TECHNISCHE UNIVERSITÄT DRESDEN

Artificial Intelligence, Computational Logic

PROBLEM SOLVING AND SEARCH IN ARTIFICIAL INTELLIGENCE

Lecture 10 Tree Decompositions

Sarah Gaggl

Dresden, 7th 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)

Fixed-Parameter Tractability (FPT) – Motivation

Some Observations

- For intractable problems, computational costs often depend primarily on some problem parameters rather than on the mere size of the instances.
- Many hard problems become tractable if some problem parameter is fixed or bounded by a fixed constant.
- Typical parameters for graphs: treewidth and cliquewidth.
  - Meta-theorems allow for rather easy proofs of FPT results w.r.t. these parameters
  - Dedicated dynamic algorithms required for practical realization!

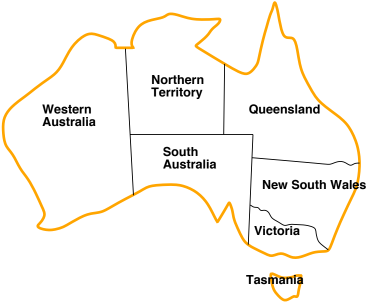
FPT is one branch in the area of Parameterized Complexity

- Downey & Fellows: Parameterized Complexity. Springer, 1999
- Flum & Grohe: Parameterized Complexity Theory. Springer, 2006
- Niedermeier: Invitation to Fixed-Parameter Algorithms. OUP, 2006

Introduction

- Many instances of constraint satisfaction problems can be solved in polynomial time if their treewidth (or hypertree width) is small.
- Solving of problems with bounded width includes two phases:
  - Generate a (hyper)tree decomposition with small width;
  - Solve a problem (based on generated decomposition) with a particular algorithm such as for example dynamic programming.
- Main idea: decomposing a problem into sub-problems of limited size allows to solve the whole problem more efficiently
- The efficiency of solving of problem based on its (hyper)tree decomposition depends on the width of (hyper)tree decomposition.
- It is of high importance to generate (hyper)tree decompositions with small width.

CSP: Map-Coloring



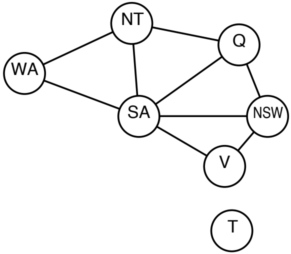
Variables WA, NT, Q, NSW, V, SA, T

Domains $D_i = \{red, green, blue\}$

Constraints: adjacent regions must have different colors e.g., $WA \neq NT$, or
$(WA, NT) \in \{(red, green), (red, blue), (green, red), (green, blue), \ldots\}$

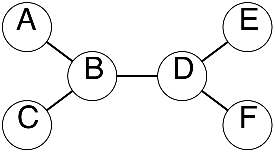
Constraint Graph

Binary CSP: each constraint relates at most two variables
Constraint graph: nodes are variables, arcs show constraints



General-purpose CSP algorithms use the graph structure
to speed up search. E.g., Tasmania is an independent sub-problem!

Tree-structured CSPs



Theorem

If the constraint graph has no loops, the CSP can be solved in $O(n\,d^2)$ time.

- Compare to general CSPs, where worst-case time is $O(d^n)$
- This property also applies to logical and probabilistic reasoning: an important example of the relation between syntactic restrictions and the complexity of reasoning.

CSP: SAT Problem

$(x_1 \lor x_2 \lor \neg x_3) \land (x_1 \lor \neg x_4 \lor x_5 \lor x_6) \land \cdots \land (x_3 \lor x_4 \lor x_7 \lor x_8) \ldots$

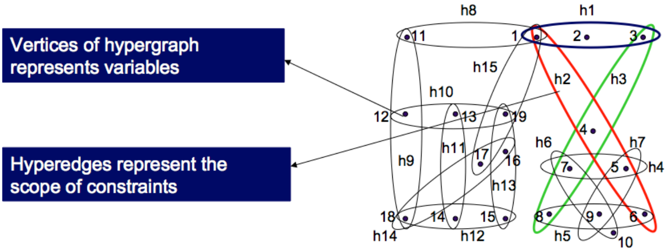
Possible CSP fomulation:

