Hypertree Decomposition and Constraint Satisfaction Problems

Loïc Blet

November 30, 2010

Abstract

In this report we present the hypertree decomposition for hypergraphs and show its use for solving Constraint Satisfaction Problems (CSP) .

We will see a first motivation for this technique when showing that hypertree decomposition strongly generalize tree decomposition. The main interest though is that, for a CSP, given a hypertree decomposition of bounded width, we know how to efficiently solve the given CSP.

Moving from there, we generalize the previous hypertree decomposition method, and we will see that it is a NP complete problem to decide whether the generalized hypertree width of a given hypergraph is less or equal than $k = 3$ (and thus the NP completeness holds if k is greater than 3). We will conclude with some ideas to go past this problem.

Contents

1 Introduction

We know that some hard problems on graphs become easy when dealing with instances on acyclic graphs. The tree decomposition method together with the associated notion of tree width is a successful attempt to transform *some* instances into tractable ones. But we would like to treat *some more*. This is the motivation for decomposing hypergraphs with hypertrees: the method is more general and will allow bigger classes of graphs to be tractable.

In our first section, we will define the kind of problems we want to solve — constraint satisfaction problems — and their associated hypergraphs. Then we will describe the tree and hypertree decomposition method on hypergraphs.

In the following section we bring some tools for comparing decomposition methods. This will allow us to prove that the hypertree decomposition method strongly generalizes the tree decomposition method.

The last section deals with an even more general method, smartly called generalized hypertree decomposition. We will see that it is a hard problem to find a generalized hypertree decomposition of bounded width.

2 Definitions

To start with, let us give some definitions, on the kind of problems we want to solve — constraint satisfaction problems — and the techniques we use to solve those — hypergraphs decomposition methods.

2.1 Constraint satisfaction problems

Definition 1 (CSP)**.** *A* constraint satisfaction problem *(CSP) is a three-tuple* $\langle \boldsymbol{U}, \boldsymbol{D}, \boldsymbol{C} \rangle$ *where*

- $\boldsymbol{U}$ *is a finite set of variables;*
- $\boldsymbol{D}$ *is a domain containing the possible values for the variables in* $\boldsymbol{U}$*; and*
- $\boldsymbol{C}$ *is a set of constraints, each constraint* $c = (S, r)$ *in* $\boldsymbol{C}$ *being defined on a set of variables* S*, its scope, taken from* $\boldsymbol{U}$ *and specifying the allowed combinations of values for that set by the relation* r*.*

In order to solve a CSP we must find an assignment (or configuration) to all the variables that satisfies all the constraints.

Definition 2 (Configuration and Solution)**.** *Let* $P = \langle \boldsymbol{U}, \boldsymbol{D}, \boldsymbol{C} \rangle$ *be a CSP:*

- *A* configuration *is a function* $k : \boldsymbol{U} \to \boldsymbol{D}$*.*

- *A configuration k is a* solution *to $c = (S, r) \in \boldsymbol{C}$ (or k satisfies c, or c is satisfied under k) if and only if $\{x_1, \ldots, x_m\}$ is the variable set S and $\{k(x_1), \ldots, k(x_m)\}$ is one of the allowed combinations of values for that set as specified by r.*

- *A configuration k is a* solution *to P if and only if k is a solution to all constraints in $\boldsymbol{C}$.*

The problem of deciding whether a CSP instance has any solution is called *constraint satisfiability*, and it is a NP complete problem in general. This lead to an active research effort to find tractable classes of CSPs. We will only deal with CSP that are tractable due to restricted structure. This is the tractable CSPs identified only based on the structure of the constraints scopes (not taking in account the relation constraint).

And from now on we can switch to pure graph theory, based on this motivation of solving CSPs !

To represent the structure of a CSP we define its associated *hypergraph* and the corresponding *primal graph* as follows.

