, we define $L_{ij} = L[(j-1)k+i, jk+i]$ and $G_{ij} = G[L_{ij}]$. Here L_{ij} is slightly different from that in the proof of Theorem 3.9. The following facts are easy to observe:

[**Fact b**] Vertices of the layer L_{jk+i} appear in both $G[L_{ij}]$ and $G[L_{i(j+1)}]$ and for fixed i, each vertex appears in at most two L_{ij}'s.

[**Fact c**] For fixed i, each edge of G appears in at least one G_{ij}.

[**Fact d**] Every vertex appears in $k+1$ (successive) sets L_{ij}.

Now, by Theorem 3.5, we construct a tree decomposition of width $3(k+1)+4$ for $G[L_{ij}]$ in $O(|V(G[L_{ij}])|)$ $(O(|V(G[L_{ij}])|^2))$ time. For fixed i, since each vertex of G appears in at most two $G[L_{ij}]$'s (see [Fact b]), constructing tree decompositions of all $G[L_{ij}]$'s takes $O(|V(G)|)$ $(O(|V(G)|^2))$ time. Since $1 \leq i \leq k$ and k is a constant, the running time for constructing tree decompositions of all $G[L_{ij}]$'s is linear (quadratic).

Now, for fixed i, we wish to construct solution Opt_i for graph G over L_{ij}'s. To this end, we solve the minimum vertex cover problem for each G_{ij} to obtain a solution Opt_{ij}. Then we let

$$Opt_i = \cup_{j\geq 0} Opt_{ij} \tag{3.6}$$

First we note that for fixed i, by [Fact c] each edge of G appears in at least one G_{ij}, and thus has at least one end-vertex in Opt_i by 3.6. Hence Opt_i is a solution for the whole graph G.

Now we compute the running time to obtain each Opt_i. The minimum vertex cover problem can be solved in $O(8^w \cdot n)$ time over each n-vertex partial w-tree whose tree decomposition is given [ALS88]. Thus computing Opt_{ij} takes $O(8^{3k}|V(G[L_{ij}])|)$ time on graph G_{ij}. Thus for fixed i, by [Fact b], computing Opt_i takes $O(8^{3k}|V(G)|)$ time.

Finally, we take Opt_m the solution with minimum weight among $Opt_1, Opt_2, \cdots, Opt_k$ as our solution on graph G and show that it has a ratio $1+1/k$ of the optimal. Suppose Opt is the minimum weighted solution on graph G. We prove that $\frac{weight(Opt_m)}{weight(Opt)} \leq \frac{k+1}{k}$. Since $Opt \cap L_{ij}$ is a vertex cover for G_{ij} and Opt_{ij} is a minimum vertex cover for G_{ij}

$$weight(Opt \cap L_{ij}) \geq weight(Opt_{ij}) \tag{3.7}$$

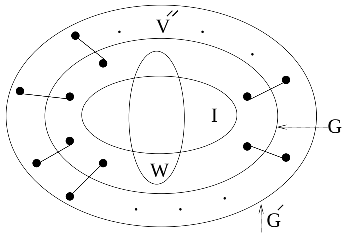

Figure 3.4: Set V'' and graphs G and G' defined in the proof of Lemma 3.7.

Using 3.6 and 3.7, we have:

$$k \cdot weight(Opt_m) \leq \sum_{i=1}^{k} weight(Opt_i) \leq^{(3.6)} \sum_{i=1}^{k} \sum_{j \geq 0} weight(Opt_{ij})$$

$$\leq^{(3.7)} \sum_{i=1}^{k} \sum_{j \geq 0} weight(Opt \cap L_{ij}) =^{[Fact\ d]} (k+1) \cdot weight(Opt).$$

The last equality follows from the fact that each vertex of Opt appears in $k+1$ L_{ij}'s (see [Fact d]).

The claimed running time follows immediately from the running time of construction of tree decompositions, the time needed to compute each Opt_i, and the number of Opt_i's. □

To find an approximation algorithm for the dominating set problem, we first introduce a generalized version of the dominating set problem (Definition 3.1) and show how we can solve this problem in linear time (Lemma 3.7). Then we use the algorithm for solving this problem to obtain a PTAS for the dominating set problem (Theorem 3.11).

Definition 3.1 *The* generalized dominating set (GDS) *problem is defined as follows. Given a vertex-weighted graph G and a set $I \subseteq V(G)$, determine a subset W of $V(G)$ of minimum weight with the property that for every $u \in I - W$ there is a $w \in W$ such that $(u, w) \in E(G)$.*

It is worth mentioning that if we set $I = V(G)$ in the GDS problem, then this problem is the same as the dominating set problem.

Lemma 3.7 *The GDS problem for given graph G and set I can be solved in time $O(3^w \cdot |V(G)|)$ when a tree decomposition of width w for G is given.*

Proof: Alber et al. proved that if a tree decomposition of width w of a non-negative vertex-weighted graph G is known, then the dominating set (DS) problem can be solved in time $O(3^w \cdot |V(G)|)$ [ABFN00]. We reduce the GDS problem to the DS problem. To solve the GDS problem on graph G, we construct graph G' on which we solve the DS problem. First we let $G' = G$ and then for each vertex $v \in V(G) - I$, we add another vertex v' with weight zero connected to v. We call this set of vertices V'' (see Figure 3.4). Suppose W is a solution to the GDS problem. We can construct a solution W' to the DS problem in graph G' by adding all vertices of V'' to W. Here each vertex of I is dominated by a vertex in W and each vertex of $V(G') - I$ is dominated by a vertex in V''. Thus W' is a dominating set for G' with the same weight of W. On the other hand, by deleting all vertices in V'' from a solution W' to the DS problem on graph G', we obtain a solution W to the GDS problem on graph G with the same weight. In fact, vertices of W' in V'' can only dominate vertices in $V(G) - I$ and thus each vertex of I is dominated by a vertex of W' which is in G, i.e. it is dominated by a vertex of W. The treewidth of G' is the same as that of G, since for $w \in V''$ connected to a vertex $v \in V(G)$ we can simply add a node whose bag contains w and v to a node of $TD(G)$ whose bag contains v. As $|V(G')| \leq 2|V(G)|$ and treewidth of G' is w, the GDS problem can be solved in $O(3^w \cdot |V(G)|)$ time using the algorithm for the DS problem. □

Theorem 3.11 *For any integer $k \geq 1$, the minimum weighted dominating set problem on $K_{3,3}$-minor-free (K_5-minor-free) graphs has a PTAS of ratio $1 + 2/k$ of the optimal with worst-case running time $O(k \cdot 3^{3k} n)$ ($O(k \cdot 3^{3k} n + k \cdot n^2)$).*

Proof: We first decompose the vertex set of G into some sets such that the subgraph induced on each set has bounded treewidth. For $1 \leq i \leq k$ and $j \geq 0$, we define $L_{ij} = L^G[(j-1)k + i - 1, jk + i]$. The following facts are easy to observe:

[**Fact g**] For fixed i, L_{ij} and $L_{i(j+1)}$ intersect only in two consecutive layers and each vertex appears in at most two L_{ij}'s.

[**Fact h**] Each vertex appears in exactly $k + 2$ (successive) sets L_{ij}.

Next, by Theorem 3.5, we construct a tree decomposition of width $3(k+2)+4$ for $G[L_{ij}]$ in $O(|V(G[L_{ij}])|)$ $(O(|V(G[L_{ij}])|^2))$ time. For fixed i, since each vertex of G appears in at most two $G[L_{ij}]$'s (see [Fact g]), constructing tree decompositions of all $G[L_{ij}]$'s takes linear (quadratic) time.

Now, for fixed i, we wish to construct solution Opt_i over all vertices in L_{ij}'s, as we did in the proofs of Theorems 3.9 and 3.10. To this end, we use the solutions to instances of the GDS problem as follows. The *interior* of each L_{ij} is defined as the set $I_{ij} = L^G[(j-1)k+i, jk+i-1]$. For $1 \le i \le k$ and $j \ge 0$, let $Opt_{ij} \subseteq L_{ij}$ be a vertex set of minimum weight with the property that

[PD] for every $u \in I_{ij} - Opt_{ij}$ there is a $v \in Opt_{ij}$ such that $(u, v) \in E(G)$.

We let

$$Opt_i = \cup_{j \ge 0} Opt_{ij} \tag{3.8}$$

By Lemma 3.7, we can obtain Opt_{ij} for graph $G[L_{ij}]$ and set I_{ij} in $O(3^{3k}|V(G[L_{ij}])|)$ time. Using the fact that for fixed i, each vertex of G appears in at most two L_{ij}'s (see [Fact g]), computing each Opt_i takes $O(3^{3k}n)$ time. In addition, by Property [PD] of Opt_{ij}'s, Opt_i is a dominating set for G (for fixed i, each vertex appears one time in an interior set I_{ij} and thus dominated by at least one vertex).

Finally, we take Opt_m the solution of minimum weight among $Opt_1, Opt_2, \cdots, Opt_k$ as our solution on graph G, and show that it has at most a ratio $1 + 2/k$ of the optimal. Suppose Opt is the minimum weight dominating set over the whole graph G. We show that $\frac{weight(Opt_m)}{weight(Opt)} \le \frac{k+2}{k} = 1 + 2/k$. We first show $Opt \cap L_{ij}$ has Property [PD] for L_{ij}. In fact, for each vertex $u \in I_{ij} \subset L_{ij}$, either $u \in Opt$ or $(u, v) \in E(G)$ where $v \in Opt$. In the latter case, v belongs to L_{ij}. Thus, in dominating set Opt, each vertex in I_{ij} is dominated by a vertex in L_{ij}. Now, since Opt_{ij} is a set of minimum weight with Property [PD], we have:

$$weight(Opt \cap L_{ij}) \ge weight(Opt_{ij}) \tag{3.9}$$

Using equations 3.8 and 3.9 and the fact that every vertex appears in exactly $k+2$ sets L_{ij} ([Fact h]), we have:

$$k \cdot weight(Opt_m) \leq \sum_{i=1}^{k} weight(Opt_i) \leq^{(3.8)} \sum_{i=1}^{k} \sum_{j \geq 0} weight(Opt_{ij})$$

$$\leq^{(3.9)} \sum_{i=1}^{k} \sum_{j \geq 0} weight(Opt \cap L_{ij}) =^{[Fact\ h]} (k+2) \cdot weight(Opt).$$

The running time follows immediately from the time needed to construct the tree decompositions, the number of Opt_i's and the time to compute each of them. □

Theorem 3.12 *For $K_{3,3}$-minor-free (K_5-minor-free) graphs, there are polynomial-time approximation algorithms whose solutions converge toward optimal as n increases for maximum independent set, minimum vertex cover and minimum dominating set.*

Proof: The running time of algorithms introduced in Corollary 3.3 and Theorems 3.10 and 3.11 is in $O(c^k n)$ ($O(c^k n + n^2)$) where k is a parameter and c is a constant. Now, by taking $k = \lceil c' \log n \rceil$, where c' is a constant, we obtain efficient polynomial-time approximation algorithms of ratio $1 + 1/(\log n)$ of the optimal (or $1 + 2/(\log n)$ for dominating set). Here $1/(\log n)$ ($2/(\log n)$) decreases as n increases. Thus the solutions converge toward optimal as n increases. □

It is worth mentioning that for several hereditary maximization problems (see Section 2.1), the function $Time_\pi(w, n)$ introduced in Theorem 3.9 is in $O(c^{p(k)} \cdot q(n))$, where c is a constant and p and q are polynomials of low degree [Bod88, TP93]. Thus, by Theorem 3.9, there are PTASs of ratio $1 + 1/k$ of the optimal for them. In addition, approaches very similar to those used in Theorems 3.9, 3.10 and 3.11 can be applied to other problems such as minimum edge dominating set, maximum triangle matching, maximum H-matching and maximum tile salvage. The full presentation of these PTASs is beyond the scope of this thesis and hence omitted. The reader is referred to papers due to Baker [Bak94] and Eppstein [Epp00] to obtain further details.

