GROW: Graph classes, Optimization, and Width parameters

Andrzej Proskurowski
(with a little help from my friends)

Overview

- Ancient history
- Why GROW?
  - Parsing structure of graphs
  - Width parameters of graphs
  - Algorithms: Dynamic Programming
- Concise description of graph classes: Obstructions

(Ancient) History

A series of meetings on the subject resulted in Special Issues of *Discrete Applied Mathematics:*

- 1988 meeting in Eugene *DAM 54(2-3)*: *Efficient Algorithms and Partial k-trees*, Arnborg, Hedetniemi, Proskurowski, Eds.

- 2001 meeting in Barcelona *DAM 145(2)*: *Structural decompositions, width parameters and graph labelings*, Kratochvil, Proskurowski, Serra, Eds.

- 2005 meeting in Prague *DAM 157(12)*, *Second Workshop on Graph Classes, Optimization, and Width Parameters*, Kratochvil, Proskurowski, Serra, Eds.

Past GROW meetings

- 2007 3rd GROW in Eugene: *DAM 158(7)*, *Third Workshop on Graph Classes, Optimization, and Width Parameters*, Heggernes, Kratochvil, Proskurowski, Eds.

- 2009 4th GROW in Bergen: *DAM 160(6)* Heggernes, Kratochvil, Proskurowski, Eds.

- 2011 5th GROW in Daejon (in press)

- 2013 6th GROW in Santorini

Participants in Santorini GROW'13

Planned conferences:

- 2015 7th GROW in Banff
- 2017 8th GROW in Montpellier
- …

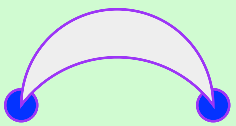
Parsing structure of graphs

- Structure of graphs:
  - Graph grammars
  - Hierarchical graphs
  - 2-structures
  - Modular decomposition
- Parsing of graphs (construction - recognition)
  - Series-parallel graphs
  - Complement-reducible graphs aka. Cographs
  - ABC-graphs
  - Partial *k*-trees

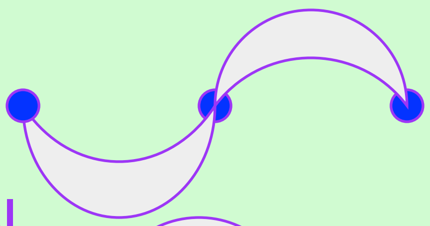
An example

- Series-parallel (sp-)graphs:
  - Start with an edge

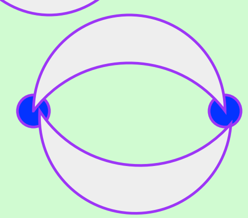
  - Assume an sp-graph



  - Combine it with another sp-graph
    - In series



    - Or parallel



Width Parameters of Graphs

- Tree- (path-) Decompositions
  - Treewidth: partial k-trees
  - Pathwidth: partial k-paths

  - Branchwidth,

  - Cliquewidth

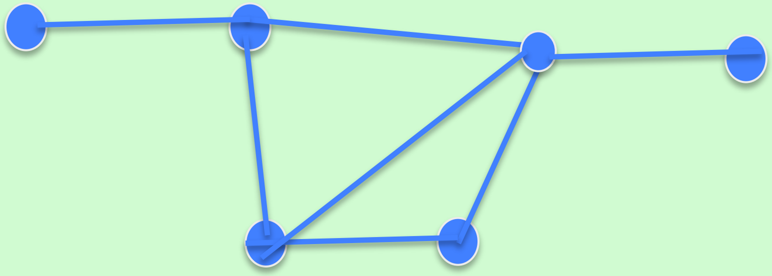
  - Rankwidth
  - Linear rankwidth

(Cubic) Tree Decomposition TD

- A cubic tree (internal nodes of degree 3) with leaf nodes labeled by elements of the graph
- Each tree branch partitions the graph elements into two blocks defined by the sets of disconnected leaves; evaluate the width function on this partition
- Maximum valuation (over all branches) determines the width of the decomposition
- The width of the graph is a minimum width over all decompositions.

