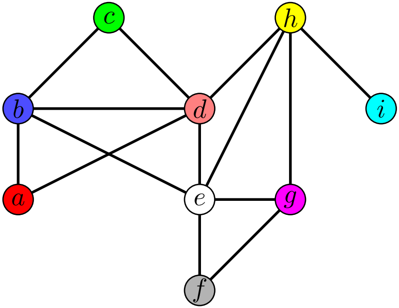
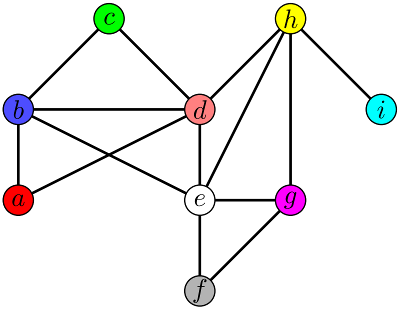
Tree Decompositions

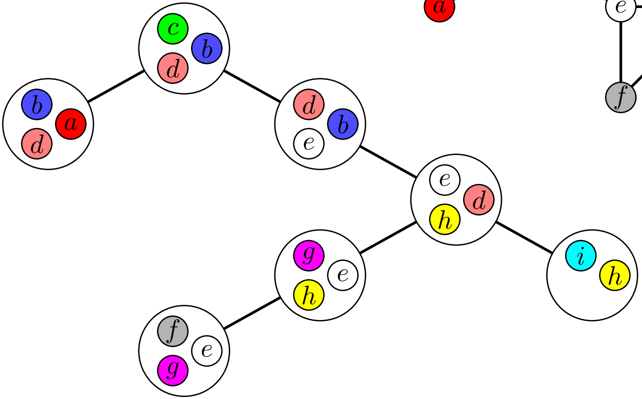
A tree decomposition of a graph G is a tree, whose nodes are called bags. Every bag is a set of nodes from G.



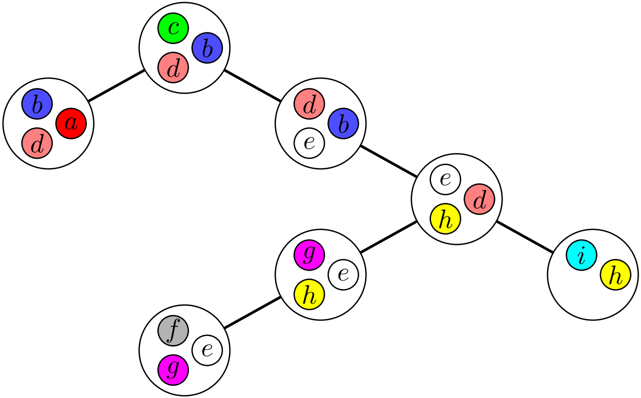
- Any node and any edge from G ist contained in at least one bag.
- A node contained in two bags A, B must be contained in any bag between A and B.

Tree Decompositions





Tree Decompositions and Treewidth

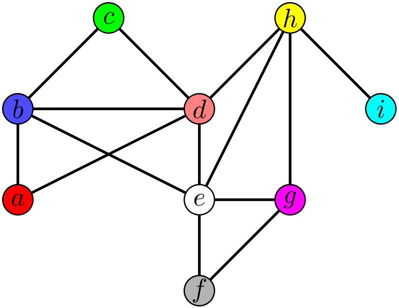


The width of a tree decompostion is the size of the largest bag minus 1.

⇒ Here, the treewidth is 2.

Tree Decompositions and Treewidth

Alternative definition:



The treewidth of G is the minimum number of cops, needed to catch a robber in G, minus 1.

Tree Decompositions and Treewidth

Given a tree decomposition of G with width w, many optimization problems on G can be solved in time $c^w \cdot poly(n)$ using dynamic programming on the tree decomposition.

Many problems can be solved fast, if a tree decomposition of small width can be found.