In this chapter, we introduced the class of clique-sum graphs, which contains $K_{3,3}$-minor-free graphs and K_5-minor-free graphs, and showed the graphs in the class have linear local treewidth. In addition, we presented a practical algorithm for constructing the tree decomposition of every subgraph induced on a constant number of consecutive layers in $K_{3,3}$-minor-free or K_5-minor-free graphs. Finally, we mentioned applications of our result to algorithms and PTASs for NP-hard problems on these graphs.

Chapter 4

Bounded fragmentation

In this chapter, we introduce the concept of bounded fragmentation and show how its existence might cause a network represented by a graph to be reliable. In addition, we present several examples of bounded fragmentation graphs. In the next chapter, we demonstrate the application of this property in solving subgraph isomorphism.

Definition 4.1 *A graph G is a $(k, g(k,n))$-*bounded fragmentation *graph if $|\mathcal{C}(G[V-S])| \leq |g(k,n)|$ for every $S \subseteq V(G)$ of size at most k, where g is a function of k and n. If g is independent of n, we simply write $g(k)$ instead of $g(k,n)$. A graph G is a* totally $g(k,n)$-bounded fragmentation *graph if it is a $(k, g(k,n))$-bounded fragmentation graph for all $0 \leq k \leq n$. A graph G is a* k-log-bounded fragmentation *graph (or just* log-bounded fragmentation *graph if it is clear from context) if G is a $(k, O(k \log n))$-bounded fragmentation graph. Finally a graph G is a* totally log-bounded fragmentation *graph if it is a k-*log-*bounded fragmentation graph for all $0 \leq k \leq n$.*

In this chapter, we consider the case in which the function g depends only on k and thus the number of components of $G[V-S]$ is constant when S has at most k vertices for k a constant. We mainly focus on this property in the rest of this chapter.

Connectivity can be considered as a measure of the reliability of a network. We suppose a network N is represented by an undirected graph G, in which two computers, namely nodes of the network, can communicate if and only if there is a path in G from one to the other. If G is k-connected, after removing at most $k-1$ vertices of G, the rest of G (which has $n - k + 1$ vertices) is still connected. This means that if at most $k - 1$ nodes of the network N fail, the rest of the nodes of the network can communicate with each other.

Bounded fragmentation also can play a role in the reliability of a network. If G is a $(k, g(k))$-bounded fragmentation graph, after removing at most k vertices we have at least one component which has $\Omega(n)$ vertices. The reason is that after removing at most k vertices the rest of the nodes fall into at most a constant number of connected components $(g(k))$ and thus one component has at least $\Omega(n)$ vertices. Thus, after the failure of at most $k - 1$ nodes of N, $\Omega(n)$ nodes in the rest of N (and not necessarily $n - k$) still can communicate with each other. Using these facts, bounded fragmentation can be considered as a generalization of connectivity.

Bounded fragmentation also can have another application in the reliability of a network. Suppose that we need to repair the network N temporarily by adding several links between the current nodes of the network (not by adding any new node because of its high cost) when the number of failing nodes in the network is at most constant k . If G is a $(k, g(k))$-bounded fragmentation graph, then we can simply repair the network by adding at most $g(k) - 1$ numbers of links, which is constant. Here, after removing the failing nodes, we find the connected components of G in $O(|V(G)|)$ time. Then we can connect these at most $g(k)$ connected components in the form of a tree, by adding at most $g(k) - 1$ edges among them. These two simultaneous properties of bounded fragmentation graphs cause their corresponding networks to be more reliable and robust.

In this chapter, first we introduce some classes and properties which cause a graph G to be a bounded fragmentation graph (Section 4.1), and then we consider the number of edges of a bounded fragmentation graph (Section 4.2).

4.1 Bounded fragmentation graphs

In this section, we focus on examples of bounded fragmentation graphs.

Lemma 4.1 *Connected graphs with constant maximum degree c are totally ck-bounded fragmentation graphs.*

Proof: The proof follows from the fact that if $\Delta(G) = c$, after removing any k vertices, $0 \leq k \leq n$, the number of connected components is at most $g(k) = ck$. □

Theorem 4.1 *If graph G has a maximum independent set of constant size c, then it is a totally c-bounded fragmentation graph.*

Proof: For any set $S \subseteq V(G)$ of size k, $0 \leq k \leq n$, at least one vertex from each connected component of $G[V - S]$ is contained in any maximum independent set. Since the size of the maximum independent set is bounded above by c, the number of connected components is bounded above by c, as well. Thus G is a totally c-bounded fragmentation graph. $\square$

In fact, we can generalize the approach used in Theorem 4.1 to other maximization problems.

The proof of the following lemma is trivial and hence omitted.

Lemma 4.2 *Let G be a graph with minimum degree $\delta(G) \geq k + h - 1$ for two positive integers k and h. Removing any set S of size at most k can not produce any component with size less than h.* $\square$

Theorem 4.2 *Let P be a hereditary maximization problem which has a non-zero solution on every connected graph of size at least h, where h is a non-negative constant. We also assume P is additive on components. For any non-negative integer k, if P on a graph G has a maximum solution of constant size c and $\delta(G) \geq k+h-1$ then G is a (k, c)-bounded fragmentation graph.*

Proof: By Lemma 4.2, we know that removing any set of size at most k can not generate any connected component with size less than h. Using our assumption, P has a non-zero solution in each component. The number of connected components is at most c, since otherwise using the maximum solution of each component, we can construct a maximal solution of the whole graph which is of size greater than c. $\square$

For example, the maximum matching problem is a hereditary problem which has a non-zero solution on every connected graph of at least two vertices.

Corollary 4.1 *For any non-negative integer k, if connected graph G has a maximum matching of constant size c and minimum degree at least $k + 1$, i.e. $\delta(G) \geq k + 1$, then it is a (k, c)-bounded fragmentation graph.* $\square$

Example 4.1 *A complete bipartite graph $K_{n-k-1,k+1}$, where $n \geq 2k+2$, has minimum degree $k+1$ and a maximum matching of size $k+1$. Hence it is a $(k, k+1)$-bounded fragmentation graph.*

The result of Theorem 4.2 can be generalized to other problems which are not necessarily hereditary.

Definition 4.2 Covering a graph by at most m vertex-disjoint paths *means the vertices of a graph can be partitioned into m subsets such that for each set S, there exists a path in a graph that contains exactly the vertices in S.*

Lemma 4.3 *Graphs whose vertices can be covered by at most c vertex-disjoint paths are totally $(k+c)$-bounded fragmentation graphs.*

Proof: The removal of a vertex from a path splits the path into at most two sub-paths and thus at most two connected components. Thus, removing any k vertices, $0 \leq k \leq n$, can add at most k connected components. Thus, we have at most $k+c$ connected components. □

Example 4.2 *Consider a Hamiltonian graph F_n which is constructed from a path of length n by connecting one of its vertices to all its non-neighbors. Since vertices of every Hamiltonian graph can be covered by one path, F_n is a totally $(k+1)$-bounded fragmentation graph.*

Eppstein without introducing bounded fragmentation implicitly relates it to other properties of graphs. He proved that a planar 3-connected graph is a totally $O(k)$-bounded fragmentation graph (Lemma 7 [Epp99]).

Clearly, a complete graph K_n is a totally (1)-bounded fragmentation graph. Intuitively, graphs with large minimum degree are bounded fragmentation graphs. In Theorem 4.3, we derive an exact bound on the minimum degree of a graph that guarantees the graph to be a bounded fragmentation graph.

Lemma 4.4 *[Wes96] Let G be a simple n-vertex graph such that for two non-negative integers h and d, $n \geq h+d$ and $\delta(G) \geq \frac{n+d(h-2)}{d+1}$. If $G-S$ has more than d components, then $|S| \geq h$. The bound is tight: there exists a graph with $\delta(G) = \lfloor \frac{n+d(h-2)-1}{d+1} \rfloor$ such that $G-S$ with $|S| < h$ has more than d components.* □

Theorem 4.3 *For each constant d, graphs with $\delta(G) \geq \frac{n+d(k-1)}{d+1}$ are (k,d)-bounded fragmentation graphs where $0 \leq k \leq n-d-1$.*

Proof: By Lemma 4.4, for $h = k+1$, after removing any set S with $|S| \leq h-1 = k$ the graph G has at most d components where $n \geq h+d = k+1+d$. Thus it is a (k,d)-bounded fragmentation graph. □

Some of the above results can be generalized to the case in which the function g is a function of both n and k. We present here two lemmas which are used in Chapter 4. The proofs of these lemmas are very similar to the proofs of Lemmas 4.1 and 4.3 and hence omitted.

Lemma 4.5 *Connected graphs with maximum degree $O(\log n)$ are totally* log*-bounded fragmentation graphs.* □

Lemma 4.6 *Graphs whose vertices can be covered by at most $O(\log n)$ vertex-disjoint paths are totally* log*-bounded fragmentation graphs.* □

4.2 Numbers of edges of bounded fragmentation graphs

As discussed before, bounded fragmentation is a measure in reliability of a network (at least in theory). However, in network design, it is beneficial to have a linear number of communication lines. Thus, an interesting question is whether it is possible to have a linear number of edges and still a graph of bounded fragmentation. The answer to this question is affirmative. Clearly, graphs with constant maximum degree and planar graphs have linear numbers of edges. As shown in Examples 4.1 and 4.2, graphs with maximum matchings of constant size or graphs coverable by a constant number of vertex-disjoint paths can also have a linear number of edges.

However, the condition stated in Theorem 4.3 is valid only for graphs with quadratic numbers of edges. Graphs with constant maximum independent sets have quadratic numbers of edges. The proof follows from the fact that if a graph G has a constant maximum independent set c, its complement $\bar{G}$ has a constant maximum clique c. By Turán's theorem [Tur41], $\bar{G}$ has at most $(1-1/(c-1))n^2/2$ edges. Thus G has a quadratic number of edges.

In this chapter, we introduced applications of bounded fragmentation graphs for networking and mentioned several instances of bounded fragmentation graphs. In Chapter 5, we state a relation between bounded fragmentation and the subgraph isomorphism problem.

Chapter 5

Subgraph isomorphism for graphs of *log*-bounded fragmentation

As mentioned in Section 2.8, the subgraph isomorphism problem can be solved in polynomial time when the source graph has bounded degree and the host graph has bounded treewidth. In this chapter, we extend this result to cover bounded fragmentation (see Chapter 4). In addition, in this chapter, we solve the subgraph isomorphism problem when the set of inputs is extended to graphs of locally bounded treewidth.

Matoušek and Thomas [MT92] proved the following theorem for subgraph isomorphism of bounded degree graphs:

Theorem 5.1 *(Theorem 5.14 [MT92]) Suppose graph G is connected, $\Delta(G) \leq c$ for constant c, and H is a partial k-tree. There are $O(|V(G)|^{k+1} \cdot |V(H)|)$-time algorithms which solve isomorphism and subgraph and induced subgraph versions of this problem.* □

We will prove the following extended version of Theorem 5.1, which includes wider classes of graphs than bounded degree graphs.

