TECHNISCHE UNIVERSITÄT DRESDEN

Artificial Intelligence, Computational Logic

PROBLEM SOLVING AND SEARCH IN ARTIFICIAL INTELLIGENCE

Lecture 11 Hypertree Decompositions

Sarah Gaggl

Dresden, 14th January 2020

DRESDEN concept Exzellenz aus Wissenschaft und Kultur

Agenda

1. Introduction
2. Uninformed Search versus Informed Search (Best First Search, A* Search, Heuristics)
3. Local Search, Stochastic Hill Climbing, Simulated Annealing
4. Tabu Search
5. Answer-set Programming (ASP)
6. Constraint Satisfaction Problems (CSP)
7. Evolutionary Algorithms/ Genetic Algorithms
8. Structural Decomposition Techniques (Tree/Hypertree Decompositions)

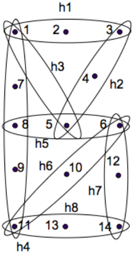
Motivation

- The structure of a large number of problems is more faithfully described by a hypergraph than by a graph
- Several NP complete problems become tractable if restricted to instances with acyclic hypergraphs
- An appropriate notion of hypergraph width should fulfil both of the following conditions
  1. Relevant hypergraph-based problems should be solvable in polynomial time for instances of bounded width
  2. For each constant k, one should be able to check in polynomial time whether a hypergraph is of width k, and, in the positive case, it should be possible to produce an associated decomposition of width k of the given hypergraph
- The hypertree decomposition is the most general method leading to large tractable classes of important problems such as constraint satisfaction problems or conjunctive queries

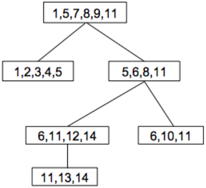
Generalized Hypertree Decomposition

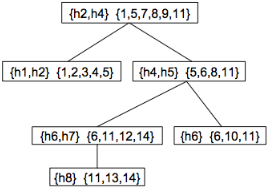
A generalized hypertree decomposition (GHD) of H is a tree decomposition of H with the following extension.

- GHD associates additionally to each node of the decomposition tree the set of hyperedges of H.
- The set of vertices associated to each node of the tree must be covered by the set of hyperedges associated to that node.
- The width of a generalized hypertree decomposition is the maximum number of hyperedges associated to a same node of the decomposition.



Tree decomposition





Generalized hypertree decomposition

Hypertree

Definition

A hypertree for a hypergraph $\mathcal{H} = (V(\mathcal{H}), H(\mathcal{H}))$ is a triple $\langle T, \chi, \lambda \rangle$, where $T = (N, E)$ is a rooted tree, and χ and λ are labeling functions which associate to each vertex $p \in N$ two sets

- $\chi(p) \subseteq V(\mathcal{H})$ and
- $\lambda(p) \subseteq H(\mathcal{H})$.

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

Hypertree Decomposition

Definition ([Gottlob et al.(2002)])

Let $\mathcal{H} = (V(\mathcal{H}), H(\mathcal{H}))$ be a hypergraph. A hypertree decomposition of $\mathcal{H}$ is a hypertree $\langle T, \chi, \lambda \rangle$ for $\mathcal{H}$ which satisfies all the following conditions:

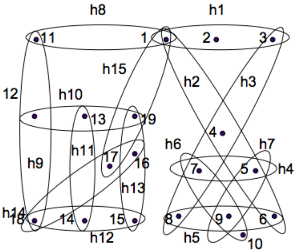
1. for each hyperedge $h \in H(\mathcal{H})$, there exists $p \in vertices(T)$ such that $vertices(h) \subseteq \chi_p$;
2. for each vertex $y \in V(\mathcal{H})$, the set $\{p \in vertices(T) \mid y \in \chi_p\}$ induces a (connected) subtree of T;
3. for each vertex $p \in vertices(T)$, $\chi_p \subseteq vertices(\lambda_p)$;
4. for each vertex $p \in vertices(T)$, $vertices(\lambda_p) \cap \chi(T_p) \subseteq \chi_p$.

