Algorithm Graph Theory: How hard is your combinatorial optimization problem?

Short Course – Lecture 5
June 9, 2017

Slides available at:
https://www.math2.rwth-aachen.de/de/mitarbeiter/koster/agtclemson

Schedule

	9.00-10.30	10.30-11.00	11:00-12:30
Wed 06/07	**Basics:** Complexity	Break	**Basics:** First Examples
Thu 06/08	**Basics:** Interval graphs	Break	**Basics:** chordal and perfect graphs
Fri 06/09	**Treewidth:** introduction	Break	**Treewidth:** Graph classes of bounded treewidth
Mon 06/12	**Treewidth:** Lower and Upper Bounds	Break	**Treewidth:** Dynamic Programming
Tue 06/13	**FPT:** Parameterized Complexity	Break	**FPT:** Kernelization
Wed 06/14	**Exact:** Branching Algorithms	Break	**Exact:** Dynamic Programming

Yesterday:

Theorem

Let G be a graph. The following properties are equivalent:

1. *G is chordal*
2. *G is the intersection graph of a collection of subtrees of a tree*
3. *there exists a tree $T = (K, L)$ such that node set K represents all maximal cliques in G and edge set L is chosen such that the subgraph induced by $K_v := \{Q \in K : v \in Q \text{ clique in } G\}$ represents a subtree.*

- For chordal graphs we have:

$$vw \in E \Leftrightarrow \exists k \in K \textbf{ with } v, w \in V_k$$

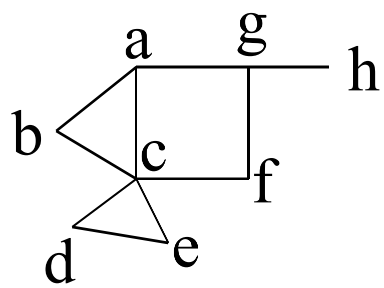
- What happens if we change to property to:

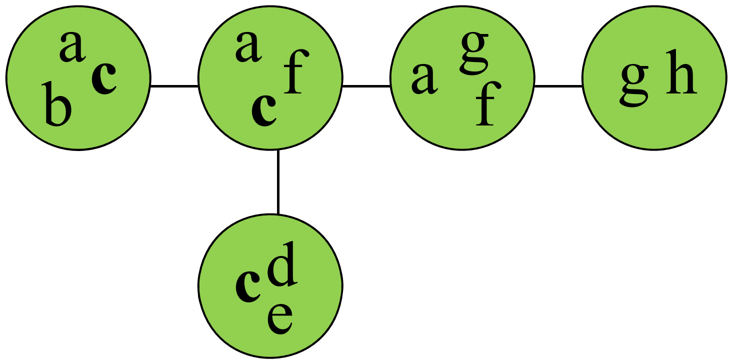
$$vw \in E \Rightarrow \exists k \in K \textbf{ with } v, w \in V_k$$

- So the vertices do not need to be pairwise adjacent in V_k

Tree Decomposition

- A tree decomposition (T,X) of G=(V,E):
  - Tree T=(I,F) with a vertex set X_i associated with every node $i \in I$
  - For all edges $\{v,w\} \in E$: there is a set X_i containing both v and w
  - For every $v \in V$: the nodes $i \in I$ that contain v form a connected subtree