Theorem 5.2 *Suppose G is a $(k+1, g(k+1, n))$-bounded fragmentation graph and H is a partial k-tree for $k \geq 2$. There are $O(g(k+1, n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1} \cdot |V(H)|)$-time algorithms which solve isomorphism and subgraph and induced subgraph versions of this problem.*

Corollary 5.1 *Testing graph isomorphism and its subgraph or induced subgraph versions have polynomial-time solutions when graph H has bounded treewidth and graph G is a* log-*bounded fragmentation graph.* □

Using Corollary 5.1, we obtain two new results. First, by Lemma 4.5, if the maximum degree of the source graph is bounded by $O(\log n)$ (and not necessarily a constant), then it is a totally log-bounded fragmentation graph and hence the problems can be solved in polynomial time. Second, the subgraph isomorphism problem can be solved in polynomial time for graphs other than bounded degree partial k-trees or k-connected partial k-trees. To justify this result, we consider the graph F_n (Example 4.2) from the class of bounded fragmentation graphs. The maximum degree of this graph is $n-1$, and its treewidth is two. If the source graph is F_n and the host graph is an arbitrary graph of bounded treewidth, then the current results can not be applied to test subgraph isomorphism for these graphs. However, by Corollary 5.1, we can solve the problem for these graphs in polynomial time. In addition, log-bounded fragmentation graphs introduced in Lemma 4.6 are not necessarily connected, but still we can solve the problems for them.

We note that there are properties other than those introduced in Lemmas 4.5 and 4.6 which guarantee a graph to be a bounded fragmentation graph, but they do not apply to partial k-trees. For example, as we showed in Chapter 4, graphs with a certain minimum degree are bounded fragmentation graphs, but this minimum degree is valid only for graphs with quadratic numbers of edges. Since all partial k-trees have linear numbers of edges [Ros74], the class of graphs to which the result applies is empty.

We now prove Theorem 5.2. First, we present our proof for the induced subgraph isomorphism problem. Then, we explain how our proof can be applied to the subgraph isomorphism problem. We note that if $|V(G)| = |V(H)|$, then the induced subgraph isomorphism problem is identical to the graph isomorphism problem. The proof of this theorem follows ideas of Matoušek and Thomas [MT92]. In fact, the main idea here is the standard dynamic programming on a tree decomposition introduced by Arnborg and Proskurowski and presented in Section 2.4. We use notions such as *solution*, *partial solution*, *characteristic* and *full set of characteristics* introduced in that section.

As mentioned in Section 2.4, Bodlaender proved that for every graph H of treewidth at most k, a nice tree decomposition of width k can be constructed in linear time [Bod98].

Using this result, we construct a nice tree decomposition of H in $O(|V(H)|)$ time and in the rest of this chapter, we assume that the given tree decomposition of H ($TD(H)$) is a nice tree decomposition.

The solution introduced in Section 2.4 for the induced subgraph isomorphism problem is an isomorphism ϕ from G into H. To define a partial solution, we consider the possible structure of an isomorphism ϕ restricted to $H_{[z]}$ for a node z of $TD(H)$. Intuitively, this restriction maps a subgraph G' of G into $H_{[z]}$; the vertices of G' are those vertices of G whose images are in $V(H_{[z]})$, and the edges of G' are edges of G images of whose end-vertices are adjacent in $H_{[z]}$. This mapping is an isomorphism φ from G' into $H_{[z]}$ such that $\varphi(v) = \phi(v)$ for $v \in G'$, and we call it a *partial isomorphism*.

Here we show that the subgraph G' has a special structure and can not be an arbitrary subgraph of G. To this end, we introduce the vertex set and the edge set of G'. Again, we consider the isomorphism ϕ from G into H from which the partial isomorphism φ is obtained. We suppose $S = \{v \in V(G) | \phi(v) \in \chi_z\}$ and $\mathcal{C}(G[V - S]) = \{C_1, \cdots, C_h\}$. Since each graph isomorphic to a connected graph is connected, $\phi(V(C_i))$, $1 \leq i \leq h$, is a connected subgraph of H. As χ_z is a separator for H (Lemma 2.2), and $\phi(V(C_i))$ is a connected subgraph of H which does not intersect χ_z, each $\phi(V(C_i))$ is completely inside of $H_{[z]}$ or completely outside of $H_{[z]}$. Let $\mathcal{D} = \{D_1, \cdots, D_l\}$ be those components of $\mathcal{C}(G[V - S])$ whose images are completely inside $H_{[z]}$. The set S and components D_i, $1 \leq i \leq l$ are shown in Figure 5.1. In fact, $S \cup V(\mathcal{D})$ is the vertex set of the subgraph G' introduced above. Thus $V(G')$ is always the union of the set $S \subseteq V(G)$ and vertex sets of a number of components of $\mathcal{C}(G[V - S])$. We now consider the edge set of G'. Since $\phi(S \cup V(\mathcal{D}))$ is a subset of $V(H_{[z]})$ and ϕ is an isomorphism form G into H, for all $u, v \in S \cup V(\mathcal{D})$ such that $\{u, v\} \in E(G)$, $\varphi(u) = \phi(u)$ and $\varphi(v) = \phi(v)$ are adjacent in $H_{[z]}$ (see Figure 5.1). Therefore G' is always an induced subgraph of G over vertices of $S \cup V(\mathcal{D})$ for $S \subseteq V(G)$ and $\mathcal{D} \subseteq \mathcal{C}(G[V - S])$ and our partial solutions are partial isomorphisms from this kind of subgraph of G into $H_{[z]}$. By this definition of a partial solution, it is the restriction of a solution to $H_{[z]}$ and naturally can be extended to a solution (see Step 3 in Bodlaender's algorithm).

We are now ready to define the most important notion, namely a characteristic of a partial solution with respect to a node z of $TD(H)$. We define the crucial part of a partial solution which is necessary to know how a partial solution can be extended to a solution. Here, we label arguments by properties in upcoming Definition 5.2. First we represent vertices and edges of G'. We specify $V(G')$ by S and $\mathcal{D}$. Two sets S and $\mathcal{D}$ also uniquely determine edges of G' (Property [Pc]). For each vertex $v \in V(\mathcal{D})$, $\varphi(v) = \phi(v)$ is inside of $V(H_{[z]}) - \chi_z$ and all neighbors of v in G are in $S \cup V(\mathcal{D})$. Since images of v and all its neighbors are present in $H_{[z]}$, intuitively, this vertex v is not a crucial vertex of G' and there is no need to know how its image is exactly mapped in $H_{[z]}$. Instead, we need to know more information about images of vertices of S. Thus we maintain a mapping $\psi(v) = \varphi(v)$ for $v \in S$ (Property [Pb]). We also note that since $\phi(v)$ is an isomorphism and $\varphi(v)$ is obtained from $\phi(v)$, $\varphi(u) \neq \varphi(v)$ for all $u, v \in S \cup V(\mathcal{D})$ (Property [Pa]). As we will formally show later, triples $(S, \mathcal{D}, \psi)$, called iso-triples, are our characteristics of partial solutions holding all the information that we maintain about our partial solutions (see Figure 5.1). We note that not each triple $(S, \mathcal{D}, \psi)$ is necessarily a characteristic. For example, in a triple $(S, \mathcal{D}, \psi)$ relative to a leaf z, if $\mathcal{D}$ is not empty, then this triple is not a characteristic, since for z, $H_{[z]} = H[\chi_z]$ and $\mathcal{D} = \emptyset$. We call a partial isomorphism φ from which a characteristic $(S, \mathcal{D}, \psi)$ is obtained an *extension* of this characteristic. We note that the characteristic $(S, \mathcal{D}, \psi)$ might be obtained from several partial isomorphisms and thus have several extensions. Let us define all these terms formally.

Definition 5.1 *An* iso-triple ξ of G into H relative to a node z of $TD(H)$ *is a triple* $(S, \mathcal{D}, \psi)$ *where:*

1. $S \subseteq V(G)$;
2. $\mathcal{D} \subseteq \mathcal{C}(G[V - S])$; *and*
3. ψ *is a one-to-one mapping from* S *into* χ_z.

Definition 5.2 *An* extension φ of an iso-triple $\xi = (S, \mathcal{D}, \psi)$ relative to a node z of $TD(H)$ *is a mapping* φ *from* $S \cup V(\mathcal{D})$ *into* $H_{[z]}$ *with these properties:*

[Pa] $\varphi(u) \neq \varphi(v)$ *for all* $u, v \in S \cup V(\mathcal{D})$;

[Pb] $\varphi(v) = \psi(v)$ *for* $v \in S$ *and* $\varphi(v) \notin \chi_z$ *for* $v \in V(\mathcal{D})$; *and*

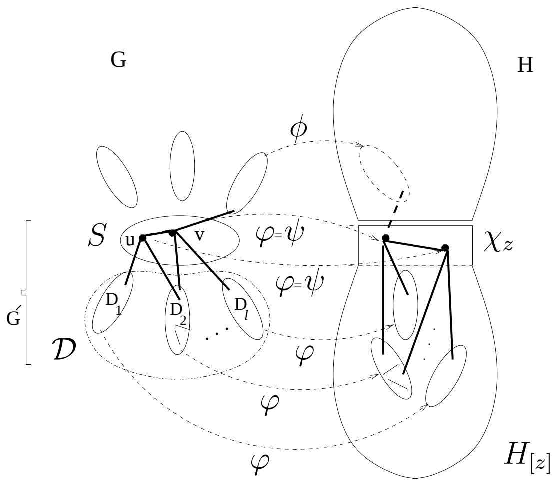

Figure 5.1: Elements of the characteristic of a partial isomorphism

[Pc] *for each* $u, v \in S \cup V(\mathcal{D})$*,* $\{u, v\} \in E(G)$ *if and only if* $\{\varphi(u), \varphi(v)\} \in E(H)$*.*

We note that all conditions in Definition 5.2 are formal descriptions of properties of an extension (or a partial isomorphism) corresponding to an iso-triple (if it exists).

Definition 5.3 *A* characteristic of a partial isomorphism (CPI) ξ of G into H relative to a node z of $TD(H)$ *is an iso-triple* $(S, \mathcal{D}, \psi)$ *which has an extension* φ *(not necessarily unique).*

According to the general dynamic programming approach, we need to identify the full set of characteristics which contains all CPIs relative to a node z of $TD(H)$. This set can be represented by an array indexed by all iso-triples $(S, \mathcal{D}, \psi)$, namely a *full set array.* If an iso-triple $(S, \mathcal{D}, \psi)$ is a CPI its corresponding element in the array is **true**, otherwise it is **false**. Later, we will show how the full set of a node can be built from the full set of its children (if they exist). Now, we show that the size of a full set is polynomial.

Lemma 5.1 *The number of all iso-triples and the number of CPIs relative to a node* z *of* $TD(H)$ *is in* $O(2^{g(k+1,n)} \cdot |V(G)|^{k+1})$*.*

Proof: Since the number of CPIs is bounded above by the number of iso-triples, it suffices to bound the number of iso-triples. By the definition of iso-triples (Point 3, Definition 5.1), for $v \in S$, $\psi(v)$ is an element of set χ_z. Since $|\chi_z| \leq k+1$ and the images of ψ are different for $u \neq v$ (Point 3, Definition 5.1), $|S| \leq k+1$ and hence there are $|V(G)|^{k+1}$ different ways to choose S. We have at most $2^{g(k+1,n)}$ choices for $\mathcal{D}$, because after choosing S, each connected component of $G[V-S]$ either belongs to $\mathcal{D}$ or does not (Point 2, Definition 5.1). Since $\psi(v)$ is a one-to-one mapping from S into χ_z, we have at most $|\chi_z|! \leq (k+1)!$ ways of constructing ψ ($|\chi_z|$ choices for the image of the first vertex of S, $(|\chi_z|-1)$ choices for the second one and so on). By multiplying all factors described above, we have in total at most

$$|V(G)|^{k+1} \cdot 2^{g(k+1,n)} \cdot (k+1)!$$

choices of iso-triples. Since k is a constant, the above number is in $O(2^{g(k+1,n)}|V(G)|^{k+1})$. $\square$

Finally, we state how the problem can be solved efficiently, if we know the full set of CPIs relative to the root r of $TD(H)$.

