Multi-Clique-Width, a Powerful New Width Parameter

Martin Fürer
Pennsylvania State University

Why tree-width?

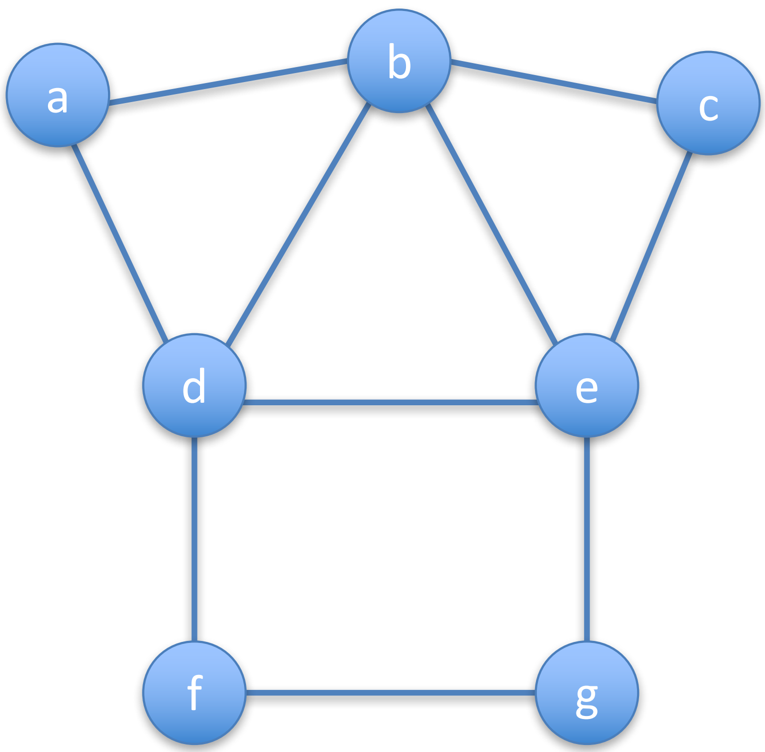
- Many combinatorial graph problems are NP-hard.
- Usually, they are easy for trees.
- One wants to extend feasibility to a somewhat more general classes of graphs.
- The tree-width measures similarity to trees.
- Low tree-width often implies efficient algorithms.

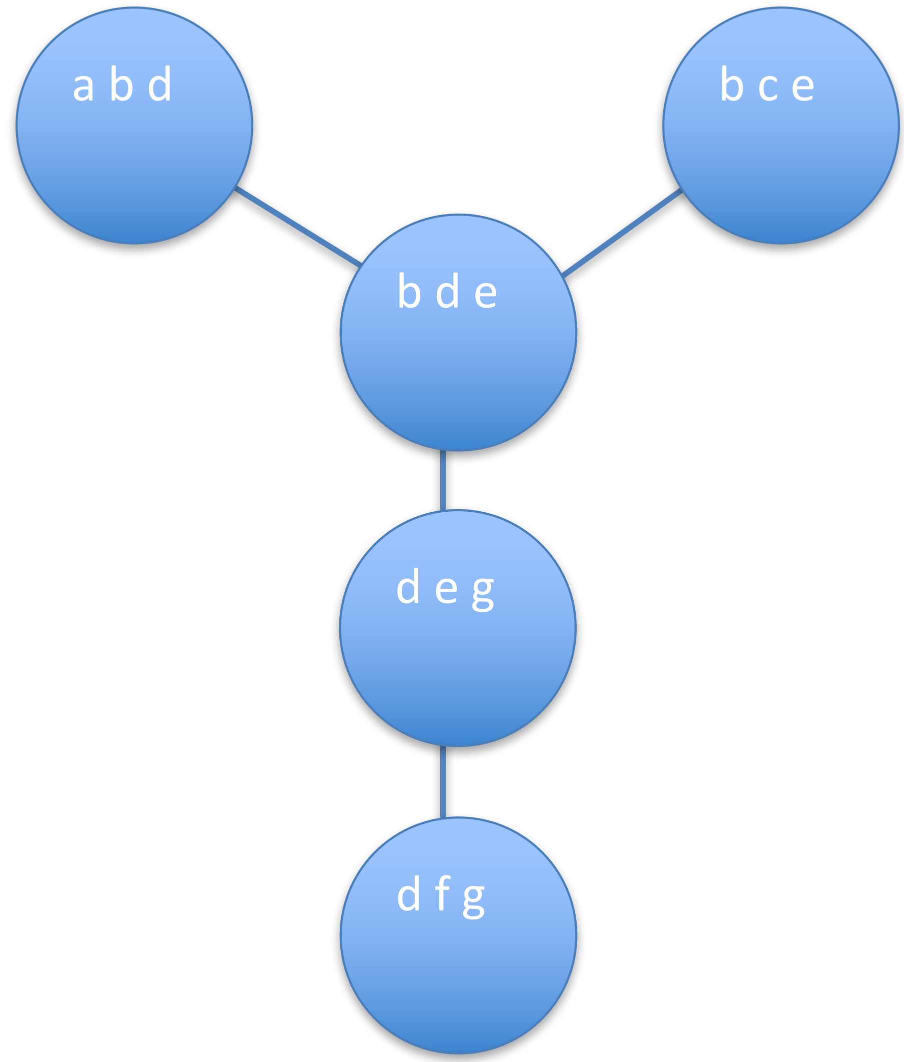
Tree decomposition

Tree decomposition of $G=(V,E)$:

- A tree with a bag X_i associated with every node i.
- Each vertex $v \in V$ belongs to at least one bag X_i
- For each edge $e=\{u,v\} \in E$, $\exists X_i \; \{u,v\} \subseteq X_i$
- For each vertex $v \in V$, the bags containing v are connected.

