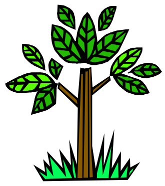
Decomposition of Graphs:
Upper bounds, Lower bounds, and Exact methods to compute Treewidth



SOFT'06

25 September 2006, Nantes

Arie Koster
koster@zib.de

Zuse Institute Berlin (ZIB), Germany
http://www.zib.de/koster/

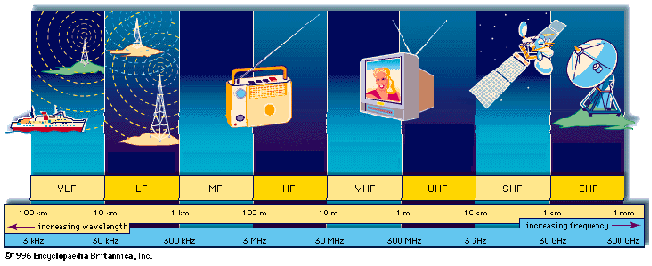
Overview

- Introduction
- Tree Decompositions
- Computing Treewidth
- Using Treewidth

3

Back in 1997 …

- Minimum Interference Frequency Assignment



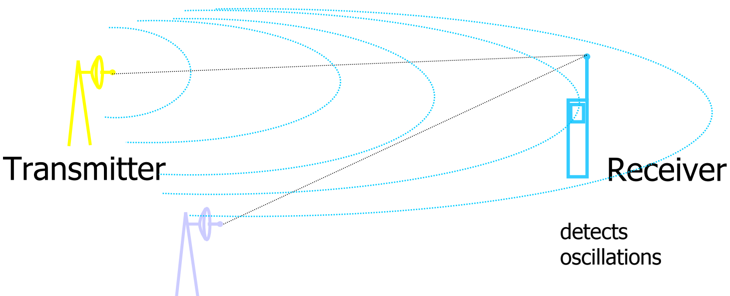
- Good upper bounds,
  neither lower bounds nor optimal solutions
- Integer linear programming does not work

→**What to do next?**

Arie Koster

4

Properties of 2G wireless communication



detects oscillations

emits electromagnetic oscillations at a frequency

Quality of the received signal: Signal-to-noise ratio

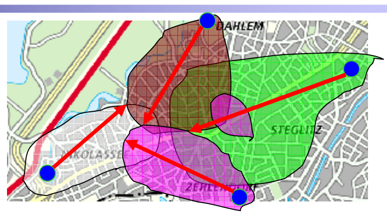
Poor signal-to-noise ratio: interference of the signal

Objective: Frequency plan without interference or, second best, with minimum interference

Arie Koster

5

Interference



Level of interference depends on

- distance between transmitters, geographical position,
- power of the signals, direction in which signals are transmitted,
- weather conditions
- assigned frequencies
  - co-channel interference
  - adjacent-channel interference

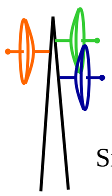
⇨ Interference is measured between pairs of transmitters

Arie Koster

6

Separation

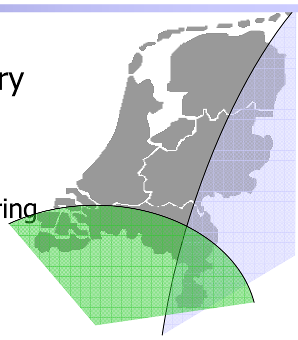
Frequencies assigned to the same location (site) have to be separated



Blocked channels

Whole spectrum is not allowed at every location:

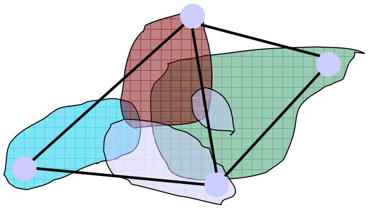
- government regulations,
- agreements with operators in neighboring regions,
- requirements military forces, etc.

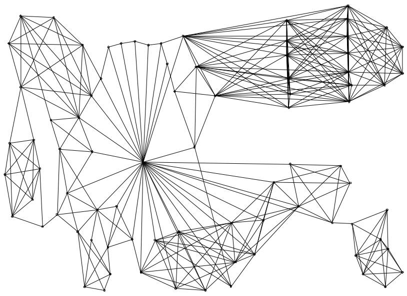


Arie Koster

7

Modeling MI-FAP





- Interference graph
- Vertices represent transmitters;
  Domain of assignable frequencies
- Edges represent constraints/interference
  Matrix with penalties for combination of frequenciesg

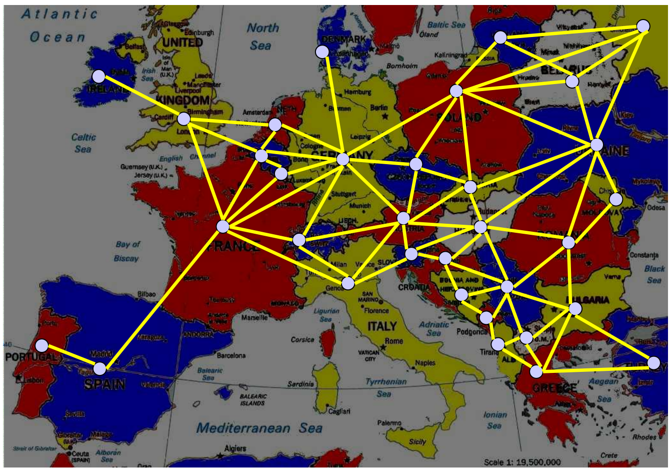
⇨ Partial Constraint Satisfaction Problem (with binary relations)

More frequency assignment: http://fap.zib.de

Arie Koster

8

Related problems - Vertex Coloring



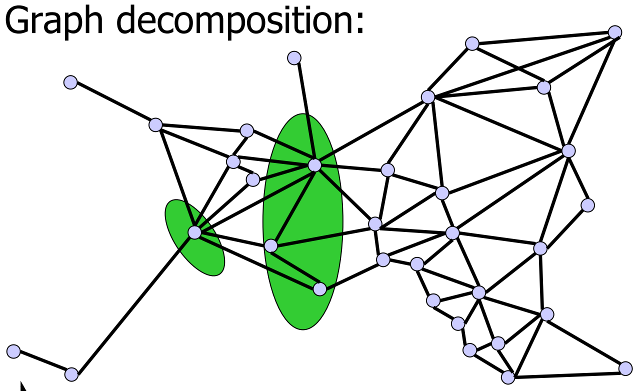
⇨ Integer Programming suffers from symmetry;
linear relaxation not strong at all

Arie Koster

9

What to do next?

