Combining Treewidth and Backdoors for CSP*†

Robert Ganian‡[1], M. S. Ramanujan[2], and Stefan Szeider[3]

1 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
rganian@ac.tuwien.ac.at
2 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
ramanujan@ac.tuwien.ac.at
3 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
sz@ac.tuwien.ac.at

Abstract

We show that CSP is fixed-parameter tractable when parameterized by the treewidth of a backdoor into any tractable CSP problem over a finite constraint language. This result combines the two prominent approaches for achieving tractability for CSP: (i) structural restrictions on the interaction between the variables and the constraints and (ii) language restrictions on the relations that can be used inside the constraints. Apart from defining the notion of backdoor-treewidth and showing how backdoors of small treewidth can be used to efficiently solve CSP, our main technical contribution is a fixed-parameter algorithm that finds a backdoor of small treewidth.

1998 ACM Subject Classification G.2.1 Combinatorics, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Algorithms and data structures, Fixed Parameter Tractability, Constraint Satisfaction

Digital Object Identifier 10.4230/LIPIcs.STACS.2017.36

1 Introduction

The Constraint Satisfaction Problem (CSP) is a central and generic computational problem which provides a common framework for many theoretical and practical applications [34]. An instance of CSP consists of a collection of variables that must be assigned values subject to constraints, where each constraint is given in terms of a relation whose tuples specify the allowed combinations of values for specified variables. The problem was originally formulated by Montanari [41], and has been found equivalent to the homomorphism problem for relational structures [22] and the problem of evaluating conjunctive queries on databases [37]. CSP is NP-complete in general, and identifying the classes of CSP instances which can be solved efficiently has become a prominent research area in theoretical computer science [10].

One of the most classical approaches in this area relies on exploiting the *structure* of how variables and constraints interact with each other, most prominently in terms of the *treewidth* of graph representations of CSP instances. The first result in this line of research dates back to 1985, when Freuder [25] observed that CSP is polynomial-time tractable if the *primal treewidth*, which is the treewidth of the *primal graph* of the instance, is bounded by a constant. A large number of related results on structural restrictions for CSP have been obtained to date (see, e.g., [13, 18, 30, 31, 40, 44]).

* A full version of the paper is available at https://arxiv.org/abs/1610.03298.
† The authors acknowledge support from Austrian Science Funds (FWF), project P26696.
‡ Robert Ganian is also affiliated with FI MU, Brno, Czech Republic.

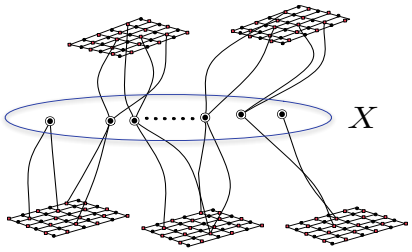

Figure 1 An illustration of instances with neither a small backdoor into CSP(Γ) for any tractable constraint language Γ, nor bounded primal treewidth. Here, X denotes a minimum strong backdoor of unbounded size into CSP(Γ) for some choice of Γ. The ellipsis represents the fact that X can be arbitrarily big, implying an unbounded number of such instances, one for each size of X.

The other leading approach used to show the tractability of constraint satisfaction relies on *constraint languages*. In this case, the polynomially tractable classes are defined in terms of a tractable *constraint language* Γ, which is a set of relations that can be used in the instance. A landmark result in this area is Schaefer's celebrated Dichotomy Theorem for Boolean CSP [46] which says that for every constraint language Γ over the Boolean domain, the corresponding CSP problem is either NP-complete or solvable in polynomial time. Feder and Vardi [22] conjectured that such a dichotomy holds for all finite constraint languages. Although the conjecture is still open it has been proven true for many important special cases (see, e.g., [6, 7, 9, 14, 17, 33]).

Tractability due to restrictions on the constraint language and tractability due to restrictions in terms of the structure of the CSP instance are often considered complementary: under structural restrictions the domain language can be arbitrary, whereas under constraint language restrictions the variables and constraints can interact arbitrarily.

One specific tool that is frequently used to build upon the constraint language approach detailed above is the notion of *backdoors*, which provides a means of relaxing celebrated results on tractable constraint languages to instances which are 'almost' tractable. In particular, this is done by measuring the size of a *strong backdoor* [47] to a selected tractable class, where a strong backdoor is a set of variables with the property that every assignment of these variables results in a CSP instance in the specified class. A natural way of defining such a class is to consider all CSP instances whose constraints use relations from a constraint language Γ, denoted by CSP(Γ). The last couple of years have seen several new results for CSP using this backdoor-based approach (see, e.g., [11, 26, 27]. In particular, the general aim of research in this direction is to obtain so-called *fixed-parameter* algorithms, i.e., algorithms where the running time only has a polynomial dependence on the input size and the exponential blow-up is restricted exclusively to the size of the backdoor (the *parameter*). Parameterized decision problems which admit such an algorithm belong to the complexity class FPT.

We note that treewidth-based and backdoor-based approaches outlined above are orthogonal to each other. Consider, for example, on the one hand a CSP instance which is tractable due to the used constraint language but which has high treewidth, or on the other hand an instance consisting of many disjoint copies of CSP instances of constant primal treewidth with a constant-size strong backdoor into a tractable constraint language (backdoor size multiplies whereas treewidth remains constant). Hence applying either of these approaches (treewidth-based and backdoor-based) alone will not yield satisfactory results for instances that are not homogeneous with respect to either of these forms of restrictions. It is certainly natural to consider the algorithmic complexity of instances which have small treewidth after

certain simple 'blocks' characterized by a tractable constraint language are removed, or instances with a large but 'well-structured' backdoor to a tractable class (see Fig. 1), but until now we lacked the theoretical tools required to capture the complexity of such instances.

Our Results. We propose and develop a hybrid framework for constraint satisfaction which combines the advantages of both the width-based and backdoor-based approaches. In particular, we introduce the notion of *backdoor-treewidth* with respect to a constraint language Γ; this is defined, roughly speaking, as the primal treewidth of the instance after contracting (possibly large) parts of the instance into single constraints, so that the remaining variables form a strong backdoor into CSP(Γ) in the original instance. We refer to Definition 5 for the formal definition of backdoor-treewidth. It is not difficult to see that backdoor-treewidth is at most the minimum of primal treewidth and the size of a backdoor into the specified class. However, backdoor-treewidth can be arbitrarily smaller than both the primal treewidth and the size of a backdoor, and hence promises to push the frontiers of tractability beyond the current state of the art.

▶ **Theorem 1.** *Let Γ be a fixed tractable finite constraint language. Then, CSP parameterized by the backdoor-treewidth with respect to Γ is* FPT.

We note that our result is in fact tight as far as the choice of the language Γ is concerned: Γ must clearly be tractable, and both the backdoor-based and width-based approaches are known to fail for infinite languages under established complexity assumptions. To be more specific, finding strong backdoors is not even FPT parameterized by backdoor size if the arity of relations in the language is unbounded [27], primal treewidth implicitly bounds the arity of relations, and both approaches require bounded domain to solve CSP in FPT time [44]. In fact, our algorithm implies that even when Γ is not a fixed constraint language, CSP is FPT when parameterized jointly by the maximum arity of the relations in Γ, the size of the domain *and* the backdoor-treewidth with respect to Γ.

Two separate problems need to be dealt with in order to use backdoor-treewidth for solving constraint satisfaction: finding a strong backdoor of small treewidth, and then using it to actually solve the CSP instance. The latter task can be solved efficiently by a dynamic programming procedure on a tree-decomposition. However, finding strong backdoors of small treewidth is considerably more complicated and forms the main technical contribution of this article. We note in particular that algorithms for finding small backdoors to tractable classes cannot be used for this purpose, since the size of the backdoors we are interested in can be very large. In fact, it is even far from obvious that we can detect a backdoor of treewidth at most k in polynomial time when k is considered a constant (and the order of the polynomial may depend on k).