General Result

Any problem with the following properties is fixed parameter tractable:

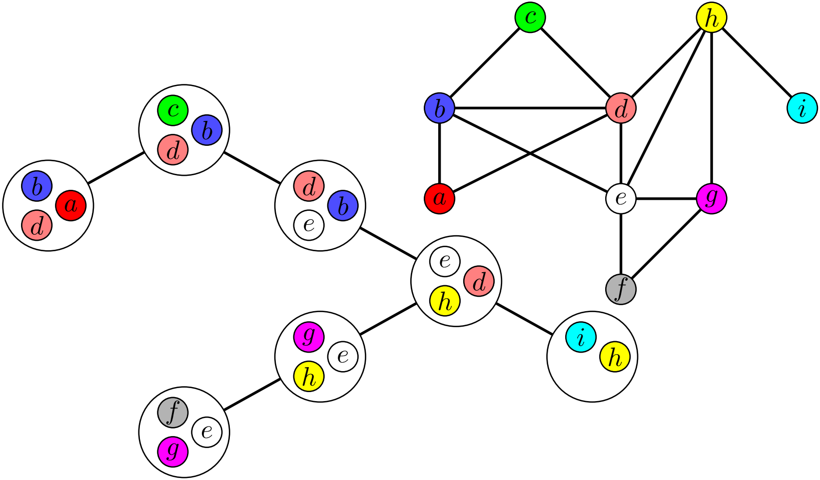
- Let $G = (V, E)$ a planar graph and k a number. Question: Does there exist some $S \subseteq V$ of size k with a certain property (e.g. S is a vertex cover)?
- There is a constant c such that the distance between any node and S is bounded by c.
- Given a tree decomposition of width w, the problem can can be solved in $f(w)n^{O(1)}$ steps.

Special cases are Vertex Cover, Independent Set, and Dominating Set.

Proof Idea

- Since any node is at most c steps away from a node in S, there is no path of length more than $2c|S|$.
- Hence, the diameter is $O(k)$, if there exists some S of size k.
- The treewidth of a planar graph with diameter d is at most $3d$ (without proof).
- If the diameter is larger than $2ck$, the output is no.
- Otherwise, we obtain a tree decomposition of width $6ck$ and can use it to solve the problem.

Dynamic Programming on a Tree Decomposition

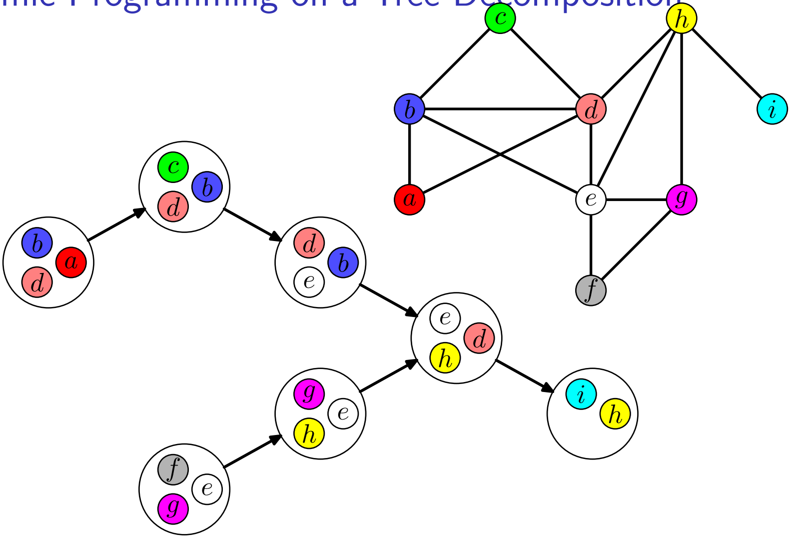


Dynamic Programming on a Tree Decomposition

General approach:

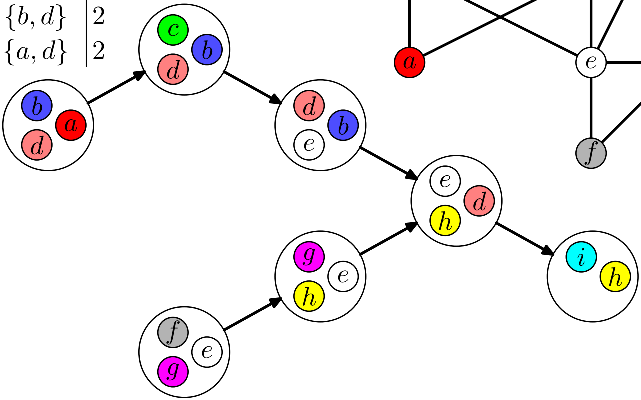
- The tree decomposition is transfered into a rooted tree: An arbitrary node becomes the root. Children point to their parents.
- A bag represents the subgraph induced by its children.
- For any bag a table is calculated, showing the optimal solutions for its subgraphs.
- The children's tables are calculated first.

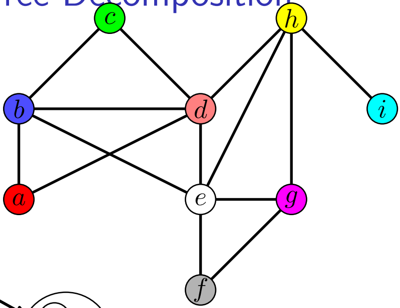
Dynamic Programming on a Tree Decomposition



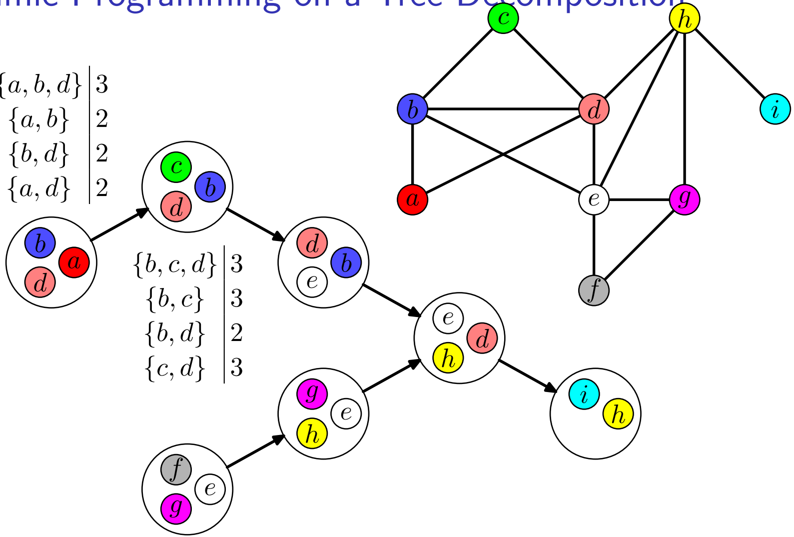
Dynamic Programming on a Tree Decomposition

$\{a,b,d\}$	3
$\{a,b\}$	2
$\{b,d\}$	2
$\{a,d\}$	2





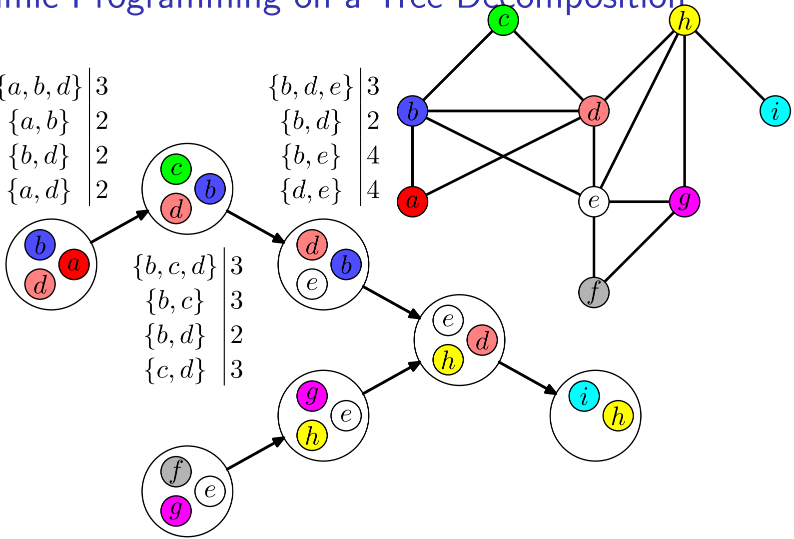
Dynamic Programming on a Tree Decomposition



