Beyond Hypertree Width: Decomposition Methods Without Decompositions

Hubie Chen and Víctor Dalmau

Departament de Tecnologia
Universitat Pompeu Fabra
Barcelona, Spain
{hubie.chen,victor.dalmau}@upf.edu

Abstract. The general intractability of the constraint satisfaction problem has motivated the study of restrictions on this problem that permit polynomial-time solvability. One major line of work has focused on structural restrictions, which arise from restricting the interaction among constraint scopes. In this paper, we engage in a mathematical investigation of generalized hypertree width, a structural measure that has up to recently eluded study. We obtain a number of computational results, including a simple proof of the tractability of CSP instances having bounded generalized hypertree width.

1 Introduction

The *constraint satisfaction problem* (CSP) is widely acknowledged as a convenient framework for modelling search problems. Instances of the CSP arise in a variety of domains, including artificial intelligence, database theory, algebra, propositional logic, and graph theory. An instance of the CSP consists of a set of constraints on a set of variables; the question is to determine if there is an assignment to the variables satisfying all of the constraints. Alternatively, the CSP can be cast as the fundamental algebraic problem of deciding, given two relational structures $\mathbf{A}$ and $\mathbf{B}$, whether or not there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. In this formalization, each relation of $\mathbf{A}$ contains the tuples of variables that are constrained together, which are often called the *constraint scopes*, and the corresponding relation of $\mathbf{B}$ contains the allowable tuples of values that the variable tuples may take.

It is well-known that the CSP, in its general formulation, is NP-complete; this general intractability has motivated a large and rich body of research aimed at identifying and understanding restricted cases of the CSP that are polynomial-time tractable. The restrictions that have been studied can, by and large, be placed into one of two categories, which–due to the homomorphism formulation of the CSP–have become known as *left-hand side* restrictions and *right-hand side* restrictions. From a high level view, left-hand side restrictions, also known as *structural restrictions*, arise from prespecifying a class of relational structures $\mathcal{A}$ from which the left-hand side structure $\mathbf{A}$ must come, while right-hand side restrictions arise from prespecifying a class of relational structures $\mathcal{B}$ from which the right-hand side structure $\mathbf{B}$ must come. As this paper is concerned principally with structural restrictions, we will not say more about right-hand side restrictions than that their systematic study has origins in a classic theorem of Schaefer [21], and that recent years have seen some exciting results on them (for instance [4, 5]).

The structural restrictions studied in the literature can all be phrased as restrictions on the hypergraph $H(\mathbf{A})$ naturally arising from the left-hand side relational structure $\mathbf{A}$, namely, the hypergraph $H(\mathbf{A})$ with an edge $\{a_1, \ldots, a_k\}$ for each tuple $(a_1, \ldots, a_k)$ of $\mathbf{A}$. Let us briefly review some of the relevant results that have been obtained on structural tractability. The tractability of left-hand side relational structures having *bounded treewidth* was shown in the constraint satisfaction literature by Dechter and Pearl [9] and Freuder [10].[1] Later, Dalmau et al. [8] building on ideas of Kolaitis and Vardi [19, 20] gave a consistency-style algorithm for deciding the bounded treewidth CSP. For our present purposes, it is worth highlighting that although the notion of bounded treewidth is defined in terms of *tree decompositions*, which can be computed efficiently (under bounded treewidth), the algorithm given by Dalmau et al. [8] does *not* compute any form of tree decomposition. Dalmau et al. also identified a natural expansion of structures having bounded treewidth that is tractable–namely, the structures *homomorphically equivalent* to those having bounded treewidth. The optimality of this latter result, in the case of bounded arity, was demonstrated by Grohe [15], who proved–roughly speaking–that if the tuples of $\mathcal{A}$ are of bounded arity and $\mathcal{A}$ gives rise to a tractable case of the CSP, then it must fall into the natural expansion identified by Dalmau et al. [8].

A number of papers, including [17, 16, 13, 14, 11, 7], have studied restrictions that can be applied to relational structures of unbounded arity. (Note that any class of relational structures of unbounded arity cannot have bounded treewidth.) In a survey [13], Gottlob et al. show that the restriction of bounded hypertree width [11] is the most powerful structural restriction for the CSP in that every other structural restriction studied in the literature is subsumed by it. Since this work [11, 13], whether or not there is a more general structural restriction than bounded hypertree width that ensures tractability, has been a tantalizing open question.

In this paper, we study *generalized hypertree width*, a structural measure for hypergraphs defined in [12] that is a natural variation of hypertree width; we call this measure *coverwidth*. Coverwidth is *trivially* upper-bounded by hypertree width, and so any class of hypergraphs having bounded hypertree width has bounded coverwidth. We define a combinatorial pebble game that can be played on any CSP instance, and demonstrate that this game is intimately linked to coverwidth (Theorem 13). Overall, the investigation we perform takes significant inspiration from methods, concepts, and ideas developed by Kolaitis, Vardi,

[1] One way to define what we mean by treewidth here is the treewidth of the graph obtained from $H(\mathbf{A})$ by drawing an edge between any two vertices that are in the same hyperedge.

and coauthors [19, 20, 8, 2] that link together CSP consistency algorithms, the existential k-pebble games of Kolaitis and Vardi [18], and bounded treewidth.

Using the pebble game perspective, we are able to derive a number of computational results. One is that the structural restriction of *bounded coverwidth* implies polynomial-time tractability; this result generalizes the tractability of bounded hypertree width. It has been independently shown by Adler et al. that the hypertree width of a hypergraph is linearly related to the coverwidth [1]. This result can be used in conjunction with the tractability of bounded hypertree width to derive the tractability of bounded coverwidth. However, we believe our proof of bounded coverwidth tractability to be simpler than the known proof of bounded hypertree width tractability [11], even though our proof is of a more general result.

