A Unified Theory of Structural Tractability for Constraint Satisfaction and Spread Cut Decomposition

David Cohen
Dept of Computer Science
Royal Holloway
University of London
Egham, Surrey, UK
D.Cohen@rhul.ac.uk

Peter Jeavons
Computing Laboratory
Oxford University
Wolfson Building, Parks Road
Oxford, UK
Peter.Jeavons@comlab.ox.ac.uk

Marc Gyssens
Department WNI
Limburgs Universitair Centrum
Universitaire Campus
B-3590 Diepenbeek, Belgium
marc.gyssens@luc.ac.be

Abstract

In this paper we introduce a generic form of structural decomposition for the constraint satisfaction problem, which we call a guarded decomposition. We show that many existing decomposition methods can be characterized in terms of finding guarded decompositions satisfying certain specified additional conditions.

Using the guarded decomposition framework we are also able to define a new form of decomposition, which we call a spread cut. We show that discovery of width k spread-cut decompositions is tractable for each k, and that the spread cut decomposition strongly generalize all existing decompositions except hypertrees. Finally we exhibit a family of hypergraphs H_n, for $n = 1, 2, 3 \ldots$, where the width of the best hypertree decomposition of each H_n is at least $3n$, but the width of the best spread-cut decomposition is at most $2n$.

1 Introduction

A constraint satisfaction problem consists of a set of variables that must be assigned values subject to certain constraints. These constraints restrict the simultaneous assignments to certain specified subsets of the variables. Many real-world problems can be represented very naturally in this framework [Dechter, 2003; Marriott and Stuckey, 1998]

Similar problems arise in the field of relational databases, where they are referred to as conjunctive query evaluation problems [Kolaitis and Vardi, 2000]. Many of the techniques developed in this paper can also be applied directly to the conjunctive query evaluation problem, but we shall not pursue this application here.

The decision problem for the general constraint satisfaction problem is NP-hard [Mackworth, 1977]. This motivates the search for more restricted subproblems which are *tractable*, that is, solvable in polynomial-time.

This paper considers subproblems of the constraint satisfaction problem which can be shown to be tractable using so-called *structural* methods or *decomposition* methods. These methods rely exclusively on using the structural properties of instances, in other words the way in which the constraints overlap each other.

A variety of such decomposition methods have been developed and applied in both the constraint satisfaction community, and the database community. Examples include methods based on the use of treewidth [Freuder, 1985], cycle cutsets [Dechter and Pearl, 1988], tree-clustering [Dechter and Pearl, 1989], hinges [Gyssens *et al.*, 1994], cycle hypercutsets and hypertrees [Gottlob *et al.*, 2000]. All of these methods rely on reducing a given problem instance to an equivalent instance with a simpler structure, which can then be solved efficiently.

In this paper we present a generic, abstract form of decomposition, which we call a *guarded decomposition*. We show that all of the earlier decomposition methods can be viewed as special cases of guarded decomposition, each characterized by some simple additional conditions.

One existing decomposition method, based on the use of hypertrees [Gottlob *et al.*, 2000; 2002], is of particular importance as it is has been shown to be strictly more general than many other decomposition methods and that, for each k, it is tractable to to discover hypertree decompositions of width at most k. The general framework presented here allows us to identify a new decomposition method, based on the use of a structure that we call a *spread cut*. We show that spread-cut decompositions generalize all of the methods previously shown to be generalized by hypertree decompositions, and still it is tractable, for each k to discover spread cut decompositions of width k. Furthermore, we exhibit a family of examples for which the spread-cut decomposition method obtains better decompositions than the hypertree decomposition method.

2 Constraint Satisfaction, Hypergraphs, and Tractability

Definition 2.1 *A **CSP** instance is a triple $P = \langle V, D, C \rangle$ where:*

- *V is a finite set of **variables**;*
- *D is a set called the **domain** of P;*
- *C is a set of **constraints**. Each constraint $c \in C$ is a pair $c = \langle \chi, \rho \rangle$ where $\chi \subseteq V$ is a set of variables, called the **scope** of c, and $\rho \subseteq D^{\chi}$ is a set of functions from the scope to the domain, called the **relation** of c.*

*A **solution** to the CSP instance $P = \langle V, D, C\rangle$ is a function from V to D whose restriction[1] to the scope of any constraint $c \in C$ is one of the functions in the relation of c.*

Example 2.2 Consider the CSP instance $P_{AG} = \langle V, D, C\rangle$ where $V = \{1, 2, \ldots, 10\}$, $D = \{0, 1\}$, and $C = \{c_1, c_2, \ldots, c_8\}$.

