Handbook of Satisfiability
Armin Biere, Marijn Heule, Hans van Maaren and Toby Walsch
IOS Press, 2008

Chapter 13

Fixed-Parameter Tractability

Marko Samer and Stefan Szeider

13.1. 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 decade, 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. Actually, 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 13.2). 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 under a certain perspective. We call such a mapping π a *satisfiability parameter* or a *parameterization* of the satisfiability problem.

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 $\mathcal{C}_0^\pi \subseteq \mathcal{C}_1^\pi \subseteq \mathcal{C}_2^\pi \subseteq \cdots$ 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 larger we make k (thus the more general the class $\mathcal{C}_k^\pi$), the worse we expect the performance guarantee for the polynomial-time algorithm for solving in-

stances in $\mathcal{C}_k^\pi$—in other words, we expect a certain *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. 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—just 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.

Hence, it is an interesting 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 important branch of algorithm design and analysis in both applied and theoretical areas of computer science; hundreds of research papers and three monographs have been published so far on the subject [DF99, FG06, Nie06]. 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 it can be solved by a fixed-parameter algorithm.

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 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 13.2. 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 parame-

terized 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 interesting 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.

In the next section we review the main concepts of parameterized complexity theory. In Section 13.3 we provide some preliminaries and introduce the basic notions of parameterized satisfiability; we also discuss parameterized Max-SAT and related parameterized optimization problems. Then three sections are devoted to satisfiability parameters of different flavors: in Section 13.4 we consider parameters based on backdoor sets relative to a polynomial-time base class; in Section 13.5 we consider parameters that measure the "tree-likeness" of instances; in Section 13.6 we consider further parameters including one that is based on graph matchings. Finally, we conclude in Section 13.7.

13.2. 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].

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 it can be solved by a fixed-parameter algorithm, 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 by means of the *vertex cover* problem parameterized by the solution size. This is the best-studied problem in parameterized complexity with a long history of improvements [CKJ01]. Let us state the parameterized vertex cover problem in the following form which is typical for parameterized complexity.

> **VC**
> *Instance:* A graph $G = (V, E)$ and a non-negative integer k.
> *Parameter:* k.
> *Question:* Is there 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 parameter but simply as 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 important technique, which shrinks the size of the given problem instance by means of (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, 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 again the **VC** problem 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 [CKX06]. The algorithm is based on more sophisticated kernelization rules and achieves a running time of $\mathcal{O}^*(1.273^k)$. Further information on kernelization can be found in Guo and Niedermeier's survey [GN07].

Next we turn our attention to fixed-parameter *intractability*, 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.
Question: Is there 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 Chapter 12 for an in-depth treatment of the ETH.

In fact, **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. 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.
Question: Is there 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 $q = 3$, Niedermeier and Rossmanith [NR03] developed a fixed-parameter algorithm with running time $\mathcal{O}^*(2.270^k)$.

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 W[1] $\subseteq$ W[2] $\subseteq$ W[3] $\subseteq \cdots \subseteq$ W[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 [DMS07a].

In recent years a further complexity class, M[1], has been subject of intensive research. The class can be defined as the class of parameterized decision problems that can be reduced to the following problem by fpt-reductions.

log-VC
Instance: A graph $G = (V, E)$ and a non-negative integer k.
Parameter: $\lfloor k/\log(|V| + |E|)\rfloor$.
Question: Does G have a vertex cover of size at most k?

It is known that FPT $\subseteq$ M[1] $\subseteq$ W[1]; the assumption FPT $\neq$ M[1] is actually equivalent to the Exponential Time Hypothesis [FG06].

Finally, let us mention the class XP consisting of those parameterized decision problems that can be solved in polynomial time if the parameter is considered as a *constant* (i.e., instances (I, k) can be solved in time $\mathcal{O}(\|I\|^{f(k)})$ for a computable function f). FPT $\neq$ XP is provably true [DF99, FG06]. Together with the classes of the weft hierarchy, we have the following chain of inclusions:

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

13.3. Parameterized SAT

In this section we first explain our terminology for CNF formulas and truth assignments, then we discuss parameterized optimization problems that are related to satisfiability, such as Max-SAT, where the solution size is considered as parameter. In the third part of this section we develop the framework for parameterized SAT decision where the parameter represents structural information of the instances. This framework will be used throughout the following sections.