Definition 5.4 *A* complete CPI ξ of G into H relative to a node z of $TD(H)$ *is a CPI $(S, \mathcal{D}, \psi)$ such that $\mathcal{D} = \mathcal{C}(G[V-S])$ (S can be empty).*

Lemma 5.2 *There exists an induced subgraph isomorphism ϕ from G into H if and only if there exists a complete CPI ξ of G into H relative to the root r of $TD(H)$.*

Proof: If an isomorphism ϕ from G into H exists, we can construct a complete CPI by restriction of ϕ to χ_r. Formally, we let $S = \{v|\phi(v) \in \chi_r\}$; $\mathcal{D} = \mathcal{C}(G[V-S])$; and $\psi(v) = \phi(v)$ for $v \in S$. We can observe that $\varphi = \phi$ is an extension for $\xi = (S, \mathcal{D}, \psi)$. More precisely, Property [Pa] of φ follows from the fact that ϕ is a one-to-one mapping. Property [Pb] follows from definitions of S and ψ and finally Property [Pc] follows from the fact that $\varphi = \phi$ is an isomorphism. On the other hand, if a complete CPI ξ of G into H relative to the root r of $TD(H)$ exists, then the extension φ of this CPI is an isomorphism ϕ from G into H. In fact, Property [Pa] of an extension guarantees ϕ to be a one-to-one mapping and Property [Pc] of an extension together the fact that $\mathcal{D} = \mathcal{C}(G[V-S])$ guarantees ϕ to be an isomorphism from $H_{[r]} = G$ into H. $\square$

It only remains to show how full set arrays of nodes of $TD(H)$ can be filled in. As $H_{[z]} = H[\chi_z]$ for a leaf z and hence $\mathcal{D}$ is empty, ψ is equal to its extension φ. Thus for each iso-triple ξ relative to a leaf, by brute force, we can easily check whether ψ is an extension of ξ or not and construct the full set array (see Algorithm A for further detail). For other nodes, we use Lemmas 5.3 and 5.4 which are similar to Lemmas 5.10 and 5.11 of Matoušek and Thomas's paper [MT92].

First, we consider how the full set array of a separator node z (see Section 2.4 for the definition) can be constructed from the full set array of its child z'.

Definition 5.5 *Suppose an iso-triple $\xi = (S, \mathcal{D}, \psi)$ relative to a separator node z of $TD(H)$ and an iso-triple $\xi' = (S', \mathcal{D}', \psi')$ relative to the child z' of z satisfy following conditions:*

1. $S = \{v \in S' | \psi'(v) \in \chi_z\}$;
2. $\mathcal{D}' = \{D' \in \mathcal{C}(G[V - S']) | D'$ *is a subgraph of some* $D \in \mathcal{D}\}$; *and*
3. $\psi(v) = \psi'(v)$ *for* $v \in S$;

Then, we say ξ is separator-consistent *with ξ'.*

The conditions stated in Definition 5.5 indicate how elements of a CPI of a separator node, *i.e.* $S, \mathcal{D}$ and ψ, are related to elements of a CPI of the child of this node. We show later (in Algorithm A on page 65) how we can use these conditions for obtaining a CPI of a separator node z from a CPI of its child z'.

Lemma 5.3 *There exists a CPI $\xi = (S, \mathcal{D}, \psi)$ relative to a separator node z of $TD(H)$ if and only if there exists a CPI $\xi' = (S', \mathcal{D}', \psi')$ relative to the child z' of z such that ξ and ξ' are separator-consistent.*

Proof: The idea of the proof follows an idea of Matoušek and Thomas (Lemma 5.10 [MT92]). Since z is a separator node, we have $\chi_z \subseteq \chi_{z'}$ and $H_{[z]} = H_{[z']}$.

First, we prove if there exists a CPI ξ relative to z then there exists a CPI ξ' relative to z' such that ξ and ξ' are separator-consistent. To this end, we extend ξ to its extension φ over $H_{[z]}$. Then we obtain ξ' by restriction of φ to $\chi_{z'}$ and show that ξ' satisfies all conditions of separator-consistency. Formally, we define $S' = \{v|\varphi(v) \in \chi_{z'}\}$; $\mathcal{D}' = \{D' \in \mathcal{C}(G[V - S'])|D'$ is a subgraph of some $D \in \mathcal{D}\}$; and $\psi'(v) = \varphi(v)$ for $v \in S'$. Using this definition of ξ', we observe that $\varphi' = \varphi$ is an extension of ξ'. First we note that by our definitions of S' and $\mathcal{D}'$, $S \cup V(D) = S' \cup V(D')$. Properties [Pa] and [Pc] of the extension φ' for ξ' follow from this fact and Properties [Pa] and [Pc] of φ for ξ. Property [Pb] of φ' follows from the definition of ψ'.

We show that all conditions of separator-consistency have been satisfied. By the definition of ψ', we have $\psi'(w) = \varphi(w)$ for $w \in S' \supseteq S$. Since by Property[Pb] of φ, $\varphi(w) = \psi(w)$ for $w \in S$ and $\chi_z \subseteq \chi_{z'}$ we have $\psi'(w) = \varphi(w) = \psi(w)$ for $w \in S$. Condition (3) is satisfied by this fact. By the definition of S' and the fact that $\psi'(w) = \varphi(w)$ for $w \in S'$, Condition (1) is also satisfied. Finally, Condition (2) is satisfied by the definition of D'.

We now prove if there exists a CPI ξ' relative to z' such that an iso-triple ξ is separator-consistent with ξ', then ξ is a CPI relative to z. We suppose there is a ξ' which satisfies the above conditions. We extend ξ' to its extension φ' over $H_{[z]}$ and show that φ' is an extension of ξ too, that is, the mapping $\varphi = \varphi'$ satisfies all properties of an extension for ξ. Since $S \subseteq S'$ (see Condition (1)), by Condition (2) we have $S \cup V(D) = S' \cup V(D')$. Thus Properties [Pa] and [Pc] of φ' follow from Property [Pa] and [Pc] of φ. By Property [Pb] of φ', we have $\varphi(v) = \varphi'(v) \notin \chi_{z'}$ for $v \in V(G) - S'$, and since $\chi_z \subseteq \chi_{z'}$, we have $\varphi(v) \notin \chi_z$ for $v \in V(G) - S'$. For $v \in S' - S$, $\varphi(v) = \varphi'(v) = \psi'(v) \notin \chi_z$ by Condition (1). By Condition (3), $\varphi(v) = \varphi'(v) = \psi'(v) = \psi(v)$ for $v \in S$. Thus Property [Pb] of φ holds. □

Finally, we consider how the full set array for a join node z (see Section 2.4 for the definition) can be built from the full set arrays of its children z_1 and z_2.

Definition 5.6 *Let z be a join node of $TD(H)$ and its children be z_1 and z_2. Suppose an iso-triple ξ relative to z and iso-triples ξ_1 and ξ_2 relative to z_1 and z_2 satisfy following conditions:*

1. *$S_i = \{v \in S|\psi(v) \in \chi_{z_i}\}$ for $i = 1, 2$;*

2. *the components of $\mathcal{D}$ are partitioned into $\mathcal{D}_1$ and $\mathcal{D}_2$;*

3. *$\psi_i(v) = \psi(v)$ for $v \in S_i$ for $i = 1, 2$; and*

4. for $u, v \in S$, $\{u, v\} \in E(G)$ if and only if $\{\psi(u), \psi(v)\} \in E(H)$

Then, we say ξ is join-consistent *with ξ_1 and ξ_2.*

The conditions stated in Definition 5.6 specify how elements of a CPI of a join node are related to elements of CPIs of its children. We demonstrate in Algorithm A how these conditions can be used for obtaining a CPI of a join node z from CPIs of its children z_1 and z_2.

Lemma 5.4 *Let z be a join node of $TD(H)$ and its children be z_1 and z_2. There exists a CPI $\xi = (S, \mathcal{D}, \psi)$ relative to z if and only if there exist CPIs $\xi_i = (S_i, \mathcal{D}_i, \psi_i)$ relative to node z_i $(i = 1, 2)$ such that ξ is join-consistent with ξ_1 and ξ_2.*

Proof: Throughout this proof, we assume $i = 1, 2$. Since z is a join node, we know $\chi_{z_i} \subseteq \chi_z$ and $H_{[z_i]}$ is an induced subgraph of $H_{[z]}$.

We prove that if there exists a CPI ξ relative to z, then there exist CPIs ξ_i relative to z_i such that ξ is join-consistent with ξ_1 and ξ_2. We suppose φ is an extension of ξ over χ_z. We obtain ξ_1 and ξ_2 from φ and show that they satisfy conditions of join-consistency.

We first state the following claim about elements of $\mathcal{D}$.

Claim 1 *Let $S_i = \{v \in S | \varphi(v) \in \chi_{z_i}\}$. For each $D \in \mathcal{D}$, either $\varphi(V(D)) \subseteq V(H_{[z_1]}) - \chi_{z_1}$ and $D \in \mathcal{C}(G[V - S_1])$ or $\varphi(V(D)) \subseteq V(H_{[z_2]}) - \chi_{z_2}$ and $D \in \mathcal{C}(G[V - S_2])$.*

Proof: Since by Property [Pb] of φ, $\varphi(V(D))$ is contained in $H_{[z]}$ but is disjoint from χ_z, $\chi_{z_i} \subseteq \chi_z$ and χ_z is a separator of $H_{[z]}$ (Lemma 2.2), either $\varphi(V(D)) \subseteq V(H_{[z_1]}) - \chi_{z_1}$ or $\varphi(V(D)) \subseteq V(H_{[z_2]}) - \chi_{z_2}$. Without loss of generality, we can assume the former case holds. We show $D \in \mathcal{C}(G[V - S_1])$. First we note that since $D \in \mathcal{D} \subseteq \mathcal{C}(G[V - S])$, all outgoing edges from $V(D)$ go into S. We now prove there is no edge between $V(D)$ and $S - S_1$, thus all outgoing edges from $V(D)$ go into S_1, and hence D is a component of $G[V - S_1]$. If there are $v \in V(D)$ and $w \in S - S_1$ such that $\{v, w\} \in E(G)$, then there exists an edge $\{\varphi(v), \varphi(w)\} \in E(H)$ where $\varphi(v) \in V(H_{[z_1]}) - \chi_{z_1}$ (by our assumption about D). By the definition of S_1, $\varphi(w) \notin \chi_{z_1}$. Since $\varphi(w)$ appears in χ_z and not χ_{z_1} and each vertex appears in bags of nodes of a connected subtree of a tree decomposition (Property (3) of tree decompositions), $\varphi(w) \notin V(H_{[z_1]})$. Thus $\varphi(v) \in V(H_{[z_1]}) - \chi_{z_1}$ and

$\varphi(w) \in V(H) - V(H_{[z_1]})$. But since χ_{z_1} is a separator (Lemma 2.2), there is no such edge $\{\varphi(v), \varphi(w)\} \in E(H)$. Using this contradiction, we finish the proof of our claim. $\square$

We now construct CPIs ξ_1 and ξ_2. First we define $\mathcal{D}_i = \{D \in \mathcal{D} | \varphi(V(D)) \in V(H_{[z_i]}) - \chi_{z_i}\}$. Using our claim, $\mathcal{D}_i \subseteq \mathcal{C}(G[V - S_i])$ and $\mathcal{D}_1$ and $\mathcal{D}_2$ partition $\mathcal{D}$. We now define mapping $\varphi_i(v) = \varphi(v)$ for $v \in S_i \cup V(\mathcal{D}_i)$ and mapping $\psi_i(v) = \varphi_i(v)$ for $v \in S_i$. We show that φ_i is an extension of iso-triple $\xi_i = (S_i, \mathcal{D}_i, \psi_i)$ and thus ξ_i is a CPI. . By the definition of S_i, $S_i \subseteq S$, and by the definition of D_i, $V(\mathcal{D}_i) \subseteq V(\mathcal{D})$, and thus $S_i \cup V(\mathcal{D}_i) \subseteq S \cup V(\mathcal{D})$. Using this fact, Properties [Pa] and [Pc] of φ_i follow from Properties [Pa] and [Pc] of φ. Property [Pb] of φ_i holds by definitions of S_i and ψ_i and the fact that $\varphi(v) \notin \chi_z$ for $v \in V(\mathcal{D})$ (by Property [Pb] for φ).

