Constraint Satisfaction with Bounded Treewidth Revisited [⋆]

Marko Samer and Stefan Szeider

Department of Computer Science
Durham University, UK

Abstract

The constraint satisfaction problem can be solved in polynomial time for instances where certain parameters (e.g., the treewidth of primal graphs) are bounded. However, there is a trade-off between generality and performance: larger bounds on the parameters yield worse time complexities. It is desirable to pay for more generality only by a constant factor in the running time, not by a larger degree of the polynomial. Algorithms with such a uniform polynomial time complexity are known as fixed-parameter algorithms.

In this paper we determine whether or not fixed-parameter algorithms for constraint satisfaction exist, considering all possible combinations of the following parameters: the treewidth of primal graphs, the treewidth of dual graphs, the treewidth of incidence graphs, the domain size, the maximum arity of constraints, and the maximum size of overlaps of constraint scopes. The negative cases are subject to the complexity theoretic assumption FPT $\neq$ W[1] which is the parameterized analog to P $\neq$ NP. For the positive cases we provide an effective fixed-parameter algorithm which is based on dynamic programming on "nice" tree decompositions.

Key words: Constraint satisfaction, parameterized complexity, treewidth

[⋆] Research supported by the EPSRC project EP/E001394/1 and the Austrian Science Funds (FWF) project P17222-N04. Preliminary version published in the Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP'06).
Authors' address: Department of Computer Science, Durham University, Science Labs, South Road, Durham DH1 3LE, England, UK. E-Mail: {marko.samer,stefan.szeider}@durham.ac.uk

1 Introduction

An instance of the constraint satisfaction problem (CSP) consists of a set of variables that range over a domain of values together with a set of constraints that allow certain combinations of values for certain sets of variables. The question is whether one can instantiate the variables in such a way that all constraints are simultaneously satisfied; in that case the instance is called consistent or satisfiable. Constraint satisfaction provides a general framework which allows direct structure-preserving encodings of numerous problems that arise in practice.

Although constraint satisfaction is NP-complete in general, many efforts have been made to identify restricted problems that can be solved in polynomial time. Such restrictions can either limit the constraints used in the instance [6] or limit the overall structure of the instance, i.e., how variables and constraints interact in the instance [8]. In this paper we focus on the latter form of restrictions which are also referred to as "structural restrictions." Structural restrictions are usually formulated in terms of certain graphs and hypergraphs that are associated with a constraint satisfaction instance as described in the following.

The *primal graph* has the variables as its vertices; two variables are joined by an edge if they occur together in the scope of a constraint. The *dual graph* has the constraints as its vertices; two constraints are joined by an edge if their scopes have variables in common. The *incidence graph* is a bipartite graph and has both the variables and the constraints as its vertices; a variable and a constraint are joined by an edge if the variable occurs in the scope of the constraint. Finally, the *constraint hypergraph* is a hypergraph whose vertices are the variables and whose hyperedges are the constraint scopes.

Fundamental classes of tractable instances are obtained if the associated (hyper)graphs are *acyclic* with respect to certain notions of acyclicity. Acyclicity can be generalized by means of (hyper)graph decomposition techniques which give rise to "width" parameters that measure how far an instance deviates from being acyclic. Freuder [12] and Dechter and Pearl [9] observed that constraint satisfaction is polynomial-time solvable if

- the *treewidth of primal graphs*, **tw**,

is bounded by a constant. The graph parameter treewidth, introduced by Robertson and Seymour in their Graph Minors Project, has become a very popular object of study as many NP-hard graph problems are polynomial-time solvable for graphs of bounded treewidth; we define treewidth in Section 2.2. In subsequent years several further structural parameters have been considered, such as

- the *treewidth of dual graphs*, $\mathbf{tw}^d$,
- the *treewidth of incidence graphs*, $\mathbf{tw}^*$,

and various width parameters on constraint hypergraphs, including

- the *(generalized) hypertree width*, (**g**)**hw**, (Gottlob, Leone, and Scarcello [15]),
- the *spread-cut width*, **scw**, (Cohen, Jeavons, and Gyssens [7]), and
- the *fractional hypertree width*, **fhw**, (Grohe and Marx [18]).

Considering constraint satisfaction instances where the width parameter under consideration is bounded by some fixed integer k gives rise to a class W_k of tractable instances. The larger k gets, the larger is the resulting tractable class W_k. However, for getting larger and larger tractable classes one has to pay by longer running times. A fundamental question is the *trade-off between generality and performance*. A typical time complexity of algorithms known from the literature are of the form

$$\mathcal{O}(\|I\|^{\mathcal{O}(f(k))}) \tag{1}$$

for instances I belonging to the class W_k; here $\|I\|$ denotes the input size of I and $f(k)$ denotes a slowly growing function. Such a running time is polynomial when k is considered as a constant. However, since k appears in the exponent, such algorithms become impractical—even if k is small—when large instances are considered. It is significantly better if instances I of the class W_k can be solved in time

$$\mathcal{O}(f(k)\,\|I\|^{\mathcal{O}(1)}) \tag{2}$$

where f is an arbitrary (possibly exponential) computable function. In that case the order of the polynomial does not depend on k, and so considering larger and larger classes does not increase the order of the polynomial. Thus, it is a significant objective to classify the trade-off between generality and performance of a width parameter under consideration: whether the parameter allows algorithms of type (1) or of type (2).

1.1 Parameterized Complexity

The framework of *parameterized complexity* provides the adequate concepts and tools for studying the above question. Parameterized complexity was initiated by Downey and Fellows in the late 1980s and has become an important branch of algorithm design and analysis; hundreds of research papers have been published in that area [10,11,21]. It turned out that the distinction between tractability of type (2) and tractability of type (1) is a robust indication of problem hardness.

A *fixed parameter algorithm* is an algorithm that achieves a running time of type (2) for instances I and parameter k. A parameterized problem is *fixed-parameter tractable* if it can be solved by a fixed-parameter algorithm. FPT denotes the class of all fixed-parameter tractable decision problems.

Parameterized complexity offers a *completeness theory*, similar to the theory of NP-completeness, that allows the accumulation of strong theoretical evidence that a parameterized problem is *not* fixed-parameter tractable. This completeness theory is based on the *weft hierarchy* of complexity classes $\mathrm{W}[1], \mathrm{W}[2], \ldots, \mathrm{W}[\mathrm{P}]$. Each class is the equivalence class of certain parameterized satisfiability problems under *fpt-reductions* (for instance, the canonical W[1]-complete problem asks whether a given 3SAT instance can be satisfied by setting at most k variables to *true*). Let Π and Π' be two parameterized problems. An *fpt-reduction R from Π to Π'* is a many-to-one transformation from Π to Π', such that (i) $(I,k) \in \Pi$ if and only if $(I',k') \in \Pi'$ with $k' \leq g(k)$ for a fixed computable function g and (ii) R is of complexity $\mathcal{O}(f(k)\, \|I\|^{\mathcal{O}(1)})$ for a computable function f. The class XP consists of parameterized problems which can be solved in polynomial time if the parameter is considered as a constant. The above classes form the chain

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

where all inclusions are assumed to be proper. A parameterized analog of Cook's Theorem [10] as well as the Exponential Time Hypothesis [11,19] give strong evidence to assume that $\mathrm{FPT} \neq \mathrm{W}[1]$. It is known that $\mathrm{FPT} \neq \mathrm{XP}$ [10]; hence the term "parameterized tractable" (which is sometimes used to indicate membership in XP [7]) must be carefully distinguished from "fixed-parameter tractable." Although XP contains problems which are very unlikely to be fixed-parameter tractable, it is often a significant improvement to show that a problem belongs to this class, in contrast to, e.g., k-SAT which is NP-complete for every constant $k \geq 3$.

The following parameterized clique-problem is W[1]-complete [10]; this problem is the basis for the hardness results considered in the sequel.

CLIQUE
Instance: A graph G and a non-negative integer k.
Parameter: k.
Question: Does G contain a clique on k vertices?

