Lower bounds for preprocessing algorithms based on protrusion replacement

Wulms, J.J.H.M.

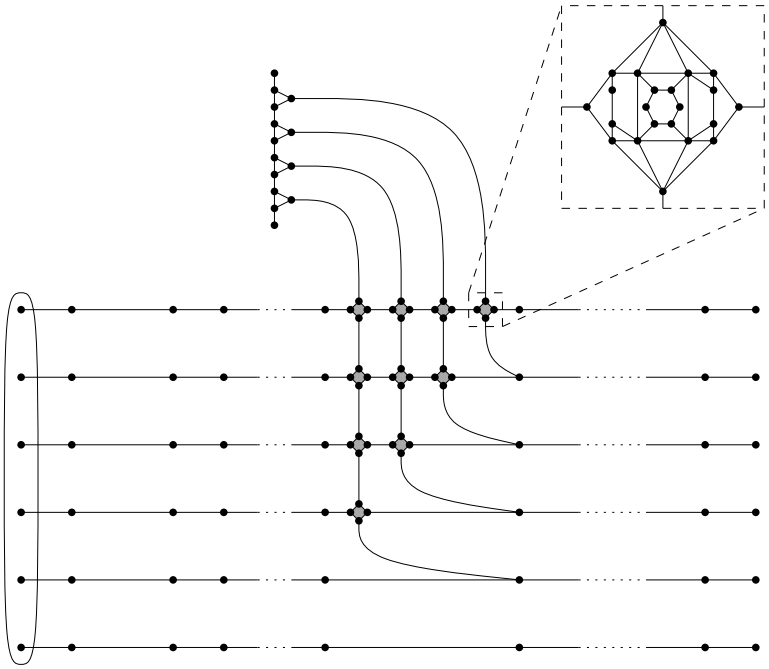
Award date:
2016

Technische Universiteit
Eindhoven
University of Technology

/ Department of Mathematics and Computer Science / Algorithms

Lower bounds for preprocessing algorithms based on protrusion replacement

Master Thesis



Jules Wulms

Committee:
Dr. Bart M. P. Jansen
Prof. Dr. Bettina Speckmann
Prof. Dr. Hans L. Bodlaender

Supervisor:
Dr. Bart M. P. Jansen

Eindhoven, Monday 4th July, 2016

[0] Image on the front cover shows a planar 6-boundaried graph of treewidth at most 12, details in Chapter 6

Abstract

In general, a kernelization algorithm is an efficient preprocessing procedure that reduces the size of the input to a difficult computational problem, without changing the answer. For optimization problems, we know how much the size of an optimum solution changes by reducing the size of the input using a kernelization algorithm. Garnero et al. [16] recently proposed a new kind of kernelization algorithm for optimization problems on planar graphs: The algorithm reduces a subgraph H of planar graph G, to a different subgraph H' called a *representative*. Subgraphs H and H' are connected to the remainder of G by t vertices. This reduction is done for multiple of such subgraphs in G, that are also connected to the remainder of G by t vertices.

The size of an optimum solution after reducing H to H', can be inferred by only looking at subgraph H and the representative H': We say that subgraph H is equivalent to representative H' if there is a constant c, such that the optimum solution of every graph G, that has H as a subgraph, changes by exactly c, by replacing H by H'. Therefore, the proposed kernelization algorithm reduces a subgraph H of G to an equivalent representative H'.

For the kernelization algorithm to be fast, one should be able to efficiently find a representative H' that is equivalent to a subgraph H in G. This is possible if subgraph H has bounded treewidth or bounded pathwidth.

Let R_t be a set of these subgraphs called representatives. Garnero et al. showed that an upper bound on the size of representatives in R_t is doubly-exponentially dependent on t, the number of vertices with which these subgraphs are connected to the remainder of a graph. We propose lower bounds for the size of these representatives, also dependent on t.

We give a lower bound of $\Omega(2^t/\sqrt{4t})$ on the number of vertices of a representative in such a set R_t for INDEPENDENT SET. This bound holds for sets of planar representatives with bounded treewidth/pathwidth. We also show that the equivalence relation that we explained before has at least $2^{2^t/\sqrt{4t}}$ equivalence classes for INDEPENDENT SET on general graphs. Furthermore, we improve on the results of Garnero et al. by giving an upper bound of 2^{2^t-1} on the number of equivalence classes for INDEPENDENT SET on general graphs. These bounds even hold for the number of equivalence classes for planar subgraphs of bounded treewidth/pathwidth.

Contents

Chapter 1

Introduction

Preprocessing is a widely known technique that is often used, either to reduce data in a data mining process [23], or to produce a *kernel* for an algorithmic problem. We are interested in the latter, namely *Kernelization* algorithms. The goal of this thesis is to develop lower bounds on the size of representatives, which are graph-theoretical objects used in a very general kernelization framework called *Meta kernelization*.

Background We will not stay within the bounds on classical computational complexity but work with techniques for *(Fixed) Parameterized Complexity* [11]. In contrast to classical complexity, in parameterized complexity we do not only have a problem instance I of size n, but it is coupled with a *parameter* k, resulting in a couple (I, k). We look into optimization problems where we want to find out whether there is an optimum solution to problem Π that has size at least/at most/exactly k. We want to find an algorithm for such a problem Π, that runs in $f(k) \cdot n^c$ time, where $f(k)$ is any function of input parameter k and c a constant. This is especially interesting for NP-hard problems, since these are not solvable in polynomial time. The parameterized version of an NP-hard problem, however, might be solvable in $f(k) \cdot n^c$ time for fixed k. We call problems *Fixed-Parameter Tractable*, or say they are in complexity class FPT, if there exists such an algorithm. The algorithms themselves are refered to as *FPT algorithms*.

Roughly speaking, a kernelization algorithm is an efficient preprocessing algorithm that reduces the size of the input to a problem Π. A kernelization algorithm gives a guarantee on the size of the resulting preprocessed instance, which is expressed in terms of input parameter k to measure the complexity of the instance. For concreteness, we mention the following well-known example. An instance (G, k) of the VERTEX COVER problem asks whether there is a set S of k vertices in graph G, such that every edge has at least one endpoint in S. While VERTEX COVER is an NP-complete problem, there is a polynomial-time algorithm that reduces any input (G, k) to an input (G', k) with the same answer and guarantees that G' has at most $2k$ vertices.

A more formal definition of kernelization has been given by Bodlaender et al. [2]:

Definition 1. A kernelization algorithm, or in short, a kernel for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that given $(I, k) \in \Sigma^* \times \mathbb{N}$, outputs in $p(|I| + k)$ time a pair $(I', k') \in \Sigma^* \times \mathbb{N}$ such that

- $(I, k) \in \Pi \Leftrightarrow (I', k)' \in \Pi$,

- $|I'|, k' \leq g(k)$,

where g is an arbitrary computable function and p a polynomial. Any function g as above is referred to as the size of the kernel.

A kernelization algorithm for a parameterized problem is said to produce a *polynomial kernel* if $g(k)$ is a polynomial function. If $g(k) = O(k)$ we speak of a *linear kernel*. Assuming we have a kernelization algorithm for a decidable problem Π, we can find an FPT algorithm to solve the problem as follows: We can find a kernel (I', k') of size $g(k)$ in $p(|I| + k)$ time, which we can brute force in $f(g(k))$ time for some arbitrary function f. Therefore we can solve the problem in $f(g(k)) + p(|I| + k)$ time.

Kernelization is an active research area, for which Lokshtanov et al. [21] gave a nice overview. A very important result by Alber et al. [1] is the discovery of a linear kernel for DOMINATING SET on planar graphs. Downey and Fellows [11] proved that there is no FPT algorithm for DOMINATING SET on general graphs. The fact that a linear kernel for DOMINATING SET on planar graphs was found, inspired other researchers to look for (linear) kernels on planar graphs. This led to the discovery of linear kernels for a variety of problems, for example FEEDBACK VERTEX SET [4], CYCLE PACKING [5], INDUCED MATCHING [18] and CONNECTED DOMINATING SET [22].

Guo and Niedermeier [17] also contributed in the discovery of linear kernels, but took it a step further by proposing a framework to find linear kernels for NP-hard problems on planar graphs. Their idea was to find a *region decomposition* of the planar graph, and reducing it using problem specific reduction rules. This was generalized by Bodlaender et al. [3] to the meta kernelization framework. They showed that for various NP-complete graph problems on planar graphs, there *exists* a kernelization algorithm that reduces any instance (G, k) to an instance (G', k') with the same answer, which has $O(k)$ vertices. Note that they only proved that there exists such an algorithm, no explicit algorithm was proposed, nor an analysis was given of the constants hidden in the $O(k)$ upper bound on the kernel size. The crucial difference from previous work is that problem independent rules can now be used to find kernels, instead of having to find new rules for every problem that kernelization is used on.

The concepts from meta kernelization that are important for this thesis are *Boundaried graphs*, *Protrusions* and *Finite Integer Index* (FII). While not going into too much detail here, we can say that t-boundaried graphs are subgraphs of a graph G, that are connected to the remainder of G by a numbered set of t vertices, called the boundary. When we talk about t-protrusions, we mean t-boundaried graphs, that have their treewidth bounded by t. This means that a protrusion can be unbounded in size, but can still be efficiently reduced to an *equivalent* graph gadgets of constant size, for problems that have the property FII. Roughly speaking, two t-boundaried graphs F and H are equivalent for an optimization problem Π, if there exists a constant c, such that for every instance of problem Π in which F occurs as a subgraph, replacing subgraph F by subgraph H changes the optimal solution value by exactly c, regardless of what the rest of the graph looks like.

In meta kernelization we find a *protrusion decomposition* of $O(k)$ protrusions in a graph G, and replace all protrusions by equivalent graph gadgets of constant size, to get a kernel G' of $O(k)$ vertices. This reduction via a protrusion decomposition only works efficiently for problems that have FII and on graphs embeddable in a surface of bounded genus, for example graphs embeddable on a sphere (genus 0/planar) or torus (genus 1). Refer to Chapter 2 for more elaborate definitions, or to the paper by Bodlaender et al. [3] for full details on meta kernelization.

As we already pointed out, first it was only proved that meta kernelization existed [3], but no algorithm was proposed to construct such kernels. This was due to the fact that no algorithm was given to find the equivalent graph gadgets of constant size, which we will call *Representa-*

tives. Fomin et al. [14] and Kim et al. [19] extended the initial meta kernelization results to other graph classes, graphs excluding a fixed minor and graphs excluding a fixed topological minor respectively. Follow-up work by Garnero et al. [16] shows how to construct a kernel using meta kernelization, and established upper bounds on the size of the constants hidden in the bound on the kernel size of $O(k)$ vertices. The authors show how to find representatives in a structural way, which also allows them to find an upper bound on the size of these representatives: Let R_t be a set of representatives that have boundary size t, one for each equivalence class. The upper bound on the size of a representative in R_t grows doubly-exponentially with t or in some cases even worse.

Problem setting The goal of this project is to find a lower bound on the number of vertices of a representative in any set R_t. We propose a process to find this lower bound and go through all the steps for the INDEPEDENT SET problem (IS), which looks for a maximum set of vertices of which none are adjacent. In this way, we can compare the existing upper bounds for meta kernelization on INDEPENDENT SET with our lower bounds, and see if our techniques look promising to find lower bounds for other problems like DOMINATING SET.

Our results We first find a definition of equivalence that is tailored to INDEPENDENT SET on t-boundaried graphs. For this definition we propose a nice representation of the equivalence classes, in the form of monotone functions which we call *t-representative functions*. For every function f in the set of t-representative functions $\mathcal{F}_t$, we construct a general graph that is not equivalent to any other graph we construct for a function $f' \in \mathcal{F}_t$, that is distinct from f. This results in a tight lower bound on the number of equivalence classes for general t-boundaried graphs, if we are able to determine the size of $\mathcal{F}_t$. The definition of equivalence for INDEPENDENT SET on t-boundaried graphs via t-representative functions also allows us to prove an upper bound of 2^{2^t-1} distinct equivalence classes, which improves on the upper bound given by Garnero et al. [16].

Since we are interested in the number of equivalence classes for protrusions, which have bounded treewidth, we come up with a construction for t-boundaried graphs of treewidth at most $t+2$, to find a lower bound in the same way as for general graphs. We can do this construction for the set of *monotone Boolean functions* that are not always zero. The number of distinct monotone Boolean functions on t variables is counted by the t-th *Dedekind number* $M(t) \geq 2^{\binom{t}{t/2}} \geq 2^{2^t/\sqrt{4t}}$. In this way we find a lower bound of $M(t)-1$ on the number of equivalence classes for t-boundaried graphs of treewidth at most $t+2$, since there is exactly one monotone Boolean function that is always zero.

For the results to be relevant for meta kernelization, they should apply to graphs of a more restricted graph class: graphs embeddable in a surface of bounded genus. We extend our previous result to planar graphs by showing that our t-boundaried graphs of treewidth at most $t+2$ can be planarized. The planar t-boundaried graphs that are the result of this planarization have treewidth at most $t+6$ and all $M(t)-1$ graphs are still in different equivalence classes.

From the previous results we can conclude that any set of representatives R_t, that consists of one representative for each equivalence class, has size at least $M(t)-1$, if we consider planar graphs only. We can now count how many distinct planar t-protrusions of size at most x there are, and if there are less than $M(t)-1$, we can conclude that there must be a representative in R_t that has size $x+1$. Using this counting argument, we prove that there is a representative in any set R_t that has $\Omega(\log M(t))$ vertices.

Since the size of the representatives is a big factor in constants of the upper bounds for meta kernelization, finding this lower bound on the size of these representatives is a meaningful result:

It shows how much room for improvement there is for the (upper bound on the) kernel size that was proposed by Garnero et al. [16]. On top of that, the result can stand on its own, since there are other (preprocessing) algorithms [13, 12] that use protrusion replacement. Finding lower bounds on the number of equivalence classes and the size of a representative can allow new results for these applications as well. To our knowledge, there are no results present in literature of lower bounds on the number of equivalence classes nor of lower bounds on the size of a representative.

Organization The thesis is structured as follows. In Chapter 2 we give preliminaries and definitions that will be used throughout the other chapters. Chapter 3 works towards the definition of equivalence for INDEPENDENT SET. In Chapter 4 we use the new definition of equivalence to find a lower and upper bound on the number of equivalence classes for general t-boundaried graphs. Chapter 5 proves a lower bound on the number of equivalence classes for t-boundaried graphs of bounded treewidth, and these results are extended to planar t-boundaried graphs of bounded treewidth in Chapter 6. Finally we use a counting argument in Chapter 7 to show that any set R_t that represents all equivalence classes containing a planar graph of treewidth at most $t + 6$, must contain at least one graph of size $\Omega(\log M(t))$.

Chapter 2

Preliminaries

When we talk about a graph $G = (V, E)$, we mean a simple graph, unless stated otherwise: G is undirected, unweighted and has no self loops or multiple edges between any pair of vertices. The set of vertices of G is V, while its edge set is E. We denote an edge between vertices u and v by the unordered pair $\{u, v\}$. The intersection $V' \cap G$ of a graph $G = (V, E)$ and a vertex set $V' \subseteq V$ will be used to indicate the set of vertices in the subgraph that is induced by V'. A *Boolean function* is a function of the form $f : \{0,1\}^n \to \{0,1\}$. Sometimes we use functions whose input is dependent on the subset S of ordered vertex set $B = \{v_1, \ldots, v_n\}$. This corresponds to a function with $\{0,1\}^n$ as input by setting the i-th variable of the input to 1 if $v_i \in S$, and otherwise setting the i-th variable to 0. An optimum solution for problem Π on graph G will be denoted by $\text{OPT}_\Pi(G)$.

Boundaried Graphs A t-boundaried graph is a graph $G = (V, E)$ with an ordered vertex set $B = \{v_1, \ldots, v_t\}$ for which holds that $B \subseteq V$. We call B the boundary of G. This definition of boundaried graphs corresponds to the definition of *Terminal graphs* [10], instead of the definition for boundaried graphs used by Bodlaender et al. [3].

If we have two t-boundaried graphs G and F, with boundaries $B_G = \{a_1, \ldots, a_t\}$ and $B_F = \{b_1, \ldots, b_t\}$ respectively, we can create graph $G \oplus F$, with boundary $B = \{v_1, \ldots, v_t\}$ by *Gluing* G and F together. We get the new graph $G \oplus F$ by taking the disjoint union of G and F and gluing the boundaries of G and F onto each other: Every vertex $v_i \in B$ is created by identifying $a_i \in B_G$ and $b_i \in B_F$. This means that v_i is connected to the same vertices as a_i and b_i are connected to in G and F respectively. If there are multiple edges between a pair of vertices in $G \oplus F$ that are introduced by the glueing, then those are removed.