13.3.1. 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 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*.

13.3.2. 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 due to Chen and Kanj [CK04] and runs in time $\mathcal{O}^*(1.3695^k)$.

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. Further results on parameterizations above a guaranteed value can be found in a recent paper 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 13.4.1.

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 [DMS07b]. 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].

13.3.3. 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 13.3.1), 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 parameterized problem.

SAT(π)
Instance: A formula F and a non-negative integer k such that $\pi(F) \leq k$.
Parameter: k.
Question: Is F satisfiable?

The problem is formulated as a *promise problem*, the promise being $\pi(F) \leq k$. In general, we need to verify the promise. This verification can be stated as the following parameterized problem.

VER(π)
Instance: A formula F and a non-negative integer k.
Parameter: k.
Question: Is $\pi(F) \leq k$?

Note that the verification of the promise often includes the computation of a certain *witness* for $\pi(F) \leq k$ in form of an auxiliary structure, which then is provided as additional input to the algorithm that solves **SAT**(π).

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. If π strictly dominates π' we also say that π is *more general* than π'. Obviously, dominance and strict dominance 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 13.3.1. *If π dominates π', then there is an fpt-reduction from* **SAT**(π') *to* **SAT**(π).

Thus, if **SAT**(π) is fixed-parameter tractable and π dominates π', then **SAT**(π') 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 **SAT**(π) and **VER**(π) are fixed-parameter tractable. We conclude this section with three trivial examples of satisfiability parameters.

Example 13.3.1. Let $\mathbf{n}(F)$ denote the number of variables of a formula F. The verification problem **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 **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 13.3.2. 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 **SAT**($\mathbf{ml}$) is not fixed-parameter tractable unless P = NP. So **SAT**($\mathbf{ml}$) is probably not even in XP. □

Example 13.3.3. 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 13.3.1 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 13.3.1) since one can easily find formulas where, say, unit propagation eliminates an arbitrarily large number of variables. □

13.4. 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 $\mathcal{C}$ by means of 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 denote the size of a smallest strong $\mathcal{C}$-backdoor set of F by $\mathbf{b}_{\mathcal{C}}(F)$. For the sake of completeness we also introduce the notion of *weak*

backdoor sets though it is less relevant for our considerations. B 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 $\mathcal{C}$.

Example 13.4.1. 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 $\mathcal{C}$.

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 notion of *backdoor trees* [SS08] makes 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.

13.4.1. 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 13.4.1 (Nishimura, Ragde, and Szeider [NRS04])**.** *For $\mathcal{C} \in \{\text{HORN}, \text{2CNF}\}$ 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, since $F[\tau] \subseteq F - B$ for every $\tau : B \to \{0,1\}$.

Lemma 13.4.2. *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 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 13.4.2 holds.

Lemma 13.4.3 ([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 13.4.2. Consider the formula F of Example 13.4.1 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 can be used to find a strong HORN-backdoor set. In particular, the algorithm of Chen et al. [CKX06] solves $\mathbf{VER}(\mathbf{b}_{\text{HORN}})$ in time $\mathcal{O}^*(1.273^k)$.

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 13.4.3, the latter is the case if and only if B is a strong 2CNF-backdoor set of F. Thus, Niedermeier and Rossmanith's algorithm for **3-HS** [NR03] solves $\mathbf{VER}(\mathbf{b}_{\text{2CNF}})$ in time $\mathcal{O}^*(2.270^k)$.

A generalization of backdoor sets, in particular HORN- and 2CNF-backdoor sets, to quantified Boolean formulas has recently been proposed by taking the variable dependencies caused by the quantifications into account [SS07b].

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, $\mathbf{VER}(\mathbf{b}_{\text{UP}})$ 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.

Theorem 13.4.4 (Szeider [Sze05])**.** *For* $\mathcal{C} \in \{\mathrm{PL}, \mathrm{UP}, \mathrm{UP} + \mathrm{PL}\}$ *the problem* $\mathbf{VER}(\mathbf{b}_{\mathcal{C}})$ *is* W[P]*-complete.*

In view of 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 do 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]. Actually, 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 13.4.2 and the following example.