This instance has ten variables which must each be assigned the value 0 or 1, subject to 8 constraints. The constraints in C are defined as follows:

$$c_1 = \left\langle \{1,2\}, D^{\{1,2\}} \right\rangle, \quad c_2 = \left\langle \{2,3,9\}, D^{\{2,3,9\}} \right\rangle,$$
$$c_3 = \left\langle \{3,4,10\}, D^{\{3,4,10\}} \right\rangle, \quad c_4 = \left\langle \{4,5\}, D^{\{4,5\}} \right\rangle,$$
$$c_5 = \left\langle \{5,6,9\}, D^{\{5,6,9\}} \right\rangle, \quad c_6 = \left\langle \{6,7,10\}, D^{\{6,7,10\}} \right\rangle,$$
$$c_7 = \left\langle \{7,8,9\}, D^{\{7,8,9\}} \right\rangle,$$

$$c_8 = \langle \{1,8,10\}, \{f \mid f(1) = f(8) = 0 \rightarrow f(10) = 1\} \rangle .$$

Note that each constraint except c_8 allows *every* assignment to the variables of its scope. The constraint c_8 does not allow the three variables of its scope all to take the value 0 simultaneously, but does allow all other assignments.

A straightforward calculation shows that this instance has exactly $7 * 2^5 = 224$ solutions. □

In order to study the *structural* properties of CSP instances, that is, the way in which the constraint scopes overlap each other, we need the standard notion of a *hypergraph*.

Definition 2.3 *A **hypergraph** is a pair $H = \langle V, E\rangle$, where V is an arbitrary set, called the **vertices** of H, and E is a set of subsets of V, called the **hyperedges** of H.*

Definition 2.4 *For any CSP instance $P = \langle V, D, C\rangle$, the **structure** of P, denoted $\sigma(P)$, is the hypergraph $\langle V, \{\chi \mid \langle \chi, \rho\rangle \in C\}\rangle$.*

For any hypergraph H, the class of all CSP instances with structure H is denoted $\Psi(H)$.

Example 2.5 Recall the CSP instance P_{AG} defined in Example 2.2. The structure of P_{AG} is the hypergraph H_{AG} illustrated in Figure 1.

The set of vertices of H_{AG} is the set $\{1, 2, \ldots, 10\}$, and the eight hyperedges of H_{AG} are the following subsets of these vertices: $e_1 = \{1, 2\}$, $e_2 = \{2, 3, 9\}$, $e_3 = \{3, 4, 10\}$, $e_4 = \{4, 5\}$, $e_5 = \{5, 6, 9\}$, $e_6 = \{6, 7, 10\}$, $e_7 = \{7, 8, 9\}$, $e_8 = \{1, 8, 10\}$. □

As indicated in the Introduction, the decision problem for the general constraint satisfaction problem is NP-hard [Mackworth, 1977], so there has been considerable interest in finding more restricted problem classes whose instances can be recognized and solved in polynomial time.

Definition 2.6 *A class I of CSP instances is called **tractable** if there is a polynomial time algorithm to decide membership of I, and a polynomial time algorithm which solves all instances in I.*

*A class I of CSP instances is called **structural** if $I = \bigcup_{H \in \mathcal{H}} \Psi(H)$ for some class of hypergraphs $\mathcal{H}$.*

[1] To simplify the presentation we assume, throughout this paper, that every variable of a CSP instance is constrained; that is, every variable occurs in the scope of some constraint.

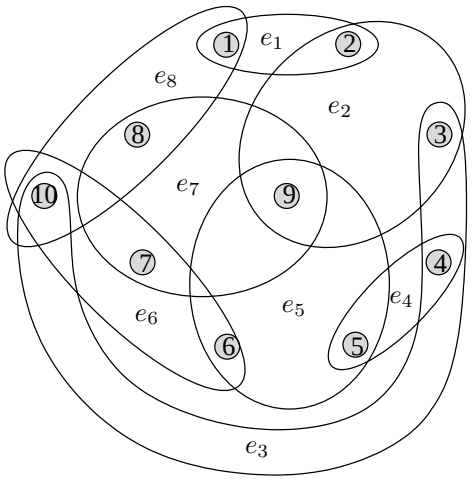

Figure 1: The hypergraph H_{AG}, which is the structure of the CSP instance P_{AG}, defined in Example 2.2.