Meta Kernelization A couple of concepts that we introduced in the previous chapter can be defined more formally. We will use most definitions as defined by Bodlaender et al. [3], with some small adjustments.

Equivalence of boundaried graphs for graph problem Π is defined as follows. We say that t-boundaried graphs G_1, G_2 are equivalent, $G_1 \equiv_\Pi G_2$, if and only if there exists *transposition constant* c, such that for every pair of boundaried graph F and parameter k holds that

$$(G_1 \oplus F, k + c) \in \Pi \Leftrightarrow (G_2 \oplus F, k) \in \Pi.$$

We can change this definition slightly to make it easier to use and understand: for optimization problems, there is a maximum/minimum k for which $(G_1 \oplus F, k) \in \Pi$ still holds. In that case

also $(G_2 \oplus F, k + c) \in \Pi$, will still hold. This means that k is the size of a optimum solution for the problem instance $(G_1 \oplus F, k)$ of this problem Π. More extreme values of k will result in $(G_1 \oplus F, k) \notin \Pi$. This leads to an equivalent definition of equivalence where $G_1 \equiv_\Pi G_2$ if and only if there exists constant c such that for all boundaried graphs F

$$\text{OPT}_\Pi(G_1 \oplus F) = \text{OPT}_\Pi(G_2 \oplus F) + c.$$

The definition for *finite integer index* that we are going to use was first introduced by Bodlaender and van Antwerpen-de Fluiter [6, 10]: The equivalence relation $\equiv_\Pi$ of t-boundaried graphs on optimization problem Π has finite integer index when the number of equivalence classes is finite for fixed boundary size t. The optimization problem for INDEPENDENT SET has finite integer index [10].

A *t-protrusion* is a t-boundaried graph, which has its treewidth bounded by t. One can efficiently solve problems on graphs of bounded treewidth, and one of those problems is finding and replacing a t-protrusion in a graph G, by an equivalent graph gadget of constant size, for problems that have FII [3].

Treewidth and Pathwidth We will denote the treewidth of a graph G by $\mathbf{tw}(G)$ and its pathwidth by $\mathbf{pw}(G)$. The treewidth of a graph has many equivalent definitions and we will use the definition by a *tree decomposition*. A tree decomposition for a graph $G = (V, E)$ is a tree T consisting of sets of vertices $X_1, \ldots, X_n$ called bags, with the following properties.

- $\bigcup_{1 \leq i \leq n} X_i = V$
- The subgraph of T induced by all bags X_i in which a vertex $v \in V$ occurs is a connected tree. Equivalently, if $v_{\in} X_i$ and $v \in X_j$ then all bags in the (unique) path between X_i and X_j contain v.
- For every $\{u, v\} \in E$ there is a bag X_i for which $u, v \in X_i$.

The width of a tree decomposition is the size of its largest bag minus one. The treewidth of graph G $\mathbf{tw}(G)$ is defined as the minimum width of any tree decomposition for G.

The pathwidth of a graph can be defined in a similar way, via a *path decomposition*. The structure build from the bags is now a path P, and the subgraph induced by all bags X_i that contain a vertex v is a connected path. For every graph G holds that the pathwidth $\mathbf{pw}(G)$ is the minimum width of any path decomposition for G. We know that $\mathbf{tw}(G) \leq \mathbf{pw}(G)$: This can be easily seen when comparing tree and path decompositions. Every path decomposition is also a tree decomposition but not the other way around. A bound on the treewidth of a graph G can therefore be proved by looking at its pathwidth.

To show that a graph G has bounded pathwidth, we can use a *Mixed Search Game* [24]. When doing a mixed search, we see graph G as a network of contaminated tunnels. Initially all edges are contaminated and the goal is to clean all the edges using cleaners. The cleaners can reside on vertices and slide along edges. When there is a cleaner on both endpoints of an edge, or a cleaner slides along an edge, that edge will be cleaned. Clean edges will become contaminated again if there is a path from a contaminated edge to a clean edge, with no cleaners on its vertices. Takahashi et al. [24] showed that for $\mathbf{ms}(G)$, the minimum number of simultaneous searches that are sufficient to clean graph G, holds that $\mathbf{pw}(G) \leq \mathbf{ms}(G) \leq \mathbf{pw}(G) + 1$. In particular, this means that for a graph G we know that $\mathbf{pw}(G)$ is at most the number of cleaners we need to execute any strategy that wins the mixed search game on G.

Miscellaneous For real $x \neq 1$ the following summation is a *geometric series* [9] and has the value $\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$. We can then derive that

$$\sum_{k=m}^{n} x^k = \sum_{k=0}^{n} x^k - \sum_{k=0}^{m-1} x^k = \frac{x^{n+1}-1}{x-1} - \frac{x^m-1}{x-1} = \frac{x^{n+1}-x^m}{x-1} = \frac{x \cdot x^n - x^m}{x-1}. \tag{2.1}$$

Furthermore, we can use *Stirling's approximation* to find a bound on certain binomial coefficients

$$\binom{2n}{n} \geq \frac{2^{2n-1}}{\sqrt{n}} = \frac{2^{2n}}{2\sqrt{n}} = \frac{4^n}{\sqrt{4n}}.$$

This approximation can be used to find a more tangible representation of the Dedekind numbers: $M(t) \geq 2^{\binom{t}{\lfloor t/2 \rfloor}} \geq 2^{2^{2\lfloor t/2 \rfloor}/\sqrt{4\lfloor t/2 \rfloor}} \geq 2^{2^t/\sqrt{4t}}$.

Chapter 3

Equivalence for Independent Set

As we have seen in Chapter 2, equivalence classes for boundaried graphs can be defined using the size of an optimum solution on the boundaried graph, glued to any other boundaried graph. In this chapter we find a new definition of equivalence classes, that expresses equivalence for INDEPENDENT SET more precisely.

Before we can elaborate on the specialized definition of equivalence, we formalize when an independent set is *consistent* with a set of vertices S:

Definition 2. Given a t-boundaried graph G with boundary vertices $B = \{v_1, v_2, \ldots, v_t\}$, we call an independent set X for G *consistent* with set $S \subseteq B$ if $X \cap B = S$

We can define a function h_G on such a set $S \subseteq B$ for t-boundaried graph G, which gives the size of the maximum independent set that is consistent with S:

Definition 3. The function h_G on t-boundaried graph G is defined as

$$h_G(S) := \max\{|X| \mid X \text{ is an independent set in } G \text{ and } X \cap B = S\}$$

This function has nice properties with respect to optimum solutions and the gluing operation on boundaried graphs $\oplus$.

Lemma 1. *For any t-boundaried graph G holds that*

1. $\max_{S \subseteq B} h_G(S) = \text{OPT}_{\text{IS}}(G)$
2. $h_G(S) + h_F(S) - |S| = h_{G \oplus F}(S)$
3. $\max_{S \subseteq B}\{h_G(S) + h_F(S) - |S|\} = \text{OPT}_{\text{IS}}(G \oplus F)$

Proof. Every property can be proved separately:

1. If we observe that an optimum solution is consistent with some $S \subseteq B$, then this follows from Definition 3.
2. Gluing two t-boundaried graphs together unifies their boundaries. Therefore, the size of a maximum independent set of a glued graph consistent with $S \subseteq B$, is equal to the sum of maximum independent sets consistent with S of the graphs that are glued together minus $|S|$, since the two boundaries are unified.

3. Combining the previous two properties, we can derive a third property which interleaves gluing and optimum solutions:
$\max_{S\subseteq B}\{h_G(S) + h_F(S) - |S|\} = \max_{S\subseteq B} h_{G\oplus F}(S) = \text{OPT}_{\text{IS}}(G \oplus F)$ □

We define another function f_G on set $S \subseteq B$ for t-boundaried graph G, which is closely related to h_G:

Definition 4. The function f_G on t-boundaried graph G is defined as

$$f_G(S) := \max\{|X| \mid X \text{ is an independent set in } G \text{ and } X \cap B \subseteq S\}$$

The relation between f_G and h_G is $f_G(S) = \max_{S'\subseteq S} h_G(S)$. We will use f_G to define equivalence on boundaried graphs for independent set.

Lemma 2. *Let G_1, G_2 be two t-boundaried graphs with boundary B, then $G_1 \equiv_{\text{IS}} G_2$ if and only if there exists a constant c such that $f_{G_1}(S) = f_{G_2}(S) + c$ for all $S \subseteq B$.*

The remainder of this section will work towards proving Lemma 2. We will prove both sides of the bi-implication separately. For one of the proofs we need a specific relation between f_G and the size of a maximum independent set under the gluing operation graphs, which we will prove first.

Lemma 3. *For all t-boundaried graphs G and F holds that*

$$\max_{S\subseteq B}\{f_G(S) + f_F(S) - |S|\} = \text{OPT}_{\text{IS}}(G \oplus F).$$

Proof. We prove the equivalence by showing that both $\leq$ and $\geq$ hold for this relation:

$(\geq)$ Since $f_G(S) = \max_{S'\subseteq S} h_G(S)$, we can conclude that $f_G(S) \geq h_G(S)$ for all t-boundaried graphs G and $S \subseteq B$. Using this fact together with the third point of Lemma 1, we know that

$$\max_{S\subseteq B}\{f_G(S) + f_F(S) - |S|\} \geq \max_{S\subseteq B}\{h_G(S) + h_F(S) - |S|\} = \text{OPT}_{\text{IS}}(G \oplus F).$$

$(\leq)$ We look at $f_G(S^*) + f_F(S^*) - |S^*|$, which uses a set $S^* \subseteq B$, for which $\max_{S\subseteq B}\{f_G(S) + f_F(S) - |S|\}$ is maximized. We are going to modify a solution X_G for G of size $f_G(S^*)$ and a solution X_F for F of size $f_F(S^*)$, both consistent with a subset of S^*. Combining these modified solutions will result in a valid solution for $G \oplus F$ of size at least $f_G(S^*) + f_F(S^*) - |S^*|$.

The set X_G is an independent set for G and X_F is an independent set for F. This means that they both can contain vertices in their boundaries. However, in $G \oplus F$ these separate boundaries are unified into a single boundary. As a result, we cannot directly conclude that there is a valid independent set for $G \oplus F$ of size $f_G(S^*) + f_F(S^*)$.

We want to show that there is an independent set for $G \oplus F$ of size at least $f_G(S^*) + f_F(S^*) - |S^*|$, which means we can remove up to $|S^*|$ vertices from X_G and X_F to find a valid solution for $G \oplus F$.

Assume without loss of generality that $V(G) \cap V(H) = B$ (the only vertices common to both graphs are those in the boundary). Therefore, to find an independent set for $G \oplus F$, we need to remove vertices from the boundaries of G and F, $B_G = X_G \cap B$ and $B_F = X_F \cap B$ respectively. Since X_G and X_F are both consistent with a subset of S^*, we know that $B_G \cup B_F \subseteq |S^*|$.

We want to remove sets $B_G \setminus B_F$ and $B_F \setminus B_G$, since combining X_G and $B_F \setminus B_G$ leads to a set that is not an independent set for $G \oplus F$. If it would be an independent set, then X_G is not an optimum: we can find a bigger solution for G by adding $B_F \setminus B_G$. Similarly, X_F and $B_G \setminus B_F$ cannot be combined to find an independent set.

Consider the multiset $X := X_G \cup X_F$, which contains the vertices of $B_G \cap B_F$ twice. Consider the effect of removing from X one occurrence of each vertex in $(B_G \setminus B_F) \cup (B_F \setminus B_G) \cup (B_F \cap B_G)$. Let X' denote the result. Since the only vertices that can occur more than once in X are those in $B_F \cap B_G$, we have that no element occurs more than once in X' and so it is a simple set. Since $B_G \cap B_F$ was part of both X_G and X_F, we know that X' is an independent set for $G \oplus F$. We removed $B_G \setminus B_F$, $B_F \setminus B_G$ and $B_G \cap B_F$, which combined are $B_G \cup B_F$. We already established that $B_G \cup B_F \subseteq S^*$, so we removed at most $|S^*|$ vertices from X_G and X_F to find an independent set for $G \oplus F$ of size at least $f_G(S^*) + f_F(S^*) - |S^*|$.

We had chosen S^* in such a way that $f_G(S^*) + f_F(S^*) - |S^*|$ was the maximum for $\max_{S \subseteq B}\{f_G(S) + f_F(S) - |S|\}$. This means that we can always find an independent set of size at least $\max_{S \subseteq B}\{f_G(S) + f_F(S) - |S|\}$. Since any independent set for $G \oplus F$ is smaller or equal in size to an optimum solution, we can conclude that $\max_{S \subseteq B}\{f_G(S) + f_F(S) - |S|\} \leq \text{OPT}_{\text{IS}}(G \oplus F)$. □

Now that we have proved Lemma 3, we can proceed with Lemmata 4 and 5, which both prove one direction of Lemma 2.

Lemma 4. *Let G_1, G_2 be two t-boundaried graphs with boundary B. If there exists a constant c such that $f_{G_1}(S) = f_{G_2}(S) + c$ for all $S \subseteq B$, then $G_1 \equiv_{\text{IS}} G_2$.*

Proof. Take an arbitrary t-boundaried graph F and glue it to G_1 to get $G_1 \oplus F$. Using Lemma 3 and the assumption that $f_{G_1}(S) = f_{G_2}(S) + c$ for all $S \subseteq B$ and some constant c, we show the following:

$$\begin{aligned}
\text{OPT}_{\text{IS}}(G_1 \oplus F) &= \max_{S \subseteq B}\{f_{G_1}(S) + f_F(S) - |S|\} \\
&= \max_{S \subseteq B}\{f_{G_2}(S) + c + f_F(S) - |S|\} \\
&= \max_{S \subseteq B}\{f_{G_2}(S) + f_F(S) - |S|\} + c \\
&= \text{OPT}_{\text{IS}}(G_2 \oplus F) + c
\end{aligned}$$

Since we have chosen F as an arbitrary t-boundaried graph, we can conclude that $G_1 \equiv_{\text{IS}} G_2$. □

Lemma 5. *Let G_1, G_2 be two t-boundaried graphs with boundary $B = \{v_1, v_2, \ldots, v_t\}$. If $G_1 \equiv_{\text{IS}} G_2$, then there exists a constant c such that $f_{G_1}(S) = f_{G_2}(S) + c$ for all $S \subseteq B$.*

Proof. To prove that there exists a constant c such that $f_{G_1}(S) = f_{G_2}(S) + c$ for all $S \subseteq B$, we will define t-boundaried graphs F_S, which can be glued to any other t-boundaried graph G. By establishing a relation between $\text{OPT}_{\text{IS}}(G \oplus F_S)$ and $f_G(S)$ we can prove the lemma.

For $S \subseteq B = \{v_1, v_2, \ldots, v_t\}$ we define the t-boundaried graph F_S with boundary B as the result of the following process: starting from an edgeless graph with vertex set B, for each $v_i \notin S$ add u_i, u_i' and the edges $\{v_i, u_i\}$ and $\{v_i, u_i'\}$ to F_S.

If we glue F_S to any other t-boundaried graph G, an optimum independent set for $G \oplus F_S$ will never include a vertex $v_i \in S$, since we get a bigger independent set by taking u_i and u_i' instead of v_i. As a consequence, u_i and u_i' will always be included in an optimum solution, since v_i will not.

Subtracting the number of u vertices (which is equal to $2(t - |S|)$) from $\text{OPT}_{\text{IS}}(G \oplus F_S)$ will result in the size of a maximal independent set for G, that is consistent with S or a subset of S. This is exactly the definition of $f_G(S)$, which means that:

$$f_G(S) = \text{OPT}_{\text{IS}}(G \oplus F_S) - 2(t - |S|)$$

Using the above relation and the fact that there exists a constant c such that $\text{OPT}_{\text{IS}}(G_1 \oplus F_S) = \text{OPT}_{\text{IS}}(G_2 \oplus F_S) + c$ (by definition of equivalence), we can show for arbitrary $S \subseteq B$ that:

$$\begin{aligned} f_{G_1}(S) &= \text{OPT}_{\text{IS}}(G_1 \oplus F_S) - 2(t - |S|) \\ &= \text{OPT}_{\text{IS}}(G_2 \oplus F_S) + c - 2(t - |S|) \\ &= \text{OPT}_{\text{IS}}(G_2 \oplus F_S) - 2(t - |S|) + c \\ &= f_{G_2}(S) + c \end{aligned}$$

Since we have chosen $S \subseteq B$ as an arbitrary subset of boundary B, we know that $f_{G_1}(S) = f_{G_2}(S) + c$ for all $S \subseteq B$. Note that this constant c is the same as the c for which equivalence $G_1 \equiv_{IS} G_2$ holds. □

Combining Lemmata 4 and 5 proves Lemma 2.