- Graph decomposition:



Coloring of Western-Europe does not depend on Eastern-Europe, only on Central-Europe

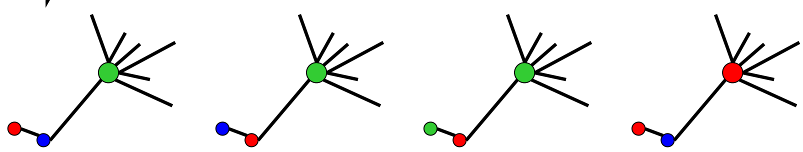
Coloring of Iberian peninsula only depends on color of France

Arie Koster

10

Graph Decomposition

Coloring of rest of Europe only depends on France



- First 3 solutions equivalent for rest of problem
- 3rd solution has lowest number of colors: preferred
- 4th solution is not equivalent for rest of problem
- Only #colors non-equivalent solutions exist

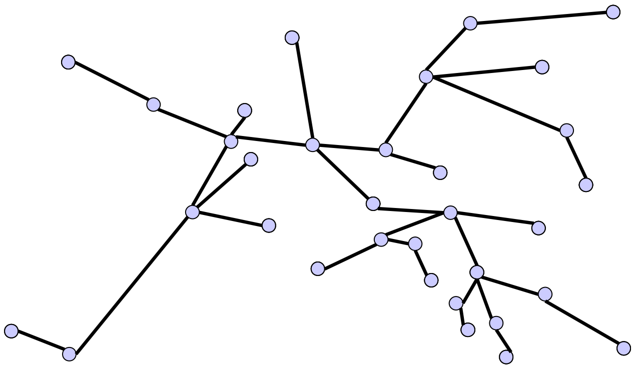
Does there exist (polynomial-time) algorithms that use this information?

Arie Koster

11

Independent Set on trees

- Repeatedly
  - Select a (all) leaf(s) of the tree in IS
  - Remove it (them) and its (their) neighbors

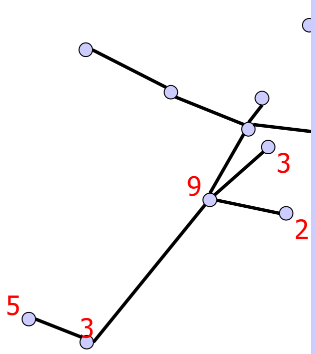


Arie Koster

12

Weighted Independent Set on trees

- Root tree at arbitrary vertex, T(v) subtree rooted at v
- A(v) = max weighted IS in T(v)
- B(v) = max weighted IS in T(v) **not** containing v



For leafs v:

- A(v)=c(v); B(v)=0

For v with children $x_1,\dots,x_k$

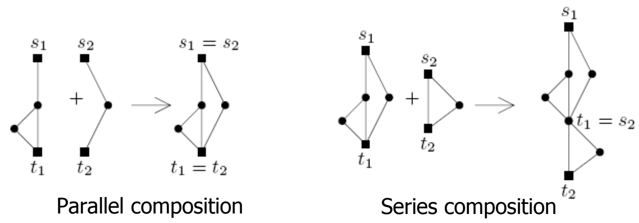
- $B(v)=A(x_1)+\dots+A(x_k)$
- $A(v)=\max\{c(v)+B(x_1)+\dots+B(x_k),B(v)\}$

B(France)=3+2+5, A(France)=9+0+0+5

Arie Koster

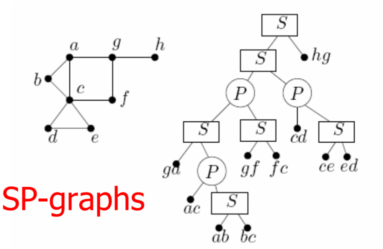
13

Beyond trees: series-parallel graphs



Parallel composition Series composition

- SP-tree: binary tree representing parallel and series composition of series-parallel graph
- leafs ~ edges
- S-nodes for series
- P-nodes for parallel
- Subtrees correspond to SP-graphs



Arie Koster

14

Weighted Independent Set on series-parallel graphs

- (G,s,t) defines SP-graph, G(i) SP-graph for subtree rooted at i
- AA(i) = max WIS containing both s and t
  AB(i) = max WIS containing s but not t
  BA(i) = max WIS containing t but not s
  BB(i) = max WIS containing neither s nor t
- Leafs: AA(i)=-∞, AB(i)=c(s), BA(i)=c(t), BB(i)=0
- Internal S-node i with children j and k (s' terminal between j,k):
  AA(i) := max{ AA(j)+AA(k)-c(s'), AB(j)+BA(k) }
  AB(i) := max{ AA(j)+AB(k)-c(s'), AB(j)+BB(k) }, …
- Internal P-node i with children j and k:
  AA(i) := AA(j)+AA(k)-c(s)-c(t)
  AB(i) := AB(j)+AB(k)-c(s), …

⇨ Generalization beyond series-parallel graphs?

Arie Koster

15

Overview

- Introduction
- Tree Decompositions
- Computing Treewidth
- Using Treewidth

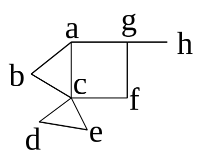
Arie Koster

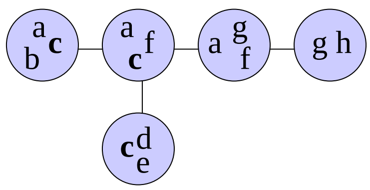
16

Definition

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



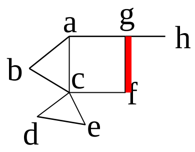


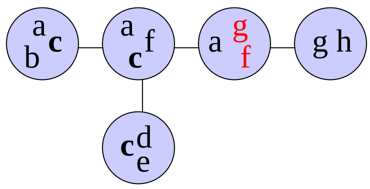
Arie Koster

17 Definition

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



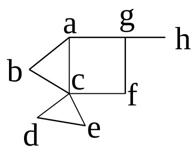


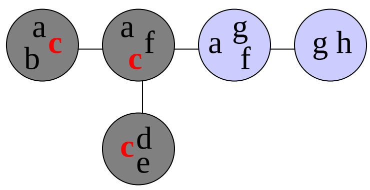
Arie Koster

18 Definition

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



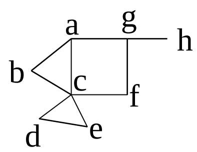


Arie Koster

Treewidth

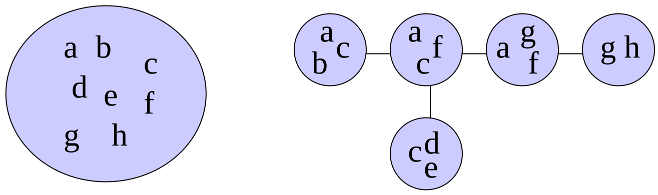
- Width of tree decomposition:

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

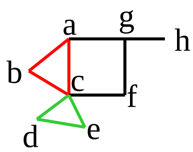


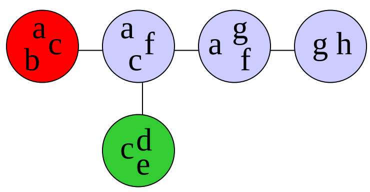
maximum bag size - 1

- 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

later more lower bounds ...