3 Guarded Decompositions

Two CSP instances with the same set of variables are called *solution equivalent* if they have the same set of solutions. All known tractable structural classes of CSP instances are characterized by showing that the instances they contain can be transformed into solution equivalent instances with simpler structure. The constraints of these transformed instances are obtained by calculating the relational joins of certain constraint relations in the original instances, and then projecting these onto new scopes. To describe this general transformation scheme we introduce the following terminology.

Definition 3.1 *A **guarded block** of a hypergraph H is a pair $\langle \lambda, \chi\rangle$ where the **guard**, λ, is a set of hyperedges of H, and the **block**, χ, is a subset of the vertices of the guard.*

Definition 3.2 *For any CSP instance P, and any guarded block $\langle \lambda, \chi\rangle$ of $\sigma(P)$, the constraint **generated** by P on $\langle \lambda, \chi\rangle$ is the constraint $\langle \chi, \rho\rangle$, where ρ is the projection onto χ of the relational join of all the constraints of P whose scopes are elements of λ.*

Given a CSP instance P, with structure H, and a collection of guarded blocks of H, we can generate constraints on each of these guarded blocks to obtain a new collection of constraints, and hence a new CSP instance. In some cases, if the guarded blocks are carefully chosen, this new instance will be solution equivalent to P, and so can be used to solve P. If this property holds for *any* CSP instance with structure H, then the collection of guarded blocks will be called a *guarded decomposition* of H, which is formally defined as follows:

Definition 3.3 *A set of guarded blocks Ξ of a hypergraph H is called a **guarded decomposition** of H if for every CSP instance $P = \langle V, D, C\rangle \in \Psi(H)$, the instance $P' = \langle V, D, C'\rangle$, where C' is the set of constraints generated by P on the members of Ξ, is solution equivalent to P.*

How can we determine whether a given collection of guarded blocks is in fact a guarded decomposition? Theorem 3.5 be-

low gives a simple and efficient way to answer this question, based on the following properties.

Definition 3.4 *A guarded block $\langle \lambda, \chi \rangle$ of a hypergraph H* ***covers*** *a hyperedge e of H if e is contained in χ.*

A set of guarded blocks Ξ of a hypergraph H is called a ***guarded cover*** *for H if each hyperedge of H is covered by some guarded block of Ξ.*

A set of guarded blocks Ξ of a hypergraph H is called a ***complete guarded cover*** *for H if each hyperedge e of H occurs in the guard of some guarded block of Ξ which covers e.*

Theorem 3.5 *A set of guarded blocks Ξ of a hypergraph H is a guarded decomposition of H if and only if it is a complete guarded cover for H.*

Proof: Let Ξ be a set of guarded blocks of a hypergraph $H = \langle V, E \rangle$.

Suppose that Ξ is not a complete guarded cover for H. Choose $e \in E$ such that there is no guarded block $\langle \lambda, \chi \rangle \in \Xi$ for which $e \in \lambda$ and $e \subseteq \chi$. We will construct an instance P_e with structure H which will witness the fact that Ξ is not a guarded decomposition of H.

Let the domain of P_e be $D = \{0, 1\}$. For each edge $f \in E, f \neq e$, let the constraint of P_e with scope f allow all mappings from f to D. Finally let the constraint of P_e with scope e, c_e, allow all mappings from e to D except for the mapping that assigns the value 0 to all the vertices of e simultaneously.

Now let $\langle \lambda, \chi \rangle$ be any guarded block in the set Ξ. If $e \notin \lambda$ then the constraint generated by P_e on $\langle \lambda, \chi \rangle$ allows every assignment. On the other hand, if $e \in \lambda$, then, by the choice of e, we know that $e \not\subseteq \chi$. In this case we observe that any projection of c_e onto any proper subset of e allows all assignments. So again the constraint generated by P_e on $\langle \lambda, \chi \rangle$ allows every assignment.