Chapter 4

Equivalence classes for general graphs

Now that we have a definition for equivalence on t-boundaried graphs that is more tailored to INDEPENDENT SET, we will use it to define general t-boundaried graphs that belong in a specific equivalence class.

Lemma 2 has given us a way to define an equivalence class for a graph G using a function f_G from $S \subseteq \{v_1, v_2, \ldots, v_t\}$ to a positive integer. Before we are looking into graphs, there are some more properties of function f_G that we are going to prove:

Lemma 6. *Let G be a t-boundaried graph with boundary B. The function f_G is a monotone function: For sets S and S', where $S \subset S' \subseteq B$, holds that $f_G(S) \leq f_G(S')$.*

Proof. Let G be an arbitrary t-boundaried graph with boundary B, and let S and S' be two sets such that $S \subset S' \subseteq B$. By Definition 4 $f_G(S)$ is the maximum of all independent sets X, such that $X \cap B \subseteq S$. Since we assumed that $S \subset S'$, for each of those X also holds that $X \cap B \subseteq S'$. Therefore the maximum of all independent sets X' for which holds that $X' \cap B \subseteq S'$ is at least as big as $f_G(S)$, hence $f_G(S') \geq f_G(S)$. □

Lemma 7. *Let G be a t-boundaried graphs with boundary B. For sets S and S', where $S \subset S' \subseteq B$ and $|S| + 1 = |S'|$, holds that $f_G(S) + 1 \geq f_G(S')$.*

Proof. Let G be an arbitrary t-boundaried graph. Look at an independent set X for G that has size $f_G(S')$ and is consistent with set $S_X \subseteq S'$, and consider the following case distinction:

- If X does not contain vertex $v \in (S' \setminus S)$, then $S_X \subseteq S$. In this case $f_G(S) \geq f_G(S')$, and hence $f_G(S) = f_G(S')$ by Lemma 6.
- If X contains the vertex $v \in (S' \setminus S)$, then $X' = X \setminus \{v\}$ is also an independent set for G. By construction of X' and the fact that $|S| + 1 = |S'|$, we know that X' is consistent with a set $S_{X'}$, for which holds that $S_{X'} \subseteq S$. This means that $f_G(S) + 1 \geq f_G(S')$.

Hence, for every t-boundaried graph G, it holds that $f_G(S) + 1 \geq f_G(S')$. □

We know by Lemma 6 that every function f_G is monotone, and Lemma 7 tells us that it can only increase its function value by at most one, with respect to the value for a subset that is one smaller. This restricts the number of functions, for which there can possibly be a different equivalence class.

If we have two t-boundaried graphs G_1, G_2 with boundary B and $G_1 \equiv_{\text{IS}} G_2$, we know that there exists a constant c such that for every S, $f_{G_1}(S) = f_{G_2}(S) + c$. At $S = \emptyset$, $f_{G_1}(S) = x$, while $f_{G_2}(S) = x + c$. For some $S' \supset S$, the optimum solution for G_1 might increase and $f_{G_1}(S') = x + 1$, and for G_2 we get $f_{G_2}(S') = x + c + 1$. However, this means that $f_{G_1}(S) = f_{G_2}(S)$ for all $S \subseteq B$, if $f_{G_1}(\emptyset) = f_{G_2}(\emptyset)$.

We can look at equivalence classes from a different perspective, and see them as sets of functions that are monotone and increase by at most one. We can take a particular function from each set/equivalence class as its representative, by taking all functions that have the same value for $S = \emptyset$.

Definition 5. We call a function f on a set $S \subseteq B = \{v_1, v_2, \ldots, v_t\}$ a t-representative if it has the following properties.

1. $f(\emptyset) = t$
2. For sets S and S', where $S \subset S' \subseteq B$, holds that $f(S) \leq f(S')$.
3. For sets S and S', where $S \subset S' \subseteq B$ and $|S| + 1 = |S'|$, holds that $f(S) + 1 \geq f(S')$.

The second and the third property ensure that such a function f has the properties in Lemmata 6 and 7, while the first property ensures that this representative function outputs t for $S = \emptyset$.

Before we move on to defining those graphs, we first introduce the notion of *false twins*.

Definition 6. Given graph $G = (V, E)$, we call two vertices $v_1, v_2 \in V$ *false twins* if they have the same open neighborhood $N(v_1) = N(v_2)$.

We call an inclusion-wise maximal set of vertices which are pairwise all false twins a *twin class*. Observe that being false twins is an equivalence relation, which means we can partition a graph into twin classes. The following lemma proves an important property of twin classes with respect to maximum independent sets.

Lemma 8. *Given graph $G = (V, E)$, twin class $T \subseteq V$ and maximal independent set X, either $T \subseteq X$ or $T \cap X = \emptyset$.*

Proof. Assume we have such a maximal independent set X, for graph G. If there is a vertex $t \in T$ for which holds that $t \in X$ then $T \subseteq X$, otherwise we can find a bigger independent set which does include the vertices in set $T \setminus X$ and X is not a maximal solution. On the other hand, there is not necessarily a vertex $t \in T$ such that $t \in X$, so in this case $T \cap X = \emptyset$. □

Using t-representative functions and false twins, we can define a graph G that has a particular function f_G.

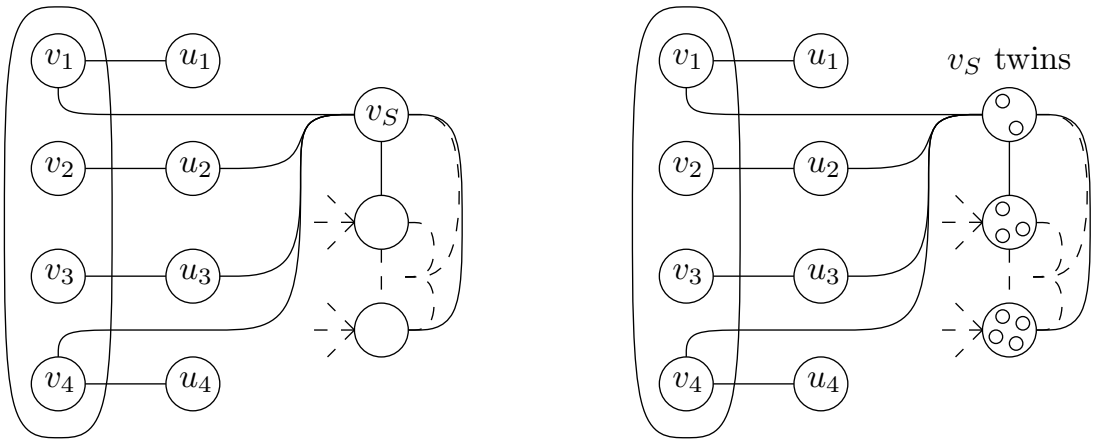
Lemma 9. *For every t-representative function f, there exists a t-boundaried graph G with boundary $B = \{v_1, v_2, \ldots, v_t\}$, such that $f_G(S) = f(S)$ for every $S \subseteq B$.*

Proof. Assume we have an arbitrary t-representative function f. We construct a t-boundaried graph G, and make sure that $f_G(S) = f(S)$ for every $S \subseteq B$, using the following process:

Starting from an edgeless graph with vertex set B, add a vertex u_i for every $v_i \in B$ and add an edge $\{u_i, v_i\}$ for every $1 \leq i \leq t$. For every S, where $f(S) > t$, we modify G in the following way: add a vertex v_S and add edges depending on whether S contains v_i for $1 \leq i \leq t$:

- If $v_i \in S$, add edge $\{u_i, v_S\}$.
- If $v_i \notin S$, add edge $\{v_i, v_S\}$.

When we have added all vertices v_S, we add edges between every pair of them, creating a single big clique. Figure 4.1a shows an example of a constructed graph.



(a) Simple graph

(b) Graph with false twins

Figure 4.1: Examples of constructed graph with boundary $\{v_1, v_2, v_3, v_4\}$ and v_S with $S = \{v_2, v_3\}$ (among other vertices)

The final part of the construction of our graph G adds false twins for the vertices in the big clique. For every vertex v_S we ensure there are $f(S) - t$ false twins (including v_S). Figure 4.1b shows the previous example after false twins have been introduced.

We want to prove that for this graph G holds that $f_G(S) = f(S)$ for all $S \subseteq B$. We will do this by showing that both $\leq$ and $\geq$ hold:

$(\geq)$ If we have a maximal independent set X for which holds that $X \cap B = S$ and for every $v_i \notin S$ we have $u_i \in X$, then by construction of G, $v_S \in X$ and by Lemma 8 all its false twins as well. Note that v_S and its twins are the only vertices that can be in an independent set together with this combination of v_i and u_i. There are $f(S) - t$ false twins and we have either $v_i \in X$ or $u_i \in X$ for $1 \leq i \leq t$, thus we know that $|X| = f(S)$. Since $f_G(S)$ is the maximum over all independent sets that are consistent with S or a subset, we can conclude that $f_G(S) \geq |X| = f(S)$

$(\leq)$ To prove $f_G(S) \leq f(S)$ for all $S \subseteq B$, we will use Definition 4 and show that for arbitrary $S \subseteq B$ and every independent set X where $X \cap B \subseteq S$ holds that $|X| \leq f(S)$. We split this in a case distinction for $X \cap B = S$ and $X \cap B \subset S$:

For arbitrary $S \subseteq B$, we look at an independent set X, with $X \cap B = S$. As we have seen before, if for every $v_i \notin S$ we have $u_i \in X$, then a maximal independent set has size at most $f(S)$. However, for this $S' \supset S$ we might have that $u_i \notin X$ for every $v_i \notin S'$. In this case we can use $v_{S'}$ and its false twins, for some $S' \supset S$. Using point 3 in Definition 5 and applying transitivity we get that $f(S) + |S' \setminus S| \geq f(S')$. This means there are at most $f(S) + |S' \setminus S| - t$ false twins for $v_{S'}$. Observe that we do not have either $v_i \in X$ or $u_i \in X$ for $1 \leq i \leq t$, since $u_i \notin X$ for every $v_i \notin S'$, resulting in $t - |S' \setminus S|$ vertices. This means that $|X| \leq f(S) + |S' \setminus S| - t + t - |S' \setminus S| = f(S)$.

Now look at an independent set X with $X \cap B = S' \subset S$. Again, if for every $v_i \notin S'$ we have $u_i \in X$, then a maximal independent set $|X|$ has size at most $f(S')$. Using point 2 in Definition 5, we know that $f(S') \leq f(S)$. If it is not the case that for every $v_i \notin S'$ we have $u_i \in X$, we can repeat the proof for the previous case to find out that $|X| \leq f(S')$. In both cases we have that $|X| \leq f(S)$.

We can now conclude that for arbitrary t-representative function f, there exists a t-boundaried graph G such that $f_G(S) = f(S)$ for all $S \subseteq B$. □

Lemma 9 shows us that we can define a graph for an arbitrary t-representative function f, such that $f_G(S) = f(S)$ for all $S \subseteq B$. We call t-representative functions f_1, f_2 distinct if for some $S \subseteq B$ we have that $f_1(S) \neq f_2(S)$. Therefore, if we look at the set $\mathcal{F}$ of distinct t-representative functions, we have a function of each equivalence class, all of which we can construct a graph for. Hence we can conclude that $|\mathcal{F}|$ is a tight lower bound on the number of equivalence classes for independent set on general t-boundaried graphs.

The final part of this section focuses on determining the size of $\mathcal{F}$. We are unable to find a precise number for $|\mathcal{F}|$, but we find a lower bound, and an upper bound, to show that the lower bound has the same order of magnitude.

Remember that a Dedekind number counts the number of distinct monotone Boolean functions, and thus $M(t)$ is the number of distinct monotone Boolean functions on t Boolean inputs.

Lemma 10. *For the set $\mathcal{F}_t$ of distinct t-representative functions holds that $|\mathcal{F}_t| \geq M(t) - 1$.*

Proof. A function $f \in \mathcal{F}_t$ can be seen as a monotone function with t Boolean variables $v_1, v_2, \ldots, v_t$ as input. An input S for f corresponds to setting every $v_i \in S$ to 1 as explained in Chapter 2. A subset of $\mathcal{F}_t$ is the set of functions whose output is also Boolean. We can see this if we map output t to 0 and output $t+1$ to 1. Note that for $t = 0$ and $t = 1$, this subset of Boolean functions is actually the whole set $\mathcal{F}$.

Consider the set $\mathcal{M}$ of all distinct monotone Boolean functions on t variables. For each function $g \in \mathcal{M}$ that is not always 1, consider the function $h(S) := g(S) + t$. Then $h(S)$ satisfies all properties of a t-representative function in Definition 5, so $h \in \mathcal{F}_t$. Since there are $M(t) - 1$ monotone Boolean functions that are not always 1, it follows that $\mathcal{F} \geq M(t) - 1$. □

This lower bound is also a lower bound on the number of equivalence classes for INDEPENDENT SET on general t-boundaried graphs. The bound will be the basis for remaining work: We now have a lower bound for general graphs, but in meta-kernelization we will only use equivalence to find equivalent graph gadgets for boundaried graphs of bounded treewidth, that can be embedded in a surface of bounded genus. Chapter 5 will elaborate on graphs with bounded treewidth, while Chapter 6 extends this to planar bounded treewidth graphs.

We conclude this section with an upper bound for $|\mathcal{F}_t|$, which directly is an upper bound on the number of equivalence classes. The upper bounds is there to put the lower bound in perspective.

Lemma 11. *For the set $\mathcal{F}_t$ of distinct t-representative functions holds that $|\mathcal{F}_t| \leq 2^{2^t - 1}$.*

Proof. Consider the set $\mathcal{M}_t$ of monotone functions that start from $f(\emptyset) = t$ and every function also has property p:

$$\text{For all sets } S \subseteq B \text{ holds that } \max_{S^* \subset S} \{f_G(S^*)\} + 1 \geq f_G(S)$$

This means that at every $S \subseteq B$, except $S = \emptyset$ the output can increase by one, or stay equal to a smaller input. There are $2^t - 1$ distinct sets $\emptyset \subset S \subseteq B$, where the output increases or stays equal. This results in $2^{2^t - 1}$ distinct functions.

Since p is a weaker version of point 3 in Definition 5, we know that for every $f \in \mathcal{F}_t$ also holds that $f \in \mathcal{M}_t$. Since every $f \in \mathcal{F}_t$ maps to itself in $\mathcal{M}_t$, and $\mathcal{F}_t$ consists of distinct functions, we can conclude that there are never more than 2^{2^t-1} distinct functions $f \in \mathcal{F}_t$. Hence we know that $|\mathcal{F}_t| \leq 2^{2^t-1}$. □

Chapter 5

Equivalence classes for graphs of bounded treewidth

We defined a set of general t-boundaried graphs, of which no pair was equivalent, to find a lower bound on the number of equivalence classes. This lower bound is not yet applicable for meta kernelization, since meta kernelization is applied to graphs that can be embedded in a surface of bounded genus, and the boundaried graphs we want to replace are protrusions, which means they have bounded treewidth. In this section we will work towards a lower bound for t-boundaried graphs of bounded treewidth.

The provable lower bound for the number of equivalence classes for general graphs was $M(t) - 1$, as proved in Lemma 10. We will work towards this lower bound of $M(t) - 1$ for graphs of bounded treewidth.

The graphs of bounded treewidth we use in this section are strongly based on a reduction from BOOLEAN SATISFIABILITY (SAT) in *Conjunctive Normal Form* (CNF) to INDEPENDENT SET developed by Lokshtanov et al. [20].

We construct a t-boundaried graph G_ϕ from a CNF formula ϕ. First we will explain how parts of graph G_ϕ are constructed, and what useful properties they have. Then we look into the construction and properties of G_ϕ itself.