We prove that all conditions of join-consistency are satisfied. Let $v \in S_i$. By the definition of ψ_i, $\psi_i(v) = \varphi_i(v)$ for $v \in S_i$. By the definition of φ_i, we have $\varphi_i(v) = \varphi(v)$ and by Property [Pb] of φ, $\varphi(v) = \psi(v)$. Thus $\psi_i(v) = \psi(v)$ for $v \in S_i$ and Condition (3) is satisfied. By the definition of S_i and the fact that $\varphi_i(v) = \varphi(v) = \psi(v)$ for $v \in S$, Condition (1) is satisfied. Condition (4) follows from Property [Pc] of φ and the fact that $\varphi(v) = \psi(v)$ for $v \in S$. Condition (2) follows from the definition of $\mathcal{D}_i$ and our claim.

We now prove if there exist CPIs ξ_i relative to z_i such that an iso-triple ξ is join-consistent with ξ_1 and ξ_2, then ξ is a CPI relative to z. We define a mapping φ for iso-triple ξ from ψ and extensions φ_i's of ψ_i's as follows: for $v \in V(\mathcal{D}_i)$, $\varphi(v) = \varphi_i(v)$ and for $v \in S$, $\varphi(v) = \psi(v)$. We note that each vertex $v \in V(\mathcal{D})$ is either in $V(\mathcal{D}_1)$ or in $V(\mathcal{D}_2)$ (Condition (2)). We now show that φ is an extension of ξ. By Property [Pa] of φ_i, $\varphi(u) \neq \varphi(v)$ for $u, v \in V(\mathcal{D}_i)$. Since ψ is a one-to-one mapping, $\varphi(u) \neq \varphi(v)$ for $u, v \in S$. For $u \in S$ and $v \in V(D_i)$, since $\psi(v) \in \chi_z$ and $\varphi(u) \notin \chi_z$, $\varphi(u) \neq \varphi(v)$. Thus Property [Pa] of φ holds. Since by the definition of φ, $\varphi(v) = \psi(v) \in \chi_z$ for $v \in S$, and $\varphi(v) = \varphi_i(v) \notin \chi_z$ for $v \in V(\mathcal{D}) = V(\mathcal{D}_1) \cup V(\mathcal{D}_2)$, Property [Pb] of φ holds. We now show Property [Pc] of φ holds. By Condition (4) of join-consistency, for $u, v \in S$, $\{u, v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(H)$. By Property [Pc] of φ_1 and φ_2, $\{u, v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(H)$ for $u, v \in V(\mathcal{D}) = V(\mathcal{D}_1) \cup V(\mathcal{D}_2)$. In the last case in which $u \in S$ and $v \in V(D)$, where $D \in \mathcal{D}$, by Condition (2) of join-consistency, either $D \in \mathcal{D}_1$ or $D \in \mathcal{D}_2$. Without loss of generality, we assume $D \in \mathcal{D}_1$. Since $\mathcal{D}_1 \subseteq \mathcal{C}(G[V - S_1])$ (by the definition of $\mathcal{D}_1$), D is a component of $G[V - S_1]$, and thus all outgoing edges from

$V(D)$ go into S_1. Thus $u \in S_1$ and by Property [Pc] of φ_1, $\{u, v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(H)$. Hence Property [Pc] of φ holds. $\square$

We are now ready to present our final algorithm. Lines 1-14 of the algorithm are preprocessing steps. In this algorithm, first we construct $\mathcal{C}(G[V - S])$ and tables *Subs* and *Sup* (lines 3-10) which are used for dealing with components of $G[V - S]$ instead of vertices of $G[V - S]$. Throughout this algorithm, we consider each component as an entity and we label it by its lowest numbered vertex in an arbitrary ordering of vertices of G fixed at beginning. We also compute all iso-triples and maintain them in the set *AllIS* (lines 11-14). Although an iso-triple is defined relative to one node, since elements S and $\mathcal{D}$ of iso-triples are the same relative to different nodes we define one set of iso-triples to all nodes. In fact, we assume ψ is a one-to-one mapping from S into set $\{1, 2, \cdots, k+1\}$ (line 13) and for each node z of $TD(H)$, we order vertices of χ_z arbitrarily and use this ψ as a mapping from S into χ_z. Using the fact that for a CPI ξ relative to a leaf, ψ is equal to the extension of ξ, we fill in full set arrays of leaves (lines 19-22). We find all CPIs of a node from CPIs of its children within two procedures *BuildFSAofSeparatorNode* and *BuildFSAofJoinNode*. Finally, we check whether or not there exists a complete CPI relative to the root r of $TD(H)$ (lines 27-28).

Algorithm A: testing induced subgraph isomorphism

```
Input:    $G$              : a $(k+1, g(k+1,n))$-bounded fragmentation graph
          $TD(H)$          : a nice tree decomposition of a partial $k$-tree $H$
Output:   true if $G$ is an induced subgraph isomorphic to $H$, false otherwise
Variables:
          $Subs[S, S', C']$ : specifies components of $G[V - S]$ which
                            are contained in $C' \in \mathcal{C}(G[V - S'])$, where $S' \subset S$
          $Sup[S, S', C]$   : specifies a component of $G[V - S']$ which
                            contains $C \in \mathcal{C}(G[V - S])$, where $S' \subset S$
          $FSA[z, \xi]$     : true if $\xi$ is in the full set of node $z$, false otherwise
          $AllIS$           : A set containing all iso-triples
begin
1     if $|V(G)| > |V(H)|$ return false;
2     let $\alpha_1, \alpha_2, \cdots, \alpha_{|TD(H)|}$ be the reverse of breadth first search order of nodes of $TD(H)$;
```

```
**for** each set $S$ of at most $k+1$ vertices of $G$
    find $\mathcal{C}(G[V-S])$
    **for** each set $S' \subset S$
        find $\mathcal{C}(G[V-S'])$
        **for** each component $C'_i \in \mathcal{C}(G[V-S'])$
            **let** $Subs[S, S', C'_i] \leftarrow \{C_{j_1}, \cdots C_{j_h}\}$ such that $V(C_{j_l}) \subseteq V(C'_i)$, $1 \le l \le h$
        **for** each component $C_i \in \mathcal{C}(G[V-S])$
            **let** $Sup[S, S', C_i] \leftarrow C'_j$ such that $V(C_i) \subseteq V(C'_j)$
**for** each $S \subseteq V(G)$ of size at most $k+1$
    **for** each $\mathcal{D} \subseteq \mathcal{C}(G[V-S])$
        **for** each one-to-one mapping $\psi$ from $S$ into set $\{1, 2, \cdots, k+1\}$
            **let** $AllIS \leftarrow AllIS \cup \{\xi = (S, \mathcal{D}, \psi)\}$
**for** each node $\alpha_i$ of $TD(H)$, $i$ from 1 to $|TD(H)|$
    **for** each iso-triple $\xi = (S, \mathcal{D}, \psi)$ in $AllIS$
        $FSA[\alpha_i, \xi] \leftarrow$ **false**;
    **if** $\alpha_i$ is a *leaf* node
        **for** each iso-triple $\xi$ in $AllIS$
            **let** $\varphi \leftarrow \psi$;
            **if** $\mathcal{D} = \emptyset$ **and**
            for each $u, v \in S$, $\{u, v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(H)$
                **let** $FSA[\alpha_i, \xi] \leftarrow$ **true**;
    **else if** $\alpha_i$ is a *separator* node
        BuildFSAofSeparatorNode($\alpha_i$);
    **else if** $\alpha_i$ is a *join* node
        BuildFSAofJoinNode($\alpha_i$);
**for** each iso-triple $\xi = (S, \mathcal{D}, \psi)$ in $AllIS$
    **if** $FSA[root, \xi] =$ **true and** $\mathcal{D} = \mathcal{C}(G[V-S])$ **return true**;
**return false**;
**end**
```

In procedure *BuildFSAofSeparatorNode*, we find all CPIs of a separator node z from

CPIs of its child z'. For each CPI ξ' relative to z' (line 51), we construct an iso-triple ξ relative to z which is separator-consistent with ξ' and thus is a CPI relative to z (lines 52-60). Condition (1) of separator-consistency between ξ and ξ' is checked in line 52, Condition (3) is tested in line 53 and Condition (2) is checked in lines 54-60. To test Condition (2), we use additional set $\mathcal{D}''$. First we find all supergraphs of components $C' \in \mathcal{D}'$ (line 56). Since a component $D \in \mathcal{D}$ is a supergraph of components in $\mathcal{D}'$ and not a supergraph of components in $\mathcal{C}(G[V - S']) - \mathcal{D}'$ (Condition (2) of separator-consistency), we again find all components of $\mathcal{C}(G[V - S'])$ which are contained in components $D \in \mathcal{D}$ (line 59) to make sure $\mathcal{D}'$ and $\mathcal{D}''$ are equal (line 59).

BuildFSAofSeparatorNode(z)

```
Input:   z   : a separator node of $TD(H)$
begin
50  let $z' \leftarrow$ the child of $z$;
51  for each iso-triple $\xi' = (S', \mathcal{D}', \psi')$ in $AllIS$ such that $FSA[z', \xi']$ is true
52      let $S \leftarrow \{v \in S' | \psi'(v) \in \chi_z\}$;
53      let $\psi(v) \leftarrow \psi'(v)$ for $v \in S$;
54      $\mathcal{D} \leftarrow \emptyset$;
55      for each $C' \in \mathcal{D}'$
56          let $\mathcal{D} \leftarrow \mathcal{D} \cup \{Sup[S', S, C']\}$;
57      $\mathcal{D}'' \leftarrow \emptyset$;
58      for each $C \in \mathcal{D}$
59          let $\mathcal{D}'' \leftarrow \mathcal{D}'' \cup Subs[S', S, C]$;
60      if $\mathcal{D}'' = \mathcal{D}'$
61          let $\xi \leftarrow (S, \mathcal{D}, \psi)$;
62          let $FSA[z, \xi] \leftarrow$ true;
end
```

In procedure *BuildFSAofJoinNode*, we find all CPIs of a join node z from CPIs of its children z_1 and z_2. For each iso-triple ξ relative to z, we construct all iso-triples ξ_1 and ξ_2 relative to its children which are join-consistent with ξ (lines 77-85). If ξ_1 and ξ_2 are CPI then ξ is also a CPI (lines 86-87). Condition (1) of join-consistency is checked in line 77, Condition (3) is tested in line 78, Condition (4) is checked in line 79 and finally Condition (2) is checked in lines 80-83. We partition elements of $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$ (line 80), but we still have to make sure that each element $D \in \mathcal{D}$, which is in $\mathcal{D}_i$, is a component of $G[V-S_i]$. Since all outgoing edges from $V(D)$ go into S, we only check whether there is any edge between $V(D)$ and $S - S_i$. If there is such edge, then the component $D' \in \mathcal{C}(G[V - S_i])$ which is a supergraph of D has at least one vertex in $S - S_i$. This condition is checked in line 83.

```
**BuildFSAofJoinNode(z)**
*Input:*   $z$   : a join node of $TD(H)$
**begin**
75  **let** $z_i \leftarrow i$th child of $z$, $i = 1, 2$;
76  **for** each iso-triple $\xi$ in $AllIS$
77      **let** $S_i \leftarrow \{v \in S | \psi(v) \in \chi_{z_i}\}$, $i = 1, 2$;
78      **let** $\psi_i(v) \leftarrow \psi(v)$ for $v \in S_i$, $i = 1, 2$;
79      **if** for all $u, v \in S$, $\{u, v\} \in E(G)$ if and only if $\{\psi(u), \psi(v)\} \in E(H)$
80          **for** each partition of elements of $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$
81              **let** $bool \leftarrow$ **true**;
82              **for** each $D \in \mathcal{D}$, which is in $\mathcal{D}_i$
83                  **if** $V(Sup[S, S_i, D]) \cap S - S_i \neq \emptyset$ **let** $bool \leftarrow$ **false**;
84              **if** $bool =$ **true**
85                  **let** $\xi_i \leftarrow (S_i, \mathcal{D}_i, \psi_i)$, $i = 1, 2$;
86                  **if** $FSA[z_1, \xi_1] = FSA[z_2, \xi_2] =$**true**
87                      **let** $FSA[z, \xi] \leftarrow$ **true**;
**end**
```

To prove the correctness of the algorithm and obtaining the running time, first we define an invariant and present two lemmas for separator and join nodes.