Our result on backdoor-treewidth also carries over to the counting variant of CSP (#CSP). #CSP is a prominent #P-complete extension of CSP problem which asks for the number of variable assignments that satisfy the given constraints. Structural restrictions as well as language restrictions have been studied for #CSP. The dynamic programming algorithm for CSP for instances of bounded primal treewidth can be readily adapted to #CSP (see, e.g., [21]). A constraint language Γ is *#-tractable* if #CSP(Γ) (#CSP restricted to instances whose constraints use only relations from Γ) can be solved in polynomial time. Bulatov [8] characterized all finite *#-tractable* constraint languages. Applying our results, we obtain the following corollary.

▶ **Corollary 2.** *Let Γ be a fixed #-tractable finite constraint language. Then,* #CSP *parameterized by the backdoor-treewidth with respect to Γ is* FPT.

(a) In the first part of our algorithm to detect strong backdoors of small treewidth, we define a notion of *boundaried* CSP instances in the spirit of boundaried graphs and show that for any $t, k \in \mathbb{N}$, there is an equivalence relation $\sim_{t,k}$ on the set of all t-boundaried CSP instances such that (i) this relation has at most $f(k,t)$ equivalence classes for some function f (all functions used in the paper can be easily seen to be computable) depending only on k and the constraint language Γ, and (ii) for any two t-boundaried CSP instances in the same equivalence class of $\sim_{t,k}$, they 'interact in the same way' with every other t-boundaried CSP-instance.

(b) We then describe an algorithm that for any given $t, k \in \mathbb{N}$ runs in time $\mathcal{O}(g(t,k))$ for some function g and actually *constructs* a set $\mathfrak{H}$ of $f(k,t)$ CSP instances, one from each equivalence class of the relation $\sim_{t,k}$. Additionally, we show that each instance in this set has size bounded by a function of k and t.

(c) In this part, we show that for any given t-boundaried CSP instance $\mathbf{I}$ whose size exceeds a certain bound depending on k and t and whose incidence graph satisfies certain connectivity properties, we can in time $g(t,k)|\mathbf{I}|^{\mathcal{O}(1)}$ correctly determine the equivalence class that this instance belongs to and compute a strictly smaller t-boundaried CSP instance $\mathbf{I}'$ which belongs to the same equivalence class of $\sim_{t,k}$ as $\mathbf{I}$. It follows that once $\mathbf{I}'$ is computed, we can 'replace' $\mathbf{I}$ with the strictly smaller $\mathbf{I}'$, without altering the existence (or non-existence) of a strong backdoor of small treewidth. Our replacement framework is inspired by the graph replacement tools dating back to the results of Fellows and Langston [23] and further developed by Arnborg, Bodlaender, and other authors [1, 3, 5, 19, 4].

(d) In this part, we utilize the *recursive-understanding* technique, introduced by Grohe et al. [32] to solve the Topological Subgraph Containment problem and used with great success in the design of FPT algorithms for several other fundamental graph problems (see [36, 12]), to recursively compute a t-boundaried subinstance with the properties required to execute Part **(c)**. Once this process terminates, we have an instance whose size is upper-bounded by a function of k and t which can be solved by brute force.

Related Work. Williams et al. [47, 48] introduced the notion of *backdoors* for the runtime analysis of algorithms for CSP and SAT, see also [35] for a more recent discussion of backdoors for SAT. A backdoor is a small set of variables whose instantiation puts the instance into a fixed tractable class (called the base class). One distinguishes between strong and weak backdoors, where for the former all instantiations lead to an instance in the base class, and for the latter at least one leads to a satisfiable instance in the base class. Backdoors have been studied under a different name by Crama et al. [16]. The study of the parameterized complexity of finding small backdoors was initiated by Nishimura et al. [42] for SAT, who considered backdoors into the classes of Horn and Krom CNF formulas. Further results cover the classes of renamable Horn formulas [43], q-Horn formulas [28] and classes of formulas of bounded treewidth [29, 24]. The detection of backdoors for CSP has been studied in several works [2, 11]. Gaspers et al. [27] obtained results on the detection of strong backdoors into *heterogeneous* base classes of the form $\mathrm{CSP}(\Gamma_1) \cup \cdots \cup \mathrm{CSP}(\Gamma_d)$ where for each instantiation of the backdoor variables, the reduced instance belongs entirely to some $\mathrm{CSP}(\Gamma_i)$ (possibly to different $\mathrm{CSP}(\Gamma_i)$'s for different instantiations). This direction was recently further generalized by Ganian et al. [26] by developing a framework for detecting strong backdoors into so-called *scattered* base classes with respect to $\Gamma_1 \ldots \Gamma_d$; there, each instantiation of the backdoor variables results in a reduced instance whose every connected component belongs entirely to some $\mathrm{CSP}(\Gamma_i)$ (possibly to different $\mathrm{CSP}(\Gamma_i)$'s for different components and different instantiations).

2 Preliminaries

We use standard graph terminology, see for instance the handbook by Diestel [20]. For $i \in \mathbb{N}$, we use $[i]$ to denote the set $\{1, \ldots, i\}$.

Constraint Satisfaction. Let $\mathcal{V}$ be a set of variables and $\mathcal{D}$ a finite set of values. A *constraint of arity ρ over $\mathcal{D}$* is a pair (S, R) where $S = (x_1, \ldots, x_\rho)$ is a sequence of variables from $\mathcal{V}$ and $R \subseteq \mathcal{D}^\rho$ is a ρ-ary relation. The set $\mathrm{var}(C) = \{x_1, \ldots, x_\rho\}$ is called the *scope* of C. An *assignment* $\alpha : X \to \mathcal{D}$ is a mapping of a set $X \subseteq \mathcal{V}$ of variables. An assignment $\alpha : X \to \mathcal{D}$ *satisfies* a constraint $C = ((x_1, \ldots, x_\rho), R)$ if $\mathrm{var}(C) \subseteq X$ and $(\alpha(x_1), \ldots, \alpha(x_\rho)) \in R$. For a set $\mathbf{I}$ of constraints we write $\mathrm{var}(\mathbf{I}) = \bigcup_{C \in \mathbf{I}} \mathrm{var}(C)$ and $\mathrm{rel}(\mathbf{I}) = \{\, R : (S, R) \in C, C \in \mathbf{I} \,\}$.

A finite set $\mathbf{I}$ of constraints is *satisfiable* if there exists an assignment that simultaneously satisfies all the constraints in $\mathbf{I}$. The *Constraint Satisfaction Problem* (CSP, for short) asks, given a finite set $\mathbf{I}$ of constraints, whether $\mathbf{I}$ is satisfiable. The *Counting Constraint Satisfaction Problem* (#CSP, for short) asks, given a finite set $\mathbf{I}$ of constraints, to determine the number of assignments to $\mathrm{var}(\mathbf{I})$ that satisfy $\mathbf{I}$. CSP is NP-complete and #CSP is #P-complete (see, e.g., [8]).

Let $\alpha : X \to \mathcal{D}$ be an assignment. For a ρ-ary constraint $C = (S, R)$ with $S = (x_1, \ldots, x_\rho)$ and $R \subseteq \mathcal{D}^\rho$, we denote by $C|_\alpha$ the constraint (S', R') obtained from C as follows. R' is obtained from R by (i) deleting all tuples $(d_1, \ldots, d_\rho)$ from R for which there is some $1 \leq i \leq \rho$ such that $x_i \in X$ and $\alpha(x_i) \neq d_i$, and (ii) removing from all remaining tuples all coordinates d_i with $x_i \in X$. S' is obtained from S by deleting all variables x_i with $x_i \in X$. For a set $\mathbf{I}$ of constraints we define $\mathbf{I}|_\alpha$ as $\{\, C|_\alpha : C \in \mathbf{I} \,\}$.