1.2 Parameterized Constraint Satisfaction

We consider any computable function p that assigns to a constraint satisfaction instance I a non-negative integer $p(I)$ as a *constraint satisfaction parameter*; constraint satisfaction instances are formally defined in Section 2.1. For a finite set $\{p_1, \ldots, p_r\}$ of constraint satisfaction parameters we consider the following generic parameterized problem:

CSP$(p_1, \ldots, p_r)$
Instance: A constraint satisfaction instance I and non-negative integers $k_1, \ldots, k_r$ with $p_1(I) \leq k_1, \ldots, p_r(I) \leq k_r$.
Parameters: $k_1, \ldots, k_r$.
Question: Is I consistent?

Slightly abusing notation, we will also write **CSP**(S) for a set S of parameters, assuming an arbitrary but fixed ordering of the parameters in S. We write $\mathbf{CSP}_{\mathbf{boole}}(S)$ to denote **CSP**(S) with the *Boolean domain* $\{0, 1\}$, and $\mathbf{CSP}_{\mathbf{bin}}(S)$ to denote **CSP**(S) where all constraints have arity at most 2.

Note that we formulate this problem as a "promise problem" in the sense that for solving the problem we do not need to verify the assumption $p_1(I) \leq k_1, \ldots, p_r(I) \leq k_r$. However, unless otherwise stated, for all cases considered in the sequel where **CSP**$(p_1, \ldots, p_r)$ is fixed-parameter tractable, also the verification of the assumption $p_1(I) \leq k_1, \ldots, p_r(I) \leq k_r$ is fixed-parameter tractable. For a constraint satisfaction instance I we have the basic parameters

- the number of variables, **vars**,
- the size of the domain, **dom**,
- the largest size of a constraint scope, **arity**, and
- the largest number of variables that occur in the *overlap* of the scopes of two distinct constraints, **ovl**.

If we parameterize by the domain size, then we have obviously an NP-complete problem, since, e.g., 3-colorability can be expressed as a constraint satisfaction problem with constant domain. Thus **CSP(dom)** is not fixed-parameter tractable unless P = NP. On the other hand, if we parameterize by the number of variables *and* the domain size, i.e., **CSP(vars, dom)**, then we have a trivially fixed-parameter tractable problem: we can decide the consistency of an instance I by checking all $\mathbf{dom}(I)^{\mathbf{vars}(I)}$ possible assignments. However, without the parameter **dom** we get **CSP(vars)**, a W[1]-complete problem [22].

Gottlob, Scarcello, and Sideri [17] have determined the parameterized complexity of constraint satisfaction with respect to the treewidth of primal graphs: **CSP(tw, dom)** is fixed-parameter tractable, and **CSP(tw)** is W[1]-hard. The parameterized complexity of constraint satisfaction with respect to other structural parameters like treewidth of dual graphs, treewidth of incidence graphs, and the more general width parameters defined in terms of constraint hypergraphs remained open. In this paper we determine exactly those combinations of parameters from **tw**, $\mathbf{tw}^d$, $\mathbf{tw}^*$, **dom**, **arity**, and **ovl** that render constraint satisfaction fixed-parameter tractable.

To this end we introduce the notion of *domination*. Let S and $S' = \{p'_1, p'_2, \ldots, p'_{r'}\}$ be two finite sets of constraint satisfaction parameters. S *dominates* S' if for every $p \in S$ there exists an r'-ary computable function f that is monotonically increasing in each argument such that for every constraint satisfaction instance I we have $p(I) \leq f(p'_1(I), p'_2(I), \ldots, p'_{r'}(I))$. See Lemma 2 for examples that illustrate this notion (if S or S' is a singleton, we omit the braces to improve readability). It is easy to see that whenever S dominates S', then fixed-parameter tractability of $\mathbf{CSP}(S)$ implies fixed-parameter tractability of $\mathbf{CSP}(S')$, and W[1]-hardness of $\mathbf{CSP}(S')$ implies W[1]-hardness of $\mathbf{CSP}(S)$ (see Lemma 1).

1.3 Results

We obtain the following classification result (see also the diagram in Figure 1 and the discussion in Section 3).

Theorem 1 (Classification Theorem) *Let* $S \subseteq \{\mathbf{tw}, \mathbf{tw}^d, \mathbf{tw}^*, \mathbf{dom}, \mathbf{arity}, \mathbf{ovl}\}$.

(1) If $\{\mathbf{tw}^*, \mathbf{dom}, \mathbf{ovl}\}$ *dominates* S*, then* $\mathbf{CSP}(S)$ *is fixed-parameter tractable.*
(2) If $\{\mathbf{tw}^*, \mathbf{dom}, \mathbf{ovl}\}$ *does not dominate* S*, then* $\mathbf{CSP}(S)$ *is not fixed-parameter tractable unless* FPT = W[1].

The complexity theoretic assumption FPT $\neq$ W[1] is discussed in Section 1.1. We establish the fixed-parameter tractability results by a dynamic programming algorithm. The established upper bounds to its worst-case running time show that the algorithm is feasible in practice. Let us remark that many constraint satisfaction instances appearing in industry have bounded overlap. For example, the `Adder`, `Bridge`, and `NewSystem` instances from *DaimlerChrysler* have by construction an overlap bounded by 2 [13].

We extend the fixed-parameter tractability result of the Classification Theorem to the additional parameters **diff** and **equiv**; definitions are given in Section 6. We show that he problems $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom}, \mathbf{diff})$ and $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom}, \mathbf{equiv})$ are fixed-parameter tractable.

The notion of domination allows us to extend the W[1]-hardness results of the Classification Theorem to all parameters that are more general than the treewidth of incidence graphs. In particular, we obtain the following corollary to Theorem 1.

Corollary 1 *The problems* $\mathbf{CSP}(p, \mathbf{dom})$ *and* $\mathbf{CSP}_{\mathbf{boole}}(p)$ *are* W[1]*-hard if* p *is any of the parameters treewidth of incidence graphs, hypertree width, generalized hypertree width, spread-cut width, and fractional hypertree width.*

Recently, Gottlob *et al.* [14] have shown that the problem of deciding whether a given hypergraph has (generalized) hypertree width at most k, is W[2]-hard with respect to the parameter k. We note that this result does not imply W[1]-hardness of $\mathbf{CSP}(\mathbf{hw}, \mathbf{dom})$ (respectively $\mathbf{CSP}(\mathbf{ghw}, \mathbf{dom})$), since it is possible to design algorithms for constraint satisfaction instances of bounded (generalized) hypertree width that avoid the decomposition step. Chen and Dalmau [5] have recently proposed such an algorithm, which, however, is not a fixed-parameter algorithm.

Our results indicate a somewhat surprising difference between Boolean constraint satisfaction and propositional satisfiability (SAT). A SAT instance is a set of clauses, representing a propositional formula in conjunctive normal form. The question is whether the instance is satisfiable. Primal, dual, and incidence graphs and the corresponding treewidth parameters $\mathbf{tw}$, $\mathbf{tw}^d$, and $\mathbf{tw}^*$ can be defined for SAT similarly as for constraint satisfaction [25], as well as the parameterized decision problem $\mathbf{SAT}(p)$ for a parameter p. In contrast to the W[1]-hardness of $\mathbf{CSP}_{\mathbf{boole}}(\mathbf{tw}^*)$, as established in Corollary 1, the problem $\mathbf{SAT}(\mathbf{tw}^*)$ is fixed-parameter tractable. This holds also true for $\mathbf{SAT}(\mathbf{tw}^d)$ and $\mathbf{SAT}(\mathbf{tw})$ since $\mathbf{tw}^*$ dominates $\mathbf{tw}^d$ and $\mathbf{tw}$. Szeider [25] proved the fixed-parameter tractability of $\mathbf{SAT}(\mathbf{tw}^*)$ by using a general result for Monadic Second Order (MSO) logic on graphs; Samer and Szeider [24] developed a dynamic programming algorithm for this problem.