Definition 5.7 *We say that the FSA array relative to a node z of $TD(H)$ is in* correct form*, if for each ξ relative to z, $FSA[z, \xi] =$* **true** *if and only if ξ is a CPI relative to z.*

Lemma 5.5 *Given the FSA array of the child z' of a separator node z in correct form, Procedure* BuildFSAofSeparatorNode *fills in the FSA array relative to z in correct form in $O(g(k+1,n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1})$ time.*

Proof: By Lemma 5.3, for each CPI ξ relative to z, there exists a CPI ξ' relative to the child z' of z. Using this fact, we try to find a CPI ξ relative to z which can be constructed from a CPI ξ' relative to z'. We construct an iso-triple ξ separator-consistent with CPI ξ'. Lines 52 and 53 correspond to Conditions (1) and (3) of separator-consistency. Element $\mathcal{D}$ of ξ is constructed in lines 54-60 in which we find all components $D \in G[V-S]$ which are supergraphs of components in $\mathcal{D}'$ and not supergraphs of components in $\mathcal{C}(G[V-S']) - \mathcal{D}'$ (see Condition (2) of separator-consistency). Thus iso-triple ξ is separator-consistent with the CPI ξ' and by Lemma 5.3, it is a CPI.

We now compute the running time. For each element of the full set array of z', we specify S and ψ for an iso-triple $\xi = (S, \mathcal{D}, \psi)$ relative to z in constant time, because the number of vertices in S and S' is at most constant $k+1$ (lines 52-53). Constructing $\mathcal{D}$ takes $O(g(k+1,n))$ time (lines 54-59), since we consider each $C' \in \mathcal{C}(G[V-S'])$ in line 55 and each $C \in \mathcal{C}(G[V-S])$ in line 58 whose number is $O(g(k+1,n))$. Thus executing lines 52-62 takes at most $O(g(k+1,n))$ time. Since the number of iso-triples checked in line 51 is at most $O(2^{g(k+1,n)}|V(G)|^{k+1})$, the procedure *BuildFSAofSeparatorNode* takes $O(g(k+1,n) \cdot 2^{g(k+1,n)}|V(G)|^{k+1})$ time. □

Lemma 5.6 *Given the FSA arrays of children z_1 and z_2 of a join node z in correct form, Procedure* BuildFSAofJoinNode *fills in the FSA array relative to z in correct form in $O(g(k+1,n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1})$ time.*

Proof: By Lemma 5.4, for each CPI ξ relative to z, there exist CPIs ξ_1 and ξ_2 relative to its children z_1 and z_2 such that ξ is join-consistent with them. Intuitively, we try to find them and use them as a certificate for ξ. Lines 77, 78 and 79 correspond to Conditions (1), (3) and (4) of join-consistency. By Condition (2) of join-consistency, we know that $\mathcal{D}_1$ and $\mathcal{D}_2$ partitions $\mathcal{D}$. Thus we examine all partitions by brute force (line 80) and for each of them we check whether $\mathcal{D}_i$ is a subset of $\mathcal{C}(G[V-S_i])$, $i = 1, 2$ (lines 81-84). We note that, in line 83, we check whether a component $D' \subseteq \mathcal{C}(G[V-S_i])$. Thus if we find two iso-triples ξ_1 and ξ_2 which are also CPIs (line 86) then ξ is also a CPI (line 87).

To compute the running time, we first observe that we can execute lines 77-79 in constant time since $|S|$ is bounded above by $k+1$. Since $\mathcal{D}$ has at most $O(g(k+1,n))$ elements, we have $O(2^{g(k+1,n)})$ choices for partitioning elements of $\mathcal{D}$ between $\mathcal{D}_1$ and $\mathcal{D}_2$ (line 80). Executing lines 82-83 takes $O(g(k+1,n))$ time, since $\mathcal{D}$ has at most $O(g(k+1,n))$ elements. Therefore, at most $O(g(k+1,n) \cdot 2^{g(k+1,n)})$ time is required for executing lines 77-83. Since the number of iso-triples checked in line 76 is at most $O(2^{g(k+1,n)}|V(G)|^{k+1})$, the running time of this procedure is in $O(g(k+1,n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1})$. □

We are now ready to prove the correctness and compute the running time of the whole algorithm.

Theorem 5.3 *The above algorithm solves the induced subgraph isomorphism problem in* $O(g(k+1,n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1} \cdot |V(H)|)$ *time.*

Proof: We first prove the correctness of the algorithm. We show that FSA array relative to each node z of $TD(H)$ is in correct form, that is, for each ξ relative to a node z, $FSA[z,\xi] = \textbf{true}$ if and only if ξ is a CPI relative to z. Using this fact, we check whether there exists a complete CPI relative to the root r of $TD(H)$ (line 28), and the correctness of the algorithm immediately follows from Lemma 5.2.

To prove our claim, we use induction on the height of a node z in $TD(H)$. If the height is zero, then z is a leaf. We consider an iso-triple ξ relative to z. Since $H_{[z]} = H[\chi_z]$ for a leaf, if ξ is a CPI then $\mathcal{D} = \emptyset$. In addition by Property [Pb] of extensions and the fact that ψ is a one-to-one mapping, we have $\varphi = \psi$. Using these facts, we only need to check Property [Pc] of extensions. We check all these conditions in line 21 and thus the claim is true for a leaf node. For a separator node z, by the induction hypothesis, the claim is true for the child z' of z and by Lemma 5.5, the claim is true for z. For a join node z, again by the induction hypothesis, the claim is true for children z_1 and z_2 of z and by Lemma 5.6 the claim is true for z.

Now, we compute the running time of the algorithm. We choose every set S with at most $k+1$ vertices in line 3, and using depth first search, we find connected components of $G[V-S]$ in $O(|V(G)|)$ time (line 4). Similarly, we can find connected components of $G[V-S']$ in line 6 in $(O(|V(G)|)$ time for each set $S' \subset S$. We can execute line 8 in $O(|V(G)|)$ time for each component $C'_i \in \mathcal{C}(G[V-S'])$, $S' \subset S$. To do that we only need to have two arrays A and A'. Array A (A') is of size $|V(G)|$ and $A[v] = C$ ($A'[v] = C'$) if vertex v is in $V(C)$ ($V(C')$), where $C \in \mathcal{C}(G[V - S])$ ($C' \in \mathcal{C}(G[V - S'])$). Then we can execute line 8 by only one pass of array A' to find the corresponding component of each vertex $v \in C'_i$ in array A. Using this fact and since the number of C_i's for each set S is at most $O(g(k+1, n))$ (line 7), we can execute lines 7-8 in $O(g(k+1, n) \cdot |V(G)|)$ time. Similarly, we can execute lines 9-10 in $O(g(k+1, n) \cdot |V(G)|)$ time. As the number of such sets S' for each set S is a constant (since k is a constant), this step takes $O(g(k+1, n) \cdot |V(G)|)$ time for each set S (lines 4-10). Since the number of choices of S is in $O(|V(G)|^{k+1})$ (line 3), the overall running time of lines 3-10 is in $O(2^{g(k+1,n)}|V(G)|^{k+1})$ time. In lines 11-14, we construct all iso-triples. As the number of iso-triples is bounded above by $O(2^{g(k+1,n)}|V(G)|^{k+1})$ by Lemma 5.1 and processing each of them takes constant time (line 14), we can execute lines 11-14 in $O(2^{g(k+1,n)}|V(G)|^{k+1})$ time. Therefore preprocessing steps (lines 1-14) takes the sum of the two aforementioned times which is in $O(g(k+1, n) \cdot |V(G)| \cdot 2^{g(k+1,n)} \cdot |V(G)|^{k+1}) = O(g(k + 1, n) \cdot |V(G)|^{k+1} \cdot |V(H)|)$ time.

For a leaf, the check in line 21 takes constant time because the number of vertices in S is at most $k + 1$. Since the number of iso-triples checked in line 19 is at most $O(2^{g(k+1,n)}|V(G)|^{k+1})$, the running time of lines 19-22 is at most $O(2^{g(k+1,n)}|V(G)|^{k+1})$. For a separator node z, by Lemma 5.5, executing the procedure *BuildFSAofSeparatorNode* takes $O(g(k + 1, n) \cdot 2^{g(k+1,n)}|V(G)|^{k+1})$ time. Finally, for a join node, the procedure *BuildFSAofJoinNode* takes $O(g(k+1, n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1})$ time (Lemma 5.6). Therefore, the running time for a node of $TD(H)$ is at most $O(g(k + 1, n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1})$ due to join nodes.

Since the number of nodes of $TD(H)$ is in $O(|V(H)|)$ and executing lines 16-17 takes $O(2^{g(k+1,n)}|V(G)|^{k+1})$ time whose time dominated by the others, the running time of the main loop of this algorithm (lines 15-26) is in $O(g(k + 1, n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1} \cdot |V(H)|)$. The condition in line 28 also can be computed in constant time and thus executing line 27-28 takes $O(2^{g(k+1,n)}|V(G)|^{k+1})$ time. Therefore the running time of the whole algorithm is in $O(g(k + 1, n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1} \cdot |V(H)|)$. □

We can also solve the subgraph isomorphism problem using the above algorithm. First, we present some definitions and lemmas.

Definition 5.8 *The* subdivision *of an edge* $\{u, w\}$ *is the operation of deleting this edge* *and adding a new vertex v and two new edges $\{u, v\}$ and $\{v, w\}$. The graph obtained from graph G by subdivisions of all its edges is denoted by G^*.*

Lemma 5.7 *If G is a partial k-tree and $k \geq 2$, then G^* is a partial k-tree.*

Proof: First we consider $TD(G)$ of width k. By Property (2) of tree decompositions, for each edge $\{u, v\} \in E(G)$, there exists a node z such that both u and v belong to χ_z. To construct a tree decomposition of width k for G^*, we first construct $TD(G)$ and then for each edge $\{u, v\} \in E(G)$, we append a node z' with $\chi_{z'} = \{u, v, w\}$ to the node z, where w is the added vertex in subdivision of $\{u, v\}$. □

Lemma 5.8 *If G is a $(k, g(k, n))$-bounded fragmentation graph and k is a constant, G^* is a $(k, O(g(k, n)))$-bounded fragmentation graph.*

Proof: Suppose a set S removed from G^* has l_1 vertices originally from G and l_2 vertices added by subdivisions. Without loss of generality, we can assume that first vertices originally from G are removed and then the remaining vertices of S are removed one at a time. Suppose we remove vertices originally from G in both G and G^*. We call a component in G^* a *new component*, if it has no corresponding component in G. We bound the number of new components. For each pair u and v of vertices originally from G, there can exist edge $\{u, v\} \in E(G)$. Subdivision of $\{u, v\}$ in G^* adds a new component which only contains w. After removing vertices originally from G, we can only have this kind of new component. Hence removing these vertices can produce at most $\binom{l_1}{2} \leq k^2 - k$ new components. We now remove the rest of vertices of S one at a time. Each of these vertices is added on one edge of G, and hence its removal in G^* can increase the number of components by one. Thus removal of these vertices can increase the number of components by at most $l_2 \leq k$. Therefore $|\mathcal{C}(G^*[V - S])|$ is at most $g(k, n) + k^2$. Since k is a constant, this number is in $O(g(k, n))$. □

Lemma 5.9 *(Lemma 1.4 [MT92]) Let G and H be two graphs. G is subgraph isomorphic to H if and only if G^* is induced subgraph isomorphic to H^*.* □

By Lemmas 5.7, 5.8 and 5.9, we can solve the subgraph isomorphism problem by means of induced subgraph versions. Here we finish the proof of Theorem 5.2.