- Trees have treewidth 1

Graphs of bounded treewidth

- Graphs of bounded treewidth generalize both trees and series-parallel graphs
  - Trees have treewidth 1 v.v.
  - Series-parallel graphs have treewidth at most 2
- Treewidth measures the tree-likeness of graphs
- Concept introduced by Robertson & Seymour in 1980s in their work on graph minor theorem

Algorithms using tree decompositions

- Step 1: Find 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.
  - Root tree (I,F)
  - Let $Y_i = \cup X_j$ 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

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 IS 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

Treewidth results

- Arnborg, Lagergren and Seese (1991), based upon the work of Courcelle (1990), showed that many NP-complete problems modeled on graphs with bounded branchwidth or treewidth can be solved in polynomial time using a branch decomposition or tree decomposition of the graph.
- NP-complete problems modeled on graphs:
  - Traveling Salesman Problem
  - Disjoint Paths Problem
  - Maximum Planar Subgraph
  - General Minor Containment
  - (Partial) Constraint Satisfaction Problems

25

Branchwidth, Treewidth, Pathwidth

Robertson and Seymour [106]: For a graph $G=(V,E)$, max{ bw(G), 2 } ≤ tw(G) + 1 ≤ max{ ⌊3/2 bw(G)⌋, 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
- Pathwidth: T is restricted to be a path; tw(G) ≤ pw(G)
- Trees do not have bounded pathwidth

ZIB Arie Koster

26

Overview

- Introduction
- Tree Decompositions
- **Computing Treewidth**
- Using Treewidth

ZIB Arie Koster

Computing Treewidth in Theory

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)) time — Arnborg et al.[13]
  - O(1.9601^n poly(n)) time — Fomin et al.[57]
  - O(2.9512^n poly(n)) time, O(poly(n)) space — [ESA2006]
  - poly(n) denotes a polynomial in n

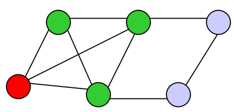
References refer to INFORMS Tutorials 2005 chapter

Computing Treewidth in Practice

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

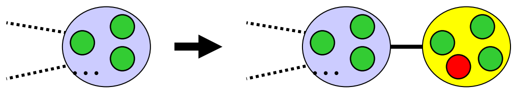
Repeatedly remove a simplicial vertex of G: $v_1,\ldots,v_n$

For i = n down to 1 do

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

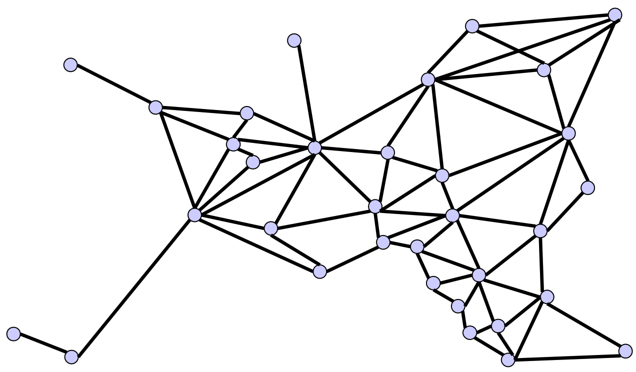
  Attach node to a node containing all neighbors of v_i in $G[v_i,\ldots,v_n]$

Return tree decomposition



Width of returned TD equals maximum clique minus 1

Example



There does not always exist a simplicial vertex in general graphs!

31

If the assumption holds:

Width of returned TD equals maximum clique minus 1

⇒ Tree Decompoisition is optimal !!!

Which graphs satisfy the assumption ?

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

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

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

⇒ Optimal algorithm for chordal graphs!

Arie Koster

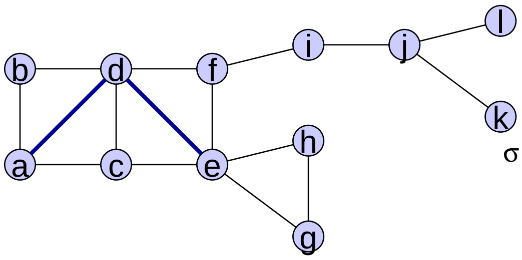
32

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.



$\sigma = [b,a,c,d,g,h,e,f,i,k,j,l]$

Arie Koster

33 Treewidth Upper bounds

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:

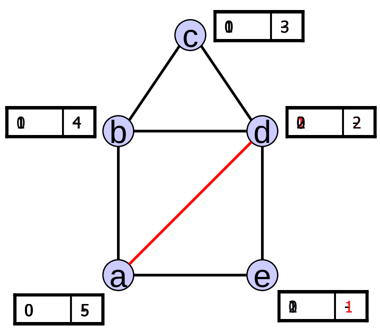
- Lexicographic Breadth First Search (LEX_M & LEX_P)
  - Rose, Tarjan & Lueker [111]
- Maximum Cardinality Search (MCS & MCS_M)
  - Tarjan & Yannakakis [120], Heggernes et al. [84]

ZIB Arie Koster

34 Treewidth Upper bounds

Maximum Cardinality Search

- MCS
  - Repeatedly select vertex with largest number of labeled neighbors



Step 0: [.,.,.,.,.]
Step 1: [.,.,.,.,a]
Step 2: [.,.,.,b,a]
Step 3: [.,.,c,b,a]
Step 4: [.,d,c,b,a]
Step 5: [e,d,c,b,a]

ZIB Arie Koster

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

Further algorithms

→Minimum separating set heuristic [83]

→Sparse Fill-In [unpublished; work in progress]

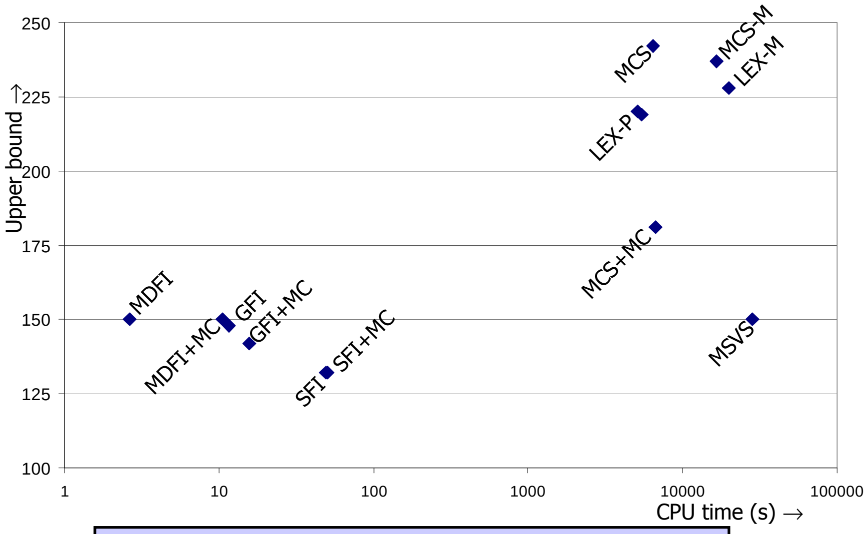
- Combination of Greedy and Minimum Degree Fill-In algorithms

→Metaheuristics

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

- Minimal Chordalization
  - Turns chordalization into a minimal one

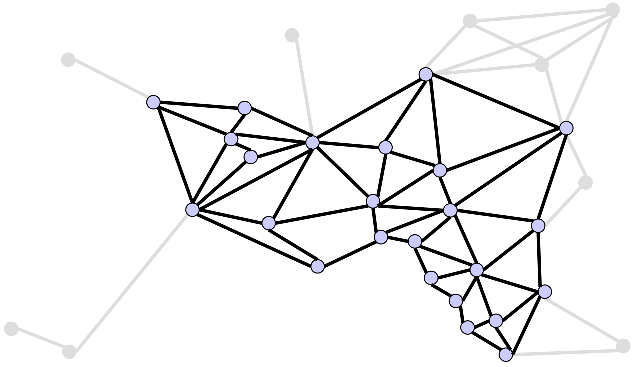
Upper bounds by example



Upper bounds for pignet2-pp (1002 vertices, 3730 edges)