Lemma 12. *For any positive integer n there is a graph G_n containing terminal vertices $v_1, \ldots, v_n$ with the following properties:*

1. *any independent set in G_n has size at most $n + 2$,*
2. *any independent set of size $n + 2$ in G_n contains at least one of the vertices $\{v_1, \ldots, v_n\}$,*
3. *for any $i \in n$, there is an independent set X of size $n + 2$ in G_n such that $X \cap \{v_1, \ldots, v_n\} = \{v_i\}$.*

Proof. Given such a positive integer n, we can construct G_n by starting from n disjoint triangles $\{u_1, v_1, w_1\}, \ldots, \{u_n, v_n, w_n\}$. We connect the triangles with edges $\{w_i, u_{i+1}\}$ for every $1 \leq i < n$, and add vertices $v_{\text{start}}, v_{\text{end}}$ with edges $\{v_{\text{start}}, u_1\}, \{w_n, v_{\text{end}}\}$. Figure 5.1a gives an overview of a constructed graph G_n (with $n = 3$).

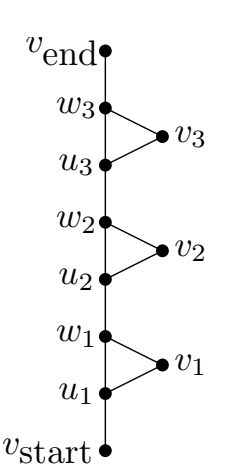
We can prove all of the above properties for such a graph G_n:

1. By construction G_n has n disjoint cliques, namely $\{u_i, v_i, w_i\}$ for every $1 \leq i < n$. From each clique, only one vertex can be in an independent set. Besides these cliques, G_n has two other vertices $v_{\text{start}}, v_{\text{end}}$, which are not connected to each other and can therefore both be in an independent set. From the previous observations follows that any independent set X in G_n has size at most $n + 2$.

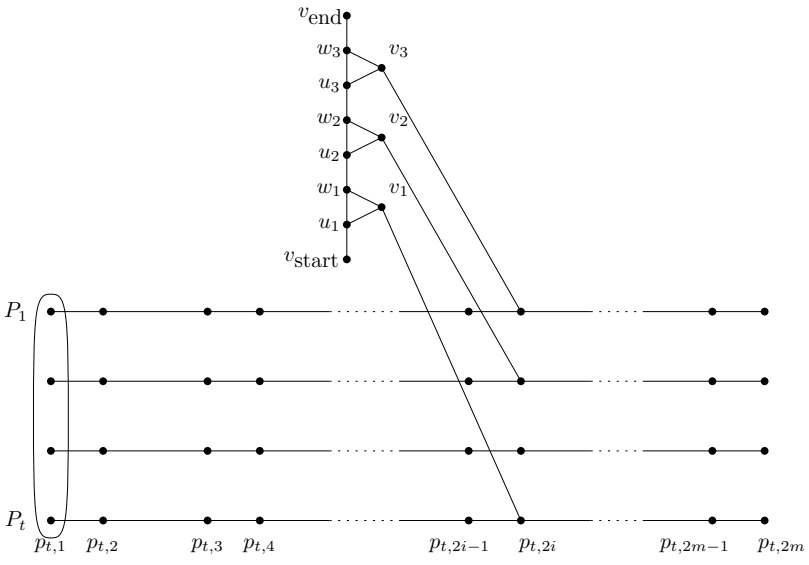
2. Assume we have an independent set X of size $n + 2$ for G_n, where $v_i \notin X$ for $1 \leq i \leq i$. Observe that $v_{\text{start}}, u_1, w_1, \ldots, u_n, w_n, v_{\text{end}}$ forms a path of $2n + 2$ vertices, and X only contains vertices from this path. Any independent set on a path never contains two subsequent vertices in the path, hence $|X| \leq n + 1$, which leads to a contradiction. We can conclude that an independent set of size $n + 2$ contains at least one of the vertices $v_1, \ldots, v_n$.

3. For $1 \leq i \leq n$, we can construct an independent set X of size $n + 2$ in G_n such that $X \cap \{v_1, \ldots, v_n\} = \{v_i\}$ as follows. Start from the independent set consisting of only $\{v_i\}$, and observe that both u_i and w_i cannot occur in the same independent set. We can look at vertices $v_{\text{start}}, u_1, w_1, \ldots, u_{i-1}, w_{i-1}$ and $u_{i+1}, w_{i+1}, \ldots, u_n, w_n, v_{\text{end}}$ as two disjoint paths. We can find maximum independent sets $X_l = \{v_{\text{start}}, w_1, \ldots, w_{i-1}\}$ and $X_r = \{u_{i+1}, \ldots, u_n, v_{\text{end}}\}$ of size i and $n - i + 1$ respectively for these two paths. Now $X = X_l \cup \{v_i\} \cup X_r$ is an independent set for G_n and $|X| = i + 1 + n - i + 1 = n + 2$. □

The graph G_n in Lemma 12 can be used as a *clause gadget*, by connecting its terminal vertices to parts of a bigger graph. The next lemma will show how we do this, in order to build t-boundaried graph G_ϕ for *monotone* CNF formula ϕ. We call a CNF formula ϕ with t variables $x_1, \ldots, x_t$ monotone when it only contains positive literals. Changing a variable x_i from 0 to 1 can only change $\phi(x_1, \ldots, x_n)$ from 0 to 1, not the other way around.



(a) Graph G_3 with terminal vertices v_1, v_2, v_3.



(b) Graph G_ϕ with only clause gadget G_{C_i} drawn.

Figure 5.1: Graph G_3 and construction of t-boundaried graph G_ϕ from a CNF formula ϕ, which shows G_3 used as clause gadget for clause $C_i = (\ell_1, \ell_2, \ell_3)$, where the literals correspond to variables x_1, x_2, x_t respectively.

Lemma 13. *For any monotone CNF formula ϕ with t variables $x_1, \ldots, x_t$ and m clauses $C_1, \ldots, C_m$, there is a t-boundaried graph G_ϕ with boundary $B = \{p_{1,1}, \ldots, p_{t,1}\}$ with the following properties:*

1. *any independent set in G_ϕ has size at most $mt + (\sum_{1 \leq i \leq m} |C_i| + 2)$,*
2. *$f_{G_\phi}(B) = mt + (\sum_{1 \leq i \leq m} |C_i| + 2)$*
3. *Assignment of values $x_1, \ldots, x_t \in \{0, 1\}$ ensures that $\phi(x_1, \ldots, x_t) = 1$ if and only if $f_{G_\phi}(\{p_{i,1} \mid x_i = 1\}) = f_{G_\phi}(B)$.*
4. *$\mathbf{pw}(G_\phi) \leq t + 2$*

Proof. Given a monotone CNF formula ϕ with t variables $x_1, \ldots, x_t$ and m clauses $C_1, \ldots, C_m$, we are going to construct t-boundaried graph G_ϕ as follows. We start by creating t paths $P_1, \ldots, P_t$, where every path P_a consists of $2m$ vertices $p_{a,1}, \ldots, p_{a,2m}$. Note that we already have all the vertices of the boundary now: $B = \{p_{1,1}, \ldots, p_{t,1}\}$.

We denote a clause C_i with positive literals as $(\ell_1, \ldots, \ell_j)$, where each ℓ_a contains the index of the corresponding variable x_{ℓ_a}. Clause gadget G_{C_i} is created by using Lemma 12 with $n = |C_i|$. In clause gadget G_{C_i} for every literal ℓ_a of C_i there is a terminal vertex v_a. We connect G_{C_i} to the paths by adding an edge $\{v_a, p_{\ell_a,2i}\}$ for every terminal vertex v_a. Doing this for all m clauses results in graph G_ϕ. Figure 5.1b shows an example of such a constructed graph.

Note that it is important that during this construction that C_i is sorted in such a way that $l_{|C_i|} < \ldots < l_1$, so that the edges that connect terminal vertices to the paths do not introduce edge crossings among each other. We will use this property in Section 6.

For G_ϕ we can prove the properties mentioned above:

1. Observe that G_ϕ consists of t paths of $2m$ vertices and m clause gadgets. As we have mentioned before, any independent set on a path never contains two subsequent vertices in the path, hence an independent set can contain at most m vertices in any of the t paths. By Lemma 12 we know that clause gadget G_{C_i} has an independent set of at most $|C_i| + 2$. In total this means we have an independent set of at most $mt + (\sum_{1 \leq i \leq m} |C_i| + 2)$.

2. There is an independent set X of size $mt + (\sum_{1 \leq i \leq m} |C_i| + 2)$, where $|X \cap G_{C_j}| = |C_j| + 2$ for $1 \leq j \leq m$ and $|X \cap P_i| = m$ for $1 \leq i \leq t$: Assume X contains vertices $p_{i,2k-1}$ for $1 \leq k \leq m$ in every path P_i. Since all terminal vertices of clause gadget G_{C_j} are either connected to a path P_i at vertex $p_{i,2j}$ or not connected to path P_i, we know that for every terminal vertex v of G_{C_j}, $X \cup \{v\}$ is an independent set. Using property 3 of Lemma 12, we know that there exists an independent set X, which has $|X \cap G_{C_j}| = |C_j| + 2$ for $1 \leq j \leq m$ when $|X \cap P_i| = m$ for $1 \leq i \leq t$. Since this X contains vertices $p_{i,2k-1}$ for $1 \leq k \leq m$ in every path P_i, hence we can conclude that $f_{G_\phi}(B) = mt + (\sum_{1 \leq i \leq m} |C_i| + 2)$.

3. We prove both sides of the bi-implication separately:

   ($\Rightarrow$) Assume we have an assignment of variables $x_1, \ldots, x_t \in \{0, 1\}$ that ensures $\phi(x_1, \ldots, x_t) = 1$. This means that every clause C_j has at least one positive literal ℓ_a for which variable x_{ℓ_a} becomes 1 by this assignment of variables. By property 3 of Lemma 12 we know that for every clause gadget G_{C_j}, we can find an independent set X_{C_j} of size $|C_j| + 2$, which contains exactly one terminal vertices. We are interested in the case where $v_a \in X_{C_j}$, corresponding to ℓ_a with $x_{\ell_a} = 1$.

Assume we have an independent set X_{P_i} of size m for every path P_i in G_ϕ, which contains for $1 \leq k \leq m$

- all vertices $p_{i,2k-1}$ if $x_i = 1$
- all vertices $p_{i,2k}$ if $x_i = 0$

We know for every terminal vertices of clause gadget G_{C_j} that it is either connected to $p_{i,2j}$ or not connected to a path P_i at all. Thus if $v_a \in X_{C_j}$ because $x_{\ell_a} = 1$ then it is connected to path P_{ℓ_a}. This means that terminal vertex v_a is adjacent to $p_{\ell_a,2j} \notin X_{P_i}$, since $p_{\ell_a,2j-1} \in X_{P_i}$ because $x_{\ell_a} = 1$. Hence $X := \bigcup_{1\leq j\leq m} X_{C_j} \cup \bigcup_{1\leq i\leq t} X_{P_i}$ is an independent set of size $mt + (\sum_{1\leq i\leq m} |C_i| + 2)$. Furthermore X_{P_i} only contains all vertices $p_{i,2j-1}$ if $x_i = 1$, so in particular $p_{i,1} \in X_{P_i}$. Hence we can conclude that $X \cap B = \{p_{i,1} \mid x_i = 1\}$.

We now know that $mt + (\sum_{1\leq i\leq m} |C_i| + 2) = |X| \leq f_{G_\phi}(\{p_{i,1} \mid x_i = 1\})$. By Lemma 6 we know that $f_{G_\phi}(\{p_{i,1} \mid x_i = 1\}) \leq f_{G_\phi}(B)$. However, property 2 of Lemma 13 tells us that $f_{G_\phi}(B) = mt + (\sum_{1\leq i\leq m} |C_i| + 2)$. Hence $f_{G_\phi}(\{p_{i,1} \mid x_i = 1\}) = f_{G_\phi}(B)$.

($\Leftarrow$) Assume that for a particular set $S \subseteq B$ holds that $f_{G_\phi}(S) = f_{G_\phi}(B)$. By property 2 of Lemma 13 we can conclude that $f_{G_\phi}(S) = mt + (\sum_{1\leq i\leq m} |C_i| + 2)$. Using Definition 4, we know that there is an independent set X of size $mt + (\sum_{1\leq i\leq m} |C_i| + 2)$ for G_ϕ such that $X \cap B \subseteq S$.

The only way X can have size $mt + (\sum_{1\leq i\leq m} |C_i| + 2)$ is when $|X \cap G_{C_j}| = |C_j| + 2$ for $1 \leq j \leq m$, as every path P_i of $2m$ vertices has $|X \cap P_i| \leq m$ (and for X must hold that $|X \cap P_i| = m$). By property 2 of Lemma 12 we know that X contains at least one terminal vertex v of every clause gadget G_{C_i}. Assume $v \in X$ is a terminal vertex of G_{C_j} and has an edge to $p_{i,2j}$. As $v \in X$, it must be that $p_{i,2j} \notin X$. Since $|X \cap P_i| = m$, we know that either $p_{i,2k-1} \in X$ or $p_{i,2k} \in X$ for $1 \leq k \leq m$. We know that $p_{i,2j} \notin X$, therefore $p_{i,2j-1} \in X$, but then $p_{i,2(j-1)} \notin X$. This pattern repeats itself for lower indices, so eventually we get $p_{i,1} \in X$.

In X there is at least one terminal vertex $v \in X$, which is connected to $p_{i,2j}$, for every G_{C_j}, and we have just seen that as a consequence $p_{i,1} \in X$. Since $|X \cap B| \subseteq S$ we can assign $x_i = 1$ if $p_{i,1} \in S$, and $x_i = 0$ if $p_{i,1} \notin S$ and have at least one variable $x_{\ell_a} = 1$ in literal ℓ_a for every $C_j \in \phi$. As ϕ is a monotone CNF, we can conclude that $\phi(x_1, \ldots, x_t) = 1$.

4. We can prove a bound on the pathwidth of graph G_ϕ by using a mixed search game, as explained in Chapter 2. We will first describe the mixed search strategy S, which we use to clean G_ϕ. When the strategy is clear, we can argue why all edges are cleaned at the end of the game and how many cleaners are sufficient to clean G_ϕ.

   Observe that two cleaners are sufficient to clean an isolated clause gadget, by moving the first cleaner from v_{start} to v_{end} and putting the second cleaner on vertex v_i when u_i, w_i are being cleaned by the first cleaner. A single path P_i in G_ϕ can be cleaned by a single cleaner. The strategy has to make sure that the edges from terminal vertices to the paths are also cleaned. We do this in phases, where each phase deals with a single clause gadget and part of the paths. We start phase i with the cleaner for path P_j at vertex $p_{j,2i-1}$, for $1 \leq j \leq t$. Every path cleaner now moves to $p_{j,2i}$. Now we clean clause gadget G_{C_i} as we just described. When the clause gadget has been cleaned and the cleaners for the clause gadget have been removed, we move every path cleaner to vertex $p_{j,2(i+1)-1}$, which is the starting position for phase

$i+1$. Figure 5.2 shows the mixed search strategy for phase i. Mixed search strategy S starts at phase 1 and ends after phase m.

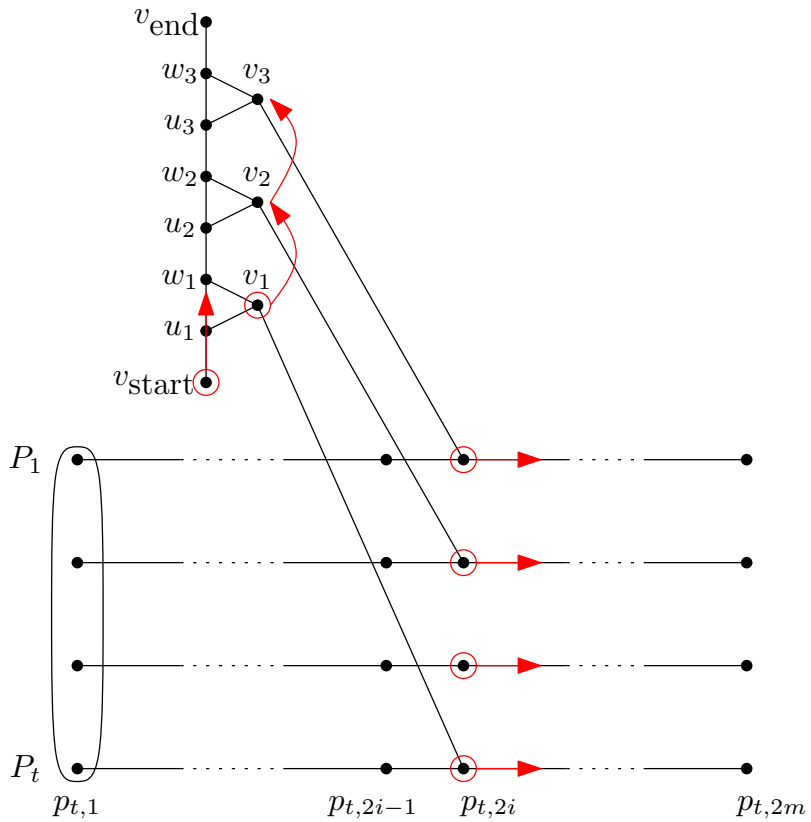


Figure 5.2: Mixed search strategy S for graph G_ϕ, where the paths are being cleaned by t cleaners, and clause gadget G_{C_i} is cleaned by 2 cleaners.