Definition 3 (Associated hypergraph for a CSP)**.** *Let $P = \langle \boldsymbol{U}, \boldsymbol{D}, \boldsymbol{C} \rangle$ be a CSP. We define its associated hypergraph $\mathcal{H}_P = (V, H)$, where:*

- $V = \boldsymbol{U}$
- $H = \{S | \forall (S, r) \in \boldsymbol{C}\}$

Definition 4 (Primal graph for a hypergraph)**.** *The primal graph of a hypergraph H is a graph with the same set of vertices as H and has a clique corresponding to each edge in H.*

The most important structural property for a CSP is *acyclicity*. It has been shown that *acyclic* CSPs are polynomially solvable.

Definition 5 (Acyclicity for a CSP)**.** *A CSP is acyclic if its primal graph is chordal and the set of its maximal cliques coincide with set the edges of its associated hypergraph.*

In practice, many CSPs are only *close* to being acyclic, and thus leading to some definitions of *nearly acyclic* CSPs and also leading to methods for *decomposing* cyclic CSPs into acyclic CSPs. This is what we will see in the next section.

2.2 Tree and hypertree decomposition

We now introduce the *tree decomposition* and the *tree width* of a graph.

Definition 6 (Tree decomposition)**.** *A* tree decomposition *of a graph $G = (V, E)$ is a pair $\langle T, \chi \rangle$, where $T = (N, F)$ is a tree, and χ is a labeling function associating to each vertex $p \in N$ a set of vertices $\chi(p) \subseteq V$, such that:*

1. *$\forall v \in V, \exists p \in N, v \in \chi(p)$;*

2. $\forall (v,u) \in E, \exists p \in N, \{v,u\} \subseteq \chi(p)$;

3. *for each vertex* $v \in G$, *the set* $\{p \in N | v \in \chi(p)\}$ *induces a subtree of* T.

We then define the *tree width* of a graph and a hypergraph.

Definition 7 (Tree width)**.** *the* width *of a tree decomposition* $\langle T, \chi \rangle$ *is* $\max_{p \in N} |\chi(p)| - 1$.
The tree width *of* G *is the minimum width over all its tree decomposition.*
The tree width *of a hypergraph* $\mathcal{H}$ *is 1 if* $\mathcal{H}$ *is acyclic, and equal to the tree width of its primal graph otherwise.*

Let us now see a generalization of this tree decomposition to hypergraphs: the *hypertree decomposition*.

Definition 8 (Hypertree for a hypergraph)**.** *A* hypertree for a hypergraph $\mathcal{H}$ *is a triple* $\langle T, \chi, \lambda \rangle$, *where* $T = (N, E)$ *is a rooted tree, and* χ *and* λ *are labeling functions which associates to each vertex* $p \in N$ *a set* $\chi(p) \subseteq vertices(\mathcal{H})$, *and a set* $\lambda(p) \subseteq edges(\mathcal{H})$.

If $T' = (N', E')$ is a subtree of T, we define $\chi(T') = \bigcup_{v \in N'} \chi(v)$. We denote the root of T by $root(T)$. For any $p \in N$, T_p denotes the subtree of T rooted at p.

Definition 9 (Hypertree decomposition)**.** *A* hypertree decomposition *of a hypergraph* $\mathcal{H}$ *is a hypertree* $HD = \langle T, \chi, \lambda \rangle$ *for* $\mathcal{H}$ *verifying the following conditions:*

1. $\forall h \in edges(\mathcal{H}), \exists p \in vertices(T), vertices(h) \subseteq \chi(p)$, *We say that* p covers h;

2. *for each vertex* $Y \in vertices(\mathcal{H})$, *the set* $\{p \in vertices(T) | Y \in \chi(p)\}$ *induces a subtree of* T;

3. $\forall p \in vertices(T), \chi(p) \subseteq vertices(\lambda(p))$;

4. $\forall p \in vertices(T), vertices(\lambda(p)) \cap \chi(T_p) \subseteq \chi(p)$.

We can now define the *hypertree width* of a hypergraph.