Arie Koster

Computing Treewidth



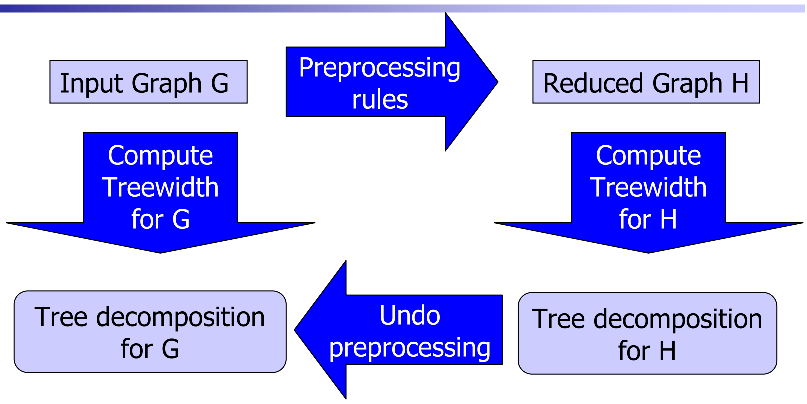
Steps so far are "as optimal as" possible!

Arie Koster

Two types of preprocessing

- Reduction rules (*Simplification*) [39]
  - Rules that change G into a smaller `equivalent' graph
  - Maintains a lower bound variable for treewidth *low*
- Safe separators (*Divide and Conquer*) [32]
  - Splits the graph into two or more smaller parts with help of a separator that is made to a clique

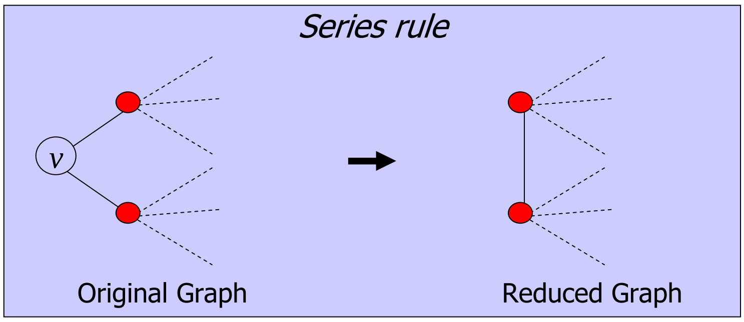
Reduction



- Safe rules that
  - Make *G* smaller
  - Maintain optimality…
- Use for preprocessing graphs when computing treewidth

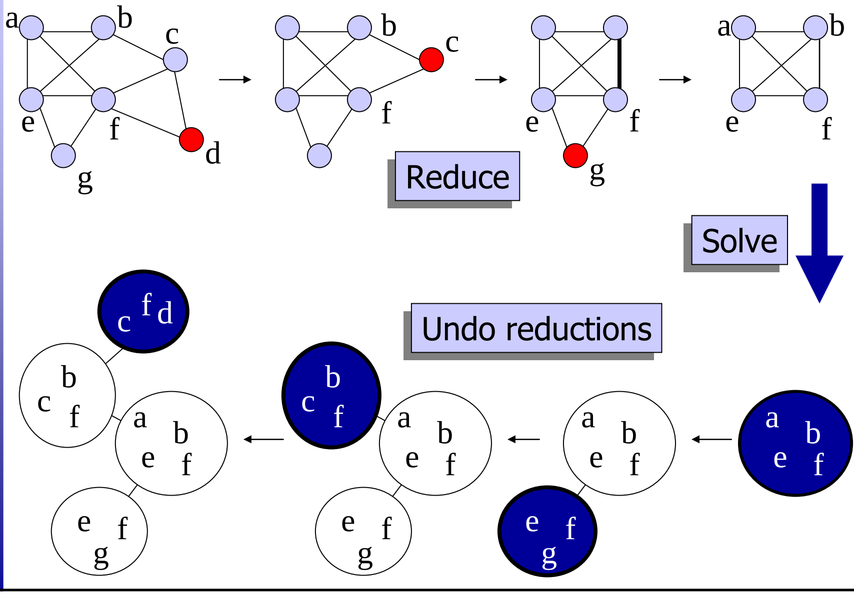
Reduction rules

- Uses and generalizes ideas and rules from algorithm to recognize graphs of treewidth ≤ 3 from Arnborg and Proskurowski
- Example: *Series rule:* remove a vertex of degree 2 and connect its neighbors



- Safe for graphs of treewidth ≥ 2

Example



43

Type of rules

- Variable: **low** (integer, lower bound on treewidth)
- Graph G
- Invariant: value of **max(low, treewidth(G))**
- Rules
  - Locally rewrite G to a graph with fewer vertices
  - Possibly update or check **low**
- We say a rule is *safe*, when it maintains the invariant.
- Use only safe rules.

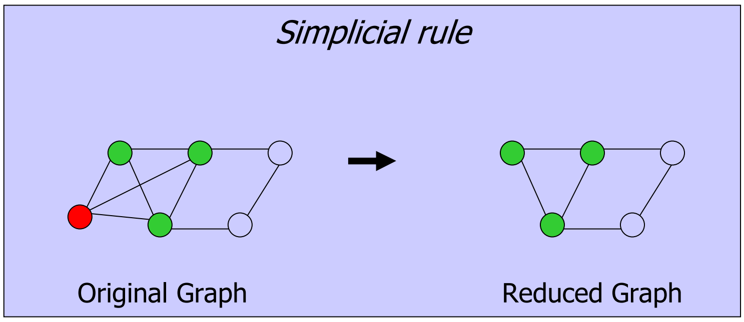
Arie Koster

44

Rule 1: Simplicial rule

- Let *v* be a simplicial vertex in *G*
- Remove *v*.
- Set *low := max (low, degree(v))*

Simplicial =
Neighbors form a clique



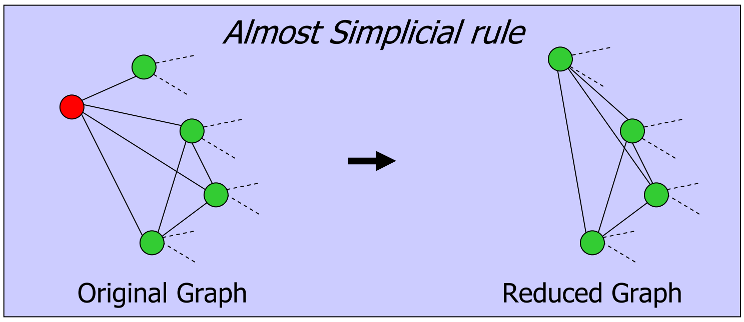
- Simplicial rule is safe.
- Special cases: islet rule (singletons), twig rule (*degree(v)* = 1)

Arie Koster

Rule 2: Almost Simplicial rule

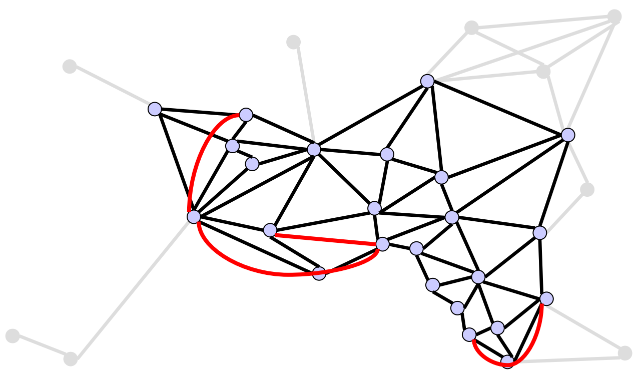
- Let v be a almost simplicial vertex in *G* and *low* ≥ *degree*(*v*)
- Remove *v*,
- turn neighbors into clique

Almost Simplicial = Neighbors except one form a clique



- Almost Simplicial rule is safe.

