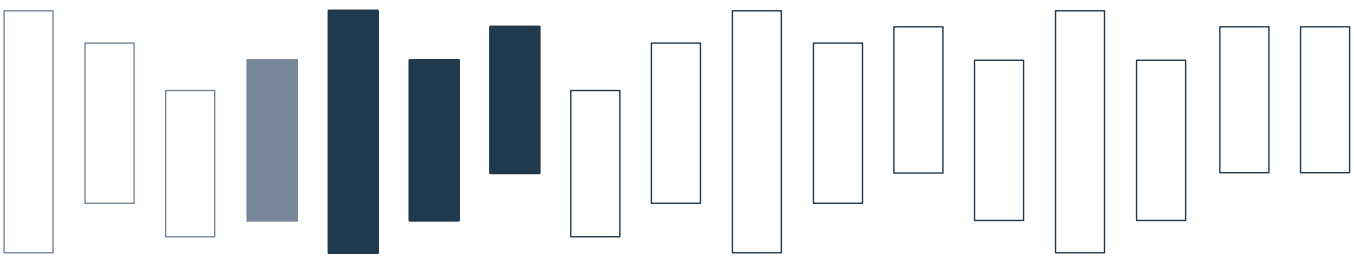


Technical Report AC-TR-21-004

February 2021

Fixed-Parameter Tractability

Marko Samer and Stefan Szeider

This is the authors' copy of Chapter 17 of the Handbook of Satisfiability, 2nd Edition, 2021, pp. 693–736, ISBN 978-1-64368-160-3.

www.ac.tuwien.ac.at/tr

Chapter 17

Fixed-Parameter Tractability

Marko Samer† and Stefan Szeider

17.1. Introduction

The propositional satisfiability problem (SAT) is famous for being the first problem shown to be NP-complete—we cannot expect to find a polynomial-time algorithm for SAT. However, over the last two decades, SAT-solvers have become amazingly successful in solving formulas with thousands of variables that encode problems arising from various application areas. Theoretical performance guarantees, however, are far from explaining this empirically observed efficiency. Theorists believe that the trivial 2^n time bound for solving SAT instances with n variables cannot be significantly improved, say to $2^{o(n)}$ (see the end of Section 17.2.1). This enormous discrepancy between theoretical performance guarantees and the empirically observed performance of SAT solvers can be explained by the presence of a certain "hidden structure" in instances that come from applications. This hidden structure greatly facilitates the propagation and simplification mechanisms of SAT solvers. Thus, for deriving theoretical performance guarantees that are closer to the actual performance of solvers, one needs to take this hidden structure of instances into account. The literature contains several suggestions for making the vague term of a hidden structure explicit. For example, the hidden structure can be considered as the "tree-likeness" or "Horn-likeness" of the instance (below we will discuss how these notions can be made precise). All such concepts have in common that one associates with a CNF formula F a non-negative integer $k = \pi(F)$; the smaller the integer, the more structured the instance from a certain perspective. We call such a mapping π a *satisfiability parameter* or a *parameterization* of the satisfiability problem. We will also write $\pi(F) = \infty$ to express that ϕ is not defined for F. Consider a satisfiability parameter π. For each integer k one can consider the class $\mathcal{C}_k^\pi$ of formulas F such that $\pi(F) \leq k$. This gives rise to an infinite hierarchy $\mathcal{C}_0^\pi \subseteq \mathcal{C}_1^\pi \subseteq \mathcal{C}_2^\pi \subseteq \cdots \mathcal{C}_\infty^\pi$ of classes. Every CNF formula F belongs to some $\mathcal{C}_k^\pi$ for k sufficiently large (namely, $k = \pi(F)$).

We are interested in satisfiability parameters π such that satisfiability of instances in $\mathcal{C}_k^\pi$ and membership in $\mathcal{C}_k^\pi$ can be decided in polynomial time (the latter property can be relaxed). The larger we make k (thus, the more general the

†1977–2010

class $\mathcal{C}_k^\pi$), the worse we expect the performance guarantee for the polynomial-time algorithm for solving instances in $\mathcal{C}_k^\pi$—in other words, we expect an inevitable *tradeoff between generality and performance*.

Assume our SAT algorithm for $\mathcal{C}_k^\pi$ runs in time $\mathcal{O}(n^k)$ on instances with n variables, then we have an example for a non-uniform polynomial-time algorithm, since the degree of the polynomial depends on k. On the other hand, a running time such as $\mathcal{O}(2^k n^3)$ establishes uniform polynomial time. For a non-uniform polynomial-time algorithm even relatively small values for k render classes $\mathcal{C}_k^\pi$ practically infeasible—take the above example of time complexity $\mathcal{O}(n^k)$ and consider an instance $F \in \mathcal{C}_{10}^\pi$ with $n = 1000$ variables. On the other hand, a uniform polynomial-time algorithm with running time such as $\mathcal{O}(2^k n^3)$ makes the satisfiability problem practically feasible for classes $\mathcal{C}_k^\pi$ as long as k remains small.

It is an intriguing research objective to design and study satisfiability parameters and to find out whether they admit uniform polynomial-time algorithms or not. Classical complexity theory does not provide the means and tools for this purpose, as the computational complexity of a problem is considered exclusively in terms of the *input size*; structural properties of instances are not represented. In the late 1980s, Rod Downey and Mike Fellows initiated the framework of *Parameterized Complexity* which resolves this shortcoming of classical theory. Their point of departure was the following observation: uniform polynomial-time algorithms exist for finding a vertex cover of size k in a graph, but apparently, no uniform polynomial-time algorithm exists for finding an independent set of size k (in both cases k is considered as the parameter). Downey, Fellows, and their collaborators have developed a rich theoretical framework for studying the computational complexity of parameterized problems. Over recent years, parameterized complexity has become an essential branch of algorithm design and analysis in both applied and theoretical areas of computer science; hundreds of research papers and several monographs have been published so far on the subject [DF99, FG06, Nie06, CFK$^+$13, DF13]. Parameterized complexity considers problem instances in a *two-dimensional* setting: the first dimension is the usual input size n, the second dimension is a non-negative integer k, the *parameter*. An algorithm that solves an instance in time $\mathcal{O}(f(k)n^c)$ is called a *fixed-parameter algorithm*; here f denotes an arbitrary computable function and c denotes a constant that is independent of n and k. Thus fixed-parameter algorithms are algorithms with a uniform polynomial-time complexity as considered in the above discussion. A parameterized problem is *fixed-parameter tractable* if a fixed-parameter algorithm can solve it. Once a problem has been found to be fixed-parameter tractable, one can try to find better and better algorithms, making the function f to grow slower, and obtaining a smaller constant c. In many cases, one can observe a trajectory of steadily improved running times for parameterized problems [Fel03]. Therefore we will mainly focus on the classification of satisfiability parameters on whether or not they admit fixed-parameter tractable satisfiability decision, and not on concrete running times.

Parameterized complexity also offers a *completeness theory* which is similar to the theory of NP-completeness in the classical setting. This completeness theory provides strong evidence that certain parameterized problems (such as the parameterized independent set problem as mentioned above) are *not* fixed-parameter

tractable. We will briefly discuss some fundamental notions of this completeness theory in Section 17.2.1. In this survey, however, we will mainly focus on positive results, describing key concepts that lead to satisfiability parameters that admit fixed-parameter algorithms. The presented negative results (i.e., hardness results) have merely the purpose of carving out territories that are very likely to be inaccessible to fixed-parameter algorithms.

The majority of combinatorial problems studied in the framework of parameterized complexity offers a "natural parameter," e.g., it is natural to parameterize the vertex cover problem by the size of the vertex cover. However, the satisfiability problem lacks a single obvious natural parameter—there are numerous possibilities for parameters. This variety, however, makes parameterized SAT a rich and exciting research area; one of its fundamental objectives is to identify satisfiability parameters that are as general as possible (i.e., for as many instances as possible one can expect that the parameter is small), and which are still accessible to fixed-parameter algorithms.

Although our main focus is satisfiability decision, we will come across several satisfiability parameters that even render the propositional model counting problem #SAT to be fixed-parameter tractable.

Next, in Section 17.2, we will provide some preliminaries: we will introduce basic notions of parameterized complexity, our terminology on CNF formulas and truth assignments, and graphs and hypergraphs associated with CNF formulas. In Section 17.3 we will introduce a general framework for parameterizing the satisfiability problem in terms satisfiability parameters, and will discuss parameterized optimization problems related to SAT. The next three sections are devoted to satisfiability parameters of different flavors: in Section 17.4 we will consider parameters based on backdoor sets relative to a polynomial-time base class; in Section 17.5 we will consider parameters that measure the "tree-likeness" of instances; in Section 17.6 we will consider further parameters including one that is based on graph matchings and one that is based on the community structure of formulas. We will conclude with final remarks in Section 17.7.

17.2. Preliminaries

17.2.1. Fixed-Parameter Algorithms

In this section, we provide a brief (and rather informal) review of some fundamental concepts of parameterized complexity. For an in-depth treatment of the subject, we refer the reader to other sources [DF99, FG06, Nie06, CFK$^+$13, DF13].

An instance of a parameterized problem is a pair (I, k) where I is the *main part* and k is the *parameter*; the latter is usually a non-negative integer. A parameterized problem is *fixed-parameter tractable* if a fixed-parameter algorithm can solve it, i.e., if instances (I, k) can be solved in time $\mathcal{O}(f(k)\|I\|^c)$ where f is a computable function, c is a constant, and $\|I\|$ denotes the size of I with respect to some reasonable encoding. FPT denotes the class of all fixed-parameter tractable decision problems.

Let us illustrate the idea of a fixed-parameter algorithm trough the *vertex cover* problem parameterized by the solution size. This is the best-studied prob-

lem in parameterized complexity with a long history of improvements [CKJ01]. Let us state the parameterized vertex cover problem:

> **VC**
> *Instance:* A graph $G = (V, E)$ and a non-negative integer k.
> *Parameter:* k.
> *Task:* Decide whether there is a subset $S \subseteq V$ of size at most k such that every edge of G has at least one of its incident vertices in S (S is a *vertex cover* of G).

Note that if we consider k not as a parameter but simply as a part of the input, then we get an NP-complete problem [GJ79]. A simple fixed-parameter algorithm for **VC** can be constructed as follows. Given an instance (G, k) of **VC**, we construct a binary search tree. The root of the tree is labeled with (G, k). We choose an arbitrary edge uv of G and observe that every vertex cover of G must contain u or v. Hence we can branch into these two cases. That is, we add two children to the root, labeled with $(G - u, k - 1)$ and $(G - v, k - 1)$, respectively (k gets decremented as we have spent one unit for taking u or v into the vertex cover). We recursively extend this branching. We stop a branch of the tree if we reach a node labeled with (G', k') such that either $k' = 0$ (we have used up the budget k) or G' has no edges (we have found a vertex cover of size $k - k'$). Note that in the second case we can find the vertex cover of size $k - k'$ by collecting the vertices that have been removed from G along the path from the root to the leaf. It is easy to see the outlined algorithm is correct and decides **VC** in time $\mathcal{O}(2^k n)$ for graphs with n vertices. Using the *$\mathcal{O}^*$-notation* [Woe03] which suppresses polynomial factors, we can state the running time of the above algorithm by the expression $\mathcal{O}^*(2^k)$.

The above algorithm for **VC** illustrates the method of *bounded search trees* for the design of fixed-parameter algorithms. *Kernelization* is another essential technique, which shrinks the size of the given problem instance by employing (polynomial-time) data reduction rules until the size is bounded by a function of the parameter k. The reduced instance is called a *problem kernel*. Once a problem kernel is obtained for a decidable problem, we know that the problem is fixed-parameter tractable, since the running time of any brute force algorithm depends on the parameter k only. The converse is also true: whenever a parameterized problem is fixed-parameter tractable, then the problem admits a polynomial-time kernelization [CCDF97]. Consider the **VC** problem again as an example. It is easy to see that a vertex v of degree greater than k must belong to every vertex cover of size at most k; hence if we have such a vertex v, we can reduce the instance (G, k) to $(G - v, k - 1)$. Assume that we are left with the instance (G', k') after we have applied the reduction rule as long as possible (if $k' < 0$, then we reject the instance). Observe that each vertex of G' can cover at most k edges. Hence, if G' has more than k^2 edges, we know that G has no vertex cover of size at most k. On the other hand, if G' has at most k^2 edges, we have a problem kernel that can be solved by brute force.

The current best worst-case time complexity for **VC** is due to Chen, Kanj, and Xia [CKX10]. The algorithm is based on more sophisticated kernelization rules and achieves a running time of $\mathcal{O}^*(1.2738^k)$. Further information on ker-

nelization can be found in Guo and Niedermeier's survey [GN07]. Gaspers and Szeider [GS14] gave several kernelization results that are specifically relevant for parameterized satisfiability.

Next, we turn our attention to fixed-parameter *in*tractability, to problems that are believed to be not fixed-parameter tractable. Consider for example the following parameterized independent set problem.

> **IS**
> *Instance:* A graph $G = (VE)$ and a non-negative integer k.
> *Parameter:* k.
> *Task:* Decide whether there is a subset $S \subseteq V$ of size at least k such that no edge of G joins two vertices in S (S is an *independent set* of G).

No fixed-parameter algorithm for this problem is known, and there is strong evidence to believe that no such algorithm exists [DF99]. For example, fixed-parameter tractability of **IS** would imply the existence of an $\mathcal{O}^*(2^{o(n)})$-time algorithm for the n-variable 3-SAT problem [FG06]. The assumption that the latter is not the case is known as the *Exponential Time Hypothesis (ETH)* [IPZ01]; see also Chapter 12 for more information on the ETH.

Thus, **VC** is fixed-parameter tractable, whereas **IS** is believed to be not. Note, however, that under classical polynomial-time many-to-one reductions, **VC** and **IS** are equivalent for trivial reasons: a graph with n vertices has a vertex cover of size k if and only if it has an independent set of size $k' = n - k$. Hence, to distinguish between fixed-parameter tractable and fixed-parameter intractable problems, one needs a notion of reduction that restricts the way of how parameters are mapped to each other. An *fpt-reduction* from a parameterized decision problem L to a parameterized decision problem L' is an algorithm that transforms an instance (I, k) of L into an instance $(I', g(k))$ of L' in time $\mathcal{O}(f(k)\|I\|^c)$ (f,g are arbitrary computable functions, c is an arbitrary constant), such that (I, k) is a yes-instance of L if and only if $(I', g(k))$ is a yes-instance of L'. It is easy to see that indeed, if L' is fixed-parameter tractable and there is an fpt-reduction from L to L', then L is fixed-parameter tractable as well. Note that the reduction from **VC** to **IS** as sketched above is not an fpt-reduction since $k' = n - k$ and so k' is not a function of k alone.

The class of problems that can be reduced to **IS** under fpt-reductions is denoted by W[1]. A problem is called W[1]-*hard* if **IS** (and so every problem in W[1]) can be reduced to it by an fpt-reduction. A problem is called W[1]-*complete* if it is W[1]-hard and belongs to W[1]. Thus, a problem is W[1]-complete if and only if it is equivalent to **IS** under fpt-reductions. A similar terminology applies to other parameterized complexity classes.

Consider the following parameterized hitting set problem (it is the basis for several hardness results that we will consider in the remainder of this chapter).

> **HS**
> *Instance:* A family $\mathcal{S}$ of finite sets $S_1, \ldots, S_m$ and a non-negative integer k.

Parameter: k.
Task: Decide if there is a subset $R \subseteq \bigcup_{i=1}^{m} S_i$ of size at most k such that $R \cap S_i \neq \emptyset$ for all $i = 1, \ldots, m$ (R is a *hitting set* of $\mathcal{S}$).

Observe that, indeed, a search tree algorithm as outlined above for **VC** does not yield fixed-parameter tractability for **HS**: since the size of the sets S_i is unbounded, a search tree algorithm would entail an unbounded branching factor. If, however, the size of the sets S_i is bounded by some constant q, then the problem (known as q-**HS**) becomes fixed-parameter tractable. The obvious search tree algorithm has time complexity $\mathcal{O}^*(q^k)$. For $q = 3$, Wahlström [Wah17] developed a fixed-parameter algorithm with running time $\mathcal{O}^*(2.076^k)$.

HS is W[1]-hard, but no fpt-reduction from **HS** to **IS** is known, and it is believed that such a reduction does not exist. In other words, **HS** appears to be harder than the problems in W[1]. The class of problems that can be reduced to **HS** under fpt-reductions is denoted by W[2]. In fact, W[1] and W[2] form the first two levels of an infinite chain of classes $\mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \mathrm{W}[3] \subseteq \cdots \subseteq \mathrm{W}[\mathrm{P}]$, the so-called "weft hierarchy." All inclusions are believed to be proper. There are several sources of theoretical evidence for assuming that the classes of the weft hierarchy are distinct from FPT: accumulating evidence [Ces06], evidence based on parameterized analogs of Cook's Theorem [DF99], and evidence obtained by proof complexity methods [DMS11].