To describe our results in greater detail, it will be useful to identify two computational problems that every form of structural restriction gives rise to: a *promise* problem, and a *no-promise* problem. In both problems, the goal is to identify all CSP instances obeying the structural restriction as either satisfiable or unsatisfiable. In the promise problem, the input is a CSP instance that is *guaranteed* to obey the structural restriction, whereas in the no-promise problem, the input is an *arbitrary* CSP instance, and an algorithm may, on an instance not obeying the structural restriction, decline to identify the instance as satisfiable or unsatisfiable. Of course, CSPs arising in practice do *not* come with guarantees that they obey structural restrictions, and hence an algorithm solving the no-promise problem is clearly the more desirable. Notice that, for any structural restriction having a polynomial-time solvable promise problem, if it is possible to solve the *identification problem* of deciding whether or not an instance obeys the restriction, in polynomial time, then the no-promise problem is also polynomial-time solvable. For bounded hypertree width, both the identification problem and the no-promise problem are polynomial-time solvable. In fact, the survey by Gottlob et al. [13] only considers structural restrictions for which the identification problem is polynomial-time solvable, and thus only considers structural restrictions for which the no-promise problem is polynomial-time solvable.

One of our main theorems (Theorem 20) is that the promise problem for bounded coverwidth is polynomial-time tractable, via a general consistency-like algorithm. In particular, we show that, on an instance having bounded coverwidth, our algorithm detects an inconsistency if and only if the instance is unsatisfiable. Our algorithm, like the consistency algorithm of Dalmau et al. [8] for bounded treewidth, can be applied to *any* CSP instance to obtain a more constrained instance; our algorithm does *not* need nor compute any form of decomposition, even though the notion of coverwidth is defined in terms of decompositions!

We then give a simple algorithm for the no-promise problem for bounded coverwidth (Theorem 21) that employs the consistency-like algorithm for the promise problem. The algorithm's behavior is reminiscent of self-reducibility arguments in computational complexity theory, and on an instance of bounded coverwidth, the algorithm is guaranteed to either report a satisfying assignment or that the instance is unsatisfiable. We believe that this result suggests an expansion of the view of structural tractability advanced in the Gottlob et al. survey [13], since we are able to give a polynomial-time algorithm for the bounded coverwidth no-promise problem without explicitly showing that there is a polynomial-time algorithm for the bounded coverwidth identification problem.

Returning to the promise problem, we then show that the tractability of structures with bounded coverwidth can be generalized to yield the tractability of structures *homomorphically equivalent* to those having bounded coverwidth (Theorem 22). This expansion of bounded coverwidth tractability is analogous to the expansion of bounded treewidth tractability carried out in [8].

We emphasize that *none* of the algorithms in this paper need or compute any type of decomposition, even though all of the structural restrictions that they address are defined in terms of decompositions.

In the full version of this paper, we use the developed theory as well as ideas in [6] to define a tractable class of quantified constraint satisfaction problems based on coverwidth.

Definitions. In this paper, we formalize the CSP as a relational homomorphism problem. We review the relevant definitions that will be used. A *relational signature* is a finite set of relation symbols, each of which has an associated arity. A *relational structure* $\mathbf{A}$ (over signature σ) consists of a universe A and a relation $R^{\mathbf{A}}$ over A for each relation symbol R (of σ), such that the arity of $R^{\mathbf{A}}$ matches the arity associated to R. We refer to the elements of the universe of a relational structure $\mathbf{A}$ as $\mathbf{A}$-elements. When $\mathbf{A}$ is a relational structure over σ and R is any relation symbol of σ, the elements of $R^{\mathbf{A}}$ are called $\mathbf{A}$-tuples. Throughout this paper, we assume that all relational structures under discussion have a finite universe. We use boldface letters $\mathbf{A}, \mathbf{B}, \ldots$ to denote relational structures.

A *homomorphism* from a relational structure $\mathbf{A}$ to another relational structure $\mathbf{B}$ is a mapping h from the universe of $\mathbf{A}$ to the universe of $\mathbf{B}$ such that for every relation symbol R and every tuple $(a_1, \ldots, a_k) \in R^{\mathbf{A}}$, it holds that $(h(a_1), \ldots, h(a_k)) \in R^{\mathbf{B}}$. (Here, k denotes the arity of R.) The *constraint satisfaction problem (CSP)* is to decide, given an ordered pair $\mathbf{A}, \mathbf{B}$ of relational structures, whether or not there is a homomorphism from the first structure, $\mathbf{A}$, to the second, $\mathbf{B}$. A homomorphism from $\mathbf{A}$ to $\mathbf{B}$ in an instance $\mathbf{A}, \mathbf{B}$ of the CSP is also called a *satisfying assignment*, and when a satisfying assignment exists, we will say that the instance is *satisfiable*.

2 Coverwidth

This section defines the structural measure of hypergraph complexity that we call *coverwidth*. As we have mentioned, coverwidth is equal to generalized hypertree width, which was defined in [12]. We begin by defining the notion of *hypergraph*.

Definition 1. *A* hypergraph *is an ordered pair* (V, E) *consisting of a* vertex set V *and a* hyperedge set E*. The elements of* E *are called* hyperedges*; each hyperedge is a subset of* V*.*

Basic to the measure of coverwidth is the notion of a tree decomposition.

Definition 2. *A* tree decomposition *of a hypergraph* (V, E) *is a pair* $(T = (I, F), \{X_i\}_{i \in I})$ *where*

- $T = (I, F)$ *is a tree, and*
- *each* X_i *(with* $i \in I$*) is called a* bag *and is a subset of* V*,*

such that the following conditions hold:

1. $V = \cup_{i \in I} X_i$.
2. *For all hyperedges* $e \in E$*, there exists* $i \in I$ *with* $e \subseteq X_i$*.*
3. *For all* $v \in V$*, the vertices* $T_v = \{i \in I : v \in X_i\}$ *form a connected subtree of* T*.*

Tree decompositions are generally applied to graphs, and in the context of graphs, the measure of *treewidth* has been heavily studied. The *treewidth* of a graph G is the minimum of the quantity $\max_{i \in I} |X_i| - 1$ over all tree decompositions of G. In other words, a tree decomposition is measured based on its largest bag, and the treewidth is then defined based on the "lowest cost" tree decomposition.

