Technical Report AC-TR-17-016

December 2017

Discovering Archipelagos of Tractability for Constraint Satisfaction and Counting

Robert Ganian, M.S. Ramanujan, and Stefan Szeider

This is the authors' copy of a paper that appeared in *ACM Trans. Algorithms* 13, 2, Article 29 (March 2017). DOI: 10.1145/3014587

www.ac.tuwien.ac.at/tr

Discovering Archipelagos of Tractability for Constraint Satisfaction and Counting

ROBERT GANIAN, M. S. RAMANUJAN, and STEFAN SZEIDER, TU Wien, Vienna, Austria

The Constraint Satisfaction Problem (CSP) is a central and generic computational problem which provides a common framework for many theoretical and practical applications. A central line of research is concerned with the identification of classes of instances for which CSP can be solved in polynomial time; such classes are often called "islands of tractability." A prominent way of defining islands of tractability for CSP is to restrict the relations that may occur in the constraints to a fixed set, called a *constraint language*, whereas a constraint language is conservative if it contains all unary relations. Schaefer's famous Dichotomy Theorem (STOC 1978) identifies all islands of tractability in terms of tractable constraint languages over a Boolean domain of values. Since then, many extensions and generalizations of this result have been obtained. Recently, Bulatov (TOCL 2011, JACM 2013) gave a full characterization of all islands of tractability for CSP and the counting version #CSP that are defined in terms of conservative constraint languages.

This article addresses the general limit of the mentioned tractability results for CSP and #CSP, that they only apply to instances where all constraints belong to a single tractable language (in general, the union of two tractable languages is not tractable). We show that we can overcome this limitation as long as we keep some control of how constraints over the various considered tractable languages interact with each other. For this purpose, we utilize the notion of a *strong backdoor* of a CSP instance, as introduced by Williams et al. (IJCAI 2003), which is a set of variables that when instantiated, moves the instance to an island of tractability, that is, to a tractable class of instances. We consider strong backdoors into *scattered classes*, consisting of CSP instances where each connected component belongs entirely to some class from a list of tractable classes. Figuratively speaking, a scattered class constitutes an *archipelago of tractability*. The main difficulty lies in finding a strong backdoor of given size k; once it is found, we can try all possible instantiations of the backdoor variables and apply the polynomial time algorithms associated with the islands of tractability on the list component-wise. Our main result is an algorithm that, given a CSP instance with n variables, finds in time $f(k)n^{\mathcal{O}(1)}$ a strong backdoor into a scattered class (associated with a list of finite conservative constraint languages) of size k or correctly decides that there is not such a backdoor. This also gives the running time for solving (#)CSP, provided that (#)CSP is polynomial-time tractable for the considered constraint languages. Our result makes significant progress towards the main goal of the backdoor-based approach to CSPs—the identification of maximal base classes for which small backdoors can be detected efficiently.

CCS Concepts: • **Theory of computation → Fixed parameter tractability;**

Additional Key Words and Phrases: Backdoor sets, constraint languages, constraint satisfaction, counting, islands of tractability

ACM Reference Format:
Robert Ganian, M. S. Ramanujan, and Stefan Szeider. 2017. Discovering archipelagos of tractability for constraint satisfaction and counting. ACM Trans. Algorithms 13, 2, Article 29 (March 2017), 32 pages.
DOI: http://dx.doi.org/10.1145/3014587

This research was supported by the Austrian Science Fund (FWF), project P26696 X-TRACT. The first author is also affiliated with FI MU, Brno, Czech Republic. The second author was supported by the Bergen Research Foundation under the grant "BeHard."

Authors' addresses: R. Ganian, M. S. Ramanujan, and S. Szeider, Technische Universität Wien, Institut für Computergraphik und Algorithmen 186/1, Favoritenstraße 9–11, 1040 Wien, Austria; emails: {rganian, ramanujan, sz}@ac.tuwien.ac.at.

1. INTRODUCTION

The Constraint Satisfaction Problem (CSP) is a central and generic computational problem that provides a common framework for many theoretical and practical applications [Hell and Nesetril 2008]. 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 Montanar [1974] and has been found to be equivalent to the homomorphism problem for relational structures [Feder and Vardi 1998] and the problem of evaluating conjunctive queries on databases [Kolaitis 2003]. In general, CSP is NP-complete. A central line of research is concerned with the identification of classes of instances for which CSP can be solved in polynomial time. Such classes are often called "islands of tractability" [Kolaitis 2003; Kolaitis and Vardi 2007].

A prominent way of defining islands of tractability for CSP is to restrict the relations that may occur in the constraints to a fixed set Γ, called a *constraint language*. A finite constraint language is *tractable* if CSP restricted to instances using only relations from Γ, denoted CSP(Γ), can be solved in polynomial time. Schaefer's famous Dichotomy Theorem [Schaefer 1978] identifies all islands of tractability in terms of tractable constraint languages over the two-element domain. Since then, many extensions and generalizations of this result have been obtained [Jeavons et al. 1997; Creignou 1995; Kolmogorov and Živný 2013; Thapper and Zivny 2013]. The Dichotomy Conjecture of Feder and Vardi [1993] claims that for every finite constraint language Γ, CSP(Γ) is either NP-complete or solvable in polynomial time. Schaefer's Dichotomy Theorem shows that the conjecture holds for two-element domains; more recently, Bulatov [2006] showed the conjecture to be true for three-element domains. Several papers are devoted to identifying constraint languages Γ for which *counting CSP*, denoted #CSP(Γ), can be solved in polynomial time [Bulatov 2013; Creignou and Hermann 1996; Bulatov and Dalmau 2007], that is, where the number of satisfying assignments can be computed in polynomial time. Such languages Γ are called *#-tractable*.

A constraint language over $\mathcal{D}$ is *conservative* if it contains all possible unary constraints over $\mathcal{D}$, and it is *semi-conservative* if it contains all possible unary constant constraints (i.e., constraints that fix a variable to a specific domain element). These properties of constraint languages are very natural, as one would expect in practical settings that the unary relations are present. Indeed, some authors (e.g., Cooper et al. [1994]) even define CSP so that every variable can have its own set of domain values, making conservativeness a built-in property. Recently, Bulatov [2011] gave a full characterization of all tractable conservative constraint languages over finite domains. Furthermore, Bulatov [2013] gave a full characterization of all #-tractable constraint languages over finite domains. Thus, Bulatov's results identify all islands of (#-)tractability over finite domains that can be defined in terms of a conservative constraint language.

A general limit of tractability results for CSP and #CSP based on constraint languages, such as the mentioned results of Schaefer and Bulatov, is that they only apply to instances where all constraints belong to a single tractable language. One cannot arbitrarily combine constraints from two or more tractable languages, as, in general, the union of two tractable languages iis not tractable (see Section 2). In this article, we show that we can overcome this limitation as long as constraints over the various considered tractable languages interact with each other in a controlled manner. For this purpose, we utilize the notion of a *strong backdoor* of a CSP instance, as introduced by Williams et al. [2003a]. A set B of variables of a CSP instance is a strong backdoor into a tractable class $\mathcal{H}$ if for all instantiations of the variables in B, the reduced instance belongs to $\mathcal{H}$. In this article, we consider strong backdoors into a *scattered class*, denoted $\mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_d$, consisting of all CSP instances $\mathbf{I}$ such that each connected component of $\mathbf{I}$ belongs entirely to some class from a list of tractable classes $\mathcal{H}_1, \ldots, \mathcal{H}_d$. Figuratively speaking, $\mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_d$ constitutes an archipelago of tractability, consisting of the islands $\mathcal{H}_1, \ldots, \mathcal{H}_d$. Our main result is the following:

Theorem 1.1. *Let $\Gamma_1, \ldots, \Gamma_d$ be semiconservative finite constraint languages over a domain $\mathcal{D}$, and let $\mathcal{D}^*$ be the language containing all relations over $\mathcal{D}$. If $\Gamma_1, \ldots, \Gamma_d$ are tractable (or #-tractable), then* CSP($\mathcal{D}^*$) *(or #CSP($\mathcal{D}^*$), respectively) can be solved in time $2^{2^{\mathcal{O}(k)}} \cdot n^{\mathcal{O}(1)}$ for instances with n variables that have a strong backdoor of size k into* CSP(Γ_1) $\oplus \cdots \oplus$ CSP(Γ_d).

Note that there are natural CSP instances that have a small strong backdoor into the scattered class CSP(Γ_1) $\oplus \cdots \oplus$ CSP(Γ_d) but require strong backdoors of arbitrarily large size into each individual base class CSP(Γ_i). The power of a strong backdoor into a scattered class over one into a single class stems from the fact that the instantiation of variables in the backdoor can serve two purposes. The first is to separate constraints into components, each belonging entirely to some CSP(Γ_i) (possibly even different CSP(Γ_i)'s for different instantiations), and the second is to modify constraints so that once modified, the component containing these constraints belongs to some CSP(Γ_i).

When using the backdoor-based approach, the main computational difficulty lies in detecting small backdoor sets into the chosen base class. This task becomes significantly harder when the base classes are made more general. However, we show that while scattered classes are significantly more general than single tractable classes, we can still detect strong backdoors into such classes in FPT time. The formal statement of this result, which represents our main technical contribution, is the following.

Lemma 1.2. *There is an algorithm that, given a CSP instance $\mathbf{I}$ and a parameter k, runs in time $2^{2^{\mathcal{O}(k)}} \cdot n^{\mathcal{O}(1)}$ and either finds a strong backdoor of size at most k in $\mathbf{I}$ into* CSP(Γ_1^*) $\oplus \cdots \oplus$ CSP(Γ_d^*) *or correctly decides that none exists.*

Here $\Gamma_i^* \supseteq \Gamma_i$ is obtained from Γ_i by taking the closure under partial assignments and by adding a redundant relation.

We remark that the finitary restriction on the constraint languages is unavoidable, since otherwise the arity of the relations or the domain size would be unbounded. However, for unbounded arity, small backdoors cannot be found efficiently even for the special case of $d = 1$ unless FPT = W[2] [Gaspers et al. 2014]. That is, there is a language Γ of unbounded arity such that if there is *any* FPT algorithm to detect a strong backdoor of size at most k into CSP(Γ), then the class W[2] is contained in FPT, which is considered to be an unlikely collapse in the parameterized complexity hierarchy. Similarly, with unbounded domain, a small strong backdoor cannot be used efficiently. For instance, the natural encoding of the W[1]-hard k-clique problem to CSP [Papadimitriou and Yannakakis 1999] only has k variables and, therefore, has a size-k strong backdoor to any base class that contains the trivial constrains with empty scopes, which is the case for any natural base class; an FPT algorithm solving such instances would once again imply FPT = W[1].

The following is a brief summary of the algorithm of Lemma 1.2. We will give a more detailed summary in Section 3.

(1) We begin by using the technique of iterative compression [Reed et al. 2004] to transform the problem into a structured subproblem, which we call Ext-SBD Comp. In this technique, the idea is to start with an subinstance and a trivial solution for this subinstance and iteratively expand the subinstances while compressing the solutions till we solve the problem on the original instance. Specifically, in

Ext-SBD Comp, we are given additional information about the desired solution in the input: we receive an “old” strong backdoor, which is slightly bigger than our target size, along with information about how this old backdoor interacts with our target solution. In order to begin the iteration, we consider the (somewhat trivial) subinstance induced on a single arbitrarily chosen constraint. We will show that this constraint must have arity bounded by k plus a constant ρ depending on the languages $\Gamma_1, \dots, \Gamma_d$. As a result, this subinstance has a trivial backdoor of size at most $k+\rho$ – simply take all the variables in the scope of this single constraint. This is the “old” strong backdoor that we use to bootstrap our iterative procedure. The details of this procedure are presented in Section 3.1.

(2) In Section 3.2, we consider only solutions for Ext-SBD Comp instances that have a certain “inseparability property” and give an FPT algorithm to test for the presence of such solutions. To be more precise, here we only look for solutions of Ext-SBD Comp that leave the omitted part of the old strong backdoor in a single connected component. We handle this case separately at the beginning because it serves as a base case in our algorithm to solve general instances. Interestingly, even this base case requires the extension of state of the art separator techniques to the CSP setting.

(3) Finally, in Section 3.3, we show how to handle general instances of Ext-SBD Comp. This part of the algorithm relies on a new *pattern replacement* technique, which shares certain superficial similarities with protrusion replacement [Bodlaender et al. 2009] but allows the preservation of a much larger set of structural properties (such as containment of disconnected forbidden structures and connectivity across the boundary). We interleave our pattern replacement procedure with the approach of “tight separator sequences” [Lokshtanov and Ramanujan 2012] as well as the algorithm designed in the previous subsection for “inseparable” instances in order to solve the problem on general instances. Before we conclude the summary, we would like to point out an interesting feature of our algorithm. At its very core, it is a branching algorithm; in FPT time, we identify a bounded set of variables that intersects some solution and then branch on this set. Note that this approach does not always result in an FPT-algorithm for computing strong backdoor sets. In fact, depending on the base class it might only imply an FPT-*approximation* algorithm (see Gaspers and Szeider [2013]). This is because we need to explore all possible assignments for the chosen variable. However, we develop a notion of *forbidden sets of constraints,* which allows us to succinctly describe when a particular set is not already a solution. Therefore, when we branch on a supposed strong backdoor variable, we simply add it to a partial solution that we maintain, and then we can at any point easily check whether the partial solution is already a solution or not. This is a crucial component of our FPT algorithm.

Related work. Williams et al. [2003a, 2003b] introduced the notion of *backdoors* for the runtime analysis of algorithms for CSP and SAT; see also Hemaspaandra and Williams [2012] 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. 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. [1997]. The study of the parameterized complexity of finding small backdoors was initiated by Nishimura et al. [2004] for SAT, who considered backdoors into the classes of Horn and Krom CNF formulas. Further results cover the classes of renamable Horn formulas [Razgon and O’Sullivan 2009], q-Horn formulas [Gaspers et al. 2013] and classes of formulas of bounded treewidth [Gaspers and Szeider 2013; Fomin et al. 2015]. The detection of backdoors for CSP has been studied for instance in Bessiere et al. [2013] and Carbonnel et al. [2014]. Gaspers et al. [2014] recently 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). Our setting is more general becuase $\mathrm{CSP}(\Gamma_1) \oplus \cdots \oplus \mathrm{CSP}(\Gamma_d) \supseteq \mathrm{CSP}(\Gamma_1) \cup \cdots \cup \mathrm{CSP}(\Gamma_d)$, and the size of a smallest strong backdoor into $\mathrm{CSP}(\Gamma_1) \cup \cdots \cup \mathrm{CSP}(\Gamma_d)$ can be arbitrarily larger than the size of a smallest strong backdoor into $\mathrm{CSP}(\Gamma_1) \oplus \cdots \oplus \mathrm{CSP}(\Gamma_d)$.

2. PRELIMINARIES

2.1. Constraint Satisfaction

Let $\mathcal{V}$ be an infinite 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. A *value assignment* (or *assignment*, for short) $\alpha : X \to \mathcal{D}$ is a mapping defined on 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 $\mathrm{rel}(\mathbf{I}) = \{ R : (S, R) \in C, C \in \mathbf{I} \}$. Unless otherwise specified, we use m and n to denote the number of constraints and variables in the instance under consideration.

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., Bulatov [2013]).

Let $\alpha : X \to \mathcal{D}$ be an assignment. For a ρ-ary constraint $C = (S, R)$ with $S = (x_1, \ldots, x_\rho)$, 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 finite domain $\mathcal{D}$ is a set Γ 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 Γ is *tractable* if for every finite subset $\Gamma' \subseteq \Gamma$, the problem $\mathrm{CSP}(\Gamma')$ can be solved in polynomial time. A constraint language Γ is *#-tractable* if for every finite subset $\Gamma' \subseteq \Gamma$, the problem $\#\mathrm{CSP}(\Gamma')$ can be solved in polynomial time.

In his seminal paper, Schaefer [1978] showed that for all constraint languages Γ over the Boolean domain $\{0, 1\}$, $\mathrm{CSP}(\Gamma)$ is either NP-complete or solvable in polynomial time. In fact, he showed that a Boolean constraint language Γ is tractable if and only if at least one of the following properties holds for each relation $R \in \Gamma$: (i) $(0, \ldots, 0) \in R$, (ii) $(1, \ldots, 1) \in R$, (iii) R is equivalent to a conjunction of binary clauses, (iv) R is equivalent to a conjunction of Horn clauses, (v) R is equivalent to a conjunction of dual-Horn clauses, and (vi) R is equivalent to a conjunction of affine formulas; Γ is then called 1-valid, 0-valid, bijunctive, Horn, dual-Horn, or affine, respectively. A Boolean language that satisfies any of these six properties is called a *Schaefer language*. A constraint language Γ over domain $\mathcal{D}$ is *conservative* if Γ contains all unary relations over $\mathcal{D}$. Except for the somewhat trivial 0-valid and 1-valid languages, all Schaefer languages are conservative. Γ is *semi-conservative* if it contains all unary relations over $\mathcal{D}$ that are singletons (i.e., constraints that fix the value of a variable to some element of $\mathcal{D}$).

A constraint language Γ is *closed under assignments* if for every $C = (S, R)$ such that $R \in \Gamma$ and every assignment α, it holds that $R' \in \Gamma$ where $C|_\alpha = (S', R')$. For a constraint language Γ over a domain $\mathcal{D}$, we denote by Γ^* the smallest constraint language over $\mathcal{D}$ that contains $\Gamma \cup \{\mathcal{D}^2\}$ and is closed under assignments; notice that Γ^* is uniquely determined by Γ. Evidently, if a language Γ is tractable (or #-tractable, respectively) and semiconservative, then so is Γ^*: first, all constraints of the form $(S, \mathcal{D}^2|_\alpha)$ can be detected in polynomial time and removed from the instance without changing the solution, and then each constraint $C' = (S', R')$ with $R' \in \Gamma^* \setminus \Gamma$ can be expressed in terms of the conjunction of a constraint $C = (S, R)$ with $R \in \Gamma$ and unary constraints over variables in $\text{var}(C) \setminus \text{var}(C')$.