We say that a parameterized problem is *XP-tractable* if instances (I, k) can be solved in time $\mathcal{O}(\|I\|^{f(k)})$ for a computable function f. The class XP consists of all XP-tractable parameterized decision problems. FPT $\neq$ XP is provably true [DF99, FG06]. Together with the classes of the weft hierarchy, we have the following chain of inclusions:

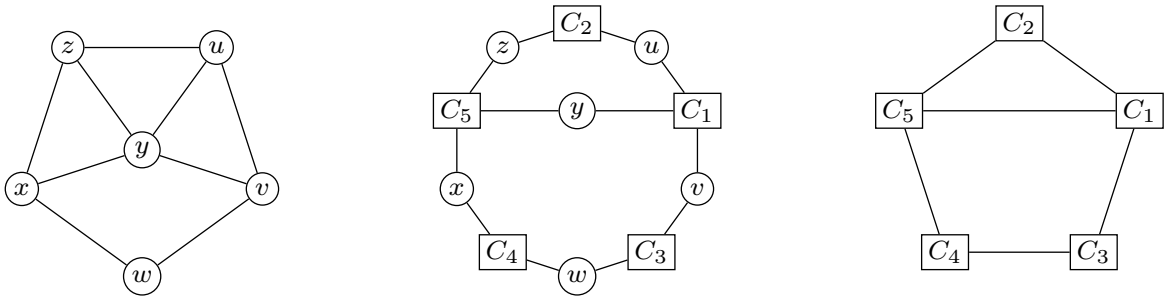
$$\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{W}[\mathrm{P}] \subseteq \mathrm{XP}.$$

17.2.2. CNF Formulas and Truth Assignments

Before discussing parameterizations of SAT, let us introduce some notation and basic notions related to SAT. We consider propositional formulas in conjunctive normal form (CNF), short *CNF formulas* or just *formulas*, represented as a finite set of *clauses*. A clause is a finite set of *literals*, and a literal is a negated or un-negated propositional *variable*. For a literal ℓ we denote by $\overline{\ell}$ the literal of opposite polarity, i.e., $\overline{x} = \neg x$ and $\overline{\neg x} = x$. We also write $x^1 = x$ and $x^0 = \neg x$. Similarly, for a set L of literals, we put $\overline{L} = \{\, \overline{\ell} : \ell \in L \,\}$. We say that two clauses C, D *overlap* if $C \cap D \neq \emptyset$, and we say that C and D *clash* if C and $\overline{D}$ overlap. For a clause C we denote by $\mathrm{var}(C)$ the set of variables that occur (negated or un-negated) in C; for a formula F we put $\mathrm{var}(F) = \bigcup_{C \in F} \mathrm{var}(C)$. We measure the *size* $\|F\|$ of a formula F by its *length* $\sum_{C \in F} |C|$.

CNF formulas F and F' are *isomorphic* if they differ only in the name of variables. That is, if $F = \{C_1, \ldots, C_m\}$, $F' = \{C'_1, \ldots, C'_m\}$, and there is a one-to-one mapping $f : \mathrm{var}(F) \to \mathrm{var}(F')$ such that $C'_i = \{\, (f(x))^\varepsilon : x^\varepsilon \in C_i, x \in \mathrm{var}(F), \varepsilon \in \{0,1\} \,\}$ holds for all $1 \leq i \leq m$.

CNF formulas F and F' are *renamings* of each other if there exists a set $X \subseteq \mathrm{var}(F)$ such that F' can be obtained from F by flipping the polarity



primal graph incidence graph dual graph

Figure 17.1. Graphs associated with the CNF formula $F = \{C_1, \ldots, C_5\}$ with $C_1 = \{u, \neg v, \neg y\}$, $C_2 = \{\neg u, z\}$, $C_3 = \{v, \neg w\}$, $C_4 = \{w, \neg x\}$, $C_5 = \{x, y, \neg z\}$; the primal graph $G(F)$, the incidence graph $G^*(F)$, and the dual graph $G^d(F)$.

of all literals $\ell \in X \cup \overline{X}$. That is, if $F = \{C_1, \ldots, C_m\}$, $F' = \{C'_1, \ldots, C'_m\}$, and $C'_i = \{\, \ell : \ell \in C_i \setminus (X \cup \overline{X}) \,\} \cup \{\, \overline{\ell} : \ell \in C_i \cap (X \cup \overline{X}) \,\}$.

A *truth assignment* is a mapping $\tau : X \to \{0, 1\}$ defined on some set X of variables. If $X = \{x\}$ we denote τ simply by "$x = 1$" or "$x = 0$". We extend τ to literals by setting $\tau(\neg x) = 1 - \tau(x)$ for $x \in X$. $F[\tau]$ denotes the formula obtained from F by removing all clauses that contain a literal ℓ with $\tau(\ell) = 1$ and by removing from the remaining clauses all literals ℓ' with $\tau(\ell') = 0$. $F[\tau]$ is the *restriction* of F to τ. Note that $\mathrm{var}(F[\tau]) \cap X = \emptyset$ holds for every truth assignment $\tau : X \to \{0, 1\}$ and every formula F. A truth assignment $\tau : X \to \{0, 1\}$ *satisfies* a clause C if $\tau(\ell) = 1$ for at least one literal $\ell \in C$; τ satisfies a formula F if it satisfies all clauses of F. Note that τ satisfies F if and only if $F[\tau] = \emptyset$. A formula F is *satisfiable* if there exists a truth assignment that satisfies F; otherwise, F is *unsatisfiable*. A truth assignment $\tau : \mathrm{var}(F) \to \{0, 1\}$ is called *total* for the formula F. A satisfying total truth assignment of F is called a *model* of F. We denote the number of models of a formula F by $\#(F)$. Two formulas are *equisatisfiable* if either both are satisfiable or both are unsatisfiable. **SAT** is the problem of deciding whether a given formula is satisfiable. **#SAT** is the problem of determining the number of models of a given formula.

Let $x \in \mathrm{var}(F)$ and $\varepsilon \in \{0, 1\}$. If $\{x^\varepsilon\} \in F$, then F and $F[x = \varepsilon]$ are equisatisfiable; $F[x = \varepsilon]$ is said to be obtained from F by *unit propagation*. If some clause of F contains x^ε but none contains $x^{1-\varepsilon}$, then x^ε is called a *pure literal* of F. If x^ε is a pure literal of F, then obviously F and $F[x = \varepsilon]$ are equisatisfiable. In that case, we say that $F[x = \varepsilon]$ is obtained from F by *pure literal elimination*.

17.2.3. Graphs and Hypergraphs Associated with CNF Formulas

In this section, we will discuss several graphs and hypergraphs that can be used to represent the structure of a CNF formula.

Perhaps the most prominent graph representation of a CNF formula F is the *primal graph* $G(F)$. The vertices of $G(F)$ are the variables of F; two variables x, y are joined by an edge if they occur in the same clause, that is, if $x, y \in \mathrm{var}(C)$

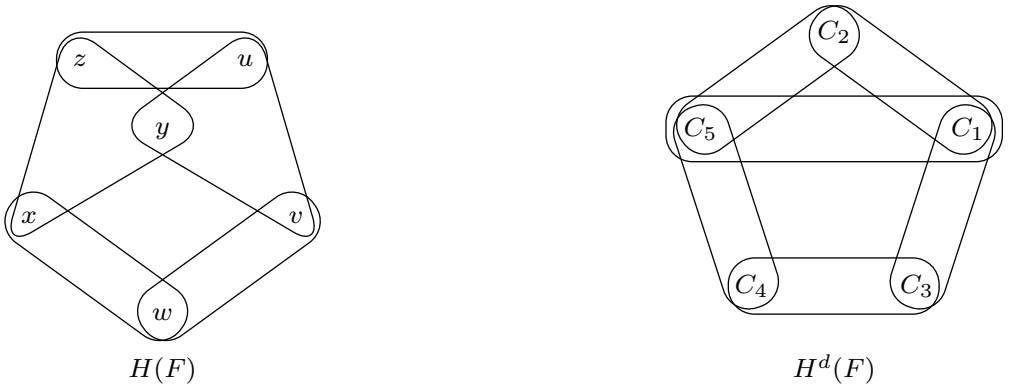


Figure 17.2. The hypergraph $H(F)$ and the dual hypergraph $H^d(F)$ associated with the CNF formula F of Figure 17.1.

for some $C \in F$. Another important graph is the *incidence graph* $G^*(F)$. The vertices of $G^*(F)$ are the variables and clauses of F; a variable x and a clause C are joined by an edge if $x \in \mathrm{var}(C)$. If we orient the edges of the incidence graph according to the polarity of the literals, then we obtain the *directed incidence graph* $D^*(F)$: an edge is directed from C to x if $x \in C$, and directed from x to C if $\neg x \in C$. In analogy to the primal graph, one can also define the *dual graph* $G^d(F)$. The vertices of $G^d(F)$ are the clauses of F; two clauses C_1, C_2 are joined by an edge if there is a variable occurring in both of them, that is, if $x \in \mathrm{var}(C_1) \cap \mathrm{var}(C_2)$ for some $x \in \mathrm{var}(F)$. We also consider the following two graphs, the *conflict graph* $G^{\lightning}(F)$ and the *consensus graph* $G^{\sun}(F)$, both having as vertices the clauses of F, where in $G^{\lightning}(F)$ two clauses C_1, C_2 are joined by an edge if they clash ($C_1 \cap \overline{C_2} \neq \emptyset$), and in $G^{\sun}(F)$ two clauses C_1, C_2 are joined by an edge if they do not clash ($C_1 \cap \overline{C_2} = \emptyset$).

Figure 17.1 shows the primal graph, the incidence graph, and the dual graph of a CNF formula.

Hypergraphs generalize graphs in the sense that each edge may connect more than just two vertices, i.e., the edges (called *hyperedges*) of a hypergraph are non-empty sets of vertices. We associate to each CNF formula F its *hypergraph* $H(F)$. The vertices of $H(F)$ are the variables of F and for each $C \in F$ the set $\mathrm{var}(C)$ represents a hyperedge of $H(F)$. The *dual hypergraph* $H^d(F)$ is defined symmetrically: the vertices of $H^d(F)$ are the clauses of F; for each variable x, the set of clauses C with $x \in \mathrm{var}(C)$ forms a hyperedge. See Figure 17.2 for examples.

17.3. Parameterized SAT

In the first part of this section we will develop the framework for parameterized SAT decision and model counting, where the parameter represents structural information of the instances. This framework will be used throughout the remainder of this chapter. In the second part of this section we will discuss parameterized optimization problems that are related to satisfiability.

17.3.1. Satisfiability Parameters

A *satisfiability parameter* is a computable function π that assigns to every formula F a non-negative integer $\pi(F)$. We assume that $\pi(F) = \pi(F')$ if two formulas F, F' are isomorphic (see Section 17.2.2), i.e., π is invariant with respect to isomorphisms.

We are interested in satisfiability parameters that allow fixed-parameter tractability of satisfiability decision for instances F with respect to the parameter $k = \pi(F)$. Accordingly, for a satisfiability parameter π we consider the following generic parameterized problem.

SAT(π)
Instance: A formula F and a non-negative integer k.
Parameter: k.
Task: Decide whether F is satisfiable, or determine that $\pi(F) > k$.

This way of parameterizing the satisfiability problem was suggested by Szeider [Sze04b]. It avoids the requirement of exactly recognizing formulas F with $\pi(F) \leq k$ and is related to the concept of a "robust algorithm" introduced by Spinrad [Spi03]. Marx and Schlotter [MS10, MS11] used the predicate "*permissive*" to describe parameterized problems of this form, in contrast to "*strict*" problems, which entail the exact determination of the parameter value. We can, as well, formulate model counting as a permissive problem:

#SAT(π)
Instance: A formula F and a non-negative integer k.
Parameter: k.
Task: Compute $\#(F)$ or determine that $\pi(F) > k$.

If we solve a parameterized satisfiability problem in the strict sense, we have to solve the parameterized *verification problem*:

VER(π)
Instance: A formula F and a non-negative integer k.
Parameter: k.
Task: Decide whether $\pi(F) \leq k$.

We assume that an affirmative solution to the verification problem is backed up with a certain *witness* for "$\pi(F) \leq k$" in form of an auxiliary structure, which can subsequently be used for deciding the satisfiability of F or counting the models of F.

In some cases, **VER**(π) is W[1]-hard or just not known to be fixed-parameter tractable, but we can still use π for fixed-parameter tractability of **SAT**(π) or **#SAT**(π) if **VER**(π) is *fixed-parameter approximable*. That is, if there is a computable function g and a fixed-parameter algorithm that, given F and k, either outputs a witness for $\pi(F) \leq g(k)$, or correctly decides that $\pi(F) > k$. We can use the witness for $\pi(F) \leq g(k)$ as an input for a parameterized algorithm that decides the satisfiability of F or computes $\#(F)$, albeit the running time is worse than if we had a witness for $\pi(F) \leq k$.

Usually it is very difficult to establish parameterized hardness results for the permissive problems **SAT**(π) and **#SAT**(π), or to provide theoretical evidence

that $\mathbf{VER}(\pi)$ is not fixed-parameter approximable. We have only very few examples for such hardness results (see, e.g., Theorems 17.4.5, 17.4.8, and 17.4.5). Although a parameterized hardness result for $\mathbf{VER}(\pi)$ does not rule out the fixed-parameter tractability of $\mathbf{SAT}(\pi)$, it still provides the valuable insight that the fixed-parameter tractability of $\mathbf{SAT}(\pi)$ in the strict sense is unlikely.

The notion of *dominance* allows us to compare two satisfiability parameters π and π' with respect to their generality. We say that π *dominates* π' if there exists a computable function f such that for every formula F we have

$$\pi(F) \leq f(\pi'(F)).$$

Furthermore, π *strictly dominates* π' if π dominates π' but not vice versa. Finally, π and π' are *domination incomparable* if neither dominates the other, and *domination equivalent* if they dominate each other. If π strictly dominates π' we also say that π is *more general* than π'. Dominance, strict dominance, and domination equivalence are transitive relations between satisfiability parameters. Furthermore, since strict dominance is antisymmetric, it gives rise to a partial ordering of satisfiability parameters. The next result follows directly from the definitions.

Lemma 17.3.1. *If π dominates π', then there is an fpt-reduction from $\mathbf{SAT}(\pi')$ to $\mathbf{SAT}(\pi)$.*

Thus, if $\mathbf{SAT}(\pi)$ is fixed-parameter tractable and π dominates π', then $\mathbf{SAT}(\pi')$ is also fixed-parameter tractable. It is an important goal to find satisfiability parameters π that are as general as possible and for which both problems $\mathbf{SAT}(\pi)$ and $\mathbf{VER}(\pi)$ are fixed-parameter tractable. We conclude this section with three trivial examples of satisfiability parameters.

Example 17.3.1. Let $\mathbf{n}(F)$ denote the number of variables of a formula F. The verification problem $\mathbf{VER}(\mathbf{n})$ is trivial. The obvious algorithm that considers all possible truth assignments of F runs in time $\mathcal{O}^*(2^{\mathbf{n}(I)})$ and is, therefore, a fixed-parameter algorithm with respect to $\mathbf{n}$. Hence $\mathbf{SAT}(\mathbf{n})$ is fixed-parameter tractable. □

A satisfiability parameter π becomes an interesting one if every class $\mathcal{C}_k^\pi$ contains formulas with an arbitrarily large number of variables; i.e., if π is more general than the satisfiability parameter $\mathbf{n}$ considered in the example above.

Example 17.3.2. Let $\mathbf{ml}(F)$ denote the *maximum length* of clauses in a SAT instance F (with $\mathbf{ml}(F) = 0$ if $F = \emptyset$). From the NP-completeness of 3SAT if follows that $\mathbf{SAT}(\mathbf{ml})$ is not fixed-parameter tractable unless P = NP. So $\mathbf{SAT}(\mathbf{ml})$ is probably not even in XP. □

Example 17.3.3. Let $\mathcal{A}$ be a deterministic polynomial-time algorithm that applies polynomial-time simplification and propagation rules to a formula without changing its satisfiability. Say, the algorithm applies *unit propagation* and *pure literal elimination* as long as possible (see Section 17.2.2 above), plus possibly some further rules. For more powerful preprocessing rules see, e.g., the work of Bacchus and Winter [BW04]. Let $\mathcal{A}(I)$ denote the instance obtained from I by

applying algorithm $\mathcal{A}$, and let $\mathbf{n}_{\mathcal{A}}(I)$ denote the number of variables of $\mathcal{A}(I)$ (if $\mathcal{A}(I)$ depends on a particular ordering of variables and clauses, let $\mathbf{n}_{\mathcal{A}}(I)$ denote the largest number over all such orderings).

It is easy to see that the problem $\mathbf{SAT}(\mathbf{n}_{\mathcal{A}})$ is fixed-parameter tractable since after the polynomial-time preprocessing we are left with an instance $\mathcal{A}(I)$ whose number of variables is bounded in terms of the parameter k, and therefore any brute-force algorithm applied to $\mathcal{A}(I)$ is a fixed-parameter algorithm. In other words, $\mathbf{SAT}(\mathbf{n}_{\mathcal{A}})$ is fixed-parameter tractable since the preprocessing provides a kernelization. $\mathbf{VER}(\mathbf{n}_{\mathcal{A}})$ is easy, as we just need to count the number of variables left after applying the polynomial-time preprocessing algorithm $\mathcal{A}$. Clearly $\mathbf{n}_{\mathcal{A}}$ is more general than $\mathbf{n}$ (Example 17.3.1) since one can easily find formulas where, say, unit propagation eliminates an arbitrarily large number of variables. □

17.3.2. Optimization Problems

Max-SAT is the optimization version of the satisfiability problem, where, given a CNF formula F and an integer k, one asks whether there is a truth assignment that satisfies at least k clauses of F. In the classical setting, Max-SAT is NP-complete even if all clauses contain at most two literals (Max-2-SAT). What happens if we consider k as the parameter?

