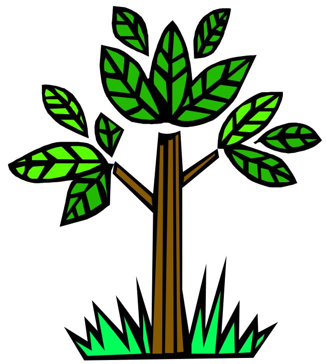
Treewidth and Integer Programming



Mixed Integer Programming workshop

June 5, 2006, Miami

Arie Koster Zuse Institute Berlin (ZIB)

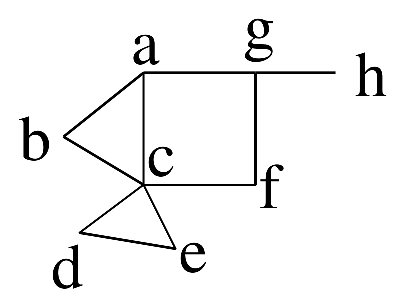
koster@zib.de http://www.zib.de/koster

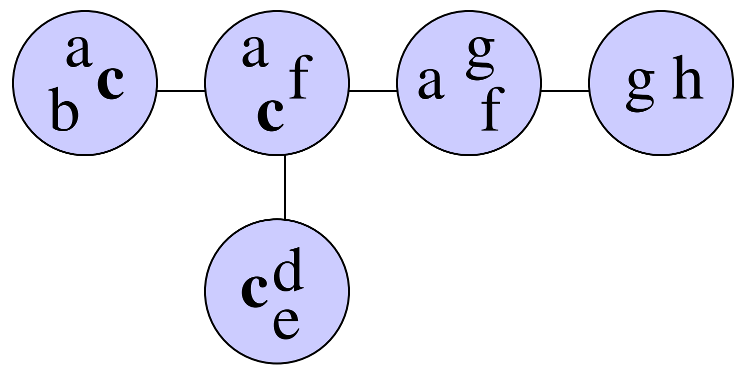
Contents

- Treewidth vs. Integer Programming
- Treewidth by Integer Programming
- Experiments

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

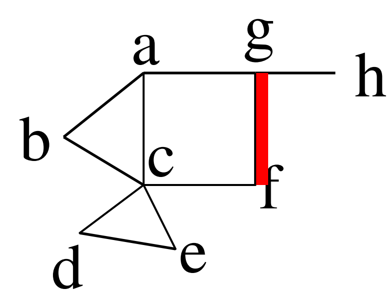


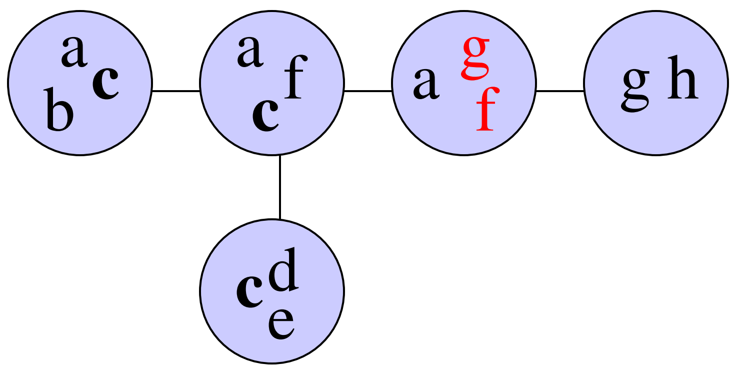


Arie Koster

Tree Decomposition

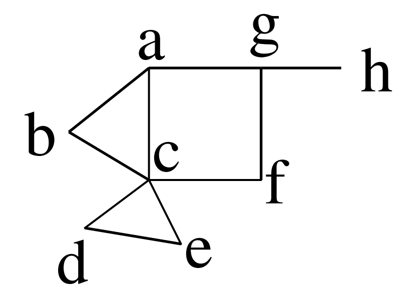
- A tree decomposition:
  - Tree with a vertex set associated with every node
  - For all edges {v,w}: there is a set containing both v and w
  - For every v: the nodes that contain v form a connected subtree