Claim 1. *Mixed search strategy S is a winning strategy for G_ϕ.*

Proof. Observe that in phase i, isolated clause gadget G_{C_i} is cleaned by S. Throughout the m phases of S, all t paths are cleaned. In phase i, the path cleaner for path P_j is located at vertex $p_{j,2i}$, when clause gadget G_{C_i} is cleaned. Since a terminal vertex of G_{C_i} is either connected to path P_j at vertex $p_{j,2i}$, or not at all, we know that all edges from terminal vertices of G_{C_i} to the paths have been cleaned. Hence no recontamination of edges that are cleaned in phases $1,\ldots,i$ can take place in phase i. After m phases, G_ϕ is completely cleaned and no recontamination has taken place. □

Claim 2. *To execute mixed search strategy S it suffices to have $t+2$ cleaners.*

Proof. To clean the t paths in G_ϕ we use t cleaners, throughout all m phases. In every phase i, we use two extra cleaners to clean clause gadget G_{C_i} and the edges connecting terminal vertices to the paths. Hence it suffices to have $t+2$ cleaners to execute S. □

By Claims 1 and 2 we can conclude that $\mathbf{ms}(G_\phi) \leq t+2$ and therefore $\mathbf{pw}(G_\phi) \leq t+2$ □

Lemma 13 proved that for a monotone CNF formula ϕ, there exists a t-boundaried graph G_ϕ whose pathwidth is bounded by $t+2$. Important is property 3: an assignment of values $x_1,\ldots,x_t \in \{0,1\}$ that ensures $\phi(x_1,\ldots,x_t) = 1$, G_ϕ corresponds to a maximum independent set X of size $mt + (\sum_{1\leq i\leq m} |C_i| + 2)$, with $X \cap B = \{p_{i,1} \mid x_i = 1\}$. We will later use this fact to create graphs that are not equivalent to each other. Furthermore, we know that $\mathbf{tw}(G_\phi) \leq \mathbf{pw}(G_\phi) \leq t+2$.

The next Lemma we are going to prove will provide a way to connect our lower bound using monotone Boolean functions (Chapter 4), to the construction of t-boundaried graphs with treewidth bounded by $t+2$.

Lemma 14. *Every monotone Boolean function f from $\{x_1, x_2, \ldots, x_n\}$ to $\{0,1\}$ with $f(1,\ldots,1) = 1$ can be represented by a monotone CNF formula ϕ on the same variables, for which holds that $f(x_1, x_2, \ldots, x_n) = \phi(x_1, x_2, \ldots, x_n)$ for all $x_1,\ldots,x_n \in \{0,1\}$.*

Proof. Let f be a Boolean function on n variables $x_1,\ldots,x_n$ for which we are going to construct a CNF formula. We start by looking at the truth table of f and find the set A of assignments $x_1,\ldots,x_n \in \{0,1\}$ such that $f(x_1,\ldots,x_n) = 0$ and there is a maximal set of variables $x_i = 1$. Note that for such an assignment $a \in A$, setting a single variable x_i from 0 to 1 will ensure that $f(x_1,\ldots,x_n) = 1$.

For every $a \in A$ we get a formula by creating a conjunction of $\neg x_i$ for $x_i = 0$ in a. This formula now express that for $a \in A$, and any assignment with a superset of variables $x_i = 0$ with respect to a, holds that $f(x_1,\ldots,x_n) = 0$. If we negate the whole conjunction, we can apply *De Morgan* (and remove double negations) to get a clause C_a, which is a disjunction of positive literals. Clause C_a expresses that for any assignment with a superset of variables $x_i = 1$ with respect to a holds that $f(x_1,\ldots,x_n) = 1$. Finally we create a monotone CNF formula ϕ by taking the conjunction of all created clauses C_a.

First consider the trivial assignment where every variable is 1. Since there are only positive literals in ϕ, we know that $\phi(1,\ldots,1) = 1$, and by assumption we have that $f(1,\ldots,1) = 1$. We will now show for non-trivial assignments that $f(x_1,\ldots,x_n) = 1 \Leftrightarrow \phi(x_1,\ldots,x_n) = 1$:

($\Rightarrow$) Assume that we have a non-trivial assignment a^* such that $f(x_1,\ldots,x_n) = 1$. This means that $a^* \notin A$, since for every $a \in A$ holds that $f(x_1,\ldots,x_n) = 0$. Therefore $a*$ has a superset of variables $x_i = 1$ with respect to every assignment $a \in A$. By construction of the clauses of ϕ, we can now conclude that for every clause C_a there is at least one variable that is set to 1, since C_a expresses that for any assignment with a superset of variables $x_i = 1$ with respect to a holds that $f(x_1,\ldots,x_n) = 1$. When every clause of ϕ has at least one variable that is 1, then $\phi(x_1,\ldots,x_n) = 1$.

($\Leftarrow$) Assume we have a non-trivial assignment a^* such that $\phi(x_1,\ldots,x_n) = 1$. Every clause has at least one variable that is set to 1. This means that a^* has a superset of variables $x_i = 1$ with respect to every $a \in A$. Since we selected every $a \in A$ as an assignment with a maximal set of variables $x_i = 0$ such that $f(x_1,\ldots,x_n) = 0$, we can conclude that for a^* holds that $f(x_1,\ldots,x_n) = 1$.

We have shown that given a monotone Boolean function f on t variables, we can construct a monotone CNF formula ϕ, for which holds that $f(x_1,\ldots,x_n) = \phi(x_1,\ldots,x_n)$. □

Lemma 14 tells us that for a given monotone Boolean function f, we can find a monotone CNF formula ϕ, which is satisfied if and only if $f(x_1, x_2, \ldots, x_n) = 1$. First we will prove how we

construct t-boundaried graph G_f with treewidth bounded by $t+2$ for monotone Boolean function f. When this relation between monotone functions and t-boundaried graphs is established, we can use it to prove a lower bound on the number of equivalence classes for t-boundaried graphs of bounded treewidth for independent set.

Lemma 15. *For any monotone Boolean function f from $\{x_1,\ldots,x_t\}$ to $\{0,1\}$ with $f(1,\ldots,1)=1$, there exists a t-boundaried graph G_f with boundary $B=\{p_{1,1},\ldots,p_{n,1}\}$ with the following properties:*

1. *any independent set in G_f has size at most $mt+(\sum_{1\leq i\leq m}|C_i|+2)$,*
2. *$f_{G_f}(B)=mt+(\sum_{1\leq i\leq m}|C_i|+2)$*
3. *Assignment of values $x_1,\ldots,x_t\in\{0,1\}$ ensures that $f(x_1,\ldots,x_t)=1$ if and only if $f_{G_f}(\{p_{i,1}\mid x_i=1\})=f_{G_f}(B)$.*
4. *$\mathbf{pw}(G_f)\leq t+2$*

Proof. Given such a monotone Boolean function f we use Lemma 14 to represent f as a monotone CNF formula ϕ, where for all $x_1,\ldots,x_t\in\{0,1\}$ holds that $f(x_1,\ldots,x_t)=\phi(x_1,\ldots,x_t)$. We can now construct graph G_f for monotone Boolean function f, by constructing G_ϕ using Lemma 13 with the monotone CNF formula ϕ that represents f. Since the properties in Lemma 15 are the same properties as in Lemma 13, G_f has all the desired properties. □

Lemma 15 shows us that for a specific monotone Boolean function f, constructed t-boundaried graph G_f has the property that an assignment of values $x_1,\ldots,x_t\in\{0,1\}$ ensures that $\phi(x_1,\ldots,x_t)=1$ if and only if $f_{G_f}(\{p_{i,1}\mid x_i=1\})=f_{G_f}(B)$. We will use this properties to prove that graphs G_1,G_2, which are constructed from distinct monotone Boolean functions f_1,f_2, are not equivalent. We call monotone Boolean functions distinct if there is some assignment of values to the variables such that $f_1(x_1,x_2,\ldots,x_t)\neq f_2(x_1,x_2,\ldots,x_t)$.

Lemma 16. *There are $M(t)-1$ distinct monotone Boolean functions f with $f(1,\ldots,1)=1$ for which there are non-equivalent t-boundaried graphs G_f with boundary B, with the property that an assignment of values $x_1,\ldots,x_t\in\{0,1\}$ ensures that $f(x_1,\ldots,x_t)=1$ if and only if $f_{G_f}(\{p_{i,1}\mid x_i=1\})=f_{G_f}(B)$.*

Proof. Assume we have two distinct monotone Boolean functions f_1,f_2, for which holds that $f_1(1,\ldots,1)=1$ and $f_2(1,\ldots,1)=1$. We construct t-boundaried graphs G_1 and G_2, for f_1 and f_2 respectively, by using Lemma 15. We call the boundary of both graphs B.

Assume that $G_1\equiv_{\text{IS}}G_2$, and observe that for every $S\subseteq B$ holds that there is a constant c such that $f_{G_1}(S)=f_{G_2}(S)+c$. We constructed G_1 using Lemma 15, so it has the property that $f_1(x_1,\ldots,x_t)=1$ for an assignment of values $x_1,\ldots,x_t\in\{0,1\}$ if and only if $f_{G_1}(\{p_{i,1}\mid x_i=1\})=f_{G_1}(B)$. The same holds for G_2.

Since $f_{G_1}(S)=f_{G_2}(S)+c$ for every $S\subseteq B$, we can derive that for every assignment of values where $f_{G_2}(\{p_{i,1}\mid x_i=1\})=f_{G_2}(B)$, also $f_{G_1}(\{p_{i,1}\mid x_i=1\})=f_{G_1}(B)$. This means that whenever $f_1(x_1,\ldots,x_t)=1$, also $f_2(x_1,\ldots,x_t)=1$, which contradicts the fact that they are distinct functions. Hence the assumption that $G_1\equiv_{\text{IS}}G_2$ is incorrect.

Since there are $M(t)-1$ distinct monotone Boolean functions f that have $f(1,\ldots,1)=1$, we can construct $M(t)-1$ graphs of which none are equivalent. □

Lemma 16 allows us to prove a lower bound on the number of equivalence classes for INDEPENDENT SET on t-boundaried graphs of treewidth at most $t+2$.

Lemma 17. *There are at least $M(t) - 1$ equivalence classes for* INDEPENDENT SET *on t-boundaried graphs of treewidth at most $t + 2$.*

Proof. We use Lemma 16 to find $M(t) - 1$ distinct Boolean functions for which Lemma 15 can construct a t-boundaried graph of treewidth $t + 2$. Since none of the $M(t) - 1$ graphs are equivalent according to Lemma 16, we have proved that there are at least $M(t) - 1$ equivalence classes for INDEPENDENT SET on t-boundaried graphs of treewidth at most $t + 2$. □

Chapter 6

Equivalence classes for planar graphs of bounded treewidth

In the previous chapter we established a lower bound for t-boundaried graphs of bounded treewidth, and in this chapter we are going to extend it to t-boundaried graphs of bounded treewidth that can be embedded in a surface of bounded genus. In particular, we work towards planar graphs, since planar graphs are embeddable in surfaces of genus zero.

We start from the graphs we have constructed in the previous section. We know there are $M(t) - 1$ graphs of treewidth at most $t + 2$ of which none are equivalent. The only thing we have to do is make them planar and ensure that property 3 of Lemma 15 still holds for the new graphs. To transform the graphs of the previous section into planar graphs we will use a *Crossover gadget* for VERTEX COVER (VC) from Garey et al. [15].

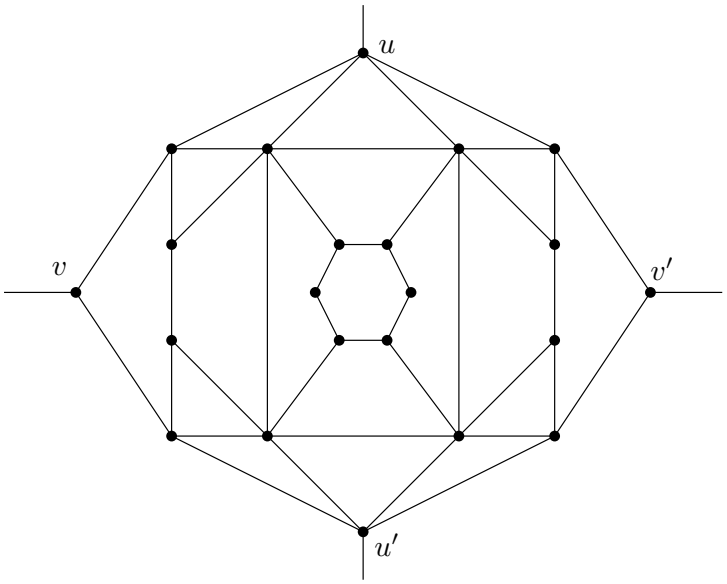
The optimization for vertex cover we are interested in is minimum vertex cover, since it is inverse of maximum independent set: If we have a graph $G = (V, E)$ for which we have found minimum vertex cover $C \subseteq V$, then the set $V \setminus C$ is a maximum independent set.

Crossover gadget $G_\times$ is illustrated in Figure 6.1a. When two edges $\{a, b\}, \{c, d\}$ cross, we can create a graph where they no longer cross by removing these edges, and connecting the endpoint to a copy of $G_\times$ as follows: $\{a, v\}, \{v', b\}, \{c, u\}, \{u', d\}$. Garey et al. [15] show that the size of minimum vertex cover C for $G_\times$ is dependent on whether C contains the vertices v, v', u and u'. Since maximum independent set is the inverse problem, we can also find such a relation for maximum independent set. In Table 6.1b we show the size of the maximum independent set X for $G_\times$, having defined i and j as

$$|\{v, v'\} \cap X| = i \qquad\qquad |\{u, u'\} \cap X| = j.$$

Lemma 18. *For any monotone Boolean function f from $\{x_1, \ldots, x_t\}$ to $\{0, 1\}$ with $f(1, \ldots, 1) = 1$, there exists a planar t-boundaried graph G_f^p with boundary $B = \{p_{1,1}, \ldots, p_{n,1}\}$ and $N_\times$ crossover gadgets, with the following properties:*

1. *any independent set in G_f^p has size at most $mt + 9N_\times + (\sum_{1 \le i \le m} |C_i| + 2)$*
2. *$f_{G_f^p}(B) = mt + 9N_\times + (\sum_{1 \le i \le m} |C_i| + 2)$*
3. *$\mathbf{pw}(G_f^p) \le t + 6$*



(a) Crossover gadget $G_\times$

j \ i	0	1	2
0	7	8	8
1	8	9	9
2	7	8	9

(b) Size of maximum IS

Figure 6.1: Crossover gadget for vertex cover from Garey et al. [15]

4. *Assignment of values* $x_1, \ldots, x_t \in \{0,1\}$ *ensures that* $f(x_1, \ldots, x_t) = 1$ *if and only if* $f_{G_f^p}(\{p_{i,1} \mid x_i = 1\}) = f_{G_f^p}(B)$.

We will show how to construct such a planar t-boundaried graph G_f^p for a monotone Boolean function f. When the construction is clear, we will prove the properties in Lemma 18 separately.

Assume we have an arbitrary monotone Boolean function f from $\{0,1\}^t$ to $\{0,1\}$ with $f(1,\ldots,1) = 1$. We can use Lemma 15 to construct t-boundaried graph G_f for monotone Boolean function f. We can use $G_\times$ to planarize t-boundaried graph G_f: We have to resolve edge crossings that occur when connecting clause gadgets to the t paths. We present a way to resolve all crossings, for which we can later prove that the graph still has a (low) bounded treewidth and the optimum solutions have nice properties.