As mentioned in the introduction, the union of two tractable constraint languages is in general not tractable. Take for instance the conservative languages $\Gamma_1 = \{\{0, 1\}^3 \setminus \{(1, 1, 1)\}\} \cup 2^{\{0,1\}}$ and $\Gamma_2 = \{\{0, 1\}^3 \setminus \{(0, 0, 0)\}\} \cup 2^{\{0,1\}}$. Using the characterization of Schaefer languages in terms of closure properties (see, e.g., Gopalan et al. [2009]), it is easy to check that Γ_1 is Horn and has none of the five other Schaefer properties; similarly, Γ_2 is dual-Horn and has none of the five other Schaefer properties. Hence, if follows by Schaefer's Theorem that CSP(Γ_1) and CSP(Γ_2) are tractable, but CSP($\Gamma_1 \cup \Gamma_2$) is NP-complete. One can find similar examples for other pairs of Schaefer languages.

2.2. Parameterized Complexity

A parameterized problem $\mathcal{P}$ is a problem whose instances are tuples (I, k), where $k \in \mathbb{N}$ is called the *parameter*. We say that a parameterized problem is *fixed parameter tractable* (FPT in short) if it can be solved by an algorithm that runs in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for some computable function f; algorithms with running time of this form are called *FPT algorithms*. The notions of W[i]-*hardness* (for $i \in \mathbb{N}$) are frequently used to show that a parameterized problem is not likely to be FPT; an FPT algorithm for a W[i]-hard problem would imply that the Exponential Time Hypothesis fails [Chen et al. 2006]. We refer the reader to other sources [Downey and Fellows 1999, 2013; Flum and Grohe 2006] for an in-depth introduction into parameterized complexity.

2.3. Backdoors, Incidence Graphs, and Scattered Classes

Let $\mathbf{I}$ be an instance of CSP over $\mathcal{D}$ and let $\mathcal{H}$ be a class of CSP instances. A set B of variables of $\mathbf{I}$ is called a *strong backdoor* into $\mathcal{H}$ if for every assignment $\alpha : B \to \mathcal{D}$ it holds that $\mathbf{I}|_\alpha \in \mathcal{H}$. Notice that if we are given a strong backdoor B of size k into a tractable (or #-tractable) class $\mathcal{H}$, then it is possible to solve CSP (or #CSP) in time $|\mathcal{D}|^k \cdot n^{\mathcal{O}(1)}$. It is thus natural to ask for which tractable classes we can find a small backdoor efficiently.

STRONG BACKDOOR DETECTION INTO $\mathcal{H}$ (SBD($\mathcal{H}$))
Setting: A class $\mathcal{H}$ of CSP instances over a finite domain $\mathcal{D}$.
Instance: A CSP instance $\mathbf{I}$ over $\mathcal{D}$ and a non-negative integer k.
Task: Find a strong backdoor in $\mathbf{I}$ into $\mathcal{H}$ of cardinality at most k, ordetermine that no such strong backdoor exists.
Parameter: k.

We remark that for any *finite* constraint language Γ, the problem SBD(CSP(Γ)) is fixed parameter tractable due to a simple folklore branching algorithm. Let ρ be the arity of the language Γ. It is clear that if there is a single constraint in the input instance whose arity exceeds $k + \rho$, then the instance has *no* strong backdoor into CSP(Γ). This is because, no matter which k variables we instantiate, the resulting constraint is still not in Γ. As a result, all interesting instances have a bound of $k + \rho$ on the arity of every constraint. The branching algorithm now simply does the following. At any step, it maintains a set Z of at most k variables which has to be extended to a strong backdoor of size at most k. As long as there is an instantiation of the variables in Z such that the resulting instance contains at least one constraint which is not in Γ, it selects an arbitrary such constraint, and branches on the at most $k + \rho$ variables in its scope by adding one of these variables to the set Z. Since at least one of these variables *must* be in any strong backdoor containing Z, this branching is exhaustive. Furthermore, since we begin by setting $Z = \emptyset$, and the size of Z increases at each branching step with a bound of k on its size, the depth of the search tree is bounded by k. This implies a bound of $(\rho + k)^k$ on the number of leaves of the search tree and hence an FPT algorithm for SBD(CSP(Γ)).

On the other hand, SBD(CSP(Γ)) is known to be W[2]-hard for a wide range of infinite tractable constraint languages Γ [Gaspers et al. 2014].

Given a CSP instance $\mathbf{I}$, we use $\mathcal{B}(\mathbf{I}) = (\mathrm{var}(\mathbf{I}) \cup \mathbf{I}, E)$ to denote the *incidence graph* of $\mathbf{I}$; specifically, $\mathbf{I}$ contains an edge $\{x, Y\}$ for $x \in \mathrm{var}(\mathbf{I})$, $Y \in \mathbf{I}$ if and only if $x \in \mathrm{var}(Y)$. We denote this graph by $\mathcal{B}$ when $\mathbf{I}$ is clear from the context. Furthermore, for a set S of variables of $\mathbf{I}$, we denote by $\mathcal{B}_S(\mathbf{I})$ the graph obtained by deleting from $\mathcal{B}(\mathbf{I})$ the vertices corresponding to the variables in S; we may also use $\mathcal{B}_S$ in short if $\mathbf{I}$ is clear from the context. W For standard graph terminology, we refer to the book by Diestel [2011].

Two CSP instances $\mathbf{I}$, $\mathbf{I}'$ are *variable disjoint* if $\mathrm{var}(\mathbf{I}) \cap \mathrm{var}(\mathbf{I}') = \emptyset$. Let $\mathcal{H}_1, \dots \mathcal{H}_d$ be classes of CSP instances. Then the *scattered class* $\mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_d$ is the class of all CSP instances $\mathbf{I}$ that may be partitioned into pairwise variable disjoint subinstances $\mathbf{I}_1, \dots \mathbf{I}_d$ such that $\mathbf{I}_i \in \mathcal{H}_i$ for each $i \in [d]$. Notice that this implies that $\mathcal{B}(\mathbf{I})$ can be partitioned into pairwise disconnected subgraphs $\mathcal{B}(\mathbf{I}_1), \dots \mathcal{B}(\mathbf{I}_d)$. If $\mathcal{H}_1, \dots \mathcal{H}_d$ are tractable, then $\mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_d$ is also tractable, since each $\mathbf{I}_i$ can be solved independently. Similarly, if $\mathcal{H}_1, \dots \mathcal{H}_d$ are #-tractable, then $\mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_d$ is also #-tractable, since the number of satisfying assignments in each $\mathbf{I}_i$ can be computed independently and then multiplied to obtain the solution.

We conclude this section by showcasing that a strong backdoor to a scattered class can be arbitrarily smaller than a strong backdoor to any of its component classes. Consider once again the tractable languages $\Gamma_1 = \{\{0,1\}^3 \setminus \{(1,1,1)\}\} \cup 2^{\{0,1\}}$ (Horn) and $\Gamma_2 = \{\{0,1\}^3 \setminus \{(0,0,0)\}\} \cup 2^{\{0,1\}}$ (dual-Horn). Then, for any $k \in \mathbb{N}$, one can find $\mathbf{I} \in \mathrm{CSP}(\Gamma_1) \oplus \mathrm{CSP}(\Gamma_2)$ such that $\mathbf{I}$ does not have a strong backdoor of size k to either of CSP(Γ_1), CSP(Γ_2).

3. STRONG-BACKDOORS TO SCATTERED CLASSES

This section is dedicated to proving our main technical lemma, restated in the following text. We would like to point out that the assumption regarding the existence of the tautological binary relation $\mathcal{D}^2$ in the languages is made purely for ease of description in the later stages of the algorithm.

LEMMA 3.1. *Let $\Gamma_1, \dots \Gamma_d$ be finite languages over a finite domain $\mathcal{D}$ that are closed under partial assignments and contain $\mathcal{D}^2$. Then* SBD(CSP(Γ_1) $\oplus \cdots \oplus$ CSP(Γ_d)) *can be solved in time $2^{2^{\mathcal{O}(k)}}|\mathbf{I}|^{\mathcal{O}(1)}$.*

Before proceeding further, we show how Lemma 3.1 is used to prove Theorem 1.1.

PROOF OF THEOREM 1.1. Let $\mathbf{I}$ be an instance of CSP($\mathcal{D}^*$). Recalling the definition of Γ^*, we use Lemma 3.1 to find a strong backdoor X of size at most k into $\mathrm{CSP}(\Gamma_1^*) \oplus \cdots \oplus \mathrm{CSP}(\Gamma_d^*)$ in time $2^{2^{\mathcal{O}(k)}}|\mathbf{I}|^{\mathcal{O}(1)}$. Since $\mathrm{CSP}(\Gamma_1^*) \oplus \cdots \oplus \mathrm{CSP}(\Gamma_d^*) \supseteq \mathrm{CSP}(\Gamma_1) \oplus \cdots \oplus \mathrm{CSP}(\Gamma_d)$, it follows that any strong backdoor into $\mathrm{CSP}(\Gamma_1) \oplus \cdots \oplus \mathrm{CSP}(\Gamma_d)$ is also a strong backdoor into $\mathrm{CSP}(\Gamma_1^*) \oplus \cdots \oplus \mathrm{CSP}(\Gamma_d^*)$. We branch over all the at most $|\mathcal{D}|^k$ assignments $\alpha : X \to \mathcal{D}$, and for each such α, we solve the instance $\mathbf{I}|_\alpha$ in polynomial time since $\text{CSP}(\Gamma_1^*) \oplus \cdots \oplus \text{CSP}(\Gamma_d^*)$ is tractable.

For the second case, let $\mathbf{I}$ be an instance of #CSP($\mathcal{D}^*$). As before, we also use Lemma 3.1 to compute a strong backdoor X into $\text{CSP}(\Gamma_1^*) \oplus \cdots \oplus \text{CSP}(\Gamma_d^*)$ of size at most k. We then branch over all the at most $|\mathcal{D}|^k$ assignments $\alpha : X \to \mathcal{D}$, and for each such α, we solve the #CSP instance $\mathbf{I}|_\alpha$ in polynomial time since $\text{CSP}(\Gamma_1^*) \oplus \cdots \oplus \text{CSP}(\Gamma_d^*)$ is #-tractable. Let cost(α) denote the number of satisfying assignments of $\mathbf{I}|_\alpha$ for each α. We then output $\sum_{\alpha:X\to\mathcal{D}} \text{cost}(\alpha)$. □

We begin our path toward a proof of Lemma 3.1 by stating the following assumption on the input instance, which can be guaranteed by simple preprocessing. Let ρ be the maximum arity of any relation in $\Gamma_1, \ldots, \Gamma_d$.

Observation 1. *Any instance* $(\mathbf{I}', k)$ *of* $\text{SBD}(\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d))$ *either contains only constraints of arity at most* $\rho + k$*, or can be correctly rejected.*

Proof. Assume that $\mathbf{I}'$ contains a constraint $C = (S, R)$ of arity $\rho' > \rho + k$. Then for every set X of at most k variables, there exists an assignment $\alpha : X \to \mathcal{D}$ such that $C|_\alpha$ has arity $\rho' > \rho$, and hence $C|_\alpha \notin \text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$. Hence, any such $(\mathbf{I}', k)$ is clearly a No-instance of $\text{SBD}(\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d))$. □

Organization of the rest of the section. The rest of this section is structured into three subsections. In Section 3.1, we use iterative compression to transform the SBD problem targeted by Lemma 3.1 into its compressed version Ext-SBD Comp. Section 3.2 develops an algorithm that correctly solves any instance of Ext-SBD Comp that has a certain inseparability property. Finally, in Section 3.3, we give a general algorithm for Ext-SBD Comp, which uses the algorithm developed in Section 3.2 as a subroutine.

3.1. Iterative Compression

We first describe a way to reduce the input instance of $\text{SBD}(\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d))$ to multiple (but a bounded number of) *structured* instances, such that solving these instances will lead to a solution for the input instance. To do this, we use the technique of iterative compression [Reed et al. 2004]. Given an instance $(\mathbf{I}, k)$ of $\text{SBD}(\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d))$ where $\mathbf{I} = \{C_1, \ldots, C_m\}$, for $i \in [m]$ we define $\mathbf{C}_i = \{C_1, \ldots, C_i\}$. We iterate through the instances $(\mathbf{C}_i, k)$ starting from $i = 1$, and for each i-th instance we use a *known* solution X_i of size at most $k + \rho$ to try to find a solution $\hat{X}_i$ of size at most k. This problem, usually referred to as the *compression* problem, is the following.

$\text{SBD}(\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d))$ Compression
Setting: Languages $\Gamma_1, \ldots, \Gamma_d$ of maximum arity ρ over a domain $\mathcal{D}$.
Instance: A CSP instance $\mathbf{I}$, a non-negative integer k and a strong backdoor set $X \subseteq \text{var}(\mathbf{I})$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$ of size at most $2k + \rho$.
Task: Find a strong backdoor in $\mathbf{I}$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$ of size at most k, or correctly determine that no such set exists.
Parameter: k.

When $\Gamma_1, \ldots, \Gamma_d$ are clear from the context, we abbreviate $\text{SBD}(\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d))$ as SBD and $\text{SBD}(\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d))$ Compression as SBD Comp. We reduce the SBD problem to m instances of the SBD Comp problem as follows. Let $\mathbf{I}'$ be an instance of SBD. The set var(C_1) is clearly a strong backdoor of size at most ρ for the instance $\mathbf{I}_1 = (\mathbf{C}_1, k, \emptyset)$ of SBD Comp. We construct and solve a sequence of SBD Comp instances $\mathbf{I}_2, \ldots \mathbf{I}_m$ by letting $\mathbf{I}_i = (\mathbf{C}_i, k, X_{i-1} \cup \text{var}(C_i))$, where X_{i-1} is the required strong backdoor of $\mathbf{I}_{i-1}$. If some such $\mathbf{I}_i$ is found to have no solution, then we can correctly reject for $\mathbf{I}'$, since $\mathbf{C}_i \subseteq \mathbf{I}'$. On the other hand, if a solution X_m is obtained for $\mathbf{I}_m$, then X_m is also a solution for $\mathbf{I}'$. Since there are m such iterations, the total time taken is bounded by m times the time required to solve the SBD COMP problem.

Moving from the compression problem to the extension version. We now show how to convert an instance of the SBD COMP problem into a bounded number of instances of the same problem where we may additionally assume the solution we are looking for *extends* part of the given strong backdoor. Formally, an instance of the EXTENDED SBD COMP problem is a tuple $(\mathbf{I}, k, S, W)$ where $\mathbf{I}$ is a CSP instance, k is a nonnegative integer such that $|S| \leq k$, $|W| \leq 2k + \rho$ and $W \cup S$ is a strong backdoor set of $\mathbf{I}$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$. The objective here is to compute a strong backdoor into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$ of size at most k, which contains S and is disjoint from W.

EXT-SBD COMP
Setting: Languages $\Gamma_1, \ldots, \Gamma_d$ of maximum arity ρ over a domain $\mathcal{D}$.
Instance: A CSP instance $\mathbf{I}$, a nonnegative integer k and disjoint variable sets S and W such that $|S| \leq k$, $|W| \leq 2k + \rho$ and $W \cup S$ is a strong backdoor set of $\mathbf{I}$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$.
Task: Find a strong backdoor in $\mathbf{I}$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$ of size at most k that extends S and is disjoint from W, or determine that no such strong backdoor set exists.
Parameter: k.

We now reduce SBD COMP to $\binom{|X|}{\leq k}$-many instances of EXT-SBD COMP as follows. Let $\mathbf{I}' = (\mathbf{I}, k, X)$ be an instance of SBD COMP. We construct $\binom{|X|}{\leq k}$-many instances of EXT-SBD COMP as follows. For every $S \in \binom{X}{\leq k}$, we construct the instance $\mathbf{I}'_S = (\mathbf{I}, k, S, X \setminus S)$. Clearly, the original instance $\mathbf{I}'$ is a YES instance of SBD COMP if and only if for some $S \in \binom{X}{\leq k}$, the instance $\mathbf{I}'_S$ is a YES instance of EXT-SBD COMP. Therefore, the time to solve the instance $\mathbf{I}'$ is bounded by $\binom{|X|}{\leq k} \leq 2^{3k+\rho}$ times the time required to solve an instance of EXT-SBD COMP. In the rest of the article, we give an FPT algorithm to solve EXT-SBD COMP, which, following our earlier discussion, implies Lemma 3.1.

LEMMA 3.2. EXT-SBD COMP *can be solved in time* $2^{2^{\mathcal{O}(k)}}|\mathbf{I}|^{\mathcal{O}(1)}$.

We first focus on solving a special case of EXT-SBD COMP and then show how this helps to solve the problem in its full generality.

3.2. Solving Nonseparating Instances

In this subsection, we restrict our attention to input instances with a certain promise on the structure of a solution. We refer to these special instances as *nonseparating instances*. These instances are formally defined as follows.

Definition 3.3. Let $(\mathbf{I}, k, S, W)$ be an instance of EXT-SBD COMP and let $Z \supseteq S$ be a solution for this instance. We call Z a **separating solution** (see Figure 1) for this instance if W is not contained in a single connected component of $\mathcal{B}_Z$ and a **nonseparating solution** otherwise. An instance is called a **separating instance** if it only has separating solutions, and it is called a **nonseparating instance** otherwise.