Example 13.4.3. 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 $\mathbf{db}_{\mathrm{RHORN}}$. □

Until recently the parameterized complexities of $\mathbf{VER}(\mathbf{b}_{\mathrm{RHORN}})$ and $\mathbf{VER}(\mathbf{db}_{\mathrm{RHORN}})$ were open. Razgon and O'Sullivan [RO08] have shown that Max-2-SAT parameterized by the number of clauses that remain unsatisfied is fixed-parameter tractable. This problem can be shown to be equivalent to $\mathbf{VER}(\mathbf{db}_{\mathrm{RHORN}})$ under fpt-reductions. On the other hand, there is an fpt-reduction from CLIQUE to $\mathbf{VER}(\mathbf{b}_{\mathrm{RHORN}})$, which shows W[1]-hardness of the latter problem.

Dilkina, Gomes, and Sabharwal [DGS07] suggest to strengthen 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 13.4.5 (Szeider [Sze08])**.** *For* $\mathcal{C} \in \{\text{HORN}^{\{\}}, \text{2CNF}^{\{\}}, \text{RHORN}^{\{\}}\}$ *the problem* $\mathbf{VER}(\mathbf{b}_\mathcal{C})$ *is* W[1]*-hard.*

13.4.2. Hitting Formulas and Clustering-Width

Iwama [Iwa89] observed that one can 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 $\tau : \text{var}(F) \to \{0,1\}$ 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 [NRS07] consider the parameterized problem of detecting CLU-backdoor sets.

Theorem 13.4.6. $\mathbf{VER}(\mathbf{b}_{\text{CLU}})$ *is* W[2]*-hard but* $\mathbf{VER}(\mathbf{db}_{\text{CLU}})$ *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 by means of 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, $x \in \text{var}(C_1 \cap C_2)$, and $y \in \text{var}(C_1 \setminus C_2)$;
2. F contains three clauses C_1, C_2, C_3 such that C_1 and C_3 do not clash, $x \in \text{var}((C_1 \setminus C_3) \cap \overline{C_2})$, and $y \in \text{var}((C_3 \setminus C_1) \cap \overline{C_2})$.

Vertex covers of obstruction graphs are closely related to backdoor sets: every deletion CLU-backdoor set of a CNF formula F is a vertex cover of the obstruction graph of F. Conversely, every vertex cover of the obstruction graph of a

CNF formula F is a strong CLU-backdoor set of F. The satisfiability parameter $\mathbf{clu}(F)$, the *clustering-width*, is defined as the size of a smallest vertex cover of the obstruction graph of F. The clustering width is more general than $\mathbf{db}_{\mathrm{CLU}}(F)$ and less general than $\mathbf{b}_{\mathrm{CLU}}(F)$.

Example 13.4.4. 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 rv, rx, ry, st, su, tw, uw, vw, xw, yw. The set $B = \{r,s,w\}$ forms a vertex cover of the obstruction graph; there is no vertex cover of size two. Hence F has clustering-width 3. 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 $B' = \{w,s\}$. □

In view of the fixed-parameter tractability of **VC**, we obtain the following theorem. Since **clu** is more general than $\mathbf{db}_{\mathrm{CLU}}$, it implies the second part of Theorem 13.4.6.

Theorem 13.4.7 (Nishimura et al. [NRS07])**.** *The problems* $\mathbf{SAT}(\mathbf{clu})$ *and* $\mathbf{VER}(\mathbf{clu})$ *are fixed-parameter tractable.*

Actually, 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 $\#(F)$ by means of 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.

13.5. 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 actually do have low treewidth [Bod93]. For taking the treewidth of a CNF formula one needs to represent the structure of a formula as a graph. 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 $x,y \in \mathrm{var}(C)$ for some $C \in F$.