By considering directions of edges in consistency checking, we can generalize the algorithm mentioned in the proof of Theorem 5.2 to directed graphs (we replace edge $\{u, v\}$ by (u, v) in lines 21 and 79). Hence if G is a directed graph with a $(k+1, g(k+1, n))$-bounded fragmentation underlying graph and H is a directed graph, whose underlying graph is a partial k-tree, then there are $O(g(k+1, n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1} \cdot |V(H)|)$-time algorithms which solve isomorphism, subgraph isomorphism and induced subgraph isomorphism.

We can also extend our polynomial-time algorithm for testing subgraph isomorphism to graphs of locally bounded treewidth.

Theorem 5.4 *Subgraph and induced subgraph isomorphism can be solved in* $O((ltw(diam(G)) \log |V(G)|) \cdot 2^{O(ltw(diam(G)) \log |V(G)|)} |V(G)|^{ltw(diam(G))+1} \cdot |V(H)|)$ *time when graph H has locally bounded treewidth and graph G is a totally* log*-bounded fragmentation graph and has constant diameter.*

Proof: The idea of the proof follows Baker's idea [Bak94]. For graph H and integers $0 \leq i \leq j$, we let $L^H[i, j] = \bigcup_{i \leq k \leq j} L_k$, where L_k (the kth layer) consists of all vertices at distance k from a fixed vertex v. We note that this definition of layers is the same as the definition of layers for clique-sum graphs introduced in Section 3.1.

We suppose d is the number of layers in graph H. For $0 \leq i \leq d - diam(G) + 1$, let $L_{i,diam(G)} = L^H[i, i + diam(G) - 1]$ and let $H[L_{i,diam(G)}]$ be the induced graph on vertices in $L_{i,diam(G)}$. Eppstein proved that if H has locally bounded treewidth, then $tw(H[L_{i,diam(G)}]) \leq ltw(diam(G))$ and thus $H[L_{i,diam(G)}]$ has bounded treewidth [Epp00]. As mentioned in Section 2.2, we can construct a tree decomposition of each graph of bounded treewidth in linear time [Bod96].

If G is (induced) subgraph isomorphic to H, then it is contained in one of the graphs $H[L_{i,diam(G)}]$. Using Corollary 5.1, we can solve graph isomorphism separately for each $H[L_{i,diam(G)}]$. The desired running time follows from Theorem 5.2 and the fact that each vertex of H has participated in at most $diam(G)$ iterations of the algorithm. $\square$

The point that the source graph G has bounded diameter is very important and without it, the problem remains NP-complete.

Theorem 5.5 *Let graph H have locally bounded treewidth with $\Delta(H) = 3$ and graph G have bounded treewidth with $\Delta(G) = 2$. The subgraph isomorphism problem for the source* *graph G and the host graph H is NP-complete.*

Proof: Let H be a planar cubic graph (every vertex has either degree three or degree zero) in which there is no face with fewer than five edges. Finding a Hamiltonian circuit in H is NP-complete [GJT76]. As mentioned in Section 2.3, planar graphs have locally bounded treewidth. By choosing G to be a cycle of length n, where n is the size of H, the subgraph isomorphism problem is equivalent to finding a Hamiltonian cycle in H and thus it is NP-complete. □

One application of Theorem 5.4 is that it gives a linear-time algorithm for testing subgraph isomorphism for fixed patterns. More precisely:

Corollary 5.2 *For a fixed pattern G and a graph H of locally bounded treewidth, subgraph isomorphism and induced subgraph isomorphism can be tested in $O(|V(H)|)$ time.*

Proof: If G is fixed, then $diam(G) = O(|V(G)|)$ is a constant. The result follows from this fact and Theorem 5.4. □

Corollary 5.2 is the same as Eppstein's result for testing subgraph isomorphism of fixed patterns on graphs of locally bounded treewidth [Epp99, Epp00]. Using this result, Eppstein also showed that other problems such as finding diameter (if we know the graph has bounded diameter), h-clustering for constant h and finding girth (if we know the graph has bounded girth) can be tested in $O(n)$ time for these graphs. The reader is referred to the original papers for details.

In this chapter, using the dynamic programming approach of Arnborg and Proskurowski (Section 2.4), we proved that the subgraph isomorphism problem has a polynomial-time solution, when the source graph G is a bounded fragmentation graph and the host graph H has bounded treewidth or when the source graph G has bounded diameter and bounded fragmentation and the host graph H has locally bounded treewidth.

Chapter 6

Conclusions and future work

Solving NP-complete problems on graphs of bounded treewidth and finding approximation algorithms for NP-optimization problems on graphs of locally bounded treewidth are the main focus of this thesis. More precisely, the results in this thesis can be divided into two main categories.

First, we introduced H-minor-free graphs as graphs of locally bounded treewidth where H is a single-crossing graph. We proved that these graphs have linear local treewidth. Using this result, we proved that the local treewidth of $K_{3,3}$-minor-free graphs and K_5-minor-free graphs is bounded by $3r + 4$. Alber et al. [ABFN00] proved planar graphs, which are both $K_{3,3}$-minor-free and K_5-minor-free graphs, have linear local treewidth. Thus our result extends the class of graphs with linear local treewidth. Small local treewidth of planar graphs was one of the bases of Baker's approach for designing practical approximation algorithms on planar graphs. Using a similar approach, we found polynomial time approximation schemes for several NP-optimization problems such as maximum independent set, minimum dominating set and minimum vertex cover on graphs excluding $K_{3,3}$ or K_5 as a minor. In addition to this main result, we proved problems such as minimum dominating set are fixed parameter tractable on these graphs.

Second, we discussed the subgraph isomorphism problem. We presented a polynomial-time algorithm for this problem when the source graph is a log-bounded-fragmentation graph and the host graph is bounded treewidth. As we showed, the class of log-bounded fragmentation graphs contains graphs other than bounded degree graphs, e.g. Hamiltonian graphs, and hence our result extends the result of Matoušek and Thomas on bounded degree source graphs and host graphs of bounded treewidth. In addition, we demonstrated how this algorithm can be generalized to graphs of locally bounded treewidth.

Beside the two main contributions mentioned above, we introduced bounded fragmentation as a measure of the reliability of a network, and we presented several classes of bounded fragmentation graphs. Here, we present several open problems that can be considered as possible extensions of this thesis.

Parallelizing exact algorithms and PTASs for problems given in Chapter 3 is a topic of interest. As the general dynamic programming approach can be parallelized easily (see Bodlaender's paper [Bod97]), in this extension, one needs to parallelize computing the tree decompositions of each constant number of consecutive layers introduced for clique-sum graphs in Chapter 3.

We suspect that Baker's approach can be applied to obtain practical PTASs for other problems. Some examples are as follows: graph s-partitioning, in which one searches for a partition of the vertex set of a graph into sets of size s and $n - s$ such that it minimizes the cutsize, maximum matching and variants of dominating sets introduced by Alber et al. [ABFN00], such as independent dominating set, total dominating set, perfect dominating set, perfect independent dominating set and total perfect dominating set. All these problems were solved for k-outerplanar graphs [BP92, DST96, ABFN00]. Using these results, one only needs to remove the layers in Baker's approach appropriately and obtain an approximation from the solution for each resulting k-outerplanar graph, as in our PTAS for minimum dominating set.

Alber et al. [ABFN00] showed that a planar graph with a dominating set of size k has treewidth $O(6\sqrt{34}\sqrt{k})$. Using this result, they concluded that finding a dominating set of size k in a planar graph can be solved in time $O(c^{\sqrt{k}}n)$, where $c = 3^{6\sqrt{34}}$ (the proof of the second result follows immediately from Theorem 3.6). We believe that in the proof of the first result one can replace the concept of outerplanarity by the concept of layers introduced for clique-sum graphs and obtain the similar result for these graphs. The reader is referred to the original paper [ABFN00] for further detail.

Another problem related to subgraph isomorphism is the problem of finding the largest common subgraph of two graphs. Brandenburg [Bra01] showed that if two graphs G and

H are k-connected partial k-trees, the problem of finding the largest common k-connected subgraph can be solved in polynomial time. In addition, using Brandenburg's ideas [Bra01] and Gupta and Nishimura's ideas [GN96b], one can observe that finding the largest common $(k-1)$-connected subgraph is NP-complete. It would be interesting to know whether or not the result can be generalized to bounded fragmentation graphs.

A trivial algorithm for testing whether a graph G is a (k, c)-bounded fragmentation graph, for constants k and c, is to check all subsets of vertices of size at most k and count the number of connected components. The running time of this algorithm is in $O(n^{k+1})$. It might be possible to give an algorithm whose running time is $O(n^d)$, where d is a constant independent of k. A randomized approach might be another way to solve this problem. Also, it seems the problem is more straightforward for planar graphs (see the paper due to Seymour and Thomas [ST94] on branchwidth of planar graphs to obtain more ideas).

In Chapter 4, we introduced several classes of bounded fragmentation graphs. Finding other classes of this kind, if they exist, is an interesting question. The relation between bounded fragmentation and treewidth is also interesting, in particular when in solving subgraph isomorphism, we search for graphs which are bounded fragmentation graphs and have bounded treewidth (see Chapter 5). A path is a bounded fragmentation graph which has bounded treewidth. Graphs coverable with a constant number of vertex-disjoint paths and graphs with maximum constant degree are the only known classes of bounded fragmentation graphs which have bounded treewidth. Considering trees as graphs of treewidth at most one and understanding their relationship with bounded fragmentation is an easy approach to attack the problem. Characterizing bounded fragmentation graphs is another possible extension of this thesis.

Finally, all graphs introduced in Chapter 4 are $(k, O(k))$-bounded fragmentation graphs. It would be instructive to determine whether there is any $(k, g(k))$-bounded fragmentation graph where $g(k)$ is not $O(k)$.

Bibliography

[ABFN00] Jochen Alber, Hans L. Bodlaender, Henning Fernau, and Rolf Niedermeier. Fixed parameter algorithms for planar dominating set and related problems. In *Algorithm theory—Scandinavian Workshop on Algorithm Theory 2000 (Bergen, 2000)*, pages 97–110. Springer, Berlin, 2000.

[ABG$^+$92] Peter J. Artymiuk, Peter A. Bath, Hans M. Grindley, Carine A. Pepperrell, Andrzej R. Poirrette, Dan W. Rice, Daniel A. Thorner, Douglas J. Wild, Peter Willet, Frank H. Allen, and Robert Taylor. Similarity searching in databases of three-dimensional molecules and macromolecules. *J. Chemical Information and Computer Sciences*, 32:617–630, 1992.

[ACP87] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, 1987.