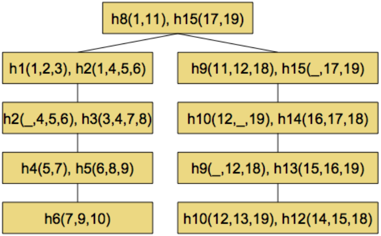
The width of the hypertree decomposition $\langle T, \chi, \lambda \rangle$ is $max_{p \in vertices(T)} |\lambda_p|$. The hypertree width, $hw(\mathcal{H})$, of $\mathcal{H}$ is the minimum width over all its hypertree decompositions.

Note: inclusion in Condition 4 is an equality, as Condition 3 implies the reverse inclusion!

Generalized Hypertree Decomposition

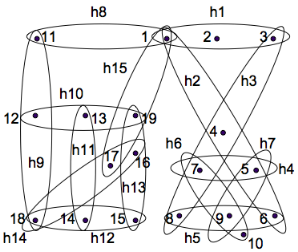
Generalized hypertree decomposition does not include condition 4) of hypertree decomposition.

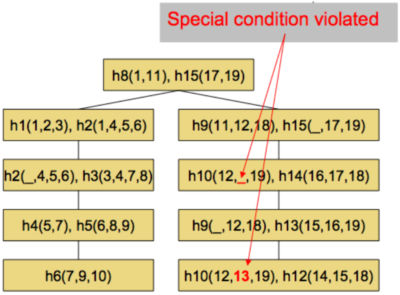




Generalized hypetree decomposition of width 2

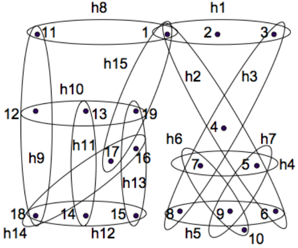
Generalized Hypertree Decomposition

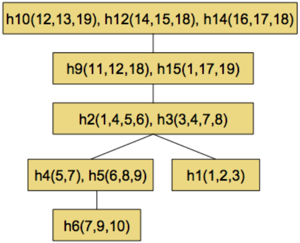




Generalized hypetree decomposition of width 2

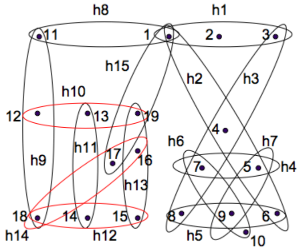
Hypertree Decomposition

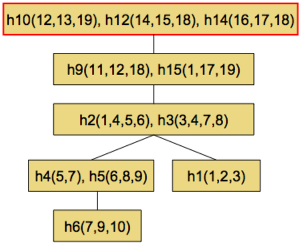




Hypertee decomposition of width 3

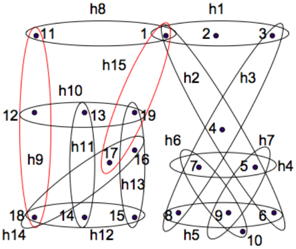
Hypertree Decomposition

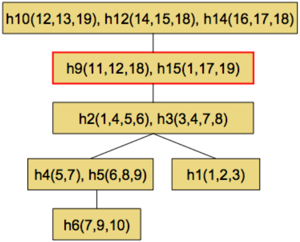




Hypertee decomposition of width 3

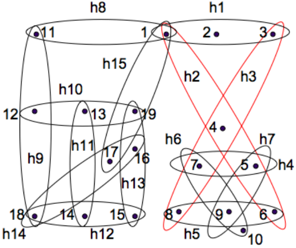
Hypertree Decomposition

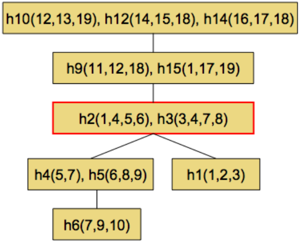




Hypertee decomposition of width 3

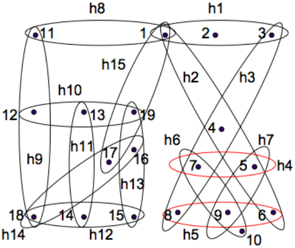
Hypertree Decomposition

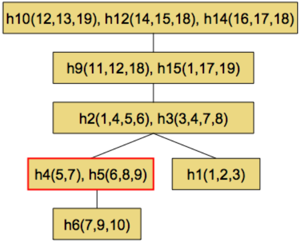




Hypertee decomposition of width 3

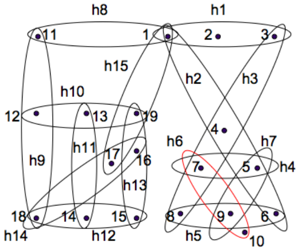
Hypertree Decomposition

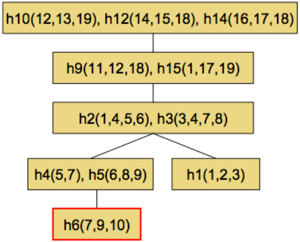




Hypertee decomposition of width 3

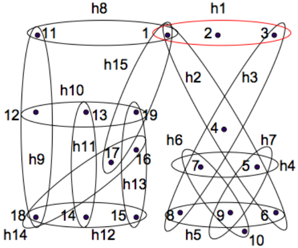
Hypertree Decomposition