Under this parameterization, Max-SAT is easily seen to be fixed-parameter tractable (we roughly follow [MR99]). Let (F, k) be an instance of Max-SAT. For a truth assignment τ we write $s(\tau)$ for the number of clauses of F that are satisfied by τ. Moreover, let τ^* denote the extension of τ that includes all variable assignments obtained by (iterated and exhaustive) application of pure literal elimination. For example, if $F = \{\{w, \overline{x}\}, \{x, y, \overline{z}\}, \{\overline{y}, z\}, \{\overline{w}, \overline{z}\}\}$ and $\tau = \{(w, 1)\}$, then $\tau^* = \{(w, 1), (x, 1), (y, 0), (z, 0)\}$. We construct a binary search tree T whose nodes are truth assignments. We start with the empty assignment as the root and extend the tree downwards as follows. Consider a node τ of T. If $s(\tau^*) \geq k$ or if $s(\tau^*) < k$ and $F[\tau^*] = \{\emptyset\}$, then we do not add any children to τ; in the first case we label τ as "success leaf," in the second case as "failure leaf." Otherwise, if $s(\tau^*) < k$ and $F[\tau^*] \neq \{\emptyset\}$, we pick a variable $x \in F[\tau^*]$ and we add below τ the children $\tau_0 = \tau \cup \{(x, 0)\}$ and $\tau_1 = \tau \cup \{(x, 1)\}$. Note that in this case both $s(\tau_0)$ and $s(\tau_1)$ are strictly greater than $s(\tau)$. It is easy to see that there exists a total truth assignment τ of F that satisfies at least k clauses if and only if T has a success leaf (for the only-if direction note that τ defines a path from the root of T to a success leaf). Since at each branching step the number of satisfied clauses increases, it follows that T is of depth at most k and so has at most 2^k leaves. Hence the search algorithm runs in time $\mathcal{O}^*(2^k)$ which renders Max-SAT fixed-parameter tractable for parameter k. By sophisticated case distinctions, one can make the algorithm significantly faster. The currently fastest algorithm is runs in time $\mathcal{O}^*(1.3248^k)$ [CXW17].

Note that one can always satisfy at least half of the clauses of a CNF formula (the all-true or the all-false assignment will do). Thus, a more challenging parameter for Max-SAT is the number $k - |F|/2$ (this constitutes a parameterization "above the guaranteed value" $|F|/2$). Indeed, by a result of Mahajan and Raman [MR99], Max-SAT is fixed-parameter tractable also under this more general setting. For an r-CNF input formula, where r is an arbitrary constant, one

can always satisfy at least a $1 - 2^{-r}$ fraction of its clauses, and such an assignment can be found in polynomial time using Johnson's Algorithm [Joh73]. Alon et al. [AGK+11] showed that Max-SAT for r-CNF formulas is fixed-parameter tractable parameterized above this bound, answering a question posed by Mahajan et al. [MRS06].

One can consider an even more challenging approach, taking the *dual parameter* $k' = |F| - k$; that is, to parameterize by the number of clauses that remain unsatisfied. It is easy to see that for every *constant* k' the problem is NP-complete in general and polynomial-time solvable for 2CNF formulas (Max-2-SAT). It follows from recent results of Razgon and O'Sullivan [RO08] that Max-2-SAT is fixed-parameter tractable for the dual parameter k'. We will return to this problem again in Section 17.4.1.

Apart from Max-SAT there are also other interesting optimization versions of satisfiability. For example, *Bounded-CNF-SAT* asks whether a CNF formula can be satisfied by setting at most k variables to true. With parameter k, Bounded-CNF-SAT is W[2]-complete; Bounded-3-CNF-SAT, however, is easily seen to be fixed-parameter tractable [DMS11]. A similar problem, *Weighted-CNF-SAT*, asks for a satisfying assignment that sets *exactly* k variables to true. Weighted-CNF-SAT is W[2]-complete; Weighted-c-CNF-SAT is W[1]-complete for every constant $c \geq 2$ [DF99].

17.4. Backdoor Sets

As outlined in the introduction, every satisfiability parameter π gives rise to the hierarchy of classes

$$\mathcal{C}_0^\pi \subseteq \mathcal{C}_1^\pi \subseteq \mathcal{C}_2^\pi \subseteq \cdots$$

where class $\mathcal{C}_k^\pi$ contains all CNF formulas F with $\pi(F) \leq k$. We call this hierarchy the *π-hierarchy*, and we refer to the class at the lowest level of the hierarchy as the *base class*. The following are necessary conditions for a class $\mathcal{C}$ of CNF formulas under which it could possibly act as the base class for the π-hierarchy of some satisfiability parameter π such that both $\mathbf{SAT}(\pi)$ and $\mathbf{VER}(\pi)$ are fixed-parameter tractable:

1. $\mathcal{C}$ is closed under isomorphism;
2. membership in $\mathcal{C}$ can be decided in polynomial time;
3. satisfiability of elements of $\mathcal{C}$ can be decided in polynomial time.

Some authors also require that a base class is *self-reducible*, that is, if $F \in \mathcal{C}$ then $F[x = 0], F[x = 1] \in \mathcal{C}$ for all $x \in \mathrm{var}(F)$. Most natural base classes are self-reducible.

Next, we will see how one can define a π-hierarchy starting at an arbitrary base class $\mathcal{C}$ utilizing the notion of "backdoor sets" which was introduced by Williams, Gomes, and Selman [WGS03] for analyzing the behavior of SAT algorithms. Actually, with different terminology and context, backdoor sets have already been studied by Crama, Elkin, and Hammer [CEH97]. Consider a CNF formula F and a set $B \subseteq \mathrm{var}(F)$ of variables. B is called a *strong $\mathcal{C}$-backdoor set* of F if for every truth assignment $\tau : B \to \{0, 1\}$ the restriction $F[\tau]$ belongs

to the base class $\mathcal{C}$. We also introduce the notion of *weak* backdoor sets. A set $B \subseteq \mathrm{var}(F)$ is called a *weak $\mathcal{C}$-backdoor set* of F if there exists truth assignment $\tau : B \to \{0,1\}$ such that the restriction $F[\tau]$ *is satisfiable* and belongs to the base class $\mathcal{C}$. We denote the size of a smallest strong $\mathcal{C}$-backdoor set of F by $\mathbf{b}_\mathcal{C}(F)$ and the size of a smallest weak a $\mathcal{C}$-backdoor set of F by $\mathbf{wb}_\mathcal{C}(F)$.

Example 17.4.1. Consider the base class HORN of Horn formulas (an instance is Horn if each of its clauses contains at most one un-negated variable) and consider the formula $F = \big\{\{u,v,w\},\ \{\overline{u},x,\overline{y}\},\ \{u,\overline{v},\overline{x},y\},\ \{v,y,\overline{z}\},\ \{u,v,\overline{w},z\}\big\}$. The set $B = \{u,v\}$, is a strong HORN-backdoor set since $F[\tau] \in$ HORN for all four truth assignments $\tau : B \to \{0,1\}$. □

Note that F is satisfiable if and only if at least one of the restrictions $F[\tau]$, $\tau : B \to \{0,1\}$, is satisfiable. Thus, if we know a strong $\mathcal{C}$-backdoor set B of F, we can decide the satisfiability of F by deciding the satisfiability of at most $2^{|B|}$ polynomial-time solvable formulas that belong to $\mathcal{C}$ (this is a $\mathcal{O}^*(2^k)$ fixed-parameter algorithm with respect to the parameter $k = |B|$). Of course we can find a $\mathcal{C}$-backdoor set of size at most k (or decide that it does not exist) by trying all subsets $B \subseteq \mathrm{var}(F)$ with $|B| \leq k$, and checking whether all $F[\tau]$, $\tau : B \to \{0,1\}$, belong to $\mathcal{C}$; consequently $\mathbf{VER}(\mathbf{b}_\mathcal{C}) \in$ XP. However, as we shall see in the following section, $\mathbf{VER}(\mathbf{b}_\mathcal{C})$ can or cannot be fixed-parameter tractable, depending on the base class $\mathcal{C}$.

For an algorithm that provides fixed-parameter tractability of $\mathbf{VER}(\mathbf{b}_\mathcal{C})$, we will always assume that it provides, as a witness for $\mathbf{b}_\mathcal{C}(C) \leq k$, a strong $\mathcal{C}$-backdoor set of F size of size $\leq k$. Hence the fixed-parameter tractability of $\mathbf{VER}(\mathbf{b}_\mathcal{C})$ implies the fixed-parameter tractability of $\mathbf{SAT}(\mathbf{b}_\mathcal{C})$.

As mentioned above, a strong $\mathcal{C}$-backdoor set of F of size k reduces the satisfiability of F to the satisfiability of at most 2^k instances in $\mathcal{C}$. The notions of *backdoor trees* and *backdoor DNFs* [SS08, OSS21] make this reduction explicit. This allows a refined worst-case estimation of the number of instances in $\mathcal{C}$ that need to be checked, which can be exponentially smaller than 2^k.

17.4.1. Horn, 2CNF, and Generalizations

HORN and 2CNF are two important base classes for which the detection of strong backdoor sets is fixed-parameter tractable.

Theorem 17.4.1 ([NRS04])**.** *For $\mathcal{C} \in \{$HORN, 2CNF$\}$ the problems $\mathbf{SAT}(\mathbf{b}_\mathcal{C})$ and $\mathbf{VER}(\mathbf{b}_\mathcal{C})$ are fixed-parameter tractable.*

The algorithms of Nishimura et al. rely on the concept of *variable deletion*. For explaining this, it is convenient to consider the following variant of backdoor sets: A set $B \subseteq \mathrm{var}(F)$ is called a *deletion $\mathcal{C}$-backdoor set* of F if $F - B$ belongs to $\mathcal{C}$. Here $F - B$ denotes the CNF formula $\{\, C \setminus (B \cup \overline{B}) : C \in F \,\}$, i.e., the formula obtained from F by removing from the clauses all literals of the form ℓ or $\overline{\ell}$ for $\ell \in B$. Let $\mathbf{db}_\mathcal{C}(F)$ denote the size of a smallest *deletion* $\mathcal{C}$-backdoor set of F. For many important base classes $\mathcal{C}$, deletion $\mathcal{C}$-backdoor sets are also strong $\mathcal{C}$-backdoor sets. In particular, this is the case if the base class is *clause induced*, i.e., if whenever F belongs to $\mathcal{C}$, all subsets of F belong to $\mathcal{C}$ as well. If $\mathcal{C}$ is clause induced, then $F[\tau] \subseteq F - B$ holds for every $\tau : B \to \{0,1\}$.

Lemma 17.4.2. *Let $\mathcal{C}$ be a clause-induced base class and let F be an arbitrary formula. Then every deletion $\mathcal{C}$-backdoor set of F is also a strong $\mathcal{C}$-backdoor set of F.*

For example, the base classes HORN and 2CNF are clause induced. For these two base classes, even the converse direction of Lemma 17.4.2 holds.

Lemma 17.4.3 ([CEH97, NRS04])**.** *Let $\mathcal{C} \in \{\text{HORN}, \text{2CNF}\}$ and let F be an arbitrary formula. Then the strong $\mathcal{C}$-backdoor sets of F are exactly the deletion $\mathcal{C}$-backdoor sets of F.*

Example 17.4.2. Consider the formula F of Example 17.4.1 and the strong HORN-backdoor set $B = \{u, v\}$ of F (note that B is also a strong 2CNF-backdoor set of F). Indeed, $F - B = \big\{\{w\},\ \{x, \overline{y}\},\ \{\overline{x}, y\},\ \{y, \overline{z}\},\ \{\overline{w}, z\}\big\}$ is a Horn formula. □

Nishimura et al. describe a fixed-parameter algorithm for the detection of strong HORN-backdoor sets. Their algorithm is based on bounded search trees similarly to the vertex cover algorithm described above. In fact, we can directly use a vertex cover algorithm. To this end, we associate with a formula F the *positive primal graph* G. The vertices of G are the variables of F, and two variables x, y are joined by an edge if and only if $x, y \in C$ for some clause C of F (negative occurrences of variables are ignored). Clearly, the positive primal graph can be constructed in time polynomial in the size of F. Now it is easy to see that for sets $B \subseteq \text{var}(F)$ the following properties are equivalent:

1. B is a strong HORN-backdoor set of F;
2. B is a deletion HORN-backdoor set of F;
3. B is a vertex cover of G.

Thus, any vertex cover algorithm, such as the $\mathcal{O}^*(1.2738^k)$ algorithm by Chen et al. [CKX10] mentioned above, can be used to find a strong HORN-backdoor set.

For the detection of strong 2CNF-backdoor sets one can apply a similar approach. Given a CNF formula F and a positive integer k, we want to determine whether F has a strong 2CNF-backdoor set of size at most k. Let $\mathcal{S}$ be the set of all size-3 subsets S of $\text{var}(F)$ such that $S \subseteq \text{var}(C)$ for some clause C of F. Evidently, S can be constructed in polynomial time. Observe that a set $B \subseteq \text{var}(F)$ is a hitting set of $\mathcal{S}$ if and only if B is a deletion 2CNF-backdoor set of F. By Lemma 17.4.3, the latter is the case if and only if B is a strong 2CNF-backdoor set of F. Thus, Wahlström's algorithm for **3-HS** [Wah17] solves $\mathbf{VER}(\mathbf{b}_{\text{2CNF}})$ in time $\mathcal{O}^*(2.076^k)$.

A generalization of backdoor sets, in particular HORN- and 2CNF-backdoor sets, to quantified Boolean formulas has been proposed by taking the variable dependencies caused by the quantifications into account [SS09].

The classes HORN and 2CNF are two of the five classes of CNF formulas that can be identified with tractable satisfiability problems considered by Schaefer in his seminal work on generalized satisfiability problems [Sch78]. The remaining three are the classes HORN^- of *anti-Horn* formulas (each clause contains at most one negative literal), and for $\varepsilon \in \{0, 1\}$ the classes ε-VAL of *ε-valid* formulas. A CNF-formula F is *ε-valid* if each nonempty clause $\emptyset \neq C \in F$ contains at

least one literal x^ε with $x \in var(C)$. The affine Boolean formulas considered by Schaefer do not correspond naturally to a class of CNF formulas; hence we do not consider them here.

For a given formula F we can compute in polynomial time a smallest deletion ε-VAL-backdoor set (which is also a smallest strong ε-VAL-backdoor set) by taking the union of $\mathrm{var}(C)$ over all $C \in F$ with $C \subseteq \{ x^{1-\varepsilon} : x \in \mathrm{var}(C) \}$. We put

$$\mathsf{Schaefer} = \{\textsc{Horn}, \textsc{Horn}^-, \mathrm{2CNF}, \text{0-}\textsc{Val}, \text{1-}\textsc{Val}\}$$

and summarize the discussed results as follows.

Theorem 17.4.4. *The problem* $\mathbf{SAT}(\mathbf{b}_\mathcal{C})$ *is fixed-parameter tractable for all* $\mathcal{C} \in \mathsf{Schaefer}$.

The detection of weak $\mathcal{C}$-backdoor sets, however, is W[2] hard for several base classes, including the classes in Schaefer. Gaspers and Szeider gave a generic reduction [GS12b, Proposition 1] that can be instantiated for various base classes $\mathcal{C}$. For the classes $\mathcal{C} \in \mathsf{Schaefer}$ hardness results even for the permissive problems $\mathbf{SAT}(\mathbf{wb}_\mathcal{C})$ are known:

Theorem 17.4.5 ([GS12b])**.** $\mathbf{SAT}(\mathbf{wb}_\mathcal{C})$ *is* W[1]*-hard for all* $\mathcal{C} \in \mathsf{Schaefer}$

A significant improvement over HORN as the base class for strong backdoor sets is the consideration of the class UP of CNF formulas that can be decided by unit propagation. That is, a CNF formula F belongs to UP if and only if after repeated application of unit propagation one is either left with the empty formula (i.e., F is satisfiable) or with a formula that contains the empty clause (i.e., F is unsatisfiable). Unfortunately, $\mathbf{VER}(\mathbf{b}_{\mathrm{UP}})$ turns out to be complete for the class W[P]. This holds also true if one considers the base class PL of CNF formulas decidable by pure literal elimination, and by the base class UP + PL of CNF formulas decidable by a combination of unit propagation and pure literal elimination. Thus UP + PL contains exactly those formulas that can be decided by the polynomial-time "subsolver" of the basic DPLL procedure [WGS03].

The following result provides strong evidence that the detection of strong backdoor sets with respect to the base classes PL, UP, and UP + PL is not fixed-parameter tractable. Let us write

$$\mathsf{Subsolver} = \{\mathrm{PL}, \mathrm{UP}, \mathrm{UP} + \mathrm{PL}\}.$$

Theorem 17.4.6 ([Sze05])**.** *For* $\mathcal{C} \in \mathsf{Subsolver}$*, the problem* $\mathbf{VER}(\mathbf{b}_\mathcal{C})$ *is* W[P]*-complete.*

Given this result, it appears to be very unlikely that one can find a size-k strong backdoor set with respect to the base class of formulas decidable by DPLL subsolvers significantly faster than by trying all sets of variables of size k. Also, the consideration of deletion backdoor sets does not offer an opportunity for overcoming this limitation: the classes UP, PL, and UP + PL are not clause induced—indeed, not every deletion backdoor set is a strong backdoor set with respect to these classes.

However, the class RHORN of *renamable Horn* formulas is an interesting base class that is clause induced. A formula is renamable Horn if some renaming

of it is Horn. It is well known that recognition and satisfiability of renamable Horn formulas is feasible in polynomial time [Lew78]. A renamable Horn formula is unsatisfiable if and only if we can derive the empty clause from it by unit propagation; also, whenever we can derive from a formula the empty clause by means of unit resolution, then some unsatisfiable subset of the formula is renamable Horn [KBL99]. Thus RHORN lies in a certain sense half way between UP and HORN. Since RHORN is clause induced, both strong and deletion backdoor sets are of relevance. In contrast to HORN, not every strong RHORN-backdoor set is a deletion RHORN-backdoor set. Indeed, $\mathbf{b}_{\mathrm{RHORN}}$ is a more general satisfiability parameter than $\mathbf{db}_{\mathrm{RHORN}}$ as can be seen from Lemma 17.4.2 and the following example.