The measure of coverwidth is also based on the notion of tree decomposition. In coverwidth, a tree decomposition is also measured based on its "largest" bag; however, the measure applied to a bag is the number of hyperedges needed to cover it, called here the *weight*.

Definition 3. *A* k*-union over a hypergraph* H *(with* $k \geq 0$*) is a union* $e_1 \cup \ldots \cup e_k$ *of* k *edges* $e_1, \ldots, e_k$ *of* H*.*

The empty set is considered to be the unique 0-union over a hypergraph.

Definition 4. *Let* $H = (V, E)$ *be a hypergraph. The* weight *of a subset* $X \subseteq V$ *is the smallest integer* $k \geq 0$ *such that* $X \cap (\cup_{e \in E} e)$ *is contained in a* k*-union over* H*.*

We measure a tree decomposition according to its heaviest bag, and define the *coverwidth* of a hypergraph according to the lightest-weight tree decomposition.

Definition 5. *The* weight *of a tree decomposition of* H *is the maximum weight over all of its bags.*

Definition 6. *The* coverwidth *of a hypergraph* H *is the minimum weight over all tree decompositions of* H*.*

It is straightforward to verify that the coverwidth of a hypergraph is equal to the generalized hypertree width of a hypergraph [12]. Since the generalized hypertree width of a hypergraph is always less than or equal to its hypertree width, coverwidth is at least as strong as hypertree width in that results on bounded coverwidth imply results on bounded hypertree width.

There is another formulation of tree decompositions that is often wieldy, see for instance [3].

Definition 7. *A* scheme *of a hypergraph $H = (V, E)$ is a graph (V, F) such that*

- *(V, F) has a perfect elimination ordering, that is, an ordering $v_1, \ldots, v_n$ of its vertices such that for all $i < j < k$, if (v_i, v_k) and (v_j, v_k) are edges in F, then (v_i, v_j) is also an edge in F, and*
- *the vertices of every hyperedge of E induce a clique in (V, F).*

It is well known that the property of having a perfect elimination ordering is equivalent to being chordal. The following proposition is also well-known.

Proposition 8. *Let H be a hypergraph. For every tree decomposition of H, there exists a scheme such that each clique of the scheme is contained in a bag of the tree decomposition. Likewise, for every scheme of H, there exists a tree decomposition such that each bag of the tree decomposition is contained in a clique of the scheme.*

Let us define the *weight* of a scheme (of a hypergraph H) to be the maximum weight (with respect to H) over all of its cliques. The following proposition is immediate from Proposition 8 and the definition of coverwidth, and can be taken as an alternative definition of coverwidth.

Proposition 9. *The coverwidth of a hypergraph H is equal to the minimum weight over all schemes of H.*

We now define the hypergraph associated to a relational structure. Roughly speaking, this hypergraph is obtained by "forgetting" the ordering of the $\mathbf{A}$-tuples.

Definition 10. *Let $\mathbf{A}$ be a relational structure. The hypergraph associated to $\mathbf{A}$ is denoted by $H(\mathbf{A})$; the vertex set of $H(\mathbf{A})$ is the universe of $\mathbf{A}$, and for each $\mathbf{A}$-tuple $(a_1, \ldots, a_k)$, there is an edge $\{a_1, \ldots, a_k\}$ in $H(\mathbf{A})$.*

We will often implicitly pass from a relational structure to its associated hypergraph, that is, we simply write $\mathbf{A}$ in place of $H(\mathbf{A})$. In particular, we will speak of k-unions over a relational structure $\mathbf{A}$.

3 Existential k-Cover Games

We now define a class of pebble games for studying the measure of coverwidth. These games are parameterized by an integer $k \geq 1$, and are called existential

k-cover games. They are based on the existential k-pebble games defined by Kolaitis and Vardi and used to study constraint satisfaction [18, 20]. The pebble game that we use is defined as follows. The game is played between two players, the *Spoiler* and the *Duplicator*, on a pair of relational structures $\mathbf{A}, \mathbf{B}$ that are defined over the same signature. Game play proceeds in rounds, and in each round one of the following occurs:

1. The Spoiler places a pebble on an $\mathbf{A}$-element a. In this case, the Duplicator must respond by placing a corresponding pebble, denoted by $h(a)$, on a $\mathbf{B}$-element.
2. The Spoiler removes a pebble from an $\mathbf{A}$-element a. In this case, the corresponding pebble $h(a)$ on $\mathbf{B}$ is removed.

When game play begins, there are no pebbles on any $\mathbf{A}$-elements, nor on any $\mathbf{B}$-elements, and so the first round is of the first type. Both of the players have an unlimited supply of pebbles. However, when placing a new pebble, the Spoiler must obey the restriction that the weight of the elements on which the Spoiler has pebbles must be bounded by k. (Here, by "weight" we are using Definition 4.) We assume that the Spoiler never places two pebbles on the same $\mathbf{A}$-element, so that h is a partial function (as opposed to a relation). The Duplicator wins the game if he can always ensure that h is a *projective homomorphism* from $\mathbf{A}$ to $\mathbf{B}$; otherwise, the Spoiler wins. A *projective homomorphism* (from $\mathbf{A}$ to $\mathbf{B}$) is a partial function h from the universe of $\mathbf{A}$ to the universe of $\mathbf{B}$ such that for any relation symbol R and any tuple $(a_1, \ldots, a_k) \in R^{\mathbf{A}}$ of $\mathbf{A}$, there exists a tuple $(b_1, \ldots, b^k) \in R^{\mathbf{B}}$ where $h(a_i) = b_i$ for all a_i on which h is defined.

We now formalize the notion of a *winning strategy* for the Duplicator in the existential k-cover game. Note that when h is a partial function, we use $\mathsf{dom}(h)$ to denote the domain of h.