Assume edge set E_{C_j} connects clause gadget G_{C_j} to the vertices $p_{i,2j}$ in paths P_i of G_f. We can limit the number of edge crossings by drawing G_f in such a way that every edge $e \in E_{C_i}$ can only cross the edges between the vertices $p_{i,2j-1}$ and $p_{i,2j}$ of path P_i and does not cross any other edge $e' \in E_{C_i}$. We already hinted at this property in Chapter 5, but to ensure that edges $e, e' \in E_{C_i}$ do not cross, clause C_i is ordered in such a way that variables x_i are sorted in the same way as paths P_i in G_f.

We construct G_f^p by replacing every edge crossing in G_f caused by the edges in E_{C_i} by a crossover gadget, which is oriented in such a way that v and v' are connect along the paths of G_f^p, while u and u' are connected along the edges E_{C_i}. Figure 6.2 shows the result of this planarization process, resulting in planar t-boundaried graph G_f^p. The crossover gadgets are drawn as grey diamonds which only show vertices v, v', u, u'. In the enlarged view the orientation of the crossover gadget becomes visible.

First we will prove some properties about the size of G_f^p.

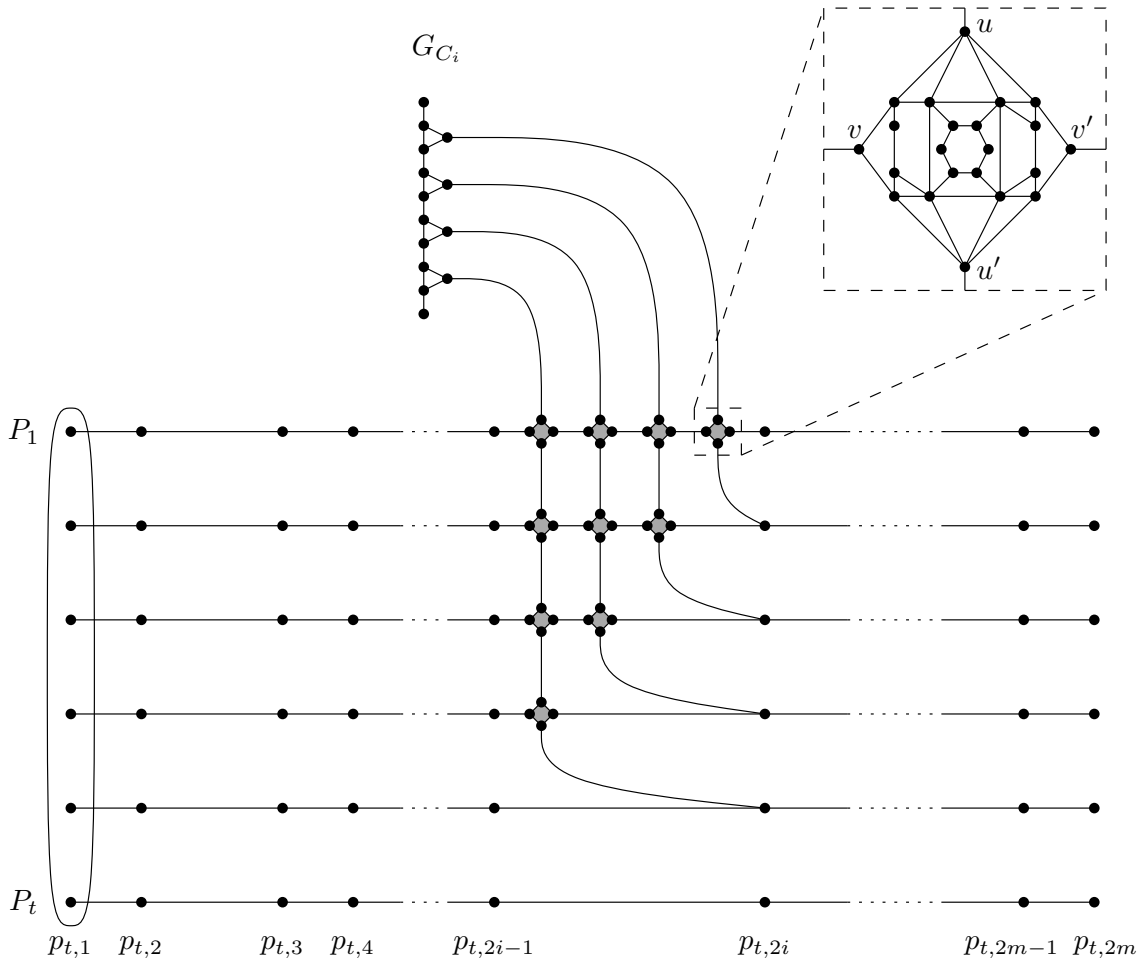


Figure 6.2: Planar t-boundaried graph G_f^p, constructed for a monotone Boolean function f. Only clause gadget G_{C_i} is drawn to show how the edge crossing caused by a single clause gadget been resolved.

Claim 3. *Any independent set in G_f^p has size at most $mt + 9N_\times + (\sum_{1\leq i\leq m} |C_i| + 2)$, where $N_\times$ is the number of crossover gadgets in G_f^p*

Proof. By property 1 of Lemma 12 every clause gadget G_{C_i} has a maximum independent set of at most $|C_i| + 2$.

As we have seen in Table 6.1b, an independent set for a crossover gadget has size at most nine, where at least one of the vertices $\{v, v'\}$ and one of the vertices $\{u, u'\}$ must be in the independent set. Using this fact we know that only one of the vertices $p_{i,2j-1}$ and $p_{i,2j}$ on path P_i can be in an independent set for some $1 \leq j \leq m$: Since we need at least one of the vertices $\{v, v'\}$ in each crossover gadget to get an independent set of size nine, they will either ensure that both $p_{i,2j-1}$ and $p_{i,2j}$ cannot be in an independent set if there is a crossover gadget in between them where both v and v' are in the independent. Conversely, one of $p_{i,2j-1}$ and $p_{i,2j}$ is in a maximal independent set if every crossover gadget in between them contain either v or v'. Since there are t paths where we

have m pairs of such vertices $p_{i,2j-1}$ and $p_{i,2j}$, we know that any maximum independent set can contain at most mt of these path vertices.

Any independent set can therefore contain at most $mt + 9N_\times + (\sum_{1\leq i\leq m} |C_i| + 2)$ vertices. □

Claim 4. *For G_f^p holds that $f_{G_f^p}(B) = mt + 9N_\times + (\sum_{1\leq i\leq m} |C_i| + 2)$.*

Proof. By property 3 of Lemma 12, we can find an independent set X_{C_i} of size $|C_i| + 2$ for every clause gadget G_{C_i}. For every path P_j, we find an independent set $X_j = \{p_{j,2k-1} \mid 1 \leq k \leq m\}$. For every crossover gadget we can find a maximum independent set $X_\times$ for which holds that $u', v' \in X_\times$ and $u, v \notin X_\times$.

Observe that, because $u, v \notin X_\times$, $X = \bigcup_{1\leq i\leq m} X_{C_i} \cup \bigcup_{1\leq j\leq t} X_{P_j} \cup \bigcup X_\times$ is an independent set for G_f^p of size $mt + 9N_\times + (\sum_{1\leq i\leq m} |C_i| + 2)$, hence $mt + 9N_\times + (\sum_{1\leq i\leq m} |C_i| + 2) \leq f_{G_f^p}(B)$. By Claim 3 we know that $f_{G_f^p}(B) \leq mt + 9N_\times + (\sum_{1\leq i\leq m} |C_i| + 2)$, thus $f_{G_f^p}(B) = mt + 9N_\times + (\sum_{1\leq i\leq m} |C_i| + 2)$. □

Since planar t-boundaried graph G_f^p is significantly more complex than G_f, we have to check whether its treewidth is still bounded. Again we can use a mixed search game to bound the pathwidth of G_f^p and using $\mathbf{tw}(G_f^p) \leq \mathbf{pw}(G_f^p)$ we find a bound on the treewidth. First we establish a mixed game strategy to clean crossover gadgets.

Lemma 19. *Crossover gadget $G_\times$ with vertices v, v', u, u' and edge $\{u, w\}$, and a cleaner on vertex v and on vertex w, can be cleaned by six cleaners. When $G_\times$ has been cleaned, there is a cleaner on v' and on u'. Removing the two initial cleaners will not contaminate $G_\times$.*

Proof. We will define a mixed search strategy S under the conditions in the lemma. For this strategy S, we will show that the strategy cleans all edges in $G_\times$ and that six cleaners suffice to carry out the strategy.

Figure 6.3 shows strategy S for crossover gadget $G_\times$. The red *tracks* that are drawn show how the cleaners slide through the gadget to clean it, but the order in which this happens will be elaborated on: We start with a cleaner on vertex v, we introduce cleaners c_1, c_2, c_3 and c_4, after which we can safely remove the cleaner at v. First cleaners c_2, c_3 move to their second positions. Then cleaners c_1, c_4 can move to their second positions. At this point, we can remove the cleaner from vertex w, since the edge $\{u, w\}$ had cleaners on both endpoints. We introduce cleaners c_5 and c_6, which can move to their third and second positions, respectively. Now cleaners c_2, c_3 move to their third positions. Cleaner c_5 can move to its forth and final position, and thereafter we can remove cleaners c_5 and c_6. Cleaner c_1 move to its final position, while a new cleaner is put on the final position for c_4, to prevent $G_\times$ from being recontaminated by the edge that connects u' to some vertex outside $G_\times$. Finally cleaners c_2, c_3 also move to their final position. To finish the cleaning of $G_\times$ we put a new cleaner on vertex v' and remove cleaners c_1, c_2, c_3, c_4.

Claim 5. *Mixed search strategy S cleans all edges of $G_\times$ without allowing any recontamination and leaves cleaners on u' and v'*

Proof. Observe that every edge e in $G_\times$ at some point during S either has two cleaners at each of its endpoints, or a cleaner slides over e. Only after cleaning $\{u, w\}$, we remove the cleaner on w to prevent recontamination on other edges incident to w. To prevent recontamination within $G_\times$ all cleaners move in such a way that there is never a path in $G_\times$ from v' to v that does not go through a vertex with a cleaner on it. Furthermore a cleaner is left behind at u', since the edge incident to

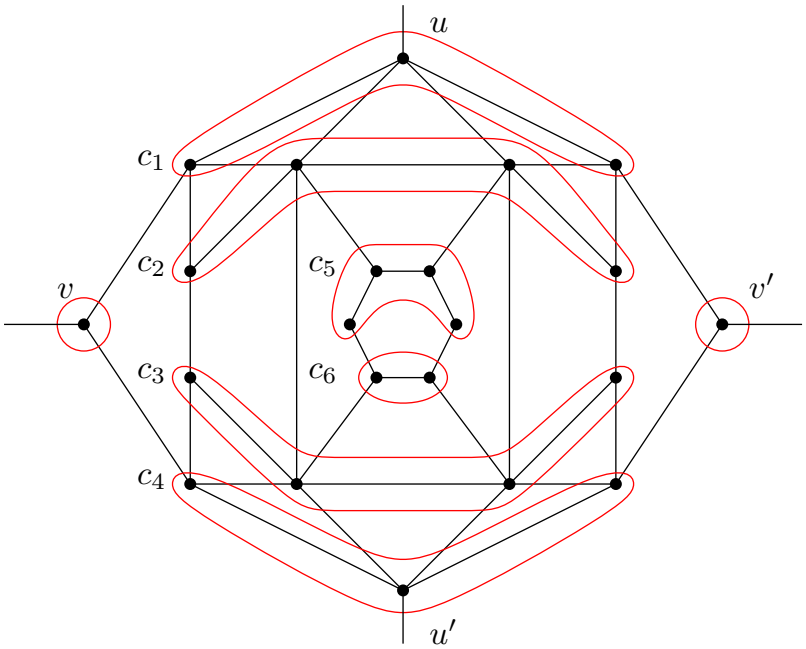


Figure 6.3: Cleaners in a mixed search game for $G_\times$. The positions for each cleaner c_i are numbered from left to right.

u' to a vertex outside $G_\times$ might not be clean. When S is finished, there is a cleaner on u' and v', all edges in $G_\times$ are cleaned, and no recontamination has taken place. □

Claim 6. *To execute mixed search strategy S it suffices to have 6 cleaners.*

Proof. We start with 2 cleaners on v and w. Then we introduce four new cleaners, adding up to six cleaners. We then remove the cleaner on v, and when c_1 is on its second position, we also remove the cleaner on w, reducing the number of cleaners to four. Cleaners c_5 and c_6 are introduced, but are also the first to be removed again, which means we still have at most six cleaners. At this point, a cleaner is left behind at u' and a new cleaner is put on the last position of c_4, which means there are five cleaners. Finally a cleaner is put on v', which means we again have six cleaners, after which c_1, c_2, c_3, c_4 are removed and S is finished. As we have seen, six cleaners suffice to carry out S. □

By Claims 5 and 6 we can conclude that mixed search strategy S fulfills all requirements of Lemma 19. □

We will use the strategy in Lemma 19 to prove a bound on the pathwidth of a planar t-boundaried graph G_f^p.

Claim 7. *For planar t-boundaried graph G_f^p holds that $\mathbf{pw}(G) \leq t + 6$.*

Proof. The mixed search strategy we are going to use to bound the pathwidth of planar t-boundaried graph G_f^p follows the same structure as the strategy we used for property 4 of

Lemma 13. However, this time we have to put in more work to get the cleaners through the crossover gadgets.

To clean G_f^p we use a similar strategy S as for the non-planarized graphs: We start by putting a cleaner at vertex $p_{i,1}$ for every path P_i. We can clean the whole graph in m phases of the following process. At the start of phase j, the cleaner on path P_i is at vertex $p_{i,2j-1}$. Between vertices $p_{i,2j-1}$ and $p_{i,2j}$ of each path, there can be crossover gadgets to connect clause gadget G_{C_j} to the paths. In order for the path cleaner on path P_i to get to vertex $p_{i,2j}$, we need to clean G_{C_j} simultaneously.

We will first look at how mixed search strategy S plays out for the whole graph G_f^p. Figure 6.4 will be used to illustrate different situations in phase j of the mixed search strategy. When we have established strategy S, we will show that it actually cleans G_f^p without any recontamination, and that $t + 6$ cleaners suffice to carry out S.

The start of phase j can be seen in Figure 6.4a: There is a cleaner on vertex $p_{i,2j-1}$ for every path P_i, and we put one cleaner on vertex v_{start} in the clause gadget G_{C_j}. The cleaner at vertex v_{start} slides it to u_1 and we introduce a cleaner at the first terminal vertex of G_{C_j}. Then the cleaner at u_1 slides to w_1. The cleaners in the t paths can now move along one edge of their path, so that the cleaner on path P_i arrives at either vertex $p_{i,2j}$, or vertex v of the first crossover gadget in path P_i. Observe that for path P_1 we can now clean the first crossover gadget using Lemma 19, since all the conditions in the lemma have been met.

Looking at Figure 6.4b, we see situation during the cleaning of the crossover gadget, that occurs right before the cleaner at the termial vertex of G_{C_j} will be removed. When the crossover gadget has been cleaned using Lemma 19, the terminal vertex will no longer have a cleaner on it, and there are cleaners on vertices u' and v'. The cleaner that is placed on vertex v' of the crossover gadget will proceed along the path until it either reaches $p_{1,2j}$ or gets to vertex v of another crossover gadget.

At this point, the first crossover gadget on path P_2 can be cleaned by Lemma 19. Figure 6.4c shows a situation that happens during this process, right before the cleaner that was left behind on the previous crossover gadget is removed. We again end up with cleaners on vertex u' and v'. We can slide the cleaner on v' to $p_{2,2j}$ or to vertex v of another crossover gadget.

Now the first crossover gadget on path P_3 can be cleaned by Lemma 19. This pattern of cleaning a crossover gadget in path P_i to enable a crossover gadget in path P_{i+1} to be cleaned, repeats itself until we reach a crossover gadget in a path P_i where u' is connected to $p_{i+1,2j}$. We now remove the cleaner on u' and arrive at the situation in Figure 6.4d.

We can now slide the cleaner on vertex w_1 in G_{C_j} towards vertex v' in G_{C_j}. At this point we can distinguish two cases:

- If we do not reach v', but vertex u_2, we can introduce a new cleaner at the next terminal vertex and slide the cleaner on u_2 to w_2. The process we described before, where crossover gadgets can be cleaned by Lemma 19 until there is a cleaner on a vertex connected to $p_{i,2j}$, for some $1 \leq i \leq t$, now repeats itself.

- If we reach v_{end}, we can remove the cleaner from v_{end}, and we are sure that the cleaner in every path P_i has reached $p_{i,2j}$. The cleaner in every path P_i can slide to vertex $p_{i,2(j+1)-1}$, and we arrive at the start of phase $j + 1$.