Example 17.4.3. For $1 \leq i \leq n$, let $F_i = \big\{\{x_i, y_i, z\},\ \{x_i, \overline{y_i}, \overline{z}\},\ \{\overline{x_i}, y_i\},\ \{\overline{x_i}, \overline{y_i}\}\big\}$, and consider $F = \bigcup_{i=1}^{n} F_i$. Evidently $\{z\}$ is a strong RHORN-backdoor set of F, since each proper subset of $\big\{\{x_i, y_i\}, \{x_i, \overline{y_i}\}, \{\overline{x_i}, y_i\}, \{\overline{x_i}, \overline{y_i}\}\big\}$, $1 \leq i \leq n$, is renamable Horn. However, every deletion RHORN-backdoor set of F must contain at least one variable x_i or y_i for all $1 \leq i \leq n$. Hence $\mathbf{b}_{\mathrm{RHORN}}(F) \leq 1$ and $\mathbf{db}_{\mathrm{RHORN}}(F) \geq n$, which shows that $\mathbf{b}_{\mathrm{RHORN}}$ is more general than $\mathbf{db}_{\mathrm{RHORN}}$. □

The detection of strong RHORN-backdoor sets is W[1]-hard [GS12b], but the detection of deletion RHORN-backdoor sets is fixed-parameter tractable:

Theorem 17.4.7 ([RO08, GS12b])**.** *The problems* $\mathbf{VER}(\mathbf{db}_{\mathrm{RHORN}})$ *and* $\mathbf{SAT}(\mathbf{db}_{\mathrm{RHORN}})$ *are fixed-parameter tractable.*

Even the corresponding permissive problem is hard as well.

Theorem 17.4.8 ([GS12b])**.** $\mathbf{SAT}(\mathbf{wb}_{\mathrm{RHORN}})$ *is* W[1]*-hard.*

Boros et al. [BCH90] introduced an interesting class of CNF formulas, later called QHORN [BHS94], with favorable algorithmic properties: both recognition as well as deciding satisfiability of QHORN formulas can be performed in linear time. The class QHORN properly contains the fundamental classes RHORN and 2CNF:

$$\mathrm{HORN} \subsetneq \mathrm{RHORN} \subsetneq \mathrm{QHORN} \supsetneq \mathrm{2CNF}.$$

A CNF formula F is in QHORN if there is a *certifying function* $\beta : \mathrm{var}(F) \cup \overline{\mathrm{var}(F)} \to \{0, \frac{1}{2}, 1\}$ with $\beta(x) = 1 - \beta(\bar{x})$ for every $x \in \mathrm{var}(F)$ such that $\sum_{l \in C} \beta(l) \leq 1$ for every clause C of F.

As QHORN generalizes RHORN, it is not surprising, that the problems $\mathbf{VER}(\mathbf{b}_{\mathrm{QHORN}})$ and $\mathbf{VER}(\mathbf{wb}_{\mathrm{QHORN}})$ are W[2]-hard as well. However, Gaspers et al. [GOR$^+$16] showed that the detection of deletion QHORN-backdoor sets is fixed-parameter approximable, which implies the fixed-parameter tractability of $\mathbf{SAT}(\mathbf{db}_{\mathrm{QHORN}})$.

Theorem 17.4.9 ([GOR$^+$16])**.** *The problem* $\mathbf{SAT}(\mathbf{db}_{\mathrm{QHORN}})$ *is fixed-parameter tractable.*

This result was later improved by Ramanujan and Saurabh [RS17] who developed a general algorithmic framework for certain skew symmetric cut problems in graphs, and applied their method to show fixed-parameter tractability of $\mathbf{VER}(\mathbf{db}_{\mathrm{QHORN}})$.

Base Class	$\mathbf{VER}(\mathbf{wb}_\mathcal{C})$ weak	$\mathbf{VER}(\mathbf{b}_\mathcal{C})$ strong	$\mathbf{VER}(\mathbf{db}_\mathcal{C})$ deletion
$\mathcal{C} \in$ Schaefer	W[2]-h [NRS04] (FPT)	FPT [NRS04]	FPT [NRS04]
$\mathcal{C} \in$ Subsolver	W[P]-c [Sze05]	W[P]-c [Sze05]	n/a
FOREST	W[2]-h [GS12a] (FPT [GS12a])	?† (?)	FPT
RHORN	W[2]-h	W[2]-h (?)	FPT [RO08]
QHORN	W[2]-h[GOR+16]	W[2]-h[GOR+16] (?)	FPT[RS17]
CLU	W[2]-h [NRS07] (FPT)	W[2]-h [NRS07] (FPT [NRS07])	FPT [NRS07]

( ) It is indicated in parentheses if the complexity of the problem for 3CNF formulas is different from general CNF or unknown.
? It is open whether the problem is fixed-parameter tractable.
† Theorem 17.4.11 shows that the problem is fixed-parameter approximable.
n/a Deletion backdoor sets are undefined for base classes that are not clause-induced.

Table 17.1. The parameterized complexity of the detection of weak, strong, and deletion $\mathcal{C}$-backdoor sets, i.e., $\mathbf{VER}(\mathbf{wb}_\mathcal{C})$, $\mathbf{VER}(\mathbf{b}_\mathcal{C})$, and $\mathbf{VER}(\mathbf{db}_\mathcal{C})$, respectively, for various base classes $\mathcal{C}$.

Table 17.1, which is adapted from a survey paper [GS12b], gives an overview of parameterized complexity results of backdoor set detection problems for various base classes.

17.4.2. Heterogeneous Base Classes

One can enhance the power of a strong backdoor set B of a formula F, by allowing that for different assignments τ to B, $F[\tau]$ belongs to different base classes. This enhancement can be expressed in terms of *heterogeneous base classes* which are the union of individual base classes.

Example 17.4.4. Consider the following CNF formula $F_n = \{C, D_1, \ldots, D_n\}$ where $C = \{x, \neg a_1, \ldots, \neg a_n\}$ and $D_i = \{\neg x, b_i, c_i\}$. It is easy to see that any strong HORN-backdoor set needs to contain at least one of the variables b_i or c_i from each clause D_i, hence such a backdoor set must be of size $\geq n$; on the other hand, any strong 2CNF-backdoor set must contain at least $n-2$ variables from the clause C; However, $F_n[x=0] \in$ HORN and $F_n[x=1] \in$ 2CNF, hence the singleton $\{x\}$ is a strong HORN$\cup$2CNF-backdoor set, where HORN$\cup$2CNF is the heterogeneos base class consisting of all Horn and all 2CNF formulas. Note that any $F \in$ HORN$\cup$2CNF contains either only Horn clauses or only 2CNF clauses, not a mixture of both. □

Weak backdoor sets with respect to heterogeneous base classes can also be considered. Identifying a base class with a class of instances that are solvable by a particular polynomial-time subsolver, one can consider a heterogeneous base class as a "portfolio subsolver," where for each instance the best suitable subsolver from the portfolio is chosen.

The concept of heterogeneos base classes was introduced by Gaspers et al. [GMO+17] who studied the parameterized complexity of backdoor detection with respect to heterogeneos base classes in the context of SAT and CSP. For SAT,

they gave a full classification of the problem for heterogeneos base classes that are composed of base classes from Schaefer. For a nonempty subset $\mathcal{S} \subseteq$ Schaefer, let $\mathcal{S}^\cup = \bigcup_{\mathcal{C}\in\mathcal{S}} \mathcal{C}$. It turned out that the detection of strong $\mathcal{S}^\cup$-backdoor sets is fixed-parameter tractable for all nonempty subsets $\mathcal{S} \subseteq$ Schaefer that do not contain any of the four pairs $\{\text{Horn}, \text{Horn}^-\}$, $\{0\text{-Val}, 1\text{-Val}\}$, $\{\text{Horn}, 1\text{-Val}\}$, $\{0\text{-Val}, \text{Horn}^-\}$, and is W[2]-hard otherwise. Thus, from the 31 nonempty subsets of Schaefer, the problem is fixed-parameter tractable for 13 of them, and W[2]-hard for the remaining 18. The fixed-parameter tractable cases can also we described in terms of the two maximal subsets $\{2\text{CNF}, \text{Horn}, 0\text{-Val}\}$ and $\{2\text{CNF}, \text{Horn}^-, 1\text{-Val}\}$:

Theorem 17.4.10 ([GMO+17]). *Let $\emptyset \neq \mathcal{S} \subseteq$ Schaefer. If $\mathcal{S} \subseteq \{2\text{CNF}, \text{Horn}, 0\text{-Val}\}$ or $\mathcal{S} \subseteq \{2\text{CNF}, \text{Horn}^-, 1\text{-Val}\}$, then $\mathbf{VER}(\mathbf{b}_{\mathcal{S}^\cup})$, and consequently $\mathbf{SAT}(\mathbf{b}_{\mathcal{S}^\cup})$, are fixed-parameter tractable. Otherwise, $\mathbf{VER}(\mathbf{b}_{\mathcal{S}^\cup})$ is* W[2]*-hard.*

Concerning the detection of weak backdoor sets, the W[2]-hardness for individual base classes from Schaefer propagates to heterogeneous base classes that are composed from subsets of Schaefer [GMO+17].

17.4.3. Acyclic Formulas

Many NP-hard problems can be solved in polynomial time for problem instances that are in a certain sense acyclic. The satisfiability problem is no exception. There are various ways of defining a CNF formula to be acyclic. Here we consider acyclicity based on (undirected) incidence graphs (see Section 17.2.3). Let Forest denote the class of CNF formulas whose undirected incidence graphs are forests. It is well known that the satisfiability problem can be solved in polynomial time for acyclic formulas [FMR08, SS10a]. The detection of deletion Forest-backdoor sets is fixed-parameter tractable as one can use variants of algorithms for finding feedback vertex sets (also known as cycle cut sets) in graphs. It is not known, whether the detection of strong Forest-backdoors is fixed-parameter tractable. However, Gaspers and Szeider [GS12a] showed that the problem is fixed-parameter approximable. They gave a fixed-parameter algorithm that either outputs a strong Forest-backdoor set of size $\leq 2^k$, or correctly decides that the given formula has no strong Forest-backdoor set of size $\leq k$. This result renders $\mathbf{SAT}(\mathbf{b}_{\text{Forest}})$ fixed-parameter tractable. The detection of weak Forest-backdoor sets is W[2]-hard, but fixed-parameter tractable if the input formula is in 3CNF [GS12a].

Theorem 17.4.11 ([GS12a]). *The problem $\mathbf{SAT}(\mathbf{b}_{\text{Forest}})$ is fixed parameter tractable.*

The class BAC of *β-acyclic* formulas, which was studied by Paulusma et al. [OPS13], is another possible base class defined in terms of acyclicity. A CNF formula is β-acyclic if its incidence graph $G^*(F)$ is chordal bipartite, which means that $G^*(F)$ has no induced cycle on six or more vertices (a cycle is induced of there are no edges between non-consecutive vertices). SAT-decision, as well as recognition, are polynomial-time solvable problems for BAC, which is a clause-induced

base class. The detection of strong BAC-backdoor sets is W[2]-hard, but it is open whether the detection of deletion BAC-backdoor sets is fixed-parameter tractable [OPS13]. The W[2]-hardness of the detection of weak BAC-backdoor sets can be established by instantiating the above mentioned generic reduction [GS12b, Proposition 1].

17.4.4. Hitting Formulas

Iwama [Iwa89] observed that one could determine the number of models of a CNF formula in polynomial time if any two clauses of the formula clash; such formulas are known as *hitting formulas* [KZ01]. Consider a hitting formula F with n variables. If a total truth assignment $\tau : \mathrm{var}(F) \to \{0,1\}$ does *not* satisfy a clause $C \in F$, it satisfies all other clauses of F. Hence we can count the total truth assignments that do not satisfy F by considering one clause after the other, and the number of models is therefore exactly $2^n - \sum_{C \in F} 2^{n-|C|}$. Of course, if a formula is a variable-disjoint union of hitting formulas—we call such a formula a *cluster formula*—we can still compute the number of models in polynomial time by taking the product of the number of models for each component. Since satisfiability (and obviously recognition) of cluster formulas can be established in polynomial time, the class CLU of cluster formulas is a base class. CLU is evidently clause induced.

Nishimura, Ragde, and Szeider considered the parameterized problem of detecting CLU-backdoor sets.

Theorem 17.4.12 ([NRS07])**.** $\mathbf{VER}(\mathbf{b}_{\mathrm{CLU}})$ *is* W[2]*-hard but* $\mathbf{VER}(\mathbf{db}_{\mathrm{CLU}})$ *is fixed-parameter tractable.*

The hardness result is obtained by an fpt-reduction from the parameterized hitting set problem **HS**. The FPT result is achieved employing an algorithm that systematically destroys certain obstructions that consist of pairs or triples of clauses. To this end, the *obstruction graph* of a CNF formula F is considered. The vertex set of this graph is the set of variables of F; two variables x, y are joined by an edge if and only if at least one of the following conditions hold:

1. F contains two clauses C_1, C_2 that do not clash, $x \in \mathrm{var}(C_1 \cap C_2)$, and $y \in \mathrm{var}(C_1 \setminus C_2)$;
2. F contains three clauses C_1, C_2, C_3 such that C_1 and C_3 do not clash, $x \in \mathrm{var}((C_1 \setminus C_3) \cap \overline{C_2})$, and $y \in \mathrm{var}((C_3 \setminus C_1) \cap \overline{C_2})$.

Since the vertex covers of the obstruction graph are exactly the deletion CLU-backdoor sets, we can find smallest deletion CLU-backdoor sets by using a vertex cover algorithm. Hence $\mathbf{VER}(\mathbf{db}_{\mathrm{CLU}}(F))$ is fixed-parameter tractable.

Example 17.4.5. Consider formula $F = \big\{\{u,v\},\ \{s,\overline{u},\overline{v}\},\ \{u,\overline{v},w,\overline{r}\},\ \{r,\overline{w},x,y\},\ \{\overline{x},y,z\},\ \{\overline{y},\overline{z}\},\ \{\overline{s},\overline{t}\},\ \{\overline{t}\},\ \{t,w\}\big\}$. The obstruction graph has the edges rw, st, tw, uw, vw; ru, rv, rx, ry, su, sv, wx, wy. The set $B = \{r,s,w\}$ forms a vertex cover of the obstruction graph; there is no vertex cover of size two. B is a deletion CLU-backdoor set and consequently also a strong CLU-backdoor set of F. There is, however, the smaller strong CLU-backdoor set $B' = \{w,s\}$. □

The algorithm of Nishimura et al. outlined above can be used to count the number $\#(F)$ of models of a given formula F. More generally, assume that we have a base class $\mathcal{C}$ such that $\#(F)$ can be computed in polynomial time for every $F \in \mathcal{C}$ (which is the case for CLU). Then, if we have a strong $\mathcal{C}$-backdoor set B of an arbitrary CNF formula F, we can compute $\#(F)$ utilizing the identity

$$\#(F) = \sum_{\tau : B \to \{0,1\}} 2^{d(F,\tau)} \, \#(F[\tau])$$

where $d(F,\tau) = |\mathrm{var}(F-B) \setminus \mathrm{var}(F[\tau])|$ denotes the number of variables that disappear from $F[\tau]$ without being instantiated. Thus determining $\#(F)$ reduces to determining the number of models for $2^{|B|}$ formulas of the base class $\mathcal{C}$. In particular, the above considerations yield a fixed-parameter algorithm for model counting parameterized by the clustering-width. Note, however, that for the classes HORN and 2CNF, the model counting problem is #P-hard (even for monotone formulas) [Rot96]. Thus knowing a small strong backdoor set with respect to these classes does not help to count the number of models efficiently.

We will see two further satisfiability parameters that generalize hitting formulas in Section 17.5.2 (conflict treewidth) and Section 17.6.2 (h-modularity).

17.4.5. Empty Clause Detection

Dilkina, Gomes, and Sabharwal [DGS07] suggested strengthening the concept of strong backdoor sets by means of *empty clause detection*. Let $\mathcal{E}$ denote the class of all CNF formulas that contain the empty clause. For a base class $\mathcal{C}$ we put $\mathcal{C}^{\{\}} = \mathcal{C} \cup \mathcal{E}$; we call $\mathcal{C}^{\{\}}$ the base class obtained from $\mathcal{C}$ by adding *empty clause detection*. Formulas can have much smaller strong $\mathcal{C}^{\{\}}$-backdoor sets than strong $\mathcal{C}$-backdoor sets; Dilkina et al. give empirical evidence for this phenomenon considering various base classes. Note that the addition of empty clause detection makes only sense for strong backdoor sets [DGS07], not for weak or deletion backdoor sets. Dilkina et al. show that given a CNF formula F and an integer k, determining whether F has a strong $\text{HORN}^{\{\}}$-backdoor set of size k is both NP-hard and co-NP-hard (here k is considered only as part of the input and not as a parameter). Thus, the non-parameterized complexity of the search problem for strong HORN-backdoor sets gets harder when empty clause detection is added. Also, the parameterized complexity gets harder, which can be shown using results from Fellows et al. [FSW06].