2 Preliminaries

2.1 Constraint Satisfaction

Formally, a constraint satisfaction instance I is a triple (V, D, F), where V is a finite set of *variables*, D is a finite set of *domain values*, and F is a finite set of *constraints*. Each constraint in F is a pair (S, R), where S, the *constraint scope*, is a sequence of distinct variables of V, and R, the *constraint relation*, is a relation over D whose arity matches the length of S. We write $var(C)$ for the set of variables that occur in the scope of a constraint C and $rel(C)$ for the relation of C. An *assignment* is a mapping $\tau : X \to D$ defined on some set X of variables. Let $C = ((x_1, \dots, x_n), R)$ be a constraint and $\tau : X \to D$ an assignment. We define

$$C[\tau] = \{\, (d_1, \dots, d_n) \in R : x_i \notin X \text{ or } \tau(x_i) = d_i,\ 1 \leq i \leq n \,\}.$$

Thus, $C[\tau]$ contains those tuples of R that do not disagree with τ at some position. Similarly, for a set $\mathcal{T}$ of assignments and a constraint C we define

$$C[\mathcal{T}] = \bigcup_{\tau \in \mathcal{T}} C[\tau].$$

An assignment $\tau : X \to D$ is *consistent* with a constraint C if $C[\tau] \neq \emptyset$. An assignment $\tau : X \to D$ *satisfies* a constraint C if $\mathit{var}(C) \subseteq X$ and τ is consistent with C. An assignment satisfies a constraint satisfaction instance I if it satisfies all constraints of I. The instance I is *consistent* (or *satisfiable*) if it is satisfied by some assignment. The *constraint satisfaction problem* **CSP** is the problem of deciding whether a given constraint satisfaction instance is satisfiable.

2.2 Tree Decompositions

Let G be a graph, let T be a tree, and let χ be a labelling of the vertices of T by sets of vertices of G. We refer to the vertices of T as "nodes" to avoid confusion with the vertices of G, and we call the sets $\chi(t)$ "bags." The pair (T, χ) is a *tree decomposition* of G if the following three conditions hold:

(1) For every vertex v of G there exists a node t of T such that $v \in \chi(t)$.
(2) For every edge vw of G there exists a node t of T such that $v, w \in \chi(t)$.
(3) For any three nodes t_1, t_2, t_3 of T, if t_2 lies on the unique path from t_1 to t_3, then $\chi(t_1) \cap \chi(t_3) \subseteq \chi(t_2)$ ("Connectedness Condition").

The *width* of a tree decomposition (T, χ) is defined as the maximum $|\chi(t)| - 1$ over all nodes t of T. The *treewidth* $\mathrm{tw}(G)$ of a graph G is the minimum width over all its tree decompositions.

As shown by Bodlaender [2], there exists for every k a linear time algorithm that checks whether a given graph has treewidth at most k and, if so, outputs a tree decomposition of minimum width. Bodlaender's algorithm does not seem feasible to implement [4]. However, there are several other known fixed-parameter algorithms that are feasible. For example, Reed's algorithm [23] runs in time $\mathcal{O}(|V| \log |V|)$ and decides either that the treewidth of a given graph $G = (V, E)$ exceeds k, or outputs a tree decomposition of width at most $4k$, for any fixed k. The algorithm produces tree decompositions with $\mathcal{O}(|V|)$ many nodes.

Let (T, χ) be a tree decomposition of a graph G and let r be a node of T. The triple (T, χ, r) is a *nice tree decomposition* of G if the following three conditions hold; here we consider $T = (V(T), E(T))$ as a tree rooted at r.

(1) Every node of T has at most two children.

(2) If a node t of T has two children t_1 and t_2, then $\chi(t) = \chi(t_1) = \chi(t_2)$; in that case we call t a *join node*.
(3) If a node t of T has exactly one child t', then exactly one of the following prevails:
   (a) $|\chi(t)| = |\chi(t')|+1$ and $\chi(t') \subset \chi(t)$; in that case we call t an *introduce node.*
   (b) $|\chi(t)| = |\chi(t')| - 1$ and $\chi(t) \subset \chi(t')$; in that case we call t a *forget node.*

Let (T, χ, r) be a nice tree decomposition of a graph G. For each node t of T let T_t denote the subtree of T rooted at t. Furthermore, let V_t denote the set of vertices of G that occur in the bags of nodes of T_t; i.e., $V_t = \bigcup_{t' \in V(T_t)} \chi(t')$.

It is well known (and easy to see) that for any constant k, given a tree decomposition of a graph $G = (V, E)$ of width k and with $\mathcal{O}(|V|)$ nodes, one can construct in linear time a nice tree decomposition of G with $\mathcal{O}(|V|)$ nodes and width at most k [4].

3 The Domination Lattice

Lemma 1 *Let S and S' be two sets of constraint satisfaction parameters such that S dominates S'. Then there is an fpt-reduction from* $\mathbf{CSP}(S')$ *to* $\mathbf{CSP}(S)$*. In particular, fixed-parameter tractability of* $\mathbf{CSP}(S)$ *implies fixed-parameter tractability of* $\mathbf{CSP}(S')$*, and* W[1]*-hardness of* $\mathbf{CSP}(S')$ *implies* W[1]*-hardness of* $\mathbf{CSP}(S)$*.*

PROOF. Let $S = \{p_1, \ldots, p_r\}$ and $S' = \{p'_1, \ldots, p'_{r'}\}$ and assume that S dominates S'. By definition, for every $i = 1, \ldots, r$ there exists an r'-ary computable function f_i that is monotonically increasing in each argument such that for every constraint satisfaction instance I we have $p_i(I) \leq f_i(p'_1(I), p'_2(I), \ldots, p'_{r'}(I))$. Consider an instance $(I, k'_1, \ldots, k'_{r'})$ of $\mathbf{CSP}(S')$; i.e., we have $p'_i(I) \leq k'_i$ for all $1 \leq i \leq r'$. We put $k_i = f_i(k'_1, k'_2, \ldots, k'_{r'})$ for all $1 \leq i \leq r$. Since f_i is monotonically increasing in each argument, we have $p_i(I) \leq f_i(p'_1(I), p'_2(I), \ldots, p'_{r'}(I)) \leq f_i(k'_1, k'_2, \ldots, k'_{r'}) = k_i$. Hence $(I, k_1, \ldots, k_r)$ is an instance of $\mathbf{CSP}(S)$. Whence we have indeed an fpt-reduction from $\mathbf{CSP}(S')$ to $\mathbf{CSP}(S)$. The second part of the lemma is a direct consequence of the first part. $\square$

Lemma 2

(1) If $S \subseteq S'$, then S dominates S'.
(2) **ovl** *dominates* **arity**.
(3) **arity** *dominates* **tw**.
(4) $\mathbf{tw}^*$ *dominates* **tw**.
(5) $\mathbf{tw}^*$ *dominates* $\mathbf{tw}^d$.
(6) **tw** *dominates* $\{\mathbf{tw}^*, \mathbf{arity}\}$.

PROOF. Parts 1 and 2 are obvious. Part 3 is also easy to see since a constraint of arity r yields a clique on r vertices in the primal graph; it is well known that if a graph G contains a clique with r vertices, then $\mathrm{tw}(G) \geq r - 1$ [3]. Part 4 follows from the inequality $\mathbf{tw}^*(I) \leq \mathbf{tw}(I) + 1$ shown by Kolaitis and Vardi [20]. A symmetric argument gives $\mathbf{tw}^*(I) \leq \mathbf{tw}^d(I) + 1$, hence Part 5 holds as well. Part 6 follows by the inequality $\mathbf{tw}(I) \leq \mathbf{tw}^*(I)(\mathbf{arity}(I) - 1)$ which is also due to Kolaitis and Vardi [20]. $\square$

We note that parts 2–5 of the above lemma are *strict* in the sense that p dominates q but q does not dominate p.

Let $\mathcal{S} = \{\mathbf{tw}, \mathbf{tw}^d, \mathbf{tw}^*, \mathbf{dom}, \mathbf{arity}, \mathbf{ovl}\}$. The following arguments will make it easier to classify $\mathbf{CSP}(S)$ for subsets S of $\mathcal{S}$.

First we note that whenever $S \cap \{\mathbf{tw}, \mathbf{tw}^d, \mathbf{tw}^*\} = \emptyset$, then S dominates $\{\mathbf{dom}, \mathbf{arity}, \mathbf{ovl}\}$. However, $\mathbf{CSP}(\mathbf{dom}, \mathbf{arity}, \mathbf{ovl})$ is not fixed-parameter tractable unless P = NP, since graph 3-colorability can be expressed as a constraint satisfaction problem with constant **dom**, **arity**, and **ovl**.