Since all constraints generated by P_e on all elements of Ξ allow every assignment, the resulting CSP instance is not solution equivalent to P_e, and hence Ξ is not a guarded decomposition of H.

For the converse, suppose that Ξ is a complete guarded cover for H. Let P be an arbitrary CSP instance with structure H. We have to show that the CSP instance P_Ξ obtained by taking the constraints generated by P on each element of Ξ is solution equivalent to P.

Clearly, any solution to P is a solution to P_Ξ. On the other hand, by the completeness of Ξ, and the construction of P_Ξ, the projection of any solution to P_Ξ onto any edge $e \in E$ must be allowed by the constraint of P with scope e. Hence any solution to P_Ξ is also a solution to P. □

4 Tractability

We have shown that guarded decompositions can be used to transform a given CSP instance to a solution equivalent instance which may have a different structure.

In order to be able to use guarded decompositions to identify *tractable* structural classes we need to impose some additional conditions to ensure we have the following properties.

Tractable construction It must be possible to generate each of the new constraints in polynomial time.

Tractable solution It must be possible to solve the resulting instances in polynomial time.

Tractable discovery It must be possible to obtain a guarded decomposition of the type we are considering for a given hypergraph (or else discover that no such decomposition exists) in polynomial time.

We will now consider extra conditions that can be imposed on a guarded decomposition to ensure that it has each of these properties.

4.1 Tractable Construction

The time complexity of a relational join operation is $O(r^k)$, where k is the number of relations being joined, and r is the maximum number of tuples in any of these relations. Hence to ensure that we have the *tractable construction* property, it is sufficient to bound the number of constraint relations that need to be combined using the relational join operation. This can be done by bounding the number of hyperedges in the guard of any guarded block used in the decomposition.

Definition 4.1 *The* ***width*** *of a set of guarded blocks is the maximum number of hyperedges in any of its guards.*

For any fixed value of k, the class of guarded decompositions of width at most k has the tractable construction property.

4.2 Tractable Solution

One way to ensure that the new instances obtained by using a guarded decomposition have the *tractable solution* property, is to ensure that the structure of these new instances is *acyclic* [Beeri *et al.*, 1983]. The property of being acyclic can be defined as follows:

Definition 4.2 *A* ***join tree*** *of a hypergraph H is a connected tree, T, whose nodes are the hyperedges of H, such that, whenever the vertex x of H occurs in two hyperedges e_1 and e_2 of H then x occurs in each node of the unique path connecting e_1 and e_2 in T. In other words, the set of nodes of T in which x occurs induces a (connected) subtree of T.*

A hypergraph is called ***acyclic*** *if it has a join tree.*

Theorem 4.3 ([Gyssens *et al.*, 1994]) *Any CSP instance whose structure is acyclic can be solved in polynomial time.*

Definition 4.4 *A* ***join tree*** *of a set of guarded blocks Ξ of H is a connected tree, T, whose nodes are the elements of Ξ, such that, whenever the vertex x of H occurs in two blocks of Ξ then x occurs in each block of the unique path connecting them in T. In other words, the set of nodes of T for which x occurs in the block induces a (connected) subtree of T.*

A set of guarded blocks is ***acyclic*** *if it has a join tree.*

By Theorem 4.3, any class of acyclic guarded decompositions has the tractable solution property. Using Theorem 3.5 and Definition 4.4 we now show that an acyclic guarded decomposition can be obtained from an arbitrary acyclic guarded cover without increasing the width, or significantly increasing the number of guarded blocks, by adding appropriate additional guarded blocks to make a complete guarded cover.

Theorem 4.5 *If the set of guarded blocks Ξ is an acyclic guarded cover for H then the set $\Xi \cup \{\langle\{e\}, e\rangle \mid e \in E\}$ is an acyclic guarded decomposition of H.*

In view of this result, we shall focus on acyclic guarded covers and use them to define classes of decompositions.

4.3 Tractable Discovery

The class of hypergraphs having a guarded cover in some fixed class Δ, with width at most k, will be denoted $C(\Delta, k)$.

Definition 4.6 *A class of guarded covers Δ will be called **parameterized tractable** [2] if for each value of k there exists a polynomial-time algorithm which, given any hypergraph $H \in C(\Delta, k)$, returns a guarded cover of H with width at most k, and given any hypergraph $H \notin C(\Delta, k)$, reports failure.*

It follows from the results above that if Δ is a class of acyclic guarded covers, which is parameterized tractable, then $C(\Delta, k)$ will be a tractable structural class of CSP instances, for each value of k.