Example *low* = 3



47

Increasing *low* further

Further rules: buddy/buddies rule, (extended) cube rule

Arnborg and Proskurowski [12]:

- tw(G)=1 if and only if G is reduced to the empty graph by islet rule (vertices of degree 0) and twig rule (vertices of degree 1)
- tw(G)=2 if and only if G is reduced to the empty graph by islet, twig, and series rule (vertices of degree 2)
- tw(G)=3 if and only if G is reduced to the empty graph by islet, twig, series, triangle, buddy, and cube rule

Low can be increased to 2, 3, and 4 respectively if these rules cannot be applied anymore and graph is not empty yet.

ZIB Arie Koster

48

Results for probabilistic networks

	original		preprocessed		
instance	**|V|**	**|E|**	**|V|**	**|E|**	**low**
alarm	37	65	**0**	**0**	**4**
barley	48	126	26	78	4
boblo	221	328	**0**	**0**	**3**
diabetes	413	819	116	276	4
link	724	1738	308	1158	4
mildew	35	80	**0**	**0**	**4**
munin1	189	366	66	188	4
munin2	1003	1662	165	451	4
munin3	1044	1745	96	313	4
munin4	1041	1843	215	642	4
munin-kgo	1066	1730	**0**	**0**	**5**

	original		preprocessed		
instance	**|V|**	**|E|**	**|V|**	**|E|**	**low**
oesoca+	67	208	14	75	9
oesoca	39	67	**0**	**0**	**3**
oesoca42	42	72	**0**	**0**	**3**
oow-bas	27	54	**0**	**0**	**4**
oow-solo	40	87	27	63	4
oow-trad	33	72	23	54	4
pignet2	3032	7264	1002	3730	4
pigs	441	806	48	137	4
ship-ship	50	114	24	65	4
vsd	38	62	**0**	**0**	**4**
water	32	123	22	96	5
wilson	21	27	**0**	**0**	**3**

→ Some cases could be solved with preprocessing to optimality

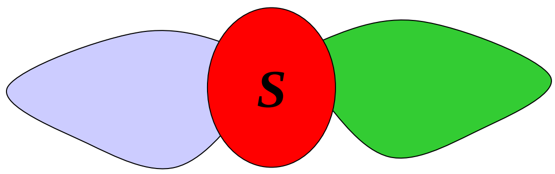
→ Often substantial reductions obtained

→ Time needed for preprocessing is small (never more than a few seconds)

ZIB Arie Koster

Graph separators

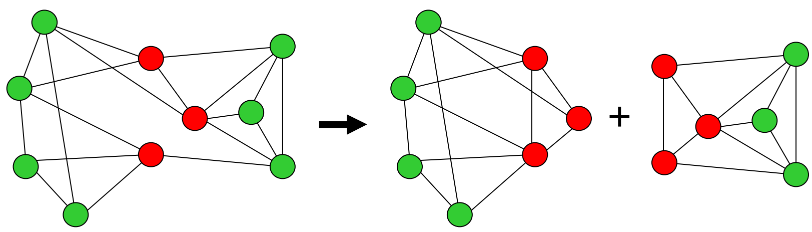
- $S \subset V$ is a *separator* of *G*, if G - S has more than one connected component
- *S* is a *minimal separator*, if *S* is a separator and *S* does not contain another separator as proper subset



Safe separator

S is safe for treewidth, or *a* safe separator if and only if the treewidth of *G* equals the maximum over the treewidth of all graphs obtained by

- Taking a connected component *W* of G - S
- Take the graph, induced by $W \cup S$
- Make *S* into a clique in that graph



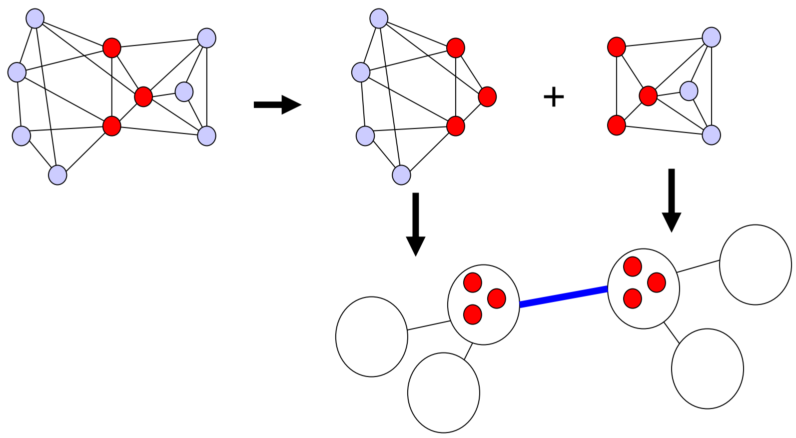
Original Graph Components

Using safe separators

- Splitting the graph for divide and conquer preprocessing
- Until no safe separators can be found
- Slower but more powerful compared to reduction
  - Most or all reduction rules can be obtained as special cases of the use of safe separators
- Look for sufficient conditions for separators to be safe

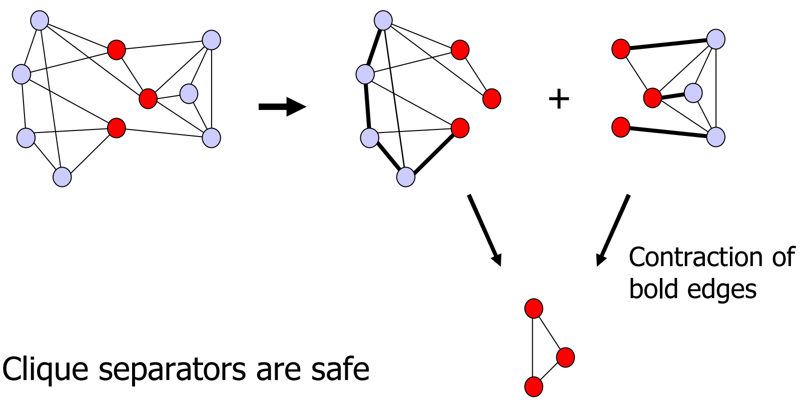
Lemma 1

Let S be a separator in G. The treewidth of G is at most the maximum over all connected components W of G of the treewidth of $G[W \cup S]$ + clique(S)



Lemma 2

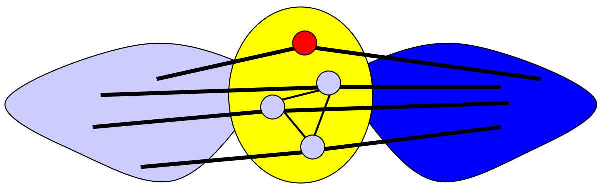
Let S be a separator. If for all components W of G-S, $G[W \cup S]$ contains a clique on S as a minor, then S is safe.