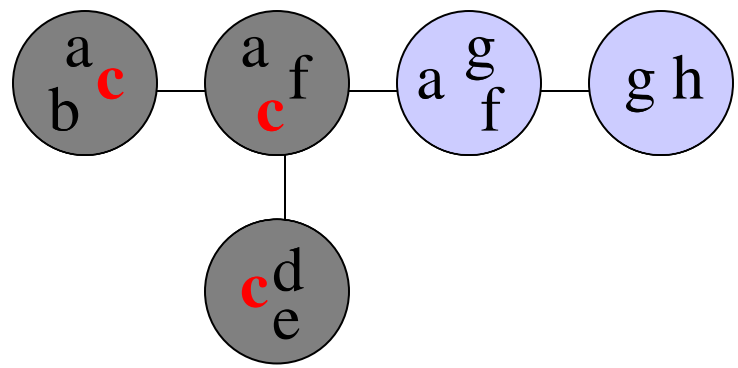




Tree Decomposition

- 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

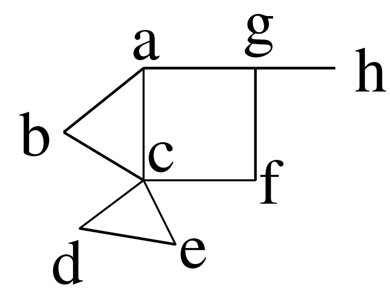




Treewidth

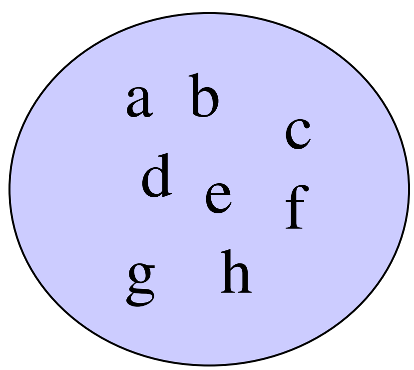
- Width of tree decomposition:

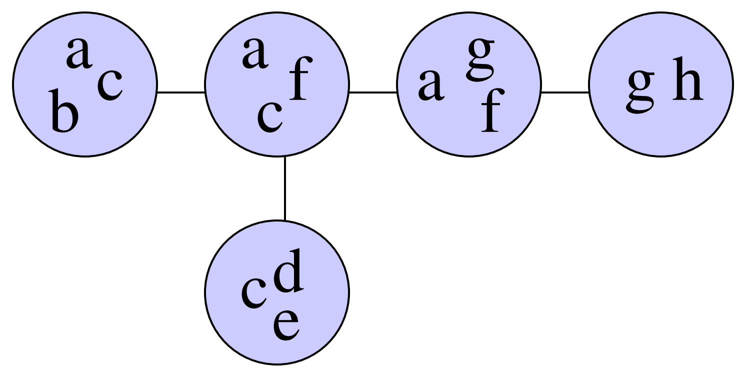
$$\max_{i \in I} |X_i| - 1$$



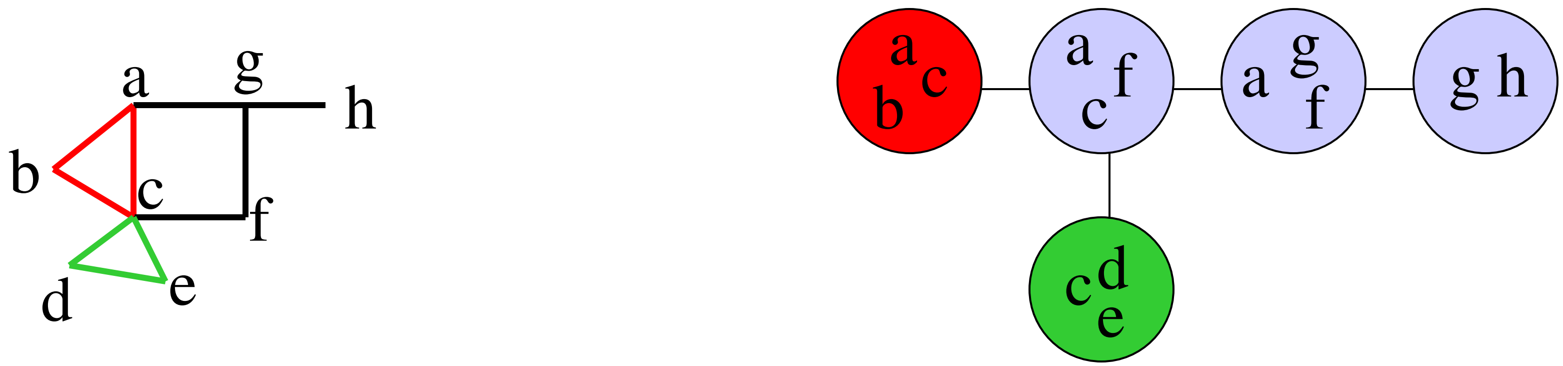
maximum bag size - 1

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





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

ZIB
Arie Koster

Branchwidth, Treewidth, Pathwidth

Robertson and Seymour [106]: For a graph $G=(V,E)$,
$\max\{ \text{bw}(G), 2 \} \leq \text{tw}(G) + 1 \leq \max\{ \lfloor 3/2\, \text{bw}(G) \rfloor, 2 \}$

- Graphs with bounded treewidth have bounded branchwidth and vice versa
- Given a branch decomposition, we can construct a tree decomposition with TD-width at most 3/2 times the BD-width

→Illya Hicks

- Pathwidth: T is restricted to be a path; $\text{tw}(G) \leq \text{pw}(G)$
- Trees do not have bounded pathwidth

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, series-parallel-graphs.
- Step 2. Use dynamic programming, bottom-up on the tree.
  - Let $Y_i = \cup X_i$ over all descendants of $i \in I$
  - Compute optimal solution in $G[Y_i]$ for each set $S \subseteq X_i$, based on the solutions for the children

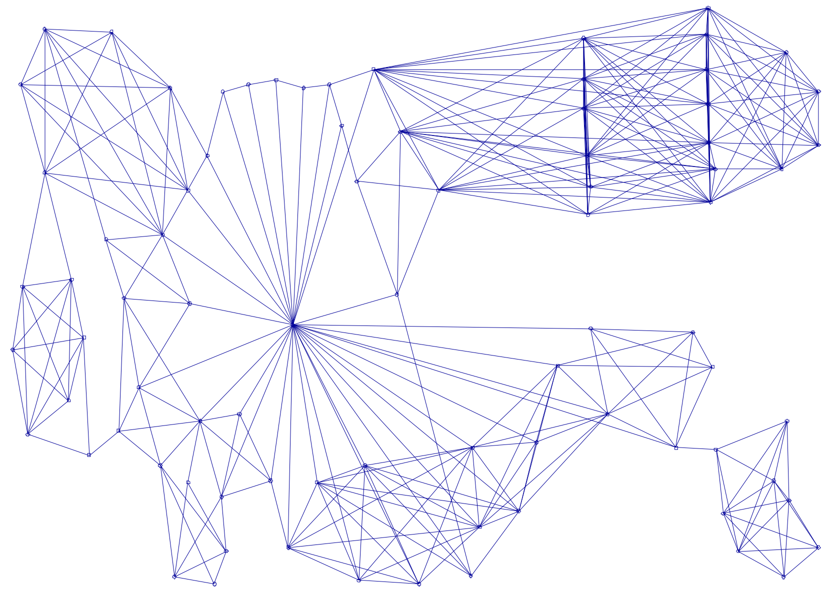
Maximum weighted independent set on graphs with treewidth k

- For node i in tree decomposition, $S \subseteq X_i$ write
  - $R(i, S)$ = maximum weight of independent set S of $G[Y_i]$ with $S \cap X_i = S$, $-\infty$ if such S 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.

⇒ Many problems can be solved in polynomial time given a graph of bounded treewidth