Having formally defined nonseparating instances, we now give an overview of the algorithm we design to solve such instances. We begin by developing the notion of a *forbidden set of constraints*. The main motivation behind the introduction of this object is that it provides us with a succinct certificate that a particular set is *not* a strong backdoor of the required kind, immediately giving us a small structure, which we must

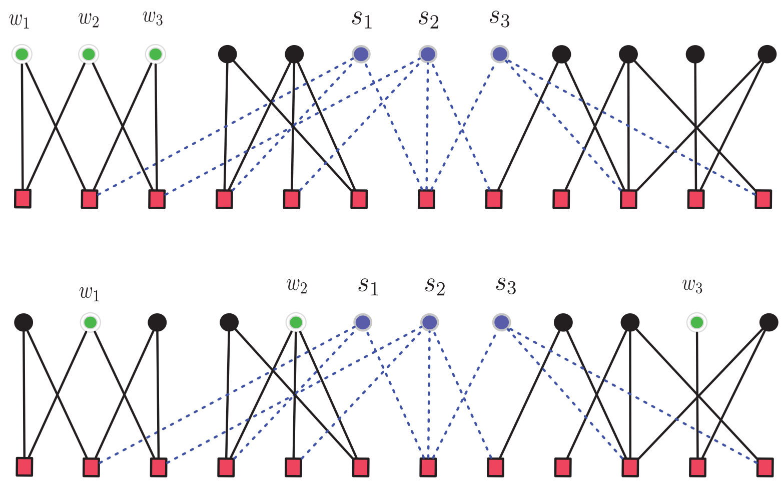

Fig. 1. An illustration of separating and nonseparating solutions. In both cases, $S = \{s_1, s_2, s_3\}$ is the hypothetical solution under consideration, while $\{w_1, w_2, w_3\}$ is the old solution. In the first figure, S is a nonseparating solution, while in the second, it is a separating solution.

exclude. However, the exclusion in this context can occur not just by instantiating a variable in the scope of one of these constraints in the solution but also due to the backdoor disconnecting these constraints. This is significantly different from standard graph problems where once we have a small violating structure, a straightforward branching algorithm can be used to eliminate this structure. However, in our case, even if we have a small violating structure, it is not at all clear how such a structure can be utilized. For this, we first set up appropriate separator machinery for CSP instances. We then argue that for any forbidden set of constraints, if a variable in the scope of these constraints is not in the solution, then one of these constraints must in fact be separated from the rest of the old strong backdoor set by the hypothetical solution. Following this, we argue that the notion of *important separators* introduced by Marx [2006] can be used to essentially narrow down the search space of separators where we must search for a solution variable. Finally, we use a branching algorithm in this significantly pruned search space of separators in order to compute a solution (if one exists). We reiterate that the notion of forbidden sets is critical in obtaining an FPT algorithm as opposed to an FPT-approximation algorithm. Now that we have given a slightly more detailed overview of this subsection, we proceed to describe our algorithm for solving nonseparating instances. We begin with the definition of forbidden constraints and then set up the separator machinery required in this as well as the next subsection.

3.2.1. Forbidden Constraints and Separator Machinery. In subsequent discussions, we deal with a fixed instance of EXT-SBD COMP, which we denote by $(\mathbf{I}, k, S, W)$.

Definition 3.4. Let $S \subseteq \text{var}(\mathbf{I})$, let $\mathbf{C} = \{C_1, \ldots, C_\ell\}$ be a set of at most d constraints and J be a subset of $[d]$. We say that $\mathbf{C}$ is ***J*-forbidden with respect to *S*** if there is an assignment $\tau : S \to \mathcal{D}$ such that for every $i \in J$ there is a $t \in [\ell]$ such that $C_t|_\tau \notin \Gamma_i$. If $J = [d]$, then we simply say that $\mathbf{C}$ is forbidden with respect to S. Furthermore, we call τ an assignment **certifying** that $\mathbf{C}$ is J-forbidden (forbidden if $J = [d]$) with respect to S.

The following observation is a consequence of the languages being closed under partial assignments.

OBSERVATION 2. *Let* $S \subseteq \text{var}(\mathbf{I})$ *and let* $\mathbf{C} = \{C_1, \ldots, C_\ell\}$ *be a set of at most* d *constraints. Then, the following statements hold.*

—*If* $\mathbf{C}$ *is forbidden with respect to* S*, then* $\mathbf{C}$ *is also forbidden with respect to every subset of* S *and in particular with respect to the set* $S \cap \text{var}(\mathbf{C})$.
—*If* $\mathbf{C}$ *is forbidden with respect to* S *and* S' *is a set of variables disjoint from* $\text{var}(\mathbf{C})$*, then* $\mathbf{C}$ *is also forbidden with respect to* $S \cup S'$ *and with respect to* S'.

The intuition behind the definition of forbidden sets is that it allows us to have succinct certificates for nonsolutions. This intuition is formalized in the following lemma.

LEMMA 3.5. *Given a CSP instance* $\mathbf{I}$*, a set* $X \subseteq \text{var}(\mathbf{I})$ *is a strong backdoor set into* $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$ *if and only if there is no connected component of* $\mathcal{B}_X$ *containing a set of constraints forbidden with respect to* X.

PROOF. Suppose that there is a connected component of $\mathcal{B}_X$ containing a set $\mathbf{C}$ that is forbidden with respect to X. By definition of forbidden sets, there is an instantiation $\tau : X \to \mathcal{D}$ such that for every $i \in [d]$, this connected component of $\mathcal{B}_X$ contains a constraint $C \in \mathbf{C}$ with the property that $C|_\tau \notin \Gamma_i$. Hence, X is by definition not a strong backdoor of $\mathbf{I}$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$.

Conversely, suppose that X is not a strong backdoor of $\mathbf{I}$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$. This implies that for some connected component of $\mathcal{B}_X$, for some instantiation $\tau : X \to \mathcal{D}$, and for every $i \in [d]$, there is a constraint C_i in this connected component such that $C_i|_\tau \notin \Gamma_i$. Consider the set $\mathbf{C} = \{C_1, \ldots, C_d\}$. Clearly, $\mathbf{C}$ is a set of constraints forbidden with respect to X. This completes the proof of the lemma. □

Now that we have defined the notion of forbidden sets and formally described its utility, we argue that one can in fact efficiently *check* whether forbidden sets exist with respect to a given variable set.

LEMMA 3.6. *Given a CSP instance* $\mathbf{I}$ *and a set* S *of variables, we can check in time* $\mathcal{O}(|\mathcal{D}|^{|S|} \cdot |I|^{\mathcal{O}(1)})$ *if there is a set of constraints forbidden with respect to* S.

PROOF. Clearly, it is sufficient to run over all the at most d-sized sets of constraints and all assignments to the variables in S and examine the reduced constraints if they belong to each of the languages $\Gamma_1, \ldots, \Gamma_d$. Since these languages are finite, the final check can be done in time $\mathcal{O}(1)$. This completes the proof of the lemma. □

We say that a variable set X *disconnects* a set $\mathbf{C}$ of constraints if the graph $\mathcal{B}_X$ has at least two connected components containing constraints from $\mathbf{C}$. Otherwise, we say that X does not disconnect $\mathbf{C}$. The following lemma argues that if there is a forbidden set of constraints with respect to some variable set, then this set has to be affected by the solution. This happens either by disconnecting the forbidden set or by directly picking (into the solution) a variable in the scope of a constraint in the forbidden set.

LEMMA 3.7. *Let* $\mathbf{I}$ *be a CSP instance, and let* $\mathbf{C}$ *be a set of constraints contained in a connected component of* $\mathbf{I}$ *and forbidden (with respect to some variable set). Let* Z *be a strong backdoor set of* $\mathbf{I}$ *into* $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$*. Then, either* Z *disconnects* $\mathbf{C}$ *or* $Z \cap \text{var}(\mathbf{C}) \neq \emptyset$.

PROOF. Suppose to the contrary that Z does not disconnect $\mathbf{C}$, and Z is disjoint from $\text{var}(\mathbf{C})$. Since Z does not disconnect $\mathbf{C}$, it must the case that $\mathbf{C}$ occurs in a single component of $\mathcal{B}_Z$. Observation 2 implies that $\mathbf{C}$ is also forbidden with respect to any set of variables disjoint from $\text{var}(\mathbf{C})$, and in particular with respect to Z. By Lemma 3.5, this contradicts our assumption that Z is a strong backdoor set of $\mathbf{I}$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$. This completes the proof of the lemma. □

Separators in general graphs. In order to extend the notion of separators in general graphs to those in incidence graphs of CSP instances, we need to set up some notation.

In order to distinguish between notions in standard graph terminology and those which we introduce for the purpose of this work, we will use the prefix *standard* to signify that we are referring to standard graph notation. We remark that the use of this prefix is restricted to the next subsection and so will not create a notational overhead for the reader in subsequent parts of the article.

Definition 3.8. Let G be a graph, and let $X, S \subseteq V(G)$ be disjoint vertex sets. We denote by $R_G(X, S)$ the set of vertices in connected components of $G - S$ which intersect X. For a set $Y \subseteq V(G) \setminus (X \cup S)$, we say that S is a **standard** X-Y **separator** in G. We say that S is a **minimal** standard X-Y separator if no proper subset of it is also a standard X-Y separator.

Definition 3.9. Let G be a graph, and let $X, Y \subseteq V(G)$ be disjoint vertex sets with S being a standard X-Y separator. We say that S **covers** a standard X-Y separator S' with respect to X if $R(X, S) \supseteq R(X, S')$, and we say that S and S' are **incomparable** if neither covers the other.

Definition 3.10. Let G be a graph, and let $X, Y \subseteq V(G)$ be disjoint vertex sets with S being a standard X-Y separator. We say that S is an **important** standard X-Y separator if there is no other standard X-Y separator S' such that $|S'| \leq |S|$ and S' covers S.

The following lemma is a fundamental component of all important separator based algorithms. This lemma was first proved by Chen et al. [2009] and guarantees a bound on the number of all "small" important standard X-Y separators and gives an FPT algorithm to enumerate all such standard separators. Since the proof for this statement given by Chen et al. is "hidden" within another proof in the cited paper, we point the reader to Cygan et al. [2015], which contains a detailed exposition of the same.

LEMMA 3.11 (CYAN ET AL. [2015]). *Let G be a graph, let $X, Y \subseteq V(G)$ be disjoint vertex sets, and let $k \geq 0$. Then, G has at most 4^k important standard X-Y separators of size at most k. Furthermore, there is an algorithm that, given the graph G and the sets $X, Y \subseteq V(G)$, runs in time $\mathcal{O}(4^k(m+n))$ and either concludes that G has no standard X-Y separator of size at most k or enumerates all important standard X-Y separators of size at most k, and there is an algorithm that runs in time $n^{\mathcal{O}(1)}$, which outputs one arbitrary standard X-Y separator which is not covered by any other standard X-Y separator. Here, m and n are the number of edges and vertices in the graph G, respectively.*

Separators in CSP instances. Here, we extend the notion of separators in graphs to those in incidence graphs of CSP instances. Although the majority of the definitions have a very natural extension save for minor technical alterations, we repeat most of them for the sake of completeness. For a variable set X, we denote by $\mathbf{C}(X)$ the set of all constraints whose scope has a nonempty intersection with X (equivalently, $\mathbf{C}(X)$ contains the neighbors of X in $\mathcal{B}$).

Definition 3.12. Let $\mathbf{I}$ be an instance of CSP. Let $X, S \subseteq \text{var}(\mathbf{I})$ be disjoint sets of variables, where $X \cup \mathbf{C}(X)$ induces a connected subgraph of $\mathcal{B} = \mathcal{B}(\mathbf{I})$. We denote by $R_{\mathcal{B}}(X, S)$ the set of variables and constraints that lie in the connected component containing X in $\mathcal{B} - S$ and by $R_{\mathcal{B}}[X, S]$ the set $R_{\mathcal{B}}(X, S) \cup S$. Similarly, we denote by $NR_{\mathcal{B}}(X, S)$ the set $(\text{var}(\mathbf{I}) \cup \mathbf{I}) \setminus R_{\mathcal{B}}[X, S]$ and by $NR_{\mathcal{B}}[X, S]$ the set $NR_{\mathcal{B}}(X, S) \cup S$. We drop the subscript $\mathcal{B}$ if it is clear from the context.

Note that in the case of incidence graphs, X-Y separators are defined only when $X \cup \mathbf{C}(X)$ induces a connected subgraph of $\mathcal{B} = \mathcal{B}(\mathbf{I})$. This is done simply to make the presentation of proofs easier in the latter part of the section.

Definition 3.13. Let $\mathbf{I}$ be an instance of CSP, and let $\mathcal{B} = \mathcal{B}(\mathbf{I})$. Let X and Y be disjoint variable sets.

—A variable set S disjoint from X and Y is said to **disconnect** X and Y (in $\mathcal{B}$) if $R_{\mathcal{B}}(X, S) \cap Y = \emptyset$.
—If $X \cup \mathbf{C}(X)$ induces a connected subgraph of $\mathcal{B}$ and S disconnects X and Y, then we say that S is an X-Y **separator** (in $\mathcal{B}$).
—An X-Y separator is said to be **minimal** if none of its proper subsets is an X-Y separator.
—An X-Y separator S_1 is said to **cover** an X-Y separator S with respect to X if $R(X, S_1) \supset R(X, S)$. If the set X is clear from the context, we just say that S_1 covers S.
—Two X-Y separators S and S_1 are said to be **incomparable** if neither covers the other.
—In a set $\mathcal{H}$ of X-Y separators, a separator S is said to be **component-maximal** if there is no separator S' in $\mathcal{H}$ which covers S. Similarly, a separator S is said to be **component-minimal** if there is no separator S' in $\mathcal{H}$ which is covered by S.
—An X-Y separator S_1 is said to **dominate** an X-Y separator S with respect to X if $|S_1| \leq |S|$ and S_1 covers S with respect to X. If the set X is clear from the context, we just say that S_1 dominates S.
—An X-Y separator S is said to be an **important** X-Y separator if it is minimal and there is no X-Y separator dominating S with respect to X.

Note that we require separators to only occur in the variable set of $\mathcal{B}$, as is reflected in the preceding definitions; this differs from the standard graph setting where the separators are not restricted to a strict subset of the vertex set. However, we will argue that the required results also carry over to this more general setting with a simple "path-preserving" modification of the graph under consideration.

Definition 3.14. For a graph G and any $A, X, Y \subseteq V(G)$ that are pairwise disjoint, we denote by $\zeta(G, A)$ the graph obtained from G by adding an edge between all those pairs $u, v \in V(G) \setminus A$ that lie in the neighborhood of the same connected component of $G[A]$ and then deleting the vertices in A.

LEMMA 3.15. *Let G be a graph, let $A, X, Y \subseteq V(G)$ be pairwise disjoint vertex sets, and let $G' = \zeta(G, A)$. Then the following statements hold.*

—*A set $S \subseteq V(G) \setminus A$ is a standard X-Y separator in G if and only if it is a standard X-Y separator in G'.*
—*If $G = \mathcal{B}(\mathbf{I})$ for some CSP instance $\mathbf{I}$, X and Y are disjoint variable sets of $\mathbf{I}$ and A is the set of all constraint vertices in $\mathcal{B}(\mathbf{I})$, then $S \subseteq V(G)$ is an X-Y separator in $\mathcal{B}(\mathbf{I})$ if and only if it is a standard X-Y separator in G'.*
—*If G and A are as defined earlier, then $S \subseteq V(G) \setminus A$ is an important X-Y separator in G if and only if it is an important standard X-Y separator in G'.*

PROOF. For the forward direction of the first statement, suppose that S is a standard X-Y separator in G disjoint from A and it is not a standard X-Y separator in G'. Let P be an X-Y path in $G' - S$. We can replace every edge (u, v) in P that is not in $E(G)$ with a u-v path in G whose internal vertices are all in A. As a result, we obtain an X-Y *walk* in $G - S$, implying an X-Y path in $G - S$, a contradiction. For the converse direction, let S be a standard X-Y separator in G'. By the definition of G', it follows that $S \subseteq V(G) \setminus A$. It remains to argue that it is a standard X-Y separator in G as well. Suppose that this is not the case, and let P be an X-Y path in G disjoint from S. If P is disjoint from A, then it is also present in G', a contradiction. Hence, P intersects A.

Let P' be an arbitrary maximal subpath of P whose endpoints x, y are not in A and all internal vertices are in A. Since the endpoints of P itself are disjoint from A, such a subpath exists. By the definition of G', it must be the case that $(x, y) \in E(G')$. This is because the path P' clearly ensures that x and y are both neighbors of some connected component of $G[A]$. Now, we can replace the path P' (and all such subpaths of P) with the corresponding edge in G' to obtain an X-Y path in G' that is disjoint from S, a contradiction. This completes the proof of the first statement.

The second statement is a direct consequence of the first and the definition of the graph G'. Hence, it only remains to prove the final statement. In order to prove this statement, it suffices to prove that for any two X-Y separators S_1 and S_2 in G, S_1 covers S_2 in the incidence graph G if and only if S_1 covers S_2 in G'. Observe that by the definition of G', it must be the case that $R_G(X, S) \cap var(\mathbf{I}) = R_{G'}(X, S)$ for all X-Y separators S. Now, suppose that S_1 covers S_2 in G but not in G'. That is, $R_G(X, S_1) \supseteq R_G(X, S_2)$ and $R_{G'}(X, S_1) \not\supseteq R_{G'}(X, S_2)$. But this is not possible because $R_G(X, S_1) \cap var(\mathbf{I}) = R_{G'}(X, S_1)$ and $R_G(X, S_2) \cap var(\mathbf{I}) = R_{G'}(X, S_2)$, and by our assumption, $R_G(X, S_1) \supseteq R_G(X, S_2)$.

We now consider the other direction. That is, suppose that S_1 covers S_2 in G' but not in G. Then, using the same argument as before, we conclude that there is a constraint vertex $c \in V(G)$ such that $c \in R_G(X, S_2) \setminus R_G(X, S_1)$. But this implies that a vertex v_1 corresponding to a variable in the scope of c is in the set $R_G(X, S_2)$ and no vertex corresponding to a variable in the scope of c is in $R_G(X, S_1)$. But this contradicts our assumption that S_1 covers S_2 in G'. Therefore, we conclude that S_1 also covers S_2 in G. This completes the proof of the lemma. □

As an immediate consequence of Lemmas 3.11 and 3.15, we obtain the following lemma, which plays a crucial role in our algorithm to compute nonseparating solutions.