Theorem 17.4.13 ([Sze08]). *For* $\mathcal{C} \in \{\text{HORN}^{\{\}}, \text{2CNF}^{\{\}}, \text{RHORN}^{\{\}}\}$ *the problem* $\mathbf{VER}(\mathbf{b}_{\mathcal{C}})$ *is* W[1]*-hard.*

17.5. Treewidth

Treewidth is an important graph invariant that measures the "tree-likeness" of a graph. Many otherwise NP-hard graph problems such as Hamiltonicity and 3-colorability are fixed-parameter tractable if parameterized by the treewidth of the input graph. It is generally believed that many practically relevant problems do have low treewidth [Bod93]. For taking the treewidth of a CNF formula, one

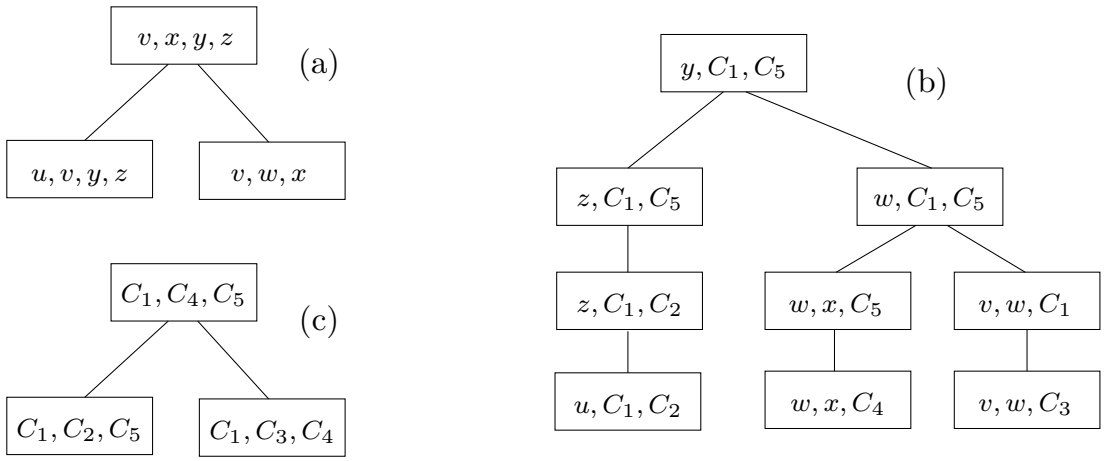


Figure 17.3. Tree decompositions of the primal graph (a), the incidence graph (b), and the dual graph (c)

needs to represent the structure of a formula as a graph. For this, we can use the graphs discussed in Section 17.2.3

Tree decompositions of graphs and the associated parameter treewidth were studied by Robertson and Seymour in their famous Graph Minors Project. A *tree decomposition* of a graph $G = (V, E)$ is a tree $T = (V', E')$ together with a labeling function $\chi : V' \to 2^V$ associating to each tree node $t \in V'$ a bag $\chi(t)$ of vertices in V such that the following three conditions hold:

1. every vertex in V occurs in some bag $\chi(t)$;
2. for every edge $xy \in E$ there is a bag $\chi(t)$ that contains both x and y;
3. if $\chi(t_1)$ and $\chi(t_2)$ both contain x, then each bag $\chi(t_3)$ contains x if t_3 lies on the unique path from t_1 to t_2.

The *width* of a tree decomposition is $\max_{t \in V'} |\chi(t)| - 1$. The *treewidth* of a graph is the minimum width over all its tree decompositions. The treewidth of a graph is a measure for its acyclicity, i.e., the smaller the treewidth, the less cyclic the graph. In particular, a graph is *acyclic* if and only if it has treewidth 1.

The above definition of a tree decomposition can be easily generalized to hypergraphs by requiring in item (2) that all vertices in each hyperedge occur together in some bag. Every tree decomposition of the primal graph $G(F)$ of a CNF formula F is a tree decomposition of the hypergraph $H(F)$. Thus, if the treewidth of the primal graph is k, the cardinality of each clause of F cannot be larger than $k + 1$.

For a CNF formula F, we introduce the following notions of treewidth: the *(primal) treewidth* $\mathbf{tw}$ of F is the treewidth of its primal graph $G(F)$, the *incidence treewidth* $\mathbf{tw}^*$ of F is the treewidth of its incidence graph $G^*(F)$, and the *dual treewidth* $\mathbf{tw}^d$ of F is the treewidth of its dual graph $G^d(F)$. Tree decompositions of the three graphs associated with formula F in Figure 17.1 are shown in Figure 17.3. Since there are no tree decompositions of these graphs of smaller width, we know that $\mathbf{tw}(F) = 3$ and $\mathbf{tw}^*(F) = \mathbf{tw}^d(F) = 2$.

Kolaitis and Vardi [KV00] have shown that always $\mathbf{tw}^*(F) \leq \mathbf{tw}(F) + 1$ and $\mathbf{tw}^*(F) \leq \mathbf{tw}^d(F) + 1$. In other words, the incidence treewidth dominates

the primal treewidth and the dual treewidth. On the other hand, there exist families of CNF formulas with incidence treewidth one and arbitrarily large primal treewidth and dual treewidth, i.e., this domination is strict.

Example 17.5.1. Consider the two families $F_n = \{\{x_1, x_2, \ldots, x_n\}\}$ and $G_n = \{\{x_1, y\}, \{x_2, y\}, \ldots, \{x_n, y\}\}$ of CNF formulas. Then $\mathbf{tw}^*(F_n) = \mathbf{tw}^*(G_n) = 1$ while $\mathbf{tw}(F_n) = \mathbf{tw}^d(G_n) = n - 1$. □

The intuitive idea of tree decompositions is to partition a graph into clusters of vertices that can be organized as a tree. The smaller the width of a tree decomposition, the more efficiently we can decide satisfiability of the corresponding CNF formula by a bottom-up dynamic programming approach on the tree decomposition. Thus, we aim to construct a tree decomposition of width as small as possible; in the optimal case, the width of the tree decomposition equals the treewidth of the graph.

In general, computing the treewidth of a graph is NP-hard [ACP87]. However, since tree decompositions with large width do not help us in deciding satisfiability efficiently, we are more interested in graphs with small treewidth. Bodlaender [Bod96] has shown that it can be decided in linear time whether the treewidth of a graph is at most k if k is a constant. This immediately implies fixed-parameter tractability of the problems $\mathbf{VER}(\mathbf{tw})$, $\mathbf{VER}(\mathbf{tw}^*)$, and $\mathbf{VER}(\mathbf{tw}^d)$. In Section 17.5.4 we will review algorithms for constructing tree decompositions.

17.5.1. Deciding Satisfiability

As mentioned above, if a tree decomposition of the primal graph, the incidence graph, or the dual graph is given, we can decide satisfiability of the corresponding CNF formula by a bottom-up dynamic programming approach on the tree decomposition. The smaller the width of the given tree decomposition, the more efficiently we can decide satisfiability. In particular, from Yannakakis's algorithm [Yan81] we obtain the following result as already observed by Gottlob et al. [GSS02].

Theorem 17.5.1. *The problem* $\mathbf{SAT}(\mathbf{tw})$ *is fixed-parameter tractable.*

To see this, consider a tree decomposition of the primal graph of a given CNF formula F and let k be the width of this tree decomposition. Note that the number of nodes of the tree can be bounded by the length $n = \|F\|$. Now, we associate with each node t of the tree a table M_t with $|\chi(t)|$ columns and at most $2^{|\chi(t)|}$ rows. Each row contains Boolean values encoding a truth assignment to the variables in $\chi(t)$ that does not falsify any clause of F. The size of each table is therefore bounded by $2^{k+1}(k+1)$ and all such tables can be computed in time $\mathcal{O}(2^k kn^2)$. In this way we can transform our SAT problem into an equivalent constraint satisfaction problem by a fixed-parameter algorithm with parameter treewidth. This constraint satisfaction problem can now be solved by Yannakakis's algorithm in time $\mathcal{O}(4^k kn)$. Yannakakis's algorithm works as follows: for each node t of the tree it is checked whether to each truth assignment in table $M_{t'}$ associated with t's parent t' there exists a consistent truth assignment in table M_t. We remove truth assignments in table $M_{t'}$ to which no such consistent truth assignment

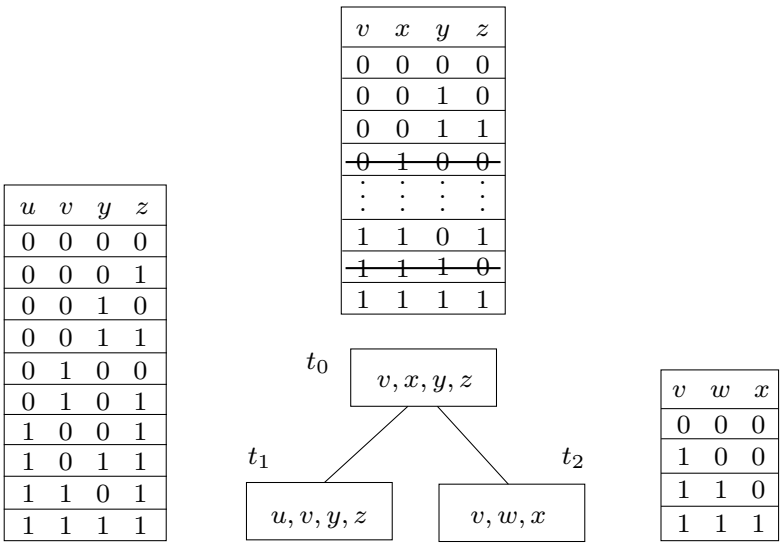


Figure 17.4. A fixed-parameter algorithm for **SAT**(**tw**)

in table M_t exists. The whole procedure works in a bottom-up manner, i.e., a node is processed if all of its children have already been processed. The CNF formula F is satisfiable if and only if some truth assignments are left in the table associated with the root after termination of this procedure. Thus, in summary, we can decide **SAT**(**tw**) in time $\mathcal{O}^*(4^k)$. By using an improved algorithm, we can decide **SAT**(**tw**) in time $\mathcal{O}^*(2^k)$ [SS10a].

Example 17.5.2. Consider the primal graph of the CNF formula F in Figure 17.1 and its tree decomposition in Figure 17.3(a). The tables associated with each tree node are shown in Figure 17.4: there are 14 truth assignments in table M_{t_0} associated with the root t_0, 10 in table M_{t_1} associated with the left child t_1, and 4 in table M_{t_2} associated with the right child t_2. Now let us start with the left child t_1 and remove the rows 1010 and 1110 from table M_{t_0} since there are no consistent truth assignments in table M_{t_1}. Then we consider the right child t_2 and remove the rows 0100, 0101, 0110, and 0111 from table M_{t_0} since there are no consistent truth assignments in table M_{t_2}. Since there are no further nodes to be processed, we are finished and know that F is satisfiable. □

Since $\mathbf{tw}^*$ strictly generalizes **tw**, the following result is stronger than Theorem 17.5.1.

Theorem 17.5.2. *The problem* **SAT**($\mathbf{tw}^*$) *is fixed-parameter tractable.*

Since incidence treewidth strictly dominates primal treewidth and dual treewidth as already mentioned above, this result implies both Theorem 17.5.1 and fixed-parameter tractability of **SAT**($\mathbf{tw}^d$). The situation is different for "generalized satisfiability" also known as "Boolean constraint satisfaction" where Boolean relations replace clauses. Generalized satisfiability is fixed-parameter tractable for the parameter primal treewidth but W[1]-hard for the parameter incidence treewidth [SS10b].

In the following, we will discuss three approaches to establishing Theorem 17.5.2.

Proof via a Logic Meta Theorem Courcelle has shown that every graph property that can be expressed in a certain formalism (monadic second-order logic, MSO) can be decided in linear time for graphs of bounded treewidth [Cou88]. This theorem applies to many NP-hard graph properties such as 3-colorability and yields fixed-parameter tractability for these problems with respect to parameter treewidth. Thus MSO theory provides a very general and convenient tool for classifying problems parameterized by treewidth as fixed-parameter tractable. Using the general methods of MSO theory, one can easily establish fixed-parameter tractability of $\mathbf{SAT}(\mathbf{tw}^*)$ [CMR01, GS08, Sze04b]. However, the algorithms obtained via the generic constructions are impractical.

Proof via Clause Splitting It is well known (see, e.g., [GJ79, p. 48]) that a CNF formula F can be transformed in polynomial time into an equisatisfiable 3CNF formula F_3, by repeatedly splitting a clause $(\ell_1 \vee \ell_2 \vee \ell_3 \vee \cdots \vee \ell_r)$ with $r \geq 4$ into two clauses $(\ell_1 \vee \ell_2 \vee x)$ and $(\neg x \vee \ell_3 \cdots \vee \ell_r)$ where x is a new variable. Samer and Szeider [SS10b, Remark, pp. 111] have shown that in general this procedure can increase the incidence treewidth arbitrarily, but if applied properly (respecting an ordering of clauses and variables that is inferred from a tree decomposition of the incidence graph), the incidence treewidth stays bounded.

Lemma 17.5.3 (Splitting Lemma [SS10b])**.** *Given a CNF formula F together with a tree decomposition of width k of the incidence graph of F. By splitting clauses we can obtain in polynomial time an equisatisfiable 3CNF formula with incidence treewidth at most $k+1$ and primal treewidth at most $3(k+1)$.*

Thus, there is an fpt-reduction from $\mathbf{SAT}(\mathbf{tw}^*)$ to $\mathbf{SAT}(\mathbf{tw})$, and hence Theorem 17.5.1 implies Theorem 17.5.2.

Proof via Dynamic Programming For more practical algorithms, however, one needs to use more closely the combinatorial structure of the particular problem at hand. Fischer et al. [FMR08] and Samer and Szeider [SS10a] presented practical fixed-parameter algorithms for the more general problem $\#\mathbf{SAT}(\mathbf{tw}^*)$ of counting the number of models. This trivially implies Theorem 17.5.2, since a CNF formula is satisfiable if and only if it has at least one model. In the following we present the algorithm introduced by Samer and Szeider. The algorithm is based on "nice" tree decompositions, which are a special kind of tree decompositions. It is well known that one can transform any tree decomposition of width k in linear time into a nice tree decomposition of width at most k [BK96, Klo94].

For each node t, we write X_t and F_t to denote the set of all variables and clauses occurring in $\chi(t')$, respectively, for some node t' in the subtree rooted at t. Moreover, we use the shorthands $\chi_v(t) = \chi(t) \cap X_t$ and $\chi_c(t) = \chi(t) \cap F_t$ for the set of variables and the set of clauses in $\chi(t)$ respectively. For each truth assignment $\alpha : \chi_v(t) \to \{0,1\}$ and subset $A \subseteq \chi_c(t)$, we define $N(t, \alpha, A)$ as the set of truth assignments $\tau : X_t \to \{0,1\}$ for which the following two conditions hold:

1. $\tau(x) = \alpha(x)$ for all variables $x \in \chi_v(t)$ and
2. A is exactly the set of clauses of F_t that are not satisfied by τ.

Now, we associate with each node t of the tree a table M_t with $|\chi(t)| + 1$ columns and $2^{|\chi(t)|}$ rows. The first $|\chi(t)|$ columns contain Boolean values encod-

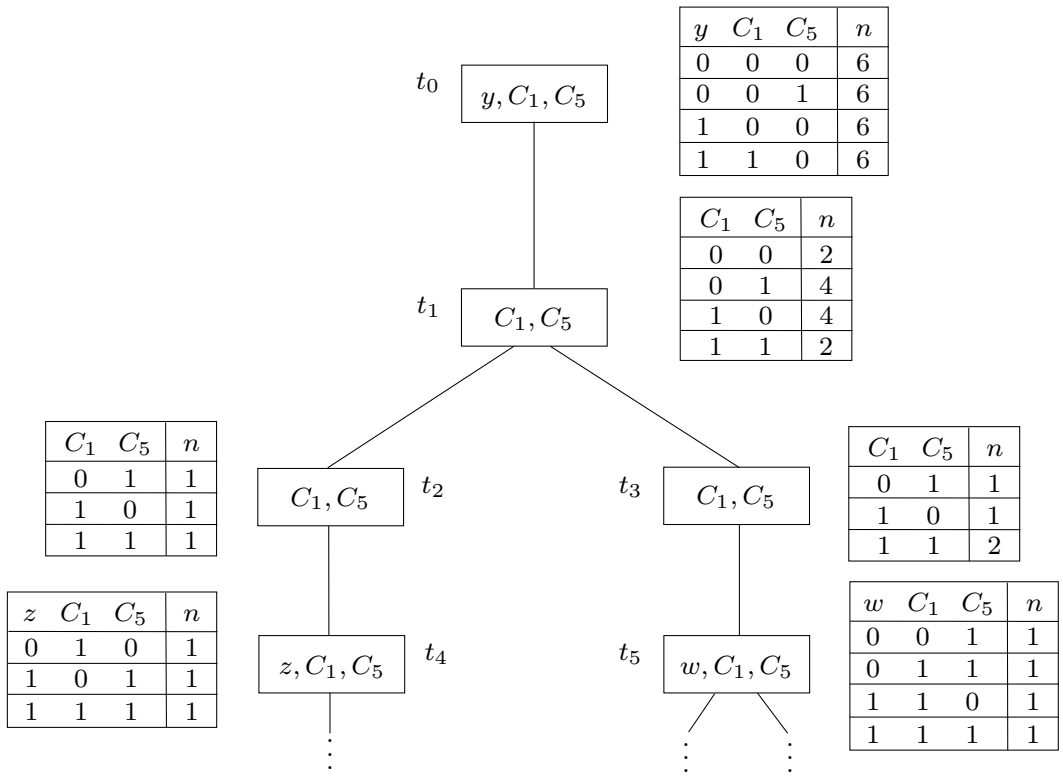


Figure 17.5. A fixed-parameter algorithm for **#SAT**(**tw***)

ing $\alpha(x)$ for variables $x \in \chi_v(t)$, and membership of C in A for clauses $C \in \chi_c(t)$. The last column contains the integer $n(t, \alpha, A) = |N(t, \alpha, A)|$. Given the tables of the children of some node t, the table M_t can be computed in time $\mathcal{O}(4^k kl)$, where l is the cardinality of the largest clause. All the tables associated with tree nodes can be computed in a bottom-up manner. The number of models of the corresponding CNF formula F is then given by $\Sigma_{\alpha:\chi_v(r)\to\{0,1\}} n(r, \alpha, \emptyset)$, where r is the root of the tree. Thus, we can decide **#SAT**(**tw***) in time $\mathcal{O}^*(4^k)$. Based on an observation by Björklund [Bjö18], Slivovsky and Szeider [SS20] improved the running time for **#SAT**(**tw***) to $\mathcal{O}^*(2^k)$ through efficiently computing *covering products* [BHKK07].