$\{a,b,d\}$	3
$\{a,b\}$	2
$\{b,d\}$	2
$\{a,d\}$	2

$\{b,c,d\}$	3
$\{b,c\}$	3
$\{b,d\}$	2
$\{c,d\}$	3

Dynamic Programming on a Tree Decomposition

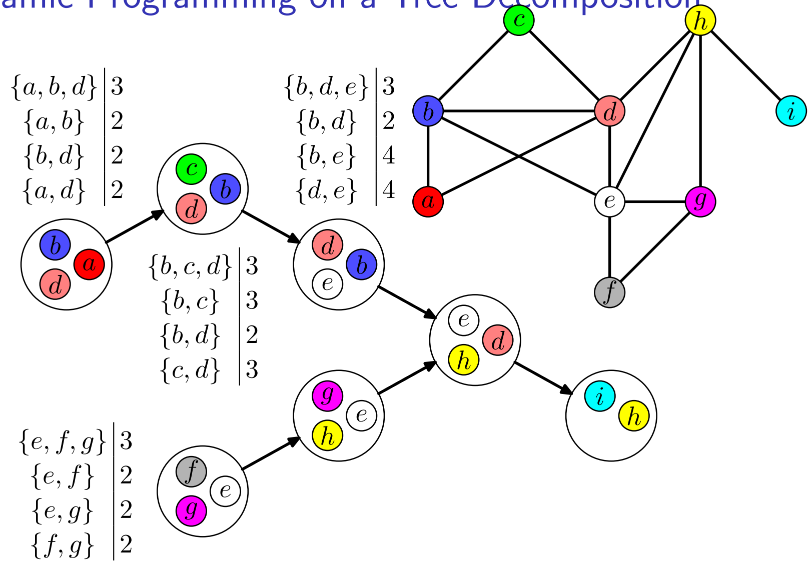


$\{a,b,d\}$	3
$\{a,b\}$	2
$\{b,d\}$	2
$\{a,d\}$	2

$\{b,c,d\}$	3
$\{b,c\}$	3
$\{b,d\}$	2
$\{c,d\}$	3

$\{b,d,e\}$	3
$\{b,d\}$	2
$\{b,e\}$	4
$\{d,e\}$	4

Dynamic Programming on a Tree Decomposition



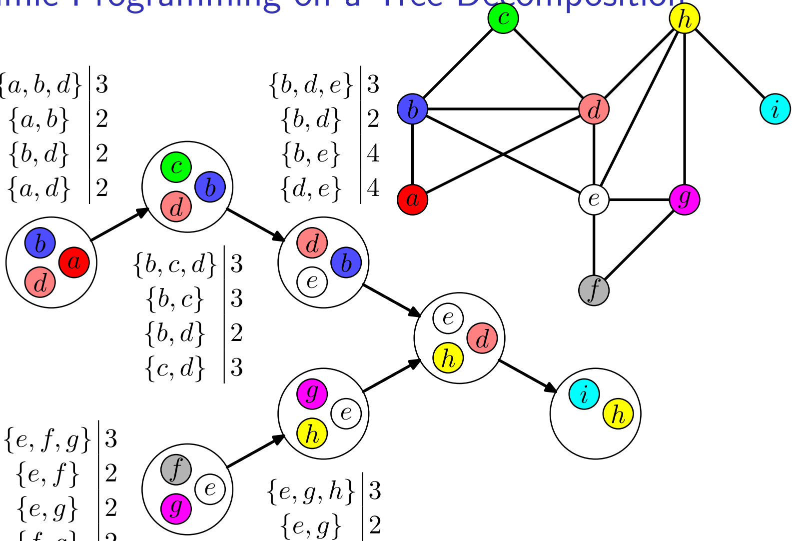
$\{a,b,d\}$	3
$\{a,b\}$	2
$\{b,d\}$	2
$\{a,d\}$	2