A *constraint language* (or *language*, for short) Γ over a domain $\mathcal{D}$ is a set of relations (of possibly various arities) over $\mathcal{D}$. By $\mathrm{CSP}(\Gamma)$ we denote CSP restricted to instances $\mathbf{I}$ with $\mathrm{rel}(\mathbf{I}) \subseteq \Gamma$. A constraint language is *tractable* if for every finite subset $\Gamma' \subseteq \Gamma$, the problem $\mathrm{CSP}(\Gamma')$ can be solved in polynomial time. A constraint language is *#-tractable* if for every finite subset $\Gamma' \subseteq \Gamma$, the problem $\#\mathrm{CSP}(\Gamma')$ can be solved in polynomial time. Throughout this paper, we make the technical assumption that every considered tractable or #-tractable constraint language Γ contains the redundant tautological relation of arity 2; note that if this is not the case, then this relation can always be added into Γ and the resulting language will still be tractable or #-tractable, respectively. Let Γ be a constraint language and $\mathbf{I}$ be an instance of CSP. A variable set X is a *strong backdoor* to $\mathrm{CSP}(\Gamma)$ if for each assignment $\alpha : X \to \mathcal{D}$ it holds that $\mathbf{I}|_\alpha \in \mathrm{CSP}(\Gamma)$.

The *primal graph* of a CSP instance $\mathbf{I}$ is the graph whose vertices correspond to the variables of $\mathbf{I}$ and where two variables a, b are adjacent iff there exists a constraint in $\mathbf{I}$ whose scope contains both a and b. The *incidence graph* of a CSP instance $\mathbf{I}$ is the bipartite graph whose vertices correspond to the variables and constraints of $\mathbf{I}$, and where vertices corresponding to a variable x and a constraint C are adjacent if and only if $x \in \mathrm{var}(C)$. Observe that an incidence graph does not uniquely define a CSP instance; however, in this paper the CSP instance from which a graph is obtained will always be clear from the context. Hence for an incidence or primal graph G we will denote the corresponding CSP instance by $\psi(G)$. Furthermore, we slightly abuse the notation and use $\mathcal{V}(G)$ to refer to the vertices of G that correspond to variables in $\psi(G)$, and $\mathcal{C}(G)$ to refer to the vertices of G that correspond to constraints in $\psi(G)$. Also, for a vertex subset $X \subseteq V(G)$, we continue to use the notations $\mathcal{V}(X)$ and $\mathcal{C}(X)$ to refer to the sets $\mathcal{V}(G) \cap X$ and $\mathcal{C}(G) \cap X$, respectively.

Treewidth. Let G be a graph. A *tree decomposition* of G is a pair $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ where T is a tree and $\mathcal{X}$ is a collection of subsets of $V(G)$ such that: (1) for each edge $e = uv \in E(G)$ there exists some $t \in V(T)$ such that $\{u, v\} \subseteq X_t$, and (2) for each vertex $v \in V(G)$, $T[\{t \mid v \in X_t\}]$ is a non-empty connected subtree of T. We call the vertices of T *nodes* and the sets in $\mathcal{X}$ *bags* of the tree decomposition $(T, \mathcal{X})$. The *width* of $(T, \mathcal{X})$ is equal to $\max\{|X_t| - 1 \mid t \in V(T)\}$ and the *treewidth* of G (denoted $tw(G)$) is the minimum width over all tree decompositions of G. The primal treewidth of a CSP instance $\mathbf{I}$ is the treewidth of its primal graph, and similarly the incidence treewidth of $\mathbf{I}$ is the treewidth of its incidence graph. We note that if the constraints have bounded arity, then any class of CSP instances has bounded primal treewidth if and only if it has bounded incidence treewidth [45].

▶ **Proposition 3** ([38]). *Let* $\mathbf{I}$ *be a* CSP *instance where the constraints have arity bounded by* $\rho \in \mathbb{N}$*. Then, the primal treewidth of the instance is at most* $\rho(t+1) - 1$ *where* t *is the incidence treewidth of the instance.*

***t*-Boundaried CSP Instances.** A *t-boundaried graph* is a graph G with a set $Z \subset V(G)$ of size at most t with each vertex $v \in Z$ having a unique label $\ell(v) \in \{1, \ldots, t\}$. We refer to Z as the *boundary* of G. For a t-boundaried graph G, $\delta(G)$ denotes the boundary of G. When it is clear from the context, we will often use the notation (G, Z) to refer to a t-boundaried graph G with boundary Z. For $P \subseteq [t]$, we use $P(G, Z)$ to denote the subset of Z with labels in P; for $i \in [t]$ we use $i(G, Z)$ instead of $\{i\}(G, Z)$ for brevity. Two t-boundaried graphs G_1 and G_2 can be 'glued' together to obtain a new graph, which we denote by $G_1 \oplus G_2$. The gluing operation takes the disjoint union of G_1 and G_2 and identifies the vertices of $\delta(G_1)$ and $\delta(G_2)$ with the same label.

In some cases, we will also use a natural notion of replacement of boundaried graphs. Let (G_1, Z_1) be a t-boundaried graph which is an induced subgraph of a graph G such that Z_1 is a separator between $V(G_1) \setminus Z_1$ and $V(G) \setminus V(G_1)$. Let (G_2, Z_2) be a t-boundaried graph. Then the operation of *replacement* of (G_1, Z_1) by (G_2, Z_2) results in the graph $G' = (G[V(G) \setminus (V(G_1) \setminus Z_1)], Z_1) \oplus (G_2, Z_2)$. Furthermore, if G was a j-boundaried graph with boundary Z and $Z \cap V(G_1) \subseteq Z_1$, then the resulting graph G' is also a j-boundaried graph with the same boundary.

In this paper, it will sometimes be useful to lift the notions of boundaries and gluing from graphs to CSP instances. A *t-boundaried incidence graph* of a CSP instance $\mathbf{I}$ is a t-boundaried graph G with boundary Z such that G is the incidence graph of $\mathbf{I}$ and $Z \subseteq \mathcal{V}$. Similarly, we call a CSP instance $\mathbf{I}$ with t uniquely labeled variables a *t-boundaried CSP instance*. Note that boundaried incidence graphs and boundaried CSP instances are de-facto interchangeable, but in some cases it is easier to use one rather than the other due to technical reasons.

The gluing operations of boundaried incidence graphs and boundaried CSP instances are defined analogously as for standard boundaried graphs. Observe that if G_1 and G_2 are t-boundaried incidence graphs of $\mathbf{I}_1$ and $\mathbf{I}_2$, respectively, then $G_1 \oplus G_2$ is also an incidence graph; furthermore, $\psi(G_1 \oplus G_2)$ is well-defined and can be reconstructed from $\mathbf{I}_1$ and $\mathbf{I}_2$.

3 Backdoor-Treewidth

In this section we give a formal definition of the notion of backdoor-treewidth.

▶ **Definition 4.** Let G be a graph and $X \subseteq V(G)$. We denote by $\mathbf{Torso}_G(X)$ the following graph defined over the vertex set X. For every pair of vertices $x_1, x_2 \in X$, we add the edge

(x_1, x_2) if (a) $(x_1, x_2) \in E(G)$ or (b) x_1 and x_2 both have a neighbor in the same connected component of $G - X$. That is, we begin with $G[X]$ and make the neighborhood of every connected component of $G - X$, a clique. When G is an incidence graph of the instance $\mathbf{I}$ and X is a set of variables of $\mathbf{I}$, we also refer to $\mathbf{Torso}_G(X)$ as $\mathbf{Torso}_\mathbf{I}(X)$.