- Probabilistic networks
- **Frequency assignment**

Minimum Interference FAP

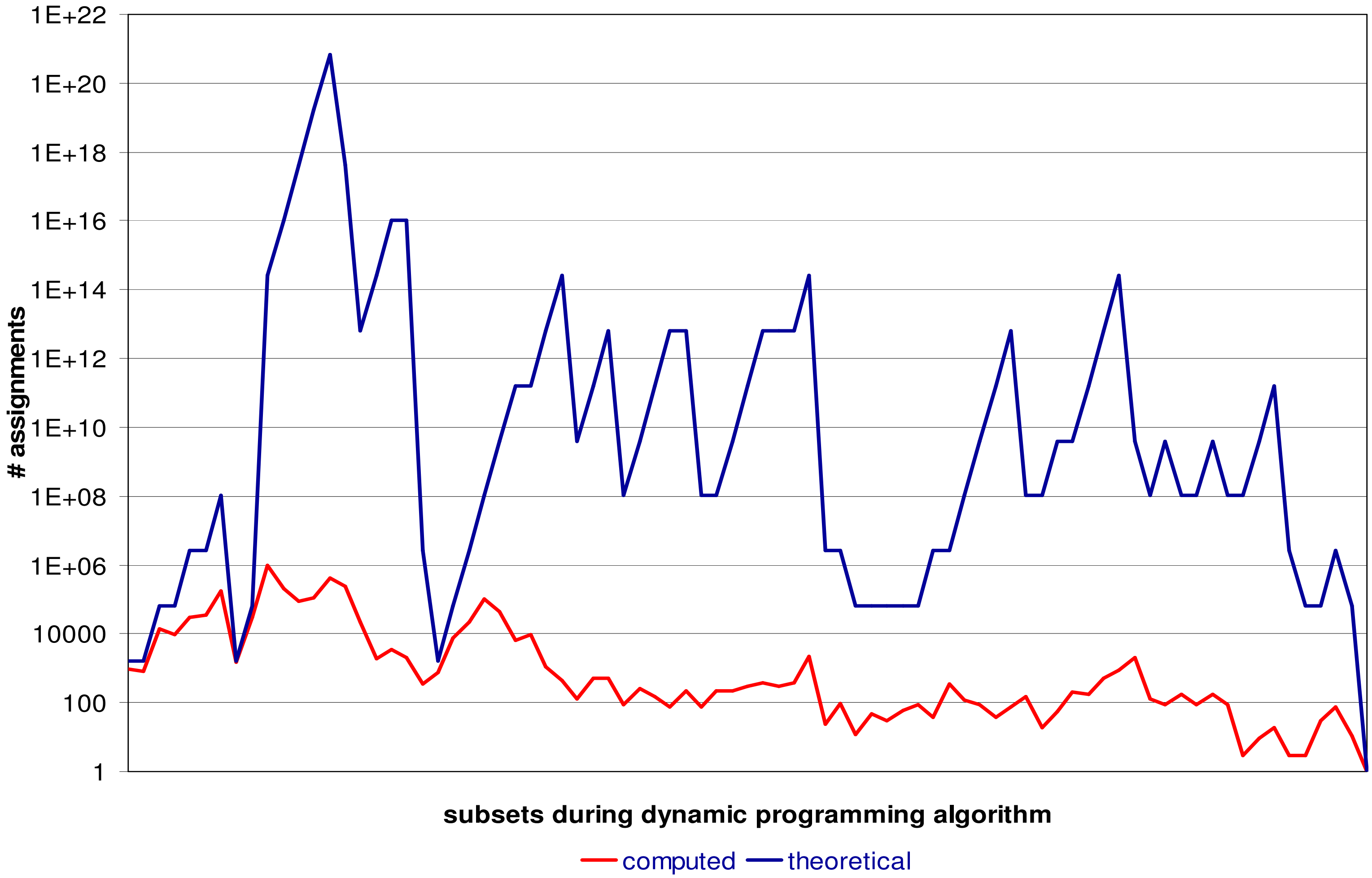


- Graph G=(V,E)
  - Vertices correspond to bi-directional connections
  - Edges indicate interference between two connections
- For every vertex v, set of frequency pairs D(v) is specified
- Interference quantified by edge penalties p(v,f ,w,g)
- Preferences for frequencies quantified by penalties q(v,f)
- Objective: Select for each vertex exactly one frequency, such that the total penalty is minimized.

Does it work in practice ?

- Only with (pre)processing techniques
  - Graph reduction
    - Vertices with degree 1 can be removed
    - Vertices with degree 2 can be removed
  - Domain reduction
    - Upper bounding
    - Dominance of domain elements

Computational Results



How do we get a tree decomposition all small width?

TREEWIDTH:
Given k ≥ 0 and G a graph, is the treewidth of G ≤ k ?