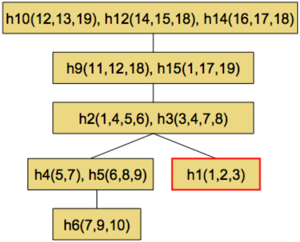




Hypertee decomposition of width 3

Hypertree Decomposition





Hypertee decomposition of width 3

Hypertree Width and CSPs

- The smaller the width of the obtained hypertree decompostion, the faster the corresponding CSP instance can be solved
- A CSP instance can be solved based on its hypertree decomposition as follows:
  - for each node t of the hypertree, all constraints in $\lambda(t)$ are "joined" into a new constraint over the variables in $\chi(t)$
  - for bounded width, i.e., for bounded cardinality of $\lambda(t)$, this yields a polynomial time reduction to an equivalent acyclic CSP instance

Algorithms for Generalized Hypertree Decomposition

- Methods based on tree decomposition
  - Generalized hypertree decomposition can be generated by algorithms for tree decomposition + Set Covering
- Hypertree decomposition based on hypergraph partitioning
- Exact methods
- Literature and benchmark instances for hypertree decomposition:
  `http://www.dbai.tuwien.ac.at/proj/hypertree/`
  `http://wwwinfo.deis.unical.it/~frank/Hypertrees/`

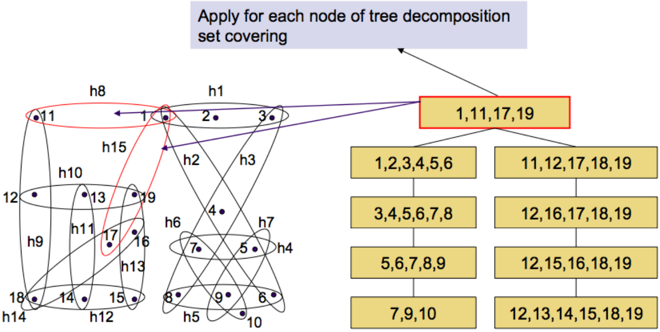
Constructing Generalized Hypertree Decomposition from Tree Decomposition

Recall, a hypertree decompostion can be devided into two parts

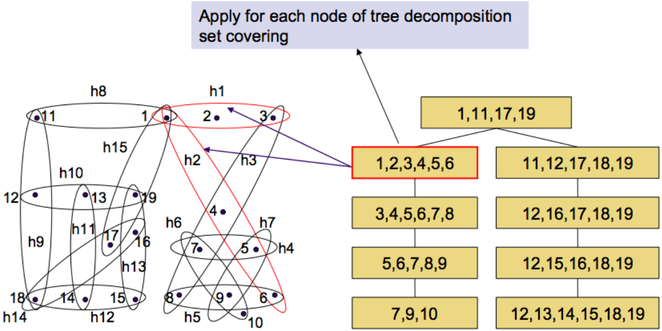
1. definition of a tree decomposition (T, χ)
2. introduction of λ such that $\chi(t) \subseteq \bigcup \lambda(t)$ for every node t.

χ-labels contain vertices of the hypergraph and λ-labels contain hyperedges, i.e., sets of vertices, of the hypergraph (covering vertices in $\chi(t)$ by hyperedges in $\lambda(t)$).