▶ **Definition 5.** Let $\mathcal{F}$ be a class of CSP instances and $\mathbf{I}$ be a CSP instance. Then the *backdoor-treewidth* of $\mathbf{I}$ with respect to $\mathcal{F}$, denoted $\mathbf{btw}_\mathcal{F}(\mathbf{I})$, is the smallest value of $tw(\mathbf{Torso}_\mathbf{I}(X))$ taken over all strong backdoors X of $\mathbf{I}$ into $\mathcal{F}$. If $\mathcal{F} = \mathrm{CSP}(\Gamma)$ for some constraint language Γ, then we call $\mathbf{btw}_\mathcal{F}$ the backdoor-treewidth with respect to Γ.

As an example, observe that in Figure 1 the graph $\mathbf{Torso}_G(X)$ is a path. Throughout this paper, we sometimes refer to backdoors of small treewidth simply as backdoors of small *width*. Next, we show how backdoors of small treewidth can be used to solve CSP and #CSP.

▶ **Lemma 6.** *Let $\mathbf{I}$ be a CSP instance over domain $\mathcal{D}$ and X be a strong backdoor of $\mathbf{I}$ to the class $\mathcal{F}$. There is an algorithm that, given $\mathbf{I}$ and X, runs in time $\mathcal{O}(|\mathcal{D}|^{tw(\mathbf{Torso}(X))}|\mathbf{I}|^{\mathcal{O}(1)})$ and correctly decides whether $\mathbf{I}$ is satisfiable or not. Furthermore, if $\mathcal{F}$ is #-tractable and X is a strong backdoor to $\mathcal{F}$, then in the same time bound one can count the number of satisfying assignments of $\mathbf{I}$.*

Sketch of Proof. The algorithm is a standard dynamic programming procedure over a bounded treewidth graph and hence we only sketch it briefly. Let G denote the incidence graph of $\mathbf{I}$ and let H denote the graph $\mathbf{Torso}(X)$ and let $(T, \mathcal{X})$ be a tree-decomposition of H of width $tw(H)$. Now, for every $v \in T$, we define the instance $\mathbf{I}_v$ as the subinstance of $\mathbf{I}$ induced on the variables in X_v, the bags below it in $(T, \mathcal{X})$, and the constraints whose scope is completely contained in the union of X_v and the bags below it. The key observation is that for any connected component of $G - X$, there is a vertex $v \in V(T)$ such that the bag X_v contains the neighbors of this component.

To solve CSP, for each $v \in T$ we define a function τ_v which maps assignments of the variables in X_v to 0 to 1. Let $\gamma : X_v \to \mathcal{D}$ be an assignment to the variables in X_v. We define $\tau_v(\gamma) = 1$ if there is a satisfying assignment for $\mathbf{I}_v$ that extends γ and $\tau_v(\gamma) = 0$ otherwise. Let v^* denote the root of T. Clearly, the instance $\mathbf{I}$ is satisfiable if and only if there is a $\gamma : X_{v^*} \to \mathcal{D}$ such that $\tau_{v^*}(\gamma) = 1$. At this point it suffices to describe how to dynamically compute the function τ_v for each node in the tree-decomposition; this step can be facilitated by the use of so-called nice tree-decompositions. The algorithm to solve #CSP is similar; there τ_v is extended by information about how many ways there are to extend an assignment to the variables in X_v to a satisfying assignment for $\mathbf{I}_v$. ◀

As the width of a backdoor can be arbitrarily smaller than its size, the width provides a much better measure of how far away an instance is from a tractable base class. In particular, the width lower-bounds both the primal treewidth and the backdoor size. We formalize this below.

▶ **Proposition 7.** *Let $\mathbf{I}$ be a CSP instance and $\mathcal{F}$ be a class of CSP instances. Let q be the primal treewidth of $\mathbf{I}$ and r be the minimum size of a strong backdoor to $\mathcal{F}$ in $\mathbf{I}$. Then $\mathbf{btw}_\mathcal{F}(\mathbf{I}) \leq \min(q, r)$.*

In order to prove Theorem 1, we give an FPT algorithm for the problem of finding strong backdoors parameterized by their width (formalized below). We note that since we state our results in as general terms as possible, the dependence on k is likely to be sub-optimal for specific languages and could be improved using properties specific to each language.

Width Strong-CSP(Γ) Backdoor Detection	**Parameter:** k
Input: CSP instance $\mathbf{I}$, integer k.	
Objective: Return a set X of variables such that X is a strong backdoor of $\mathbf{I}$ to CSP(Γ) of width at most k or correctly conclude that no such set exists.	

The main technical content of the article then lies in the proof of the following theorem.

▶ **Theorem 8.** Width Strong-CSP(Γ) Backdoor Detection *is* FPT *for every finite* Γ.

Before we proceed to the description of the algorithms, we state the following simple and obvious preprocessing routine (correctness is argued in the full version of this paper, available at `https://arxiv.org/abs/1610.03298`) which will allow us to infer certain structural information regarding interesting instances of this problem.

▶ **Reduction Rule 9.** *Given a CSP instance* $\mathbf{I}$ *and an integer* k *as an instance of* Width Strong-CSP(Γ) Backdoor Detection*, if there is a constraint in* $\mathbf{I}$ *of arity at least* $p + k + 2$ *where* p *is the maximum arity of a relation in* Γ*, then return* NO*.*

4 The Finite State Lemma

In this section, we prove that the problem Width Strong-CSP(Γ) Backdoor Detection has finite state; this will allow us to construct a finite set of bounded-size representatives (Section 5) which will play a crucial role in the proof of Theorem 8 (Section 6). Let Γ be a finite constraint language; throughout the rest of the paper, we work with this fixed constraint language. We begin by defining a relation over the set of boundaried incidence graphs.

▶ **Definition 10.** Let $k, t \in \mathbb{N}$ and let (G_1, Z_1) and (G_2, Z_2) be t-boundaried incidence graphs of CSP instances $\mathbf{I}_1$ and $\mathbf{I}_2$ with boundaries Z_1 and Z_2 respectively. Then, we say that $(\mathbf{I}_1, Z_1) \sim_{t,k} (\mathbf{I}_2, Z_2)$ (or $(G_1, Z_1) \sim_{t,k} (G_2, Z_2)$) if for every t-boundaried CSP instance $\mathbf{I}_3$ with incidence graph G_3, the instance $\psi(G_1 \oplus G_3)$ has a strong backdoor set of width at most k into CSP(Γ) if and only if the instance $\psi(G_2 \oplus G_3)$ has a strong backdoor set of width at most k into CSP(Γ).

It is clear that $\sim_{t,k}$ is an equivalence relation. Generally speaking, the high-level goal of this section is to prove that $\sim_{t,k}$ has finite index. This is achieved by introducing a second, more technical equivalence $\equiv_{t,h,\varepsilon}$ which captures all the information about how a t-boundaried incidence graph (G, Z) contributes to the (non)-existence of a strong backdoor of small width after gluing. Observe that for a set X which has vertices from 'both' sides of a boundary the graph $\mathbf{Torso}(X)$ may have edges crossing this boundary. Since we need to take this behaviour into account, proving this lemma is in fact much more involved than might be expected at first glance.