→ Clique separators are safe

→ Separators of size 0 and 1 are safe

Safeness of minimal almost clique separators



S is *almost clique* when S-v is a clique for some vertex v

- If one component is contracted to the red vertex, the separator turns into a clique: minimal almost clique separators are safe!

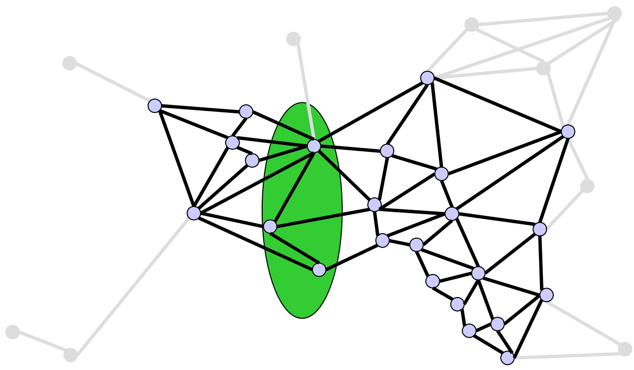
→ Minimal Separators of size 2 are safe

→ \`Almost all' minimal separators of size 3 are safe

  - only 3 independent vertices can be non-safe

- Minimal separators of size 3 that split off at least two vertices are safe

A safe separator in Europe …



Results for probabilistic networks

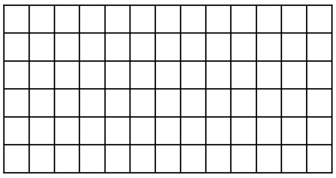
	size		separators			output			
instance	|V|	|E|	clique	almost-clique	size 3	# graphs	# cliques	# To Do	low
barley-pp	26	78	0	**7**	0	8	7	1	5
diabetes-pp	116	276	0	**85**	0	86	84	2	4
link-pp	308	1158	0	0	0	1	0	1	4
munin1-pp	66	188	0	**2**	0	3	2	1	4
munin2-pp	165	451	**6**	**13**	**4**	24	12	12	4
munin3-pp	96	313	**2**	**2**	**2**	7	4	3	4
munin4-pp	215	642	**3**	**4**	0	8	2	6	4
oesoca+-pp	14	75	0	0	0	1	0	1	9
oow-trad-pp	23	54	0	0	**1**	2	1	1	4
oow-solo-pp	27	63	0	0	**1**	2	0	2	4
pathfinder-pp	12	43	0	**5**	0	6	6	**0**	6
pignet2-pp	1002	3730	0	0	0	1	0	1	4
pigs-pp	48	137	0	**1**	0	2	1	1	5
ship-ship-pp	24	65	0	0	0	1	0	1	4
water-pp	22	96	0	**1**	0	2	1	1	6

Why Lower Bounds?

- Benchmark quality of constructed tree decompositions (upper bounds)
- Speed up of branch & bound methods (e.g. Gogate & Dechter [63])
- Indicates expected performance of dynamic programming algorithms

⇨ Very dense areas in graphs contribute to treewidth

⇨ Grid structures contribute to treewidth



tw(G)=min(n,m)

Induced subgraphs

Theorem *The treewidth of a graph can not increase by taking subgraphs*

H subgraph of *G*

$$\left.\begin{array}{r} tw(H) \leq tw(G) \\ LB(G) \leq tw(G) \end{array}\right\} \quad LB(H) \leq tw(G)$$

Corollary *If the LB can increase by taking subgraphs, an improved lower bound can be found by taking the maximum over all subgraphs:*

$$\max_{H \subseteq G} LB(H) \leq tw(G)$$

Foundations II

Theorem *The treewidth of a graph can not increase by taking minors*

H minor of *G*

$$\left.\begin{array}{l} tw(H) \leq tw(G) \\ LB(G) \leq tw(G) \end{array}\right\} \quad LB(H) \leq tw(G)$$

Corollary *If the LB can increase by taking minors, an improved lower bound can be found by taking the maximum over all minors:*

$$\max_{H \prec G} LB(H) \leq tw(G)$$

Degree-Based Lower Bounds I

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*

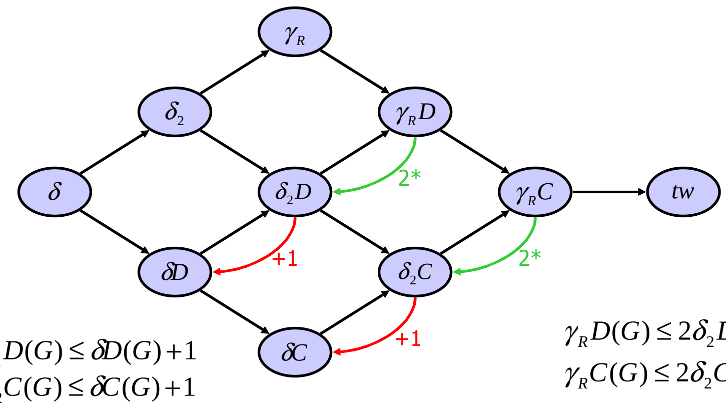
$$\delta D(G) = \max_{H \subseteq G} \delta(H) \leq tw(G)$$

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

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

Relationships

⟶ = less than or equal



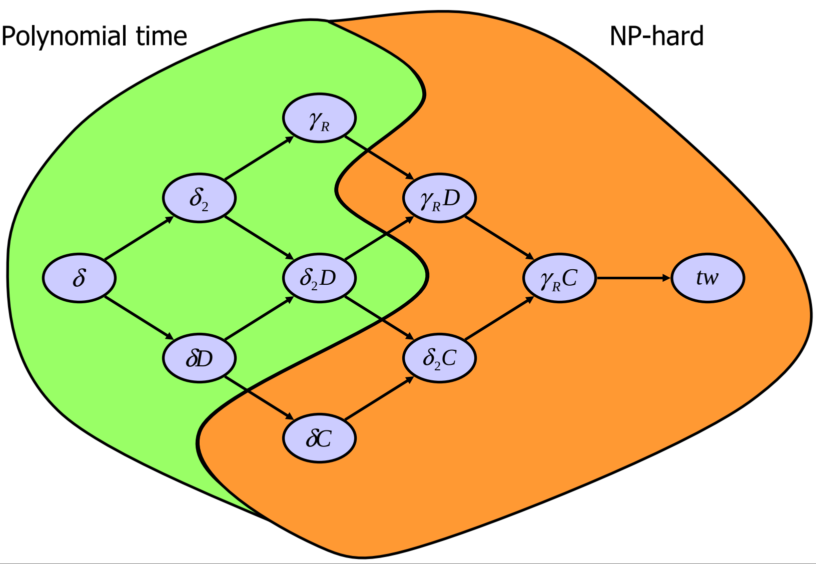
$\delta_2 D(G) \le \delta D(G) + 1$
$\delta_2 C(G) \le \delta C(G) + 1$