Definition 11. *A* winning strategy *for the Duplicator in the existential k-cover game on relational structures $\mathbf{A}, \mathbf{B}$ is a non-empty set H of projective homomorphisms (from $\mathbf{A}$ to $\mathbf{B}$) having the following two properties.*

1. *(the "forth" property) For every $h \in H$ and $\mathbf{A}$-element $a \notin \mathsf{dom}(h)$, if $\mathsf{dom}(h) \cup \{a\}$ has weight $\leq k$, then there exists a projective homomorphism $h' \in H$ extending h with $\mathsf{dom}(h') = \mathsf{dom}(h) \cup \{a\}$.*
2. *The set H is closed under subfunctions, that is, if $h \in H$ and h extends h', then $h' \in H$.*

As we mentioned, the definition of this game is based on the *existential k-pebble game* introduced by Kolaitis and Vardi [18, 20]. In the *existential k-pebble game*, the number of pebbles that the Spoiler may use is bounded by k, and the Duplicator need only must ensure that h is a *partial homomorphism*. A close relationship between this game and bounded treewidth has been identified [2].

Theorem 12. *[2] Let $\mathbf{A}$ and $\mathbf{B}$ be relational structures. For all $k \geq 2$, the following are equivalent.*

– *There is a winning strategy for the Duplicator in the existential k-pebble game on $\mathbf{A}, \mathbf{B}$.*
– *For all relational structures $\mathbf{T}$ of treewidth $< k$, if there is a homomorphism from $\mathbf{T}$ to $\mathbf{A}$, then there is a homomorphism from $\mathbf{T}$ to $\mathbf{B}$.*

We have the following analog of Theorem 12.

Theorem 13. *Let $\mathbf{A}$ and $\mathbf{B}$ be relational structures. For all $k \geq 1$, the following are equivalent.*

– *There is a winning strategy for the Duplicator in the k-cover game on $\mathbf{A}, \mathbf{B}$.*
– *For all relational structures $\mathbf{T}$ of coverwidth $\leq k$, if there is a homomorphism from $\mathbf{T}$ to $\mathbf{A}$, then there is a homomorphism from $\mathbf{T}$ to $\mathbf{B}$.*

Proof. ($\Rightarrow$) Let H be a winning strategy for the Duplicator in the k-cover game on $\mathbf{A}$ and $\mathbf{B}$, let $\mathbf{T}$ be any structure of coverwidth $\leq k$, let f be any homomorphism from $\mathbf{T}$ to $\mathbf{A}$, let $G = (T, F)$ be a scheme for $\mathbf{T}$ of weight $\leq k$, and let $v_1, \ldots, v_n$ be a perfect elimination ordering of G.

We shall construct a sequence of partial mappings $g_0, \ldots, g_n$ from T to B such that for each i:

1. $\mathsf{dom}(g_i) = \{v_1, \ldots, v_i\}$, and
2. for every clique $L \subseteq \{v_1, \ldots, v_i\}$ in G, there exists a projective homomorphism $h \in H$ with domain $f(L)$ in the winning strategy of the Duplicator, such that for every $v \in L$, $h(f(v)) = g_i(v)$.

We define g_0 to be the partial function with empty domain. For every $i \geq 0$, the partial mapping g_{i+1} is obtained by extending g_i in the following way. As $v_1, \ldots, v_n$ is a perfect elimination ordering, the set

$$L = \{v_{i+1}\} \cup \{v_j : j < i+1, (v_j, v_{i+1}) \in F\}$$

is a clique of G. Define L' as $L \setminus \{v_{i+1}\}$. By the induction hypothesis, there exists $h \in H$ such that for every $v \in L'$, $h(f(v)) = g_i(v)$. Let us consider two cases.

If $f(v_{i+1}) = f(v_j)$ for some $v_j \in L'$ then we set $g_{i+1}(v_{i+1})$ to be $g_i(v_j)$. Note that in this case property (2) is satisfied, as every clique in G containing v_{i+1} is contained in L and h serves as a certificate. (For any clique not containing v_{i+1}, we use the induction hypothesis.)

Otherwise, that is, if $f(v_{i+1}) \neq f(v_j)$ for all $v_j \in L'$, we do the following. First, since the weight of L is bounded above by k and f defines an homomorphism from $\mathbf{T}$ to $\mathbf{A}$ then the weight of $f(L)$ is also bounded by k. Observe that $f(L) = \mathsf{dom}(h) \cup \{f(v_{i+1})\}$. By the forth property of winning strategy there exists an extension $h' \in H$ of h that is defined over v_{i+1}. We set $g_{i+1}(v_{i+1})$ to be $h'(f(v_{i+1}))$. Note that h' certifies that property (2) is satisfied for very clique containing v_{i+1}; again, any clique not containing v_{i+1} is covered by the induction hypothesis.

Finally, let us prove that g_n indeed defines an homomorphism from $\mathbf{T}$ to $\mathbf{B}$. Let R be any relation symbol and let $(t_1, \ldots, t_l)$ be any relation in $R^{\mathbf{T}}$. We want to show that $(g_n(t_1),\ldots,g_n(t_l))$ belongs to $R^{\mathbf{B}}$. Since G is an scheme for $\mathbf{T}$, $\{t_1,\ldots,t_l\}$ constitutes a clique of G. By property (2) there exists $h\in H$ such that $h(f(t_i))=g(t_i)$ for all i. Observing that as f is an homomorphism from $\mathbf{T}$ to $\mathbf{A}$, we can have that $(f(t_1),\ldots,f(t_l))$ belongs to $R^{\mathbf{A}}$. Finally, as h is a projective homomorphism from $\mathbf{A}$ to $\mathbf{B}$, the tuple $(h(f(t_1)),\ldots,h(f(t_l)))$ must be in $\mathbf{B}$.