⇨ Computing TREEWIDTH is NP-hard — Arnborg et al.[13]

⇨ Linear time algorithm for TREEWIDTH if k not part of the input — Bodlaender [25]

- Exponential in k
- Not practical, even for k as small as 4

⇨ Several exponential time algorithms

- O(2^n poly(n)) — Arnborg et al.[13]
- O(1.9601^n poly(n)) — Fomin et al.[57]
- poly(n) denotes a polynomial in n

References refer to Tutorials 2005 chapter

Exact & approx. algorithms

- $O(\log k)$ approximation algorithm — Amir [9], Bouchitté et al. [41]

Computational approaches

- Branch-and-Bound algorithm — Gogate and Dechter [63]
  $O(2^{k+2})$ algorithm — Shoikhet and Geiger [117]
- Experiments with $O(2^n \text{poly}(n))$ time+memory algorithm — Bodlaender et al., ESA 2006
- Experiments with integer programming formulation (B&C)

References refer to Tutorials 2005 chapter

Other approaches

→Heuristic algorithms based on chordal graphs

→Minimum separating set heuristic [83]

→Metaheuristics

- Tabu Search [45]
- Simulated Annealing [79]
- Genetic algorithm [92]

→Preprocessing

- Reduction rules [39]
- Safe Separators [32]

References refer to Tutorials 2005 chapter

Treewidth Lower Bounds

Lemma *The minimum degree of a graph is a lower bound for treewidth*

$$\delta(G) \leq tw(G)$$

Corollary *The degeneracy of a graph is a lower bound for treewidth*

$$\delta D(G) = \max_{H \subseteq G} \delta(H) \leq tw(G)$$

Corollary *The contraction degeneracy of a graph is a lower bound for treewidth*

$$\delta C(G) = \max_{H \pi G} \delta(H) \leq tw(G)$$

See [36,37,38,88], Tutorials 2005 chapter

Contents

- Treewidth vs. Integer Programming
- Treewidth by Integer Programming
- Experiments

Treewidth by IP ? Chordal graphs

Chordal graph:
Every cycle of size at least 4 contains a chord

Gavril (1974): 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.

Let $\mathbf{H}(G)$ be the set of all chordalizations of G.

$$tw(G) = \min_{H \in \mathrm{H}(G)} \omega(H) - 1$$

⇒ Select best H and compute maximum clique size!

Related questions

Fill-in:
Minimum #edges to be added to obtain a chordal graph.

There exists a chordalization $H = (V, E \cup F)$ of G with $|F| = k$ if and only if the fill-in of G is k.

$$fi(G) = \min_{H \in \mathrm{H}(G)} |E_H| - |E_G|$$

Weighted treewidth (weights c(v))**:**
Minimum over all tree decompositions of the maximum product $\prod_{v \in X_i} c(v)$ over all bags $i \in I$.

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

$$\log(wtw(G)) = \min_{H \in \mathrm{H}(G)} \omega(H, \log(c))$$

Arie Koster

Chordalization polytope (1)

➡ All three problems need chordalization of G

Chordalization polytope:
Convex hull of all chordalizations *H* of *G*.

➡ How to identify whether a graph is chordal or not?

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

Perfect Elimination Scheme $\sigma = [v_1,\ldots,v_n]$**:**
Ordering of the vertices such that for all i, v_i is a simplicial vertex of the induced graph $G[v_i,\ldots,v_n]$

Chordalization polytope (2)

$$x_{vw} = \begin{cases} 1 & \text{if } vw \in E \cup F \text{ and } \pi(v) < \pi(w) \\ 0 & \text{otherwise} \end{cases}$$

Existence of edges

$$x_{vw} + x_{wv} = 1 \quad vw \in E$$

$$x_{vw} + x_{wv} \leq 1 \quad vw \notin E$$

Simplicity of vertices

$$y_{uv} + y_{uw} \leq 1 + y_{vw} + y_{wv} \quad u, v, w \in V$$

Ordering of vertices

$$\left(\sum_{i=1}^{|C|-1} y_{\rho(i)\rho(i+1)} \right) + y_{\rho(|C|)\rho(1)} \leq |C| - 1 \quad \forall C \subseteq V, |C| \geq 3, \rho : \{1, \ldots, |C|\} \to C$$