[ACPS93] Stefan Arnborg, Bruno Courcelle, Andrzej Proskurowski, and Detlef Seese. An algebraic theory of graph reduction. *J. Assoc. Comput. Mach.*, 40(5):1134–1164, 1993.

[AF93] Karl Abrahamson and Michael R. Fellows. Finite automata, bounded treewidth and well-quasiordering. In *Graph structure theory (Seattle, WA, 1991)*, pages 539–563. Amer. Math. Soc., Providence, RI, 1993.

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[ALS88] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Problems easy for tree-decomposable graphs (extended abstract). In *Automata, languages and programming (Tampere, 1988)*, pages 38–51. Springer, Berlin, 1988.

[AP86] Stefan Arnborg and Andrzej Proskurowski. Characterization and recognition of partial 3-trees. *SIAM J. Algebraic Discrete Methods*, 7(2):305–314, 1986.

[AP89] Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Appl. Math.*, 23(1):11–24, 1989.

[APC90] Stefan Arnborg, Andrzej Proskurowski, and Derek G. Corneil. Forbidden minors characterization of partial 3-trees. *Discrete Math.*, 80(1):1–19, 1990.

[Arn85] Stefan Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability—a survey. *BIT*, 25(1):2–23, 1985.

[Asa85] Takao Asano. An approach to the subgraph homeomorphism problem. *Theoret. Comput. Sci.*, 38(2-3):249–267, 1985.

[AST90] Noga Alon, Paul Seymour, and Robin Thomas. A separator theorem for for graphs with excluded minor and its applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (Baltimore, MD, 1990)*, pages 293–299, 1990.

[Bak94] Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. Assoc. Comput. Mach.*, 41(1):153–180, 1994.

[BdF96] Hans L. Bodlaender and Babette de Fluiter. Reduction algorithms for constructing solutions in graphs with small treewidth. In *Computing and combinatorics (Hong Kong, 1996)*, pages 199–208. Springer, Berlin, 1996.

[BDK00] Hajo J. Broersma, Elias Dahlhaus, and Ton Kloks. A linear time algorithm for minimum fill-in and treewidth for distance hereditary graphs. *Discrete Appl. Math.*, 99(1-3):367–400, 2000.

[BGHK95] Hans L. Bodlaender, John R. Gilbert, Hjálmtýr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2):238–255, 1995.

[BK96] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996.

[BKK95] Hans L. Bodlaender, Ton Kloks, and Dieter Kratsch. Treewidth and pathwidth of permutation graphs. *SIAM J. Discrete Math.*, 8(4):606–616, 1995.

[BLW87] Marshall W. Bern, Eugene L. Lawler, and Alice L. Wong. Linear-time computation of optimal subgraphs of decomposable graphs. *J. Algorithms*, 8(2):216–235, 1987.

[BM76] John A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* American Elsevier Publishing Co., Inc., New York, 1976.

[BM93] Hans L. Bodlaender and Rolf H. Möhring. The pathwidth and treewidth of cographs. *SIAM J. Discrete Math.*, 6(2):181–188, 1993.

[Bod88] Hans L. Bodlaender. Dynamic programming algorithms on graphs with bounded treewidth. In *Proceedings of Fifteenth International Colloquium on Automata, Languages and Programming (Tampere, 1988)*, pages 105–119. Springer, Berlin, 1988.

[Bod96] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[Bod97] Hans L. Bodlaender. Treewidth: algorithmic techniques and results. In *Mathematical foundations of computer science 1997 (Bratislava, 1997)*, pages 19–36. Springer, Berlin, 1997.

[Bod98] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoret. Comput. Sci.*, 209(1-2):1–45, 1998.

[BP92] Thang Nguyen Bui and Andrew Peck. Partitioning planar graphs. *SIAM J. Comput.*, 21(2):203–215, 1992.

[Bra01] Franz J. Brandenburg. Pattern matching problems in graphs. Manuscript, 2001.

[BT97] Hans L. Bodlaender and Dimitrios M. Thilikos. Treewidth for graphs with small chordality. *Discrete Appl. Math.*, 79(1-3):45–61, 1997.

[CGH75] Ernest J. Cockayne, Sue Goodman, and Stephen Hedetniemi. A linear time algorithm for the domination number of a tree. *Inform. Process. Lett.*, 4(1):41–44, 1975.

[CH95] Zhi-Zhong Chen and Xin He. NC algorithms for partitioning planar graphs into induced forests and approximating NP-hard problems. In *Graph-theoretic concepts in computer science (Aachen, 1995)*, pages 275–289. Springer, Berlin, 1995.

[Che95] Zhi-Zhong Chen. NC algorithms for partitioning sparse graphs into induced forests with an application. In *Sixth Annual International Symposium on Algorithms and Computation (Cairns, 1995)*, pages 428–437. Springer, Berlin, 1995.

[Che98] Zhi-Zhong Chen. Efficient approximation schemes for maximization problems on $K_{3,3}$-free or K_5-free graphs. *J. Algorithms*, 26(1):166–187, 1998.

[CNS82] Norishige Chiba, Takao Nishizeki, and Nobuji Saito. An approximation algorithm for the maximum independent set problem on planar graphs. *SIAM J. Comput.*, 11(4):663–675, 1982.

[Cou90] Bruno Courcelle. Graph rewriting: an algebraic and logic approach. In *Handbook of theoretical computer science, Vol. B*, pages 193–242. Elsevier, Amsterdam, 1990.

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Springer-Verlag, New York, 1999.

[DLP00a] Anders Dessmark, Andrzej Lingas, and Andrzej Proskurowski. Faster algorithms for subgraph isomorphism of k-connected partial k-trees. *Algorithmica*, 27(3-4):337–347, 2000.

[DLP00b] Anders Dessmark, Andrzej Lingas, and Andrzej Proskurowski. Maximum packing for k-connected partial k-trees in polynomial time. *Theoret. Comput. Sci.*, 236(1-2):179–191, 2000.

[DST96] Josep Díaz, Maria J. Serna, and Jacobo Torán. Parallel approximation schemes for problems on planar graphs. *Acta Inform.*, 33(4):387–408, 1996.

[Epp99] David Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms Appl.*, 3:no. 3, 27 pp. (electronic), 1999.

[Epp00] David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3-4):275–291, 2000.

[FG99] Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable graphs. In *Automata, languages and programming (Prague, 1999)*, pages 331–340. Springer, Berlin, 1999.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman and Co., San Francisco, Calif., 1979.

[GJT76] Michael R. Garey, David S. Johnson, and Robert E. Tarjan. The planar hamiltonian circuit problem is NP-complete. *SIAM J. Comput.*, 5(6):704–714, 1976.

[GN94] Arvind Gupta and Naomi Nishimura. Sequential and parallel algorithms for embedding problems on classes of partial k-trees. In *Algorithm theory—Scandinavian Workshop on Algorithm Theory 1994 (Aarhus, 1994)*, pages 172–182. Springer, Berlin, 1994.

[GN96a] Arvind Gupta and Naomi Nishimura. Characterizing the complexity of subgraph isomorphism for graphs of bounded path-width. In *STACS 96 (Grenoble, 1996)*, pages 453–464. Springer, Berlin, 1996.

[GN96b] Arvind Gupta and Naomi Nishimura. The complexity of subgraph isomorphism for classes of partial k-trees. *Theoret. Comput. Sci.*, 164(1-2):287–298, 1996.

[GNPR00] Arvind Gupta, Naomi Nishimura, Andrzej Proskurowski, and Prabhakar Ragde. Embeddings of k-connected graphs of pathwidth k. In *Algorithm theory—Scandinavian Workshop on Algorithm Theory 2000 (Bergen, 2000)*, pages 111–124. Springer, Berlin, 2000.

[Gro] Martin Grohe. Local tree-width, excluded minors, and approximation algorithms. To appear in Combinatorica in 2001.

[Har69] Frank Harary. *Graph Theory.* Addison-Wesley Publishing Co., Reading, Mass.-Menlo Park, Calif.-London, 1969.

[HNRT01] Mohammadtaghi Hajiaghayi, Naomi Nishimura, Prabhakar Ragde, and Dimitrios M. Thilikos. Fast approximation schemes for $K_{3,3}$-minor-free or K_5-minor-free graphs. In *Euroconference on Combinatorics, Graph Theory and Applications 2001 (Barcelona, 2001).* 2001.

[Klo93] Ton Kloks. Treewidth of circle graphs. In *Algorithms and computation (Hong Kong, 1993)*, pages 108–117. Springer, Berlin, 1993.

[KM92] André Kézdy and Patrick McGuinness. Sequential and parallel algorithms to find a K_5 minor. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms (Orlando, FL, 1992)*, pages 345–356, 1992.

[Lag96] Jens Lagergren. Efficient parallel algorithms for graphs of bounded tree-width. *J. Algorithms*, 20(1):20–44, 1996.

[Lee90] Jan van Leeuwen. Graph algorithms. In *Handbook of theoretical computer science, Vol. A*, pages 525–631. Elsevier, Amsterdam, 1990.

[Lin89] Andrzej Lingas. Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoret. Comput. Sci.*, 63(3):295–302, 1989.

[LS88] Andrzej Lingas and Maciej M. Sysło. A polynomial-time algorithm for subgraph isomorphism of two-connected series-parallel graphs. In *Automata, languages and programming (Tampere, 1988)*, pages 394–409. Springer, Berlin, 1988.

[LT80] Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980.

[Mat78] David W. Matula. Subtree isomorphism in $O(n^{5/2})$. *Ann. Discrete Math.*, 2:91–106, 1978. Algorithmic aspects of combinatorics (Conf., Vancouver Island, B.C., 1976).

[MT92] Jiří Matoušek and Robin Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Math.*, 108(1-3):343–364, 1992. Topological, algebraical and combinatorial structures. Frolík's memorial volume.

[NR99] Rolf Niedermeier and Peter Rossmanith. Upper bounds for vertex cover further improved. In *STACS 99 (Trier, 1999)*, pages 561–570. Springer, Berlin, 1999.

[Ros74] Donald J. Rose. On simple characterizations of k-trees. *Discrete Math.*, 7:317–322, 1974.

[RS86] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

[RS93] Neil Robertson and Paul Seymour. Excluding a graph with one crossing. In *Graph structure theory (Seattle, WA, 1991)*, pages 669–675. Amer. Math. Soc., Providence, RI, 1993.

[San96] Daniel P. Sanders. On linear recognition of tree-width at most four. *SIAM J. Discrete Math.*, 9(1):101–117, 1996.

[See96] Detlef Seese. Linear time computable problems and first-order descriptions. *Math. Structures Comput. Sci.*, 6(6):505–526, 1996. Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation (Volterra, 1995).

[SSR94] Ravi Sundaram, Karan S. Singh, and Pandu C. Rangan. Treewidth of circular-arc graphs. *SIAM J. Discrete Math.*, 7(4):647–655, 1994.

[ST94] Paul D. Seymour and Robin Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[Sys82] Maciej M. Sysło. The subgraph isomorphism problem for outerplanar graphs. *Theoret. Comput. Sci.*, 17(1):91–97, 1982.

[TP93] Jan A. Telle and Andrzej Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In *Proceedings of Third Workshop on Algorithms and Data Structures (Montreal, 1993)*, pages 610–621. Springer, Berlin, 1993.

[Tur41] Paul Turán. Eine extremalaufgabe aus der graphentheorie. *Mat. Fiz Lapook*, 48(1):436–452, 1941.

[Wag37] Kehrer Wagner. Über eine Eigenschaft der eben Komplexe. *Deutsche Math.*, 2:280–285, 1937.

[Wes96] Douglas B. West. *Introduction to Graph Theory.* Prentice Hall Inc., Upper Saddle River, NJ, 1996.

[Yan78] Mihalis Yannakakis. Node- and edge-deletion NP-complete problems. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, CA, 1978)*, pages 253–264. ACM press, New York, 1978.