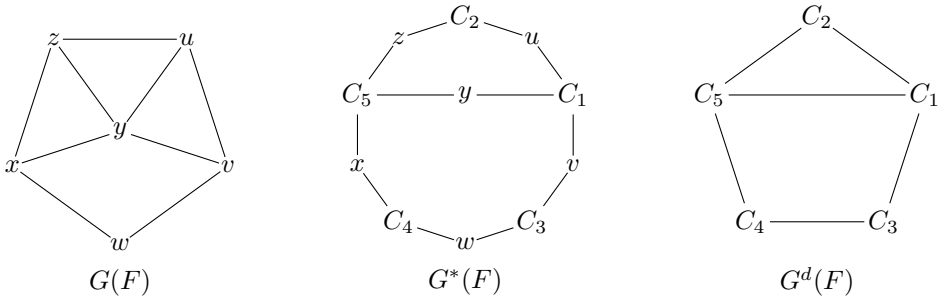

Figure 13.1. Graphs associated with the CNF formula $F = \{C_1, \ldots, C_5\}$ with $C_1 = \{u, \neg v, \neg y\}$, $C_2 = \{\neg u, z\}$, $C_3 = \{v, \neg w\}$, $C_4 = \{w, \neg x\}$, $C_5 = \{x, y, \neg z\}$; the primal graph $G(F)$, the incidence graph $G^*(F)$, and the dual graph $G^d(F)$.

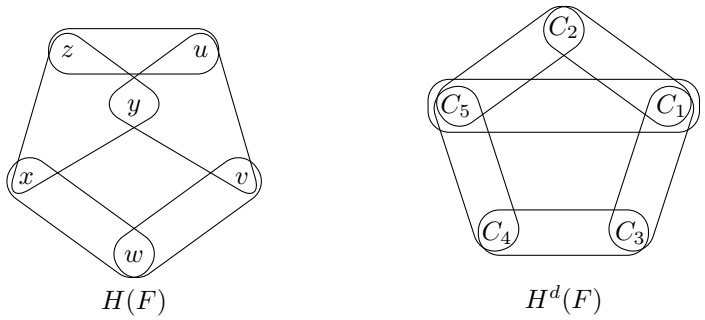

Figure 13.2. The hypergraph $H(F)$ and the dual hypergraph $H^d(F)$ associated with the CNF formula F of Figure 13.1.

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 $x \in \mathrm{var}(C)$. 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 $x \in \mathrm{var}(F)$. Figure 13.1 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 $\mathrm{var}(C)$ of variables 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 13.2 for examples.

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

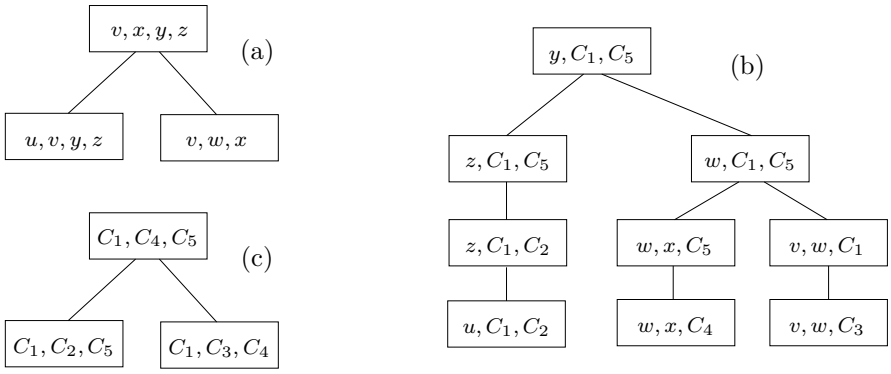

Figure 13.3. Tree decompositions of the primal graph (a), the incidence graph (b), and the dual graph (c)

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 is. 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 CNF formulas 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 13.1 are shown in Figure 13.3. 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 13.5.1. 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, our aim is to construct a tree decomposition of width as small as possible; in the optimal case a tree decomposition of minimal width, the treewidth, can be found.

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 bounded 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 13.5.2 we will review algorithms for constructing tree decompositions.

13.5.1. 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 13.5.1. *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 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 $\mathbf{SAT}(\mathbf{tw})$ in time $\mathcal{O}^*(4^k)$. By using an improved algorithm, we can decide $\mathbf{SAT}(\mathbf{tw})$ in time $\mathcal{O}^*(2^k)$ [SS07a].