Variables $x_1, x_2, x_3, \ldots$

Domains $0, 1$

Constraints
- C1: $(x_1 \lor x_2 \lor \neg x_3) \rightarrow$ true
- C2: $(x_1 \lor \neg x_4 \lor x_5 \lor x_6) \rightarrow$ true
- $\ldots$

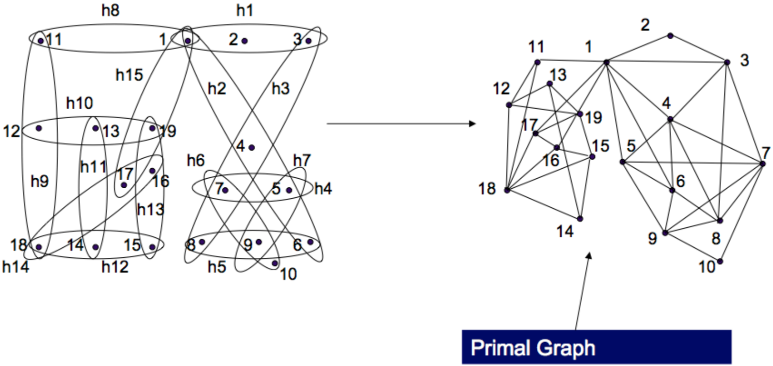
CSP and Hypergraph



$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_4 \vee x_5 \vee x_6) \wedge (x_3 \vee x_4 \vee x_7 \vee x_8) \ldots$

In general worst case complexity: $2^{\text{NumberOfVariables}} = 2^{19}$

Hypergraph and its Primal Graph



CSP and (Hyper)treewidth

- In general exponential worst case complexity.
- Can we solve this instance more efficiently (or in polynomial time)?
- Yes, if it has a small (hyper) treewidth!!!

Tree Decomposition

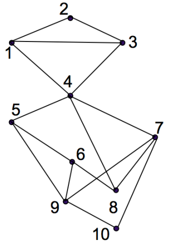
Definition

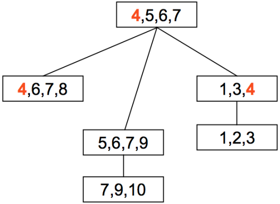
Tree Decomposition Let $G = (V, E)$ be a graph. A tree decomposition of G is a pair (T, χ), where $T = (I, F)$ is a tree with node set I and edge set F, and $\chi = \{\chi_i : i \in I\}$ is a family of subsets of V, one for each node of T, such that

1. $\bigcup_{i \in I} \chi_i = V$,
2. for every edge $(v, w) \in E$, there is an $i \in I$ with $v \in \chi_i$ and $w \in \chi_i$, and
3. for all $i, j, k \in I$, if j is on the path from i to k in T, then $\chi_i \cap \chi_k \subseteq \chi_j$.

The width of a tree decomposition is $max_{i \in I} |\chi_i| - 1$.
The treewidth of a graph G, denoted by $tw(G)$, is the minimum width over all possible tree decompositions of G.

Tree Decomposition - Example





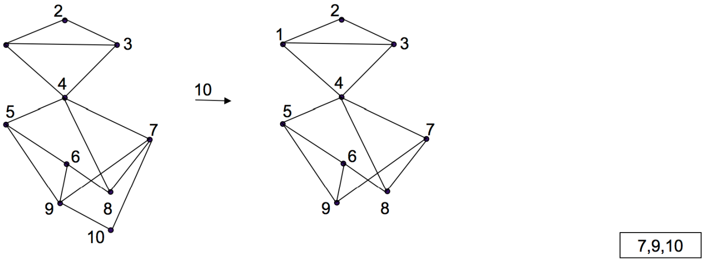
All pairs of vertices that are connected appear in some node of the tree.
Connectedness condition for vertices

Elimination Ordering

- For the given problem find the tree decomposition with minimal width -> NP hard.
- There exists a perfect elimination ordering which produces tree decomposition with treewidth (smallest width).
- Tree decomposition problem $\rightarrow$ search for the best elimination ordering of vertices!
- Permutation Problem $\rightarrow$ similar to TSP.