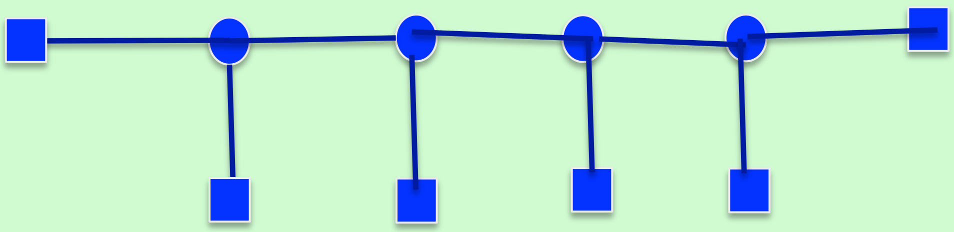
Rankwidth

- Leaves of the TD tree are labeled by vertices of the graph
- Width of a branch is the rank of the adjacency matrix of the partition

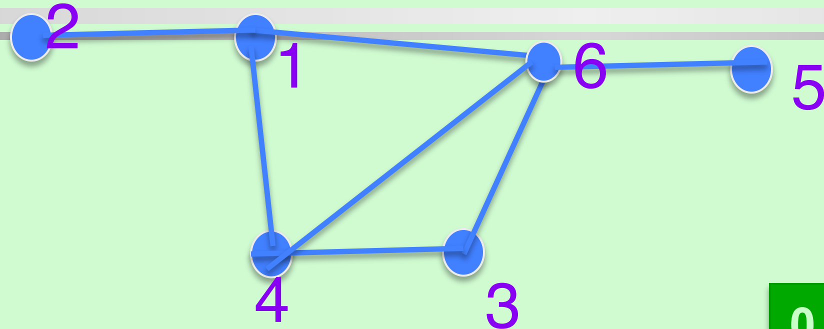


(linear width)

- tree of TD has linear structure: a caterpillar



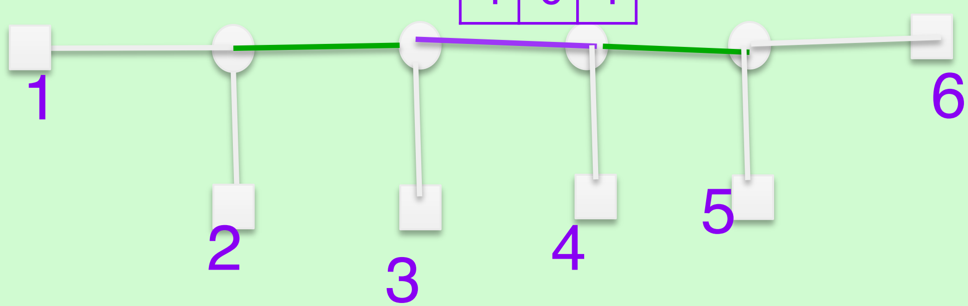
(example of width)



0	1	0	1
0	0	0	0

1	0	1
0	0	0
1	0	1

0	1
0	0
0	1
0	1



Obstructions: Concise Description of Graph Classes

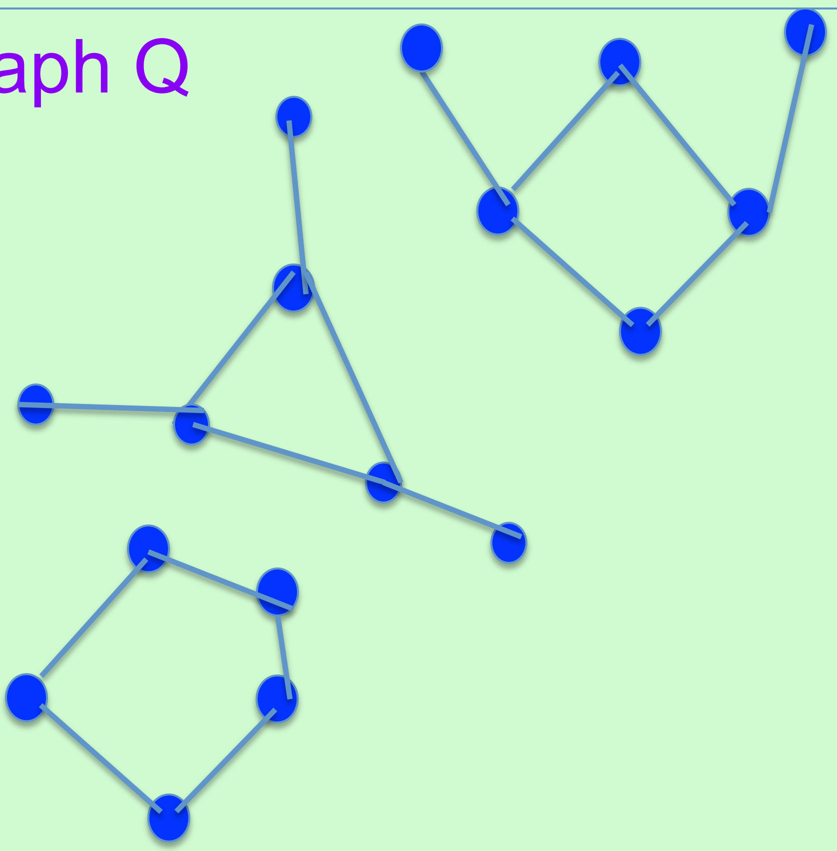
- Classes closed under embedding operation
  - Induced subgraph
  - Topological
  - Minor
- Minimal graphs outside the class of interest
- Examples of (minor) obstructions
  - Planar graphs: $\{K_5, K_{3,3}\}$
  - Treewidth 3 graphs: $\{K_5, 2W_4, M_8, P_{10}\}$
  - Linear rankwidth 1: $\{C_5, N, Q\}$