($\Leftarrow$) We shall construct a winning strategy H for the Duplicator. We need a few definitons. Fix a sequence $a_1,\ldots,a_m$ of elements of A. A *valid tuple* for $a_1,\ldots,a_m$ is any tuple $(\mathbf{T},G,v_1,\ldots,v_m,f)$ where $\mathbf{T}$ is a relational structure, G is an scheme of weight k for $\mathbf{T}$, $\{v_1,\ldots,v_m\}$ is a clique of G, and f is an homomorphism from $\mathbf{T},v_1,\ldots,v_m$ to $\mathbf{A},a_1,\ldots,a_m$. (By a homomorphism from $\mathbf{T},v_1,\ldots,v_m$ to $\mathbf{A},a_1,\ldots,a_m$, we mean a homomorphism from $\mathbf{T}$ to $\mathbf{A}$ that maps v_i to a_i for all i.) By $S(\mathbf{T},G,v_1,\ldots,v_m,f)$ we denote the set of all mappings h with domain $\{a_1,\ldots,a_m\}$ such that there is an homomorphism from $\mathbf{T},v_1,\ldots,v_m$ to $\mathbf{B},h(a_1),\ldots,h(a_m)$. We are now in a situation to define H. H contains for every subset $a_1,\ldots,a_m$ of weight at most k, every partial mapping h that is contained in all $S(\mathbf{T},G,v_1,\ldots,v_m,f)$ where $(\mathbf{T},G,v_1,\ldots,v_m,f)$ is a valid tuple for $a_1,\ldots,a_m$.

Let us show that H is indeed a winning strategy. First, observe that H is nonempty, as it contains the partial function with empty domain. Second, let us show that H contains only projective homomorphisms. Indeed, let h be any mapping in H with domain $a_1,\ldots,a_m$, let R be any relation symbol and let $(c_1,\ldots,c_l)$ be any tuple in $R^{\mathbf{A}}$. Let us define $\mathbf{T}$ to be the substructure (not necessarily induced) of $\mathbf{A}$ with universe $\{a_1,\ldots,a_k,c_1,\ldots,c_l\}$ containing only the tuple $(c_1,\ldots,c_l)$ in $R^{\mathbf{T}}$. It is easy to verify that the graph $G=(\{a_1,\ldots,a_k,c_1,\ldots,c_l\},F)$ where $F=\{(a_i,a_j):i\neq j\}\cup\{(c_i,c_j):i\neq j\}$ is an scheme of $\mathbf{T}$ of weight $\leq k$. Consequently, $(\mathbf{T},G,a_1,\ldots,a_m,id)$ is a valid tuple for $a_1,\ldots,a_m$ and therefore there exists an homomorphism g from $\mathbf{T}$ to $\mathbf{B}$, and hence satisfying $(g(c_1),\ldots,g(c_l))\in R^{\mathbf{B}}$, such that $g(a_i)=h(a_i)$ for all $i=1,\ldots k$.

To show that H is closed under subfunctions is rather easy. Indeed, let h' be any mapping in H with domain $a_1\ldots,a_m$. We shall see that the restriction h of h' to $\{a_1,\ldots,a_{m-1}\}$ is also in H. Let $(\mathbf{T},G,v_1,\ldots,v_{m-1},f)$ be any valid tuple for $a_1,\ldots,a_{k-1}$. We construct a valid tuple $(\mathbf{T}',G',v_1,\ldots,v_m,f')$ for $a_1,\ldots,a_m$ in the following way: v_m is a new (not in the universe of $\mathbf{T}$) element, $\mathbf{T}'$ is the structure obtained from $\mathbf{T}$ by adding v_m to the universe of $\mathbf{T}$ and keeping the same relations, f' is the extension of f in which v_m is map to a_m, and G' is the scheme of $\mathbf{T}$ obtained by adding to G an edge (v_j,v_m) for every $j=1,\ldots,m-1$. Since $(\mathbf{T}',G',v_1,\ldots,v_m,f')$ is a valid tuple for $a_1,\ldots,a_m$ and $h'\in H$, there exists an homomorphism g' from $\mathbf{T}',v_1,\ldots,v_m$ to $\mathbf{B},h'(a_1),\ldots,h'(a_m)$. Observe then that the restriction g of g' to $\{a_1,\ldots,a_{m-1}\}$ defines then an homomorphism from $\mathbf{T},v_1,\ldots,v_{m-1}$ to $\mathbf{B},h(a_1),\ldots,h(m_1)$.

Finally, we shall show that H has the forth property. The proof relies in the following easy properties of the valid tuples. Let $a_1,\ldots,a_m$ be elements of A and let $(\mathbf{T}_1,G_1,v_1,\ldots,v_m,f_1)$ and let $(\mathbf{T}_2,G_2,v_1,\ldots,v_m,f_2)$ be valid tuples for $a_1, \ldots, a_m$ such that $T_1 \cap T_2 = \{v_1, \ldots, v_m\}$, let $\mathbf{T}$ be $\mathbf{T}_1 \cup \mathbf{T}_2$ (that is, the structure $\mathbf{T}$ whose universe is the union of the universes of $\mathbf{T}_1$ and $\mathbf{T}_2$, and in which $R^{\mathbf{T}} = R^{\mathbf{T}_1} \cup R^{\mathbf{T}_2}$ for all relation symbols R), $G = G_1 \cup G_2$ and let f be the mapping from the universe T of $\mathbf{T}$ to B that sets a to $f_1(a)$ if $a \in T_1$ and to $f_2(a)$ if $a \in T_2$ (observe that f_1 and f_2 coincide over $\{v_1, \ldots, v_m\}$). Then $(\mathbf{T}, G, v_1, \ldots, v_m, f)$ is a valid tuple for $a_1, \ldots, a_m$. We call $(\mathbf{T}, G, v_1, \ldots, v_m, f)$, the *union* of $(\mathbf{T}_1, G_1, v_1, \ldots, v_m, f_1)$ and $(\mathbf{T}_2, G_2, v_1, \ldots, v_m, f_2)$. Furthermore, $S(\mathbf{T}, G, v_1, \ldots, v_m, f) \subseteq S(\mathbf{T}_1, G_1, v_1, \ldots, v_m, f_1) \cap S(\mathbf{T}_2, G_2, v_1, \ldots, v_m, f_2)$ (in fact, this is an equality, although we do not need the equality in our proof).