Possible elimination ordering for graph in previous slide:
$10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

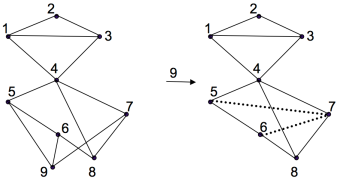
Perfect Elimination Ordering



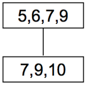
Vertex 10 is eliminated from the graph. All neighbors of 10 are connected and a tree node is created that contains vertex 10 and its neighbors.

Elimination ordering: $10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

Perfect Elimination Ordering ctd.



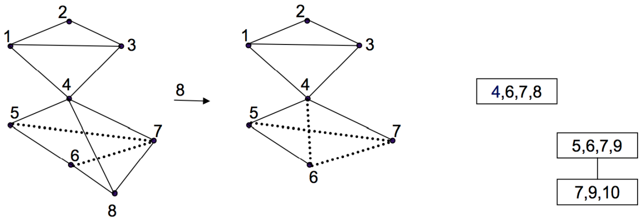
The tree decomposition node with vertices [7,9,10] is connected with the tree decomposition node which is created when the next vertex which appears in [7,9,10] is eliminated (in this case vertex 9)



Vertex 9 is eliminated from the graph. All neighbors of vertex 9 are connected and a new tree node is created.

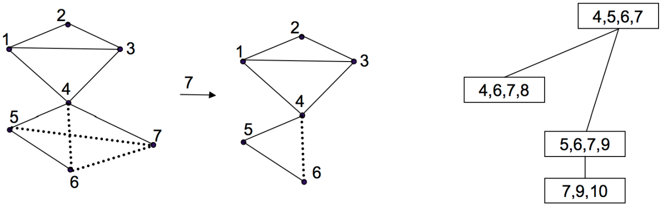
Elimination ordering: $10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

Perfect Elimination Ordering ctd.



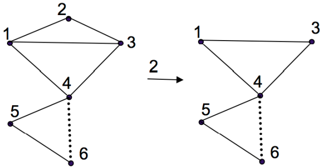
Elimination ordering: $10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

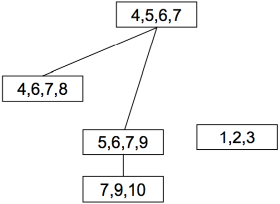
Perfect Elimination Ordering ctd.



Elimination ordering: $10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

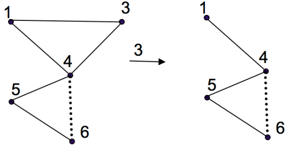
Perfect Elimination Ordering ctd.

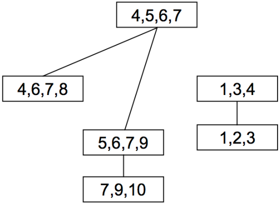




Elimination ordering: $10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

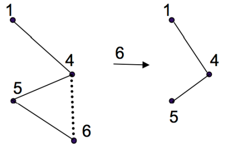
Perfect Elimination Ordering ctd.

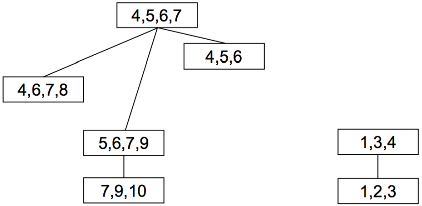




Elimination ordering: $10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

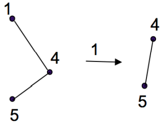
Perfect Elimination Ordering ctd.

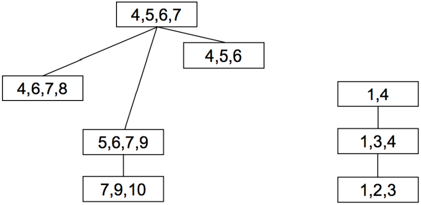




Elimination ordering: $10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

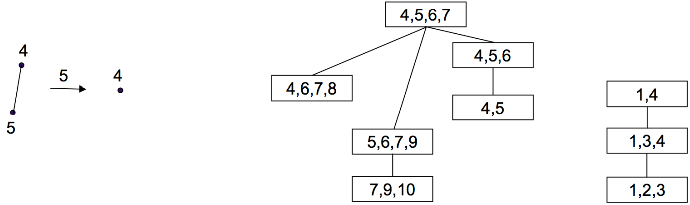
Perfect Elimination Ordering ctd.