Objectives

Treewidth

$$\begin{aligned} \min \quad & z \\ s.t. \quad & z \geq \sum_{w \neq v} y_{vw} \quad v \in V \end{aligned}$$

Fill-in

$$\begin{aligned} \min \quad & f \\ s.t. \quad & f = \sum_{vw \notin E} (y_{vw} + y_{wv}) \end{aligned}$$

Weighted Treewidth

$$\begin{aligned} \min \quad & w \\ s.t. \quad & w \geq \log(c_v) \sum_{w \neq v} \log(c_w) y_{vw} \quad v \in V \end{aligned}$$

$$y \in C$$

Chordalization polytope

Contents

- Treewidth vs. Integer Programming
- Treewidth by Integer Programming
- Experiments

Separation of ordering inequalities

$$\left(\sum_{i=1}^{|C|-1} y_{\rho(i)\rho(i+1)}\right) + y_{\rho(|C|)\rho(1)} \leq |C| - 1 \quad \forall C \subseteq V, |C| \geq 3, \rho : \{1,\ldots,|C|\} \to C$$

- Inequality for every subset & every order of the subset
- Implicit consideration by separation

$$\left(\sum_{i=1}^{|C|-1} \left(y_{\rho(i)\rho(i+1)} - 1\right)\right) + \left(y_{\rho(|C|)\rho(1)} - 1\right) \leq -1$$

$$x_{vw} := 1 - y_{vw} \quad \Rightarrow \quad \left(\sum_{i=1}^{|C|-1} x_{\rho(i)\rho(i+1)}\right) + x_{\rho(|C|)\rho(1)} \geq 1$$

- Separation by shortest path computation in auxiliary digraph

Simplicity of vertices

$$y_{uv} + y_{uw} \leq 1 + y_{vw} + y_{wv} \quad u, v, w \in V$$

- Inequality for every triple of vertices
- Always satisfied if $vw \in E$
- Other implicitly handled by separation (lazy cuts)

Cliques

- Ordering represents a chordal graph

Dirac (1961): Every non-complete chordal graph has two nonadjacent simplicial vertices

- Without loss of generality, we can put an arbitrary vertex at the end of the ordering

Tarjan & Yannakakis (1984): Ordering can be build from the back, selecting recursively vertex with highest number of ordered neighbors

- Without loss of generality, we can put a (maximal/maximum) clique in G at the end of the ordering

Instances

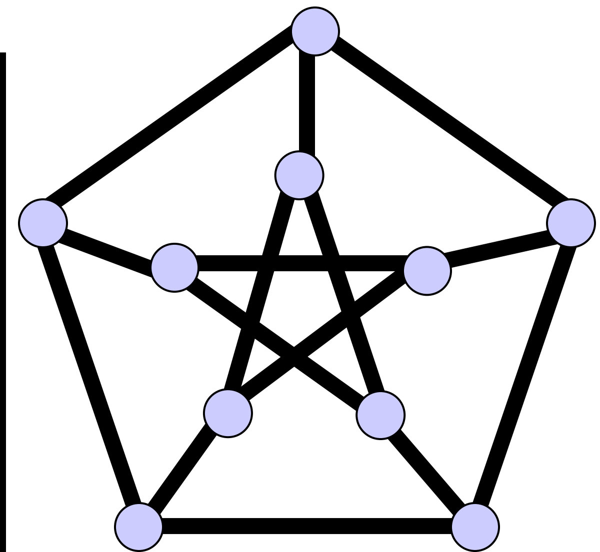
- Randomly generated partial-k-trees (Shoiket&Geiger,1998)
  - Generate k-tree
  - Randomly remove p% of the edges
  - →treewidth at most k
  - →n=100, k=10, p=30/40/50
- Instances from frequency assignment, probabilistic networks, …

Computational framework

- SCIP (http://scip.zib.de/) with CPLEX 10.0 as LP solver

Petersen graph

Objective	Strategy	CPU time (s)	B&C nodes	Gap (%)
Treewidth	none	449.18	278018	0
Treewidth	maximum clique	0.43	57	0
Fill-in	none	>3600	>886765	41.18
Fill-in	maximum clique	1.27	379	0



