Treewidth

Algorithms and Networks

Universiteit Utrecht

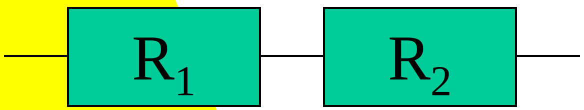
Overview

- Historic introduction: Series parallel graphs
- Dynamic programming on trees
- Dynamic programming on series parallel graphs
- Treewidth
- Dynamic programming on graphs of small treewidth
- Finding tree decompositions

Computing the Resistance With the Laws of Ohm

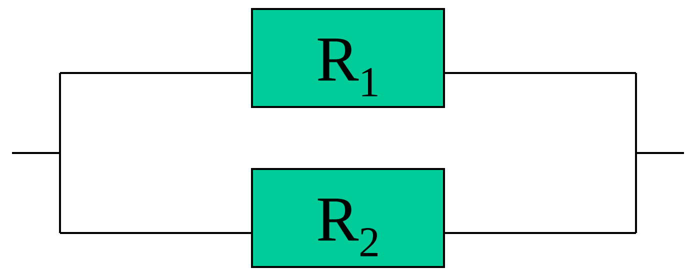
1789-1854

$$R = R_1 + R_2$$



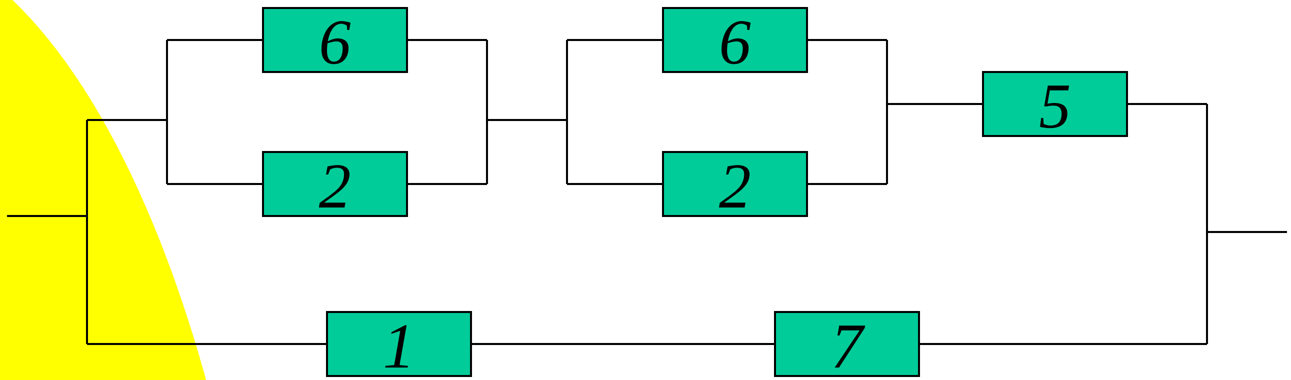
*Two resistors in **series***

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$



*Two resistors in **parallel***

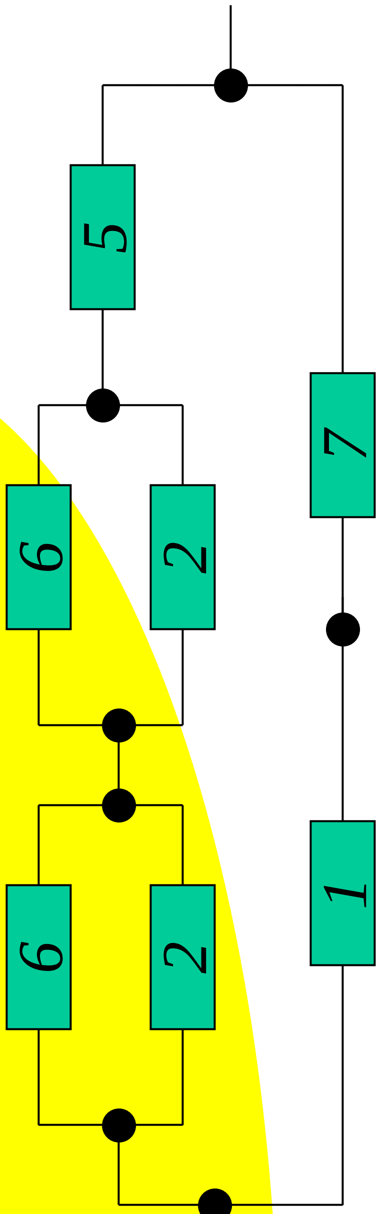
Repeated use of the rules

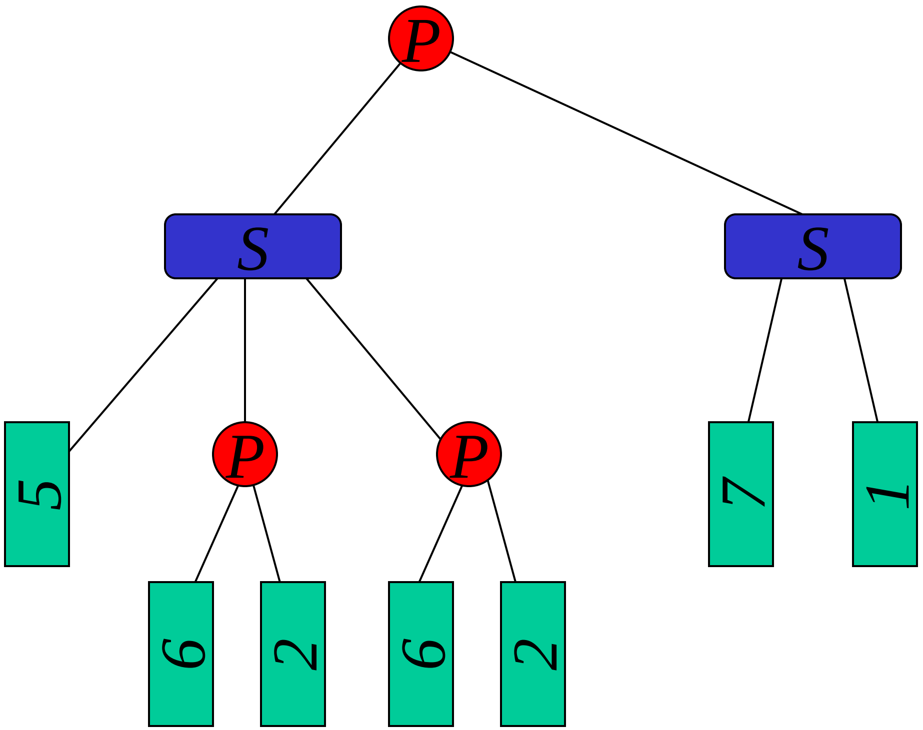


Has resistance 4

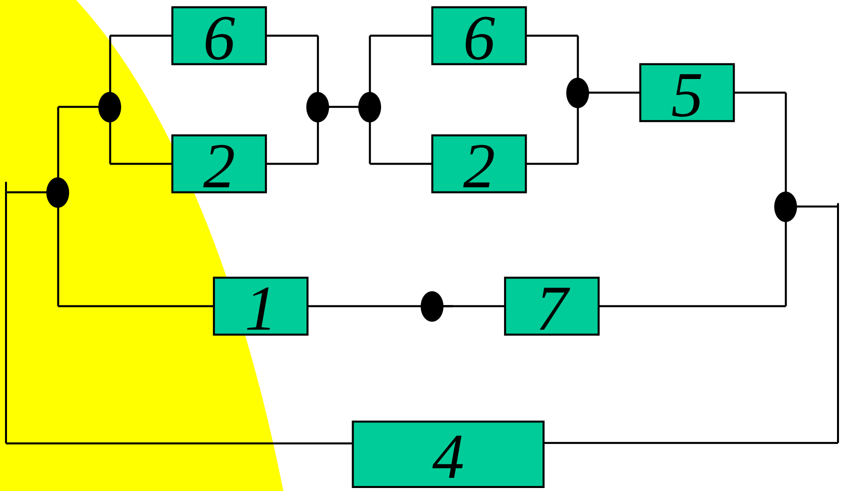
1/6 + 1/2 = 1/(1.5)
1.5 + 1.5 + 5 = 8
1 + 7 = 8
1/8 + 1/8 = 1/4

A tree structure





Carry on!



$\frac{1}{4} + \frac{1}{4} = \frac{1}{2}$

- Internal structure of graph can be forgotten once we know essential information about it!

Using tree structures for solving hard problems on graphs 1

- Network is ‘*series parallel graph*’
- 196*, 197*: many problems that are *hard* for general graphs are *easy* for
  - Trees
  - Series parallel graphs
- Many well-known problems

e.g.: NP-complete

Linear / polynomial time computable

Weighted Independent Set

- Independent set: set of vertices that are pair wise non-adjacent.
- Weighted independent set
  – Given: Graph G=(V,E), weight w(*v*) for each vertex *v*.
  – Question: What is the maximum total weight of an independent set in G?
- NP-complete

Weighted Independent Set on Trees

- On trees, this problem can be solved in linear time with dynamic programming.