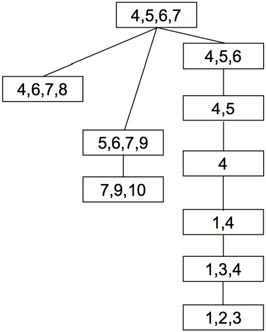
Elimination ordering: $10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

Perfect Elimination Ordering ctd.



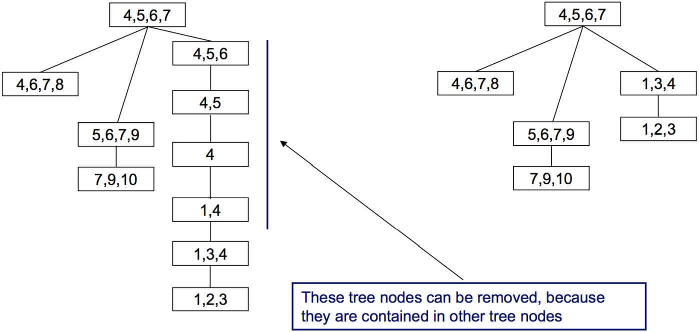
Elimination ordering: $10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

Perfect Elimination Ordering ctd.



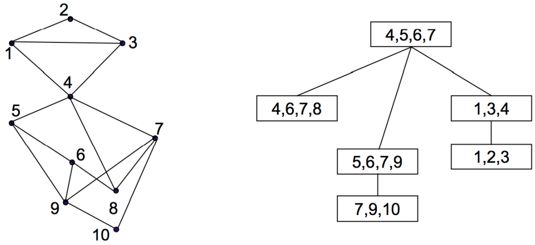
Elimination ordering: $10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

Perfect Elimination Ordering ctd.



Elimination ordering: $10, 9, 8, 7, 2, 3, 6, 1, 5, 4$

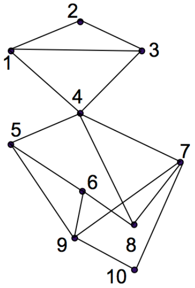
Tree Decomposition of a Graph



Width: max(vertices in tree node)$-1 = 3$.

Treewidth: minimal width over all possible tree decomposition.

Example (Another Tree Decomposition)



Elimination ordering: $4, 3, 10, 5, 6, 7, 1, 2, 9, 8$

Group-Work! What is the worst case complexity to solve the CSP on the constructed TD?

Bounded Treewidth for CSP

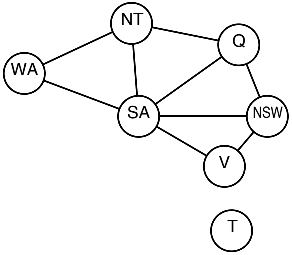
If a graph has treewidth k, and we are given the corresponding tree decomposition, then the problem can be solved in $O(nd^{k+1})$ time.

n - number of variables,
d - maximum domain size of any variable in the CSP.

But, finding the decomposition with minimal treewidth is NP-hard.

$\rightarrow$ Heuristic methods work well in practice!

Solving Problems based on TD



- Naive approach: try all possibilities d^n combinations
- Make tree decomposition and solve each subproblem independently (blackboard)
- If one subproblem has no solution ⇒ the whole problem has no solution
- Elimination ordering: V, NSW, Q, NT, T, WA, SA

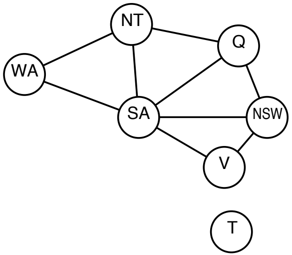
Algorithms for Finding Good Elimination Ordering

- Exact Methods
  - Branch and bound
  - $A*$
- (Meta)Heuristic Methods
  - Maximum Cardinality Search (MCS)
  - Min-Fill Heuristic
  - Tabu Search
  - Genetic Algorithms
  - Iterated Local Search

Maximum Cardinality Search (MCS)

1. Select a random vertex in graph to be the first in elimination ordering
2. Pick next vertex that has highest connectivity with vertices previously selected in elimination ordering (ties are broken randomly)
3. Repeat Step 2 until whole ordering is complete

Example (Graph Coloring)



On blackboard!