$\{b,d,e\}$	3
$\{b,d\}$	2
$\{b,e\}$	4
$\{d,e\}$	4

$\{b,c,d\}$	3
$\{b,c\}$	3
$\{b,d\}$	2
$\{c,d\}$	3

$\{e,f,g\}$	3
$\{e,f\}$	2
$\{e,g\}$	2
$\{f,g\}$	2

Dynamic Programming on a Tree Decomposition



$\{a, b, d\}$	3
$\{a, b\}$	2
$\{b, d\}$	2
$\{a, d\}$	2

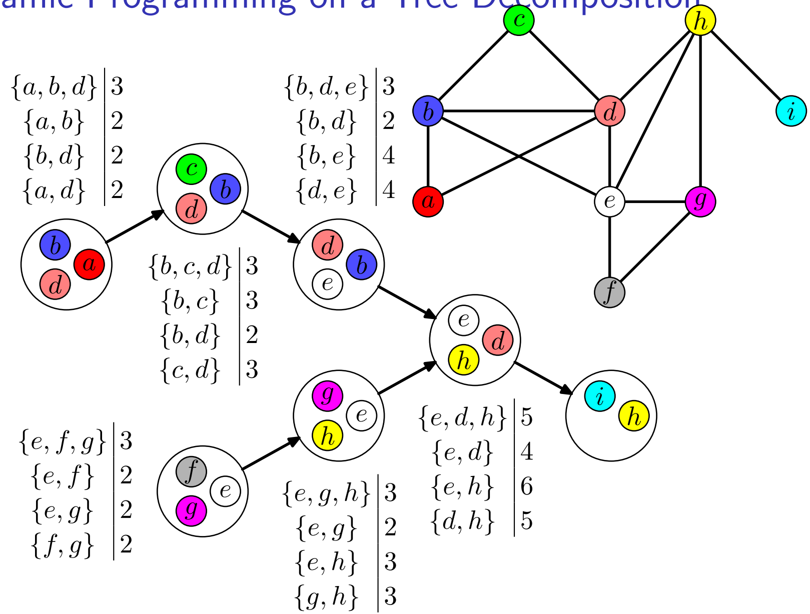
$\{b, c, d\}$	3
$\{b, c\}$	3
$\{b, d\}$	2
$\{c, d\}$	3

$\{b, d, e\}$	3
$\{b, d\}$	2
$\{b, e\}$	4
$\{d, e\}$	4

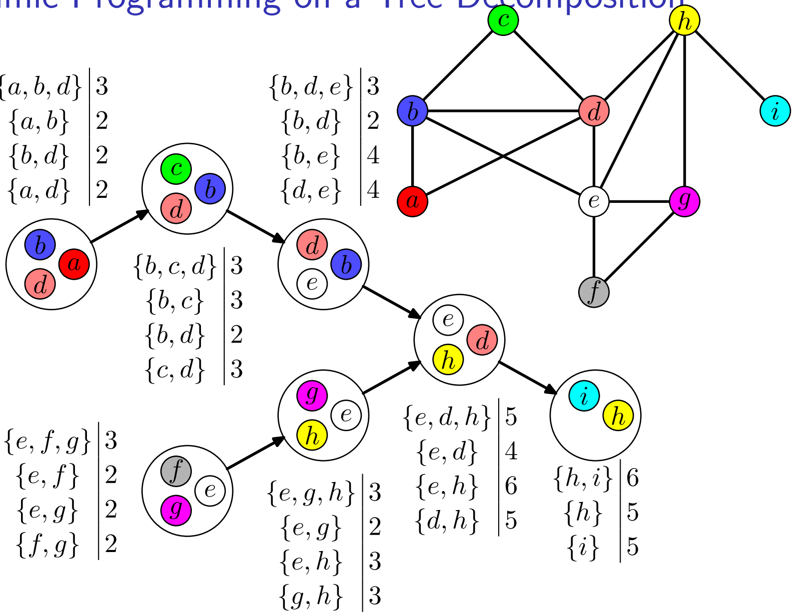
$\{e, f, g\}$	3
$\{e, f\}$	2
$\{e, g\}$	2
$\{f, g\}$	2