To define $\equiv_{t,h,\varepsilon}$, we will first need the notion of a *configuration*, which can be thought of as one possible way a t-boundaried graph can interact via gluing; this is then tied to the notion of a *realizable configuration*, which is a configuration that actually can occur in the graph (G, Z). We let $(G_1, Z_1) \equiv_{t,h,\varepsilon} (G_1, Z_1)$ if and only if both boundaried graphs have the same set of realizable configurations. Before we proceed to the technical definition of a configuration, we need one more bit of notation. Since we will often be dealing with labeled minors, we fix a pair of symbols $\square$ and $\Diamond$ and express all relevant label sets using these symbols. Specifically, for $r, s \in \mathbb{N}$ we let $\mathfrak{L}(r, s)$ denote the set $2^{\{\square_1, \ldots, \square_r\} \bigcup \{\Diamond_1, \ldots, \Diamond_s\}}$.

▶ **Definition 11.** Let $h, t \in \mathbb{N}$. A **(t, h)-configuration** is a tuple $(P, w, w', \mathcal{P}, \mathcal{P}', \gamma, \mathfrak{H})$, where:

1. P is a subset of $[t]$,
2. $w, w' \in \mathbb{N}$ and $w' \leq (w+1)t$,
3. $\mathcal{P} = \{Q_1, \ldots, Q_r\}$ is a partition of $[t] \setminus P$,
4. $\mathcal{P}' \in 2^{\binom{P}{2}} \times 2^{\binom{[w']}{2}} \times 2^{P \times [w']}$,
5. $\gamma : \mathcal{P} \to 2^P \times 2^{[w']}$,
6. $\mathfrak{H}$ is a collection of labeled graphs on at most h vertices where the label set is $\mathfrak{L}(t, w')$.

For a set $Q \in \mathcal{P}$ with $\gamma(Q) = (J_1, J_2)$, we denote by $\gamma^i(Q)$ the set J_i for each $i \in \{1, 2\}$. A (t, h)-configuration $(P, w, w', \mathcal{P}, \mathcal{P}', \gamma, \mathfrak{H})$ is called a **(t, h, ε)-configuration** if $w \leq \varepsilon$ and we denote the set of such (t, h)-configurations by $\mathfrak{S}(t, h, \varepsilon)$.

Let us informally break down the intuition behind the above definition. t corresponds to the size of the boundary of the associated t-boundaried incidence graph (as we will see in the next definition), and h is an upper bound on the size of forbidden minors for our target treewidth. The (t, h)-configuration then captures the following information about interactions between a t-boundaried incidence graph (G_1, Z_1) and a potential solution after gluing:

- **(a)** P represents the part of the boundary that intersects a backdoor of small width,
- **(b)** w' represents neighbors of the remainder of the boundary outside of G_1,
- **(c)** w represents the target treewidth of the torso,
- **(d)** $\mathcal{P}$ represents how the part of the boundary outside of the strong backdoor will be partitioned into connected components, i.e., how it will 'collapse' into the torso,
- **(e)** $\mathcal{P}'$ represents all the new edges that will be created in the torso due to collapsing of parts outside of the torso,
- **(f)** γ represents connections between connected components in the boundary outside of the strong backdoor and relevant variables in the strong backdoor, which is the second part of information needed to encode the collapse of these components into the torso,
- **(g)** $\mathfrak{H}$ represents 'parts' of all minors of size at most h present in the torso inside G_1.

In order to formally capture the intuition outlined above, we define the result of 'applying' a configuration on a t-boundaried incidence graph.

▶ **Definition 12.** Let $h, t \in \mathbb{N}$, (G, Z) be the t-boundaried incidence graph of a t-boundaried CSP instance $\mathbf{I}$ and $\omega = (P, w, w', \mathcal{P}, \mathcal{P}', \gamma, \mathfrak{H})$ be a (t, h)-configuration. We associate with G and ω an incidence graph G^ω which is defined as follows. We begin with the graph G, add w' new variables $l^\omega_1, \ldots, l^\omega_{w'}$, denoting the set comprising these vertices by L_ω. For every $J \subseteq [w']$, we denote by $J(L_\omega)$ the set $\{l^\omega_i | i \in J\}$. For each $Q \in \mathcal{P}$, let $(J^Q_1, J^Q_2) = \gamma(Q)$ and add $|Q| - 1$ redundant binary constraints $C^Q_1, \ldots, C^Q_{|Q|-1}$ (we have assumed that Γ also contains a tautological relation of arity 2) and connect these with the variables in $Q(G, Z)$ to form a path which alternates between a vertex/variable in $Q(G, Z)$ and a vertex/variable in $\{C^Q_1, \ldots, C^Q_{|Q|-1}\}$. Following this, for every variable u in $J_1(G, Z) \cup J_2(L_\omega)$, we add a redundant binary constraint C_u and set $\mathrm{var}(C_u)$ as u and an arbitrary variable in $Q(G, Z)$. This completes the definition of G^ω. We also define the graph $\tilde{G}^\omega$ as the graph obtained from G^ω by doing the following. Let $\mathcal{P}' = (X_1, X_2, X_3)$ where $X_1 \subseteq \binom{P}{2}$, $X_2 \subseteq \binom{[w']}{2}$ and $X_3 \subseteq P \times [w']$. For every pair $(i, j) \in X_1$, we add the edge $(i(G, Z), j(G, Z))$. Similarly, for every pair $(i, j) \in X_2$, we add the edge (l^ω_i, l^ω_j). Finally, for every pair $(i, j) \in X_3$, we add the edge $(i(G, Z), l^\omega_j)$. This completes the description of $\tilde{G}^\omega$.

The graph G^ω defined above can be seen as an enrichment of G by (1) adding strong backdoor variables which will be affected by a collapse of the boundary into the torso

$(l^\omega_1, \ldots, l^\omega_{w'})$ and (2) enforcing the assumed partition of part of the boundary into connected components (as per $\mathcal{P}$) and (3) adding connections of these components both into the rest of the boundary and vertices l^ω_i (as per γ). The graph $\tilde{G}^\omega$ is then an extension of G^ω by edges which will be created in the torso. Note that while G^ω is an incidence graph, $\tilde{G}^\omega$ is not necessarily a bipartite graph.

With $\tilde{G}^\omega$ in hand, we can finally formally determine whether the information contained in a given configuration is of any relevance for the given graph. This is achieved via the notion of *realizability*.

▶ **Definition 13.** Let $h, t \in \mathbb{N}$, (G, Z) be the t-boundaried incidence graph corresponding to a t-boundaried CSP instance $\mathbf{I}$ and let $\omega = (P, w, w', \mathcal{P}, \mathcal{P}', \gamma, \mathfrak{H})$ be a (t, h)-configuration. We say that ω is a **realizable** configuration in (G, Z) if, and only if, there is a subset $S^* \subseteq \mathcal{V}(G)$ with the following properties:

1. $S^* \cap Z = P(G, Z)$,
2. $tw(\mathbf{Torso}_{\tilde{G}^\omega}(S^* \cup L_\omega))$ is at most w,
3. $\mathfrak{H}$ is precisely the set of all labeled minors of $(\mathbf{Torso}_{\tilde{G}^\omega}(S^* \cup L_\omega), \Lambda_\omega)$ with at most h vertices,
4. S^* is a strong backdoor of $\psi(G)$ into CSP(Γ).

If the above conditions hold, we say that S^* **realizes** ω in (G, Z).

We let $\mathfrak{S}((G, Z), h, \varepsilon)$ denote the set of all realizable $(|Z|, h, \varepsilon)$-configurations in (G, Z). We ignore the explicit reference to Z in the notation if it is clear from the context. We let $h^*(k)$ denote the upper bound on the size of forbidden minors for graphs of treewidth at most k given in [39]. For technical reasons, we will be in fact concerned with minors of size slightly greater than $h^*(k)$, and hence for $t \in \mathbb{N}$ we set $h^*(k, t) = h^*(k) + t \cdot (k + 1)$.