Min-Fill Heuristic

1. Select vertex which adds smallest number of edges when eliminated (ties are broken randomly) to be first vertex in elimination ordering
2. Pick next vertex that adds the minimum number of edges when eliminated from graph
3. Repeat Step 2 until whole ordering is constructed

Note

When the vertex is eliminated from graph, all its neighbors are connected (new edges are inserted in the graph)

Tabu Search

Moves:

- Swap to nodes in the elimination ordering
- Neighborhood: All possible solutions that can be obtained with swap of two vertices
- Tabu list: moved nodes are made tabu for several iterations (Diversification of search)
- Aspiration criterion: override tabu if solution is outstanding
- Frequency-based memory

Genetic Algorithms

- Population of randomly created individuals
- Tournament selection selects an individual by randomly choosing a group of several individuals from former population
- Individual of highest fitness (smallest width) within group is selected for next population
- Applied until enough individuals have entered next population

Crossover Operators

Order Crossover (OX)

- Selects crossover area within the parents by randomly selecting two positions within the ordering
- Elements in crossover area of first parent are copied to offspring
- Starting at end of crossover area all elements outside the area are inserted in same order in which they occur in second parent

1	2	3	4	5	6	7	8
2	4	6	8	7	5	3	1

$\Rightarrow$

8	7	3	4	5	1	2	6
4	5	6	8	7	1	2	3

Crossover Operators ctd.

Order-based Crossover (OBX)

- Selects at random several positions in the parent orderings by tossing a coin for each position
- Elements of first parent at these positions are inserted in first child in the order of the second parent
- Elements of second parent at these positions are inserted in second child in the order of the first parent

1	2	3	4	5	6	7	8
2	4	6	8	7	5	3	1

⇒

1	2	3	4	6	5	7	8
2	4	3	8	7	5	6	1

Mutation Operators

Exchange Mutation Operator (EM)

Randomly selects two elements and exchanges them.

1	2	3	4	5	6	7	8	$\Rightarrow$	1	2	6	4	5	3	7	8

Mutation Operators

Exchange Mutation Operator (EM)

Randomly selects two elements and exchanges them.

1	2	3	4	5	6	7	8

$\Rightarrow$

1	2	6	4	5	3	7	8

Insertion Mutation Operator (ISM)

Randomly chooses an element and moves it to randomly selected position.

1	2	3	4	5	6	7	8

$\Rightarrow$

1	2	4	5	6	7	3	8

Which Algorithm to Use?

- If width is not critical then MCS or Min-fill
- For exact solutions and small examples: Branch and Bound or $A*$
- For longer examples and better width: MCS, Min-Fill, Iterated local search or GA

Summary

- Problems with small treewidth are solvable in polynomial time (if treedecomposition is given)
- Idea of decomposing a problem in smaller sub-problems to solve them more efficiently
- Width of treedecomposition depends on the elimination ordering
- Finding treewidth is NP hard
- Tree decomposition problem $\Rightarrow$ search for the best elimination ordering of vertices!
  - Branch and bound
  - $A*$
  - Maximum Cardinality Search (MCS)
  - Min-Fill Heuristic
  - Tabu Search
  - Genetic Algorithms
  - Iterated Local Search
- Literature and Benchmark Instances for tree decomposition: TreewidthLIB `http://www.staff.science.uu.nl/~bodla101/treewidthlib/index.php`

References

Thomas Hammerl, Nysret Musliu and Werner Schafhauser.
Metaheuristic Algorithms and Tree Decomposition, Handbook of Computational Intelligence, pp 1255–1270, Springer, 2015.

Hans L. Bodlaender, Arie M.C.A. Koster.
Treewidth computations I. Upper bounds, Comput. 208(2): 259–275, 2010.