$\{e, g, h\}$	3
$\{e, g\}$	2
$\{e, h\}$	3
$\{g, h\}$	3

Dynamic Programming on a Tree Decomposition



Dynamic Programming on a Tree Decomposition



$\{a,b,d\}$	3
$\{a,b\}$	2
$\{b,d\}$	2
$\{a,d\}$	2

$\{b,c,d\}$	3
$\{b,c\}$	3
$\{b,d\}$	2
$\{c,d\}$	3

$\{b,d,e\}$	3
$\{b,d\}$	2
$\{b,e\}$	4
$\{d,e\}$	4

$\{e,d,h\}$	5
$\{e,d\}$	4
$\{e,h\}$	6
$\{d,h\}$	5

$\{e,f,g\}$	3
$\{e,f\}$	2
$\{e,g\}$	2
$\{f,g\}$	2

$\{e,g,h\}$	3
$\{e,g\}$	2
$\{e,h\}$	3
$\{g,h\}$	3

$\{h,i\}$	6
$\{h\}$	5
$\{i\}$	5

Extended Monadic Second Order Graph Theory

We introduce a new logic called MSO-logic.

This logic contains variables for nodes, edges, sets of nodes, and sets of edges.

There are the quantifiers $\exists$ and $\forall$ and operators $\wedge$, $\vee$ and $\neg$.

Furthermore, the following relations are included:

$$u \in U, \ d \in D, \ inc(d, u), \ adj(u, v), \ \cdot = \cdot$$

where u, v are node variables, d is an edge variable, U is a node set variable, and D is an edge set variable.

Extended Monadic Second Order Graph Theory

A graph can either satisfy a formula or not. This allows for a description of graph classes by formulas.

Example

Which graphs satisfy the following formula:

$$\exists u \exists v \exists w (adj(u, v) \wedge adj(u, w) \wedge adj(v, w))$$

Is there a formula describing bipartite graphs ?

Courcelle's Theorem

Theorem

Let $\mathcal{G}$ be a graph class, described by a formula in MSO-logic. The following proplem is fixed parameter tractable:

Input: Graph G with treewidth k
Parameter: k
Question: Does G belong to $\mathcal{G}$

Proof

difficult. . .

Courcelle's Theorem

Theorem

Let $\mathcal{G}$ be a graph class, described by a formula in MSO-logic. The following proplem is fixed parameter tractable:

Input: Graph G with treewidth k
Parameter: k
Question: Does G belong to $\mathcal{G}$

Proof

difficult. . .

Courcelle's Theorem

We could solve Vertex Cover parameterized by treewidth.

Using Courcelle's Theorem, the size of the formula depends on the size of the vertex cover, we are searching:

$$\exists v_1 \dots \exists v_k \forall e (inc(e, v_1) \vee \cdots \vee inc(e, v_k))$$

Why is this a problem?

Courcelle's Theorem (Extension)

We extend the MSO-logic:

We allow the following new quantifier:

$$\min U\ \forall e \exists u (u \in U \land inc(e,u))$$

Whenever the treewidth is bounded, a minimal set of nodes U, satisfying an arbitrary MSO formula $F(U)$, can be calculated in polynomial time.

Courcelle's Theorem (Extension)

Let G be a graph with vertex labels from $\{1, \dots, c\}$. The corresponding sets of nodes are $\{V_1, \dots, V_c\}$.

We can express inclusion in V_i.

Example

$$\max U \subseteq V_1 \forall x \in V_2 \exists y (adj(x, y) \vee y \notin U)$$

(Is there a set U of red nodes such that any blue node has a neighbor not belonging to U.)

This problem is called Red Blue Nonblocker.