Mixed search strategy S consists of the phases 1 to m as described above. We can prove that S cleans G_f^p without recontamination and it suffices to use $t + 6$ cleaners.

Claim 8. *Mixed search strategy S cleans all edges of G_f^p without allowing any recontamination.*

Proof. Observe that phase j starts with a cleaner on every vertex $p_{i,2j-1}$ for every path P_i, and the edge between $p_{i,2j-1}$ and $p_{i,2(j-1)}$ is clean. This means that if we clean all the vertices between $p_{i,2j-1}$ and $p_{i,2j}$ in phase j, without recontaminating the edge between $p_{i,2j-1}$ and $p_{i,2(j-1)}$, then after m phases, G_f^p is cleaned without allowing any recontamination.

During phase j of S, isolated clause gadget G_{C_i} is cleaned by sliding cleaner c_1 from v_{start} to v_{end}, and putting cleaner c_2 on the terminal vertex that is connected to u_x and w_x, when c_1 arrives at u_x. By Lemma 19 we know that when c_2 is removed, then G_{C_i} cannot be recontaminated via the terminal vertex. In fact, we can only use Lemma 19 when c_1 is on w_x, since we can then ensure that removing c_2 will not contaminate the crossover gadget.

Path P_i is cleaned between vertices $p_{i,2j-1}$ and $p_{i,2j}$ during phase j. We either slide the cleaner on path P_i along a single edge of the path, or we use Lemma 19 to clean a crossover gadget along the path. In both cases we clean part of the path and do not allow recontamination.

What is left are the edges going from some terminal vertex towards a vertex $p_{i,2j}$. Since we ordered the variables in clause C_j in the same way as the paths, the first crossover gadgets we encounter going from were introduced to resolve crossings for the edge that goes to the vertex $p_{x,2j}$ with the highest index x for all the edges of that particular clause. The second crossover gadget between $p_{i,2j-1}$ and $p_{i,2j}$, resolves crossings for the edge to $p_{y,2j}$ with y being the second highest index, and so on. This means that after we cannot apply Lemma 19 any more, we end up with a cleaner on some vertex u', and there is no uncleaned crossover on the next path left. At the start of the phase, and after every application of Lemma 19 we move the cleaner on path P_i to either vertex v of the next crossover gadget, or to $p_{i,2j}$. Therefore there must be a cleaner on $p_{i,2j}$, which is the vertex u' is connected to, which means we have also cleaned the whole path from terminal vertex to $p_{i,2j}$.

As we have seen, in phase j we can clean all the edges between $p_{i,2j-1}$ and $p_{i,2j}$, without allowing any recontamination. This means that after m phases, G_f^p is fully cleaned and no recontamination has taken place. □

Claim 9. *To execute mixed search strategy S it suffices to have $t + 6$ cleaners.*

Proof. We start with t cleaners, having one cleaner on each path. We then place a single cleaner on v_{start}, that will slide towards v_{end}. If we reach v_{end}, we have the cleaner in path P_i on vertex $p_{i,2j}$, which means this cleaner in the clause gadget will be used throughout a whole phase.

Finally there is a cleaner that we put on a terminal vertex. This cleaner will be removed by Lemma 19, and at the end of that process be left behind on vertex u' of the crossover gadget that was being cleaned. Lemma 19 stated that it suffices to use six cleaners for cleaning a crossover gadget, counting the cleaner on vertex v and the cleaner on the other end of the edge connected to vertex u, and the cleaners we leave at vertices u' and v' of the crossover gadget.

This means that we use t cleaners on the paths and one cleaner in the clause gadget. During the process of cleaning a crossover gadget we use at most six cleaner, and take one of the path cleaners (the one on vertex v of the crossover gadget) and the cleaner that starts at a terminal vertex and is left behind at u' after every application of Lemma 19. In total there are $t + 1 + 1 + 6 - 2$ cleaners in play at the same time during mixed search strategy S. Hence it suffices to have $t + 6$ cleaners. □

By Claims 8 and 9 we can conclude that $\mathbf{ms}(G_f^p) \leq t + 6$ and therefore $\mathbf{pw}(G_f^p) \leq t + 6$ □

We know by Claim 7 that planar t-boundaried graph G_f^p still has a bounded treewidth, and this bound is not much higher than for the non-planarized graph G_f. Next we will prove that even for

planar t-boundaried graph G_f^p the property 3 of Lemma 15 still holds.

Claim 10. *For planar t-boundaried graph G_f^p with boundary $B = \{p_{1,1}, \ldots, p_{n,1}\}$ holds that an assignment of values $x_1, \ldots, x_t \in \{0,1\}$ ensures that $f(x_1, \ldots, x_t) = 1$ if and only if $f_{G_f}(\{p_{i,1} \mid x_i = 1\}) = f_{G_f}(B)$.*

Proof. We prove both sides of the bi-implication separately:

($\Rightarrow$) Assume we have an assignment of variables $x_1, \ldots, x_t \in \{0,1\}$ that ensures $f(x_1, \ldots, x_t) = 1$. We used Lemma 13 and 14 in the construction of G_f^p, so to construct G_f^p we used a monotone CNF formula ϕ for which $\phi(x_1, \ldots, x_t) = 1$ if and only if $f(x_1, \ldots, x_t) = 1$. This means that every clause $C_j \in \phi$ has at least one positive literal ℓ_a for which variable x_{ℓ_a} becomes 1 by this assignment of variables. By property 3 of Lemma 12 we know that for every clause gadget G_{C_j}, we can find an independent set X_{C_j} of size $|C_j| + 2$, which contains exactly one terminal vertices. We are interested in the case where $v_a \in X_{C_j}$, corresponding to ℓ_a with $x_{\ell_a} = 1$.

Assume we have an independent set X_{P_i} of size m for every path P_i in G_f^p, which contains for $1 \leq k \leq m$

- all vertices $p_{i,2k-1}$ if $x_i = 1$
- all vertices $p_{i,2k}$ if $x_i = 0$

For the crossover gadgets we find independent sets as follows. We call the set of crossover gadgets between $p_{i,2k-1}$ and $p_{i,2k}$ a *row*. The set of crossover gadgets between a terminal vertex v_a and path vertex $p_{\ell_a,2k}$ will be called a *column*. By Table 6.1b we know that there is an independent set of size nine for every crossover gadget $X_\times$ in the column of terminal vertex $v_a \in G_{C_j}$ and row between $p_{i,2k-1}$ and $p_{i,2k}$ such that:

- $u' \in X_\times$ and $u \notin X_\times$ if $v_a \in X_{C_j}$, and $u \in X_\times$ and $u' \notin X_\times$ otherwise;
- $v' \in X_\times$ and $v \notin X_\times$ if $p_{i,2k-1} \in X_{P_i}$, and $v \in X_\times$ and $v' \notin X_\times$ if $p_{i,2k} \in X_{P_i}$

We know all terminal vertices of clause gadget G_{C_j} are either connected to path vertex $p_{i,2j}$ by a column of crossover gadgets or not connected to a path P_i at all. If for l_a the variable $x_{\ell_a} = 1$, then $v_a \in X_{C_i}$ is connected to vertex $p_{\ell_a,2j}$ of path P_{ℓ_a} by a column of crossover gadgets. Since $x_{\ell_a} = 1$, we know that $p_{\ell_a,2j} \notin X_{P_{\ell_a}}$, and because terminal vertex $v_a \in X_{C_i}$ we have $u' \in X_\times$ and $u \notin X_\times$ for all crossover gadgets in that column. Hence $X := \bigcup_{1 \leq j \leq m} X_{C_j} \cup \bigcup_{1 \leq i \leq t} X_{P_i} \cup \bigcup X_\times$ is an independent set of size $mt + 9N_\times + (\sum_{1 \leq i \leq m} |C_i| + 2)$. Furthermore X_{P_i} only contains all vertices $p_{i,2j-1}$ if $x_i = 1$, so in particular $p_{i,1} \in X_{P_i}$. Hence we can conclude that $X \cap B = \{p_{i,1} \mid x_i = 1\}$.

We now know that $mt + 9N_\times + (\sum_{1 \leq i \leq m} |C_i| + 2) = |X| \leq f_{G_f^p}(\{p_{i,1} \mid x_i = 1\})$. By Lemma 6 we know that $f_{G_f^p}(\{p_{i,1} \mid x_i = 1\}) \leq f_{G_f^p}(B)$. However, Claim 3 tells us that any independent set for G_f^p has size at most $mt + 9N_\times + (\sum_{1 \leq i \leq m} |C_i| + 2)$ thus $f_{G_f^p}(B) \leq mt + 9N_\times + (\sum_{1 \leq i \leq m} |C_i| + 2)$. Hence $f_{G_f^p}(\{p_{i,1} \mid x_i = 1\}) = f_{G_f^p}(B)$.

($\Leftarrow$) Assume that for a particular set $S \subseteq B$ holds that $f_{G_f^p}(S) = f_{G_f^p}(B)$. By Claim 4 we can conclude that $f_{G_f^p}(S) = mt + 9N_\times + (\sum_{1\leq i\leq m} |C_i| + 2)$. Using Definition 4, we know that there is an independent set X of size $mt + 9N_\times + (\sum_{1\leq i\leq m} |C_i| + 2)$ for G_f^p such that $X \cap B \subseteq S$.

The only way X can have size $mt + 9N_\times + (\sum_{1\leq i\leq m} |C_i| + 2)$ is when $|X \cap G_{C_j}| = |C_j| + 2$ for $1 \leq j \leq m$, since $|X \cap G_\times| \leq 9$ for all crossover gadgets $G_\times$ and $|X \cap P_i| \leq m$ for $1 \leq i \leq t$. By property 2 of Lemma 12 we know that X contains at least one terminal vertex v of every clause gadget G_{C_i}. We can now prove the following claim.

Claim 11. *If clause gadget G_{C_i} has a terminal vertex $v \in X$ connected to $p_{i,2j}$ by a column of crossover gadgets, then $p_{i,1} \in X$.*

Proof. Since X has size $mt + 9N_\times + (\sum_{1\leq i\leq m} |C_i| + 2)$ and for every clause gadget $|C_i|$ holds that $|X \cap G_{C_j}| = |C_j| + 2$, we know that $mt + 9N_\times$ vertices of X are located in the paths and crossover gadgets. There are two ways in which this can happen:

- For every crossover gadget $G_\times$ we have $|X \cap G_\times| = 9$. In this case, for every path P_i, we know that $|X \cap P_i| = m$. This means that for every crossover gadget in the column that connects terminal vertex v to $p_{i,2j}$, it must be that X contains u', and thus $p_{i,2j} \notin X$. For every crossover gadget $G_\times$ on P_i holds $|X \cap G_\times| = 9$, hence $X \cap G_\times$ contains either v or v' but not both, as $|X \cap P_i| = m$. Since $p_{i,2j} \notin X$, the crossover gadgets between $p_{i,2j-1}$ and $p_{i,2j}$ all have $v' \in X$, thus $p_{i,2j-1} \in X$ otherwise $|X \cap P_i| < m$. We also know that $p_{i,2j-1}$ has an edge to $p_{i,2(j-1)}$, so $p_{i,2(j-1)} \notin X$. This continues in this way, and so every $p_{i,2k-1} \in X$ for $1 \leq k \leq j$, so in particular $p_{i,1} \in X$.

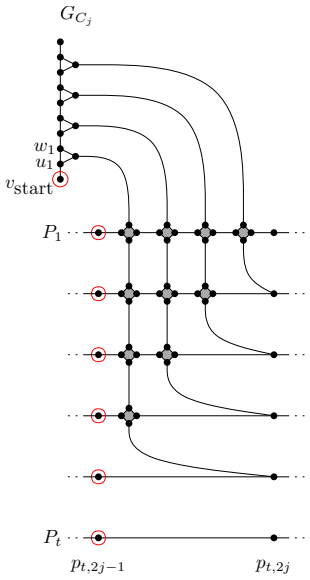
- There is a crossover gadget $G_\times$ for which $|X \cap G_\times| < 9$, which means there is a path P_a for which $|X \cap P_a| > m$. The only way that P_a can have $|X \cap P_a| > m$ is when $p_{a,2k-1} \in X$ and $p_{a,2k} \in X$ for some $1 \leq k \leq m$, since there is an edge between every $p_{a,2k}$ and $p_{a,2(k+1)-1}$. As a consequence, crossover gadget $G_\times$ must be between $p_{a,2k-1} \in X$ and $p_{a,2k} \in X$ and has $v, v' \notin X$, resulting in $|X \cap G_\times| < 9$. Observe that we need that $p_{a,2(k-1)} \notin X$ and $p_{a,2(k+1)-1} \notin X$, otherwise X is not an independent set. To ensure that $|X \cap P_a| > m$, this situation where $p_{a,2k-1}, p_{a,2k} \in X$ and $p_{a,2(k-1)}, p_{a,2(k+1)-1} \notin X$ can only occur once in path P_i. Since in our case $p_{i,2j} \notin X$, such a construction can only happen for a $j' > j$. Therefore we know that $p_{i,2j-1} \in X$. Again every $p_{i,2k-1} \in X$ for $1 \leq k \leq j$, so in particular $p_{i,1} \in X$. □

We used Lemma 13 and 14 to construct G_f^p, so there is a monotone CNF ϕ for which $\phi(x_1, \ldots, x_t) = 1 \Leftrightarrow f(x_1, \ldots, x_t) = 1$, that we used during this construction. In X there is at least one terminal vertex $v_a \in X$, connected to $p_{\ell_a,2j}$ by a column of crossover gadgets, for every G_{C_j}, and by Claim 11 we know $p_{\ell_a,1} \in X$ is a consequence of this. Since $|X \cap B| \subseteq S$, we can assign $x_i = 1$ if $p_{i,1} \in S$, and $x_i = 0$ if $p_{i,1} \notin S$ and have at least one variable $x_{\ell_a} = 1$ for every $C_j \in \phi$ that was used to construct G_f^p. As ϕ is a monotone CNF formula, this assignment ensures that that $\phi(x_1, \ldots, x_t) = 1$ and therefore also $f(x_1, \ldots, x_t) = 1$. □

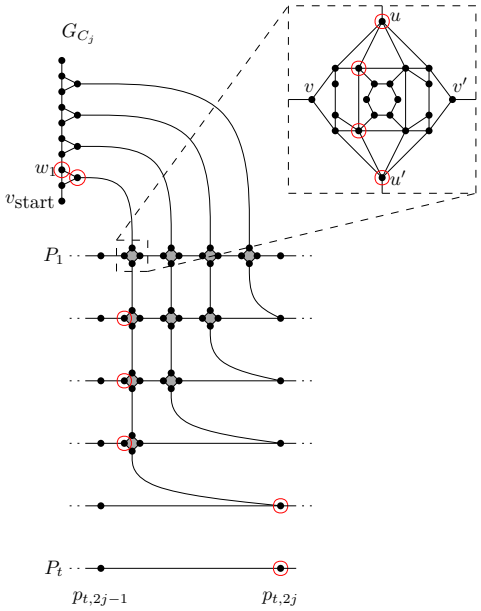
Claims 3, 4, 7 and 10 combined prove Lemma 18. Using Lemma 18 we can now prove a lower bound on the number of equivalence classes for INDEPENDENT SET on planar t-boundaried graphs of treewidth at most $t + 6$. The lower bound is the same bound as we had for t-boundaried graphs of treewidth at most $t + 2$.