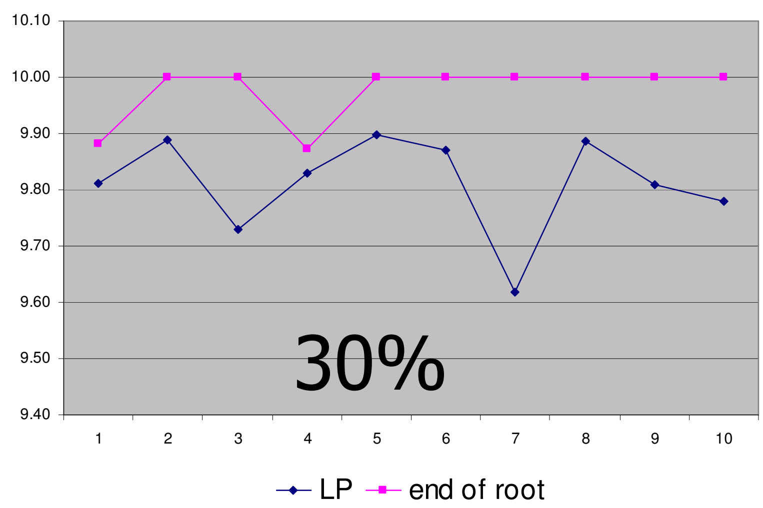
⇒ Maximum clique breaks symmetries(?); simplifies computation

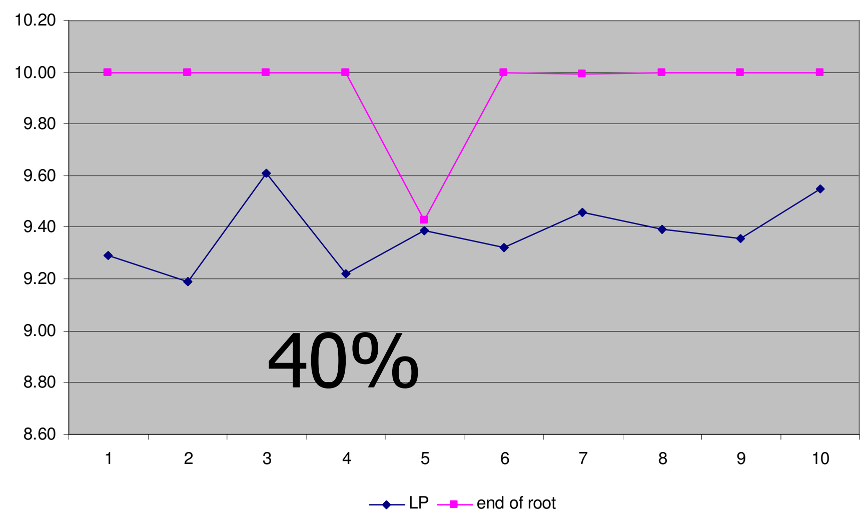
⇒ Fill-in more difficult than treewidth???

Results partial k-trees: treewidth

Treewidth

30%: 4 out of 10 solved within 1 hour CPU time
40%: 1 out of 10 solved within 1 hour CPU time





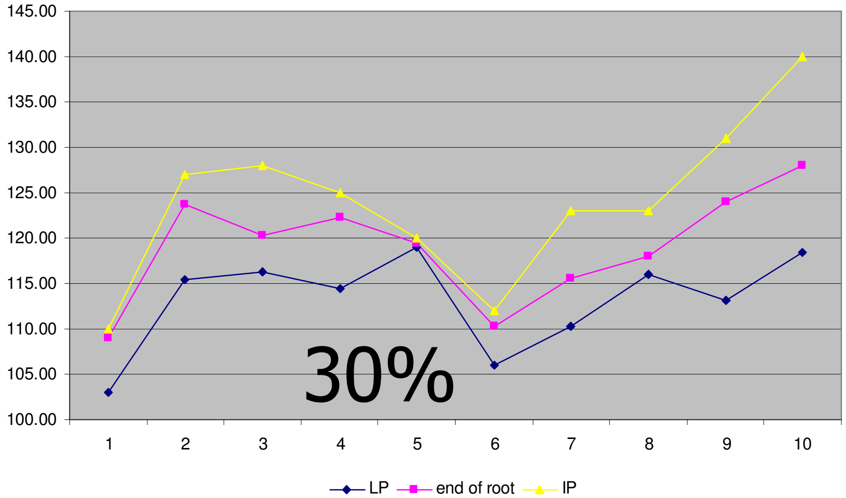
Very good lower bound, difficult to find optimal solution