Example 17.5.3. Consider the incidence graph of the CNF formula F in Figure 17.1 and its tree decomposition in Figure 17.3(b). A fragment of the corresponding nice tree decomposition and the tables associated with each tree node are shown in Figure 17.5. Note that we omit for simplicity those rows from the tables where $n = 0$. We assume that the tables M_{t_4} and M_{t_5} associated with the nodes t_4 and t_5 respectively have already been computed in a bottom-up manner starting from the leaves. For example, the entries in table M_{t_4} mean that (i) there exists exactly one truth assignment $\tau : X_{t_4} \to \{0, 1\}$ such that $\tau(z) = 0$ and τ satisfies all clauses in F_{t_4} except C_1, (ii) there exists exactly one truth assignment $\tau : X_{t_4} \to \{0, 1\}$ such that $\tau(z) = 1$ and τ satisfies all clauses in F_{t_4} except C_5, and (iii) there exists exactly one truth assignment $\tau : X_{t_4} \to \{0, 1\}$ such that $\tau(z) = 1$ and τ satisfies all clauses in F_{t_4} except C_1 and C_5. The next step is to compute the tables M_{t_2} and M_{t_3} from tables M_{t_4} and M_{t_5} respectively.

Since t_2 and t_3 are forget nodes (the variable z has been forgotten in t_2 and the variable w has been forgotten in t_3), this can be done according to the rule for forget nodes as given in [SS10a]. Now we compute table M_{t_1} from tables M_{t_2} and M_{t_3} according to the rule for join nodes. Finally, we compute table M_{t_0} from table M_{t_1} according to the rule for introduce nodes. From table M_{t_0} we can now see that there are exactly 12 truth assignments $\tau : X_{t_0} \to \{0, 1\}$ such that τ satisfies all clauses in F_{t_0} (for 6 of these truth assignments it holds that $\tau(y) = 0$ and for 6 of them it holds that $\tau(y) = 1$), where $X_{t_0} = \mathrm{var}(F)$ and $F_{t_0} = F$. Consequently, the CNF formula F has exactly 12 models. □

Bacchus, Dalmao, and Pitassi [BDP03] presented another fixed-parameter algorithm for computing the number of models of a CNF formula F. The parameter in their algorithm is the *branchwidth* of the hypergraph $H(F)$. Similar to tree decompositions, branch decompositions and the corresponding branchwidth were introduced by Robertson and Seymour in their Graph Minors Project. It is well known that a graph with treewidth k has branchwidth at most $k + 1$ and that a graph with branchwidth k has treewidth at most $3k/2$ [RS91]. Thus, primal treewidth and branchwidth are domination equivalent satisfiability parameters. Bacchus et al. define a static ordering of the variables of F based on the branch decomposition of $H(F)$ and run a DPLL procedure with caching on this ordering. In particular, they decompose the input formula and intermediate formulas into disjoint components; these components are cached when they are solved the first time, which allows truncating the search-tree of the DPLL procedure. The resulting algorithm runs in time $2^{\mathcal{O}(k)}n^c$, where k is the branchwidth, n is the number of variables, and c is a constant.

17.5.2. Consensus Treewidth and Conflict Treewidth

Recall from above the definitions of the consensus graph $G^{\dot{\vee}}(F)$ and the conflict graph $G^{\dot{\wedge}}(F)$, associated with a CNF formula F. We define the *consensus treewidth* $\mathbf{tw}^{\dot{\vee}}(F) = \mathrm{tw}(G^{\dot{\vee}}(F))$ and the *conflict treewidth* $\mathbf{tw}^{\dot{\wedge}}(F) = \mathrm{tw}(G^{\dot{\wedge}}(F))$. Ganian and Szeider [GS17] introduced these two satisfiability parameters and studied their parameterized complexity. The consensus treewidth turned out to be an interesting parameter which is not dominated by any known satisfiability parameter and admits fixed-parameter tractability.

Theorem 17.5.4 ([GS17])**.** *The problem* $\#\mathbf{SAT}(\mathbf{tw}^{\dot{\vee}})$ *is fixed-parameter tractable.*

On the other hand, conflict treewidth seems to be not that useful:

Theorem 17.5.5 ([GS17])**.** *The problem* $\mathbf{SAT}(\mathbf{tw}^{\dot{\wedge}})$ *is* W[1]*-hard.*

W[1]-hardness is avoided by bounding the size of clauses and eliminating pure literals. For such formulas, however, conflict treewidth is dominated by incidence treewidth.

17.5.3. Expression Treewidth

Before discussing algorithms for computing tree decompositions, let us briefly mention a very general approach to applying treewidth to CNF formulas (or rather

Boolean functions in general), that was suggested by Jha and Suciu [JS12]. They define the *expression treewidth* of a Boolean function f as the smallest treewidth of any Boolean circuit representing the function. The circuit is considered as a directed acyclic graph D, and its treewidth is taken from the undirected graph $\underline{D}$ obtained from D by ignoring the orientation of edges. By letting f_F denote the Boolean function represented by a CNF formula F, we can define the satisfiability parameter expression treewidth $\mathbf{xtw}(F)$ as the expression treewidth of f_F. Expression treewidth strictly dominates incidence treewidth [JS12]. Courcelle's meta-theorem [Cou88] can be used to show that $\mathbf{SAT}(\mathbf{xtw})$ is fixed-parameter tractable if the circuit D with minimal treewidth is provided as the input. Bova and Szeider [BS17] have shown that $\mathbf{VER}(\mathbf{xtw})$ is at least decidable, using a meta-theorem by Seese [See91]. It would be interesting to know more efficient algorithms for this problem.

17.5.4. Tree Decomposition Algorithms

As for most algorithms, also in the case of computing tree decompositions, there has to be a tradeoff made between runtime, space requirement, and simplicity. In the following, we use n to denote the number of vertices of the given graph. The current fastest exact tree decomposition algorithm runs in time $\mathcal{O}^*(1.8899^n)$ and is due to Fomin, Kratsch, and Todinca [FKT04] and Villanger [Vil06]. This algorithm is based on the computation of potential maximal cliques. Bodlaender et al. [BFK$^+$06] developed a simpler algorithm based on a recursive divide-and-conquer technique that requires polynomial space and runs in time $\mathcal{O}^*(4^n)$. For special classes of graphs, however, there exist exact tree decomposition algorithms that run in polynomial (or even linear) time [Bod93].

Polynomial-time algorithms also exist in the case of bounded treewidth. These algorithms are fixed-parameter algorithms: Reed's algorithm [Bod93, Ree92] decides in time $\mathcal{O}(n \log n)$ whether the treewidth of a graph is at most k and, if so, computes a tree decomposition of width at most $3k + 2$. Bodlaender and Kloks [BK96] developed an algorithm with the same asymptotic runtime as Reed's algorithm that decides whether the treewidth of a graph is at most k and, if so, computes a tree decomposition of width at most k. Bodlaender [Bod96] improved this result to a linear-time algorithm. The hidden constant factors in the runtime of the latter two algorithms, however, are very large so that they are only practical for very small k (e.g., up to $k = 5$) [BK96, Bod05].

Algorithms that approximate treewidth by finding tree decompositions of *nearly* minimal width give a guarantee on the quality of the output. Bodlaender et al. [BDD$^+$16] gave a single exponential algorithm that runs in $2^{O(k)}n$ time, and either outputs a tree decomposition of the given graph G of width at most $5k + 4$, or determines that $\mathrm{tw}(G) > k$.

In practice, it often suffices to obtain tree decompositions of small width without any guarantees. There exist several powerful tree decomposition heuristics for this purpose. In the worst case, the width of tree decompositions obtained by such heuristics can be far from treewidth; however, their width is often small in practically relevant cases. An important class of tree decomposition heuristics is based on finding an appropriate linear ordering of the vertices from which

a tree decomposition can be constructed [Bod05]. *Minimum degree* and *minimum fill-in* [Bod05], *lexicographic breadth-first search* [RTL76], and *maximum cardinality search* [TY84] are well-known examples of such ordering heuristics. Koster, Bodlaender, and van Hoesel [KBvH01a, KBvH01b] compared several tree decomposition heuristics by empirical evaluation.

We refer the interested reader to Bodlaender's excellent survey papers [Bod93, Bod05] for a more extensive overview of tree decomposition algorithms.

One can also use SAT solvers to determine the treewidth of graphs. Samer and Veith [SV09] gave a first SAT encoding, that, given a graph G and an integer k, produces a CNF formula $F(G,k)$ which is satisfiable if and only if $\mathrm{tw}(G) \leq k$. This approach was further improved [BJ14, BBE17], as well as used to improve a heuristically computed tree decomposition locally [FLS17].

17.5.5. Beyond Treewidth

Beside treewidth, other decomposition-based measures for the tractability of certain computation problems with graph and hypergraph representations have been proposed in the literature. One of the most prominent examples is *clique-width* [CER93]. Intuitively, the clique-width $\mathrm{cwd}(G)$ of a graph G is the smallest number of colors required to construct the graph by means of certain operations that do not distinguish between vertices of the same color. Clique-width is defined for undirected graphs and directed graphs. Let $\underline{D}$ be the undirected graph obtained from a directed graph D by ignoring the orientation of edges. Then $\mathrm{cwd}(\underline{D}) \leq \mathrm{cwd}(D)$.

Bounding the clique-width of primal graphs does not help with deciding satisfiability: one can easily make the primal graph a clique as follows: Let F be a CNF formula and x a new variable not occurring in F. Now consider the CNF formula F^* obtained from F by adding the two clauses $C = \mathrm{var}(F) \cup \{x\}$ and $C' = \{x\}$. Clearly F and F^* are equisatisfiable and even share the same number of models. Now the primal graph of F^* is a clique and so it has clique-width 2. For the incidence graph, however, bounding the clique-width helps, as we shall see below.

We distinguish between $\mathbf{cwd}^*(F) = \mathrm{cwd}(G^*(F))$ and $\mathbf{dcwd}^*(F) = \mathrm{cwd}(D^*(F))$. From results by Courcelle and Olariu [CO00] it follows that $\mathbf{cwd}^*$ strictly dominates $\mathbf{dcwd}^*$, and $\mathbf{dcwd}^*$ strictly dominates $\mathbf{tw}^*$. The problems $\mathbf{VER}(\mathbf{cwd}^*)$ and $\mathbf{VER}(\mathbf{dcwd}^*)$ are not known to be fixed-parameter tractable, but fixed-parameter approximable [Oum05, FMR08].

Theorem 17.5.6 ([OPS13])**.** *The problem* $\mathbf{SAT}(\mathbf{cwd}^*)$ *is* W[1]*-hard.*

This hardness even holds if the corresponding decomposition is provided with the input.

Theorem 17.5.7 ([SS13])**.** $\#\mathbf{SAT}(\mathbf{cwd}^*)$ *is XP-tractable.*

This result is obtained by a dynamic programming algorithm in which certain projections of truth assignments play an important role. Such projections have already been used previously for showing XP-tractability of **#SAT** for the satisfiability parameter *modular treewith* [PSS16] which is the treewidth of the

incidence graph, taken after the contraction of modules (i.e., of vertices with the same neighbors). Modular treewidth strictly dominates treewidth and is, in turn, strictly dominated by clique-width. Sæther et al. [STV15] generalized the algorithm underlying Theorem 17.5.7 to be applicable to larger classes of formulas, however, requiring the corresponding decomposition to be provided with the input.

Theorem 17.5.8 ([CMR01, FMR08])**.** **#SAT**(**dcwd***) *is fixed-parameter tractable.*

The fixed-parameter tractability of **#SAT**(**dcwd***) follows from meta-theorem of monadic second-order logic [CMR01] similarly as in the case of treewidth. Also a direct dynamic programming algorithm is known [FMR08]. For computing clique-width decompositions of (directed) incidence graphs one can use SAT-encodings [HS15, Par16].

Rank-width [RS86] is a graph invariant that is similar to clique-width and can also be defined for directed and undirected incidence graphs. The corresponding satisfiability parameters are domination equivalent with directed and undirected clique-width, respectively. For satisfiability checking and model counting, directed rank-width has an advantage over **dcwd*** since the corresponding verification problem is known to be fixed-parameter tractable and so one saves the approximation error. Moreover, even if optimal clique-width or rank-width decompositions are provided, the rank-width based algorithm can be exponentially faster than the clique-width based algorithm [GHO13].

Generalizations of treewidth like hypertree-width [GLS02], spread-cut width [CJG08], and fractional hypertree-width [GM06] are defined for hypergraphs in the context of constraint satisfaction and conjunctive database queries. According to the current status of knowledge, they have no relevance for the satisfiability problem of CNF formulas [SS10a]. This can be seen, using the simple construction we considered at the beginning of Section 17.5.5: the hypergraph $H(F^*)$ is acyclic [GLS02], and so hypertree-width, spread-cut width, and fractional hypertree-width of $H(F^*)$ are 1. A similar construction can be made with respect to the dual hypergraph $H^d(F)$ [SS10a].

17.6. Further Satisfiability Parameters

In this section, we will discuss further satisfiability parameters that are (i) based on a combination of backdoor sets and treewidth, (ii) based on a community structure in the formula, and (iii) based on matchings in the incidence graph of the formula.

17.6.1. Hybrid Satisfiability Parameters

The general approaches to satisfiability parameters as discussed in the previous two sections (based on backdoors and decompositions, respectively) are complementary. Take, for instances $\mathbf{b}_{\text{HORN}}$ (size of a smallest strong HORN-backdoor) and **tw*** (treewidth of the incidence graph). The two satisfiability parameters are incomparable, as one can construct Horn formulas of arbitrarily large incidence

treewidth (i.e., $\mathbf{b}_{\text{HORN}}$ is 0 and **tw** is unbounded), and formulas of arbitrarily large $\mathbf{b}_{\text{HORN}}$ and and bounded primal treewidth.

Several ways for combining the strengths of the two approaches have been considered which we will briefly discuss.

Strong Backdoors into Bounded Incidence Treewidth Gaspers and Szeider [GS13] considered strong $\mathcal{W}_{\leq t}$-backdoor sets where $\mathcal{W}_{\leq t}$ consists of all CNF formulas F with $\mathbf{tw}^*(F) \leq t$, where $t > 0$ is a fixed constant. $\mathcal{W}_{\leq t}$ is suitable as a base class (Theorem 17.5.2), even **#SAT** is polynomial-time tractable for $\mathcal{W}_{\leq t}$. For $t = 1$ we obtain the base class FOREST discussed above. It is crucial to consider strong backdoor sets, not just deletion backdoor sets, as $\mathbf{db}_{\mathcal{W}_{\leq t}}$ is easily seen to be dominated by $\mathbf{tw}^*$, whereas $\mathbf{b}_{\mathcal{W}_{\leq t}}$ strictly dominates $\mathbf{tw}^*$ [GS13]. When we know a strong $\mathcal{W}_{\leq t}$-backdoor set of size k of a formula F, we can compute $\#(F)$ in $\mathcal{O}^*(2^k)$ time. The main problem is finding such a backdoor set, i.e., $\mathbf{VER}(\mathbf{b}_{\mathcal{W}_{\leq t}})$. Gaspers and Szeider [GS13] showed that this problem is fixed-parameter approximable, which implies the following result.

Theorem 17.6.1 ([GS13])**.** *For every $t \geq 1$, $\mathbf{\#SAT}(\mathbf{b}_{\mathcal{W}_{\leq t}})$ is fixed-parameter tractable.*

By adjusting t and backdoor set size, one can fine-tune the algorithm behind Theorem 17.6.1 to a particular class of input formulas. Fomin at al. [FLM+15] considered the problems $\mathbf{SAT}(\mathbf{wb}_{\mathcal{W}_{\leq t}})$ and $\mathbf{SAT}(\mathbf{b}_{\mathcal{W}_{\leq t}})$ for the special case where the input is an r-CNF formula for an arbitrary constant r. They showed that the corresponding permissive problems are fixed-parameter tractable. Their algorithms avoid the computation of a fixed-parameter approximation and achieves a single exponential running time.

Backdoor Treewidth Next we consider an approach due to Ganian et al. [GRS17b, GRS17a], where one considers not the size of a smallest strong $\mathcal{C}$-backdoor set of a given CNF formula F as the parameter, but the treewidth of a strong $\mathcal{C}$-backdoor set of smallest treewidth. The treewidth of a strong $\mathcal{C}$-backdoor set B is taken in terms of a *torso graph* G_F^B. The vertices of G_F^B are the variables in B, and an edge connects two variables $x, y \in B$ if and only if the incidence graph $G^*(F)$ contains a path between x and y that does not traverse any vertex in X (except x and y). Thus, the torso graph is obtained by "collapsing" (possibly large parts of) the incidence graph to single edges. The *$\mathcal{C}$-backdoor treewidth* of F, denoted $\mathbf{tw}_{\mathcal{C}}(F)$, is the smallest $\text{tw}(G_F^B)$ over all strong $\mathcal{C}$-backdoor sets B of F. Figure 17.6 shows an example for this construction. We note that, in general, a strong $\mathcal{C}$-backdoor set B of F with smallest $\text{tw}(G_F^B)$ might not be among the smallest strong $\mathcal{C}$-backdoor sets of F. In the following we focus on base classes $\mathcal{C} \in \{\text{HORN}, \text{HORN}^{-1}, \text{2CNF}\}$.