Lemma 3.16 (Cygan et al. [2015]). *For every $k \geq 0$, there are at most 4^k important X-Y separators of size at most k. Furthermore, there is an algorithm that runs in time $\mathcal{O}(4^k k|\mathbf{I}|)$ that enumerates all such important X-Y separators, and there is an algorithm that runs in time $|\mathbf{I}|^{\mathcal{O}(1)}$ that outputs one arbitrary component-maximal X-Y separator.*

Before we proceed to the description of our algorithm to solve nonseparating instances, we make the following observation.

Observation 3. *Let S_1 and S_2 be two minimal X-Y separators in an incidence graph $\mathcal{B}$ where S_2 dominates S_1. Then, S_2 disconnects $(S_1 \setminus S_2)$ and Y.*

Proof. Note that in order to prove the statement, it suffices to prove that every vertex in $S_1 \setminus S_2$ is in the set $R(X, S_2)$. Since no vertex in Y is in the same component as a vertex of $R(X, S_2)$ in the graph $G - S_2$, the statement follows.

Now, suppose to the contrary that some vertex $u \in S_1 \setminus S_2$ is *not* in the set $R(X, S_2)$. Since S_1 is a minimal X-Y separator, it must be the case that every vertex of S_1 is adjacent to some vertex of $R(X, S_1)$. In particular, u is adjacent to a vertex $v \in R(X, S_1)$. However, since S_2 dominates S_1, it must be the case that S_2 covers S_1. By definition, we infer that $R(X, S_2) \supseteq R(X, S_1)$, implying that $v \in R(X, S_2)$ and hence u is also adjacent to a vertex in $R(X, S_2)$. Since $u \notin S_2$, this implies that $u \in R(X, S_2)$, a contradiction to our assumption that $u \in S_1 \setminus S_2$. This completes the proof of the statement. □

3.2.2. Computing Nonseparating Solutions. We begin with the following preprocessing rule, which can be applied irrespectively of the existence of a nonseparating solution.

Preprocessing Rule 1. *Let $(\mathbf{I}, k, S, W)$ be an instance of* Ext-SBD Comp*, and let Z be a connected component in $\mathcal{B}_S$ that does not contain a set of constraints forbidden with respect to S. Let* var(Z) *denote the set of variables in Z, and let* $\mathbf{C}(Z)$ *denote the set of* *constraints in Z. Construct the instance $\mathbf{I}' = \mathbf{C} \setminus \mathbf{C}(Z)$ with variable set $\text{var}(\mathbf{I}) \setminus \text{var}(Z)$, set $W' = W \setminus Z$, and return the instance $(\mathbf{I}', k, S, W')$.*

A preprocessing rule is said to be correct if the instance of Ext-SBD Comp returned by an execution of said rule is equivalent to the input instance of Ext-SBD Comp. That is, both the given instance and the returned instance are Yes instances or they are both No instances. For an instance $(\mathbf{I}, k, S, W)$ and a set Z satisfying the property mentioned in the description of the above Preprocessing Rule, the phrase *apply Preprocessing Rule 1 on* $(\mathbf{I}, k, S, W)$*,* Z refers to the action of executing Preprocessing Rule 1 with input $(\mathbf{I}, k, S, W)$ and Z. Similarly, for an instance $(\mathbf{I}, k, S, W)$, the preceding preprocessing rule is said to be *applicable* if there is a connected component Z in $\mathcal{B}_S$, which satisfies the properties required by Preprocessing Rule 1.

Lemma 3.17. *Preprocessing Rule 1 is correct. Furthermore, it can be decided in time $\mathcal{O}(|\mathcal{D}|^{|S|}|\mathbf{I}|^{\mathcal{O}(1)})$ whether this rule is applicable and, if so, applied. Finally, if the rule is not applicable, then every connected component of $\mathcal{B}_S$ intersects W.*

Proof. We first argue the correctness of the rule. Let $\mathcal{X}$ be the component of $\mathcal{B}_S$ removed by an application of the reduction rule, and let $(\mathbf{I}', k, S, W')$ be the resulting reduced instance of Ext-SBD Comp. We claim that $(\mathbf{I}, k, S, W)$ is a Yes instance if and only if $(\mathbf{I}', k, S, W')$ is a Yes instance of Ext-SBD Comp.

Since $\mathbf{I}'$ is an induced subinstance of $\mathbf{I}$, any solution Z for $(\mathbf{I}, k, S, W)$ also represents a solution $Z \setminus \mathcal{X}$ for $(\mathbf{I}', k, S, W')$. For the converse direction, consider a solution $Z' \supseteq S$ of $(\mathbf{I}', k, S, W')$. By Lemma 3.5, this implies that there is no component in $\mathcal{B}(\mathbf{I}')_{Z'}$ that contains a set of constraints forbidden with respect to Z'. But $\mathcal{X}$ also contains no set of constraints forbidden with respect to Z'. This is because $S \subseteq Z'$. Hence, we conclude that Z' is also a solution for $(\mathbf{I}, k, S, W)$. This completes the proof of correctness of the preprocessing rule.

It follows from Lemma 3.6 that the applicability of the rule can be decided in time $\mathcal{O}(|\mathcal{D}|^{|S|}|\mathbf{I}|^{\mathcal{O}(1)})$. It is straightforward to infer from the description of the rule that it can be applied in polynomial time given the instance $\mathbf{I}$ and the connected component Z.

We now prove the final statement of the lemma. Suppose that the rule is not applicable and there is a component $\mathcal{X}$ of $\mathcal{B}_S$ disjoint from W. Since the rule is not applicable, there is a set $\mathbf{C}$ of constraints in $\mathcal{X}$ forbidden with respect to S. Since $\mathcal{X}$ is disjoint from W, it must be the case that $\mathbf{C}$ is contained in a single component of $\mathcal{B}_{W \cup S}$. Furthermore, since $W \cap \text{var}(\mathbf{C}) = \emptyset$, it must be the case that $\mathbf{C}$ is also forbidden with respect to $W \cup S$ (by Observation 2), a contradiction to the assumption that $W \cup S$ is a strong backdoor set for the given CSP instance. This completes the proof of the lemma. □

Note that checking for applicability and applying the preceding rule requires time $\mathcal{O}(|\mathcal{D}|^{|S|}|\mathbf{I}|^{\mathcal{O}(1)})$, where $|S| \leq k$. Since this running time is subsumed by the time stated in Lemma 3.2, we will henceforth assume without loss of generality that we have exhaustively applied Preprocessing Rule 1 on any given instance and it is no longer applicable. The next lemma shows that if the given instance has a (not necessarily nonseparating) solution that is known to separate a singleton variable set say $\{v\}$ and the set W, then there is a set of important separators which contain some solution vertex.

Lemma 3.18. *Let $(\mathbf{I}, k, S, W)$ be an instance of* Ext-SBD Comp *and let Z be a solution for this instance. Furthermore, let v be a variable such that Z disconnects $\{v\}$ and W. Then there is a solution which contains an important v-W separator of size at most k in $\mathcal{B}_S$.*

Proof. By Lemma 3.17, since Preprocessing Rule 1 is not applicable, it must be the case that every connected component of $\mathcal{B}_S$ intersects W. In particular, the connected component of $\mathcal{B}_S$ which contains v intersects W. However, Z disconnects $\{v\}$ and W. Therefore, it must be the case that $Z \setminus S$ contains a *nonempty* set, say A which is a minimal v-W separator of size at most k in the graph $\mathcal{B}_S$. If A is also an important v-W separator in $\mathcal{B}_S$, then we are done. Suppose that this is not the case. Then there is a v-W separator in $\mathcal{B}_S$, say B, which dominates A. We claim that the set $Z' = (Z \setminus A) \cup B$ is also a solution for the given instance.

Clearly, Z' is no larger than Z, contains S and is disjoint from W. It remains to show that Z' is also a strong backdoor for $\mathbf{I}$. By Lemma 3.5, it suffices to show that there is no connected component of $\mathcal{B}_{Z'}$ that contains a set of constraints forbidden with respect to Z'. Suppose to the contrary that there is a component $\mathcal{X}$ in $\mathcal{B}_{Z'}$ containing a set of constraints forbidden with respect to Z', and let $\mathbf{C}$ be this set. We now consider two cases.

Case 1: $\mathcal{X}$ is disjoint from W. Since $\mathcal{X}$ is already disjoint from Z', it follows that it is also disjoint from S. Hence, it is disjoint from $W \cup S$ and there is a component $\mathcal{Y}$ in $\mathcal{B}_{W\cup S}$ such that $\mathcal{X} \subseteq \mathcal{Y}$. Furthermore, since Z' is disjoint from W by definition and by assumption, $\text{var}(\mathbf{C}) \subseteq \mathcal{X} \cup Z'$ we conclude that $\text{var}(\mathbf{C})$ is also disjoint from W. Therefore, we conclude that $\mathbf{C}$ is forbidden with respect to $W \cup S$ (by Observation 2). Since we have already argued that $\mathbf{C}$ is contained in a single component of $\mathcal{B}_{W\cup S}$, we infer that $W \cup S$ is not a strong backdoor for the given instance, a contradiction to the assumption that $(\mathbf{I}, k, S, W)$ is a valid instance of Ext-SBD Comp. This completes the argument for this case.

Case 2: $\mathcal{X}$ intersects W. Due to Observation 3, it must be the case that B disconnects $A \setminus B$ and W. This is because B dominates A. Furthermore, the proof of Observation 3 shows that $A \setminus B$ is contained in $R(v, B)$. Also note that it follows from the definition of Z' that $Z \setminus Z' = A \setminus B$.

We now argue that $A \setminus B$ and hence $Z \setminus Z'$ is disjoint from the component $\mathcal{X}$. If this were not the case, then there is a path in $\mathcal{B}$ from a vertex of $A \setminus B$ to a vertex of W, which is disjoint from Z'. But we have already argued that $A \setminus B$ is contained in $R(v, B)$, which is disjoint from W and $B \subseteq Z'$, implying a contradiction. Hence, we conclude that $Z \setminus Z'$ is disjoint from the component $\mathcal{X}$. This inference has two important immediate consequences. First, $\text{var}(C)$ is disjoint from $Z \setminus Z'$, and secondly, $\mathcal{X}$ (and hence $\mathbf{C}$) is contained within a connected component of $\mathcal{B}_Z$.

Finally, we argue that $\mathbf{C}$ is forbidden with respect to Z. By the first statement of Observation 2, since $\mathbf{C}$ is forbidden with respect to Z', it is forbidden with respect to every subset of Z'. Hence, we infer that $\mathbf{C}$ is forbidden with respect to $Z \cap Z'$. Since we know that $Z \setminus Z'$ is disjoint from $\text{var}(\mathbf{C})$, the second statement of Observation 2 implies that $\mathbf{C}$ is also forbidden with respect to $(Z \cap Z') \cup (Z \setminus Z') = Z$. Since we have already concluded that $\mathbf{C}$ lies in a connected component of $\mathcal{B}_Z$, we can invoke Lemma 3.5 to contradict our assumption that Z is a strong backdoor for $\mathbf{I}$. This completes the argument for the second case and hence the proof of the lemma. □

We use the above lemma along with Lemma 3.16 to obtain our algorithm for nonseparating instances.

Lemma 3.19. *Let* $(\mathbf{I}, k, S, W)$ *be a nonseparating instance of* Ext-SBD Comp. *Then it can be solved in time* $2^{\mathcal{O}(k^2)}|\mathbf{I}|^{\mathcal{O}(1)}$.

Proof. We first check using the algorithm of Lemma 3.6 whether there is a set of constraints forbidden with respect to S contained in a single component of $\mathcal{B}_S$. If there is no such set, then we correctly conclude using Lemma 3.5 that the given instance is a Yes instance and return Yes. On the other hand, if we detect such a set of constraints and $|S| = k$, then we can correctly conclude using Lemma 3.5 that the given instance is a No instance and return No.

We are now left with the case when $|S| < k$ and $\mathbf{C}$ is a set of constraints forbidden with respect to S and contained in a component of $\mathcal{B}_S$. We construct $|\text{var}(\mathbf{C}) \setminus S|$ new instances of Ext-SBD Comp, one corresponding to each variable in $\text{var}(\mathbf{C}) \setminus S$ as follows. For a variable $x \in \text{var}(\mathbf{C}) \setminus S$, we construct the instance $(\mathbf{I}, k, S_x, W)$ where $S_x = S \cup \{x\}$. We recursively run this algorithm on these instances and return Yes if at least one of the recursive calls returned Yes. Otherwise, we proceed to the next step.

In this step, for every $v \in \text{var}(\mathbf{C})$, important v-W separator A of size at most k in $\mathcal{B}_S$ and variable $u \in A$, we construct an instance $(\mathbf{I}, k, S_{v,A,u}, W)$, where $S_{v,A,u} = S \cup \{u\}$. We recursively run this algorithm on these instances and return Yes if at least one of the recursive calls returned Yes. If none of these recursive calls returned No, then we return No. This completes the description of the algorithm.

We now prove the correctness of the algorithm by induction on $k - |S|$. In the base case, when $|S| = k$, the correctness of the algorithm is clear. Hence, we consider the case when $|S| < k$. In order to prove the correctness of the algorithm in this case, we argue that there is always a solution that intersects either $\text{var}(\mathbf{C})$ or an important v-W separator for some $v \in \text{var}(\mathbf{C})$. Let $Z \supseteq S$ be a nonseparating solution for the given instance. By Lemma 3.7, either $\mathbf{C}$ is disconnected by Z or Z intersects $\text{var}(\mathbf{C})$. Suppose that $\mathbf{C}$ is not disconnected by Z. Observe that it cannot be the case that $Z \cap \text{var}(\mathbf{C}) = S$ since this would then imply that $\mathbf{C}$ is also forbidden with respect to Z, a contradiction. Therefore, if $\mathbf{C}$ is not disconnected by Z, then there is a variable, say x, in $Z \cap \text{var}(\mathbf{C})$ which is not in S. Hence, the instance $(\mathbf{I}, k, S_x, W)$ is a Yes instance, and since $k - |S_x| < k - |S|$, by the induction hypothesis we will have correctly concluded that $(\mathbf{I}, k, S_x, W)$ is a Yes instance and hence that $(\mathbf{I}, k, S, W)$ is a Yes instance.

We now describe how the algorithm accounts for the case when Z does not intersect $\text{var}(\mathbf{C})$. In this case, $\mathbf{C}$ is disconnected by Z. By Lemma 3.17, the connected component of $\mathcal{B}_S$ containing $\mathbf{C}$ also intersects W in $\mathcal{B}_S$. Since Z is a nonseparating solution, there is a connected component of $\mathcal{B}_Z$ which contains W. However, since Z must necessarily disconnect $\mathbf{C}$, it must be the case that at least one constraint, say $C \in \mathbf{C}$ is *not* contained in this connected component. Since $\text{var}(\mathbf{C}) \cap Z$ is empty, it follows that $\text{var}(C) \cap Z$ is also empty. As a result, $\text{var}(C)$ is disjoint from the connected component of $\mathcal{B}_Z$ containing W. Let $v \in \text{var}(C)$ be an arbitrary variable.

Then by Lemma 3.18, we know that there is a solution Y for the given instance that contains an important v-W separator of size at most k in $\mathcal{B}_S$. Since the connected component of $\mathcal{B}_S$ containing v also intersects W, the solution Y contains a vertex u, which is in an important v-W separator of size at most k, say A such that $u \notin S$. This implies that the instance $(\mathbf{I}, k, S_{v,A,u}, W)$ is a Yes instance, and since $k - |S_{v,A,u}| < k - |S|$, by the induction hypothesis, we will have correctly concluded that $(\mathbf{I}, k, S_{v,A,u}, W)$ is a Yes instance and hence that $(\mathbf{I}, k, S, W)$ is a Yes instance. This completes the proof of correctness of the algorithm, and we move on to the running time analysis.

We now bound the running time as follows. Observe that in all, we recurse on at most $(\rho + k)d + (\rho + k)d \cdot k \cdot 4^k$. The first term is due to the bound of $(\rho + k)d$ on the size of the set $|\text{var}(\mathbf{C})|$ and the second term is due to the size of $|\text{var}(\mathbf{C})|$ and the bound on the number of important separators given by Lemma 3.16). Since $|S|$ strictly increases in each recursive call, the depth of the resulting search tree is bounded by k. We spend time $2^{\mathcal{O}(k)}|\mathbf{I}|^{\mathcal{O}(1)}$ (due to Lemma 3.17 and Lemma 3.16) at each node of the search tree, and hence the bound on the running time follows. This completes the proof of the lemma. □

3.3. Solving General Instances

In this subsection, we describe our algorithm to solve general instances of Ext-SBD Comp by using the algorithm to check for nonseparating solutions as a subroutine. Essentially, this phase of our algorithm is a more powerful version of the algorithm described in the previous subsection. The main idea behind this part of the algorithm is the following. Since W (the old solution) has size bounded by $2k+\rho$, we can efficiently "guess" a partition of W as (W_1, W_2), where $W_1 \subset W$ is exactly the subset of W which occurs in a particular connected component after removing some hypothetical solution S. Once we guess W_1 and W_2, we know that the solution we are looking for separates W_1 and W_2. However, while it is tempting to narrow our search space down to important W_1-W_2 separators at this point, it is fairly easy to see that such an approach would be incorrect. Indeed, consider the following hypothetical example. Let $k = 2$, and let S_1 and S_2 both be W_1-W_2 separators of size 2 such that S_2 dominates S_1. Hence, S_1 *cannot* be an important W_1-W_2 separator. Furthermore, we can easily ensure that these are the *only* 2 W_1-W_2 separators of size at most 2 in the entire graph. We will now describe the constraints in the instance in such a way that S_1 is in fact the only possible solution. Suppose that $R_{\mathcal{B}_S}(W_1, S_1)$ and $R_{\mathcal{B}_S}(W_1, S_2)$ both induced connected subgraphs of $\mathcal{B}$. Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be sets of constraints that are forbidden with respect to both S_1 and S_2. Furthermore, $\mathbf{C}_2 \cap R_{\mathcal{B}_S}(W_1, S_1)$, $\mathbf{C}_2 \cap R_{\mathcal{B}_S}(W_1, S_2)$, $\mathbf{C}_2 \setminus R_{\mathcal{B}_S}(W_1, S_1)$, $\mathbf{C}_2 \setminus R_{\mathcal{B}_S}(W_1, S_2)$ are all nonempty. That is, S_1 and S_2 both disconnect the set $\mathbf{C}_2$. However, $\mathbf{C}_1 \cap R_{\mathcal{B}_S}(W_1, S_1) \neq \emptyset$, $\mathbf{C}_1 \setminus R_{\mathcal{B}_S}(W_1, S_1) \neq \emptyset$, and $\mathbf{C}_1 \subseteq R_{\mathcal{B}_S}(W_1, S_2)$. That is, *only* S_1 disconnects $\mathbf{C}_1$ and $\mathbf{C}_1$ lies within a connected component of $\mathcal{B}_{S_2}$. Finally, suppose that these are the only forbidden sets of constraints with respect to S_1 or S_2. Now, due to Lemma 3.5, only S_1 can be a solution for this instance. Hence, computing the set of important W_1-W_2 separators alone does not help us in solving general instances.