- Choose root r. For each v, T(v) is subtree with v as root.
- Write

  A(v) = maximum weight of independent set S in T(v)

  B(v) = maximum weight of independent set S in T(v), such that $v \notin$ S.

Recursive formulations

- If v is a leaf:
  - $A(v) = w(v)$
  - $B(v) = 0$
- If v has children $x_1, \ldots, x_r$:

  $A(v) = \max\{ w(v) + B(x_1) + \ldots + B(x_r),$
  $A(x_1) + \ldots A(x_r) \}$

  $B(v) = A(x_1) + \ldots A(x_r)$

Linear time algorithm

- Compute A(v) and B(v) for each v, bottom-up.
  - E.g., in postorder
- Constructing corresponding sets can also be done in linear time.

Second example: Weighted dominating set

- A set of vertices S is ***dominating***, if each vertex in G belongs to S or is adjacent to a vertex in S.
- Problem: given a graph G with vertex weights, what is the minimum total weight of a dominating set in G?
- Again, NP-complete, but linear time on trees.

Subproblems

- C(v) = minimum weight of dominating set S of T(v)
- D(v) = minimum weight of dominating set S of T(v) with $v \in S$.
- E(v) = minimum weight of a set S of T(v) that dominates all vertices, except possibly v.

Recursive formulations

- If v is a leaf, …
- If v has children $x_1, \ldots, x_r$:
  - C(v) = the minimum of:
    - w(v) + E(x_1) + … + E(x_r)
    - C(x_1) + … + C(x_{i-1}) + D(x_i) + C(x_{i+1}) + … + C(x_r), over all i, $1 \leq i \leq r$.
  - D(v) = w(v) + E(x_1) + … + E(x_r)
  - E(v) = min { w(v) + E(x_1) + … + E(x_r), C(x_1) + … + C(x_r) }

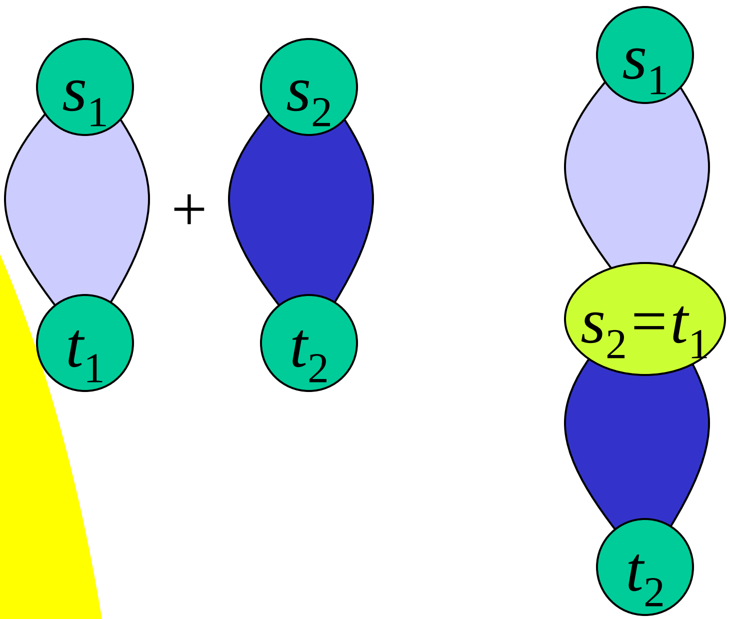
Gives again a linear time algorithm

- Compute bottom up (e.g., postorder), and use another type of dynamic programming for the values C(v).
- Constructing sets can also be done in linear time