$\gamma_R D(G) \le 2\delta_2 D(G)$
$\gamma_R C(G) \le 2\delta_2 C(G)$

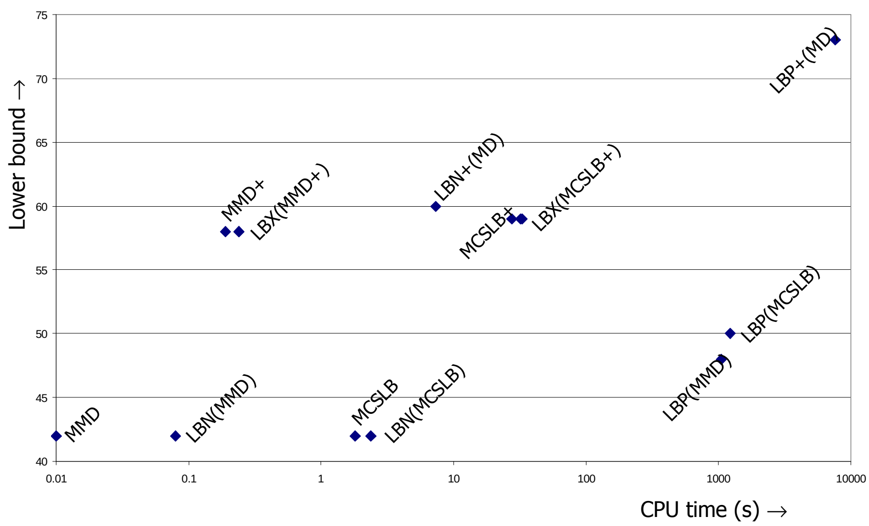
Complexity

Polynomial time

NP-hard



Lower bounds by example



Lower bounds for graph queen15-15

Planar Graphs

Theorem *Planarity is closed under taking minors*

G planar, *H* minor of *G*

$$\left.\begin{array}{l}\delta C(G) \le tw(G) \\ \delta(H) \le 5\end{array}\right\} \quad \delta C(G) \le 5$$

Theorem *The genus of G cannot increase by taking minors*

G graph of genus k, *H* minor of *G*

$$\left.\begin{array}{l}\delta C(G) \le tw(G) \\ \delta(H) \le 5+k\end{array}\right\} \quad \delta C(G) \le 5+k$$

⇨ Alternative lower bound by **Brambles** [36, ESA2005]

Brambles

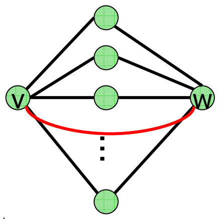
- tw(n x n grid) = n
- Search for n x n grids as minor of G
- Two different algorithms
  - General graphs:

    BFS + connectivity closure; max disjoint paths
  - Planar graphs:

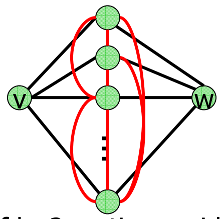
    Partition outer face; max disjoint paths in north-south, west-east
- Robertson, Seymour, Thomas '94: every planar graph of treewidth k has a ck x ck grid as minor

⇨ 2nd algorithm gives a constant approximation for treewidth on planar graphs!

Lower bounds by obstruction



Assume:
k neighbors
tw(G)≤k-2



Clique of k+2 vertices: width≥k+1

⇨ Edge {v,w} must be in chordalization

⇨ Lower bound by Clautiaux et al.:

Compute initial LB

Repeat

    Assume tw(G) ≤ LB; Add edges by argument above

    Compute new LB'

    If LB' > LB, LB := LB+1

Until LB' ≤ LB; return LB

67

Exact methods

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

Let **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!

ZIB Arie Koster

68

Chordalization polytope

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

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

Existence of edges

$$y_{vw} + y_{wv} = 1 \quad vw \in E$$
$$y_{vw} + y_{wv} \leq 1 \quad vw \notin E$$

Simplicity of vertices

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

ZIB Arie Koster

69 Treewidth Exact

Chordalization polytope

Ordering of vertices

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

$$\omega(H) = \max_{i=1,\ldots,n} \left|N_{H[v_i,\ldots,v_n]}(v_i)\right| + 1$$

$$tw(H) = \omega(H) - 1 = \max_{i=1,\ldots,n} \left|N_{H[v_i,\ldots,v_n]}(v_i)\right|$$

Treewidth

$$\min\left\{\max_{v\in V} \sum_{w\neq v} y_{vw} : y \in C\right\}$$

Chordalization polytope

Arie Koster

70 Treewidth Exact

Objectives

Treewidth

$$\min \quad z$$
$$s.t. \quad z \ge \sum_{w\neq v} y_{vw} \quad v \in V$$

Fill-in

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

Weighted Treewidth

$$\min \quad w$$
$$s.t. \quad w \ge \log(c_v) + \sum_{w\neq v} \log(c_w) y_{vw} \quad v \in V$$

$$y \in C$$ Chordalization polytope

Arie Koster

Separation of ordering inequalities

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

- 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

Cliques

- Ordering represents a chordal graph

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

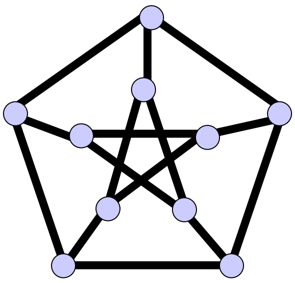
- Without loss of generality, we can put an arbitrary vertex at the end of the ordering

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

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???

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

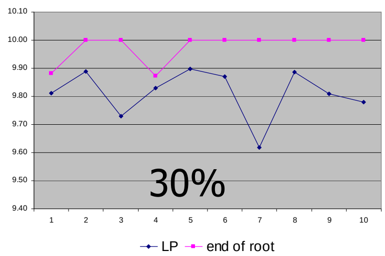
- SCIP (http://scip.zib.de/) with CPLEX 10.0 as LP solver