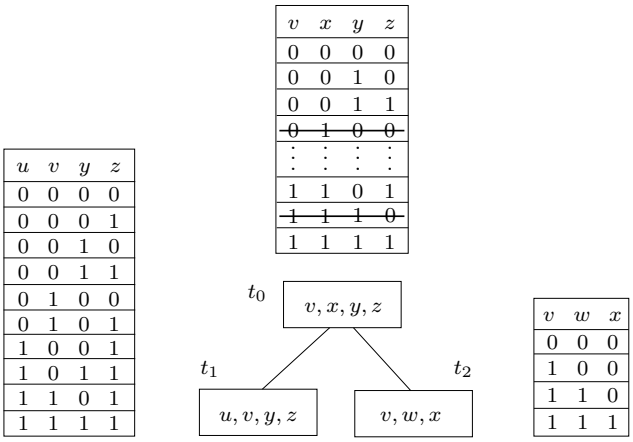

Figure 13.4. A fixed-parameter algorithm for **SAT**(**tw**)

Example 13.5.2. Consider the primal graph of the CNF formula F in Figure 13.1 and its tree decomposition in Figure 13.3(a). The tables associated with each tree node are shown in Figure 13.4: 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. □

The following result is stronger than Theorem 13.5.1.

Theorem 13.5.2. *The problem* **SAT**(**tw***) *is fixed-parameter tractable.*

Since incidence treewidth strictly dominates primal treewidth and dual treewidth as already mentioned above, this result implies both Theorem 13.5.1 and fixed-parameter tractability of **SAT**(**tw**^{d}). The situation is different for "generalized satisfiability" also known as "Boolean constraint satisfaction" where clauses are replaced by Boolean relations. Generalized satisfiability is fixed-parameter tractable for the parameter primal treewidth but W[1]-hard for the parameter incidence treewidth [SS06].

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 yield 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 **SAT**(**tw***) [CMR01, GS08, Sze04b]. However, the algorithms obtained via the generic constructions are impractical.

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 [SS07a] presented practical fixed-parameter algorithms for the more general problem **#SAT**($\mathbf{tw}^*$) of counting the number of models. This trivially implies Theorem 13.5.2, 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 encoding $\alpha(x)$ 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**($\mathbf{tw}^*$) in time $\mathcal{O}^*(4^k)$.

Example 13.5.3. Consider the incidence graph of the CNF formula F in Figure 13.1 and its tree decomposition in Figure 13.3(b). A fragment of the corresponding nice tree decomposition and the tables associated with each tree node are shown in Figure 13.5. 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 [SS07a]. 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

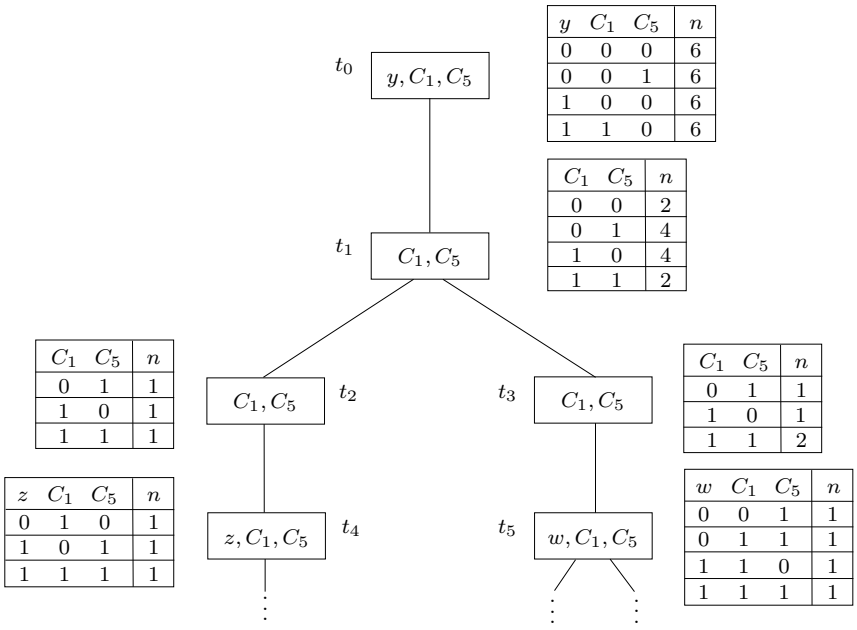

Figure 13.5. A fixed-parameter algorithm for **#SAT(tw*)**

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 $X_{t_0} = \text{var}(F)$ 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 *branch-width* of the hypergraph $H(F)$. Similar to tree decompositions, branch decompositions and the corresponding branch-width were introduced by Robertson and Seymour in their Graph Minors Project. It is well known that a graph with treewidth k has branch-width at most $k + 1$ and that a graph with branch-width k has treewidth at most $3k/2$ [RS91]. 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 to truncate the search-tree of the DPLL procedure. The resulting algorithm runs in time $2^{\mathcal{O}(k)}n^c$, where k is the branch-width, n is the number of variables, and c is a constant.