5 Known Tractable Structural Classes

We will now show that many existing decomposition methods for the CSP can be defined in terms of finding acyclic guarded covers with particular special properties.

We first note that for historical reasons some existing decomposition methods for the CSP make use of *extended guards* that contain both vertices and hyperedges. To be able to present all of these methods in a uniform way we introduce the idea of transforming a CSP instance by adding a unary constraint c_v, for each variable v, where $c_v = \langle\{v\}, \{v\}^D\rangle$. This corresponds to extending the structure of the instance to ensure that it includes a hyperedge for each variable.

Definition 5.1 *Let $H = \langle V, E\rangle$ be any hypergraph. A guarded cover for the hypergraph $\langle V, E \cup \{\{v\} \mid v \in V\}\rangle$ is called an **extended cover** for H.*

The following result shows that the existence of an extended cover of width k is equivalent to the existence of a standard guarded cover of width k. This means that extended covers are simply a notational convenience and do not allow better decompositions.

Theorem 5.2 *If a hypergraph H has an acyclic extended cover of width k, then H also has an acyclic guarded cover of width at most k with at most the same number of guarded blocks.*

We also note that many existing decomposition methods are based on guarded covers in which every block is exactly the union of the edges of its guard.

Definition 5.3 *A guarded cover Ξ is **edge-defined** if the block of each guarded block in Ξ is exactly the set of vertices contained in the hyperedges of its guard.*

We can now characterize many known structural decomposition methods (see [Gottlob *et al.*, 2000] for traditional definitions) in terms of a corresponding class of acyclic guarded covers with certain additional conditions.

[2] The notion of being parameterized tractable that we have defined here is strictly weaker than the well-known notion of being fixed parameter tractable.

In each of the cases listed below (except for query decompositions), it has been shown that the additional conditions are sufficient to ensure that the corresponding class of acyclic guarded covers, Δ, is parameterized tractable. Hence, in all cases (except query decompositions), the class of CSP instances in $C(\Delta, k)$ is a tractable structural class.

- A **biconnected-component** decomposition of a hypergraph is an edge-defined acyclic complete guarded cover, Ξ, satisfying an **articulation condition**:

$$\forall \langle\lambda_1, \chi_1\rangle, \langle\lambda_2, \chi_2\rangle \in \Xi, \quad |\chi_1 \cap \chi_2| \leq 1$$

- A **cycle-hypercutset** decomposition of a hypergraph $H = \langle V, E\rangle$ is an edge-defined acyclic complete guarded cover, Ξ, satisfying a **simplicity condition**:

$$\exists C \subseteq E, \forall \langle\lambda, \chi\rangle \in \Xi, \quad |\lambda - C| \leq 1$$

- A **hinge-tree** decomposition of a hypergraph H is an edge-defined acyclic complete guarded cover, Ξ, satisfying a **separation condition**:

$$\forall \langle\lambda_1, \chi_1\rangle, \langle\lambda_2, \chi_2\rangle \in \Xi, \exists e \in \lambda_1 \cap \lambda_2, \chi_1 \cap \chi_2 \subseteq e.$$

- A **cycle-cutset** decomposition of a hypergraph H is an edge-defined acyclic extended cover, Ξ, such that every hyperedge of every guarded block of Ξ consists of a single vertex, and Ξ satisfies a **simplicity condition**:

$$\exists C \subseteq V, \forall \langle\lambda, \chi\rangle \in \Xi, \exists e \in E, (\chi - C) \subseteq e$$

- A **query decomposition** of a hypergraph $H = \langle V, E\rangle$ is a pair $\langle\Xi, T\rangle$ where Ξ is a complete edge-defined acyclic extended cover of H and T is a join tree of Ξ that satisfies a **connectedness condition**:

$$\forall e \in E, \quad \{\langle\lambda, \chi\rangle \in \Xi \mid e \in \lambda\} \text{ is connected in } T.$$

- A **hypertree decomposition** of a hypergraph H is a pair $\langle\Xi, T\rangle$, where Ξ is an acyclic guarded cover and T is a rooted join tree of Ξ, which satisfies the following **descendant condition**:

$$\forall \langle\lambda, \chi\rangle \in \Xi, \quad \left(\left(\bigcup_{e \in \lambda} e\right) \cap \bigcup_{\langle\lambda_i, \chi_i\rangle \in D_{\langle\lambda,\chi\rangle}} \chi_i\right) \subseteq \chi$$

  where $D_{\langle\lambda,\chi\rangle}$ is the set of all descendants of $\langle\lambda, \chi\rangle$ in T. The minimum width of any hypertree decomposition of a hypergraph H is called the **hypertree width** of H.