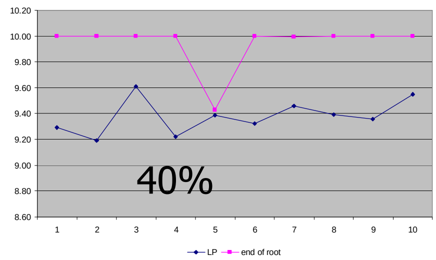
75

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

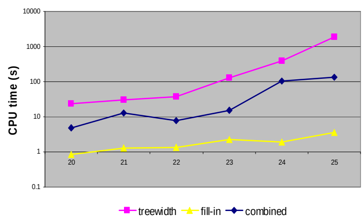
Arie Koster

76

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

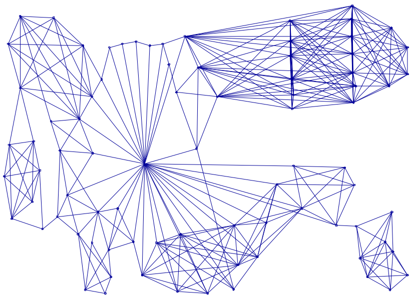


$$\min z + \frac{1}{\frac{1}{2}n(n-1)-m+1} f$$

Arie Koster

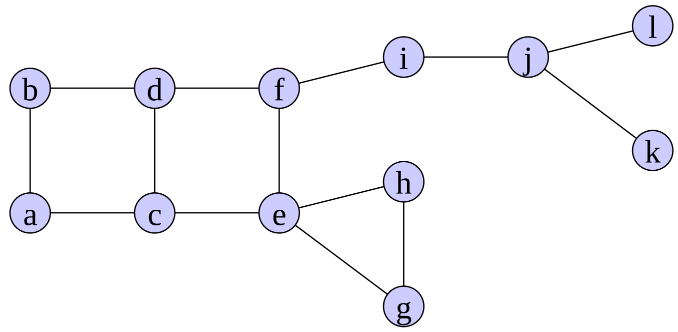
Overview

Minimum Interference FAP



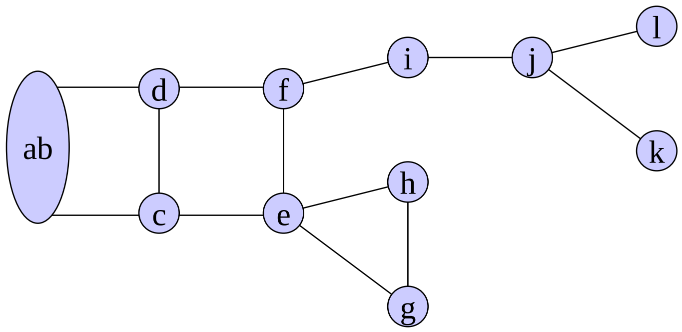
- Graph G=(V,E)
  - Vertices correspond to bi-directional connections
  - Edges indicate interference between two connections
- For every vertex v, set of frequency pairs D(v) is specified
- Interference quantified by edge penalties p(v,f ,w,g)
- Preferences for frequencies quantified by penalties q(v,f)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

Dynamic Programming Algorithm



Contract vertices according to tree-decomposition.

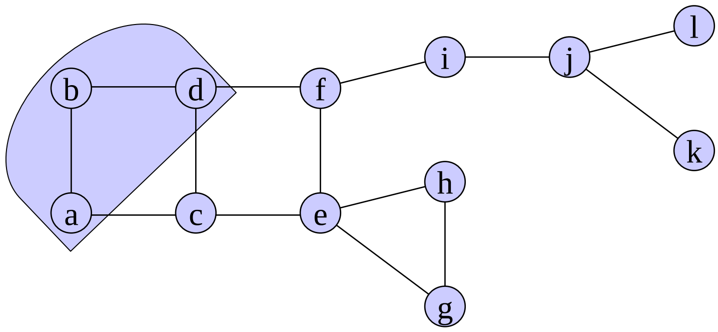
Dynamic Programming Algorithm



Contract vertices according to tree-decomposition.

$$D_{ab} = D_a \times D_b$$

Dynamic Programming Algorithm



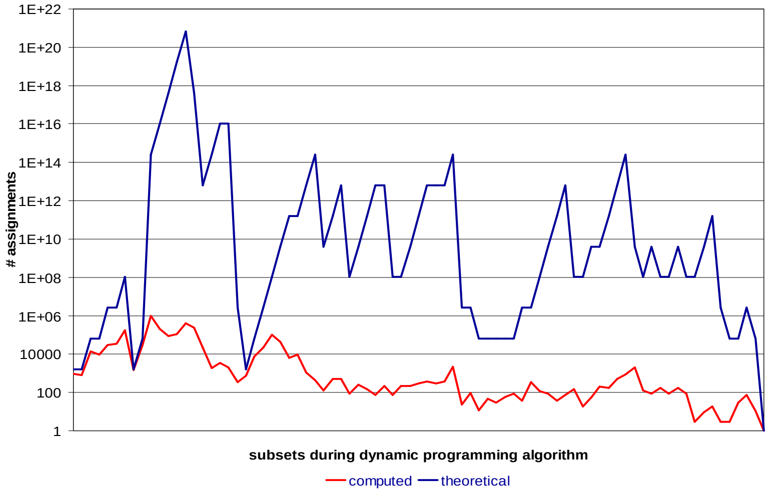
Contract vertices according to tree-decomposition.

$D_{abd} \subset D_{ab} \times D_d$ vertex *b* is not connected with rest of the graph.

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



Results Tree Decomposition

Instance	LP	QP	CSP	Tree Decomposition		Upper Bound
				Preprocessing	DP	
CELAR06	5	-	3389	0	3389	3389
CELAR07	5	-	-	0	-	343592
CELAR08	-	-	-	0	-	262
CELAR09	-	14969	-	11391	15571	15571
CELAR10	-	31204	-	31516	Solved	31516
GRAPH05	-	-	-	221	Solved	221
GRAPH06	-	-	-	4112	4123	4123
GRAPH07	-	-	-	4324	Solved	4324
GRAPH11	-	-	-	2553	-	3080
GRAPH12	-	-	-	11496	11827	11827
GRAPH13	-	-	-	8676	-	10110

Further results

- CALMA benchmarks:
  - For 7 of the 11 instances optimal solution found
  - For the other 4 instances lower bounds in the range 57.3% to 98.2% of the upper bound
- Tree Decomposition can be used to solve optimization problems in practice
  - Application to other optimization problems

Open problems

- Is TREEWIDTH polynomial for planar graphs ?
- Is TREEWIDTH NP-hard for planar graphs ?
- Does there exist (practical) integer programming formulations for computing treewidth?
- How good can the contraction degeneracy be in general graphs (as lower bound for tw(G)) ?
- Do other heuristics than MCS have a lower-bounding counter-part ?

87

Open problems

Which optimization problems
can be solved in practice with
Graph Decomposition-based algorithms

?

Arie Koster

88

Further reading

- *Branch and Tree Decomposition Techniques for Discrete Optimization*, INFORMS TutORials in Operations Research Series, Chapter 1, 2005 (with Illya Hicks, E. Kolotoğlu)
- *Combinatorial Optimisation on Graphs of bounded Treewidth*, The Computer Journal, 2006, to appear (with H. Bodlaender)
- *Solving Partial Constraint Satisfaction Problems with Tree Decomposition*, Networks 40, 2002 (with S. van Hoesel, A. Kolen)
- *Lower Bounds for Minimum Interference Frequency Assignment Problems*, Ricerca Operativa 30, 2000 (with S. van Hoesel, A. Kolen)
- *Pre-processing rules for triangulation of probabilistic networks*, Computational Intelligence 21, 2005 (with H. Bodlaender, F. van den Eijkhof)
- *Safe Separators for Treewidth*, Discrete Mathematics 306, 2006 (with H. Bodlaender)
- *Contraction and Treewidth Lower Bounds*, Journal of Graph Algorithms and Applications 10/ ESA 2004, LNCS 3221 (with H. Bodlaender, T. Wolle)
- *Treewidth Lower Bounds with Brambles*, ESA 2005, LNCS 3669 (with H. Bodlaender, A. Grigoriev)
- *On Exact Algorithms for Treewidth*, ESA 2006, LNCS 4168 (with H. Bodlaender, F. Fomin, D. Kratsch, D. Thilikos)
- *On the Chordalization Polytope and Treewidth*, in preparation
- http://fap.zib.de http://www.zib.de/koster/ http://www.cs.uu.nl/people/hansb/treewidthLIB/ koster@zib.de

Arie Koster