Definition 10 (hypertree width)**.** *the* width *of a hypertree decomposition* $\langle T, \chi, \lambda \rangle$ *is* $\max_{p \in vertices(T)} |\lambda(p)|$.
The hypertree width *of a hypergraph is the minimum width over all its hypertree decompositions.*

The acyclic hypergraphs have a hypertree width of one.

2.3 An example of hypertree decomposition for a CSP

Let us illustrate some of the definitions we gave. We first define a crossword CSP, then give its associated hypergraph, and a hypertree decomposition of minimal width. This whole example is taken from [GLS99].

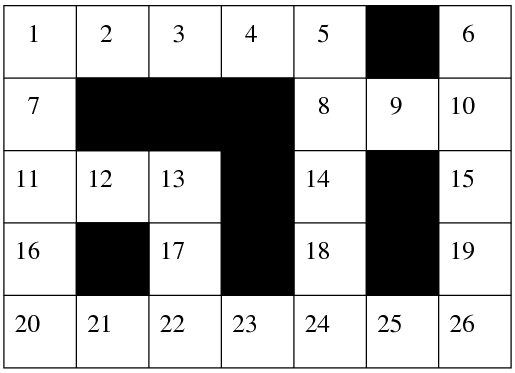

Figure 1: A crossword puzzle

Example 1 (A crossword CSP)**.** *Figure 1 shows a crossword grid, that can be expressed as a CSP. A solution to this puzzle is an assignment of a letter to each numbered box, such that each horizontal or vertical array is assigned a legal word.*

Let us model this as a CSP $P\langle \boldsymbol{U}, \boldsymbol{D}, \boldsymbol{C}\rangle$*. The domain* $\boldsymbol{D}$ *is the set of letters of the alphabet. There is a variable* i *for each numbered box in the grid. There is a constraint* c *for each array* A *of numbered boxes. The scope of* c *is the list of variables corresponding to the boxes in* A*. So, in this example, we have the following constraints. For the horizontal constraints:* $C_{1H} = (\{1, 2, 3, 4, 5\}, r_{1H})$*,* $C_{8H} = (\{8, 9, 10\}, r_{8H})$*, ... For the vertical constraints:* $C_{1V} = (\{1, 7, 11, 16, 20\}, r_{1V})$*,* $C_{5V} = (\{5, 8, 14, 18, 24\}), r_{5V})$*, ... Where subscripts* H *and* V *distinguish horizontal and vertical constraints. We then have to define each relation* r*, based on a dictionary for instance.*

Now we can get the hypergraph of this CSP.

Example 2 (Hypergraph of the crossword CSP)**.** *As the constraints are already represented in a graphical way thanks to the grid of the puzzle, the hypergraph is easy to draw. Recall that there will be a vertex for each variable in the CSP, and an edge for each constraint scope. We obtain the hypergraph shown in figure 2.*

To conclude this section, we give a hypertree decomposition of the previous hypergraph $\mathcal{H}_{cp}$, and show that it has minimal width.

Example 3 (Hypertree decomposition)**.** *Figure 3 shows a hypertree decomposition of the hypergraph* $\mathcal{H}_{cp}$*. Each box* b *in the figure represents a node* v *of the hypertree decomposition. In each box, the left set is* $\chi(v)$ *and the right set is* $\lambda(v)$*. We see that this hypertree decomposition is of width 2 (as* $\forall v|\lambda(v)| = 2$*). As the hypergraph* $\mathcal{H}_{cp}$ *is clearly cyclic its hypertree width is strictly larger than 1. Thus the hypertree width of* $\mathcal{H}_{cp}$ *is 2.*

In the next section we compare the tree and hypertree decomposition methods.