We use $\Upsilon(t, h, \varepsilon)$ to denote a computable upper bound on the number of (t, h, ε)-configurations. Observe that setting $\Upsilon(t, h, \varepsilon) = 2^t \cdot \varepsilon \cdot \varepsilon t \cdot t^t \cdot 2^{\binom{t}{2}} \cdot 2^{\binom{(\varepsilon+1)t}{2}} \cdot 2^{t^2(\varepsilon+1)} \cdot 2^{t^2(\varepsilon+1) \cdot 2^{\binom{h}{2}} h^{2(\varepsilon+1)t}}$ is sufficient. We now give the formal definition of the refined equivalence relation.

▶ **Definition 14.** Let $t, h \in \mathbb{N}$ and let $(\mathbf{I}_1, Z_1)$ and $(\mathbf{I}_2, Z_2)$ be t-boundaried CSP instances with t-boundaried incidence graphs (G_1, Z_1) and (G_2, Z_2) respectively. Then, $(\mathbf{I}_1, Z_1) \equiv_{t,h,\varepsilon} (\mathbf{I}_2, Z_2)$ (or $(G_1, Z_1) \equiv_{t,h,\varepsilon} (G_2, Z_2)$) if $\mathfrak{S}((G_1, Z_1), h, \varepsilon) = \mathfrak{S}((G_2, Z_2), h, \varepsilon)$.

From these definitions, it is straightforward to verify that $\equiv_{t,h,\varepsilon}$ is indeed an equivalence and the number of equivalence classes induced by this relation over the set of all t-boundaried incidence graphs is at most $2^{\Upsilon(t,h,\varepsilon)}$. The main lemma of this section, Lemma 15, then links $\equiv_{t,h,\varepsilon}$ to $\sim_{t,k}$, and in particular shows that the former is a refinement of the latter. We note that the more refined $\equiv_{t,h,\varepsilon}$ is used throughout the paper; it is not merely a tool for showing finite-stateness of $\sim_{t,k}$.

▶ **Lemma 15.** *Let $k, t \in \mathbb{N}$ and let (G_1, Z_1), (G_2, Z_2) be two t-boundaried incidence graphs satisfying $(G_1, Z_1) \equiv_{t,h^*(k,t),k} (G_2, Z_2)$. Then, $(G_1, Z_1) \sim_{t,k} (G_2, Z_2)$.*

5 Computing a Bound on the Size of a Minimal Representative of $\sim_{t,k}$

In this section, we define a function α such that for every $t, k \in \mathbb{N}$, every equivalence class of $\sim_{t,k}$ contains a boundaried incidence graph whose size is bounded by $\alpha(t, k)$. In order to do so, we use the fact the relation $\equiv_{t,h^*(k,t),k}$ refines $\sim_{t,k}$. The following is a brief sketch of the proof strategy.

- In the first step, we show that for any t-boundaried incidence graph (G, Z) whose *treewidth* is bounded as a function of t and k and size exceeds a certain bound also depending only on t and k, there is a strictly smaller t-boundaried graph (G', Z') such that $(G, Z) \equiv_{t,h^*(k,t),k} (G', Z')$. This in turn implies that for any t-boundaried incidence graph (G, Z) whose *treewidth* is bounded by a function of t and k there is a t-boundaried graph (G', Z') such that $(G, Z) \equiv_{t,h^*(k,t),k} (G', Z')$ and the size of G' is bounded by a function of t and k.
- In the second step, we show that for any t-boundaried incidence graph (G, Z), there is a t-boundaried incidence graph (G', Z') such that G' has treewidth bounded by a function of k and t and $(G, Z) \sim_{t,k} (G', Z')$. Combining these two steps, we obtain the following lemma.

▶ **Lemma 16.** *Let $k, t \in \mathbb{N}$. There exists a computable function $\mathfrak{y}(t, k)$ and a set $\mathfrak{F}_s(t, h^*(k, t), k)$ of at most $\mathfrak{y}(t, k)$ t-boundaried CSP instances that contains a t-boundaried CSP instance from every equivalence class of $\sim_{t,k}$. Furthermore, given k and t, the set $\mathfrak{F}_s(t, h^*(k, t), k)$ can be computed in time $\mathcal{O}(|\mathfrak{F}_s(t, h^*(k, t), k)|)$.*

6 The FPT Algorithm for Width Strong-CSP(Γ) Backdoor Detection

An often-used approach in the design of FPT algorithms for graph problems is that of finding a sufficiently small separator in the graph and then reducing one of the sides. In the technique of 'recursive understanding' introduced by Grohe et al. [32], this is achieved by performing this step *recursively* until we arrive at a separator where the side we want to reduce has certain connectivity-based structure using which we can find a way reduce it without recursing further. This approach has been combined with various problem specific reduction rules at the bottom to obtain parameterized algorithms for several well-studied problems. These include the k-way Cut problem, solved by Kawarabayashi and Thorup [36], Steiner Cut and Unique Label Cover – both solved by Chitnis et al. [12]. In this section, we will employ this technique to design our algorithm for Width Strong-CSP(Γ) Backdoor Detection. We begin by defining a notion of *nice* instances which basically capture the kind of instances we will be dealing with at the bottom of our recursion.

6.1 Solving Nice Instances

▶ **Lemma 17.** *There is a function $\mathfrak{Z} : \mathbb{N}^2 \to \mathbb{N}$ and an algorithm that, given a CSP instance $\mathbf{I}$ with incidence graph G and positive integers $\beta, k \in \mathbb{N}$, runs in time $\mathcal{O}(\mathfrak{Z}(\beta, k)|G|^2)$ and either computes a strong backdoor into* CSP(Γ) *of width at most k or correctly concludes that $\mathbf{I}$ has no backdoor set X of width at most k that satisfies the following properties:*
1. *$G - X$ has exactly one connected component C of size at least $\beta + 1$.*
2. *$|V(G) \setminus N[C]| \leq \beta$*

We now give the definition of 'nice' instances. Generally speaking, these are instances which fall into either the bounded 'classical' treewidth case or bounded backdoor size case.

▶ **Definition 18.** Let $k, \beta \in \mathbb{N}$ and $\mathbf{I}$ be a CSP instance with incidence graph G. We say that $\mathbf{I}$ is (β, k)-*nice* if $tw(G) \leq \beta + k$ or if $\mathbf{I}$ has some strong backdoor set of width at most k, then it also has a strong backdoor set X of width at most k such that $G - X$ has exactly one connected component C of size at least $\beta + 1$, and $|V(G) \setminus N[C]| \leq \beta$.

We now formally show that given a (β, k)-nice incidence graph, one can detect strong backdoor sets of small width in FPT time parameterized by $\beta + k$. This will later be used to compute small representatives of large boundaried CSP instances (specifically, in Lemma 23).