Second, if a set S dominates a proper subset S' of S, then $\mathbf{CSP}(S)$ and $\mathbf{CSP}(S')$ are of the same parameterized complexity; this follows by Lemmas 1 and 2(1); in this case we can disregard S. For example, we can disregard the set $\{\mathbf{tw}, \mathbf{arity}\}$ since it dominates $\{\mathbf{tw}\}$ by Lemma 2(3). Similarly we can disregard the set $\{\mathbf{tw}, \mathbf{ovl}\}$ since it dominates $\{\mathbf{tw}, \mathbf{arity}\}$ by Lemma 2(2) and so it dominates $\{\mathbf{tw}\}$ as well.

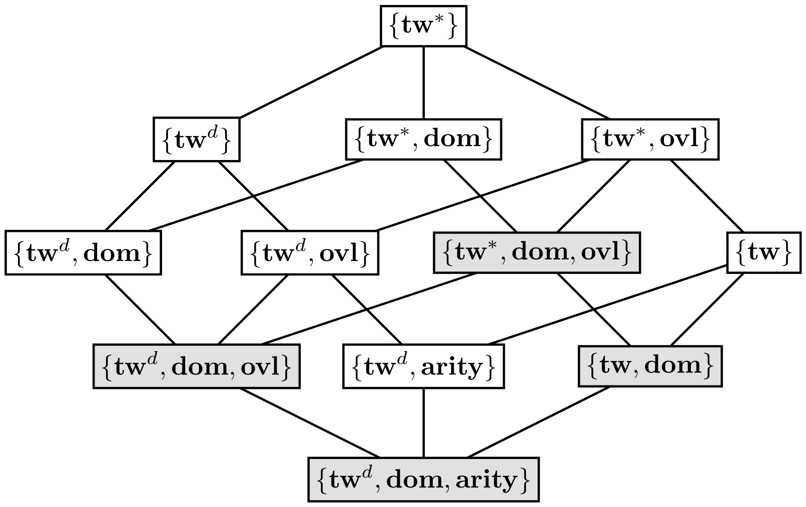
Third, by Lemma 2(4 and 6), every set $S \cup \{\mathbf{tw}^*, \mathbf{arity}\}$ has the same parameterized complexity as the set $S \cup \{\mathbf{tw}\}$.

Hence, in order to establish Theorem 1, it suffices to classify the parameterized complexity of $\mathbf{CSP}(S)$ for the following twelve sets $S \subseteq \mathcal{S}$.

$$
\begin{array}{llll}
\{\mathbf{tw}\}, & \{\mathbf{tw}, \mathbf{dom}\}, & \{\mathbf{tw}^d\}, & \{\mathbf{tw}^d, \mathbf{dom}\}, \\
\{\mathbf{tw}^*\}, & \{\mathbf{tw}^*, \mathbf{dom}\}, & \{\mathbf{tw}^d, \mathbf{ovl}\}, & \{\mathbf{tw}^d, \mathbf{dom}, \mathbf{ovl}\}, \\
\{\mathbf{tw}^*, \mathbf{ovl}\}, & \{\mathbf{tw}^*, \mathbf{dom}, \mathbf{ovl}\}, & \{\mathbf{tw}^d, \mathbf{arity}\}, & \{\mathbf{tw}^d, \mathbf{dom}, \mathbf{arity}\}.
\end{array}
$$

Figure 1 shows the relationships among all twelve sets as implied by Lemma 2: a set S dominates a set S' if and only if there is a path running downwards from S to S'; in fact, it can be easily shown that whenever one of the twelve sets dominates another, it strictly dominates the other set.

The sets S for which $\mathbf{CSP}(S)$ is fixed-parameter tractable according to Theorem 1 are indicated in the diagram by shaded boxes. In view of the relation-



$\{\mathbf{tw}^*\}$
$\{\mathbf{tw}^d\}$
$\{\mathbf{tw}^*, \mathbf{dom}\}$
$\{\mathbf{tw}^*, \mathbf{ovl}\}$
$\{\mathbf{tw}^d, \mathbf{dom}\}$
$\{\mathbf{tw}^d, \mathbf{ovl}\}$
$\{\mathbf{tw}^*, \mathbf{dom}, \mathbf{ovl}\}$
$\{\mathbf{tw}\}$
$\{\mathbf{tw}^d, \mathbf{dom}, \mathbf{ovl}\}$
$\{\mathbf{tw}^d, \mathbf{arity}\}$
$\{\mathbf{tw}, \mathbf{dom}\}$
$\{\mathbf{tw}^d, \mathbf{dom}, \mathbf{arity}\}$

Fig. 1. Domination lattice

ships between the sets, Theorem 1 is established if we show (i) W[1]-hardness of $\mathbf{CSP}(\mathbf{tw}^d, \mathbf{dom})$, (ii) W[1]-hardness of $\mathbf{CSP}(\mathbf{tw}^d, \mathbf{arity})$, and (iii) fixed-parameter tractability of $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom}, \mathbf{ovl})$.

4 Proof of the W[1]-hardness Results

We give an fpt-reduction from **CLIQUE** to $\mathbf{CSP_{boole}}(\mathbf{tw}^d)$. To this aim, consider an instance of **CLIQUE**, i.e., a graph $G = (\{v_1, \dots, v_n\}, E)$ and an integer k. We construct a Boolean constraint satisfaction instance $I = (\{\, x_{i,j} : 1 \leq i \leq k, 1 \leq j \leq n \,\}, \{0,1\}, F)$ such that I is consistent if and only if there exists a clique of size k in G.

First we construct the relation $R \subseteq \{0,1\}^{2n}$ that encodes the edges of G using Boolean values 0 and 1 as follows: for each edge $v_p v_q$ of G, $1 \leq p < q \leq n$, we add to R the $2n$-tuple

$$(t_{p,1}, \dots, t_{p,n}, t_{q,1}, \dots, t_{q,n})$$

where $t_{p,i} = 1$ if and only if $p = i$, and $t_{q,i} = 1$ if and only if $q = i$, $1 \leq i \leq n$. We let F be the set of constraints

$$C_{i,j} = ((x_{i,1}, \dots, x_{i,n}, x_{j,1}, \dots, x_{j,n}), R)$$

for $1 \leq i < j \leq k$. It is easy to verify that G contains a clique on k vertices if and only if I is consistent. The reduction can be carried out in polynomial time.

Next we construct a trivial tree decomposition of the dual graph of I by creating a single tree node and putting all constraints into its bag. Since

$|F| = \binom{k}{2}$, the width of our decomposition is $\binom{k}{2} - 1$. Hence, $\mathbf{tw}^d(I) \leq \binom{k}{2} - 1$. Thus we have indeed an fpt-reduction from **CLIQUE** to $\mathbf{CSP_{boole}}(\mathbf{tw}^d)$ and $\mathbf{CSP}(\mathbf{tw}^d, \mathbf{dom})$. Consequently, the latter problems are W[1]-hard. Since $\mathbf{tw}^*$ dominates $\mathbf{tw}^d$ (Lemma 2(5)), also $\mathbf{CSP_{boole}}(\mathbf{tw}^*)$ and $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom})$ are W[1]-hard (Lemma 1).

In turn, it is well known that the treewidth of incidence graphs is dominated by each of the parameters (generalized) hypertree width, fractional hypertree width, and spread-cut width of constraint hypergraphs [7,16,18,20]. Hence, Corollary 1 follows by Lemma 1.

The W[1]-hardness of $\mathbf{CSP}(\mathbf{tw}^d, \mathbf{arity})$ can be shown by an fpt-reduction from **CLIQUE** as well. This reduction, which is easier than the reduction above, was used by Papadimitriou and Yannakakis for showing W[1]-hardness of $\mathbf{CSP}(\mathbf{vars})$: given a graph $G = (V, E)$ and an integer k, we construct a CSP instance $I = (\{x_1, \ldots, x_k\}, V, F)$ where F contains constraints $((x_i, x_j), E)$ for all $1 \leq i < j \leq k$. Evidently, G contains a clique of size k if and only if I is consistent. Since there are $\binom{k}{2}$ constraints, the dual graph of I has a trivial tree decomposition of width $\binom{k}{2} - 1$. Thus $\mathbf{tw}^d(I) \leq \binom{k}{2} - 1$ and $\mathbf{arity}(I) = 2$. Whence $\mathbf{CSP_{bin}}(\mathbf{tw}^d, \mathbf{arity})$ and $\mathbf{CSP}(\mathbf{tw}^d, \mathbf{arity})$ are W[1]-hard.