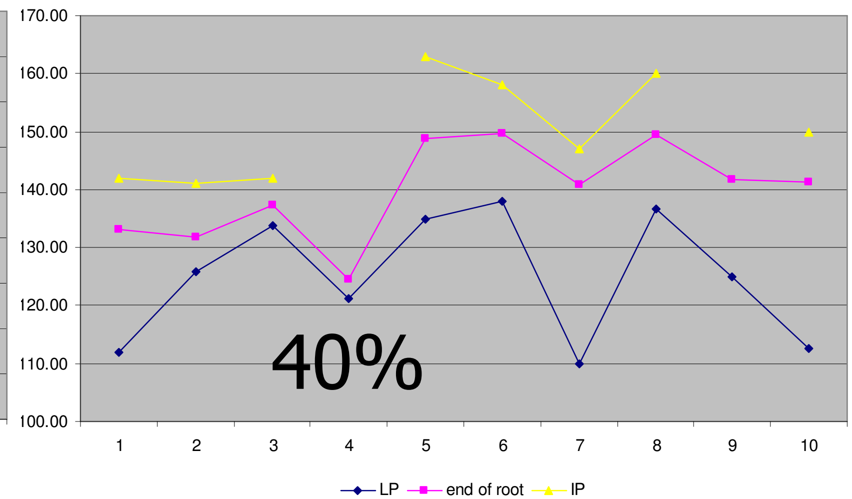
Results partial-k-trees: fill-in

Fill-in

30%: On average solved in 1085 seconds

40%: 8 out of 10 solved within 1 hour of CPU time





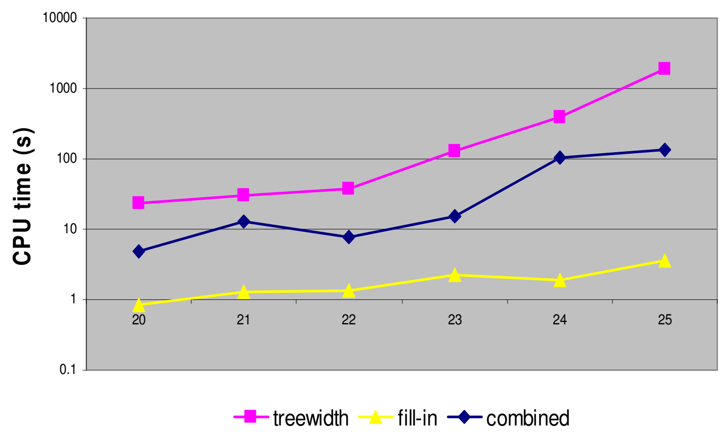
Relatively easy to solve

Arie Koster

Results realistic instances

minors of link-pp selected; ω(G)=9, tw(G)=13

				treewidth		fill-in		Combined	
instance	|V|	|E|	fi(G)	CPU(s)	#nodes	CPU(s)	#nodes	CPU(s)	#nodes
link-pp-minor-020	20	125	29	23.42	9680	0.86	2	4.88	1307
link-pp-minor-021	21	130	35	29.91	7238	1.29	9	13.15	2767
link-pp-minor-022	22	137	38	37.82	5858	1.33	1	7.88	349
link-pp-minor-023	23	144	40	128.21	16131	2.25	2	15.22	986
link-pp-minor-024	24	151	43	399.61	27125	1.93	2	103.50	8568
link-pp-minor-025	25	156	48	1875.24	94369	3.61	3	133.67	6861



$$\min z + \frac{1}{\frac{1}{2}n(n-1)-m+1} f$$

Concluding remarks

- Treewidth is moving from theory to practice; IP can help
- Chordalization polytope can tackle three problems: treewidth, minimum fill-in, and weighted treewidth
- More knowledge on chordalization polytope required, in particular for (weighted) treewidth

- To test treewidth of graphs from applications, contact me: koster@zib.de
- Publications: http://www.zib.de/koster/
- Overview of most treewidth computations: **TreewidthLIB** at http://www.cs.uu.nl/people/hansb/treewidthLIB/