▶ **Lemma 19.** *There is a function $\hat{\mathfrak{X}} : \mathbb{N} \to \mathbb{N}$ and an algorithm that, given $\beta, k \in \mathbb{N}$, a (β, k)-nice CSP instance $\mathbf{I}$ with the incidence graph G, runs in time $\mathcal{O}(\hat{\mathfrak{X}}(\beta + k)|G|^2)$ and either computes a strong backdoor set into $\mathrm{CSP}(\Gamma)$ of width at most k or correctly detects that such a set does not exist.*

Proof. If $tw(G) \leq \beta + k$, then we can solve the problem directly by applying Courcelle's Theorem [15], as follows. First, recall that the arity of any constraint which appears in the CSP instance $\psi(G)$ is upper-bounded by k plus the maximum arity of relations in Γ. Hence we can assume that the number of relations which appear in the constraints of $\psi(G)$ is bounded by a function of k, and we can think of G as having vertex labels which specify which relation is used in each constraint vertex and edge labels which specify the order in which variables appear in the incident constraint. Second, for a j-ary relation R which appears in a constraint C in $\psi(G)$, we say that a subset α of $\{1, \ldots, j\}$ is a *valid choice* if the variables occurring in positions α in C form a strong backdoor for $\{C\}$ into $\mathrm{CSP}(\Gamma)$. Note that the set of valid choices for all of the relations which occur in a constraint in $\psi(G)$ can be precomputed in advance. Then the problem can be formulated in Monadic Second Order logic with a sentence stating the following: there exists a set T of variables such that (1) for each constraint C with label R it holds that the edges between T and C correspond to a valid choice for R, and (2) the torso of T does not contain any of the forbidden minors for treewidth at most k. Indeed, condition (1) ensures that T forms a backdoor to $\mathrm{CSP}(\Gamma)$ and condition (2) ensures that T has width at most k.

Otherwise, we execute the algorithm of Lemma 17 that runs in time $\mathcal{O}(\hat{\mathfrak{Z}}(\alpha)|G|^2)$. The function $\mathfrak{X}$ is obtained from the function $\mathfrak{Z}$ and the dependence of the algorithm on $\beta + k$ in the case of bounded treewidth. ◀

6.2 Computing a Minimal Representative

In this subsection, we show that if a t-boundaried instance has a certain guarantee on the (non-)existence of a small separator separating two large parts of the instance from each other, then we can compute a t-boundaried instance of bounded size which is equivalent to it under the relation $\sim_{t,k}$.

▶ **Definition 20.** Let G be an incidence graph and (A, S, B) be a partition of $V(G)$ where $S \subseteq \mathcal{V}(G)$ and $N(A), N(B) \subseteq S$. We call (A, S, B) a (q, k)-*separation* if S has size at most k, and A and B have size at least q.

▶ **Lemma 21.** *Let G be the incidence graph of a CSP instance $\mathbf{I}$. If G has no $(q, k+1)$-separation then $\mathbf{I}$ is (q, k)-nice.*

▶ **Lemma 22.** *Let $t \in \mathbb{N}$ and $\mathbf{I}_1$ be a t-boundaried CSP instance with t-boundaried incidence graph (G, Z). Let $k, q \in \mathbb{N}$ be such that G does not admit a $(8^q, k+1)$-separation, and let (H, J) be the t-boundaried incidence graph of a t-boundaried CSP instance $\mathbf{I}_2$ such that the size of $V(H)$ is at most some $r \in \mathbb{N}$. Then the incidence graph $G \oplus H$ corresponding to the instance $\mathbf{I}_1 \oplus \mathbf{I}_2$ has no $(8^q + r, k+1)$-separation.*

For the following lemma, recall the definition of the set $\mathfrak{F}_s(t, h^*(k,t), k)$ (Lemma 16). The proof relies on Lemmas 22, 21, and 19.

▶ **Lemma 23.** *Let $t \in \mathbb{N}$ and $\mathbf{I}_1$ be a t-boundaried CSP instance with incidence graph G and boundary Z. Further, let $k, q \in \mathbb{N}$ be such that $t \leq 2(k+1)$, $|V(G)| > q$, and G does not admit a $(8^q, k+1)$-separation. Let (H, J) be the t-boundaried incidence graph of a t-boundaried*

CSP instance $\mathbf{I}_2$ *in* $\mathfrak{F}_s(t, h^*(k,t), k)$*. Then the instance* $\mathbf{I}_1 \oplus \mathbf{I}_2$ *is* $(8^q + \alpha(k, 2(k+1)), k)$*-nice. Furthermore, if* $q = \alpha(k, 2(k+1))$ *then one can compute in time* $\mathcal{O}(\mathfrak{M}(k)|G|^2)$ *a* t*-boundaried CSP instance* $\mathbf{I}_1^*$ *of size at most* q *such that* $\mathbf{I}_1 \sim_{t,k} \mathbf{I}_1^*$*, for some function* $\mathfrak{M}$*.*

6.3 Solving the Problem via Recursive Understanding

In this subsection, we complete our algorithm for Width Strong-CSP(Γ) Backdoor Detection by describing the recursive phase of our algorithm and the way we utilize the subroutines described earlier to solve the problem. We note that variants of Lemma 24, Lemma 25 and Lemma 27 are well-known in literature (see for example [12]). However the parameters involved in these lemmas are specific to the application. Furthermore, our proofs are simpler and avoid the color coding technique employed in [12]. Following that, we use Lemma 24 to obtain the final ingredient for our algorithm.

▶ **Lemma 24.** *There is an algorithm that, given an incidence graph* G *and* $q, k \in \mathbb{N}$*, runs in time* $\mathcal{O}((2q)^k \cdot |G|^2)$ *and either computes a* (q,k)*-separation or concludes correctly that there is no* (q,k)*-separation* (A,S,B) *where* A *and* B *are connected.*

▶ **Lemma 25.** *There is an algorithm that, given an incidence graph* G *and* $q, k \in \mathbb{N}$*, runs in time* $\mathcal{O}((q+k)^k|G|^2)$ *and either computes a* (q,k)*-separation in* G *or correctly concludes that* G *has no* $(8^q, k)$*-separation.*

▶ **Observation 26.** *Let* (G,Z) *be a* t*-boundaried graph with* $|V(G)| > q$ *and* $t \leq 2(k+1)$ *and let* (A,S,B) *be a* $(q, k+1)$*-separation in* G*. Then, one of the pairs* $(G[A \cup S], S \cup (Z \cap A))$ *or* $(G[B \cup S], S \cup (Z \cap B))$ *is a* t'*-boundaried graph with* $t' \leq 2(k+1)$*.*

▶ **Lemma 27.** *There is an algorithm that, given a* t*-boundaried graph* (G,Z) *with* $|V(G)| > q$ *and* $t \leq 2(k+1)$*, in time* $\mathcal{O}((q+k)^k|G|^3)$ *returns a* t'*-boundaried graph* (G',Z') *where* G' *is a subgraph of* G*,*

(a) $|V(G')| > q$*,*
(b) $t' \leq 2k+1$*, and*
(c) G' *has no* $(8^q, k+1)$*-separation.*

Proof. We begin by executing the algorithm of Lemma 25. If this algorithm returns that G has no $(8^q, k+1)$-separation then we terminate the algorithm and return the graph (G,Z) itself. Otherwise, let (X,S,Y) be the $(q,k+1)$-separation returned by this algorithm. By Observation 26, we may assume w.l.o.g. that $(G[X \cup S], S \cup (Z \cap X))$ is a t''-boundaried graph where $t'' \leq 2(k+1)$. We now set $G := G[X \cup S]$, $Z := S \cup (Z \cap X)$ and recurse. Since the depth of recursion is bounded by the size of the input graph and each step takes time $\mathcal{O}((q+k)^k|G|^2)$, the lemma follows. ◀

Algorithm for the Decision version of Theorem 8. Let $\mathbf{I}$ be the given input CSP instance and let G be its incidence graph. We begin by setting $q = \alpha(k, 2(k+1))$, choosing the boundary Z to be the empty set and then executing the algorithm of Lemma 27 to compute a t-boundaried graph (G',Z') where G' is a subgraph of G, $|V(G')| > q$ and $t \leq 2(k+1)$ such that G' has no $(8^q, k+1)$-separation. Next, we invoke Lemma 23 on the corresponding CSP instance, say $\mathbf{I}'$, to compute in time $\mathcal{O}(\mathfrak{M}(k)|G|^2)$ a t-boundaried CSP instance $\mathbf{I}''$ such that $\mathbf{I}' \sim_{t,k} \mathbf{I}''$. We then set $\mathbf{I} = \mathbf{I}'' \oplus (\psi(G - (V(G') \setminus Z')))$ and recursively check for the presence of a strong backdoor set of width at most k for this instance. Since we strictly reduce the size of the instance in each step, the depth of the recursion is bounded linearly in the size of the initial input, implying FPT running time.