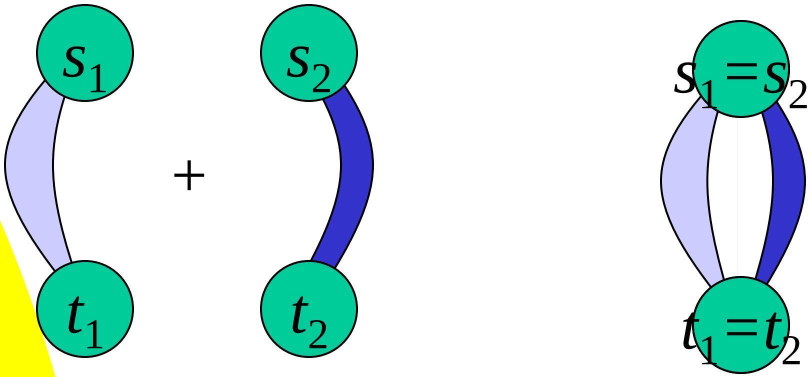
Generalizing to series parallel graphs

- A 2-terminal graph is a graph G=(V,E) with two special vertices *s* and *t*, its *terminals*.
- A 2-terminal (multi)-graph is **series parallel**, when it is:
  - A single edge (*s*,*t*).
  - Obtained by series composition of 2 series parallel graphs
  - Obtained by parallel composition of 2 series parallel graphs