Embeddings: Guest into Host

- Mapping of elements of G into elements of H
- Embeddings of a graph G in H
  - Topological: edges of G into internal vertex-disjoint paths of H
  - Minor: vertices of G into connected subsets of vertices of H
  - Vertex minor: vertices of G into vertices of H, modulo local equivalence

Obstructions to Linear Rankwidth 1

- (half) cube graph Q
- net graph N
- cycle C_5

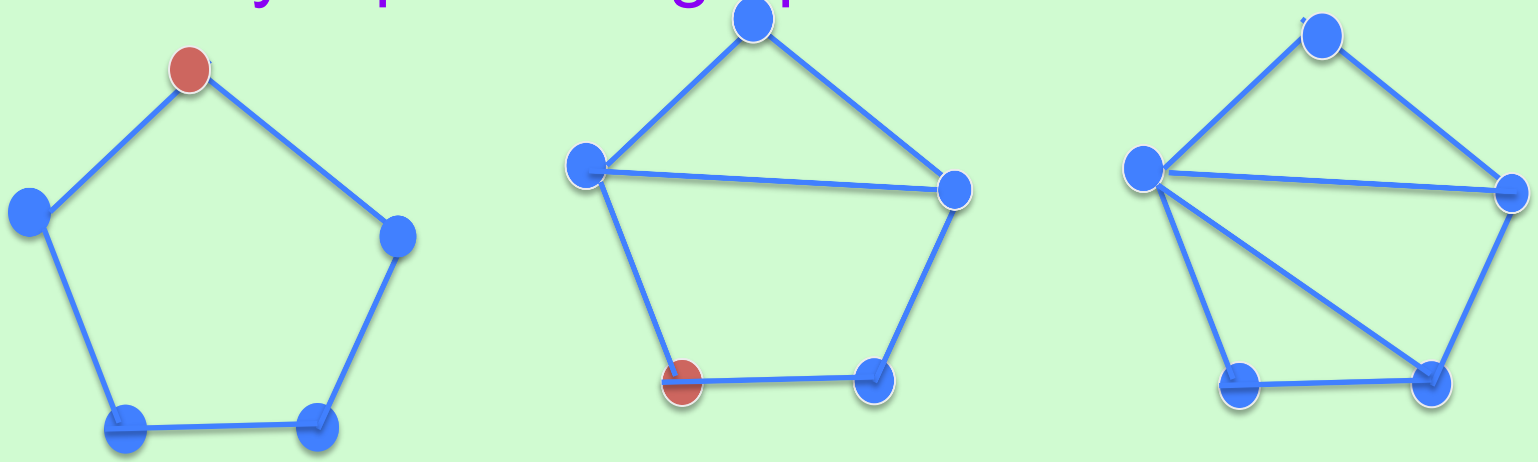


Vertex Minors

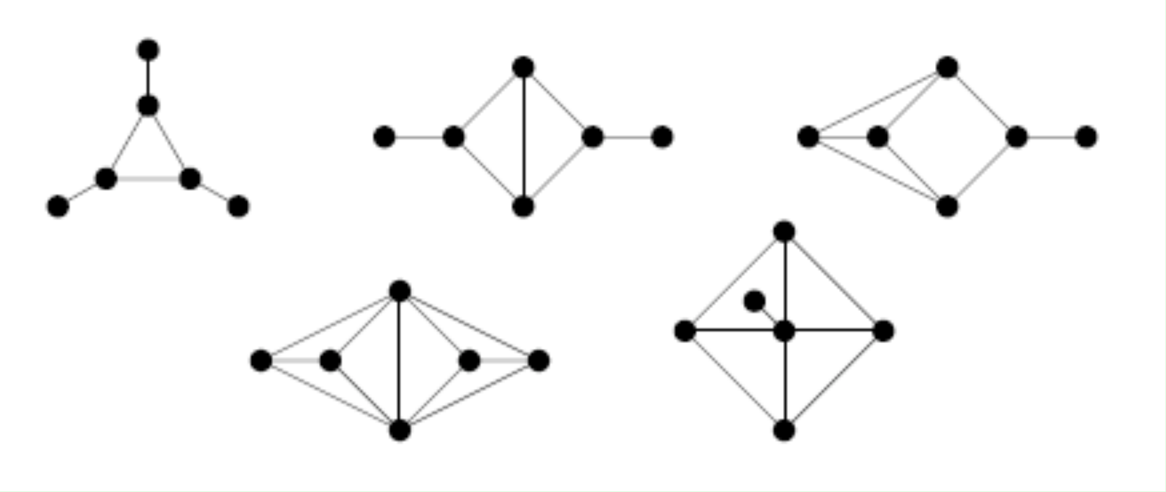
- Vertex minor: vertices of G into vertices of H, modulo local equivalence
- Local equivalence, G~G$_*$v, where $_*$v denotes
  - L*ocal complementation* at vertex v of G: complementing adjacencies of the neighborhood of v in G.

Local complementation

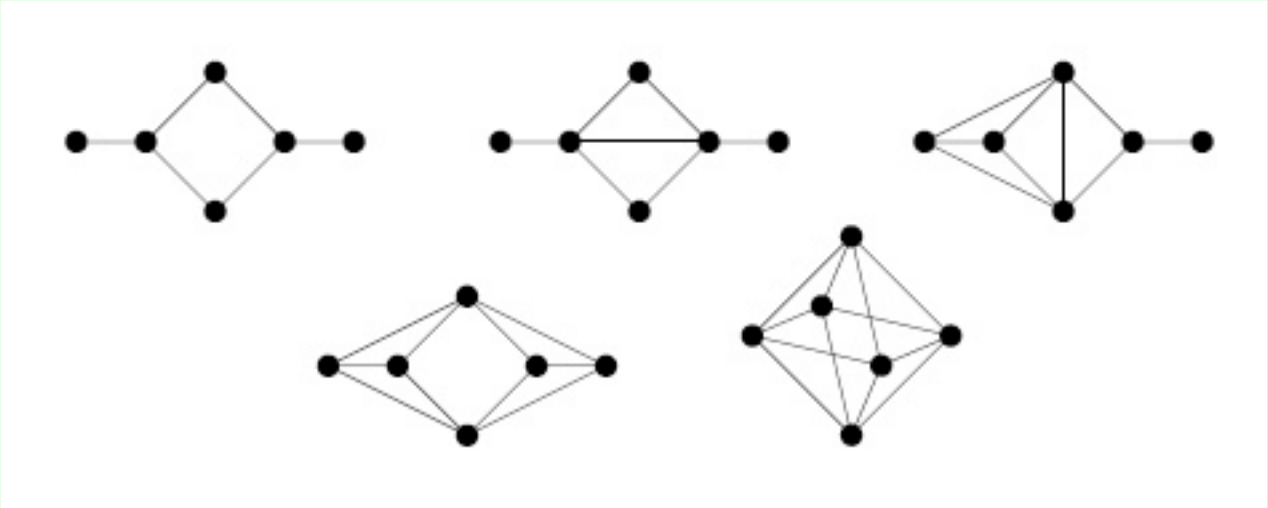
- Locally equivalent graphs



Graphs locally equivalent to N



Graphs locally equivalent to Q



(Optimization) Algorithms

- Recursive structure of solutions:
  - Optimal solution is a function of optimal solutions to smaller (sub-) problems
- Dynamic Programming
  - A bottom-up traversal of the tree of sub-problems
  - Representative solutions to be used recursively
- Tree Decomposition guides DP algorithm
  - The width of the input graph determines size of sub-problem solutions that need to be kept

Stages of complexity

Everybody knows that NP-completeness most probably implies exponential complexity (some say that it stands for "not polynomial", a subtle joke)

A recent hierarchy of complexity classes is W-hierarchy which includes "fixed parameter tractable" (FPT) problems on the lowest level

Fixed Parameter Tractability

- In general, the time complexity of an algorithm acting on input with length n and a parameter (say, treewidth) k is O($f(n,k)$)
- For a fixed k, this may be polynomial (in n) even though k may be in the exponent, $n^{g(k)}$
- Of course, we would prefer k <u>not</u> in the exponent, as in $f(n,k)=h(k)n^c$
- While $h(k)$ is often hyper-exponential, width-based algorithms are often linear ($c=1$)

That's all folks!