If we know a strong $\mathcal{C}$-backdoor set B with $\text{tw}(G_F^B) = k$, we can compile connected components of $G^*(F) - B$ into constraints, and so represent F by an equisatisfiable Boolean CSP instance over the variables B, whose primal treewidth is k. Solving this CSP instance is fixed-parameter tractable parameterized by k [GSS02]. Hence again, the challenging task is to find the set B, which was shown by Ganian et al. [GRS17a] to be fixed-parameter tractable:

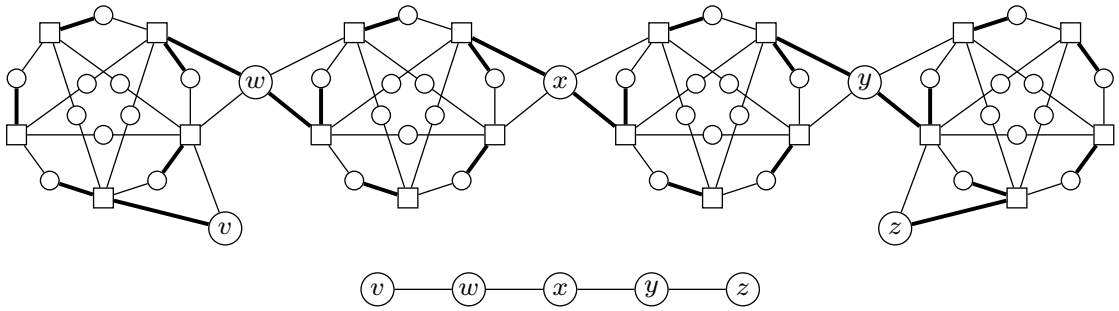


Figure 17.6. Top: an example of a formula F, drawn by its incidence graph, where positive occurrences of variables are indicated with bold edges. $B = \{v, w, x, y, z\}$ forms a strong HORN-backdoor set of F. Bottom: the torso graph G_F^B, which is a path and has, therefore, treewidth 1. Consequently, the HORN-backdoor treewidth is $\mathbf{b}_{\text{HORN}}(F) = 1$.

Theorem 17.6.2 ([GRS17a]). *For $\mathcal{C} \in \{$HORN, HORN$^-$, 2CNF$\}$, the problems* $\mathbf{VER}(\mathbf{tw}_\mathcal{C})$ *and* $\mathbf{SAT}(\mathbf{tw}_\mathcal{C})$ *are fixed-parameter tractable.*

17.6.2. Modularity

Networks that arise from real-world applications frequently exhibit a certain *community structure*, where nodes form strongly interconnected communities which are sparsely connected with each other. With the notion of *modularity* [New03, New06, NG04, ZPWL13] one can measure to what extent a network exhibits such a structure. It was empirically observed, that the performance of SAT solvers shows some correlation with the modularity of the input formulas [ABGL14, NGF$^+$14]. However, the presence of a community structure is not a guarantee for a formula to be tractable. In fact, it is not difficult to show that SAT remains NP-hard for highly modular instances [GS15, MFS16]. Based on this observation, Ganian and Szeider proposed the satisfiability parameter *h-modularity* which is inspired by the general concept of modularity, but in contrast to the existing notion does provide performance guarantees for SAT decision and even model counting. This satisfiability parameter is based on the splitting of the clauses of a given formula into classes, called h-communities, where each class is a hitting formula (see Section 17.4.4), and therefore strongly interconnected. The h-communities are only sparsely connected with each other, as the graph representing their interconnection (the community graph) as small treewidth.

More specifically, we call a subset H of a formula F to be a *hitting community* (or *h-community* in brief) in F if H is a hitting formula. The *degree* $\deg(H)$ of an h-community H is the number of edges in the dual graph of F between a clause in H and a clause outside of H. A *hitting community structure* (or *h-structure* in brief) $\mathcal{P}$ is a partitioning of F into h-communities, and the degree $\deg(\mathcal{P})$ of $\mathcal{P}$ is $\max\{\deg(H) : H \in \mathcal{P}\}$.

To measure the treewidth of an h-structure $\mathcal{P}$, we construct a *community graph* $G(\mathcal{P})$ as follows. The vertices of $G(\mathcal{P})$ are the h-communities in $\mathcal{P}$, and two vertices A, B in $G(\mathcal{P})$ are joined by an edge if and only if there exist clauses $C \in A$ and $D \in B$ which are adjacent.

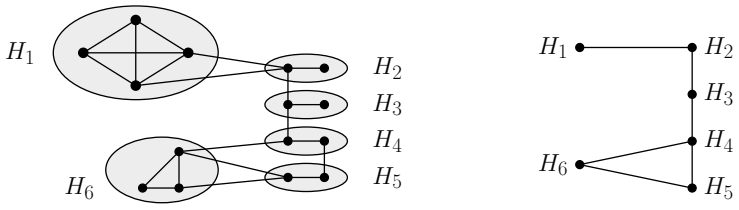


Figure 17.7. The dual graph (left) and community graph (right) of the formula F and the h-structure $\mathcal{P}$.

We define the *h-modularity* of an h-structure $\mathcal{P}$ as the maximum over $\deg(\mathcal{P})$ and $\mathrm{tw}(G(\mathcal{P}))$. The h-modularity $\mathbf{h\text{-}mod}(F)$ of a formula F is then defined as the minimum $\mathbf{h\text{-}mod}(\mathcal{P})$ over all h-structures $\mathcal{P}$ of F.

Example 17.6.1. Consider the formula F which can be split into h-communities $H_1 = \{\{x, y, \neg a\}, \{\neg x, y, a\}, \{x, \neg y\}, \{\neg x, \neg y\}\}$, $H_2 = \{\{a, b, c\}, \{\neg b\}\}$, $H_3 = \{\{c, \neg d, e, f\}, \{d, \neg e\}\}$, $H_4 = \{\{f, \neg g, \neg h\}, \{h, \neg i\}\}$, $H_5 = \{\{i, \neg j\}, \{j, k, l, \neg m, \neg n\}\}$, and $H_6 = \{\{u, \neg v, g, \neg k, l, m, \neg n\}, \{u, v, \neg l, \neg u, v\}\}$. Figure 17.7 (left) shows the dual graph of F with the h-communities indicated. Figure 17.7 (right) shows the corresponding community graph, which is of treewidth 2. The h-communities H_1 and H_3 have degree 2, and all other h-communities have degree 3. Therefore the h-modularity of F is at most $\max(3, 2) = 3$. □

Once we know a community structure $\mathcal{P}$ of a CNF formula F, then we can solve **SAT** and **#SAT** by a fixed-parameter algorithm for parameter $\mathbf{h\text{-}mod}(\mathcal{P})$ by compiling it into an instance of the Sum-of-Products problems (the counting version of CSP), and solve it using known results [BDP09, GKSS18]. For actually finding a community structure $\mathcal{P}$ of small h-modularity, we know a fixed-parameter approximation algorithm [GS15].

Theorem 17.6.3 ([GS15])**.** *The problem* **#SAT**(**h-mod**) *is fixed-parameter tractable.*

17.6.3. Matchings

A *matching* in a graph is a set of edges such that every vertex is incident with at most one edge of the matching. A CNF formula is called *matched* if its incidence graph has a matching such that all clauses are incident with an edge of the matching. Matched formulas are always satisfiable since one can satisfy each clause independently by choosing the right truth value for the variable that is associated with it via the matching.

Example 17.6.2. Consider the CNF formula $F = \{C_1, \ldots, C_4\}$ with $C_1 = \{v, y, z\}$, $C_2 = \{\overline{y}, \overline{x}\}$, $C_3 = \{\overline{v}, \overline{z}, w\}$, $C_4 = \{y, x, \overline{w}\}$. The set $M = \{vC_1, yC_2, zC_3, xC_4\}$ is a matching in the incidence graph of F that covers all clauses. Hence F is a matched formula and it is indeed satisfiable: we put $\tau(v) = 1$ to satisfy C_1, $\tau(y) = 0$ to satisfy C_2, $\tau(z) = 0$ to satisfy C_3, and $\tau(x) = 1$ to satisfy C_4. □

The notion of *maximum deficiency* (first used by Franco and Van Gelder [FV03] in the context of CNF formulas) allows to gradually extend the nice properties from matched formulas to more general classes of formulas. The maximum deficiency of a formula F, denoted by $\mathbf{md}(F)$, is the number of clauses remaining uncovered by a largest matching of the incidence graph of F. The parameters **md** and $\mathbf{tw}^*$ are domination incomparable [Sze04b]. The term "maximum deficiency" is motivated by the equality

$$\mathbf{md}(F) = \max_{F' \subseteq F} \mathbf{d}(F')$$

which follows from Hall's Theorem. Here $\mathbf{d}(F')$ denotes the *deficiency* of F', the difference $|F'| - |\mathrm{var}(F')|$ between the number of clauses and the number of variables. The problem $\mathbf{VER}(\mathbf{md})$ can be solved in polynomial time, since a largest matching in a bipartite graph can be found in polynomial time trough Hopcroft and Karp's algorithm [HK73, LP86] (and the number of uncovered clauses remains the same whatever largest matching one considers).

Deficiency and maximum deficiency have been studied in the context of *minimal unsatisfiable formulas*, i.e., unsatisfiable formulas that become satisfiable by removing any clause. Let **MU** denote the recognition problem for minimal unsatisfiable formulas. By a classic result of Papadimitriou and Wolfe [PW88], the problem **MU** is DP-complete; DP is the class of problems that can be considered as the difference of two problems in NP and corresponds to the second level of the Boolean Hierarchy [Joh90]. Kleine Büning [Kle00] initiated the study of **MU** parameterized by the deficiency **d**. Since $\mathbf{d}(F) = \mathbf{md}(F) \geq 1$ holds for minimal unsatisfiable formulas F [AL86], algorithms for $\mathbf{SAT}(\mathbf{md})$ are of relevance. Fleischner et al. [FKS02] have shown that one can decide the satisfiability of formulas with maximum deficiency bounded by a constant in polynomial time.

As a consequence, minimal unsatisfiable formulas with deficiency bounded by a constant can be recognized in polynomial time. The order of the polynomial that bounds the running time of Fleischner et al.'s algorithm depends on k; hence, it only establishes that $\mathbf{SAT}(\mathbf{md})$ and $\mathbf{MU}(\mathbf{d})$ are in XP. Szeider [Sze04a] developed an algorithm that decides satisfiability and minimal unsatisfiability of formulas with maximum deficiency k in time $\mathcal{O}^*(2^k)$, thus establishing the following result.

Theorem 17.6.4 ([Sze04a])**.** *The problems* $\mathbf{SAT}(\mathbf{md})$ *and* $\mathbf{MU}(\mathbf{d})$ *are fixed-parameter tractable.*

Key for Szeider's algorithm is a polynomial-time procedure that either decides the satisfiability of a given formula F or reduces F to an equisatisfiable formula F^* with $\mathbf{md}(F^*) \leq \mathbf{md}(F)$, such that

$$\mathbf{md}(F^*[x=0]) < \mathbf{md}(F^*) \text{ and } \mathbf{md}(F^*[x=1]) < \mathbf{md}(F^*) \text{ for all } x \in \mathrm{var}(F^*);$$

a formula F^* with this property is called **md**-*critical*. In particular, a formula is **md**-critical if every literal of F^* occurs in at least two clauses and for every non-empty set X of variables of F^* there are at least $|X| + 2$ clauses C of F^* such that $\mathrm{var}(C) \cap X \neq \emptyset$. The above reduction is applied at every node of a (DPLL-type) binary search tree. Since at every step from a node to one of its

children the maximum deficiency of the formula gets reduced, it follows that the height of the search tree is bounded in terms of the maximum deficiency of the given formula, yielding the fixed-parameter tractability of $\mathbf{SAT}(\mathbf{md})$.

Let r be a positive integer and let $\mathcal{M}_r$ denote the class of formulas F with $\mathbf{md}(F) \leq r$. Since both recognition and satisfiability of formulas in $\mathcal{M}_r$ can be solved in polynomial time and, since $\mathcal{M}_r$ is clause induced, it makes sense to consider $\mathcal{M}_r$ as the base class for strong and deletion backdoor sets. However, such backdoor sets are difficult to find:

Theorem 17.6.5 ([Sze08]). *The problems* $\mathbf{VER}(\mathbf{b}_{\mathcal{M}_r})$ *and* $\mathbf{VER}(\mathbf{db}_{\mathcal{M}_r})$ *are* W[2]*-hard for every* $r \geq 1$.

17.7. Concluding Remarks

We close this chapter by briefly mentioning further research on the parameterized complexity of problems related to propositional satisfiability.

For example, Fellows, Szeider and Wrightson [FSW06] have studied the problem of finding in a given CNF formula F a *small unsatisfiable subset* S parameterized by the number of clauses of S. The problem is W[1]-complete, but fixed-parameter tractable for several classes of CNF formulas, including formulas with planar incidence graphs and formulas with both clause size and occurrence of variables bounded. W[1]-hardness prevails if the input formula is Horn [dHKS17]. This is in stark contrast to the case where the input is a 2CNF formula; then the problem can be solved even in polynomial time [BM07].

Propositional proof complexity is a further area of research that is related to satisfiability and admits parameterizations. In particular, one can study proofs that establish that a given CNF formula cannot be satisfied by setting at most k variables to true; k is considered as the parameter. Dantchev, Martin, and Szeider [DMS11] have studied the proof complexity of resolution for such "parameterized contradictions," with very interesting follow-up work [ABdR$^+$18, BGLR12].

Another promising line of research that connects parameterized complexity with SAT solving is the exploration of *fixed-parameter tractable reductions to SAT*. The idea is to solve problems that are believed to be harder than NP (e.g., problems from the second level of the Polynomial Hierarchy) by a fixed-parameter algorithm that can call a SAT solver as an auxiliary device. Such algorithms where developed for disjunctive answer-set programming and propositional abduction problems [PRS13, FS15]. De Haan and Szeider [dHS17, dHS19] developed a parameterized hardness theory for such problems which lies the grounds for classifying problems that admit such fixed-parameter tractable reductions to SAT, and those that don't.

We hope that this survey provides a stimulating starting point for further research on satisfiability and related topics that fruitfully utilizes concepts of parameterized complexity theory.

Acknowledgment
This work was supported by the EPSRC, project EP/E001394/1 "Fixed-Parameter Algorithms and Satisfiability."

References

[ABdR+18] A. Atserias, I. Bonacina, S. F. de Rezende, M. Lauria, J. Nordström, and A. A. Razborov. Clique is hard on average for regular resolution. In *STOC*, pages 866–877. ACM, 2018.

[ABGL14] C. Ansótegui, M. L. Bonet, J. Giráldez-Cru, and J. Levy. The fractal dimension of SAT formulas. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*, volume 8562 of *LNCS*, pages 107–121. Springer-Verlag, 2014.

[ACP87] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987.

[AGK+11] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo. Solving MAX-r-SAT above a tight lower bound. *Algorithmica*, 61(3):638–655, 2011.

[AL86] R. Aharoni and N. Linial. Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *Journal of Combinatorial Theory, Series A*, 43(2):196–204, 1986.

[BBE17] M. Bannach, S. Berndt, and T. Ehlers. Jdrasil: A modular library for computing tree decompositions. In C. S. Iliopoulos, S. P. Pissis, S. J. Puglisi, and R. Raman, editors, *16th International Symposium on Experimental Algorithms, SEA 2017, June 21-23, 2017, London, UK*, volume 75 of *LIPIcs*, pages 28:1–28:21. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[BCH90] E. Boros, Y. Crama, and P. L. Hammer. Polynomial-time inference of all valid implications for Horn and related formulae. *Ann. Math. Artif. Intell.*, 1:21–32, 1990.

[BDD+16] H. L. Bodlaender, P. l. G. n. s. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM Journal of Computing*, 45(2):317–378, 2016.

[BDP03] F. Bacchus, S. Dalmao, and T. Pitassi. Algorithms and complexity results for #SAT and Bayesian inference. In *Proc. 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, pages 340–351. IEEE Computer Society, 2003.

[BDP09] F. Bacchus, S. Dalmao, and T. Pitassi. Solving #SAT and Bayesian inference with backtracking search. *J. Artif. Intell. Res.*, 34:391–442, 2009.

[BFK+06] H. L. Bodlaender, F. V. Fomin, A. M. C. A. Koster, D. Kratsch, and D. M. Thilikos. On exact algorithms for treewidth. In *Proc. 14th Annual European Symposium on Algorithms (ESA'06)*, volume 4168 of *LNCS*, pages 672–683. Springer-Verlag, 2006.

[BGLR12] O. Beyersdorff, N. Galesi, M. Lauria, and A. A. Razborov. Parameterized bounded-depth frege is not optimal. *ACM Trans. Comput. Theory*, 4(3):7:1–7:16, 2012.

[BHKK07] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Fourier meets möbius: fast subset convolution. In *STOC*, pages 67–74. ACM, 2007.

[BHS94] E. Boros, P. L. Hammer, and X. Sun. Recognition of q-Horn formulae in linear time. *Discrete Applied Mathematics*, 55(1):1–13, 1994.

[BJ14] J. Berg and M. Järvisalo. SAT-based approaches to treewidth computation: An evaluation. In *Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI'14*, pages 328–335, Limassol, Cyprus, November 2014. IEEE Computer Soc.

[Bjö18] A. Björklund. Personal Communication, 2018.

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

[BM07] J. Buresh-Oppenheim and D. G. Mitchell. Minimum 2CNF resolution refutations in polynomial time. In J. Marques-Silva and K. A. Sakallah, editors, *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31, 2007, Proceedings*, volume 4501 of *LNCS*, pages 300–313. Springer-Verlag, 2007.

[Bod93] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2):1–22, 1993.

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

[Bod05] H. L. Bodlaender. Discovering treewidth. In *Proc. 31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'05)*, volume 3381 of *LNCS*, pages 1–16. Springer-Verlag, 2005.

[BS17] S. Bova and S. Szeider. Circuit treewidth, sentential decision, and query compilation. In E. Sallinger, J. V. den Bussche, and F. Geerts, editors, *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 233–246. ACM, 2017.

[BW04] F. Bacchus and J. Winter. Effective preprocessing with hyper-resolution and equality reduction. In *Proc. 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03), Selected and Revised Papers*, volume 2919 of *LNCS*, pages 341–355. Springer-Verlag, 2004.

[CCDF97] L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied Logic*, 84(1):119–138, 1997.