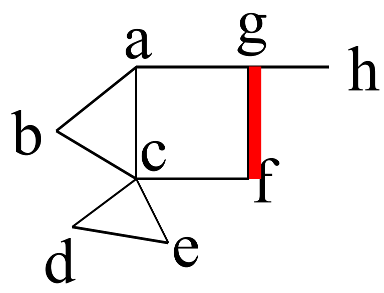


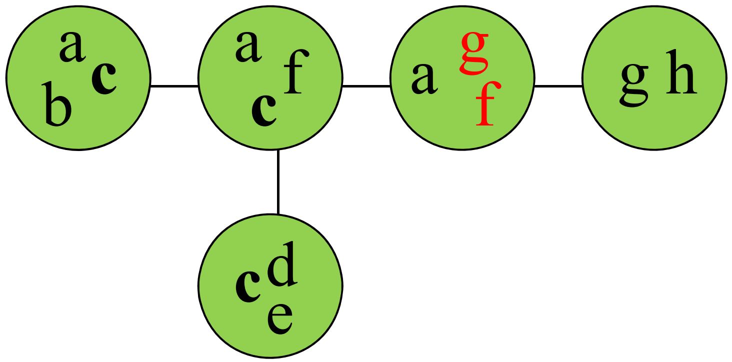


CLEMSON UNIVERSITY RWTH AACHEN UNIVERSITY

Tree Decomposition

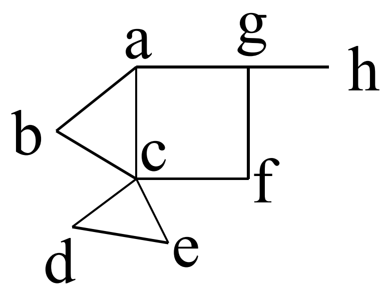
- A tree decomposition:
  - Tree with a vertex set associated with every node
  - For all edges {v,w}: there is a set containing both v and w
  - For every v: the nodes that contain v form a connected subtree

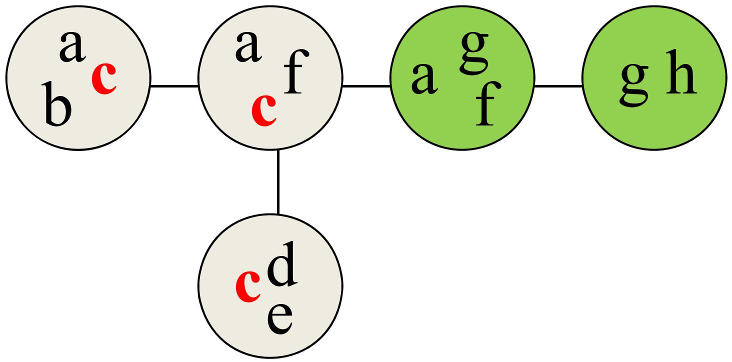




Tree Decomposition

- A tree decomposition:
  - Tree with a vertex set associated with 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

- Condition 3 can be replaced by
  - For all triples $i,j,k \in I$ with j on the path between i and k in T, it holds that $X_i \cap X_k \subseteq X_j$
- Given (X,T), we can define the **chordalization** of G:
  - $G(X,T)=(V(X,T),E(X,T))$ with
  - $V(X,T)=V$
  - $E(X,T)=\{vw: v,w \in V, \exists i \in I \text{ s.t. } v,w \in X_i\}$

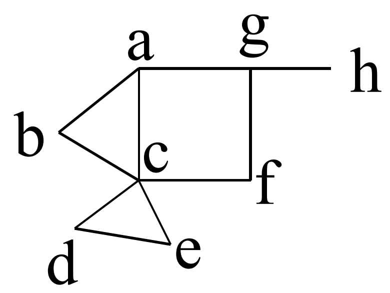
Lemma: $G(X,T)$ is chordal

Lemma: G is chordal if and only if there exists a (X,T) s.t. $G=G(X,T)$

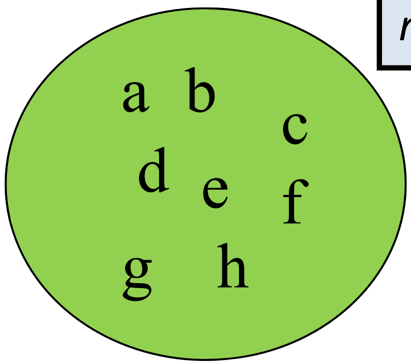
Treewidth

- **Width** of tree decomposition:

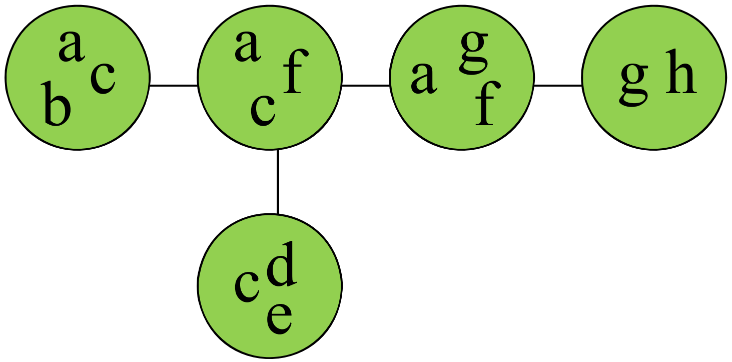
$$\max_{i \in I} |X_i| - 1$$



- **Treewidth** of graph *G*: tw(*G*)=
  minimum width over all tree decompositions of *G*.