However, while we are not able to narrow our search space of W_1-W_2 separators to only important W_1-W_2 separators, we show that it is indeed possible to prune the search space down to a set of separators that is much larger than the set of important separators, but whose size is bounded by a function of k nevertheless. Once we do that, the rest of the algorithm is a branching algorithm searching through this space. The main technical content in this part of our algorithm lies in showing that it is sufficient to restrict our search to an *efficiently computable* bounded set of separators. We next give a brief description of the approach we follow to achieve this objective.

At a high level, we use the approach introduced in Lokshtanov and Ramanujan [2012]. However, there are significant obstacles that arise due to the fact that we are dealing with scattered classes of CSPs. The crux of the idea is the following. We define a laminar family of W_1-W_2 separators that have a certain monotonicity property. Informally speaking, we partition the separators into "good" and "bad" separators so that (under some ordering) all the good separators occur continuously followed by all the bad separators. Following this, we pick the middle separators in this family—the "last" good separator and the "first" bad separator—and show that deleting either of these separators must necessarily disconnect the hypothetical solution we are attempting to find. Roughly speaking, once we have computed the laminar family of separators, we delete the middle separators and perform the same procedure recursively on the connected component intersecting W_1. Since the solution has size bounded by k, it cannot be broken up more than k times and hence the number of levels in the recursion is also bounded by k. We then show that essentially the union of the middle separators computed at the various levels of recursion of this algorithm has size $f(k)$ and furthermore it is sufficient to restrict our search for a W_1-W_2 separator to this set.

We begin by defining a *connecting gadget* that consists of redundant constraints and whose purpose is purely to encode connectivity at crucial points of the algorithm.

Definition 3.20. Let $\mathbf{I}$ be a CSP instance, and let $X = \{x_1, \ldots, x_\ell\}$ be a set of variables. Let $\mathbf{I}'$ be the instance obtained from $\mathbf{I}$ as follows. Add $\ell - 1$ new tautological binary constraints $T_1, \ldots, T_{\ell-1}$, and for each $i \in [\ell - 1]$, define the scope of T_i as $\{x_i, x_{i+1}\}$. We refer to $\mathbf{I}'$ as *the instance obtained from* $\mathbf{I}$ *by adding the connecting gadget on* X.

LEMMA 3.21. *Let $(\mathbf{I}, k, S, W)$ be an instance of* EXT-SBD COMP *and let Z be a separating solution for this instance. Let $\mathcal{X}$ be a component of $\mathcal{B}_Z$ and let $W_1 = W \cap \mathcal{X}$. Let $\mathbf{I}'$ be the CSP instance obtained from $\mathbf{I}$ be adding the connecting gadget on W_1. Then Z is also a solution for $(\mathbf{I}', k, S, W)$.*

PROOF. Let $\mathcal{B}'$ be the incidence graph of the CSP instance $\mathbf{I}'$. If Z is not a solution for the instance $(\mathbf{I}', k, S, W)$, then there must be a component of $\mathcal{B}'_Z$ containing a set $\mathbf{C}$ of constraints forbidden with respect to Z. Observe that since by assumption the languages $\Gamma_1, \dots, \Gamma_d$ all contain the tautological binary relation, $\mathbf{C}$ is disjoint from the constraints that were added to $\mathbf{I}$ to construct $\mathbf{I}'$. Finally, any set of constraints from $\mathbf{I}$ which occur together in the same component of $\mathcal{B}_Z$ also occur together in the same component of $\mathcal{B}'_Z$ and vice versa. This implies that $\mathbf{C}$ is a set of constraints forbidden with respect to Z and is contained in a single component of $\mathcal{B}_Z$, a contradiction. This completes the proof of the lemma. □

From this point on, we assume that if an instance $(\mathbf{I}, k, S, W)$ of EXT-SBD COMP is a *separating* instance, then it will be represented as a tuple $(\mathbf{I}, k, S, W_1, W_2)$, where $W_1 \subset W$ and $W_2 = W \setminus W_1$ with the connecting gadget added on W_1. Note that, since $|W| \leq 2k + \rho$, we will later on be allowed to branch over all partitions of W into W_1 and W_2 in time $2^{2k+\rho}$. Our objective now is to check if there is a strong backdoor set for $\mathbf{I}$ extending S, disjoint from W and separating W_1 from W_2. For this, we need to introduce the notions of tight separator sequences and pattern replacement procedures.

3.3.1. Tight Separator Sequences. Let $\mathbf{I}$ be a set of constraints and let Y be a subgraph of $\mathcal{B} = \mathcal{B}(\mathbf{I})$. We use $\mathbf{I}[Y]$ to denote $\mathbf{I} \cap Y$. For expositional clarity, we will usually enforce $\text{var}(\mathbf{I} \cap Y)$ to also lie in Y. That is, the variables in the scope of all constraints whose corresponding vertex is in Y, are also present in Y.

Definition 3.22. Let $(\mathbf{I}, k, S, W_1, W_2)$ be an instance of EXT-SBD COMP. We call a W_1-W_2 separator X in $\mathcal{B}_S$ *ℓ-good* if there is a variable set $K \subseteq R_{\mathcal{B}_S}(W_1, X)$ of size at most ℓ such that $K \cup X \cup S$ is a strong backdoor set of $\mathbf{I}[R_{\mathcal{B}_S}[W_1, X] \cup S]$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$, and we call it *$\ell$-bad* otherwise.

LEMMA 3.23 (MONOTONOCITY LEMMA). *Let $(\mathbf{I}, k, S, W_1, W_2)$ be an instance of* EXT-SBD COMP, *and let X and Y be disjoint W_1-W_2 separators in $\mathcal{B}_S$ such that X covers Y. If X is* ℓ-good, *then so is Y. Consequently, if Y is* ℓ-bad, *then so is X.*

PROOF. Suppose that X is ℓ-good, and let K be a variable set of size at most ℓ such that $K \cup X \cup S$ is a strong backdoor set into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$ for the subinstance $\mathbf{I}' = \mathbf{I}[R_{\mathcal{B}_S}[W_1, X] \cup S]$. Let $K' = K \cap R_{\mathcal{B}_S}[W_1, Y]$. We claim that Y is ℓ-good and that $P = K' \cup Y \cup S$ is a strong backdoor set for the subinstance $\hat{\mathbf{I}} = \mathbf{I}[R_{\mathcal{B}_S}[W_1, Y] \cup S]$.

If this were not the case, then there is a set $\mathbf{C}$ of constraints that are contained in a single component of $\mathcal{B}(\hat{\mathbf{I}}) - P$ and are forbidden with respect to the set P. By the first statement of Observation 2, we conclude that $\mathbf{C}$ is also forbidden with respect to $K' \cup S$.

Now, we observe that since $\mathbf{C}$ lies in the set $R_{\mathcal{B}_S}[W_1, Y]$, no constraint in $\mathbf{C}$ can have in its scope a variable in $X \cup (K \setminus K')$. This is because otherwise at least one vertex in X or $K \setminus K'$ would be contained in the set $R_{\mathcal{B}_S}[W_1, Y]$. But in the former case, we obtain a contradiction to the assumption in the premise of the lemma that X covers Y and in the latter case, we obtain a contradiction to the definition of K', which is defined to be the set of *all* vertices in K that are also in $R_{\mathcal{B}_S}[W_1, Y]$. Therefore, by the second statement of Observation 2, $\mathbf{C}$ being forbidden with respect to $K' \cup S$ implies that it is also forbidden with respect to $(K' \cup S) \bigcup (X \cup (K \setminus K')) = X \cup K \cup S$.

Finally, since $\mathbf{C}$ lies in a single component of $\mathcal{B}(\hat{\mathbf{I}}) - P$ and $(K \setminus K') \cup X$ is disjoint from $R_{\mathcal{B}_S}[W_1, Y] \cup S$, it must be the case that $\mathbf{C}$ also lies in a single component of

$\mathcal{B}(\mathbf{I}') - (K \cup X \cup S)$. But this results in a contradiction to our assumption that $K \cup X \cup S$ is a strong backdoor set for the instance $\mathbf{I}'$. This completes the proof of the lemma. □

Definition 3.24. Let $(\mathbf{I}, k, S, W_1, W_2)$ be an instance of Ext-SBD Comp, and let $\mathcal{B} = \mathcal{B}(\mathbf{I})$. Let X and Y be W_1-W_2 separators in $\mathcal{B}_S$ such that Y dominates X. Let ℓ be the smallest i for which X is i-good. If Y is ℓ-good, then we say that Y **well-dominates** X. If X is ℓ-good and there is no $Y \neq X$ that well-dominates X, then we call X **ℓ-important**.

The following lemma is a analogous to (indeed a strengthening of) Lemma 3.18 and describes when one W_1-W_2 separator is *at least as good as* another W_1-W_2 separator with respect to the solution.

Lemma 3.25. *Let* $(\mathbf{I}, k, S, W_1, W_2)$ *be an instance of* Ext-SBD Comp, *let* $\mathcal{B} = \mathcal{B}(\mathbf{I})$, *and let Z be a solution for this instance. Let $P \subseteq Z \setminus S$ be a nonempty minimal W_1-W_2 separator in $\mathcal{B}_S$, and let P' be a W_1-W_2 separator in $\mathcal{B}_S$ well-dominating P. Then there is also a solution for the given instance containing P'.*

Proof. Let $K \subseteq R_{\mathcal{B}_S}(W_1, P)$ be a minimal subset of Z such that $K \cup P \cup S$ is a strong backdoor set of the instance $\mathbf{I}' = \mathbf{I}[R_{\mathcal{B}_S}[W_1, P] \cup S]]$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$. Let $\ell = |K|$, and let $Q = (Z \cap [R_{\mathcal{B}_S}[W_1, P]) \cup S$. We first argue that $Q = K \cup P \cup S$. That is, there is *no other* vertex of Z in $K \subseteq R_{\mathcal{B}_S}[W_1, P]$ other than those in $K \cup P$.

Claim 1. $Q = K \cup P \cup S$.

Proof. Suppose to the contrary that there is a vertex $z \in (Z \setminus Q) \cap R_{\mathcal{B}_S}[W_1, P]$. Since $P \subseteq Q$, it must be the case that $z \in (Z \setminus Q) \cap R_{\mathcal{B}_S}(W_1, P)$. We now argue that $\hat{Z} = Z \setminus \{z\}$ is a strong backdoor of $\mathbf{I}$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$, contradicting the minimality of Z.

Suppose to the contrary that $\hat{Z}$ is not a strong backdoor of $\mathbf{I}$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$, and let $\mathcal{X}$ be a component of $\mathcal{B}_{\hat{Z}}$ containing a set $\mathbf{C}$ of constraints forbidden with respect to $\hat{Z}$. Clearly, it cannot be the case that $\mathcal{X}$ is disjoint from $R_{\mathcal{B}_S}(W_1, P)$ because that would imply that $\mathcal{X}$ is also a component of $\mathcal{B}_Z$ which contains a set of constraints forbidden with respect to Z. Hence, it must be the case that $\mathcal{X}$ intersects $R_{\mathcal{B}_S}(W_1, P)$. Furthermore, since $P \subseteq \hat{Z}$, it must be the case that $\mathcal{X} \subseteq R(W_1, P)$ and hence $\text{var}(\mathbf{C}) \subseteq R_{\mathcal{B}_S}[W_1, P]$. By the first statement of Observation 2, this implies that $\mathbf{C}$ is forbidden with respect to $(\hat{Z} \cap R_{\mathcal{B}_S}[W_1, P]) \cup S = ((Z \cap (R_{\mathcal{B}_S}[W_1, P])) \setminus \{z\}) \cup S$. However, by the definition of z, we know that $((Z \cap (R_{\mathcal{B}_S}[W_1, P])) \setminus \{z\}) \cup S \supseteq K \cup P \cup S$. Hence, by the first statement of Observation 2, we conclude that $\mathbf{C}$ is forbidden with respect to $K \cup P \cup S$. But this contradicts the definition of K as a minimal subset of Z such that $K \cup P \cup S$ is a strong backdoor of $\mathbf{I}[R_{\mathcal{B}_S}[W_1, P] \cup S]$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$. Hence, we conclude that $Q = K \cup P \cup S$, completing the proof of the claim. □

From the preceding claim, it follows that Q is a strong backdoor set of the instance $\mathbf{I}' = \mathbf{I}[R_{\mathcal{B}_S}[W_1, P] \cup S]]$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$. Now, since P is ℓ-good and P' well-dominates P, it follows that P' dominates P *and* P' is also ℓ-good. That is, there is a set $K' \subseteq R_{\mathcal{B}_S}(W_1, P')$ of size at most ℓ such that $Q' = K' \cup P' \cup S$ is a strong backdoor of the instance $\mathbf{I}' = \mathbf{I}[R_{\mathcal{B}_S}[W_1, P'] \cup S]]$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$.

We claim that $Z' = (Z \setminus Q) \cup Q'$ is a solution for the instance $(\mathbf{I}, k, S, W_1, W_2)$. Since $|Q| = |K| + |P| + |S|$, $|Q'| = |K'| + |P'| + |S|$, $|K'| \leq |K|$, and $|P'| \leq |P|$, it follows that $|Q'| \leq |Q|$. Hence, Z' is no larger than Z. It now remains to prove that Z' is a strong backdoor set of $\mathbf{I}$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$. Suppose that Z' is not a strong backdoor set, and let $\mathcal{X}$ be a connected component of $\mathcal{B}_{Z'}$ containing a set $\mathbf{C}$ of constraints forbidden with respect to Z'. We now break into the following two cases.

Case 1: $\mathcal{X} \cap R_{\mathcal{B}_S}(W_1, P') = \emptyset$. In this case, $\mathcal{X}$ is also disjoint from $R_{\mathcal{B}_S}[W_1, P']$ and $\text{var}(\mathbf{C}) \cap R_{\mathcal{B}_S}(W_1, P') = \emptyset$. That is, no constraint in $\mathbf{C}$ has in its scope a variable in $R_{\mathcal{B}_S}(W_1, P')$. By the first statement of Observation 2, $\mathbf{C}$ is also forbidden with respect to every subset of Z' and in particular with respect to the subset $J = (Z' \setminus R_{\mathcal{B}_S}[W_1, P']) \cup (P' \cap Z)$. But by the definition of Z', we know that $J \subseteq Z$ and furthermore, $Z \setminus J \subseteq R_{\mathcal{B}_S}(W_1, P)$, which is in turn disjoint from $\text{var}(\mathbf{C})$. Hence, by the second statement of Observation 2, we conclude that $\mathbf{C}$ is also forbidden with respect to $J \cup (Z \setminus J) = Z$.

Finally, since P' dominates P, it follows that $R_{\mathcal{B}_S}(W_1, P) \subset R_{\mathcal{B}_S}(W_1, P')$. Hence, it must be the case that there is a connected component $\mathcal{Y}$ in $\mathcal{B}_Z$ such that $\mathcal{X} \subseteq \mathcal{Y}$. Therefore, we conclude that $\mathbf{C}$ is a set of constraints forbidden with respect to Z and occurring inside a connected component of $\mathcal{B}_Z$. This contradicts our assumption that Z is a strong backdoor of $\mathbf{I}$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$.

Case 2: $\mathcal{X} \cap R_{\mathcal{B}_S}(W_1, P') \neq \emptyset$. By the definition of Z', it must be the case that $\mathcal{X}$ is contained in the set $R_{\mathcal{B}_S}(W_1, P')$. Then, $\text{var}(\mathbf{C}) \setminus R_{\mathcal{B}_S}[W_1, P'] = \emptyset$. That is, no constraint in $\mathbf{C}$ has in its scope a variable outside $R_{\mathcal{B}_S}[W_1, P']$, and hence $Z' \setminus Q'$ is disjoint from $R_{\mathcal{B}_S}(W_1, P')$. By the first statement of Observation 2, it follows that $\mathbf{C}$ is also forbidden with respect to every subset of Z' and in particular, with respect to the subset $J = (Z' \cap R_{\mathcal{B}_S}[W_1, P']) \cup S$, which is by definition equal to Q'.

Finally, since $\mathcal{X}$ is contained in the set $R_{\mathcal{B}_S}(W_1, P')$, it follows that $\mathcal{X}$ is also a connected component in the graph $\mathcal{B}(\mathbf{I}'')_S - Q'$. Hence, we conclude that $\mathbf{C}$ is forbidden with respect to Q' and lies in a connected component of $\mathcal{B}'_S - Q'$ where $\mathcal{B}' = \mathcal{B}(\mathbf{I}')$, contradicting our assumption that Q' is a strong backdoor set of $\mathbf{I}'$ into $\text{CSP}(\Gamma_1) \oplus \cdots \oplus \text{CSP}(\Gamma_d)$.