For establishing Theorem 1 it remains to show that $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom}, \mathbf{ovl})$ is fixed-parameter tractable.

5 Fixed-Parameter Algorithm for $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom}, \mathbf{ovl})$

For this section, let (T, χ, r) be a nice tree decomposition of width k of the incidence graph of a constraint satisfaction instance $I = (V, D, F)$.

For each node t of T, let F_t denote the set of all the constraints in V_t, and let X_t denote the set of all variables in V_t; that is, $F_t = V_t \cap F$ and $X_t = V_t \cap V$. We also use the shorthands $\chi_c(t) = \chi(t) \cap F$ and $\chi_v(t) = \chi(t) \cap V$ for the set of variables and the set of constraints in $\chi(t)$, respectively. Moreover, $\chi_v^*(t)$ denotes the set of all variables of X_t that are in $\chi_v(t)$ or in $var(C_1) \cap var(C_2)$ for two distinct constraints $C_1, C_2 \in \chi_c(t)$. That is, $\chi_v^*(t)$ is the set of variables in $\chi_v(t)$ together with all "forgotten" variables in X_t that occur in at least two constraints in $\chi_c(t)$.

Let t be a node of T and let $\alpha : \chi_v^*(t) \to D$ be an assignment. We define $N(t, \alpha)$ as the set of assignments $\tau : X_t \to D$ such that $\tau|_{\chi_v^*(t)} = \alpha$ and τ is consistent with all constraints in F_t. Consequently, I is consistent if and only if $N(r, \alpha) \neq \emptyset$ for some $\alpha : \chi_v^*(r) \to D$. The following lemmas show that we can decide consistency of I by dynamic programming along a bottom-up traversal of T.

Lemma 3 *Let t be a* join node *of T with children t_1, t_2. Let $\alpha : \chi_v^*(t) \to D$ be an assignment and $\alpha_i = \alpha|_{\chi_v^*(t_i)}$, $i = 1, 2$. Then the following holds:*

(1) $N(t, \alpha) \neq \emptyset$ if and only if $N(t_1, \alpha_1) \neq \emptyset$, $N(t_2, \alpha_2) \neq \emptyset$, and $C[N(t_1, \alpha_1)] \cap C[N(t_2, \alpha_2)] \neq \emptyset$ for all $C \in \chi_c(t)$.
(2) If $N(t, \alpha) \neq \emptyset$, then $C[N(t, \alpha)] = C[N(t_1, \alpha_1)] \cap C[N(t_2, \alpha_2)]$ for all $C \in \chi_c(t)$.

PROOF. Part 1: For the *only if* direction, let $\tau \in N(t, \alpha)$, and $\tau_1 = \tau|_{X_{t_1}}$ and $\tau_2 = \tau|_{X_{t_2}}$. It is then easy to verify that $\tau_1 \in N(t_1, \alpha_1)$ and $\tau_2 \in N(t_2, \alpha_2)$. Moreover, it holds that $C[\tau] \neq \emptyset$ for all $C \in \chi_c(t)$, which implies $C[\tau_1] \cap C[\tau_2] \neq \emptyset$ for all $C \in \chi_c(t)$. Hence, we have $C[N(t_1, \alpha_1)] \cap C[N(t_2, \alpha_2)] \neq \emptyset$ for all $C \in \chi_c(t)$. For the *if* direction, let $\tau_1 \in N(t_1, \alpha_1)$ and $\tau_2 \in N(t_2, \alpha_2)$ such that $C[\tau_1] \cap C[\tau_2] \neq \emptyset$ for all $C \in \chi_c(t)$. Now, let us define the assignment $\tau : X_t \to D$ by $\tau|_{X_{t_1}} = \tau_1$ and $\tau|_{X_{t_2}} = \tau_2$. To verify that $\tau \in N(t, \alpha)$, note that, by the Connectedness Condition, we have $V_{t_1} \cap V_{t_2} = \chi(t)$, that is, $X_{t_1} \cap X_{t_2} = \chi_v(t) \subseteq \chi_v^*(t)$ and $F_{t_1} \cap F_{t_2} = \chi_c(t)$. Moreover, it holds that $\chi_v^*(t_1) \cup \chi_v^*(t_2) = (\chi_v^*(t) \cap X_{t_1}) \cup (\chi_v^*(t) \cap X_{t_2}) = \chi_v^*(t) \cap (X_{t_1} \cup X_{t_2}) = \chi_v^*(t) \cap X_t = \chi_v^*(t)$. Part 2 follows immediately from the above constructions. $\square$

Lemma 4 *Let t be an* introduce node *with child t' where $\chi(t) = \chi(t') \cup \{x\}$ for a variable x. Let $\alpha : \chi_v^*(t') \to D$ be an assignment and $\beta = \alpha \cup \{(x, d)\}$ for some domain element $d \in D$. Then the following holds:*

(1) $N(t, \beta) \neq \emptyset$ if and only if $N(t', \alpha) \neq \emptyset$ and $C[N(t', \alpha)] \cap C[\{(x, d)\}] \neq \emptyset$ for all $C \in \chi_c(t)$.
(2) If $N(t, \beta) \neq \emptyset$, then $C[N(t, \beta)] = C[N(t', \alpha)] \cap C[\{(x, d)\}]$ for all $C \in \chi_c(t)$.

PROOF. Part 1: For the *only if* direction, let $\tau \in N(t, \beta)$ and $\tau' = \tau|_{X_{t'}}$. Thus, it follows that $\tau' \in N(t', \alpha)$. Moreover, it holds that $C[\tau] \neq \emptyset$ for all $C \in \chi_c(t)$, which implies $C[\tau'] \cap C[\{(x, d)\}] \neq \emptyset$ for all $C \in \chi_c(t)$. Hence, we have $C[N(t', \alpha)] \cap C[\{(x, d)\}] \neq \emptyset$ for all $C \in \chi_c(t)$. For the *if* direction, let $\tau' \in N(t', \alpha)$ such that $C[\tau'] \cap C[\{(x, d)\}] \neq \emptyset$ for all $C \in \chi_c(t)$. Now, let us define the assignment $\tau : X_t \to D$ by $\tau|_{X_{t'}} = \tau'$ and $\tau(x) = d$ for the single variable $x \in X_t \setminus X_{t'}$. It is then easy to show that $\tau \in N(t, \beta)$. Part 2 follows immediately from the above constructions. $\square$

Lemma 5 *Let t be an* introduce node *with child t' where $\chi(t) = \chi(t') \cup \{B\}$ for a constraint B. Let $\alpha : \chi_v^*(t) \to D$ be an assignment. Then the following holds:*

(1) $N(t,\alpha) \neq \emptyset$ if and only if $N(t',\alpha) \neq \emptyset$ and $B[\alpha] \neq \emptyset$.
(2) If $N(t,\alpha) \neq \emptyset$, then $C[N(t,\alpha)] = C[N(t',\alpha)]$ for all $C \in \chi_c(t')$.