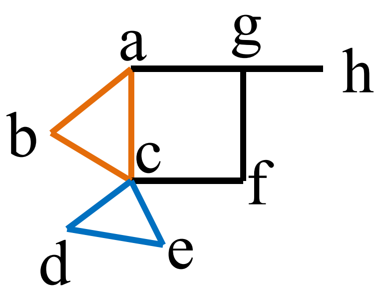


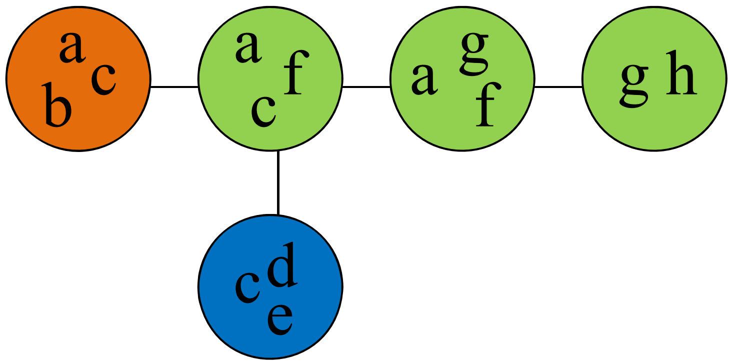
maximum bag size - 1



Lemma/Definition: tw(G)=min {ω(G(X,T))-1: (X,T) t.d.}

First observations



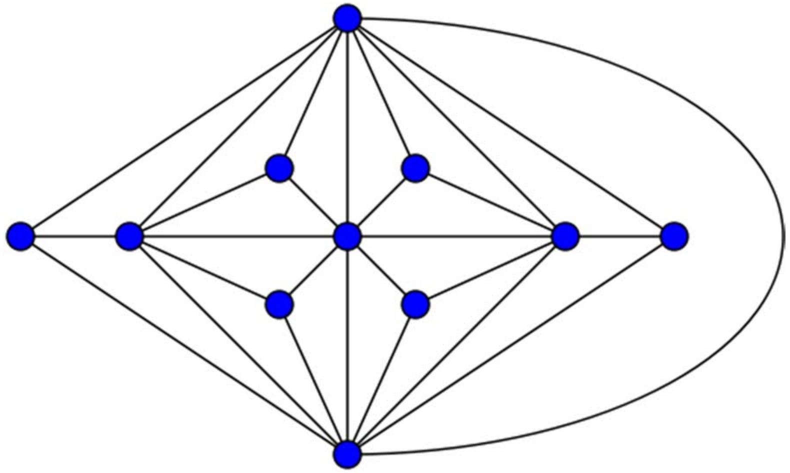


Each clique has to be part of at least one node

Clique number - 1 is a lower bound for treewidth

Trees have treewidth 1

k-trees and partial k-trees



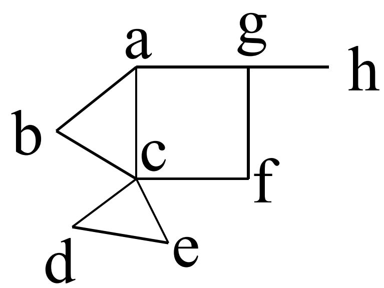
Source: wikipedia/David Eppstein

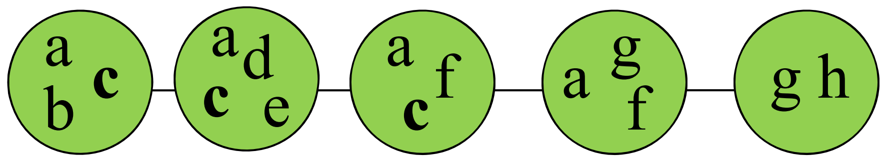
- a ***k*-tree** is an undirected graph formed by
  - starting with a (k + 1)-vertex complete graph and then
  - repeatedly adding vertices in such a way that each added vertex has exactly k neighbors that, together, the k + 1 vertices form a clique
- A partial-k-tree is a subgraph of a k-tree

Theorem: A graph has treewidth at most k if and only if it is a partial k-tree

Path Decomposition

- A path decomposition:
  - Path with a vertex set associated with 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 subpath





Lemma: G is interval if and only if there exists a path decomposition (X,T) s.t. G=G(X,T)

Treewidth

- Treewidth (and pathwidth) of a subgraph of G is bounded from above by tw(G) (resp. pw(G))
- Treewidth of G equals the maximum treewidth of its two-connected components
- Pathwidth of G equals the maximum pathwidth of ist connected components

Lemma: Let G be a connected graph with at least 2 vertices. Then, tw(G)=1 if and only if G is a tree.