Having obtained a contradiction in either case, we conclude that Z' is indeed a solution for the instance $(\mathbf{I}, k, S, W_1, W_2)$, and since $Z' \subseteq P'$, the lemma follows.

Lemmas 3.18 and 3.25 imply that it is sufficient to compute either a variable separated from W by some solution or a separator well-dominating (the separating part of) some solution. Furthermore, Lemma 3.25 allows us to restrict our attention to ℓ-important separators for values of $\ell \leq k$. In the rest of this section, we describe a subroutine that runs in FPT time and always achieves one of the aforementioned objectives. We use the notion of tight separator sequences (defined later in the text) to streamline our search for this variable/separator.

Definition 3.26. For every $k \geq 1$, a tight X-Y separator sequence of order k is a set $\mathcal{H}$ of X-Y separators with the following properties.

—Every separator has size at most k.
—The separators are pairwise disjoint.
—For any pair of separators in the set, one covers the other.
—The set is maximal with respect to the above properties.

See Figure 2 for an illustration of a tight separator sequence.

Lemma 3.27. *Given a CSP instance* $\mathbf{I}$*, disjoint variable sets* X*,* Y*, and an integer* k*, a tight* X-Y *separator sequence* $\mathcal{H}$ *of order* k *can be computed in time* $|\mathbf{I}|^{\mathcal{O}(1)}$.

Proof. If there is no X-Y separator of size at most k, then we stop the procedure. Otherwise, we compute an arbitrary component-maximal X-Y separator S of size at most k. This can be done in polynomial time by the algorithm of Lemma 3.16. We add this separator to the family $\mathcal{H}$, set $Y := S$, and iterate this process. We claim that the resulting set is a tight X-Y separator sequence of order k. It is clear that the first three properties of a tight separator sequence are always satisfied in any iteration. As for the maximality of the set $\mathcal{H}$ that is finally computed, observe that if there is a

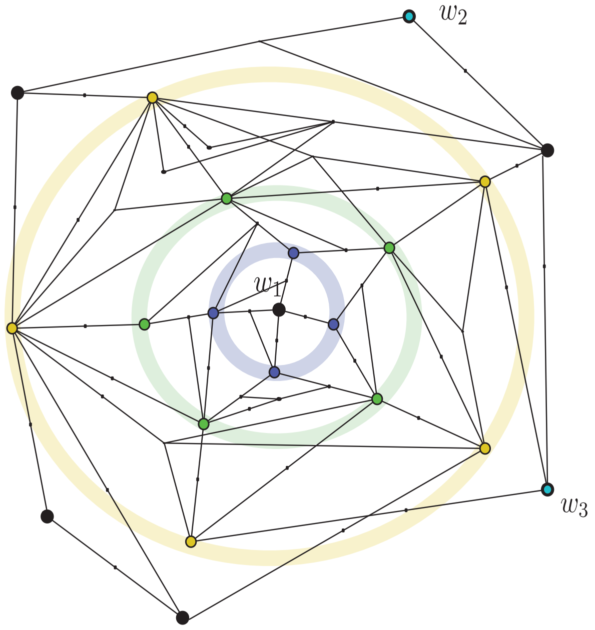

Fig. 2. An illustration of a tight w_1-$\{w_2, w_3\}$ separator sequence. Each shaded layer corresponds to a separator in the sequence.

separator, say P, that can be added to the $\mathcal{H}$ without violating maximality, then it either contradicts the component-maximality of at least one separator in $\mathcal{H}$ or it contradicts the termination of the process. The former case occurs if P covers at least one separator in $\mathcal{H}$ and the latter occurs if P is covered by all separators in $\mathcal{H}$. This completes the proof of the lemma. $\square$

3.3.2. Boundaried CSPs and Replacements.

Definition 3.28. A t-boundaried CSP instance is a CSP instance $\mathbf{I}$ with t distinguished labeled variables. The set $\partial(\mathbf{I})$ of labeled variables is called the **boundary** of $\mathbf{I}$ and the variables in $\partial(\mathbf{I})$ are referred to as the **terminal variables**. Let $\mathbf{I}_1$ and $\mathbf{I}_2$ be two t-boundaried CSP instances and let $\mu : \partial(\mathbf{I}_1) \to \partial(\mathbf{I}_2)$ be a bijection. We denote by $\mathbf{I}_1 \otimes_\mu \mathbf{I}_2$ the t-boundaried CSP instance obtained by the following **gluing** operation. We take the disjoint union of the constraints in $\mathbf{I}_1$ and $\mathbf{I}_2$ and then identify each variable $x \in \partial(\mathbf{I}_1)$ with the corresponding variable $\mu(x) \in \partial(\mathbf{I}_2)$.

We also define the notion of boundaried CSP instances with an annotated set of variables. The key difference between the boundary and the annotation is that the annotated set of variables plays no part in gluing operations. Formally,

Definition 3.29. A t-boundaried CSP instance with an annotated set is a t-boundaried CSP instance $\mathbf{I}$ with a second set of distinguished but unlabeled vertices disjoint from the boundary. The set of annotated vertices is denoted by $\Delta(\mathbf{I})$.

Before proceeding to the technical Lemma 3.30, we give an informal outline of its claim and intended use. Consider an instance of EXT-SBD COMP with a solution Z that is disjoint from and incomparable to some ℓ-good separator P. Then some part of $Z \setminus S$, say K, lies in $R_{B_S}[W_1, P] \cup S$. We show that, by carefully replacing parts outside of $R_{B_S}[W_1, P] \cup S$ with a small gadget, we can obtain an instance $\mathbf{I}'$, which preserves the part of K inside $R_{B_S}[W_1, P] \cup S$. We also show that some part of K had to lie outside of $R_{B_S}[W_1, P] \cup S$, and hence the solution we seek in $\mathbf{I}'$ is strictly smaller than in $\mathbf{I}$; once we find a solution in $\mathbf{I}'$, we can use it to find one in $\mathbf{I}$. Furthermore, the number of possible boundaried instances that need to be considered for this replacement is bounded by a function of k, which allows exhaustive branching.

OBSERVATION 4. *Let S_1 and S_2 be two disjoint and incomparable X-Y separators. Then, $S_2^r = R(X, S_1) \cap S_2$, $S_1^r = R(X, S_2) \cap S_1$ are both nonempty. Furthermore, $S_2^{nr} = S_2 \setminus S_2^r$ and $S_1^r = S_1 \setminus S_1^r$ are also both nonempty.*

LEMMA 3.30. *Let $(\mathbf{I}, k, S, W_1, W_2)$ be an instance of* EXT-SBD COMP, *and let Z be a solution for this instance. Let Q be a minimal part of $Z \setminus S$ separating W_1 from W_2 in $\mathcal{B}_S$, let $K = (Z \cap R_{\mathcal{B}_S}[W_1, Q])$, and let $\ell = |K \setminus Q|$. Let P be a minimal W_1-W_2 separator in $\mathcal{B}_S$ that is disjoint from K and incomparable with Q, let Q^r be $Q \cap R_{\mathcal{B}_S}(W_1, P)$ and $Q^{nr} = Q \setminus Q^r$. Similarly, let P^r be $P \cap R_{\mathcal{B}_S}(W_1, Q)$ and $P^{nr} = P \setminus P^r$, and suppose that $W_1 \cup P^r$ has the connecting gadget on it. Let $K^r = (K \cap R_{\mathcal{B}_S}[W_1, P])$. Let $\mathbf{I}_1 = \mathbf{I}[R_{\mathcal{B}_S}[W_1, P] \cup S]$ be a boundaried CSP instance with $P^r \cup S$ as the boundary.*

Then, there exists a $|P^r \cup S|$-boundaried CSP instance $\hat{\mathbf{I}}$ with an annotated set of variables, and a bijection $\mu : \partial(\mathbf{I}) \to P^r \cup S$ such that the glued CSP instance $\mathbf{I}' = \mathbf{I}_1 \otimes_\mu \hat{\mathbf{I}}$ has the following properties.

(1) *The set $W_1 \cup P^{nr} \cup S$ is a strong backdoor set into $\mathrm{CSP}(\Gamma_1) \oplus \cdots \oplus \mathrm{CSP}(\Gamma_d)$ in the CSP instance $\mathbf{I}'$.*
(2) *The set Q^r is a $|K^r \setminus Q^r|$-good W_1-P^{nr} separator in $\mathcal{B}_{S\cup\Delta(\hat{\mathbf{I}})}(\mathbf{I}') = \mathcal{B}'_{S\cup\Delta(\hat{\mathbf{I}})}$.*
(3) *For any Q' that is a W_1-P^{nr} separator in $\mathcal{B}'_{S\cup\Delta(\hat{\mathbf{I}})}$ well dominating Q^r in $\mathbf{I}'$, the set $Q' \cup Q^{nr}$ well dominates the W_1-W_2 separator Q in $\mathcal{B}_S$.*
(4) *If v is a variable disconnected from $W_1 \cup P^{nr}$ by K^r in $\mathcal{B}'_{S\cup\Delta(\hat{\mathbf{I}})}$, then v is in $R(W_1, P)$ and v is disconnected from $W_1 \cup W_2$ by K in $\mathcal{B}_S$.*
(5) *There is a constant η and a family $\mathcal{H}$ of boundaried CSP instances with an annotated variable set such that $\mathcal{H}$ contains $\hat{\mathbf{I}}$, has size bounded by $2^{2^{\eta k}}$ and can be computed in time $2^{2^{\eta k}} k^{\mathcal{O}(1)}$.*

PROOF. We first describe the instance $\hat{\mathbf{I}}$ and then prove that it has the properties claimed by the lemma. Let $K^{nr} = (K \setminus K^r) \cup Q^{nr}$. Consider the subinstance $\mathbf{I}_2 = \{C \in \mathbf{I} : \mathrm{var}(C) \subseteq (NR_{\mathcal{B}_S}(W_1, P) \cap R_{\mathcal{B}_S}(W_1, Q)) \cup P^r \cup K^{nr}\}$. In other words, we take the set of constraints containing variables that are either disconnected from W_1 by P but not disconnected from W_1 by Q, or occur in $P^r \cup K^{nr}$. Two vertices $v, w \in \mathcal{B}_{S\cup K^{nr}}(\mathbf{I}_2)$ are $\mathbf{I}_2$-*connected* if they are connected in $\mathcal{B}_{S\cup K^{nr}}(\mathbf{I}_2)$; similarly, a vertex set A of $\mathcal{B}_{S\cup K^{nr}}(\mathbf{I}_2)$ is $\mathbf{I}_2$-connected if A is a connected vertex set in the graph $\mathcal{B}_{S\cup K^{nr}}(\mathbf{I}_2)$. Notice that P^r is $\mathbf{I}_2$-connected by assumption.

We now perform the following marking scheme on this hypothetical subinstance $\mathbf{I}_2$. For every assignment τ of $S \cup K^{nr}$, and for each $J \subseteq [d]$, if there exists a set $\mathbf{C}$ of constraints such that

(1) $\mathbf{C}|_\tau$ is J-forbidden (w.r.t. $\emptyset$), and
(2) $\mathbf{C} \cup P^r$ is $\mathbf{I}_2$-connected,

then we *mark* the constraints in one such set $\mathbf{C}$. Since Q is ℓ-good and in particular since $Q \cup K \cup S$ is a strong backdoor set in $\mathbf{I}[R_{\mathcal{B}_S}[W_1, Q] \cup S]$, every relation occurring in $\mathbf{C}$ belongs to one of the finite languages $\Gamma_1, \ldots, \Gamma_d$, and hence $|\mathbf{C}|$ does not depend on k. Since the number of possible assignments τ is bounded by $2^{\mathcal{O}(k)}$, we observe that the set M of all marked constraints has cardinality $2^{\mathcal{O}(k)}$.

To complete our construction of $\hat{\mathbf{I}}$, we begin by setting $\hat{\mathbf{I}} = \mathbf{I}_2[M \cup \mathrm{var}(M) \cup P^r \cup S \cup K^{nr} \cup Q^{nr}]$. We then add the connecting gadget on the set $(\mathrm{var}(M) \cup P^r) \setminus (S \cup K^{nr})$. Finally, we define the boundary of $\hat{\mathbf{I}}$ to be $P^r \cup S$ and define the annotated set $\Delta(\mathbf{I})$ to be the set K^{nr}. From the bound on $|M|$, it is readily observed that $|\hat{\mathbf{I}}| \leq 2^{\mathcal{O}(k)}$. We now prove that $\mathbf{I}' = \mathbf{I}_1 \otimes_{identity} \hat{\mathbf{I}}$ satisfies the properties claimed by the lemma.

Claim 2. *The set $W_1 \cup P^{nr} \cup S$ is a strong backdoor set into* $\mathrm{CSP}(\Gamma_1) \oplus \cdots \oplus \mathrm{CSP}(\Gamma_d)$ *in the CSP instance* $\mathbf{I}'$.

We actually prove a stronger claim, specifically that already $W_1 \cup S$ is a strong backdoor set (into $\mathrm{CSP}(\Gamma_1) \oplus \cdots \oplus \mathrm{CSP}(\Gamma_d)$) in $\mathbf{I}'$. Assume the converse; then there exists a set $\mathbf{C}$ of forbidden constraints in $\mathbf{I}'$ with respect to $W_1 \cup S$ that occur in a connected component of $\mathcal{B}_{W_1 \cup S}(\mathbf{I}')$. We first show that $\mathbf{C}$ must also be connected in $\mathcal{B}_{W_1 \cup S \cup W_2}(\mathbf{I})$. Consider any path between $v, w \in \mathrm{var}(\mathbf{C})$ in $\mathcal{B}_{W_1 \cup S}(\mathbf{I}')$, and some constraint $T = (S, \mathcal{D}^2)$ on this path that is not present in $\mathcal{B}_{W_1 \cup S \cup W_2}(\mathbf{I})$; observe that T must be a connecting gadget. Let v', w' be the neighbors of T. By construction, v', w' are $\mathbf{I}_2$-connected and hence also connected in $\mathcal{B}_S(\mathbf{I})$. Since $(W_1 \cup S \cup W_2) \cap \mathcal{B}_S(\mathbf{I}') = \emptyset$, it follows that any such v', w' and consequently also v, w are connected in $\mathcal{B}_{W_1 \cup S \cup W_2}(\mathbf{I})$.

So, $\mathbf{C}$ is also connected in $\mathcal{B}_{W_1 \cup S \cup W_2}(\mathbf{I})$. Since $\mathrm{var}(\mathbf{C}) \cap W_2 = \emptyset$, we conclude that $\mathbf{C}$ is also forbidden with respect to $W_1 \cup S \cup W_2$. This contradicts the fact that $W_1 \cup W_2 \cup S$ is a solution to our initial instance of Ext-SBD Comp.

Claim 3. *The set Q^r is a $|K^r \setminus Q^r|$-good W_1-P^{nr} separator in $\mathcal{B}_{S \cup \Delta(\hat{\mathbf{I}})}(\mathbf{I}') = \mathcal{B}'_{S \cup \Delta(\hat{\mathbf{I}})}$.*

We first prove that Q^r is indeed a separator as claimed. Suppose for a contradiction that there exists a path α in $\mathcal{B}'_S$ between $v \in W_1$ and $w \in P^{nr}$. By definition, Q^r is a W_1-P^{nr} separator in $\mathcal{B}_S$, and so α must necessarily contain an edge $v'w'$ in $\mathcal{B}'_{S \cup \Delta(\hat{\mathbf{I}})} - Q^r$, which is not in $\mathcal{B}_S - Q^r$; by construction, this implies that $v'w'$ are $\mathbf{I}_2$-connected. For any such $v'w'$, there hence exists some path α' in $\mathcal{B}_S(\mathbf{I}_2) \subseteq \mathcal{B}_S$, and from this it follows that v, w are also connected in $\mathcal{B}_S - Q_r$. This contradicts the fact that Q^r is a W_1-P^{nr} separator in $\mathcal{B}_S$, and so Q^r must also be a W_1-P^{nr} separator in $\mathcal{B}'_{S \cup \Delta(\hat{\mathbf{I}})}$.

Next, we argue that Q^r is $|K^r \setminus Q^r|$-good in $\mathcal{B}'_{S \cup K^{nr}}$, and that this is in fact witnessed by $K^r \setminus Q^r$. Indeed, assume for a contradiction that there exists a set $\mathbf{C}$ of constraints in $\mathbf{I}'$ that are connected and forbidden with respect to $S \cup K^{nr} \cup Q^r \cup (K^r \setminus Q^r) = S \cup K$. Clearly, none of the constraints in $\mathbf{C}$ contain the relation $\mathcal{D}^2$, and hence $\mathbf{C} \subseteq \mathbf{I}$. Furthermore, $\mathbf{C}$ is also connected in $\mathbf{I}$ by the same path argument as in Claim 2: any path between $c_1, c_2 \in \mathbf{C}$ that contains edges either exists in $\mathcal{B}_{S \cup K}$ or can be replaced by a new path in $\mathcal{B}_{S \cup K}$ that uses $\mathbf{I}_2$-connectivity to circumvent connectivity constraints. But since $\mathrm{var}(\mathbf{C}) \cap (S \cup K)$ is the same in $\mathbf{I}$ and $\mathbf{I}'$, we conclude that $\mathrm{var}(\mathbf{C})$ is forbidden with respect to $S \cup K$ in $\mathbf{I}$. This yields a contradiction with Q being ℓ-good in $\mathbf{I}$ as witnessed by $K \setminus Q$.