13.5.2. 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 exist also in the case of bounded treewidth. In fact, 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. One such guarantee is the *relative performance ratio* ρ (or *performance ratio*, for short), which guarantees that the tree decomposition found by the approximation algorithm has width at most ρ times the treewidth.The first such algorithm is due to Bodlaender et al. [BGHK95]; it runs in polynomial time and has a performance ratio of $\mathcal{O}(\log n)$. Bouchitté et al. [BKMT04] and Amir [Ami01] improved this algorithm to a performance ratio of $\mathcal{O}(\log k)$. It is an open problem whether there exists a polynomial time approximation algorithm for treewidth with constant performance ratio.

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 are 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 on tree decomposition algorithms.

13.5.3. Beyond Treewidth

Beside treewidth, other (more general) 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 of a graph 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. Courcelle and Olariu [CO00] have shown that a graph with treewidth k has clique-width at most $2^{k+1}+1$. On the other hand, every clique on $n \geq 2$ vertices has clique-width 2 but treewidth $n-1$. Thus, clique-width strictly dominates treewidth.

For a CNF formula F, we define the clique-width **cwd** of F as the clique-width of its incidence graph $G^*(F)$. The following results show that clique-width allows fixed-parameter tractable SAT decision; the known algorithms however are impractical and of theoretical interest only. The fixed-parameter tractability of **SAT**(**cwd**) follows from monadic second-order theory [CMR01] similarly as in the case of treewidth. For **VER**(**cwd**), there exists a fixed-parameter approximation algorithm: Oum [Oum05] developed an algorithm that, for constant k, runs in time $\mathcal{O}(n^4)$ and decides whether the clique-width of the input graph is at most k and, if so, constructs a decomposition of width at most $2^{3k+2}-1$.

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 [SS07a]. In particular, let F be a CNF formula and x be 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 $C' = \{x\}$. Since every variable of F' occurs in the single clause C, the associated hypergraph $H(F')$ is acyclic [GLS02]. Thus, hypertree-width, spread-cut width, and fractional hypertree-width of $H(F')$ are one. However, satisfiability of F' is NP-hard since F' is satisfiable if and only if F is satisfiable. A similar construction can be made with respect to the dual hypergraph $H^d(F)$ [SS07a].

13.6. 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 to it via the matching.

Example 13.6.1. 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 **VER**(**md**) can be solved in polynomial time, since a largest matching in a bipartite graph can be found in polynomial time by means of 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 clauses. 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 **SAT**(**md**) are of relevance. Fleischner, Kullmann, and Szeider [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 **SAT**(**md**) and **MU**(**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 13.6.1 (Szeider [Sze04a])**.** *The problems* **SAT**(**md**) *and* **MU**(**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 its child 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 **SAT**(**md**).

Let r be a non-negative 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. Szeider [Sze08] has shown the following hardness result.

Theorem 13.6.2. *The problems* $\mathbf{VER}(\mathbf{b}_{\mathcal{M}_r})$ *and* $\mathbf{VER}(\mathbf{db}_{\mathcal{M}_r})$ *are* W[2]*-hard for every* $r \geq 0$.

13.7. 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. Similar results are shown for a related parameter, the *number of resolution steps* required to refute a given CNF formula.

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 [DMS07a] have studied the proof complexity of resolution for such "parameterized contradictions."

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

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

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

[Ami01] Eyal Amir. Efficient approximations for triangulation of minimum treewidth. In *Proc. 17th Conference on Uncertainty in Artificial Intelligence (UAI'01)*, pages 7–15. Morgan Kaufmann, 2001.

[BDP03] Fahiem Bacchus, Shannon Dalmao, and Toniann 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.

[BFK+06] Hans L. Bodlaender, Fedor V. Fomin, Arie M. C. A. Koster, Dieter Kratsch, and Dimitrios 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.

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

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

[BKMT04] Vincent Bouchitté, Dieter Kratsch, Heiko Müller, and Ioan Todinca. On treewidth approximations. *Discrete Applied Mathematics*, 136(2-3):183–196, 2004.