- Pathwidth of a binary tree can be O(log n)

Graphs with pw(G)=1

- A caterpillar tree is a tree where all nodes have distance at most 1 to a central path

Lemma: Let G be a connected graph with at least 2 vertices. Then, pw(G)=1 if and only if G is a caterpillar tree.

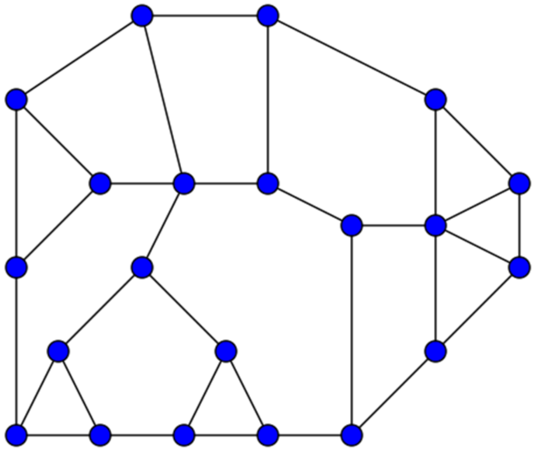
Graphs with tw(G)≤2

Source: wikipedia/David Eppstein

- A **two-terminal graph** (TTG) is a graph with two distinguished vertices, *s* and *t* called *source* and *sink*, respectively.
- The **parallel composition** $Pc = Pc(X,Y)$ of two TTGs *X* and *Y* is a TTG created from the disjoint union of graphs *X* and *Y* by merging the sources of *X* and *Y* to create the source of *Pc* and merging the sinks of *X* and *Y* to create the sink of *Pc*.
- The **series composition** $Sc = Sc(X,Y)$ of two TTGs *X* and *Y* is a TTG created from the disjoint union of graphs *X* and *Y* by merging the sink of *X* with the source of *Y*. The source of *X* becomes the source of *Sc* and the sink of *Y* becomes the sink of *Sc*.
- A **two-terminal series-parallel graph** (TTSPG) is a graph that may be constructed by a sequence of series and parallel compositions starting from a set of copies of a single-edge graph K_2 with assigned terminals.
- ***Definition***. Finally, a graph is called **series-parallel (sp-graph)**, if it is a TTSPG when some two of its vertices are regarded as source and sink.

Lemma: Let G be a connected graph with at least 2 vertices. Then, tw(G)≤2 if and only if G is a series-parallel graph or a tree.

Graphs with tw(G)≤3



- A **Halin graph** is a type of planar graph, constructed by connecting the leaves of a tree into a cycle. The tree must have at least four vertices, none of which has exactly two neighbors

Lemma: Let G be a Halin graph. Then, tw(G)≤3.

Decision problems

- TREEWIDTH: given G, k, is tw(G)≤k?
- PATHWIDTH: given G, k, is pw(G)≤k?

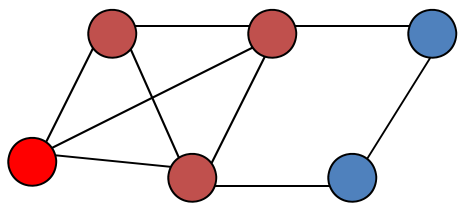
Theorem: TREEWIDTH and PATHWIDTH are NP-complete.

Theorem (Bodlaender, 1996): There is a polynom p(.) and an algorithm that computes a tree decomposition (T,I) with width k=tw(G) in time $O(n*2^{p(k)})$

Reconsider our first observation:

Each (maximal) clique has to be part of at least one node

Simplicial vertex:
A vertex is simplicial if all its neighbors are mutually adjacent



A simplicial vertex is part of only one maximal clique

A simplicial vertex has to occur in only one TD-node

A first algorithm:

Assumption: G has a simplicial vertex, and after ist removal there is again and again a simplicial vertex

---

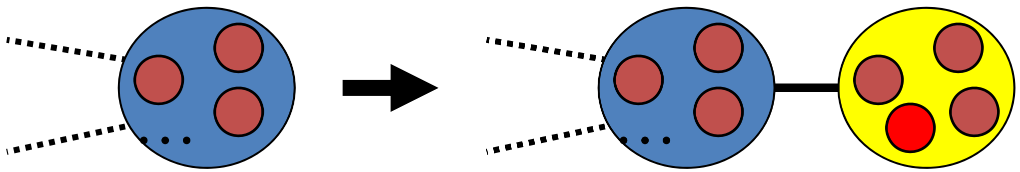
Repeatedly remove a simplicial vertex of G: $v_1,\ldots,v_n$