PROOF. Part 1: For the *only if* direction, let $\tau \in N(t,\alpha)$ and $\tau' = \tau|_{X_{t'}}$. So we have $\tau' \in N(t',\alpha)$. Moreover, since $C[\tau] \neq \emptyset$ for all $C \in \chi_c(t)$, we know that $B[\tau] \neq \emptyset$. Thus, since $\tau|_{\chi_v^*(t)} = \alpha$, we obtain $B[\alpha] \neq \emptyset$. For the *if* direction, let $\tau' \in N(t',\alpha)$ and $B[\alpha] \neq \emptyset$. By the construction of a tree decomposition of an incidence graph, we know that $var(B) \cap X_t \subseteq \chi_v(t) \subseteq \chi_v^*(t)$. Thus, since $\tau'|_{\chi_v^*(t)} = \alpha$, we have $B[\tau'] \neq \emptyset$. So it can be easily verified that $\tau' \in N(t,\alpha)$. Part 2 follows immediately from the above constructions. $\square$

Lemma 6 *Let t be a* forget node *with child t' where $\chi(t) = \chi(t') \setminus \{x\}$ for a variable x. Let $\alpha : \chi_v^*(t) \to D$ be an assignment. If $x \in \chi_v^*(t)$, then the following holds:*

(1) $N(t,\alpha) \neq \emptyset$ if and only if $N(t',\alpha) \neq \emptyset$.
(2) If $N(t,\alpha) \neq \emptyset$, then $C[N(t,\alpha)] = C[N(t',\alpha)]$ for all $C \in \chi_c(t)$.

Otherwise, if $x \notin \chi_v^*(t)$, then the following holds:

(1) $N(t,\alpha) \neq \emptyset$ if and only if $N(t',\alpha \cup \{(x,d)\}) \neq \emptyset$ for some $d \in D$.
(2) If $N(t,\alpha) \neq \emptyset$, then $C[N(t,\alpha)] = \bigcup_{d \in D} C[N(t',\alpha \cup \{(x,d)\})]$ for all $C \in \chi_c(t)$.

PROOF. The case of $x \in \chi_v^*(t)$ is trivial, since $\chi_v^*(t) = \chi_v^*(t')$ and $\chi_c(t) = \chi_c(t')$. Let us therefore consider the more interesting case of $x \notin \chi_v^*(t)$. Part 1: For the *only if* direction, let $\tau \in N(t,\alpha)$. Thus, since $X_t = X_{t'}$, we know that $\tau(x) = d$ for the single variable $x \in \chi_v^*(t') \setminus \chi_v^*(t)$ and some domain element $d \in D$. Consequently, $\tau \in N(t',\alpha \cup \{(x,d)\})$ for some $d \in D$. For the *if* direction, let $\tau' \in N(t',\alpha \cup \{(x,d)\})$ for some $d \in D$. Then we trivially have $\tau' \in N(t,\alpha)$. Part 2 follows immediately from the above constructions. $\square$

Lemma 7 *Let t be a* forget node *with child t' where $\chi(t) = \chi(t') \setminus \{B\}$ for a constraint B. Let $\alpha : \chi_v^*(t) \to D$ be an assignment. Then the following holds:*

(1) $N(t,\alpha) \neq \emptyset$ if and only if $N(t',\alpha') \neq \emptyset$ for some $\alpha' : \chi_v^*(t') \to D$ s.t. $\alpha = \alpha'|_{\chi_v^*(t)}$.
(2) If $N(t,\alpha) \neq \emptyset$, then $C[N(t,\alpha)] = \bigcup_{\alpha = \alpha'|_{\chi_v^*(t)}} C[N(t',\alpha')]$ for all $C \in \chi_c(t)$.

PROOF. Part 1: For the *only if* direction, let $\tau \in N(t,\alpha)$. Thus, since $X_t = X_{t'}$, we know that for all variables $x \in \chi_v^*(t') \setminus \chi_v^*(t)$ there exists some domain element $d \in D$ such that $\tau(x) = d$. Consequently, $\tau \in N(t',\alpha')$ for some $\alpha' : \chi_v^*(t') \to D$ such that $\alpha = \alpha'|_{\chi_v^*(t)}$. For the *if* direction, let $\tau' \in N(t',\alpha')$

for some $\alpha' : \chi_v^*(t') \to D$ such that $\alpha = \alpha'|_{\chi_v^*(t)}$. Then we trivially have $\tau' \in N(t, \alpha)$. Part 2 follows immediately from the above constructions. $\square$

Lemma 8 *Let t be a* leaf node *and $\alpha : \chi_v^*(t) \to D$ be an assignment. Then the following holds:*

(1) $N(t, \alpha) \neq \emptyset$ if and only if $C[\alpha] \neq \emptyset$ for all $C \in \chi_c(t)$.
(2) If $N(t, \alpha) \neq \emptyset$, then $C[N(t, \alpha)] = C[\alpha]$ for all $C \in \chi_c(t)$.

PROOF. Since $X_t = \chi_v(t) = \chi_v^*(t)$ and $F_t = \chi_c(t)$ for every leaf node t, we immediately obtain the above properties. $\square$

In the following, we represent the sets $C[N(t, \alpha)]$ for each $\alpha : \chi_v^*(t) \to D$ and $C \in \chi_c(t)$ by a table M_t with at most $|\chi_v^*(t)| + |\chi_c(t)|m$ columns and $|D|^{|\chi_v^*(t)|}$ rows, where $m = \max_{C \in F} |rel(C)|$. The first $|\chi_v^*(t)|$ columns of M_t contain values from D encoding $\alpha(x)$ for variables $x \in \chi_v^*(t)$. The further columns of M_t represent the tuples in $rel(C)$ for $C \in \chi_c(t)$ and contain Boolean values. We denote by $M_t(\alpha)$ the row of M_t that encodes α, and by $M_t(\alpha, C)$ the set of tuples in $rel(C)$ that have the Boolean value 1 in row $M_t(\alpha)$. Note that rows $M_t(\alpha)$ with $N(t, \alpha) = \emptyset$ could be omitted from the table for efficiency reasons; in this case, one puts $C[N(t, \alpha)] = \emptyset$ for all $C \in \chi_c(t)$. The tables M_t are constructed according to Algorithm 1 and the following lemma. Note that we understand by the *size* of a constraint relation the product of its arity and the number of its tuples.

Lemma 9 *Let t be a node of T. Given the tables of the children of t, we can compute the table M_t in time $\mathcal{O}(d^p pks)$, where $p = |\chi_v^*(t)|$, $d = |D|$, and s is the size of a largest constraint relation of constraints of I.*

PROOF. To check the running time for computing M_t, let $q = |\chi_c(t)|$, l be the size of the largest constraint scope, and m be the maximal number of tuples in a constraint relation; recall that the width of the tree decomposition under consideration is k. Now, let us distinguish between the different kinds of nodes.

Case 1. Let t be a *join node* with children t_1, t_2. We compute the table M_t from the tables M_{t_1} and M_{t_2} according to Lemma 3 as follows: for each of the d^p choices of α, we consider the rows $M_{t_1}(\alpha|_{\chi_v^*(t_1)})$ and $M_{t_2}(\alpha|_{\chi_v^*(t_2)})$ and set $M_t(\alpha, C)$ to $M_{t_1}(\alpha, C) \cap M_{t_2}(\alpha, C)$ for each of the q constraints $C \in \chi_c(t)$. Finding $M_{t_1}(\alpha|_{\chi_v^*(t_1)})$ and $M_{t_2}(\alpha|_{\chi_v^*(t_2)})$ and computing $M_t(\alpha, C)$ for all $C \in \chi_c(t)$ can be accomplished in time $\mathcal{O}(p + qm)$. If $M_t(\alpha, C) = \emptyset$ for some $C \in \chi_c(t)$, we put $M_t(\alpha, C) = \emptyset$ for all $C \in \chi_c(t)$. Hence, we can compute M_t in time $\mathcal{O}(d^p(p + qm)) \subseteq \mathcal{O}(d^p pks)$.