Lemma 20. *There are at least $M(t) - 1$ equivalence classes for independent set on planar t-boundaried graphs of treewidth at most $t + 6$.*

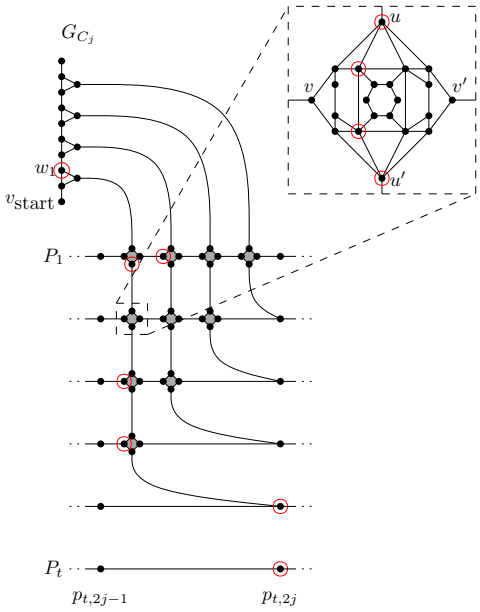
Proof. We use Lemma 16 to find $M(t) - 1$ distinct Boolean functions for which Lemma 18 can construct a planar t-boundaried graph of treewidth at most $t + 6$. Since none of the $M(t) - 1$ graphs are equivalent by Lemma 16, we have proved that there are at least $M(t) - 1$ equivalence classes for INDEPENDENT SET on planar t-boundaried graphs of treewidth at most $t + 6$. □



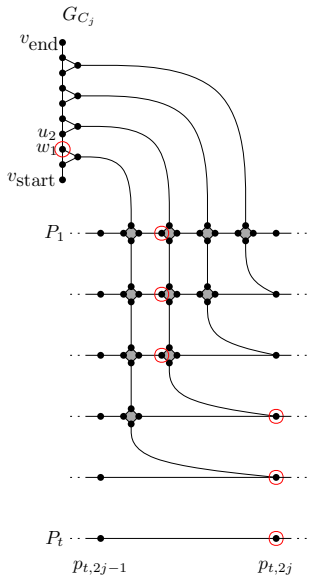
(a) Placement of cleaners at start of phase j



(b) Cleaning first crossover gadget of first column



(c) Cleaning second crossover gadget of first column



(d) Cleaners after cleaning first column

Figure 6.4: Phase j of a mixed search strategy for G_f^p

Chapter 7

Size of a representative

In the previous chapter we investigated the equivalence classes for INDEPENDENT SET on t-boundaried graphs. For different types of graphs we proposed lower bounds on the number of equivalence classes. We will now use these results to derive a lower bound on the size of a representative for planar t-boundaried graphs of bounded treewidth.

Let R_t be a set of planar t-boundaried graphs of treewidth at most $t+6$, such that for any planar t-boundaried graph H of treewidth at most $t+6$, there is some graph in R_t that is equivalent to H. This set R_t contains a representative for every equivalence class.

Lemma 21. *Every set of representatives R_t for* INDEPENDENT SET, *such that for any planar t-boundaried graph H of treewidth at most $t+6$ there is a planar t-boundaried graph $H_R \in R$ with $H \equiv_{\text{IS}} H_R$, contains a graph with $\Omega(\log M(t))$ vertices.*

Proof. Observe that for every equivalence class there is a distinct graph in R_t. To prove a lower bound on the size of one of these representatives, we can count how many distinct small planar t-boundaried graphs of treewidth at most $t+6$ there are: Remember that the bound we derived on the number of equivalence classes for planar t-boundaried graphs of bounded treewidth in Lemma 20 is $M(t)-1$. If there are less than $M(t)-1$ graphs of size at most x, then there is a representative of size at least $x+1$. This means that, to prove the statement in the lemma, we have to show that there are constants c, t_0, such that for every $t \geq t_0$ holds that

$$\sum_{n=t}^{c \log M(t)} N(n) < M(t) - 1.$$

We count $N(n)$, which is the number of distinct planar t-boundaried graphs of treewidth at most $t+6$ with n vertices. We show that the number of graphs of size between $n=t$ and $c \log M(t)$ does not add up to at least one graph for each of the $M(t)-1$ equivalence classes.

To find an upper bound on $\sum_{n=t}^{c \log M(t)} N(n)$, we split it into two separate sums:

$$\sum_{n=t}^{t^2-1} N(t) + \sum_{n=t^2}^{c \log M(t)} N(t)$$

There are at most 31^n unlabeled planar graphs of size n, according to Bonichon et al. [7]. However we want to count graphs that have a labeled boundary of t vertices, and there are less

than $31^n n^t$ different planar graphs with a labeled boundary of t vertices. We use $N(t) \leq 31^n n^t$, since this overestimates the number of distinct planar t-boundaried graphs. However, for the smaller graphs we substitute $N(t)$ for the upper bound on the number of labeled graphs $2^{n(n-1)/2}$. This bound is easier to work with, although it is an even bigger overestimation.

$$\sum_{n=t}^{c \log M(t)} N(n) < \sum_{n=t}^{t^2-1} 2^{n(n-1)/2} + \sum_{n=t^2}^{c \log M(t)} 31^n n^t \tag{7.1}$$

We chose to split at $n = t^2$, since we can prove that $n^t \leq 2^n$, which in return allows us to simplify the sum for the bigger graphs.

Claim 12. *For every $t \geq 4$ and $t^2 \leq n \leq 2^t$, the inequality $n^t \leq 2^n$ holds.*

Proof. By substituting $n = t^2$ in $n^t \leq 2^n$, we get $t^{2t} \leq 2^{t^2}$, for which we can show the following

$$\begin{aligned}
(t^2)^t &\leq 2^{t^2} && \\
2^{\log t^2 * t} &\leq 2^{t^2} && a = 2^{\log a} \\
\log t^2 * t &\leq t^2 && \log \text{ of both sides} \\
\log t^2 &\leq t && \text{division by } t \\
2 \log t &\leq t && \\
2t/2 &\leq t && \log t \leq t/2 \text{ for } t \geq 4
\end{aligned}$$

Similarly by substituting $n = 2^t$, we get $(2^t)^t = 2^{t^2}$ □

Using Claim 12 we simplify the second sum in Inequality 7.1 as follows.

$$\sum_{n=t^2}^{c \log M(t)} 31^n n^t < \sum_{n=t^2}^{c \log M(t)} 31^n 2^n = \sum_{n=t^2}^{c \log M(t)} 62^n. \tag{7.2}$$

Finally we can use the sum of a geometric series shown in Chapter 2 to find an upper bound on Inequality 7.1 and with that a bound on $\sum_{n=t}^{c \log M(t)} N(n)$.

$$\begin{aligned}
\sum_{n=t}^{c \log M(t)} N(n) &< \sum_{n=t}^{t^2-1} 2^{n(n-1)/2} + \sum_{n=t^2}^{c \log M(t)} 31^n n^t && \text{Inequality 7.1} \\
&< \sum_{n=t}^{t^2-1} 2^{n(n-1)/2} + \sum_{n=t^2}^{c \log M(t)} 62^n && \text{Inequality 7.2} \\
&= 2 \cdot 2^{(t^2-1)(t^2-2)/2} - 2^{t(t-1)/2} + \sum_{n=t^2}^{c \log M(t)} 62^n && \text{Equation 2.1} \\
&= 2 \cdot 2^{(t^2-1)(t^2-2)/2} - 2^{t(t-1)/2} + \frac{62 \cdot 62^{c \log M(t)} - 62^{t^2}}{61} && \text{Equation 2.1} \\
&= 2 \cdot 2^{(t^2-1)(t^2-2)/2} - 2^{t(t-1)/2} + \frac{62}{61} \cdot (62^c)^{\log M(t)} - \frac{62^{t^2}}{61} && \\
&= 2 \cdot 2^{(t^2-1)(t^2-2)/2} - 2^{t(t-1)/2} + \frac{62}{61} \cdot (3/2)^{\log M(t)} - \frac{62^{t^2}}{61} && \text{Choose } c = \frac{\log(3/2)}{\log 62}
\end{aligned}$$

One can check that for $t \geq t_0 = 19$ holds that

$$2 \cdot 2^{(t^4-3*t^2+2)/2} - 2^{t(t-1)/2} + \frac{62}{61} \cdot (3/2)^{\log M(t)} - \frac{62^{t^2}}{61} < 2^{\log M(t)} - 1 = M(t) - 1.$$

We have now found constants $c = \frac{\log(3/2)}{\log 62}$ and $t_0 = 19$, hence $\Omega(\log M(t))$ is a lower bound on the size of a representative in the set R_t of representatives for INDEPENDENT SET, such that for any planar t-boundaried graph H of treewidth at most $t+6$ there is a planar t-boundaried graph $H_R \in R$ with $H \equiv_{\text{IS}} H_R$. □

Using Stirling's approximation that we showed in Chapter 2 we can simplify the bound on the number of vertices of a representative to $\Omega(\log M(t)) \geq \Omega(2^t/\sqrt{4t})$.

Chapter 8

Conclusions

We have presented a new way to find lower bounds on a representative in the set of representatives R_t, by counting equivalence classes: We successfully find lower bounds on the number of equivalence classes for INDEPENDENT SET on many types of graphs. Then we apply a counting argument to find a lower bound on the size of a representative. This counting argument can be applied to any problem for which we can find a lower bound on the number of equivalence classes and an upper bound on the number of distinct t-protrusions in a certain graph class. It would be very interesting to see if we can find similar results for DOMINATING SET as we did for INDEPENDENT SET. For VERTEX COVER, the complement of INDEPENDENT SET, there are already good results for the running time of a kernelization algorithm and the kernel size [8], so results via meta kernelization will not improve on them. On the other hand, finding new bounds for DOMINATING SET by using protrusion replacement or meta kernelization, can lead to new results for specific graph classes.

We expect the same lower bounds for DOMINATING SET as we presented for INDEPENDENT SET. When solving DOMINATING SET, we look for a set of vertices such that every vertex in a graph is either in the set or has an edge to a vertex in the set. There is a reduction from VERTEX COVER to DOMINATING SET, that for every edge $\{u, v\}$ introduces two new vertices and connects them both to u and v by an edge. The new vertices can be dominated by taking either u or v in the dominating set, which corresponds to a vertex cover on the initial graph, where we want at least one endpoint of each edge in the vertex cover. We can apply this reduction on our planar t-boundaried graphs of bounded pathwidth, while keeping the graph planar and increasing the pathwidth by at most one. This indicates that we can probably find just as many equivalence classes for DOMINATING SET on these graphs. We should however check if these new boundaried graphs are all still in different equivalence classes.

Going through the same steps for DOMINATING SET, as we did in this thesis for INDEPENDENT SET, will probably result in better lower bounds. An indication for this is the lower bound by Lokshtanov et al. [20] for DOMINATING SET based on SETH. This exponential lower bound has a base of 3, while the exponential lower bound for INDEPENDENT SET has a base of 2. The paper by Lokshtanov et al. [20] is also a good starting point in the search for bounded treewidth graph gadgets for DOMINATING SET: It shows a reduction from CNF Satisfiability to DOMINATING SET, which results in bounded pathwidth graphs. However, choosing the boundary naively in these graphs will result in a bound on the pathwidth that is exponential in the boundary size.

In this thesis we also propose an upper and lower bound on the number of equivalence classes

for INDEPENDENT SET on general t-boundaried graphs. The bounds are 2^{2^t-1} and $M(t)-1$ respectively. We can find a pair of matching bounds if t-representative functions can be counted precisely, in the same way as Dedekind numbers count the number of monotone Boolean functions. Note that the upper bound of 2^{2^t-1} is already an improvement on the bound of $(t+1)^{2^t}$ which can be found by following the approach by Garnero et al. [16].

Since we have found an upper bound of 2^{2^t-1} equivalence classes for general t-boundaried graphs, we know that our lower bound of $M(t)-1$ for t-boundaried graphs of treewidth at most $t+2$ and planar t-boundaried graphs of treewidth at most $t+6$, is already very close to the upper bound: $2^{2^t/\sqrt{4t}} \leq M(t)-1 \leq 2^{2^t-1}$. These results for t-protrusions of bounded treewidth actually also hold for t-protrusions of bounded pathwidth, since we proved the treewidth bounds using mixed search games.

The bound of $\Omega(\log M(t)) \geq \Omega(2^t/\sqrt{4t})$ on the number of vertices in a representative for INDEPENDENT SET on planar t-boundaried graphs of treewidth at most $t+6$ is less tight. The existing upper bound that can be inferred by following Garnero et al. [16] is $2^{(t+1)^{2^t}}$. This bound is found by dynamic programming on a (binary/nice) tree decomposition: If one can find a tree decomposition, one can also find a tree decomposition that is a binary tree. The main idea behind the dynamic programming bound is that in every path from the root of the tree decomposition to a leaf, a representative of a particular equivalence class can only occur once. Since they proved an upper bound of $(t+1)^{2^t}$ on the number of equivalence classes, there can be $2^{(t+1)^{2^t}}$ paths from root to leaf in a binary tree decomposition, in which no representative occurs twice. Even though we improved this upper bound to $2^{2^{2^t-1}}$, by improving the bound on the number of equivalence classes, it is still far off our lower bound of $\Omega(2^t/\sqrt{4t})$.

Deriving a similar upper bound for planar boundaried graphs of bounded pathwidth using path decompositions leads to an upper bound of 2^{2^t-1}, since a representative for a particular equivalence class can only occur once on the path of the path decomposition. This upper bound is already a lot closer to our lower bound, which is also an indication that there might still be room for improvement when it comes to the upper bounds.

The upper bounds via dynamic programming can be applied to any problem, and we improved the upper bound on the number of equivalence classes by investigating protrusion replacement for INDEPENDENT SET specifically. Investigating the upper bounds for small representatives on specific problems can probably lead to better bounds as well. On the other hand, the lower bound on the number of equivalence classes for INDEPENDENT SET is already close to the corresponding upper bound, and we proposed a way to find matching bounds. Therefore, the best way to improve our process of finding a lower bound on the number of vertices of a representative for INDEPENDENT SET would be by being smarter about the way we apply the counting argument.

Bibliography

[1] Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. Polynomial-time data reduction for dominating set. *Journal of the ACM (JACM)*, 51(3):363–384, 2004. 2

[2] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009. 1

[3] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 629–638. IEEE, 2009. 2, 5, 6

[4] Hans L. Bodlaender and Eelko Penninkx. A linear kernel for planar feedback vertex set. In *Parameterized and Exact Computation*, pages 160–171. Springer, 2008. 2

[5] Hans L. Bodlaender, Eelko Penninkx, and Richard B. Tan. A linear kernel for the k-disjoint cycle problem on planar graphs. In *Algorithms and Computation*, pages 306–317. Springer, 2008. 2

[6] Hans L. Bodlaender and Babette van Antwerpen-de Fluiter. Reduction algorithms for graphs of small treewidth. *Information and Computation*, 167(2):86–119, 2001. 6

[7] Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, Dominique Poulalhon, and Gilles Schaeffer. Planar graphs, via well-orderly maps and trees. *Graphs and Combinatorics*, 22(2):185–202, 2006. 36

[8] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40):3736–3756, 2010. 39

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*, volume 6. MIT press Cambridge, 2001. 7

[10] Babette de Fluiter. *Algorithms for graphs of small treewidth*. PhD thesis, Utrecht University, 1997. 5, 6

[11] Rodney G. Downey and Michael R. Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012. 1, 2

[12] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. Hitting forbidden minors: Approximation and kernelization. *SIAM Journal on Discrete Mathematics*, 30(1):383–410, 2016. 4

[13] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar f-deletion: Approximation, kernelization and optimal fpt algorithms. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 470–479. IEEE, 2012. 4

[14] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 503–510. Society for Industrial and Applied Mathematics, 2010. 3

[15] Michael R. Garey, David S. Johnson, and Larry Stockmeyer. Some simplified NP-complete graph problems. *Theoretical computer science*, 1(3):237–267, 1976. 25, 26

[16] Valentin Garnero, Christophe Paul, Ignasi Sau, and Dimitrios M. Thilikos. Explicit linear kernels via dynamic programming. *SIAM Journal on Discrete Mathematics*, 29(4):1864–1894, 2015. iii, 3, 4, 40

[17] Jiong Guo and Rolf Niedermeier. Linear problem kernels for np-hard problems on planar graphs. In *Automata, languages and programming*, pages 375–386. Springer, 2007. 2

[18] Iyad Kanj, Michael J. Pelsmajer, Marcus Schaefer, and Ge Xia. On the induced matching problem. *Journal of Computer and System Sciences*, 77(6):1058–1070, 2011. 2

[19] Eun J. Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. *ACM Transactions on Algorithms (TALG)*, 12(2):21, 2015. 3

[20] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 777–789. SIAM, 2011. 17, 39

[21] Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization–preprocessing with a guarantee. In *The Multivariate Algorithmic Revolution and Beyond*, pages 129–161. Springer, 2012. 2

[22] Daniel Lokshtanov, Matthias Mnich, and Saket Saurabh. Linear kernel for planar connected dominating set. In *Theory and Applications of Models of Computation*, pages 281–290. Springer, 2009. 2

[23] Dorian Pyle. *Data preparation for data mining*, volume 1. Morgan Kaufmann, 1999. 1

[24] Atsushi Takahashi, Shuichi Ueno, and Yoji Kajitani. Mixed searching and proper-path-width. *Theoretical Computer Science*, 137(2):253–268, 1995. 6