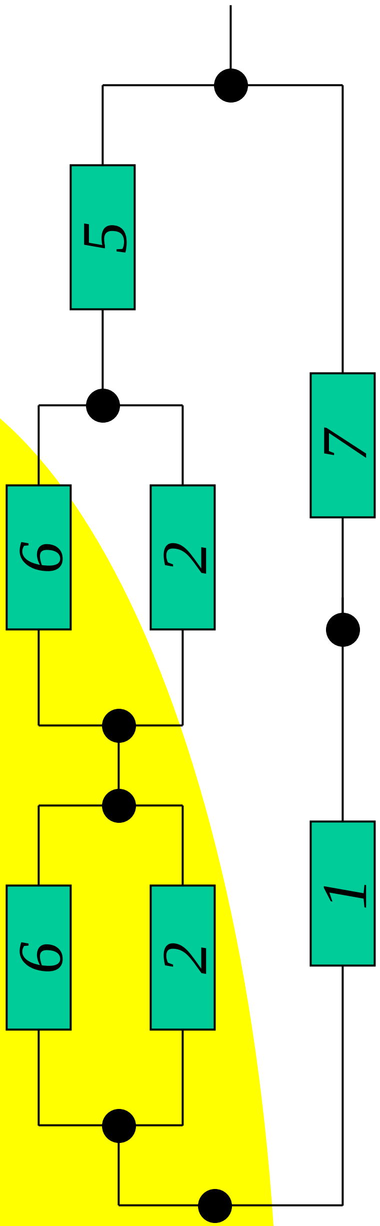
Series composition

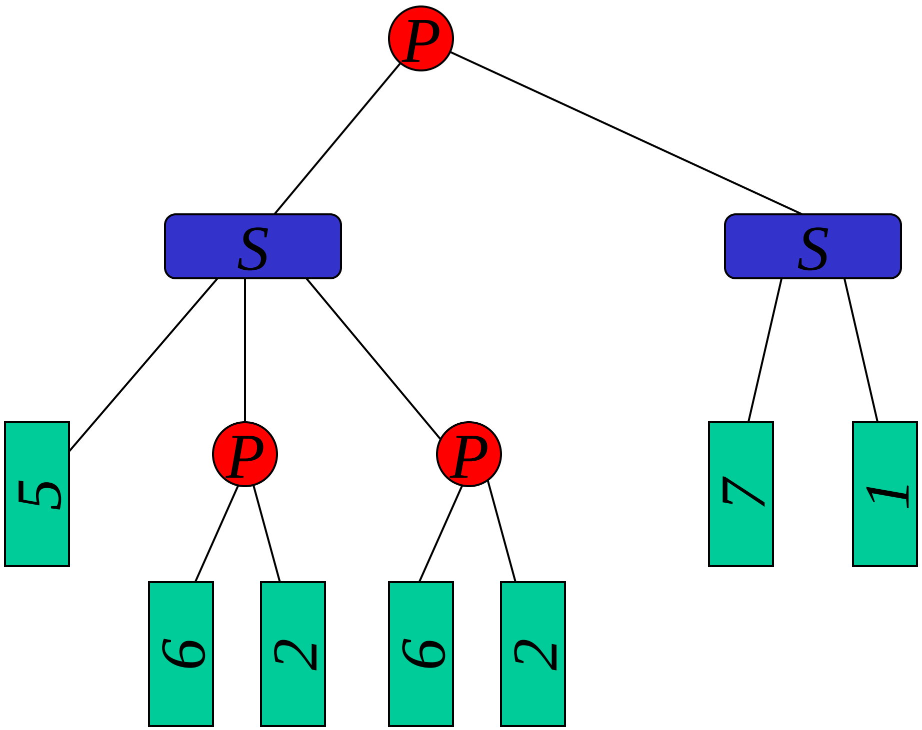


Parallel composition

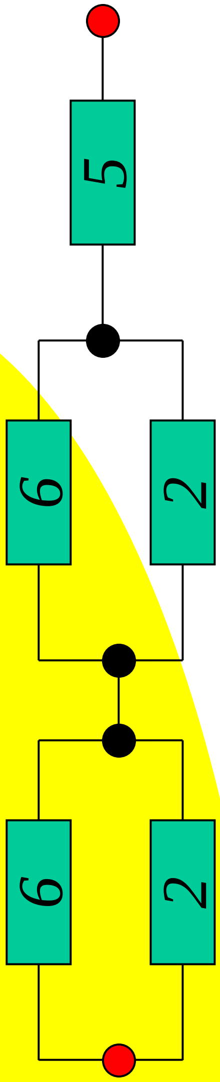


Series Parallel Graphs have an SP-tree

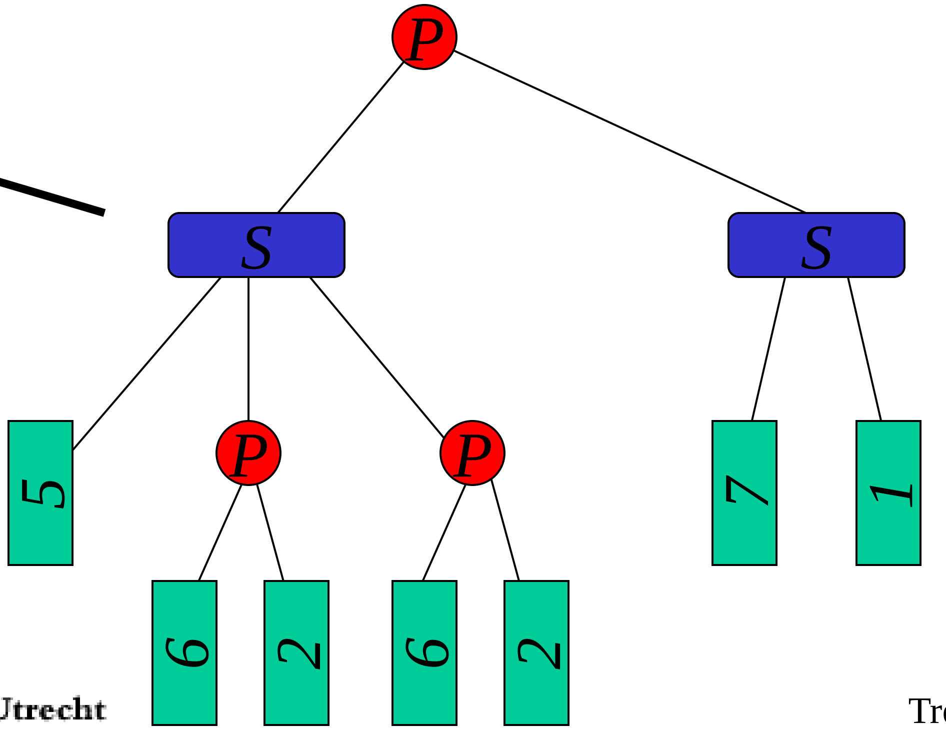




G(*i*)



- Associate to each node *i* of SP tree a 2-terminal graph G(*i*).



Maximum weighted independent set for series parallel graphs

- G(i), say with terminals s and t
- AA(i) = maximum weight of independent set S of G(i) with $s \in S, t \in S$
- BA(i) = maximum weight of independent set S of G(i) with $s \notin S, t \in S$
- AB(i) = maximum weight of independent set S of G(i) with $s \in S, t \notin S$
- BB(i) = maximum weight of independent set S of G(i) with $s \notin S, t \notin S$

Maximum weighted independent set of series parallel graphs 2

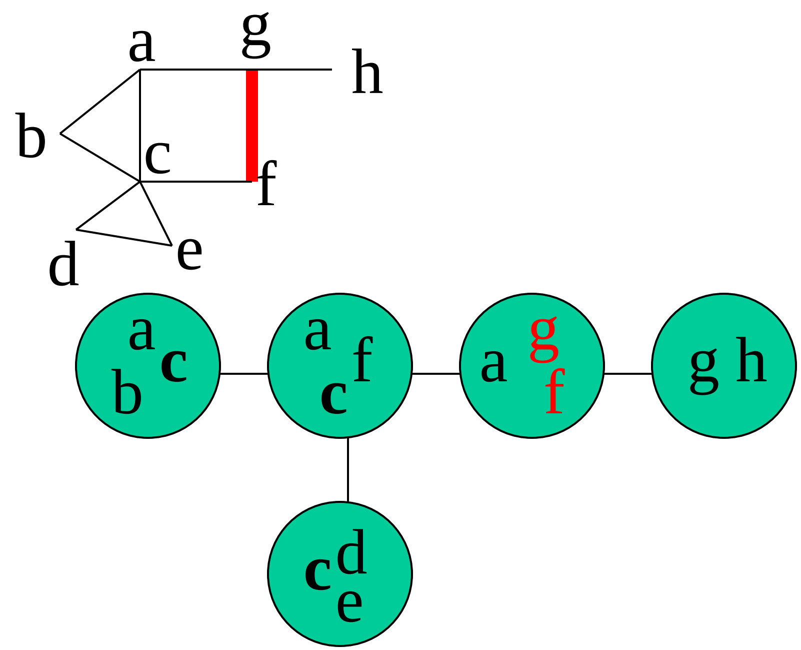
- Computing AA, AB, BA, BB for
  - Leaves of SP-tree: trivial
  - Series, parallel composition: case analysis, using values for sub-sp-graphs G(i_1), G(i_2)
  - E.g., series operation, s' terminal between i_1 and i_2
    - AA(i) = max {AA(i_1)+AA(i_2) – w(s'), AB(i_1) + BA(i_2) }
- O(1) time per node of SP-tree: O(n) total.

Many generalizations

- Many other problems
- Other classes of graphs to which we can assign a *tree-structure*, including
  - Graphs of treewidth k, for small k.