Algorithm 1 Solving $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom}, \mathbf{ovl})$

```
procedure buildM($t$) begin
    if isJoinNode($t$) then begin                                        // see Lemma 3
        $t_1 := getChild1(t)$;  $t_2 := getChild2(t)$;
        buildM($t_1$);  buildM($t_2$);
        for all $\alpha : \chi_v^*(t) \to D$ do begin
            $\alpha_1 := \alpha|_{\chi_v^*(t_1)}$; $\alpha_2 := \alpha|_{\chi_v^*(t_2)}$;
            if $\exists C \in \chi_c(t)$ such that $M_{t_1}(\alpha_1, C) \cap M_{t_2}(\alpha_2, C) = \emptyset$ then
                for all $C \in \chi_c(t)$ do $M_t(\alpha, C) := \emptyset$;
            else
                for all $C \in \chi_c(t)$ do $M_t(\alpha, C) := M_{t_1}(\alpha_1, C) \cap M_{t_2}(\alpha_2, C)$;
        end
    end
    else if isIntroNode($t$) then begin
        $t' := getChild(t)$;
        buildM($t'$);
        if $\chi(t) \setminus \chi(t')$ is a variable then begin              // see Lemma 4
            $x := \chi(t) \setminus \chi(t')$;
            for all $\alpha : \chi_v^*(t') \to D$ and $d \in D$ do begin
                $\beta := \alpha \cup \{(x, d)\}$;
                if $\exists C \in \chi_c(t)$ such that $M_{t'}(\alpha, C) \cap C[\{(x, d)\}] = \emptyset$ then
                    for all $C \in \chi_c(t)$ do $M_t(\beta, C) := \emptyset$;
                else
                    for all $C \in \chi_c(t)$ do $M_t(\beta, C) := M_{t'}(\alpha, C) \cap C[\{(x, d)\}]$;
            end
        end
        else $\chi(t) \setminus \chi(t')$ is a constraint begin               // see Lemma 5
            $B := \chi(t) \setminus \chi(t')$;
            for all $\alpha : \chi_v^*(t) \to D$ do
                if $B[\alpha] = \emptyset$ or $\exists C \in \chi_c(t')$ such that $M_{t'}(\alpha, C) = \emptyset$ then
                    for all $C \in \chi_c(t)$ do $M_t(\alpha, C) := \emptyset$;
                else begin
                    $M_t(\alpha, B) := B[\alpha]$;
                    for all $C \in \chi_c(t')$ do $M_t(\alpha, C) := M_{t'}(\alpha, C)$;
                end
        end
    end
    else if isForgetNode($t$) then begin
        $t' := getChild(t)$;
        buildM($t'$);
        if $\chi(t') \setminus \chi(t)$ is a variable then begin              // see Lemma 6
            $x := \chi(t') \setminus \chi(t)$;
            if $x \in \chi_v^*(t)$ then
                for all $\alpha : \chi_v^*(t) \to D$ and $C \in \chi_c(t)$ do
                    $M_t(\alpha, C) := M_{t'}(\alpha, C)$;
            else $x \notin \chi_v^*(t)$
                for all $\alpha : \chi_v^*(t) \to D$ and $C \in \chi_c(t)$ do
                    $M_t(\alpha, C) := \bigcup_{d \in D} M_{t'}(\alpha \cup \{(x, d)\}, C)$;
        end
        else $\chi(t') \setminus \chi(t)$ is a constraint                      // see Lemma 7
            for all $\alpha : \chi_v^*(t) \to D$ and $C \in \chi_c(t)$ do
                $M_t(\alpha, C) := \bigcup_{\alpha' : \chi_v^*(t') \to D, \alpha = \alpha'|_{\chi_v^*(t)}} M_{t'}(\alpha', C)$;
    end
    else if isLeaf($t$) then                                              // see Lemma 8
        for all $\alpha : \chi_v^*(t) \to D$ do
            if $\exists C \in \chi_c(t)$ such that $C[\alpha] = \emptyset$ then
                for all $C \in \chi_c(t)$ do $M_t(\alpha, C) := \emptyset$;
            else
                for all $C \in \chi_c(t)$ do $M_t(\alpha, C) := C[\alpha]$;
end
```

Case 2. Let t be an *introduce node* with child t'. We compute the table M_t from table $M_{t'}$ according to Lemma 4 and Lemma 5 as follows:
(a) If $\chi(t) = \chi(t') \cup \{x\}$ for a variable x, we consider the rows $M_t(\alpha \cup \{(x, d)\})$ for each of the d^{p-1} choices of α, and set $M_t(\alpha \cup \{(x, d)\}, C)$ to $M_{t'}(\alpha, C) \cap C[\{(x, d)\}]$ for each of the q constraints $C \in \chi_c(t)$. Finding $M_t(\alpha \cup \{(x, d)\})$ for all $d \in D$ and computing $M_t(\alpha \cup \{(x, d)\}, C)$ for all $C \in \chi_c(t)$ can be accomplished in time $\mathcal{O}(dq(l + m))$. If $M_t(\alpha \cup \{(x, d)\}, C) = \emptyset$ for some $C \in \chi_c(t)$, we put $M_t(\alpha \cup \{(x, d)\}, C) = \emptyset$ for all $C \in \chi_c(t)$. Hence, we can compute M_t in time $\mathcal{O}(d^p q(l + m)) \subseteq \mathcal{O}(d^p pks)$.
(b) If $\chi(t) = \chi(t') \cup \{B\}$ for a constraint B, we consider the row $M_t(\alpha)$ for each of the d^p choices of α, and set $M_t(\alpha, B)$ to $B[\alpha]$. Computing $M_t(\alpha, B)$ can be accomplished in time $\mathcal{O}(l+mp)$. If $M_t(\alpha, B) = \emptyset$, we put $M_t(\alpha, C) = \emptyset$ for all $C \in \chi_c(t)$; otherwise, we set $M_t(\alpha, C)$ to $M_{t'}(\alpha, C)$ for all $C \in \chi_c(t')$. Hence, we can compute M_t in time $\mathcal{O}(d^p(l + mp + mq)) \subseteq \mathcal{O}(d^p pks)$.

Case 3. Let t be a *forget node* with child t'. We compute the table M_t from table $M_{t'}$ according to Lemma 6 and Lemma 7 as follows:
(a) If $\chi(t) = \chi(t') \setminus \{x\}$ for a variable x, we distinguish between two cases: (i) If $x \in \chi_v^*(t)$, we consider the row $M_t(\alpha)$ for each of the d^p choices of α, and set $M_t(\alpha, C)$ to $M_{t'}(\alpha, C)$ for each of the q constraints $C \in \chi_c(t)$. Computing $M_t(\alpha, C)$ can be accomplished in time $\mathcal{O}(qm)$. (ii) If $x \notin \chi_v^*(t)$, we consider the rows $M_t(\alpha \cup \{(x, d)\})$ for each of the d^{p-1} choices of α, and set $M_t(\alpha, C)$ to $\bigcup_{d \in D} M_{t'}(\alpha \cup \{(x, d)\}, C)$ for each of the q constraints $C \in \chi_c(t)$. Finding $M_t(\alpha \cup \{(x, d)\})$ for all $d \in D$ and computing $M_t(\alpha, C)$ can be accomplished in time $\mathcal{O}(dqm)$. Hence, we can compute M_t in time $\mathcal{O}(p+d^p qm) \subseteq \mathcal{O}(d^p pks)$.
(b) If $\chi(t) = \chi(t') \setminus \{B\}$ for a constraint B, we compute $\chi_v^*(t)$ from $\chi_v^*(t')$ in time $\mathcal{O}(p + qm)$. Then, we consider the row $M_t(\alpha)$ for each of the d^p choices of α' such that $\alpha = \alpha'|_{\chi_v^*(t)}$, and set $M_t(\alpha, C)$ to $\bigcup_{\alpha=\alpha'|_{\chi_v^*(t)}} M_t(\alpha', C)$ for each of the q constraints $C \in \chi_c(t)$. Finding $M_t(\alpha)$ and computing $M_t(\alpha, C)$ can be accomplished in time $\mathcal{O}(p+qm)$. Hence, we can compute M_t in time $\mathcal{O}(d^p(p+qm)) \subseteq \mathcal{O}(d^p pks)$.