[CEH97] Y. Crama, O. Ekin, and P. L. Hammer. Variable and term removal from Boolean formulae. *Discrete Applied Mathematics*, 75(3):217–230, 1997.

[CER93] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46(2):218–270, 1993.

[Ces06] M. Cesati. Compendium of parameterized problems. `http://cesati.sprg.uniroma2.it/research/compendium/`, September 2006.

[CFK+13] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Texts in Computer Science. Springer, 2013.

[CJG08] D. Cohen, P. Jeavons, and M. Gyssens. A unified theory of structural tractability for constraint satisfaction problems. *Journal of Computer and System Sciences*, 74(5):721–743, 2008.

[CKJ01] J. Chen, I. A. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001.

[CKX10] J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.

[CMR01] B. Courcelle, J. A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1-2):23–52, 2001.

[CO00] B. Courcelle and S. Olariu. Upper bounds to the clique-width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

[Cou88] B. Courcelle. The monadic second-order logic of graphs: Definable sets of finite graphs. In *Proc. 14th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'88)*, volume 344 of *LNCS*, pages 30–53. Springer-Verlag, 1988.

[CXW17] J. Chen, C. Xu, and J. Wang. Dealing with 4-variables by resolution: an improved MaxSAT algorithm. *Theoretical Computer Science*, 670:33–44, 2017.

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[DF13] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[DGS07] B. N. Dilkina, C. P. Gomes, and A. Sabharwal. Tradeoffs in the complexity of backdoor detection. In *Proc. 13th International Conference on Principles and Practice of Constraint Programming (CP'07)*, volume 4741 of *LNCS*, pages 256–270. Springer-Verlag, 2007.

[dHKS17] R. de Haan, I. Kanj, and S. Szeider. On the parameterized complexity of finding small unsatisfiable subsets of CNF formulas and CSP instances. *ACM Trans. Comput. Log.*, 18(3):Art. 21, 46, 2017.

[dHS17] R. de Haan and S. Szeider. Parameterized complexity classes beyond Para-NP. *Journal of Computer and System Sciences*, 87:16–57, 2017.

[dHS19] R. de Haan and S. Szeider. Compendium of parameterized problems at higher levels of the polynomial hierarchy. *MDPI Algorithms*, 12(9):1–28, 2019.

[DMS11] S. S. Dantchev, B. Martin, and S. Szeider. Parameterized proof complexity. *Computational Complexity*, 20(1):51–85, 2011.

[Fel03] M. R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in fpt. In H. L. Bodlaender, editor, *Graph-Theoretic Concepts in Computer Science (WG 2003)*, volume 2880 of *LNCS*, pages 1–12. Springer-Verlag, 2003.

[FG06] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-

Verlag, 2006.

[FKS02] H. Fleischner, O. Kullmann, and S. Szeider. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theoretical Computer Science*, 289(1):503–516, 2002.

[FKT04] F. V. Fomin, D. Kratsch, and I. Todinca. Exact (exponential) algorithms for treewidth and minimum fill-in. In *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *LNCS*, pages 568–580. Springer-Verlag, 2004.

[FLM+15] F. V. Fomin, D. Lokshtanov, N. Misra, M. S. Ramanujan, and S. Saurabh. Solving d-sat via backdoors to small treewidth. In P. Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 630–641. SIAM, 2015.

[FLS17] J. K. Fichte, N. Lodha, and S. Szeider. SAT-based local improvement for finding tree decompositions of small width. In S. Gaspers and T. Walsh, editors, *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10491 of *LNCS*, pages 401–411. Springer-Verlag, 2017.

[FMR08] E. Fischer, J. A. Makowsky, and E. V. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discrete Applied Mathematics*, 156(4):511–529, 2008. DOI: 10.1016/j.dam.2006.06.020.

[FS15] J. K. Fichte and S. Szeider. Backdoors to tractable answer set programming. *Artificial Intelligence*, 220:64–103, March 2015.

[FSW06] M. R. Fellows, S. Szeider, and G. Wrightson. On finding short resolution refutations and small unsatisfiable subsets. *Theoretical Computer Science*, 351(3):351–359, 2006.

[FV03] J. Franco and A. Van Gelder. A perspective on certain polynomial time solvable classes of satisfiability. *Discrete Applied Mathematics*, 125(2):177–214, 2003.

[GHO13] R. Ganian, P. Hlinený, and J. Obdrzálek. Better algorithms for satisfiability problems for formulas of bounded rank-width. *Fund. Inform.*, 123(1):59–76, 2013.

[GJ79] M. R. Garey and D. R. Johnson. *Computers and Intractability.* W. H. Freeman and Company, 1979.

[GKSS18] R. Ganian, E. J. Kim, F. Slivovsky, and S. Szeider. Sum-of-products with default values: Algorithms and complexity results. In M. Alamaniotis, editor, *Proceedings of ICTAI 2018, the 30th IEEE International Conference on Tools with Artificial Intelligence*, 2018. To appear.

[GLS02] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64(3):579–627, 2002.

[GM06] M. Grohe and D. Marx. Constraint solving via fractional edge covers. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'06)*, pages 289–298. ACM Press, 2006.

[GMO+17] S. Gaspers, N. Misra, S. Ordyniak, S. Szeider, and S. Zivny. Backdoors into heterogeneous classes of SAT and CSP. *Journal of Computer and System Sciences*, 85:38–56, 2017.

[GN07] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(2):31–45, 2007.

[GOR+16] S. Gaspers, S. Ordyniak, M. S. Ramanujan, S. Saurabh, and S. Szeider. Backdoors to q-Horn. *Algorithmica*, 74(1):540–557, 2016.

[GRS17a] R. Ganian, M. S. Ramanujan, and S. Szeider. Backdoor treewidth for SAT. In S. Gaspers and T. Walsh, editors, *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10491 of *LNCS*, pages 20–37. Springer-Verlag, 2017.

[GRS17b] R. Ganian, M. S. Ramanujan, and S. Szeider. Combining Treewidth and Backdoors for CSP. In H. Vollmer and B. Vallee, editors, *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:17, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[GS08] G. Gottlob and S. Szeider. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems. *The Computer Journal*, 51(3):303–325, 2008. DOI: 10.1093/comjnl/bxm056.

[GS12a] S. Gaspers and S. Szeider. Backdoors to acyclic SAT. In A. Czumaj, K. Mehlhorn, A. M. Pitts, and R. Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, volume 7391 of *LNCS*, pages 363–374. Springer-Verlag, 2012.

[GS12b] S. Gaspers and S. Szeider. Backdoors to satisfaction. In H. L. Bodlaender, R. Downey, F. V. Fomin, and D. Marx, editors, *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *LNCS*, pages 287–317. Springer-Verlag, 2012.

[GS13] S. Gaspers and S. Szeider. Strong backdoors to bounded treewidth SAT. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 489–498. IEEE Computer Society, 2013.

[GS14] S. Gaspers and S. Szeider. Guarantees and limits of preprocessing in constraint satisfaction and reasoning. *Artificial Intelligence*, 216:1–19, 2014.

[GS15] R. Ganian and S. Szeider. Community structure inspired algorithms for SAT and #SAT. In M. Heule and S. Weaver, editors, *18th International Conference on Theory and Applications of Satisfiability Testing (SAT 2015), September 24-27, 2015, Austin, Texas*, number 9340 in LNCS, pages 223–237. Springer-Verlag, 2015.

[GS17] R. Ganian and S. Szeider. New width parameters for model counting. In S. Gaspers and T. Walsh, editors, *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference,*

Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings, volume 10491 of *LNCS*, pages 38–52. Springer-Verlag, 2017.

[GSS02] G. Gottlob, F. Scarcello, and M. Sideri. Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artificial Intelligence*, 138(1-2):55–86, 2002.

[HK73] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal of Computing*, 2(4):225–231, 1973.

[HS15] M. Heule and S. Szeider. A SAT approach to clique-width. *ACM Trans. Comput. Log.*, 16(3):24, 2015.

[IPZ01] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity. *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[Iwa89] K. Iwama. CNF-satisfiability test by counting and polynomial average time. *SIAM Journal of Computing*, 18(2):385–391, 1989.

[Joh73] D. S. Johnson. Approximation algorithms for combinatorial problems. In *Fifth Annual ACM Symposium on Theory of Computing (Austin, Tex., 1973)*, pages 38–49. Assoc. Comput. Mach., New York, 1973.

[Joh90] D. S. Johnson. A catalog of complexity classes. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2, pages 67–161. Elsevier Science Publishers, 1990.

[JS12] A. K. Jha and D. Suciu. On the tractability of query compilation and bounded treewidth. In A. Deutsch, editor, *15th International Conference on Database Theory, ICDT '12, Berlin, Germany, March 26-29, 2012*, pages 249–261. ACM, 2012.

[KBL99] H. Kleine Büning and T. Lettmann. *Propositional logic: Deduction and algorithms*. Cambridge University Press, 1999.

[KBvH01a] A. M. C. A. Koster, H. L. Bodlaender, and S. P. M. van Hoesel. Treewidth: Computational experiments. Technical Report ZIB 01-38, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2001.

[KBvH01b] A. M. C. A. Koster, H. L. Bodlaender, and S. P. M. van Hoesel. Treewidth: Computational experiments. *Electronic Notes in Discrete Mathematics*, 8:54–57, 2001.

[Kle00] H. Kleine Büning. On subclasses of minimal unsatisfiable formulas. *Discrete Applied Mathematics*, 107(1–3):83–98, 2000.

[Klo94] T. Kloks. *Treewidth: Computations and Approximations*. Springer-Verlag, 1994.

[KV00] P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61(2):302–332, 2000.

[KZ01] H. Kleine Büning and X. Zhao. Satisfiable formulas closed under replacement. *Electronic Notes in Discrete Mathematics*, 9:48–58, 2001.

[Lew78] H. R. Lewis. Renaming a set of clauses as a Horn set. *Journal of the ACM*, 25(1):134–135, 1978.

[LP86] L. Lovász and M. D. Plummer. *Matching Theory*. Number 29 in Annals of Discrete Mathematics. North-Holland Publishing Co., Ams-

terdam, 1986.

[MFS16] N. Mull, D. J. Fremont, and S. A. Seshia. On the hardness of SAT with community structure. In *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *LNCS*, pages 141–159. Springer, 2016.

[MR99] M. Mahajan and V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, 31(2):335–354, 1999.

[MRS06] M. Mahajan, V. Raman, and S. Sikdar. Parameterizing MAX SNP problems above guaranteed values. In *Proc. 2nd International Workshop on Parameterized and Exact Computation (IWPEC'06)*, volume 4169 of *LNCS*, pages 38–49. Springer-Verlag, 2006.

[MS10] D. Marx and I. Schlotter. Parameterized complexity and local search approaches for the stable marriage problem with ties. *Algorithmica*, 58(1):170–187, 2010.

[MS11] D. Marx and I. Schlotter. Stable assignment with couples: parameterized complexity and local search. *Discrete Optim.*, 8(1):25–40, 2011.

[New03] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[New06] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[NG04] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, February 2004.

[NGF+14] Z. Newsham, V. Ganesh, S. Fischmeister, G. Audemard, and L. Simon. Impact of community structure on SAT solver performance. In C. Sinz and U. Egly, editors, *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8561 of *LNCS*, pages 252–268. Springer-Verlag, 2014.

[Nie06] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Number 31 in Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[NRS04] N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *Proc. 7th International Conference on Theory and Applications of Satisfiability Testing (SAT'04)*, pages 96–103. Informal Proceedings, 2004.

[NRS07] N. Nishimura, P. Ragde, and S. Szeider. Solving #SAT using vertex covers. *Acta Informatica*, 44(7-8):509–523, 2007.

[OPS13] S. Ordyniak, D. Paulusma, and S. Szeider. Satisfiability of acyclic and almost acyclic CNF formulas. *Theoretical Computer Science*, 481:85–99, 2013.

[OSS21] S. Ordyniak, A. Schidler, and S. Szeider. Backdoor DNFs. Technical Report AC-TR-21, Algorithms and Complexity Group, TU Wien,

2021.

[Oum05] S.-i. Oum. Approximating rank-width and clique-width quickly. In *Proc. 31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG'05)*, volume 3787 of *LNCS*, pages 49–58. Springer-Verlag, 2005.

[Par16] A. Parlak. A SAT approach to clique-width of a digraph and an application on model counting problem. Master's thesis, TU Wien, Algorithms and Complexity Group, 2016. Supervised by Stefan Szeider.

[PRS13] A. Pfandler, S. Rümmele, and S. Szeider. Backdoors to abduction. In *Proceedings of IJCAI 2013, the 23th International Joint Conference on Artificial Intelligence, August 3-9, 2013, Beijing, China*, 2013.

[PSS16] D. Paulusma, F. Slivovsky, and S. Szeider. Model counting for CNF formulas of bounded modular treewidth. *Algorithmica*, 76(1):168–194, 2016.

[PW88] C. H. Papadimitriou and D. Wolfe. The complexity of facets resolved. *Journal of Computer and System Sciences*, 37(1):2–13, 1988.

[Ree92] B. A. Reed. Finding approximate separators and computing tree-width quickly. In *Proc. 24th Annual ACM symposium on Theory of Computing (STOC'92)*, pages 221–228. ACM Press, 1992.

[RO08] I. Razgon and B. O'Sullivan. Almost 2-SAT is fixed-parameter tractable. In *Proc. 35th International Colloquium on Automata, Languages and Programming (ICALP'08), Track A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *LNCS*, pages 551–562. Springer-Verlag, 2008.

[Rot96] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

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

[RS91] N. Robertson and P. D. Seymour. Graph minors X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.

[RS17] M. S. Ramanujan and S. Saurabh. Linear-time parameterized algorithms via skew-symmetric multicuts. *ACM Transactions on Algorithms*, 13(4):Art. 46, 25, 2017.

[RTL76] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal of Computing*, 5(2):266–283, 1976.

[Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, Calif., 1978)*, pages 216–226. ACM, 1978.

[See91] D. Seese. The structure of the models of decidable monadic theories of graphs. *Annals of Pure and Applied Logic*, 53(2):169–195, 1991.

[Spi03] J. P. Spinrad. *Efficient Graph Representations*. Fields Institute Monographs. AMS, 2003.

[SS08] M. Samer and S. Szeider. Backdoor trees. In *Proc. 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, pages 363–368. AAAI

Press, 2008.

[SS09] M. Samer and S. Szeider. Backdoor sets of quantified Boolean formulas. *Journal of Automated Reasoning*, 42(1):77–97, 2009.

[SS10a] M. Samer and S. Szeider. Algorithms for propositional model counting. *Journal of Discrete Algorithms*, 8(1):50–64, 2010.

[SS10b] M. Samer and S. Szeider. Constraint satisfaction with bounded treewidth revisited. *Journal of Computer and System Sciences*, 76(2):103–114, 2010.

[SS13] F. Slivovsky and S. Szeider. Model counting for formulas of bounded clique-width. In L. Cai, S. Cheng, and T. W. Lam, editors, *Algorithms and Computation - 24th International Symposium, ISAAC 2013, Hong Kong, China, December 16-18, 2013, Proceedings*, volume 8283 of *LNCS*, pages 677–687. Springer-Verlag, 2013.

[SS20] F. Slivovsky and S. Szeider. A faster algorithm for propositional model counting parameterized by incidence treewidth. In L. Pulina and M. Seidl, editors, *Proceedings of SAT 2020, The 23rd International Conference on Theory and Applications of Satisfiability Testing*, volume 12178 of *LNCS*, pages 267–276. Springer-Verlag, 2020.

[STV15] S. H. Sæther, J. A. Telle, and M. Vatshelle. Solving #sat and MAXSAT by dynamic programming. *J. Artif. Intell. Res.*, 54:59–82, 2015.

[SV09] M. Samer and H. Veith. Encoding treewidth into SAT. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT 2009*, volume 5584 of *LNCS*, pages 45–50. Springer-Verlag, 2009.

[Sze04a] S. Szeider. Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. *Journal of Computer and System Sciences*, 69(4):656–674, 2004.

[Sze04b] S. Szeider. On fixed-parameter tractable parameterizations of SAT. In *Proc. 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03), Selected and Revised Papers*, volume 2919 of *LNCS*, pages 188–202. Springer-Verlag, 2004.

[Sze05] S. Szeider. Backdoor sets for DLL subsolvers. *Journal of Automated Reasoning*, 35(1-3):73–88, 2005. Reprinted as Chapter 4 of the book "SAT 2005 – Satisfiability Research in the Year 2005", edited by E. Giunchiglia and T. Walsh, Springer-Verlag, 2006.

[Sze08] S. Szeider. Matched formulas and backdoor sets. *Journal on Satisfiability, Boolean Modeling and Computation*, 6:1–12, 2008.

[TY84] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal of Computing*, 13(3):566–579, 1984.

[Vil06] Y. Villanger. Improved exponential-time algorithms for treewidth and minimum fill-in. In *Proc. 7th Latin American Symposium on Theoretical Informatics (LATIN'06)*, volume 3887 of *LNCS*, pages 800–811. Springer-Verlag, 2006.

[Wah17] M. Wahlström. *Algorithms, measures and upper bounds for satis-*

fiability and related problems. PhD thesis, Linköpings Universitet, 2017.

[WGS03] R. Williams, C. P. Gomes, and B. Selman. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search. In *Proc. 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*, pages 222–230. Informal Proceedings, 2003.

[Woe03] G. J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Proc. 5th International Workshop on Combinatorial Optimization (AUSSOIS'01) — "Eureka, You Shrink!", Revised Papers*, volume 2570 of *LNCS*, pages 185–208. Springer-Verlag, 2003.

[Yan81] M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. 7th International Conference on Very Large Data Bases (VLDB'81)*, pages 81–94. IEEE Computer Society, 1981.

[ZPWL13] W. Zhang, G. Pan, Z. Wu, and S. Li. Online community detection for large complex networks. In F. Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. IJCAI/AAAI, 2013.