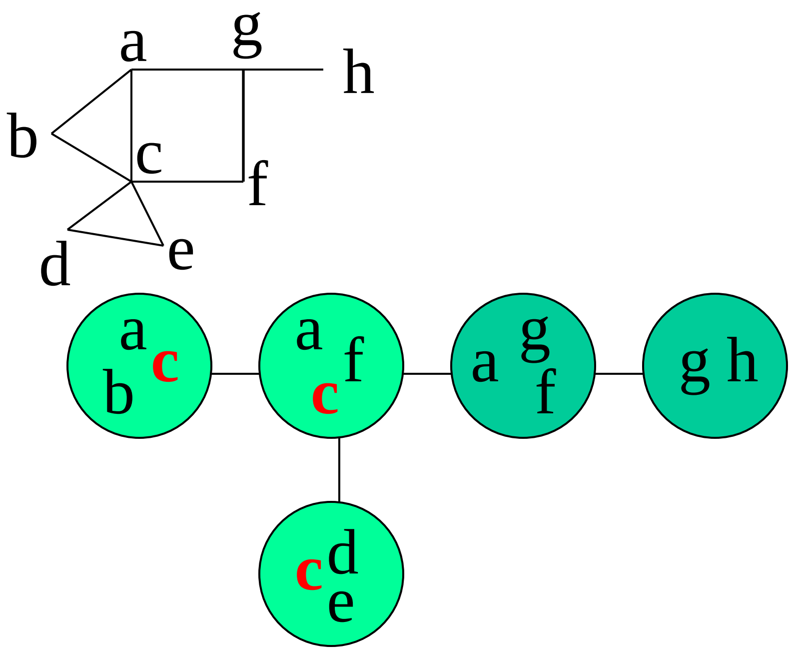
Tree decomposition

- A tree decomposition:
  - Tree with a vertex set associated to every node.
  - For all edges {*v*,*w*}: there is a set containing both *v* and *w*.
  - For every *v*: the nodes that contain *v* form a connected subtree.



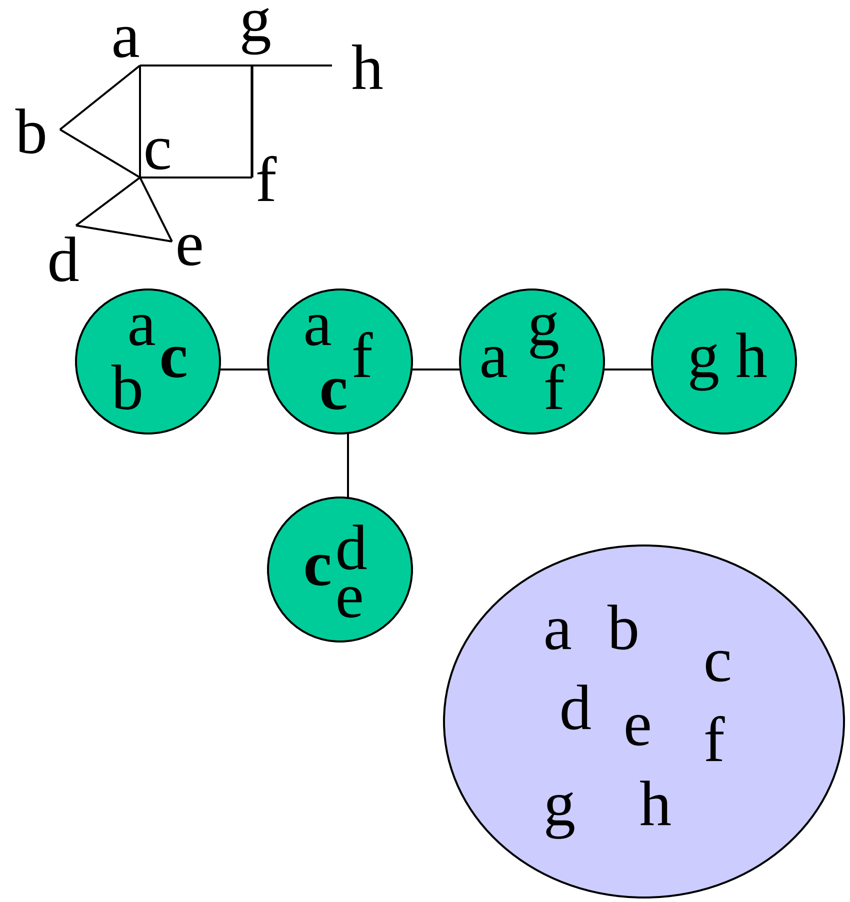
Tree decomposition

- A tree decomposition:
  - Tree with a vertex set associated to every node.
  - For all edges {*v*,*w*}: there is a set containing both *v* and *w*.
  - For every *v*: the nodes that contain *v* form a connected subtree.



Treewidth (definition)

- Width of tree decomposition:
  $\max_{i \in I} |X_i| - 1$
- Treewidth of graph G: tw(G)= minimum width over all tree decompositions of G.



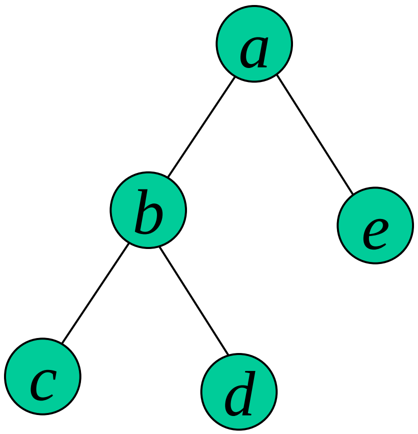
Universiteit Utrecht

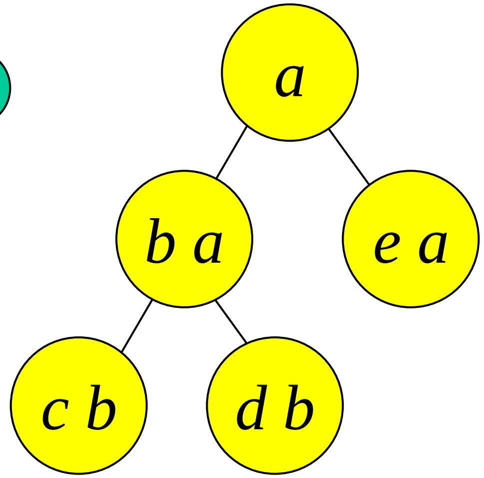
Some graphs have small treewidth

- Appearing in some applications (e.g., probabilistic networks)
- Trees have treewidth 1
- Series Parallel graphs have treewidth 2.
- …

Trees have treewidth one

- Choose a root r
- Take $X_r = \{r\}$, and for each other node i: $X_i = \{i, parent(i)\}$
- T with these bags gives a tree decomposition of width 2