Case 4. Let t be a *leaf node*. We compute the table M_t according to Lemma 8 as follows: for each of the d^p choices of α, we consider the row $M_t(\alpha)$ and set $M_t(\alpha, C)$ to $C[\alpha]$ for each of the q constraints $C \in \chi_c(t)$. Computing $M_t(\alpha, C)$ for all $C \in \chi_c(t)$ can be accomplished in time $\mathcal{O}(q(l + pm))$. If $M_t(\alpha, C) = \emptyset$ for some $C \in \chi_c(t)$, we put $M_t(\alpha, C) = \emptyset$ for all $C \in \chi_c(t)$. Hence, we can compute M_t in time $\mathcal{O}(d^p q(l + pm)) \subseteq \mathcal{O}(d^p pks)$. □

Theorem 2 *Given a constraint satisfaction instance I together with a nice tree decomposition (T, χ) of the incidence graph of I. Let d be the size of the domain of I and let s be the size of a largest constraint relation of constraints of I. Furthermore, let k be the width and n the number of nodes of (T, χ), and let p denote the maximum $|\chi_v^*(t)|$ over all nodes t of T. Then we can decide in time $\mathcal{O}(d^p pksn)$ whether I is consistent.*

PROOF. We construct the tables M_t for all nodes t of T in a bottom up ordering, starting from the leaf nodes. By Lemma 9, each table can be computed in time $\mathcal{O}(d^p pks)$. Since I is consistent if and only if M_r is nonempty, the theorem follows. $\square$

Corollary 2 $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom}, \mathbf{ovl})$ *is fixed-parameter tractable.*

PROOF. Let I be a constraint satisfaction instance whose incidence graph has treewidth at most k and $c = \mathbf{ovl}(I)$. Recall from Section 2.2 that we can find in linear time a nice tree decomposition of the incidence graph of G of width at most k. Now the corollary follows immediately from Theorem 2 and the fact that $|\chi_v^*(t)| \leq k + c\,k^2$ holds for all nodes t of T. $\square$

Corollary 2 provides the last step for the proof of Theorem 1.

6 Fixed-Parameter Tractability for Further Parameters

In this section, we describe two further constraint satisfaction parameters for which Theorem 2 applies.

Let $I = (V, D, F)$ be a constraint satisfaction instance. For a subset F' of F we define $\delta_I(F')$ as the set of variables that occur in the scopes of all constraints of F' but in no scope of constraints of $F \setminus F'$; i.e., $\delta_I(F') = (\bigcap_{C \in F'} \mathit{var}(C)) \setminus (\bigcup_{C \in F \setminus F'} \mathit{var}(C))$. We define $\mathbf{equiv}(I)$ as the maximum size of $\delta_I(F')$ over all subsets $F' \subseteq F$ that contain at least two constraints. Furthermore, we define $\mathbf{diff}(I)$ as the maximum size of $\mathit{var}(C_1) \setminus \mathit{var}(C_2)$ over all pairs of constraints $C_1, C_2 \in F$.

Corollary 3 $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom}, \mathbf{equiv})$ *is fixed-parameter tractable.*

PROOF. Let I be a constraint satisfaction instance whose incidence graph has treewidth at most k. We compute in linear time a nice tree decomposition (T, χ, r) of the incidence graph of I of width at most k. Let $q = \mathbf{equiv}(I)$. Evidently, we have $|\chi_v^*(t)| \leq k + q\,2^k$ for all nodes t of T. Hence the corollary follows immediately from Theorem 2. $\square$

Note that the running time of our fixed-parameter algorithm for $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom}, \mathbf{ovl})$ is significantly smaller than the running time for $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom}, \mathbf{equiv})$. However, there exist instances with bounded **equiv** and arbitrarily large **ovl** (i.e., **equiv** dominates **ovl**, but not *vice versa*). For example, let us construct an instance in the following way: we start with any constraint C_0 and add in each step a new constraint C_n and a new variable x_n such that $\bigcap_{0 \leq i \leq n} var(C_i) = \{x_n\}$. By construction $\mathbf{equiv}(I) = 1$, but $\mathbf{ovl}(I) \geq n$ since $|C_0 \cap C_1| = n$.

Corollary 4 $\mathbf{CSP}(\mathbf{tw}^*, \mathbf{dom}, \mathbf{diff})$ *is fixed-parameter tractable.*

PROOF. Again, let $I = (V, D, F)$ be a constraint satisfaction instance whose incidence graph has treewidth at most k. Let $q' = \mathbf{diff}(I)$ and let $d = |D|$. We compute in linear time a nice tree decomposition (T, χ, r) of the incidence graph of I of width at most k and n nodes. Next we obtain from I a solution-equivalent constraint satisfaction instance I' by computing the join of all constraints in $\chi_c(t)$ for each node t of T. A constraint $C = ((x_1, \ldots, x_r), R)$ is the *join* of constraints $C_1, \ldots, C_t$ if $var(C) = \bigcup_{i=1}^{t} var(C_i)$ and if R consists of all tuples $(\tau(x_1), \ldots, \tau(x_r))$ for assignments τ that are consistent with all C_i, where $1 \leq i \leq t$ (cf. [1]). Note that the resulting relation of a join operation of two constraints over D with relations of size at most s can be of size at most $2s\, d^{q'}$ by our restriction. Thus, the join of at most k constraints can be computed in time $\mathcal{O}(k\, s^2 d^{q'(k-1)})$ and the size of the largest relation of I' is bounded by $k\, s\, d^{q'(k-1)}$. Moreover, note that the tree decomposition of the incidence graph of I gives rise to a tree decomposition of the incidence graph of I'; for the latter we have $\chi_v^*(t) = \chi_v(t)$, that is, $|\chi_v^*(t)| \leq k$, for all nodes t of T. Hence, in view of Theorem 2, we obtain a running time of $\mathcal{O}(d^{(q'+1)k} k^3 s^2 n)$ for the dynamic programming algorithm. $\square$

Since $\mathbf{tw}^*$ dominates $\mathbf{tw}^d$, we obtain from these corollaries that the problems $\mathbf{CSP}(\mathbf{tw}^d, \mathbf{dom}, \mathbf{equiv})$ and $\mathbf{CSP}(\mathbf{tw}^d, \mathbf{dom}, \mathbf{diff})$ are fixed-parameter tractable.

7 Conclusion

We have presented a general framework for studying the trade-off between generality and performance for parameterized constraint satisfaction. Within our framework we have classified the parameterized complexity of combinations of natural parameters including the treewidth of primal, dual, and incidence graphs, the domain size, and the size of overlaps of constraint scopes. The parameterized complexity of further parameters and their combinations remain open for future research. Furthermore, it would be interesting to extend the hardness results of this paper to completeness results for classes of the weft hierarchy.

Acknowledgment

We thank Moshe Vardi for suggesting us to include the treewidth of dual graphs into our considerations.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

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

[3] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

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

[5] H. Chen and V. Dalmau. Beyond hypertree width: Decomposition methods without decompositions. In *Proc. 11th International Conference on Principles and Practice of Constraint Programming (CP'05)*, vol. 3709 of LNCS, pages 167–181. Springer-Verlag, 2005.

[6] D. Cohen and P. Jeavons. The complexity of constraint languages. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, part I, chapter 8. Elsevier, 2006.

[7] D. Cohen, P. Jeavons, and M. Gyssens. A unified theory of structural tractability for constraint satisfaction and spread cut decomposition. In *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 72–77. Professional Book Center, 2005.

[8] R. Dechter. Tractable structures for constraint satisfaction problems. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, part I, chapter 7. Elsevier, 2006.

[9] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38(3):353–366, 1989.

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

[11] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer-Verlag, 2006.

[12] E. C. Freuder. A sufficient condition for backtrack-bounded search. *Journal of the ACM*, 32(4):755–761, 1985.

[13] T. Ganzow, G. Gottlob, N. Musliu, and M. Samer. A CSP hypergraph library. Technical report DBAI-TR-2005-50, Database and Artificial Intelligence Group, Vienna University of Technology, 2005.

[14] G. Gottlob, M. Grohe, N. Musliu, M. Samer, and F. Scarcello. Hypertree decompositions: Structure, algorithms, and applications. In *Proc. 31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG'05)*, vol. 3787 of LNCS, pages 1–15. Springer-Verlag, 2005.

[15] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions: A survey. In *Proc. 26th International Symposium on Mathematical Foundations of Computer Science (MFCS'01)*, vol. 2136 of LNCS, pages 37–57. Springer-Verlag, 2001.

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

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

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

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

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

[21] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[22] C. H. Papadimitriou and M. Yannakakis. On the complexity of database queries. *Journal of Computer and System Sciences*, 58(3):407–427, 1999.

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

[24] M. Samer and S. Szeider. A fixed-parameter algorithm for #SAT with respect to parameter incidence treewidth. Technical report arXiv:cs.DS/0610174, 2006.

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