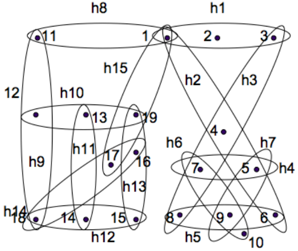
Constructing Generalized Hypertree Decomposition from Tree Decomposition ctd.

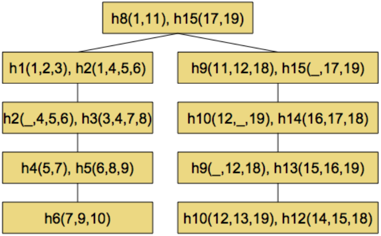


Constructing Generalized Hypertree Decomposition from Tree Decomposition ctd.



Generalized Hypertree Decomposition





Generalized hypetree decomposition of width 2

Hypertree Decomposition Based on Hypergraph Partitioning

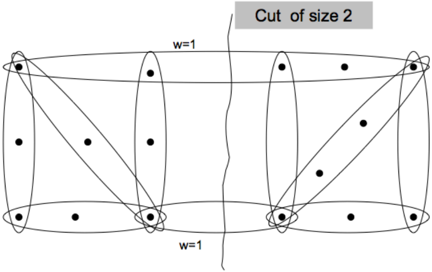
A method for generation of generalized hypertree decompositions based on recursive partitioning of the hypergraph [Dermaku et al.(2008)].

Hypergraph Partitioning

Given a hypergraph $\mathcal{H}(V, H)$ with weighted vertices and hyperedges.

- Find a partition of set V in two (or k) disjoint subsets such that the number of vertices in each set V_i is bounded, and the function defined over hyperedges is optimized.
- Most commonly used objective is to minimize the sum of the weights of hyperedges connecting two or more subsets.

Hypergraph Partitioning



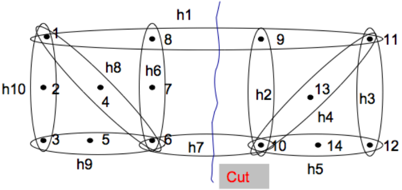
Hypergraph partitioning with constraint about the number of vertices in each partition is NP-Complete problem!

Generation of Hypertree Decomposition by Hypergraph Partitioning

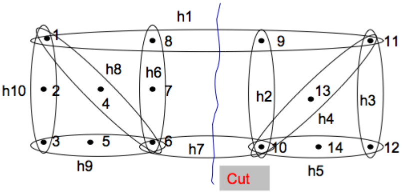
- Does recursive partitioning of hypergraph lead to "good" hypertree decomposition?
- Every cut in hypergraph partitioning can be considered as a node in a hypertree decomposition (called separator)
- Add a special hyperedge to each subgraph containing the vertices in the intersection between the subgraphs to enforce joint appearence in the χ-label of a later generated node
- Connectedness condition for variables should be ensured!
- How to evaluate a cut whose separator contains such hyperedges?
  - associate weights to hyperedges
  - weight 1 for all ordinary hyperedges
  - (W+) weight of special hyperedge: number of ordinary hyperedges needed to cover the vertice of the special hyperedge
  - other weighting schemes associate different weights to special hyperedges (always weight 1 or weight 2)
  - cut evaluates as the sum of weights of all hyperedges in the separator
- Nodes of hypertree are connected at the end of partitioning