Claim 4. *For any Q' that is a W_1-P^{nr} separator in $\mathcal{B}'_{S \cup \Delta(\hat{\mathbf{I}})}$ well dominating Q^r in $\mathbf{I}'$, the set $Q' \cup Q^{nr}$ well dominates the W_1-W_2 separator Q in $\mathcal{B}_S$.*

The set $\hat{Q} = Q' \cup Q^{nr}$ dominates Q in $\mathcal{B}_S$ by definition; therefore, it suffices to prove that $\hat{Q}$ is ℓ-good in $\mathcal{B}_S$. Let Y' be the variable set certifying that Q' is $|K^r \setminus Q^r|$-good in $\mathcal{B}'_{S \cup K^{nr}}$. We claim that $Y = Y' \cup (K^{nr} \setminus Q^{nr})$ certifies that $\hat{Q}$ is ℓ-good in $\mathcal{B}_S$, and argue that $|Y| \leq \ell$. Since $|Y'| \leq |K^r \setminus Q^r|$ by assumption and both Y' and $|K^r \setminus Q^r|$ are disjoint from $K^{nr} \setminus Q^{nr}$, it follows that $|Y| \leq |(K^r \setminus Q^r) \cup (K^{nr} \setminus Q^{nr})| = |(K \setminus Q)| = \ell$.

Now, assume for a contradiction that there exists a set $\mathbf{C}$ of constraints in $\mathbf{I}[R_{\mathcal{B}_S}[W_1, \hat{Q}] \cup S]$ which are connected and forbidden with respect to $\hat{Q} \cup S \cup Y$. Observe that $\mathbf{C} \not\subseteq \mathbf{I}_1$, since then $\mathbf{C}$ would also be forbidden with respect $S \cup K^{nr} \cup Q' \cup Y'$ and connected in $\mathcal{B}'_S$, which would contradict our assumption on Q'. So it must be the case that $\mathbf{C}_2 = \mathbf{C} \cap \mathbf{I}_2$ is nonempty. First, we consider the simpler case where $\mathbf{C}_2 \subseteq \hat{\mathbf{I}}$; by the construction of $\hat{\mathbf{I}}$ and in particular the addition of the connectivity gadgets, it follows that $\mathbf{C}$ would then also be connected in $\mathbf{I}'$ and hence forbidden in $\mathbf{I}'$ with respect to $S \cup K^{nr} \cup Q' \cup Y'$. This excludes the existence of any set $\mathbf{C}$ of forbidden constraints with respect to $\hat{Q} \cup S \cup Y$ such that $\mathbf{C} \subseteq \mathbf{I}'$.

Next, we proceed the general case where $\mathbf{C}_2$ contains a subset of constraints, say $\mathbf{C}_2^+$, which are not contained in $\mathbf{I}'$ and which are pairwise $\mathbf{I}_2$-connected; we refer to any such set $\mathbf{C}_2^+$ as a *leftover* of $\mathbf{C}$. Let us now fix $\mathbf{C}$ to be some set of forbidden constraints in $\mathbf{I}$, which has a minimum number of leftovers; in the previous paragraph, we have argued that $\mathbf{C}$ must contain at least one leftover, and we will reach a contradiction by showing that there exists a set $\mathbf{C}'$ of forbidden constraints with respect $\hat{Q} \cup S \cup Y$ with less leftovers than $\mathbf{C}$.

Let τ be the assignment of $S \cup \hat{Q} \cup Y$ certifying that $\mathbf{C}$ is forbidden in $\mathbf{I}$, and let τ' be the restriction of τ to $K^{nr} \cup S$. Let $\mathbf{C}_2^+$ be some leftover of $\mathbf{C}$, and let $J \subseteq [d]$ be such that $\mathbf{C}_2^+|_\tau$ is J-forbidden with respect to $S \cup Y_2'$ in $\mathbf{I}$. Since $\mathbf{C}_2^+$ was not marked during our construction of $\hat{\mathbf{I}}$, there must exist another "marked" set of constraints $\mathbf{C}_2^M$ in $\hat{\mathbf{I}}$ with the following properties:

—$\mathbf{C}_2^M|_{\tau'}$ is also J-forbidden (w.r.t. $\emptyset$), and
—$\mathbf{C}_2^M \cup P^r$ is $\mathbf{I}_2$-connected.

To finish the argument, let $\mathbf{C}^M = (\mathbf{C} \setminus \mathbf{C}_2^+) \cup \mathbf{C}_2^M$. By construction, $\mathbf{C}^M$ is connected in $\mathcal{B}_{S\cup\hat{Q}\cup Y}(\mathbf{I})$; indeed, C_2^M is pairwise $\mathbf{I}_2$-connected and is connected to at least one variable $p \in P^r$, which was $\mathbf{I}_2$-connected to $\mathbf{C}_2^+$, which in turn guarantees connectivity to the rest of $\mathbf{C}$. Furthermore, by the construction of $\mathbf{C}^M$ we observe that $\mathbf{C}^M \setminus \mathbf{C}_2^M$ is $([d] \setminus J)$-forbidden with respect to $S \cup \hat{Q} \cup Y$ and $\mathbf{C}_2^M$ is J-forbidden with respect to $S \cup \hat{Q} \cup Y$ (because $(S \cup \hat{Q} \cup Y) \cap \mathbf{I}_2 = K^{nr} \cup S$); so, $\mathbf{C}^M$ must be forbidden with respect to $S \cup \hat{Q} \cup Y$ in $\mathbf{I}$. Since $\mathbf{C}^M$ has one less leftover than $\mathbf{C}$, we have reached a contradiction to the existence of $\mathbf{C}$.

Claim 5. *If v is a variable disconnected from $W_1 \cup P^{nr}$ by K^r in $\mathcal{B}'_{S\cup\Delta(\hat{\mathbf{I}})}$, then v is in $R(W_1, P)$ and v is disconnected from $W_1 \cup W_2$ by K in $\mathcal{B}_S$.*

By construction of $\hat{\mathbf{I}}$, it is easy to see that v is not separated from P^r by K^r in $\mathcal{B}'_{S\cup\Delta(\hat{\mathbf{I}})}$, and due to the connecting gadget on $W_1 \cup P^r$, v is not separated from W_1 as well. We now prove the second part of the claim. Assume for a contradiction that there exists a path α from v to $W_1 \cup W_2$ in $\mathcal{B}_{S\cup K}$. First, consider the case where α does not intersect $\mathbf{I}_2$. Then α must exist in $\mathbf{I}_1$ and in particular also in $\mathcal{B}'_{S\cup K^{nr}}$, which contradicts our assumption on v.

On the other hand, assume α does intersect $\mathbf{I}_2$. Since P^r is by definition a W_1-$(\mathbf{I}_2 \setminus P^r)$ separator in $\mathcal{B}_S$, this means that α must intersect P^r; let a be the first vertex in P^r on the path α from v. Since α ends in $W_1 \cup W_2$, neither of which intersect with $\mathcal{B}_{S\cup K^{nr}}(\mathbf{I}_2)$, there must be a last vertex b in $\mathcal{B}(\mathbf{I}_2)$ on α (in other words, the path leaves $\mathcal{B}_{S\cup K^{nr}}(\mathbf{I}_2)$ from b and does not return there). Since $b \notin Q \subseteq K$ by assumption, it must follow that $b \in P^r$. But then a, b are connected by a connectivity gadget in $\mathbf{I}'$, and hence there is a path of length 2 between a and b in $\mathcal{B}'_{S\cup K^{nr}}$ which guarantees the existence of a v-$(W_1 \cup W_2)$ path α' in $\mathcal{B}'_{S\cup K^{nr}}$, a contradiction.

Claim 6. *There is a constant η and a family $\mathcal{H}$ of boundaried CSP instances such that $\mathcal{H}$ contains $\hat{\mathbf{I}}$, has size bounded by $2^{2^{\eta k}}$ and can be computed in time $2^{2^{\eta k}} k^{\mathcal{O}(1)}$.*

For the proof of the final statement, observe that $\hat{\mathbf{I}}$ is an instance containing $2^{\mathcal{O}(k)}$ constraints and variables, at most k of which are marked. Since the number of such marked instances is bounded by $2^{2^{\mathcal{O}(k)}}$ and these can be enumerated in time $2^{2^{\mathcal{O}(k)}}$, the proof of the lemma is complete. □

3.3.3. The Algorithm for General Instances. The goal of this section is to show how our algorithm handles general instances. Our first order of business is proving Lemma 3.31. This lemma shows that when solving an instance $(\mathbf{I}, k, S, W_1, W_2)$ of Ext-SBD Comp, if S *already* intersects all W_1-W_2 paths, then we can simply focus on solving a nonseparating instance induced on a carefully chosen subset of variables and constraints. This immediately leads to Preprocessing Rule 2. Lemma 3.32 then forms the core component of our algorithm. We conclude this section by summarizing the algorithm (Lemma 3.33) and giving a proof of Lemma 3.2.

Lemma 3.31. *Let $(\mathbf{I}, k, S, W_1, W_2)$ be an instance of* Ext-SBD Comp, *let Z be a solution for this instance, and let $Z' = Z \cap R_{\mathcal{B}}[W_1, S]$. Let $\mathbf{I}'$ denote the instance $\mathbf{I}[R_{\mathcal{B}}[W_1, S]]$. If S disconnects W_1 and W_2, then $(\mathbf{I}', |Z'|, S, W_1)$ is a nonseparating* Yes *instance of* Ext-SBD Comp *and conversely for any nonseparating solution Z'' for the instance $(\mathbf{I}', |Z'|, S, W_1)$, the set $\hat{Z} = (Z \setminus Z') \cup Z''$ is a solution for the original instance.*

Proof. Suppose that Z' is not a strong backdoor set for the instance $(\mathbf{I}', |Z'|, S, W_1)$. Let $\mathcal{B}'$ be the incidence graph of the instance $\mathbf{I}'$. Then, some component of $\mathcal{B}'_{Z'}$ contains a set $\mathbf{C}$ of constraints forbidden with respect to Z'. However, since S disconnects these constraints from $Z \setminus Z'$, it must be the case that these constraints also occur in a single connected component of $\mathcal{B}_Z$. Also, since $Z \setminus Z'$ is disjoint from var($\mathbf{C}$), by Observation 2 we know that $\mathbf{C}$ is also forbidden with respect to $Z' \cup (Z \setminus Z') = Z$, a contradiction. Therefore, Z' is indeed a strong backdoor set for the instance $(\mathbf{I}', |Z'|, S, W_1)$. Since the connecting gadget has been added on W_1 (by assumption), it must be the case that Z' is a nonseparating solution for the instance $(\mathbf{I}', |Z'|, S, W_1)$.

For the converse direction, suppose that the set $\hat{Z}$ is not a solution for the original instance, and let $\mathbf{C}$ be a set of constraints in a connected component of $\mathcal{B}_{\hat{Z}}$ that is forbidden with respect to $\hat{Z}$. Clearly, $\mathbf{C}$ is contained entirely inside one of the sets $R_{\mathcal{B}}(W_1, S)$ or $NR_{\mathcal{B}}(W_1, S)$. We first consider the case when $\mathbf{C}$ is contained in $R_{\mathcal{B}}(W_1, S)$. Then, it must be the case that $\mathbf{C}$ is also in a single connected component of $\mathcal{B}' - Z''$. However, since Z'' is assumed to be a strong backdoor set for the CSP instance $\mathbf{I}'$, $\mathbf{C}$ is not forbidden with respect to Z'' and hence also not forbidden with respect to $\hat{Z}$. On the other hand, we consider the case when $\mathbf{C}$ is contained in $NR_{\mathcal{B}}(W_1, S)$. Then $\mathbf{C}$ must be contained in some component of $NR_{\mathcal{B}}(W_1, S) - \hat{Z}$ and in particular in some component of $NR_{\mathcal{B}}(W_1, S) - (Z \setminus Z'') = NR_{\mathcal{B}}(W_1, S) - (Z \setminus Z')$. Also, since Z' is disjoint from var($\mathbf{C}$), by Observation 2 we know that $\mathbf{C}$ is also forbidden with respect to $Z' \cup (Z \setminus Z') = Z$, a contradiction. □

Preprocessing Rule 2. *Let $(\mathbf{I}, k, S, W_1, W_2)$ be an instance of* Ext-SBD Comp. *If S disconnects W_1 from W_2, then compute a nonseparating solution Z' for the instance $(\mathbf{I}', k', S, W_1)$, where $\mathbf{I}'$ denotes the instance $\mathbf{I}[R_{\mathcal{B}}[W_1, S]]$ and k' is the least possible value of $i \leq k$ such that $(\mathbf{I}', i, S, W_1)$ is a* Yes *instance. Delete Z', and return the instance $(\mathbf{I} - Z', k - |Z'|, S, W_2)$.*

It follows from Lemma 3.31 that the preceding rule is correct (see paragraph following Preprocessing Rule 1 for definition of a rule being correct), and we obtain a bound on the running time from that of the algorithm of Lemma 3.19. Henceforth, we assume that in any given instance of Ext-SBD Comp, the preceding rule is not applicable. We now move to the description of the subroutine, which is at the heart of our main algorithm. Recall that every solution Z to an instance of Ext-SBD Comp by assumption contains

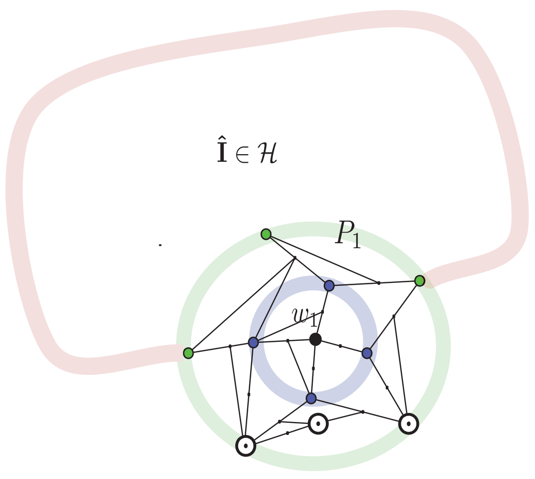

Fig. 3. An illustration of the instance obtained by gluing some $\hat{\mathbf{I}} \in \mathcal{H}$ to the instance $\mathbf{I}[R[W_1, P_1] \cup S]$.

an ℓ-good W_1-W_2 separator X (here, as well as further on, by separator we implicitly mean a separator in $\mathcal{B}_S$ unless stated otherwise).

Lemma 3.32. *Let $(\mathbf{I}, k, S, W_1, W_2)$ be an instance of* Ext-SBD Comp*, and let $0 \leq \ell, \lambda \leq k$. There is an algorithm that, given a tuple $< (\mathbf{I}, k, S, W_1, W_2), \lambda, \ell >$ runs in time $2^{2^{\mathcal{O}(k)}}|\mathbf{I}|^{\mathcal{O}(1)}$ and returns a set $\mathcal{R}$ which is disjoint from S and contains at most $2^{2^{\mathcal{O}(k)}}$ variables such that for every ℓ-important W_1-W_2 separator X of size at most λ in $\mathcal{B}_S$ and for every solution $Z \supseteq X$ for the given instance of* Ext-SBD Comp*,*

—*$\mathcal{R}$ intersects X or*
—*there is a variable in $\mathcal{R}$ which is separated from W by Z or*
—*$\mathcal{R}$ intersects $Z \setminus S$.*

Proof. Recall that Preprocessing Rule 2 is assumed to have been exhaustively applied, and hence, there must be some W_1-W_2 path in $\mathcal{B}_S$. Similarly, if there is no W_1-W_2 separator of size at most λ in $\mathcal{B}_S$, then we return No, that is, the tuple is invalid. Otherwise, we execute the algorithm of Lemma 3.27 to compute a tight W_1-W_2 separator sequence $\mathcal{I}$ of order λ. We then partition $\mathcal{I}$ into ℓ-good and ℓ-bad separators. We do this by testing each separator in the sequence for the presence of nonseparating solutions (this is sufficient due to the presence of the connecting gadget, which is assumed to have been placed on W_1), and this can be done in time $2^{\mathcal{O}(\ell^2)}|\mathbf{I}|^{\mathcal{O}(1)}$ by invoking Lemma 3.19. Now, let P_1 be component-maximal among the ℓ-good separators in $\mathcal{I}$ (if any exist), and let P_2 be component-minimal among the ℓ-bad separators in $\mathcal{I}$ (if any exist). We set $\mathcal{R} := P_1 \cup P_2$. For each $i \in \{1, 2\}$, we now do the following.

We execute the algorithm of Lemma 3.30, Claim 6 to compute a family $\mathcal{H}$ of bounded-aried CSP instances with an annotated set of variables. Then, for every choice of $P_i^r \subseteq P_i$, for every instance $\hat{\mathbf{I}} \in \mathcal{H}$ with $|P_i^r \cup S|$ terminals and for every possible bijection $\delta : \partial(\hat{\mathbf{I}}) \to P_i^r \cup S$, we construct the glued CSP instance $\mathbf{I}_{P_i^r,\delta} = \mathbf{I}[R[W_1, P_i] \cup S] \otimes_\delta \hat{\mathbf{I}}$ (see Figure 3, where the boundary of the instance $\mathbf{I}[R[W_1, P_i] \cup S]$ is defined as $P_i^r \cup S$ and the connecting gadget is added on $W_1 \cup P_i^r$. We then recursively invoke this algorithm on the tuple $< (\mathbf{I}_{P_i^r,\delta}, k - j, S \cup \tilde{S}, W_1, P_i \setminus P_i^r), \lambda', \ell' >$ for every $0 \leq \lambda' < \lambda$, $1 \leq j \leq k - 1$ and $0 \leq \ell' \leq \ell$, where $\tilde{S}$ is the annotated set of variables in $\hat{\mathbf{I}}$. We add the union of the variable sets returned by these recursive invocations to $\mathcal{R}$ and return the resulting set. This completes the description of the algorithm. We now proceed to the proof of correctness of this algorithm.