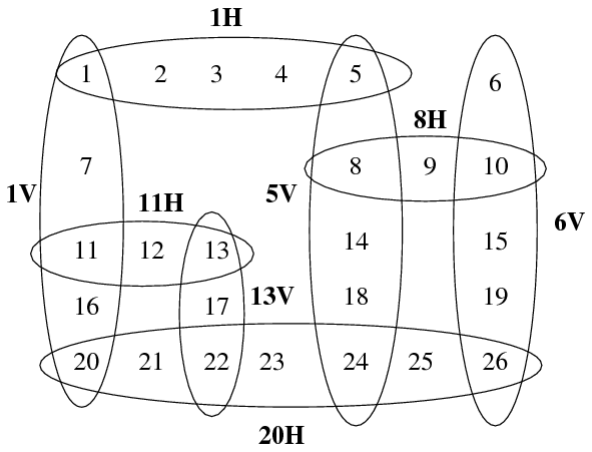

Figure 2: The hypergraph $\mathcal{H}_{cp}$ for the crossword puzzle

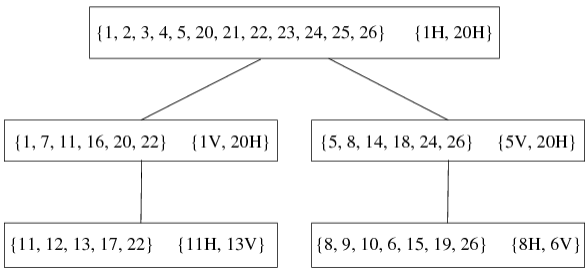

Figure 3: A hypertree decomposition of the hypergraph $\mathcal{H}_{cp}$

3 Hypertree decomposition strongly generalizes tree decomposition [GLS99]

We have seen two different decomposition method, the tree and hypertree decomposition, denoted TD and HD respectively. They allow us to define classes of tractable CSPs:

- $C(TD, 1) \subset C(TD, 2) \subset \ldots \subset C(TD, i) \subset \ldots$
- $C(HD, 1) \subset C(HD, 2) \subset \ldots \subset C(HD, i) \subset \ldots$

Each such class $C(D, k)$ is solvable in time bounded by a polynomial. This is achieved thanks to a decomposition of width k given by method D. All of the following steps must be tractable for $C(D, k)$ to be tractable:

- Checking membership of a CSP p in $C(D, k)$.
- Computing a CSP decomposition of P with method D.
- Transforming P into an equivalent acyclic CSP P'.
- Solving P'.

We need tools to compare different decomposition method. We thus introduce the definitions of *generalization* and *beating* which will lead to *strong generalization* when combined.

Definition 11 (Generalization)**.** *We say that* D_1 generalizes D_2 *if:*

$$\exists \delta, \forall k, C(D_2, k) \subseteq C(D_1, k + \delta)$$

The meaning of this is that any tractable class of CSPs according to D_1 *is also tractable according to* D_2*.*

Definition 12 (Beating)**.** *A method* D_1 beats *a method* D_2 *if:*

$$\exists k, \forall m, C(D_1, k) \nsubseteq C(D_2, m)$$

We understand this as: some problems are tractable according to D_1 *and not tractable according to* D_2*.*

The final comparison tool is the concept of *strong generalization*. Decomposition method D_1 *strongly generalizes* decomposition method D_2 if D_1 both generalizes and beats method D_2.

Let us now prove the following theorem:

Theorem 1. *Hypertree decomposition strongly generalizes tree decomposition*

Proof 1. *First we have to show that hypertree decomposition generalizes tree decomposition. This follows from the definition of the methods : the hypertree decomposition method is a generalization of the tree decomposition method, meaning that any tree decomposition can be transformed into a hypertree decomposition of equal or lesser width. So Hypertree decomposition generalizes tree decomposition.*

Then we have to show that hypertree decomposition beats tree decomposition. Let us consider the class of hypergraphs K_n that have n vertices and only one edge that covers all n vertices. A hypertree decomposition of this would obviously be of width 1, hence we have:

$$\bigcup_{n>1} K_n \subseteq C(HD, 1)$$

On the other hand, a tree decomposition of this, would be a tree decomposition of the primal graph of K_n, which is the clique of size n, leading to a width equal to n, hence we have:

$$\forall d, \bigcup_{n>1} K_n \nsubseteq C(TD, d)$$

Thus we can conclude that hypertree decomposition strongly generalizes tree decomposition.