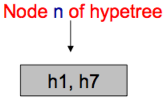
From Partitioning to Hypertree

All hyperedges have weight 1

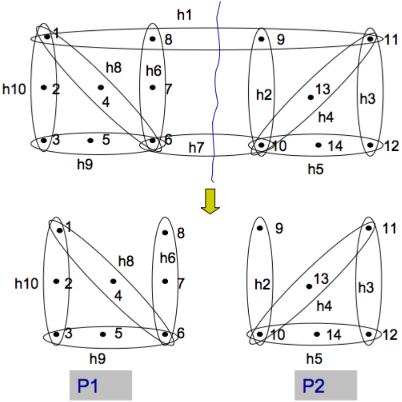


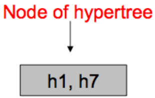
From Partitioning to Hypertree



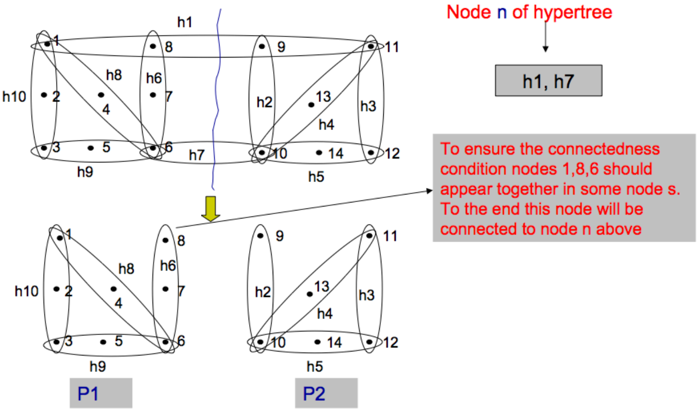


From Partitioning to Hypertree

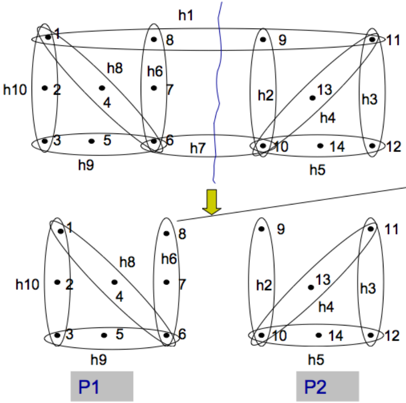




From Partitioning to Hypertree



From Partitioning to Hypertree



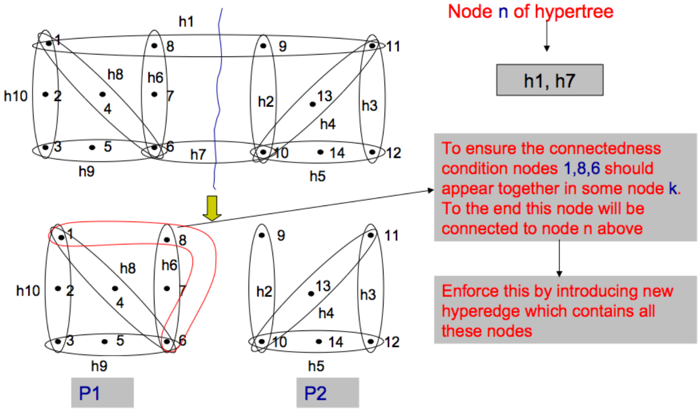
Node n of hypertree

h1, h7

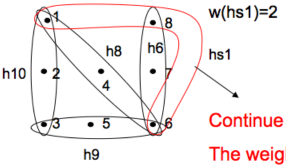
To ensure the connectedness condition nodes 1,8,6 should appear together in some node k. To the end this node will be connected to node n above

Enforce this by introducing new hyperedge which contains all these nodes

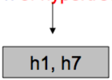
From Partitioning to Hypertree



From Partitioning to Hypertree



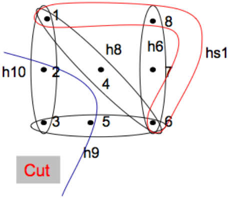
Node n of hypertree

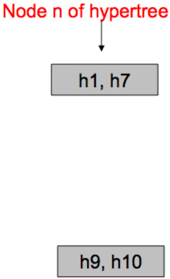


Continue recursively the partitioning

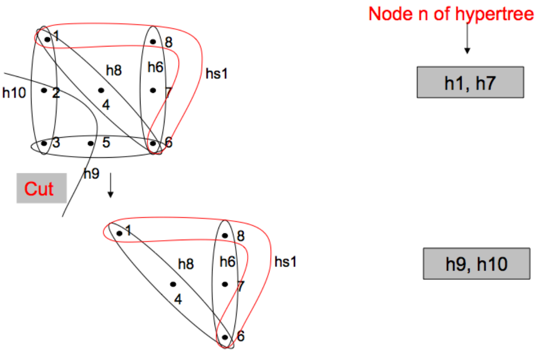
The weight of hyperedge hs1 is the sum of weights of hyperedges which cover this hyperedge: in this case w(hs1)=2