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

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

[Bod05] Hans 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.

[BW04] Fahiem Bacchus and Jonathan 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] Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied Logic*, 84(1):119–138, 1997.

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

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

[Ces06] Marco Cesati. Compendium of parameterized problems. URL: `http://bravo.ce.uniroma2.it/home/cesati/research/compendium.pdf`, September 2006.

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

[CK04] Jianer Chen and Iyad A. Kanj. Improved exact algorithms for MAX-SAT. *Discrete Applied Mathematics*, 142(1-3):17–27, 2004.

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

[CKX06] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved parameterized upper bounds for vertex cover. In *Proc. 31st International Symposium on Mathematical Foundations of Computer Science (MFCS'06)*, volume 4162 of *LNCS*, pages 238–249. Springer-Verlag, 2006.

[CMR01] Bruno Courcelle, Johann A. Makowsky, and Udi 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] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique-width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

[Cou88] Bruno 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.

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

[DGS07] Bistra N. Dilkina, Carla P. Gomes, and Ashish 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.

[DMS07a] Stefan Dantchev, Barnaby Martin, and Stefan Szeider. Parameterized proof complexity. In *Proc. 48th Annual Symposium on Foundations of Computer Science (FOCS'07)*, pages 150–160. IEEE Computer Society, 2007.

[DMS07b] Stefan Dantchev, Barnaby Martin, and Stefan Szeider. Parameterized proof complexity: A complexity gap for parameterized tree-like resolution. Technical Report TR07-001, *Electronic Colloquium on Computational Complexity* (ECCC), January 2007.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Springer-Verlag, 2006.

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

[FKT04] Fedor V. Fomin, Dieter Kratsch, and Ioan 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.

[FMR08] Eldar Fischer, Johann A. Makowsky, and Elena V. Ravve. Count-

ing 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.

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

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

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

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

[GM06] Martin Grohe and Dániel 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.

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

[GS08] Georg Gottlob and Stefan 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.

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

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

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

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

[Joh90] David 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.

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

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

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

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

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

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

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

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

[LP86] László Lovász and Michael D. Plummer. *Matching Theory*. Number 29 in Annals of Discrete Mathematics. North-Holland Publishing Co., Amsterdam, 1986.

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

[MRS06] Meena Mahajan, Venkatesh Raman, and Somnath 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.

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

[NR03] Rolf Niedermeier and Peter Rossmanith. An efficient fixed-parameter algorithm for 3-hitting set. *Journal of Discrete Algorithms*, 1(1):89–102, 2003.

[NRS04] Naomi Nishimura, Prabhakar Ragde, and Stefan 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] Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Solving #SAT using vertex covers. *Acta Informatica*, 44(7-8):509–523, 2007.

[Oum05] Sang-il 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.

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

[Ree92] Bruce 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] Igor Razgon and Barry 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] Daniel Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

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

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

[SS06] Marko Samer and Stefan Szeider. Constraint satisfaction with bounded treewidth revisited. Submitted for publication. Preliminary version published in *Proc. 12th International Conference on Principles and Practice of Constraint Programming (CP'06)*, volume 4204 of *LNCS*, pages 499–513. Springer-Verlag, 2006.

[SS07a] Marko Samer and Stefan Szeider. Algorithms for propositional model counting. In *Proc. 14th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'07)*, volume 4790 of *LNCS*, pages 484–498. Springer-Verlag, 2007.

[SS07b] Marko Samer and Stefan Szeider. Backdoor sets of quantified Boolean formulas. In *Proc. 10th International Conference on Theory and Applications of Satisfiability Testing (SAT'07)*, volume 4501 of *LNCS*, pages 230–243. Springer-Verlag, 2007.

[SS08] Marko Samer and Stefan Szeider. Backdoor trees. In *Proc. 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, pages 363–368. AAAI Press, 2008.

[Sze04a] Stefan 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] Stefan 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] Stefan 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] Stefan Szeider. Matched formulas and backdoor sets. *Journal on Satisfiability, Boolean Modeling and Computation*, 6:1–12, 2008.

[TY84] Robert E. Tarjan and Mihalis 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] Yngve 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.

[WGS03] Ryan Williams, Carla P. Gomes, and Bart Selman. On the con-

nections 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] Gerhard 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] Mihalis 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.