6 Comparing Decompositions

The relative strengths of different decomposition techniques derived from acyclic guarded covers can be compared using the measures developed by Gottlob et al [2000].

Definition 6.1 *Let Δ_1 and Δ_2 be any two classes of guarded covers. We say that Δ_1 **generalizes** Δ_2 if there exists a value $\varepsilon > 0$ such that, for every k, $C(\Delta_2, k) \subseteq C(\Delta_1, k + \varepsilon)$.*

*We say that Δ_1 **strongly generalizes** Δ_2 if Δ_1 generalizes Δ_2, and there exists k for which there does not exist $\varepsilon > 0$ with $C(\Delta_1, k) \subseteq C(\Delta_2, k + \varepsilon)$.*

Hypertrees have been shown to strongly generalize all the other parameterized tractable classes of decompositions defined in the previous section [Gottlob *et al.*, 2000]. Furthermore it has recently been shown that the hypertree width of any hypergraph is at most three times the minimal possible width of any acyclic guarded cover [Adler, 2004]. It follows that hypertrees cannot themselves be strongly generalized by any other class of acyclic guarded covers.

However, we will show in the next section that it is possible to define a parameterized tractable class Δ of acyclic guarded covers, such that for some families of hypergraphs, the minimal width of a guarded cover in Δ is smaller than the hypertree width by some constant factor. Such an improvement allows the corresponding instances to be solved exponentially faster than by using hypertree decompositions.

Definition 6.2 *Let Δ_1 and Δ_2 be any two classes of guarded covers. We say that Δ_1 **can be superior** to Δ_2 if there is some $r > 1$, such that for every k, $C(\Delta_2, rk) \not\subseteq C(\Delta_1, k)$.*

7 Spread Cuts

In this section we will define a new class of acyclic guarded covers, called *spread cuts*. We show that classes of instances with bounded spread cut width are tractable and that spread cuts can be superior to hypertrees.[3]

Throughout this section, we will write $\bigcup\lambda$ to refer to the vertices of a set of hyperedges λ; that is, $\bigcup\lambda = \bigcup_{e\in\lambda} e$.

Definition 7.1 *Let $H = \langle V, E\rangle$ be a hypergraph and $\chi \subseteq V$ be a set of vertices. We say that two hyperedges $e, f \in E$ are χ-**adjacent** if $e \cap f \not\subseteq \chi$.*

*A χ-**path** connecting e to f is a sequence $e = e_0, e_1, \ldots, e_{r-1}, e_r = f$ such that e_i is χ-adjacent to e_{i+1}, for $i = 0, \ldots, r-1$.*

*A set of hyperedges $C \subseteq E$ is χ-**connected** if, for every pair of hyperedges in C, there is a χ-path connecting them.*

*A set of hyperedges is a **hyperedge** χ-**component** of H if it is a maximal non-empty χ-connected subset of E.*

*A non-empty set of vertices C is a **vertex** χ-**component** of H if there is some hyperedge χ-component C_χ for which $C = \bigcup C_\chi - \chi$.*

In this paper, unless otherwise stated, a component will mean a vertex component.

Example 7.2 Consider again the hypergraph H_{AG} defined in Example 2.5 and illustrated in Figure 1.

Now consider the guarded block $\langle\lambda, \chi\rangle$ with $\lambda = \{e_2, e_6\}$ and $\chi = \{3, 6, 7, 9, 10\}$.

There are three hyperedge χ-components: $\{e_6\}$, $\{e_1, e_2, e_7, e_8\}$ and $\{e_3, e_4, e_5\}$. Hence, there are two vertex χ-components: $\{4, 5\}$ and $\{1, 2, 8\}$.