□

We ensured that going to hypertree decomposition is useful since it is strongly more general than the tree decomposition. Further results made it clear that a CSP of bounded hypertree width can be solved efficiently. But still, we could have a more general method, and we will study a bit the generalized hypertree width.

4 Finding a good generalized hypertree decomposition is hard [GMS09]

Recall the definition of hypertree decomposition (see section 2.2). The fourth condition of this definition was put there only because it was needed to prove that any CSP of bounded hypertree width is efficiently solvable. Thus this condition is not intrinsic and can be removed in order to create a more general decomposition method, hence the name of generalized hypertree decomposition.

Definition 13 (Generalized hypertree decomposition)**.** *A* generalized hypertree decomposition *of a hypergraph $\mathcal{H}$ is a hypertree $GHD = \langle T, \chi, \lambda \rangle$ for $\mathcal{H}$ verifying the following conditions:*

1. $\forall h \in edges(\mathcal{H}), \exists p \in vertices(T), vertices(h) \subseteq \chi(p)$*, We say that p* covers *h;*
2. *for each vertex $Y \in vertices(\mathcal{H})$, the set $\{p \in vertices(T) | Y \in \chi(p)\}$ induces a subtree of T;*
3. $\forall p \in vertices(T), \chi(p) \subseteq vertices(\lambda(p))$*;*

And we get the generalized hypertree width definition the same way as we got the definition of hypertree width.

Before stating the main result of this section, let us introduce some needed material for the upcoming proof.

First we define the $\leq$ relation on hypergraphs: we write $H_1 \leq H_2$ if each edge of H_1 is contained in at least one edge of H_2. Now we can state what a *tree projection* is.

Definition 14 (tree projection)**.** *A* tree projection *of H with respect to G is an acyclic hypergraph H' such that $G \leq H' \leq H$.*

Then for a hypergraph $H = (V, E)$ we denote by H^k the hypergraph (V, E^k) where E^k is the set of all unions of k or less hyperedges from H. We can see, from the definitions of generalized hypertree decomposition and of tree projection, that the following lemma holds:

Lemma 1. *A hypergraph H has a generalized hypertree width bounded by k if and only if H^k has a tree projection with respect to H.*

Let us move on to the hardness result for this decomposition method.

Theorem 2. *Deciding whether a hypergraph H has a generalized hypertree width of at most 3 is NP Complete.*

Proof 2 (sketch)**.** *The aim is to polynomially transform 3SAT into an instance of the tree projection problem of the form (G, G^3). We will admit the NP completeness of the tree projection problem.*

The graph G will be a complicated construction. The idea is to have a first subhypergraph H with some special properties, based on a propositional formula in conjunctive normal form ρ. To construct G we put two copies H and H' of the subhypergraph mentioned before and link these with well chosen hyperedges. Then the projection tree T of G with respect to G^3 is carefully created as path between the subpart H to the subpart H'.

The construction ensures that for any variable x_i of ρ, $\rho(x_i) = 1$ if a given vertex is in a specific set of the projection tree and $\rho(x_i) = 0$ otherwise. This conclude the (sketch of the) proof.

□

5 Conclusion

Starting from the tree decomposition method we have defined a generalization and seen that hypertree decomposition, while being strongly more general still defines tractable classes of CSPs. By giving up on one condition in the definition of this method we ended up with generalized hypertree decomposition. this technique has a more intuitive definition but decomposing a hypergraph with this method will not be tractable in general since deciding whether a hypergraph is of bounded generalized hypertree width is a NP complete problem. This motivates the search for decomposition method more general than hypertree decomposition but less than the generalized version, and this search has been started in [GMS09].

References

[GLS99] Georg Gottlob, Nicola Leone, and Francesco Scarcello. A comparison of structural csp decomposition methods. In *Proceedings of the 16th international joint conference on Artifical intelligence - Volume 1*, pages 394–399, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[GMS09] Georg Gottlob, Zoltán Miklós, and Thomas Schwentick. Generalized hypertree decompositions: Np-hardness and tractable variants. *J. ACM*, 56:30:1–30:32, September 2009.