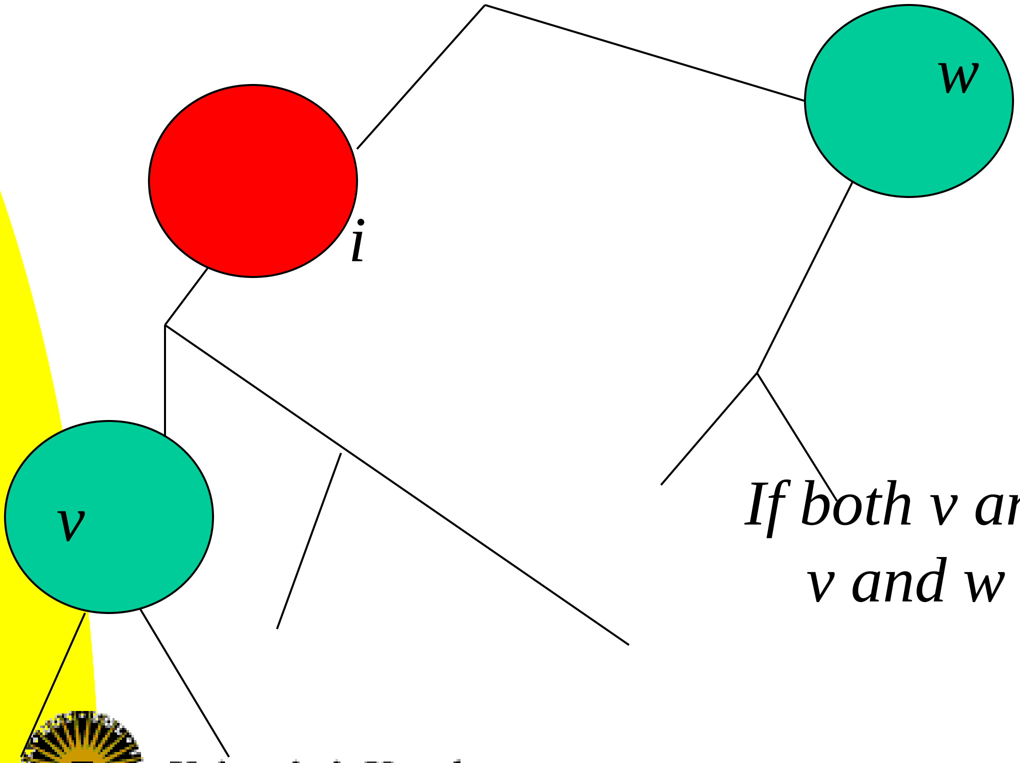
Algorithms using tree decompositions

- Step 1: Find a tree decomposition of width bounded by some small k.
  - Heuristics.
  - $O(f(k)n)$ in theory.
  - Fast $O(n)$ algorithms for $k=2$, $k=3$.
  - By construction, e.g., for trees, sp-graphs.
- Step 2. Use dynamic programming, bottom-up on the tree.

Determining treewidth

- Treewidth problem (decision version):
  - Given: Graph G, integer k
  - Question: Is the treewidth of G at most k
- Treewidth problem (construction version):
  - Given: Graph G
  - Question: construct a tree decomposition of G with minimum width
- NP-complete (Arnborg, Proskurowski)
- ☹(n) time algorithm for fixed k (B.)
- Practical O(n) algorithms for k= 1, 2, 3 (Arnborg, Corneil, Proskurowski)
- Practical O*(2^n) algorithm for small graphs (B. et al.)
- Many (often good) heuristics

Separator property



If both v and w not in X_i, then v and w are not adjacent

Nice tree decompositions

- Rooted tree, and four types of nodes *i*:
  - *Leaf*: leaf of tree with $|X_i| = 1$.
  - *Join*: node with two children j, j' with $X_i = X_j = X_{j'}$.
  - *Introduce*: node with one child j with $X_i = X_j \cup \{v\}$ for some vertex v
  - *Forget*: node with one child j with $X_i = X_j - \{v\}$ for some vertex v
- There is always a nice tree decomposition with the same width.

Define G(i)

- Nice tree decomposition.
- For each node i, G(i) subgraph of G, formed by all nodes in sets X_j, with $j=i$ or j a descendant of i in tree.
  - Notate: G(i) = (V(i), E(i)).

Maximum weighted independent set on graphs with treewidth k

- For node i in tree decomposition, $S \subseteq X_i$ write
  - $R(i, S)$ = maximum weight of independent set W of $G(i)$ with $W \cap X_i = S$,
    - $-\infty$ if such W does not exist

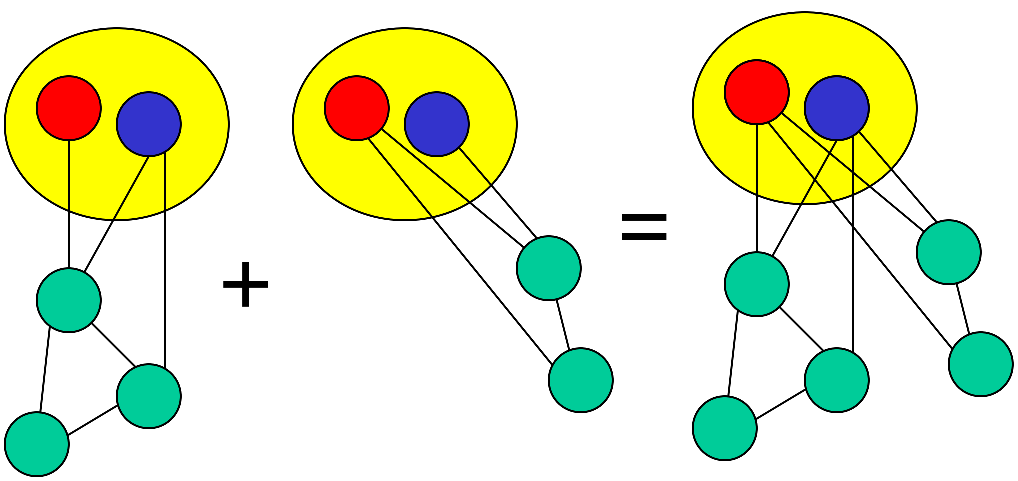
Leaf nodes

- Let i be a leaf node. Say $X_i = \{v\}$.
- $R(i,\{v\}) = w(v)$
- $R(i, \emptyset) = 0$

G(*i*) *is a graph with one vertex*

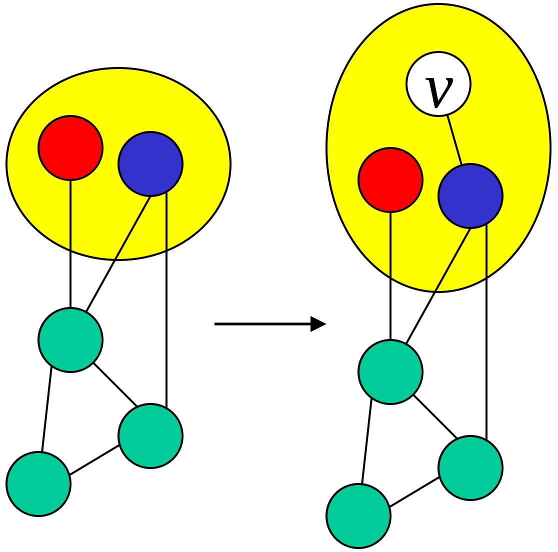
Join nodes

- Let i be a join node with children j_1, j_2.
- $R(i, S) = R(j_1, S) + R(j_2, S) - w(S)$.