Let h be any mapping in H, let $\{a_1, \ldots, a_{m-1}\}$ be its domain, and let a_m be any element in the universe of $\mathbf{A}$ such that $\{a_1, \ldots, a_m\}$ has weight $\leq k$. Let us assume, towards a contradiction, that there is not extension h' of h in $\mathcal{H}$. Then there exists a *finite* collection $\{(\mathbf{T}_i, G_i, v_1, \ldots, v_m, f_i) : i \in I\}$ of valid tuples for $a_1, \ldots, a_m$ such that the intersection $\bigcap_{i \in I} S(\mathbf{T}_i, G_i, v_1, \ldots, v_m, f_i)$ does not contain any extension of h. We can rename the elements of the universes so that for every different $i, j \in I$ we have that $T_i \cap T_j = \{v_1, \ldots, v_m\}$.

Let $(\mathbf{T}, G, v_1, \ldots, v_m, f)$ be the union of $(\mathbf{T}_i, G_i, v_1, \ldots, v_m, f_i)$, $i \in I$, which is a valid tuple for $a_1, \ldots, a_m$. Since

$$S(\mathbf{T}, G, v_1, \ldots, v_m, f) \subseteq \bigcap_{i \in I} S(\mathbf{T}_i, G_i, v_1, \ldots, v_m, f_i)$$

we can conclude that $S(\mathbf{T}, G, v_1, \ldots, v_m, f)$ does not contain any extension of h. To finish the proof, it is only necessary to observe that $(\mathbf{T}, G, v_1, \ldots, v_{m-1}, f)$ is a valid tuple for $a_1, \ldots, a_{m-1}$ and since $S(\mathbf{T}, G, v_1, \ldots, v_m, f)$ does not contain any extension of h, $S(\mathbf{T}, G, v_1, \ldots, v_{m-1}, f)$ cannot contain h, in contradiction with $h \in H$. □

Theorem 13 can be easily applied to show that in an instance $\mathbf{A}, \mathbf{B}$ of the CSP, if the left-hand side structure has coverwidth bounded by k, then deciding if there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ is equivalent to deciding the existence of a Duplicator winning strategy in the existential k-cover game.

Theorem 14. *Let $\mathbf{A}$ be a relational structure having coverwidth $\leq k$, and let $\mathbf{B}$ be an arbitrary relational structure. There is a winning strategy for the Duplicator in the k-cover game on $\mathbf{A}, \mathbf{B}$ if and only if there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$.*

We will use this theorem in the next section to develop tractability results. Although we use Theorem 13 to derive this theorem, we would like to emphasize that the full power of Theorem 13 is not needed to derive it, as pointed out in the proof.

Proof. If there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$, the Duplicator can win by always setting pebbles according the homomorphism. The other direction is immediate from Theorem 13 (note that we only need the forward implication and $\mathbf{T} = \mathbf{A}$). □

4 The Algorithmic Viewpoint

The previous section introduced the *existential k-cover game*. We showed that deciding a CSP instance of bounded coverwidth is equivalent to deciding if the Duplicator has a winning strategy in the existential k-cover game. In this section, we show that the latter property–the existence of a Duplicator winning strategy–can be decided algorithmically in polynomial time. To this end, it will be helpful to introduce the notion of a *compact winning strategy.*

Definition 15. *A* compact winning strategy *for the Duplicator in the existential k-cover game on relational structures $\mathbf{A}, \mathbf{B}$ is a non-empty set H of projective homomorphisms (from $\mathbf{A}$ to $\mathbf{B}$) having the following properties.*

1. *For all $h \in H$, $\mathsf{dom}(h)$ is a k-union (over $\mathbf{A}$).*
2. *For every $h \in H$ and for every k-union U (over $\mathbf{A}$), there exists $h' \in H$ with $\mathsf{dom}(h') = U$ such that for every $v \in \mathsf{dom}(h) \cap \mathsf{dom}(h')$, $h(v) = h'(v)$.*

Proposition 16. *In the existential k-cover game on a pair of relational structures $\mathbf{A}, \mathbf{B}$, the Duplicator has a winning strategy if and only if the Duplicator has a compact winning strategy.*

Proof. Suppose that the Duplicator has a winning strategy H. Let C be the set containing all functions $h \in H$ such that $\mathsf{dom}(h)$ is a k-union. We claim that C is a compact winning strategy. Clearly C satisfies the first property of a compact winning strategy, so we show that it satisfies the second property. Suppose $h \in C$ and let U be a k-union. By the subfunction property of a winning strategy, the restriction r of h to $\mathsf{dom}(h) \cap U$ is in H. By repeated application of the forth property, there is an extension e of r that is in H and has domain U, which serves as the desired h'.

Now suppose that the Duplicator has a compact winning strategy C. Let H be the closure of C under subfunctions. We claim that H is a winning strategy. It suffices to show that H has the forth property. Let $h \in H$ and suppose that a is an $\mathbf{A}$-element where $\mathsf{dom}(h) \cup \{a\}$ has weight $\leq k$. Let U be a k-union such that $\mathsf{dom}(h) \cup \{a\} \subseteq U$. By definition of H, there is a function $e \in C$ extending h. Apply the second property of a compact winning strategy to e and U to obtain an $e' \in C$ with domain U such that for every $v \in \mathsf{dom}(e) \cap \mathsf{dom}(e')$, $e(v) = e'(v)$. Notice that $\mathsf{dom}(h) \subseteq \mathsf{dom}(e) \cap \mathsf{dom}(e')$. Thus, the restriction of e' to $\mathsf{dom}(h) \cup \{a\}$ is in H and extends h. $\square$

We have just shown that deciding if there is a winning strategy, in an instance of the existential k-cover game, is equivalent to deciding if there is a compact winning strategy. We now use this equivalence to give a polynomial-time algorithm for deciding if there is a winning strategy.

Theorem 17. *For all $k \geq 1$, there exists a polynomial-time algorithm that, given a pair of relational structures $\mathbf{A}, \mathbf{B}$, decides whether or not there is a winning strategy for the Duplicator in the existential k-cover game on $\mathbf{A}, \mathbf{B}$.*

Proof. By Proposition 16, it suffices to give a polynomial-time algorithm that decides if there is a compact winning strategy. It is straightforward to develop such an algorithm based on the definition of compact winning strategy. Let H be the set of all functions h such that $\mathsf{dom}(h)$ is a k-union (over $\mathbf{A}$) and such that h is a projective homomorphism from $\mathbf{A}$ to $\mathbf{B}$. Iteratively perform the following until no changes can be made to H: for every function $h \in H$ and every k-union U, check to see if there is $h' \in H$ such that the second property (of compact winning strategy) is satisfied; if not, remove h from H. Throughout the algorithm, we have maintained the invariant that any compact winning strategy must be a subset of H. Hence, if when the algorithm terminates H is empty, then there is no compact winning strategy. And if H is non-empty when the algorithm terminates, H is clearly a compact winning strategy.