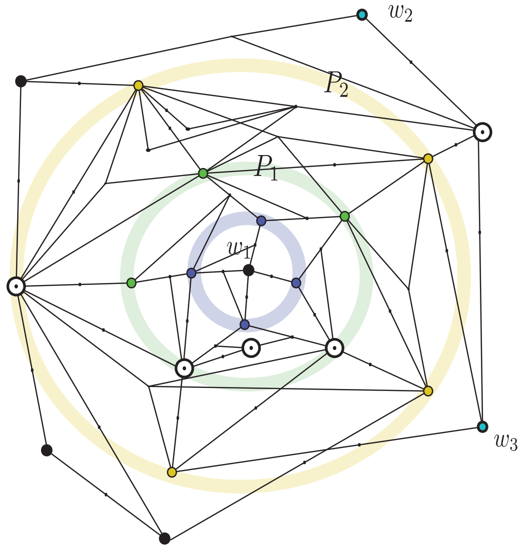

Fig. 4. An illustration of the case where X (the dotted circles) is incomparable with P_1 and P_2.

Correctness. We prove the correctness by induction on λ. Consider the base case, when $\lambda = 1$ and there is a path from W_1 to W_2 in $\mathcal{B}_S$. We argue the correctness of the base case as follows. Let X be an ℓ-important W_1-W_2 separator; since X has size 1, it cannot be incomparable with any distinct W_1-W_2 separator of size 1. Therefore, X has to be equal to P_1 or covered by P_1 (P_1 exists since X itself is ℓ-good by assumption). In either case, we are correct. The former case is trivially accounted for because P_1 is contained in $\mathcal{R}$, and the latter case is accounted for because then P_1 would clearly well-dominate X, which itself is ℓ-important, leading to a contradiction. We now move to the induction step with the induction hypothesis being that the algorithm runs correctly (the output satisfies the properties claimed in the statement of the lemma) on all tuples where $\lambda < \hat{\lambda}$ for some $\hat{\lambda} \geq 2$. Now, consider an invocation of the algorithm on a tuple with $\lambda = \hat{\lambda}$.

Let Z be a solution for this instance containing the ℓ-important separator X, that is, $X \subseteq (Z \setminus S)$, of size at most λ. If X intersects $P_1 \cup P_2$, then the algorithm is correct because $\mathcal{R}$ intersects $Z \setminus S$. Therefore, we may assume that X is disjoint from $P_1 \cup P_2$. Now, suppose that X is covered by P_1. In this case, since P_1 is also ℓ-good and has size at most λ, by the definition of well-domination, P_1 well-dominates X, contradicting our assumption that X is ℓ-important.

Similarly, by the Monotonocity Lemma (Lemma 3.23), because X is ℓ-good and P_2 is not, it cannot be the case that X covers P_2. Now, suppose that X covers P_1 and is itself covered by P_2. However, due to the maximality of the tight separator sequence, X must be contained in $\mathcal{I}$. But this contradicts our assumption that P_1 is a component-maximal ℓ-good separator in the sequence $\mathcal{I}$ and $P_1 \neq X$.

Finally, we are left with the case when X is incomparable with P_1 (if P_1 is defined) or P_2 (otherwise). Without loss of generality, suppose that X is incomparable with P_1 (see Figure 4). The argument for the case when P_1 does not exist is analogous and follows by simply replacing P_1 with P_2 in the proof.

Let $K \subseteq Z$ be a strong backdoor set for $\mathbf{I}[R[W_1, X] \cup S]$ extending $X \cup S$. If $P_1 \cap K$ is nonempty, then $P_1 \cap Z$ is nonempty as well, and since $\mathcal{R}$ contains the vertices in P_1, the algorithm is correct. Therefore, we assume that P_1 and K are disjoint. Now, let $X^r = R_{\mathcal{B}_S}(W_1, P_1) \cap X$ and $X^{nr} = X \setminus X^r$. Similarly, let $P_1^r = R_{\mathcal{B}_S}(W_1, X) \cap P_1$ and $P_1^{nr} = P_1 \setminus P_1^r$. Since X and P_1 are incomparable, the sets X^r, X^{nr}, P_1^r and P_1^{nr} must all be nonempty. Let $K^r = (K \cap R(W_1, P)) \cup S$. Furthermore, if any variable in P_1^r is not in the same component as W_1 in $\mathcal{B}_Z$, then $\mathcal{R}$ contains a variable separated from W by Z, also implying that the algorithm is correct. Hence, we may assume that P_1^r is contained in the same component as W_1 in $\mathcal{B}_Z$. Observe that the sets defined earlier satisfy the premises of Lemma 3.30 with $P = P_1$ and $Q = X$ in the statement of the lemma. Therefore, there is a $|P_1^r \cup S|$-boundaried instance $\hat{\mathbf{I}}$ with an annotated set $\tilde{S}$ and an appropriate bijection $\mu : \partial(\hat{\mathbf{I}}) \to P_1^r \cup S$ with the properties claimed in the statement of Lemma 3.30. Now, consider the recursion of the algorithm on the tuple $<(\mathbf{I}_{P_1^r,\mu}, k_1, S \cup \tilde{S}, W_1, P_1^{nr}), \lambda', \ell'>$, where $\mathbf{I}_{P_1^r,\mu}$ is the instance obtained by gluing together $\mathbf{I}[R[W_1, P_1] \cup S]$ (with $P_1^r \cup S$ as the boundary) and $\hat{\mathbf{I}}$ via the bijection μ with the connecting gadget added on $W_1 \cup P_1^r$, $\lambda' = |X^r|$, $k_1 = |K^r|$, and $\ell' = |K^r|$.

In order to apply the induction hypothesis on the execution of the algorithm on this tuple, we need to prove that the tuple is "valid," that is, it satisfies the conditions in the premise of the lemma. In order to prove this, it is sufficient for us to prove that $(\mathbf{I}_{P_1^r,\mu}, k_1, S \cup \tilde{S}, W_1, P_1^{nr})$ is indeed a valid instance of Ext-SBD Comp. For this to hold, it must be the case that $W_1 \cup P_1^{nr} \cup S$ is a strong-backdoor set for the CSP instance $\mathbf{I}_{P_1^r,\mu}$. But this property is indeed guaranteed by Lemma 3.30, Claim 2. Therefore, the tuple satisfies the conditions in the premise of the lemma, and since $\lambda' < \lambda = \hat{\lambda}$, we may apply the induction hypothesis.

Since X is ℓ-important, from Lemma 3.30, Claims 3 and 4 it follows that X^r must also be k_1-important in $\mathbf{I}_{P_1^r,\mu}$. By the induction hypothesis, the algorithm is correct on this tuple and the returned set, call it $\mathcal{R}'$, either intersects X^r (due to the aforementioned argument using Claims 3 and 4), or contains a variable separated from $W_1 \cup P^{nr}$ by K^r or contains a variable in K^r. By Lemma 3.30, Claim 5 it holds that in the second case $\mathcal{R}'$ contains a variable separated from W by Z. In the third case, $\mathcal{R}'$ contains a variable in $Z \setminus S$ because $K^r \subseteq K \subseteq Z$ and $\mathcal{R}'$ is disjoint from $S \cup \tilde{S} \supseteq S$. Since $\mathcal{R}' \subseteq \mathcal{R}$, we conclude the correctness of the algorithm, and we now move on to the analysis of the running time and the size of the returned set $\mathcal{R}$.

Bounding the set $\mathcal{R}$. Recall that since λ, which is bounded above by k, is required to be nonnegative in a valid tuple, the depth of the search tree is bounded by k. Furthermore, the number of branches initiated at each node of the search tree is at most $k^3 \cdot k! \cdot g(k)$, where $g(k)$ is the number of boundaried CSP instances in the set $\mathcal{H}$ (k^3 for the choice of λ', j and ℓ' and $k!$ for the choice of the bijection δ). Since Lemma 3.30 guarantees a bound of $2^{2^{\eta k}}$ on $|\mathcal{H}|$ for some constant η, we conclude that the number of internal nodes in the search tree is bounded by $2^{2^{\eta' k}}$ for some constant η'. Finally, since at each internal node we add at most $2k$ vertices (corresponding to $P_1 \cup P_2$), we conclude that the set returned has size $2^{2^{\mathcal{O}(k)}}$.

Running time. The analysis for the running time is similar to the proof of the bound on $\mathcal{R}$. We already have a bound on the number of nodes of the search tree. The claimed bound on the running time of the whole algorithm follows from the observation that the time spent at each node of the search tree is dominated by the time required to execute the algorithms of Lemmas 3.19 and 3.30, which in turn is bounded by $2^{2^{\mathcal{O}(k)}}|\mathbf{I}|^{\mathcal{O}(1)}$. This completes the proof of the lemma. □

Lemma 3.33. *There is an algorithm that, given an instance* $(\mathbf{I}, k, S, W_1, W_2)$ *of* Ext-SBD Comp, *runs in time* $2^{2^{\mathcal{O}(k)}}|\mathbf{I}|^{\mathcal{O}(1)}$ *and either computes a solution for this instance which is a* W_1-W_2 *separator or correctly concludes that no such solution exists.*

Proof. For every $0 \leq \ell, \lambda \leq k$, we invoke Lemma 3.32 on the tuple $<(\mathbf{I}, k, S, W_1, W_2), \lambda, \ell>$ to compute a set $\mathcal{R}_{\ell,\lambda}$. We then set $\mathcal{R}$ to be the set obtained by taking the union of the sets $\mathcal{R}_{\ell,\lambda}$ for all possible values of ℓ and λ. Following that, we simply branch on which vertex v in $\mathcal{R}$ is added to S, creating a new instance $(\mathbf{I}, k, S \cup \{v\}, W_1, W_2)$ of Ext-SBD Comp. If $|S \cup \{v\}| > k$, we return No. If Preprocessing Rule 2 applies, we use it to reduce the instance; if this results in a nonseparating instance, we use Lemma 3.19 to solve the instance. Otherwise, we iterate all of the above on this new instance.

The bound on the running time of this algorithm follows from the total depth of the branching tree being bounded by k and the width of each branch being bounded by $|\mathcal{R}| \leq 2^{2^{\mathcal{O}(k)}}$ (the cost of branching over ℓ, λ and of applying Lemma 3.19 and Preprocessing Rule 2 is dominated by the above). The correctness follows from the correctness of Lemma 3.32, of Preprocessing Rule 2, and of Lemma 3.19. This completes the proof of the lemma, and we now have our algorithm to handle separating instances of Ext-SBD Comp. $\square$

We conclude this section by combining the algorithms for separating and nonseparating instances to present our complete algorithm for Ext-SBD Comp (Lemma 3.2).

Proof of Lemma 3.2. Let $(\mathbf{I}, k, S, W)$ be the input instance of Ext-SBD Comp. We first apply Lemma 3.19 to check if there is a non-separating solution for this instance. If not, then we branch over all $W_1 \subset W$ and for each such choice of W_1 we add the connecting gadget on W_1 and apply Lemma 3.33 to check if $(\mathbf{I}, k, S, W_1, W_2 = W \setminus W_1)$ has solution which is a W_1-W_2 separator. The correctness and claimed running time bound both follow from those of Lemmas 3.19 and 3.33. $\square$

4. CONCLUDING REMARKS

We have presented an FPT algorithm that can find strong backdoors to scattered base classes of CSP and #CSP. This algorithm allows us to lift known tractability results based on constraint languages from instances over a single tractable language to instances containing a mix of constraints from distinct tractable languages. The instances may also contain constraints that only belong to a tractable language after the backdoor variables have been instantiated, where different instantiations may lead to different tractable languages. Formally, we have applied the algorithm to CSP and #CSP, but it clearly applies also to other versions of CSP, such as MAX-CSP (where the task is to simultaneously satisfy a maximum number of constraints) or various forms of weighted or valued CSPs.

Our work opens up several avenues for future research. First, the runtime bounds for finding backdoors to scattered base classes provided in this work are very likely suboptimal due to us having to obtain a unified algorithm for *every* scattered set of finite constraint languages. Therefore, it is quite likely that a refined study for scattered classes of specific constraint languages using their inherent properties will yield significantly better runtimes. Second, graph modification problems and in particular the study of efficiently computable modulators to various graph classes has been an integral part of parameterized complexity and has led to the development of several powerful tools and techniques. We believe that the study of modulators to "scattered graph classes" could prove equally fruitful and, as our techniques are mostly graph based, our results as well as techniques could provide a useful starting point toward future research in this direction.

REFERENCES

Christian Bessiere, Clément Carbonnel, Emmanuel Hebrard, George Katsirelos, and Toby Walsh. 2013. Detecting and exploiting subproblem tractability. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence,* Francesca Rossi (Ed.). IJCAI/AAAI.

Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. 2009. (Meta) kernelization. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS'09)*. IEEE Computer Soc., Los Alamitos, CA, 629–638.

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

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

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

Andrei A. Bulatov and Víctor Dalmau. 2007. Towards a dichotomy theorem for the counting constraint satisfaction problem. *Inf. Comput.* 205, 5 (2007), 651–678.

Clément Carbonnel, Martin C. Cooper, and Emmanuel Hebrard. 2014. On backdoors to tractable constraint languages. In *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming (CP'14),* Vol. 8656. Springer Verlag, 224–239.

Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. 2006. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.* 72, 8 (2006), 1346–1367.

Jianer Chen, Yang Liu, and Songjian Lu. 2009. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica* 55, 1 (2009), 1–13.

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

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

Nadia Creignou. 1995. A dichotomy theorem for maximum generalized satisfiability problems. *J. Comput. Syst. Sci.* 51, 3 (1995), 511–522.

Nadia Creignou and Miki Hermann. 1996. Complexity of generalized satisfiability counting problems. *Information and Computation* 125, 1 (1996), 1–12.

Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer. DOI:http://dx.doi.org/10.1007/978-3-319-21275-3

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

R. G. Downey and M. R. Fellows. 1999. *Parameterized Complexity*. Springer Verlag, New York.

Rodney G. Downey and Michael R. Fellows. 2013. *Fundamentals of Parameterized Complexity*. Springer.

Tomás Feder and Moshe Y. Vardi. 1993. Monotone monadic SNP and constraint satisfaction. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal (Eds.). ACM, 612–622.

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

Jörg Flum and Martin Grohe. 2006. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series, Vol. XIV. Springer Verlag, Berlin.

Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, M. S. Ramanujan, and Saket Saurabh. 2015. Solving d-SAT via backdoors to small treewidth. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*. 630–641. DOI:http://dx.doi.org/10.1137/1.9781611973730.43

Serge Gaspers, Neeldhara Misra, Sebastian Ordyniak, Stefan Szeider, and Stanislav Zivny. 2014. Backdoors into heterogeneous classes of SAT and CSP. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence,* Carla E. Brodley and Peter Stone (Eds.). AAAI Press, 2652–2658.

Serge Gaspers, Sebastian Ordyniak, M. S. Ramanujan, Saket Saurabh, and Stefan Szeider. 2013. Backdoors to q-Horn. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS'13)*, Natacha Portier and Thomas Wilke (Eds.), Vol. 20. Leibniz-Zentrum fuer Informatik, 67–79.

Serge Gaspers and Stefan Szeider. 2013. Strong backdoors to bounded treewidth SAT. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS'13).* IEEE Computer Society, 489–498.

Parikshit Gopalan, Phokion G. Kolaitis, Elitza Maneva, and Christos H. Papadimitriou. 2009. The connectivity of boolean satisfiability: computational and structural dichotomies. *SIAM J. Comput.* 38, 6 (2009), 2330–2355.

Pavol Hell and Jaroslav Nesetril. 2008. Colouring, constraint satisfaction, and complexity. *Comput. Sci. Rev.* 2, 3 (2008), 143–163.

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

Peter Jeavons, David Cohen, and Marc Gyssens. 1997. Closure properties of constraints. *J. ACM* 44, 4 (1997), 527–548.

Phokion G. Kolaitis. 2003. Constraint satisfaction, databases, and logic. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*. Morgan Kaufmann, 1587–1595.

Phokion G. Kolaitis and Moshe Y. Vardi. 2007. A logical approach to constraint satisfaction. In *Finite Model Theory and its Applications*. Springer Verlag, 339–370.

Vladimir Kolmogorov and Stanislav Živný. 2013. The complexity of conservative valued CSPs. *J. ACM* 60, 2 (2013), Art. 10, 38.

Daniel Lokshtanov and M. S. Ramanujan. 2012. Parameterized tractability of multiway cut with parity constraints. In *ICALP 2012, Automata, Languages, and Programming. Part I (Lecture Notes in Computer Science)*, Vol. 7391. Springer Verlag, 750–761.

Dániel Marx. 2006. Parameterized graph separation problems. *Theoret. Comput. Sci.* 351, 3 (2006), 394–406.

Ugo Montanari. 1974. Networks of constraints: Fundamental properties and applications to picture processing. *Inf. Sci.* 7 (1974), 95–132.

Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. 2004. Detecting backdoor sets with respect to horn and binary clauses. In *Proceedings of 7th International Conference on Theory and Applications of Satisfiability Testing.* 96–103.

Christos H. Papadimitriou and Mihalis Yannakakis. 1999. On the complexity of database queries. *J. Comput. Syst. Sci.* 58, 3 (1999), 407–427.

Igor Razgon and Barry O'Sullivan. 2009. Almost 2-SAT is fixed parameter tractable. *J. Comput Syst. Sci.* 75, 8 (2009), 435–450.

Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. 2004. Finding odd cycle transversals. *Oper. Res. Lett.* 32, 4 (2004), 299–301.

Thomas J. Schaefer. 1978. The complexity of satisfiability problems. In *Conference Record of the 10th Annual ACM Symposium on Theory of Computing*. ACM, 216–226.

Johan Thapper and Stanislav Zivny. 2013. The complexity of finite-valued CSPs. In *Proceedings of the Symposium on Theory of Computing Conference (STOC'13).* ACM, 695–704.

Ryan Williams, Carla Gomes, and Bart Selman. 2003a. Backdoors to typical case complexity. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, Georg Gottlob and Toby Walsh (Eds.). Morgan Kaufmann, 1173–1178.

Ryan Williams, Carla Gomes, and Bart Selman. 2003b. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search. In *Informal Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*. 222–230.

Received October 2015; revised October 2016; accepted November 2016