Proof of Theorem 1 and Corollary 2. Using the self-reducibility of the problem and the algorithm for the decision variant of Theorem 8, one can compute a strong backdoor set of width at most k (if it it exists). Following this, one can execute the algorithm of Lemma 6 to solve CSP and #CSP. ◀

7 Concluding Remarks

We have introduced the notion of backdoor treewidth for CSP and #CSP by combining the two classical approaches of placing structural restrictions and language restrictions, respectively, on the input. Thus the presented results represent a new "hybrid" approach for solving CSPand #CSP. Our main result, Theorem 1, is quite broad as it covers all tractable finite constraint languages combined with the graph invariant treewidth. This can be seen as the base case of a general framework which combines a specific graph invariant of the torso graph with a specific class of constraint languages. Therefore, we hope it will stimulate further research in this direction.

References

1 Stefan Arnborg, Bruno Courcelle, Andrzej Proskurowski, and Detlef Seese. An algebraic theory of graph reduction. *J. ACM*, 40(5):1134–1164, 1993. doi:10.1145/174147.169807.

2 Christian Bessiere, Clément Carbonnel, Emmanuel Hebrard, George Katsirelos, and Toby Walsh. Detecting and exploiting subproblem tractability. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. IJCAI/AAAI, 2013.

3 Hans L. Bodlaender and Babette de Fluiter. Reduction algorithms for constructing solutions in graphs with small treewidth. In Jin-Yi Cai and Chak Kuen Wong, editors, *COCOON'96*, LNCS, pages 199–208. Springer, 1996.

4 Hans L. Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. *SIAM J. Comput.*, 27:1725–1746, 1998.

5 Hans L. Bodlaender and Babette van Antwerpen-de Fluiter. Reduction algorithms for graphs of small treewidth. *Inf. Comput.*, 167:86–119, 2001.

6 Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. of the ACM*, 53(1):66–120, 2006.

7 Andrei A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Log.*, 12(4):Art. 24, 66, 2011.

8 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. of the ACM*, 60(5):Art 34, 41, 2013.

9 Andrei A. Bulatov, Andrei A. Krokhin, and Peter Jeavons. The complexity of maximal constraint languages. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 667–674. ACM, 2001.

10 Clément Carbonnel and Martin C. Cooper. Tractability in constraint satisfaction problems: a survey. *Constraints*, 21(2):115–144, 2016.

11 Clément Carbonnel, Martin C. Cooper, and Emmanuel Hebrard. On backdoors to tractable constraint languages. In *Principles and Practice of Constraint Programming – 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, volume 8656 of *Lecture Notes in Computer Science*, pages 224–239. Springer Verlag, 2014.

12 Rajesh Hemant Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. Designing FPT algorithms for cut problems using randomized con-

tractions. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 460–469, 2012.

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

14 Martin C. Cooper, David A. Cohen, and Peter G. Jeavons. Characterising tractable constraints. *Artificial Intelligence*, 65(2):347–361, 1994.

15 Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

16 Y. Crama, O. Ekin, and P. L. Hammer. Variable and term removal from Boolean formulae. *Discr. Appl. Math.*, 75(3):217–230, 1997.

17 Víctor Dalmau. A new tractable class of constraint satisfaction problems. In *AMAI, 6th International Symposium on Artificial Intelligence and Mathematics, Fort Lauderdale, Florida, USA, January 5-7, 2000*, 2000.

18 Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In Pascal Van Hentenryck, editor, *Principles and Practice of Constraint Programming – CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings*, volume 2470 of *Lecture Notes in Computer Science*, pages 310–326. Springer Verlag, 2002.

19 Babette de Fluiter. *Algorithms for Graphs of Small Treewidth.* PhD thesis, Utrecht University, 1997.

20 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

21 Tommy Färnqvist. Constraint optimization problems and bounded tree-width revisited. In Nicolas Beldiceanu, Narendra Jussien, and Eric Pinson, editors, *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems – 9th International Conference, CPAIOR 2012, Nantes, France, May 28 June1, 2012. Proceedings*, volume 7298 of *Lecture Notes in Computer Science*, pages 163–179. Springer Verlag, 2012.

22 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

23 Michael R. Fellows and Michael A. Langston. An analogue of the Myhill-Nerode theorem and its use in computing finite-basis characterizations (extended abstract). In *FOCS*, pages 520–525, 1989.

24 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, M. S. Ramanujan, and Saket Saurabh. Solving d-SAT via backdoors to small treewidth. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 630–641. SIAM, 2015.

25 Eugene C. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32(4):755–761, 1985.

26 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1670–1681, 2016.

27 Serge Gaspers, Neeldhara Misra, Sebastian Ordyniak, Stefan Szeider, and Stanislav Zivny. Backdoors into heterogeneous classes of SAT and CSP. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27-31, 2014, Québec City, Québec, Canada.*, pages 2652–2658. AAAI Press, 2014.

28 Serge Gaspers, Sebastian Ordyniak, M. S. Ramanujan, Saket Saurabh, and Stefan Szeider. Backdoors to q-horn. *Algorithmica*, 74(1):540–557, 2016. doi:10.1007/s00453-014-9958-5.

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

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

31 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. of the ACM*, 54(1), 2007.

32 Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 479–488, 2011.

33 Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990.

34 Pavol Hell and Jaroslav Nesetril. Colouring, constraint satisfaction, and complexity. *Computer Science Review*, 2(3):143–163, 2008.

35 Lane A. Hemaspaandra and Ryan Williams. SIGACT news complexity theory column 76: an atypical survey of typical-case heuristic algorithms. *SIGACT News*, pages 70–89, 2012.

36 Ken-ichi Kawarabayashi and Mikkel Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 160–169, 2011.

37 Phokion G. Kolaitis. Constraint satisfaction, databases, and logic. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 1587–1595. Morgan Kaufmann, 2003.

38 Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. of Computer and System Sciences*, 61(2):302–332, 2000. Special issue on the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (Seattle, WA, 1998).

39 Jens Lagergren. Upper bounds on the size of obstructions and intertwines. *J. Comb. Theory, Ser. B*, 73(1):7–40, 1998.

40 Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. of the ACM*, 60(6):Art. 42, 51, 2013.

41 Ugo Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

42 Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *Proceedings of SAT 2004 (Seventh International Conference on Theory and Applications of Satisfiability Testing, 10–13 May, 2004, Vancouver, BC, Canada)*, pages 96–103, 2004.

43 Igor Razgon and Barry O'Sullivan. Almost 2-SAT is fixed parameter tractable. *J. of Computer and System Sciences*, 75(8):435–450, 2009.

44 Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *J. Discrete Algorithms*, 8(1):50–64, 2010.

45 Marko Samer and Stefan Szeider. Constraint satisfaction with bounded treewidth revisited. *J. of Computer and System Sciences*, 76(2):103–114, 2010.

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

47 Ryan Williams, Carla Gomes, and Bart Selman. Backdoors to typical case complexity. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003*, pages 1173–1178. Morgan Kaufmann, 2003.

48 Ryan Williams, Carla Gomes, and Bart Selman. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search. In *Informal Proc. of the Sixth International Conference on Theory and Applications of Satisfiability Testing, S. Margherita Ligure – Portofino, Italy, May 5-8, 2003 (SAT 2003)*, pages 222–230, 2003.