There are four hyperedge $\bigcup\lambda$-components: $\{e_2\}$, $\{e_6\}$, $\{e_1, e_7, e_8\}$ and $\{e_3, e_4, e_5\}$. Hence, there are two vertex $\bigcup\lambda$-components: $\{4, 5\}$ and $\{1, 8\}$.

Notice that each vertex χ-component meets (has non-empty intersection with) exactly one vertex $\bigcup\lambda$-component.

[3]It is still an open research question whether hypertrees can be superior to spread-cuts.

This special property of this guarded block relies on χ being "large enough". If χ were empty, then there would be just one vertex χ-component and this component would meet both vertex $\bigcup\lambda$-components. □

If we restrict the possible blocks for a given guard appropriately then we can polynomially bound the number of guarded blocks, whilst still getting effective decompositions. This motivates the following definition.

Definition 7.3 *A guarded block $\langle\lambda, \chi\rangle$ of a hypergraph H has **unbroken components** if each χ-component of H meets at most one $\bigcup\lambda$-component of H and $\{e_1 \cap e_2 \mid e_1, e_2 \in \lambda\} \subseteq \chi$.*

*A **spread cut** of H is an acyclic guarded cover for H in which all of the guarded blocks have unbroken components.*

Example 7.4 Consider again the hypergraph H_{AG} defined in Example 2.5 and illustrated in Figure 1.

The minimal width of any hypertree decomposition of H_{AG} is three[4].

The following set of guarded blocks is a spread cut of H_{AG} of width two: $\langle\{e_2, e_6\}, \{3, 6, 7, 9, 10\}\rangle$, $\langle\{e_3, e_5\}, \{3, 4, 5, 6, 9, 10\}\rangle$, $\langle\{e_2, e_8\}, \{1, 2, 3, 8, 9, 10\}\rangle$, $\langle\{e_3, e_7\}, \{3, 7, 8, 9, 10\}\rangle$. □

7.1 Spread Cuts are Parameterized Tractable

We will now define a canonical form for guarded blocks with unbroken components, and show that if a hypergraph H has a spread cut of width k, then it also has a spread cut of width at most k, in which each guarded block is canonical.

Definition 7.5 *Let λ be any set of hyperedges of H. We define the **label**, $L_\lambda(v)$, of any vertex of H to be the set of hyperedge $\bigcup\lambda$-components which include a hyperedge containing v.*

*We say that a guarded block $\langle\lambda, \chi\rangle$ is **canonical** if for each hyperedge $e \in \lambda$ the vertices of e outside of χ are exactly those with a particular label. That is,*

$$\forall v \in e - \chi, \forall w \in e, \quad w \notin \chi \leftrightarrow L_\lambda(w) = L_\lambda(v).$$

Theorem 7.6 *If H has a spread cut of width k then H has a spread cut of width at most k, in which each guarded block is canonical.*

Proof: Due to space limitations, we can only give an outline of the proof.

The idea is to show that, for any guarded block $\langle\lambda, \chi\rangle$ with unbroken components, and for any hyperedge $e \in \lambda$, $|\bigcup_{v\in e-\chi} L_\lambda(v)| \leq 2$. Given this fact, we can start with any spread cut and, for each hyperedge of each guard, remove from the block any vertex with the same label as any vertices already outside of the block. It is straightforward to show that the result is still an acyclic guarded cover, and hence still a spread cut with the same width, where each guarded block is now canonical. □

Using this canonical form, we can establish the following result.

Theorem 7.7 *Spread cuts are parameterized tractable.*

[4]This is most easily shown by using the "Robbers and Marshals" characterization of hypertree width [Gottlob *et al.*, 2003]. However, due to space restriction we omit this proof.

Proof: Due to space limitations, we can only give an outline of the proof.

The proof idea is that we only have to establish in polynomial time whether there exists a spread cut of width at most k in which each guarded block is canonical. We do this by first defining an object called a k-spread, which is a triple, $\langle \lambda, \chi, C \rangle$, where $\langle \lambda, \chi \rangle$ is any canonical guarded block with unbroken components, $|\lambda| \leq k$, and C is a χ-component.