The number of k-unions (over $\mathbf{A}$) is polynomial in the number of tuples in $\mathbf{A}$. Also, for each k-union U, the number of projective homomorphisms h with $\mathsf{dom}(h) = U$ from $\mathbf{A}$ to $\mathbf{B}$ is polynomial in the number of tuples in $\mathbf{B}$. Hence, the size of the original set H is polynomial in the original instance. Since in each iteration an element is removed from H, the algorithm terminates in polynomial time. $\square$

The algorithm we have just described in the proof of Theorem 17 may appear to be quite specialized. However, we now show that essentially that algorithm can be viewed as a *general* inference procedure for CSP instances in the vein of existing consistency algorithms. In particular, we give a general algorithm called *projective k-consistency* for CSP instances that, given a CSP instance, performs inference and outputs a more constrained CSP instance having exactly the same satisfying assignments as the original. On a CSP instance $\mathbf{A}, \mathbf{B}$, the algorithm might detect an *inconsistency*, by which we mean that it detects that there is no homomorphism from $\mathbf{A}$ to $\mathbf{B}$. If it does not, then it is guaranteed that there is a winning strategy for the Duplicator.

Definition 18. *The* projective k-consistency algorithm *takes as input a CSP instance $\mathbf{A}, \mathbf{B}$, and consists of the following steps.*

- *Create a new CSP instance $\mathbf{A}', \mathbf{B}'$ as follows. Let the universe of $\mathbf{A}'$ be the universe of $\mathbf{A}$, and the universe of $\mathbf{B}'$ be the universe of $\mathbf{B}$. Let the signature of $\mathbf{A}'$ and $\mathbf{B}'$ contain a relation symbol R_U for each k-union U over $\mathbf{A}$. For each k-union U, the relation $R_U^{\mathbf{A}'}$ is defined as $(u_1, \ldots, u_m)$, where $u_1, \ldots, u_m$ are exactly the elements of U in some order; and $R_U^{\mathbf{B}'}$ is defined as the set of all tuples $(b_1, \ldots, b_m)$ such that the mapping taking $u_i \to b_i$ is a projective homomorphism from $\mathbf{A}$ to $\mathbf{B}$.*
- *Iteratively perform the following until no changes can be made: remove any $\mathbf{B}'$-tuple $(b_1, \ldots, b_m)$ that is not a projective homomorphism.*
  We say that a $\mathbf{B}'$-tuple $(b_1, \ldots, b_m) \in R_U^{\mathbf{B}'}$ is a projective homomorphism if, letting $(u_1, \ldots, u_m)$ denote the unique element of $R_U^{\mathbf{A}'}$, the function taking $u_i \to b_i$ is a projective homomorphism from $\mathbf{A}'$ to $\mathbf{B}'$.
- *Report an inconsistency if there are no $\mathbf{B}'$-tuples remaining.*

Theorem 19. *For each $k \geq 1$, the projective k-consistency algorithm, given as input a CSP instance $\mathbf{A}, \mathbf{B}$:*

- *runs in polynomial time,*
- *outputs a CSP instance $\mathbf{A}', \mathbf{B}'$ that has the same satisfying assignments as $\mathbf{A}, \mathbf{B}$, and*
- *reports an inconsistency if and only if the Duplicator does not have a winning strategy in the existential k-cover game on $\mathbf{A}, \mathbf{B}$.*

Proof. The first property is straightforward to verify. For the second property, observe that, each time a tuple is removed from $\mathbf{B}'$, the set of satisfying assignments is preserved. For the third property, observe that, associating $\mathbf{B}'$-tuples to functions as in Definition 18, the behavior of the projective k-consistency algorithm is identical to the behavior of the algorithm in the proof of Proposition 16. □

By using the results presented in this section thus far, it is easy to show that CSP instances of bounded coverwidth are tractable. Define the coverwidth of a CSP instance $\mathbf{A}, \mathbf{B}$ to be the coverwidth of $\mathbf{A}$. Let $\mathsf{CSP}[\mathsf{coverwidth} \leq k]$ be the restriction of the CSP to all instances of coverwidth less than or equal to k.

Theorem 20. *For all $k \geq 1$, the problem $\mathsf{CSP}[\mathsf{coverwidth} \leq k]$ is decidable in polynomial time by the projective k-consistency algorithm. In particular, on an instance of $\mathsf{CSP}[\mathsf{coverwidth} \leq k]$, the projective k-consistency algorithm reports an inconsistency if and only if the instance is not satisfiable.*

Proof. Immediate from Theorem 14 and the third property of Theorem 19. □

Note that we can derive the tractability of CSP instances having bounded hypertree width immediately from Theorem 20.

Now, given a CSP instance that is promised to have bounded coverwidth, we can use projective k-consistency to decide the instance (Theorem 20). This tractability result can in fact be pushed further: we can show that there is a generic polynomial-time that, given an *arbitrary* CSP instance, is *guaranteed* to decide instances of bounded coverwidth. Moreover, whenever an instance is decided to be a "yes" instance by the algorithm, a satisfying assignment is constructed.

Theorem 21. *For all $k \geq 1$, there exists a polynomial-time algorithm that, given any CSP instance $\mathbf{A}, \mathbf{B}$,*

1. *outputs a satisfying assignment for $\mathbf{A}, \mathbf{B}$,*
2. *correctly reports that $\mathbf{A}, \mathbf{B}$ is unsatisfiable, or*
3. *reports "I don't know".*

The algorithm always performs (1) or (2) on an instance of $\mathsf{CSP}[\mathsf{coverwidth} \leq k]$.