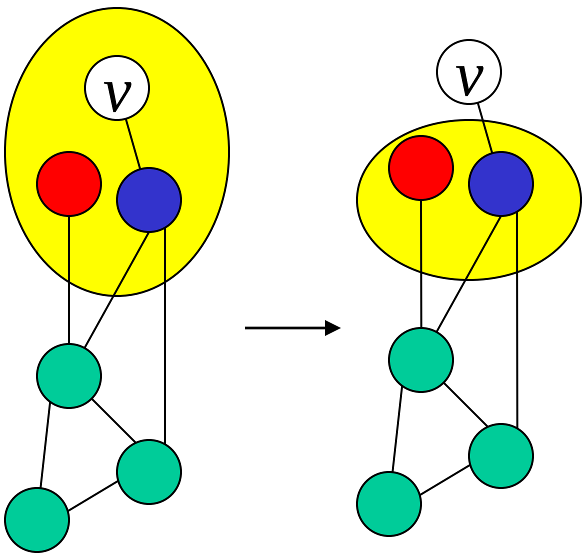
Introduce nodes

- Let i be a node with child j, with $X_i = X_j \cup \{v\}$.
- Let $S \subseteq X_j$.
- $R(i,S) = R(j,S)$.
- If v not adjacent to vertex in S: $R(i,S\cup\{v\})=R(j,S) + w(v)$
- If v adjacent to vertex in S: $R(i,S \cup \{v\}) = -\infty$.



Forget nodes

- Let i be a node with child j, with $X_i = X_j - \{v\}$.
- Let $S \subseteq X_i$.
- $R(i,S) = \max(R(j,S), R(j,S \cup \{v\}))$



Maximum weighted independent set on graphs with treewidth *k*

- For node i in tree decomposition, $S \subseteq X_i$ write
  - $R(i, S)$ = maximum weight of independent set W of $G(i)$ with $W \cap X_i = S$, $-\infty$ if such W does not exist
- Compute for each node i, a table with all values $R(i, \ldots)$.
- Each such table can be computed in $O(2^k)$ time when treewidth at most k.
- Gives $O(n)$ algorithm when treewidth is (small) constant.

Frequency assignment problem

- Given:
  - *Graph* $G=(V,E)$
  - *Frequency set* $F(v) \subseteq \mathbf{N}$ for all $v \in V$
  - *Cost function*
    - $c(e,r,s)$, $e = \{v,w\}$, r a frequency of v, s a frequency of w
- Question
  - Find a function g with
    - For all $v \in V$: $g(v) \in F(v)$
    - The total sum over all edges $e=\{v,w\}$ of $\mathbf{c(e,g(v),g(w))}$ is as small as possible

Frequency assignment when treewidth is small

- Suppose sets F(v) are *small*
- Suppose G has small treewidth
- Algorithm exploits tree decomposition

  What tables are we computing?
  - Leaf: trivial
  - Introduce: …
  - Forget: projection
  - Join: sum but subtract double terms

General method

- Compute a tree decomposition
  - E.g., with minimum degree heuristic
  - Make it nice
  - Use dynamic programming
- Works for many problems
  - Courcelle: those that can be formulated in monadic second order logic
  - Practical: TSP, frequency assignment, problems on planar graphs like dominating set, probabilistic inference

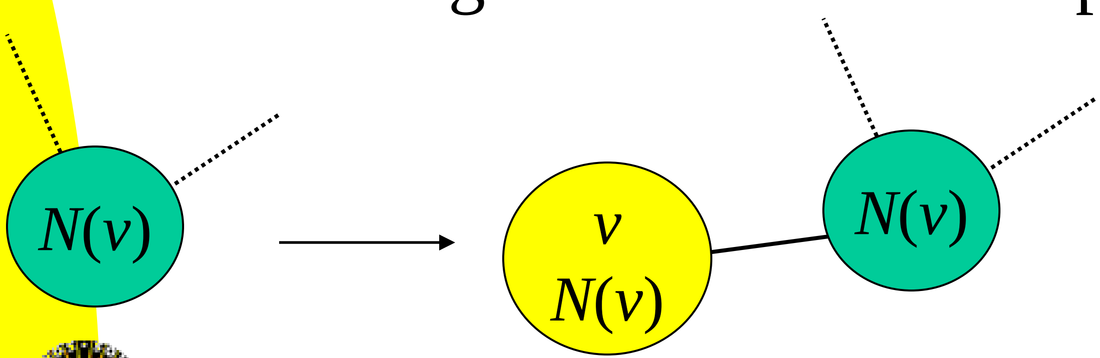
A lemma

- Let $(\{X_i \mid i \in I\}, T)$ be a tree decomposition of G. Let Z be a clique in G. Then there is a $j \in I$ with $Z \subseteq X_j$.
  - Proof: Take arbitrary root of T. For each $v \in Z$, look at highest node containing *v*. Look at such highpoint of maximum depth.