For i = n down to 1 do

  Construct a TD-node with v_i and all its neighbors in $G[v_i,\ldots,v_n]$

  Attach node to a node containing all neighbors of v_i in $G[v_i,\ldots,v_n]$

Return tree decomposition



⇒ Width of returned TD equals maximum clique minus 1

Analyzing the algorithm

Width of returned TD equals maximum clique minus 1

⇒ Tree Decompoisition is optimal !!!

Which graphs satisfy the assumption ?

⇒ G is chordal iff there exists a perfect elimination scheme [59,64]

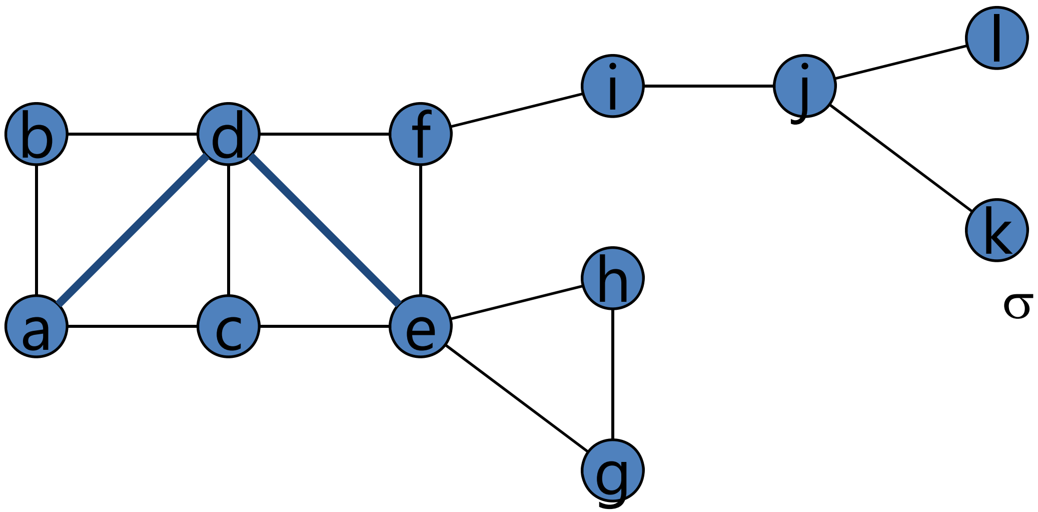
⇒ Optimal algorithm for chordal graphs!

Non-chordal graphs

What to do with non-chordal graphs ?

Gavril [59]: A graph $G=(V,E)$ is chordal if and only if there exists a tree $T=(I,F)$ such that one can associate with each vertex $v\in V$ a subtree $T_v=(I_v,F_v)$ of T, such that $vw\in E$ if and only if $I_v\cap I_w\neq\emptyset$.

⇒ There exists a chordalization $H=(V,E\cup F)$ of G with maximum clique size $k+1$ if and only if the treewidth of G is k.



σ = [b,a,c,d,g,h,e,f,i,k,j,l]

Chordalization Algorithms

Find chordalization of G with small maximum clique size

- Adapt algorithms to test if a graph is chordal
- Algorithms for related MIN-FILL-IN problem

Dirac, 1961: Every non-complete triangulated graph has two nonadjacent simplicial vertices

Without loss of generality an arbitrary vertex can be put at the end of the elimination scheme

Linear time algorithms to test graph chordality:

- Lexicographic Breadth First Search (LEX_M & LEX_P)
  - Rose, Tarjan & Lueker [111]
- Maximum Cardinality Search (MCS & MCS_M)
  - Tarjan & Yannakakis [120], Heggernes et al.[84]

Minimum Fill-In problem

MINIMUM FILL-IN:

min{ |F| : (V,E+F) is chordal }

⇒ Computing MINIMUM FILL-IN is NP-hard

Heuristics:

- Greedy Fill-In
  - repeatedly select vertex that introduces least number of edges to be simplicial
  - remove vertex, add fill-in edges
- Minimum Degree Fill-In
  - repeatedly select vertex with smallest degree
  - remove vertex, add fill-in edges