Proof. The algorithm is a simple extension of the projective k-consistency algorithm. First, the algorithm applies the projective k-consistency algorithm; if an inconsistency is detected, then the algorithm terminates and reports that $\mathbf{A}, \mathbf{B}$ is unsatisfiable. Otherwise, it initializes V to be the universe A of $\mathbf{A}$, and does the following:

- If V is empty, terminate and identify the mapping taking each $a \in A$ to the $\mathbf{B}$-element in $R_a^{\mathbf{B}}$, as a satisfying assignment.
- Pick any variable $v \in V$.
- Expand the signature of $\mathbf{A}, \mathbf{B}$ to include another symbol R_v with $R_v^{\mathbf{A}} = \{(v)\}$.
- Try to find a $\mathbf{B}$-element b such that when $R_v^{\mathbf{B}}$ is set to $\{(b)\}$, no inconsistency is detected by the projective k-consistency algorithm on the expanded instance.
  - If there is no such $\mathbf{B}$-element, terminate and report "I don't know".
  - Otherwise, set $R_v^{\mathbf{B}}$ to such a $\mathbf{B}$-element, remove v from V, and repeat from the first step using the expanded instance.

If the procedure terminates from V being empty in the first step, the mapping that is output is straightforwardly verified to be a satisfying assignment.

Suppose that the algorithm is given an instance of $\mathsf{CSP}[\mathsf{coverwidth} \leq k]$. If it is unsatisfiable, then the algorithm reports that the instance is unsatisfiable by Theorem 20. So suppose that the instance is satisfiable. We claim that each iteration preserves the satisfiability of the instance. Let $\mathbf{A}, \mathbf{B}$ denote the CSP instance at the beginning of an arbitrary iteration of the algorithm. If no inconsistency is detected after adding a new relation symbol R_v with $R_v^{\mathbf{A}} = \{(v)\}$ and $R_v^{\mathbf{B}} = \{(b)\}$, there must be a satisfying assignment mapping v to b by Theorem 20. Note that adding unary relation symbols to a CSP instance does not change the coverwidth of the instance. $\square$

We now expand the tractability result of Theorem 20, and show the tractability of CSP instances that are *homomorphically equivalent* to instances of bounded coverwidth. Formally, let us say that $\mathbf{A}$ and $\mathbf{A}'$ are homomorphically equivalent if there is a homomorphism from $\mathbf{A}$ to $\mathbf{A}'$ as well as a homomorphism from $\mathbf{A}'$ to $\mathbf{A}$. Let $\mathsf{CSP}[\mathcal{H}(\mathsf{coverwidth} \leq k)]$ denote the restriction of the CSP to instances $\mathbf{A}, \mathbf{B}$ where $\mathbf{A}$ is homomorphically equivalent to a relational structure of coverwidth less than or equal to k.

Theorem 22. *For all $k \geq 1$, the problem $\mathsf{CSP}[\mathcal{H}(\mathsf{coverwidth} \leq k)]$ is decidable in polynomial time by the projective k-consistency algorithm. In particular, on an instance of $\mathsf{CSP}[\mathcal{H}(\mathsf{coverwidth} \leq k)]$, the projective k-consistency algorithm reports an inconsistency if and only if the instance is not satisfiable.*

References

1. Isolde Adler, Georg Gottlob, and Martin Grohe. Hypertree-width and related hypergraph invariants. In preparation.

2. A. Atserias, Ph. G. Kolaitis, and M. Y. Vardi. Constraint propagation as a proof system. In *CP 2004*, 2004.
3. Hans L. Bodlaender. Discovering treewidth. In *SOFSEM 2005*, 2005.
4. Andrei Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proceedings of 43rd IEEE Symposium on Foundations of Computer Science*, pages 649–658, 2002.
5. Andrei Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of 18th IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 321–330, 2003. Extended version appears as Oxford University technical report PRG-RR–03-01.
6. Hubie Chen and Victor Dalmau. From pebble games to tractability: An ambidextrous consistency algorithm for quantified constraint satisfaction. Manuscript, 2005.
7. D. Cohen, P. Jeavons, and M. Gyssens. A unified theory of structural tractability for constraint satisfaction and spread cut decomposition. To appear in IJCAI 2005, 2005.
8. Victor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Constraint Programming '02*, LNCS, 2002.
9. Rina Dechter and Judea Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, pages 353–366, 1989.
10. Eugene Freuder. Complexity of k-tree structured constraint satisfaction problems. In *AAAI-90*, 1990.
11. G. Gottlob, L. Leone, and F. Scarcello. Hypertree decomposition and tractable queries. *Journal of Computer and System Sciences*, 64(3):579–627, 2002.
12. G. Gottlob, L. Leone, and F. Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *Journal of Computer and System Sciences*, 66:775–808, 2003.
13. Georg Gottlob, Nicola Leone, and Francesco Scarcello. A comparison of structural csp decomposition methods. *Artif. Intell.*, 124(2):243–282, 2000.
14. Georg Gottlob, Nicola Leone, and Francesco Scarcello. The complexity of acyclic conjunctive queries. *Journal of the ACM*, 43(3):431–498, 2001.
15. Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. In *FOCS 2003*, pages 552–561, 2003.
16. M. Gyssens, P.G. Jeavons, and D.A. Cohen. Decomposing constraint satisfaction problems using database techniques. *Artificial Intelligence*, 66(1):57–89, 1994.
17. M. Gysssens and J. Paradaens. A decomposition methodology for cyclic databases. In *Advances in Database Theory*, volume 2, pages 85–122. Plenum Press, New York, NY, 1984.
18. Ph.G. Kolaitis and M.Y. Vardi. On the expressive power of Datalog: tools and a case study. *Journal of Computer and System Sciences*, 51(1):110–134, 1995.
19. Ph.G. Kolaitis and M.Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61:302–332, 2000.
20. Ph.G. Kolaitis and M.Y. Vardi. A game-theoretic approach to constraint satisfaction. In *Proceedings 17th National (US) Conference on Artificial Intellignece, AAAI'00*, pages 175–181, 2000.
21. Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.