The minimum degree heuristic

A heuristic for treewidth
Works often well

- Repeat:
  - Take vertex v of minimum degree
  - Make neighbors of v a clique
  - Remove v, and repeat on rest of G
  - Add v with neighbors to tree decomposition



Other heuristics

- Minimum fill-in heuristic
  - Similar to minimum degree heuristic, but takes vertex with smallest *fill-in:*
    - Number of edges that must be added when the neighbours of v are made a clique
- Other choices of vertices, refining, using separators, …

Representation as permutation

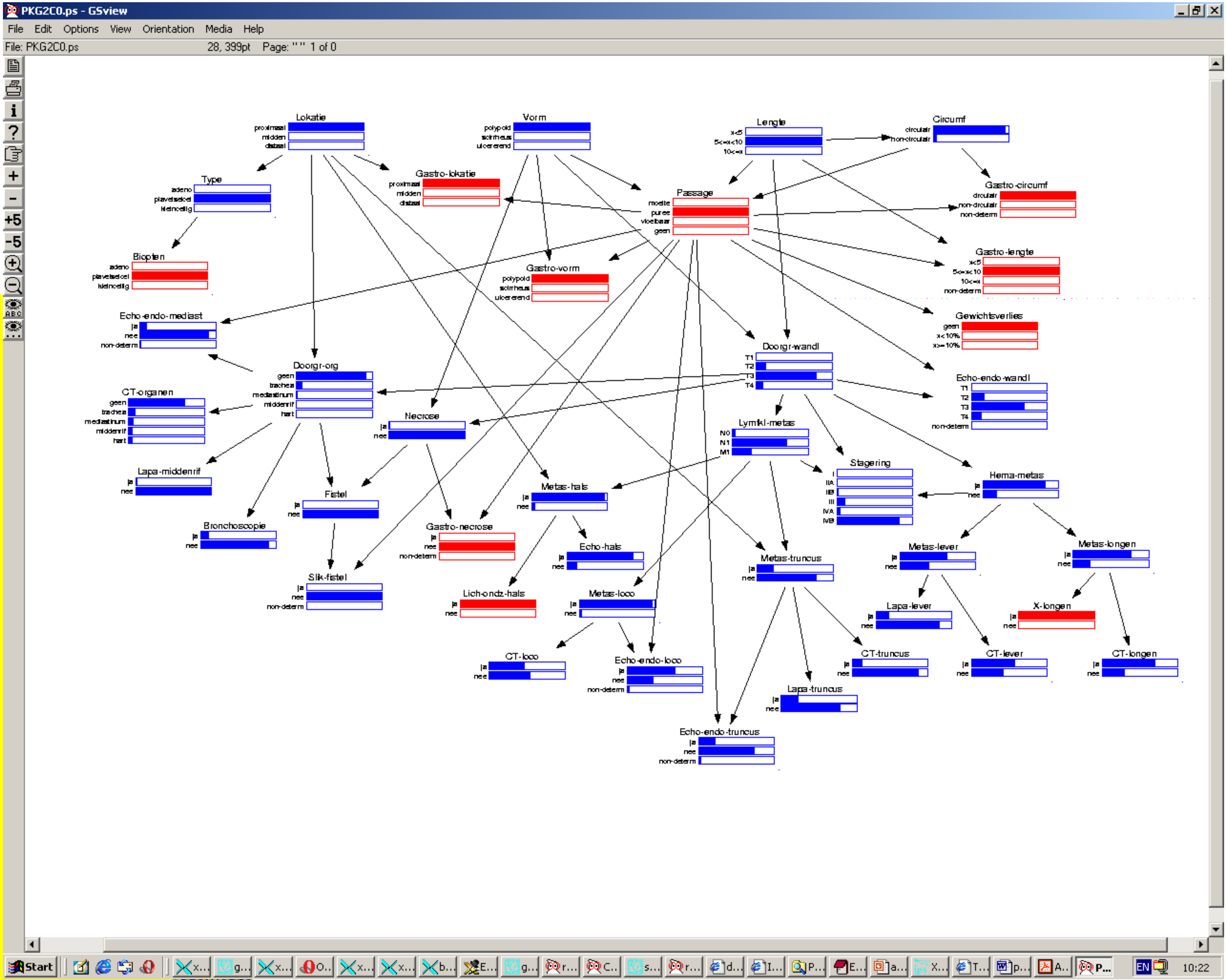
- A correspondence between tree decompositions and permutations of the vertices
  - Repeat: remove superfluous leaf bag, or take vertex that appears in 1 leaf bag and no other bag
  - Make neighbours of $v = \pi(1)$ into a clique; recursively make tree decomposition of graph – v; add bag with v and neighbours
- Used in heuristics, and local search methods (e.g., taboo search, simulated annealing) and genetic algorithms

Connection to Gauss eliminating

- Consider Gauss elimination on a symmetric matrix
- For n by n matrix M, let G_M be the graph with n vertices, and edge (i,j) if $M_{ij} \neq 0$
- If we eliminate a row and corresponding column, effect on G is:
  - Make neighbors of v a clique
  - Remove v

Application: Probabilistic networks

- Lauritzen-Spiegelhalter algorithm for inference on probabilistic networks (belief networks) uses a tree decomposition of the *moralized* form of the network
- Underlying several modern decision support networks


PKG2C0.ps - GSview
File Edit Options View Orientation Media Help
File: PKG2C0.ps 28, 399pt Page: "" 1 of 0
Lokatie
Vorm
Lengte
Circumf
Type
Gastro-lokatie
Passage
Gastro-circumf
Biopten
Gastro-vorm
Gastro-lengte
Echo-endo-mediast
Gewichtsverlies
Doorgr-wandl
Doorgr-org
Echo-endo-wandl
CT-organen
Necrose
Lymfkl-metas
Stagering
Lapa-middenrif
Fistel
Metas-hals
Hema-metas
Bronchoscopie
Gastro-necrose
Echo-hals
Metas-lever
Metas-longen
Metas-truncus
Slik-fistel
Lich-ondz-hals
Metas-loco
Lapa-lever
X-longen
CT-loco
Echo-endo-loco
CT-truncus
CT-lever
CT-longen
Lapa-truncus
Echo-endo-truncus
Start
EN
10:22

Designing a DP algorithm

- Methodology:
  1. What are "partial certificates"?
  2. What characterizes a partial certificate (essential for extending to full certificate)? Gives set of subproblems
  3. Give recurrences for subproblems
  4. Find order in which recurrences are evaluated; or use memorization
  5. Give algorithm; possibly save memory or make construction version

Conclusions

- Dynamic programming for graphs with tree-like structure
- Works for a large collection of problems, as long as there is (and we can find) such a structure…