We then define the predicate k-split() (recursively) on k-spreads. We say that k-split$(\langle \lambda, \chi, C \rangle)$ holds if C is empty, or if there exists some canonical guarded block, $\langle \lambda_s, \chi_s \rangle$, where $\chi_s \subseteq C \cup \chi$ and, for every k-spread $\langle \lambda_s, \chi_s, C_s \rangle$:

$C_s \cap C \neq \emptyset \Rightarrow \forall e \in E, (e \cap C_s \neq \emptyset \Rightarrow e \cap \chi \subseteq \chi_s)$, and
$C_s \subseteq C \Rightarrow$ k-split$(\langle \lambda_s, \chi_s, C_s \rangle)$.

It is easy to show that the number of k-spreads of a hypergraph $H = \langle V, E \rangle$ is at most $(|E| + 1)^{k+2}$. It follows (as in the corresponding proof for hypertrees, Lemma 14, Page 30, of Gottlob et al [2002]) that k-split$(\langle \emptyset, \emptyset, V \rangle)$ can be evaluated in at most $|E|^{4k+5}$ steps.

Finally, we show that H has a spread cut with canonical guarded blocks of width at most k if and only if k-split$(\langle \emptyset, \emptyset, V \rangle)$. The only hard part of this proof is to show that the set of guarded blocks used in the proof of k-split$(\langle \emptyset, \emptyset, V \rangle)$ is a guarded cover. This proof is exactly analogous to the corresponding proof for the algorithm that finds normal form hypertree decompositions (See the proof of Property 1 in Lemma 13, Page 28, of Gottlob et al [2002]). □

The proof of Theorem 7.7 also shows that classes of instances with bounded spread cut width are tractable.

7.2 Spread Cuts and Hypertrees

To establish that spread cuts strongly generalize all of the parameterized tractable decomposition methods listed in Section 5 excluding hypertrees we simply refer to the proofs of strong generalization given by Gottlob et al [2000] and observe that every hypertree (of small width) used to establish strong generalization in that paper is also a spread cut.

To show that spread cuts can be superior to hypertrees we establish the following result.

Proposition 7.8 *There exists a family of hypergraphs $H_n, n = 1, 2, 3, \ldots$, such that H_n has a spread cut with width $2n$, but the hypertree width of H_n is at least $3n$.*

Proof: Due to space restrictions we can only give an outline of the proof.

The idea is to use n copies of the hypergraph H_{AG} defined in Example 2.5 to construct H_n. We add edges linking vertices in different copies if they correspond to vertices of hyperedges that meet in H_{AG}. Using the "Robbers and Marshals" characterization of hypertree width [Gottlob *et al.*, 2003], we then show that the hypertree width of H_n is at least $3n$. (If there are fewer than $3n$ marshals, the robber can always choose to play in the copy of H_{AG} with at most two marshals on its hyperedges.)

Finally, the width $2n$ spread cut is just the spread cut of H_{AG} with width 2 described in Example 7.4 "multiplied" over the n copies of H_{AG}. □

8 Conclusion

We have introduced the general notion of a guarded decomposition and shown how it can be used to describe many known tractable structural subproblems of the CSP. We have then defined a new form of decomposition, the spread cut, which sits in the same place in the generalization hierarchy as hypertrees. We have shown that spread cuts are parameterized tractable and that they allow us to obtain decompositions with a smaller width than any hypertree decomposition for some classes of hypergraphs.

It is an open question whether the general class of all acyclic guarded covers is itself parameterized tractable.

References

[Adler, 2004] I. Adler. Marshals, monotone marshals, and hypertree-width. *Journal of Graph Theory*, 47(4):275–296, December 2004.

[Beeri *et al.*, 1983] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM*, 30:479–513, 1983.

[Dechter and Pearl, 1988] R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34(1):1–38, 1988.

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

[Dechter, 2003] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

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

[Gottlob *et al.*, 2000] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. *Artificial Intelligence*, 124:243–282, 2000.

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

[Gottlob *et al.*, 2003] G. Gottlob, N. 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.

[Gyssens *et al.*, 1994] M. Gyssens, P.G. Jeavons, and D.A. Cohen. Decomposing constraint satisfaction problems using database techniques. *Artificial Intelligence*, 66(1):57–89, 1994.

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

[Mackworth, 1977] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[Marriott and Stuckey, 1998] K. Marriott and P. Stuckey. *Programming with Constraints*. MIT Press, Cambridge, Massachusetts, 1998.