From Partitioning to Hypertree

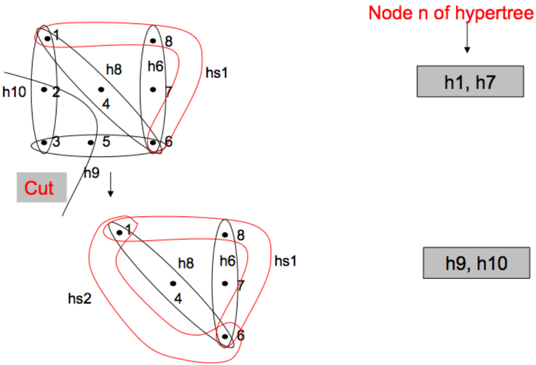




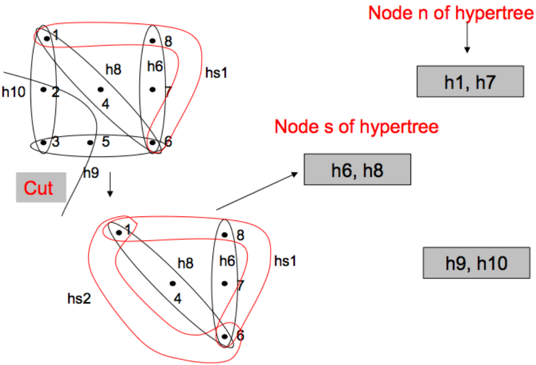
From Partitioning to Hypertree



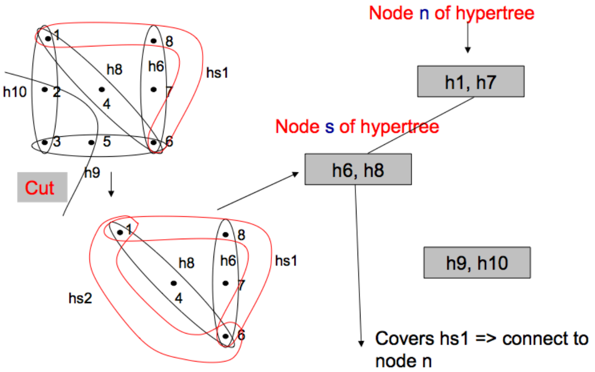
From Partitioning to Hypertree



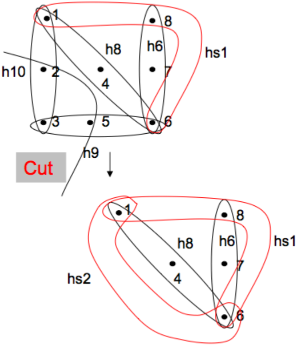
From Partitioning to Hypertree

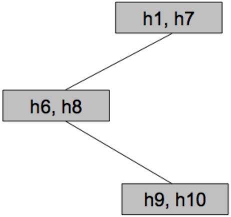


From Partitioning to Hypertree



From Partitioning to Hypertree





Summary

- Hypertree decomposition is a method leading to a large class of tractable problems such as CSP
- Computation of generalized hypertree decomposition based
  - on tree decompostion + Set Covering
  - hypergraph patitioning

References

Georg Gottlob, Nicola Leone, and Francesco Scarcello.
Hypertree decompositions and tractable queries, Journal of Computer and System Sciences, 64(3):579–627, 2002. ISSN 0022-0000.

Georg Gottlob, Nicola Leone, and Francesco Scarcello.
Hypertree Decompositions: A Survey, In J. Sgall, A. Pultr, and P. Kolman, editors, MFCS 2001, LNCS 2136, pages 37–57. Springer Berlin Heidelberg, 2001.

Artan Dermaku, Tobias Ganzow, Georg Gottlob, Ben McMahan, Nysret Musliu, and Marko Samer.
Heuristic methods for hypertree decomposition, In Alexander Gelbukh and EduardoF. Morales, editors, MICAI 2008: Advances in Artificial Intelligence, volume 5317 of LNCS, pages 1–11. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-88635-8.