Agenda

1. Introduction
2. Uninformed Search versus Informed Search (Best First Search, A* Search, Heuristics)
3. Constraint Satisfaction
4. Answer Set Programming (ASP)
5. Structural Decomposition Techniques (Tree/Hypertree Decomposition)
6. Local Search, Stochastic Hill Climbing, Simulated Annealing
7. Tabu Search
8. Evolutionary Algorithms/ Genetic Algorithms
9. Adversial Search and Game Playing
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 Decomposition

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

Let \( H = (V(H), E(H)) \) be a hypergraph, consisting of a nonempty set \( V(H) \) of vertices, and a set \( E(H) \) of subsets of \( V(H) \), the hyperedges of \( H \). A hypertree decomposition of \( H \) is a hypertree \( \langle T, \chi, \lambda \rangle \) for \( H \) which satisfies all the following conditions:

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

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

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

Special condition violated

Generalized hypertree decomposition of width 2
Hypertree Decomposition

Hypertree decomposition of width 3
Hypertree Decomposition

Hypertree decomposition of width 3
Hypertree Decomposition

Hypertree decomposition of width 3
Hypertree Decomposition

Hypertree decomposition of width 3
Hypertree Decomposition

Hypertree decomposition of width 3
Hypertree Decomposition

Hypertree decomposition of width 3
Hypertree Decomposition
Solving problems based on hypertree decomposition
\[ d(Y, P) \]

\[ r(Y, Z, U) \]

\[ s(Z, U, W) \]

\[ t(V, Z) \]
d: 3 8
   3 7
   5 7
   6 7

r(Y,Z,U)

s(Z,U,W)

r: 3 8 9
   9 3 8
   8 3 8
   3 8 4
   3 8 3
   8 9 4
   9 4 7

t(V,Z)

s: 3 8 9
   9 3 8
   8 3 8
   3 8 4
   3 8 3
   8 9 4
   9 4 7
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://www.dbai.tuwien.ac.at/proj/hypertree/)
  [http://wwwinfo.deis.unical.it/~frank/Hypertrees/](http://wwwinfo.deis.unical.it/~frank/Hypertrees/)
Constructing Generalized Hypertree Decomposition from Tree Decomposition

Apply for each node of tree decomposition set covering

TU Dresden, 20th May 2014
Constructing Generalized Hypertree Decomposition from Tree Decomposition

Apply for each node of tree decomposition set covering

TU Dresden, 20th May 2014 PSSAI slide 37 of 55
Generalized Hypertree Decomposition

Generalized hypertree 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 $H(V, E)$ with weighted vertices and hyperedges.

- Find a partitions 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 ore more subsets.
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.
- Nodes of hypertree are connected at the end of partitioning.
- Connectedness condition for variables should be ensured!
From Partitioning to Hypertree

All hyperedges have weight 1

Cut
From Partitioning to Hypertree
From Partitioning to Hypertree
Node \( n \) of hypertree

To ensure the connectedness condition, nodes 1, 8, 6 should appear together in some node \( s \). To the end, this node will be connected to node \( n \) above.
From Partitioning to Hypertree

Node $n$ of hypertree

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

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

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

Node $n$ of hypertree

$\{h_1, h_7\}$

$\{h_9, h_{10}\}$
From Partitioning to Hypertree

Node n of hypertree

Cut

h1, h7

h9, h10
From Partitioning to Hypertree

Node n of hypertree

h1, h7

h9, h10

Cut
From Partitioning to Hypertree
From Partitioning to Hypertree

Node n of hypertree

Node s of hypertree

Cut

Covers hs1 => connect to node n
From Partitioning to Hypertree