A graph with tree-width k=2





Tree-width

Tree-width tw(G): Smallest k, having a tree decomposition with all bags of size ≤ k +1.

There are many efficient algorithms for graphs of small tree-width.

What does "efficient" mean here?

Fixed-Parameter Tractable (FPT)

- A problem is fixed-parameter tractable with respect to a parameter k, if instances with size n and parameter k can be handled in time $f(k)\ n^{O(1)}$ for any computable function f.
- This is much better than XP, where the time is $n^{f(k)}$.
- Both are polynomial time for bounded k.
- Many NP-hard problems are FPT with respect to tree-width.

Semi-smooth tree decomposition

Def: A semi-smooth tree decomposition is a rooted tree decomposition where the bag X_i of every node i contains exactly 1 vertex that is not in the bag of the parent node. For rooted trees T with $v \in X_i \setminus X_{p(i)}$ for p(i) being the parent of i, we say that node i is the home of vertex v.

Example: Maximum Independent Set (MIS)

- Dynamic programming:
- Bottom-up in the tree, for every subset S of the vertices in a bag of i, determine the size of a MIS in the subgraph induced by vertices in the subtree of i containing exactly the vertices of S from the bag of i.
- Time: $O(2^k n)$.
- Fixed parameter tractable (FTP).
- Courcelles (1993) theorem: Linear time FPT for all Monadic Second Order properties of vertices and edges.

We want other graph classes

- Bounded tree-width graphs are sparse.
- Most problems are easy for simple dense graphs like K_n or K_{pq}.
- Expand to a nice class?
- Intuitive property: Easily formed by adding all edges between two sets of vertices.
- Clique-width measures the complexity of such constructions.

k-expression defining a labeled graph

- Label set = [k] ={1,2,…,k}.
- Operations:
  - i(v) create vertex v with label i.
  - $\eta_{i,j}$ create edges between all vertices labeled i and j (for i≠j).
  - $\rho_{i\to j}$ change all labels i to j.
  - $\oplus$ disjoint union (binary operation)
- At the end, forget the labels.
- Clique-width cw(G) = smallest number of labels that can produce G.
- E.g., a clique of any size has clique-width 2.

Meta-theorem

Courcelle, Makowsky, Rotics 2000:

Monadic second order properties of vertices (with edge relation) are FPT with the parameter being the clique-width.

Tree-width versus clique-width

- K_n has clique-width 2, but tree-width n-1.
- Bounded tree-width implies bounded clique-width (Courcelle, Olariu 2000).

  (Non-trivial, as the definitions are very different.)
- Tree-width k implies clique-width $\leq 3 \cdot 2^{k-1}$.
- There are graphs with tree-width k and clique-width $\geq 2^{(k-3)/2}$ (Corneil, Rotic 2006).

Unsatisfactory (to me)

- Complicated relationship between tree-width and clique-width, even though bounded tree-width implies bounded clique-width.
- Want better understanding of this relationship.

Multi-clique-width

- Defined like clique-width, but with every vertex allowed to have any subset of labels.
- Just as natural as clique-width.
- Much more powerful and still easy to use for algorithm design.
- Still bounded tree-width implies bounded multi-clique-width, but without exponential blow-up:

  $\mathrm{mcw}(G) \leq \mathrm{tw}(G) + 2.$
- Naturally, $\mathrm{mcw}(G) \leq \mathrm{cw}(G)$.
- For some classes of graphs, the multi-clique-width is exponentially smaller than the clique-width.

Definition of multi-clique-width

- Multi-k-expression
- Label set = [k] ={1,2,…,k}.
- Operations:
  - $m\langle i_1,\ldots,i_j\rangle$: Create m new vertices with label set $\{i_1,\ldots,i_j\}$.
  - $\eta_{i,j}$: Create edges between all vertices labeled i and j. (Allowed when no vertex has label i and label j.)
  - $\rho_{i\to S}$: Replace label i by the set S of labels.
  - ε_i: Delete the label i from all vertices. (Special case of $\rho_{i\to S}$.)
  - $\oplus$: Disjoint union.
- Multi-clique-width mcw(G) = smallest number of labels that can produce G.
- At the end forget the labels.
- The multi-k-expression defines its parse tree.

Basic Properties

- $mcw(G) \le tw(G) + 2$.
  - Top down, assign numbers from [k+1] to the vertices, such that all numbers in any bag are distinct.
  - Handle a semi-smooth decomposition tree bottom up:
  - At the home of vertex v, create v in an auxiliary leaf.
  - v's labels are k+2 and the numbers assigned to neighboring vertices in the home bag of v.
  - If i is the number assigned to v, create all edges between label i and label k+2,
  - i.e., connect v to all neighbors that have already been constructed.
  - Delete labels i and k+2.

- $mcw(G) \le cw(G) \le 2^{mcw(G)}$.
  - The first inequality is trivial.
  - Exponential blow up, because every set of colors has to be represented by one new color.
- For some classes of graphs, the multi-clique-width is exponentially smaller than the clique-width.

Example: The Independent Set Polynomial

- Definition: $I(x) = \sum a_i x^i$ with a_i = number of independent sets of size i.
- (Maximum Independent Set is easier.)
- Define the k-labeled independent set polynomial:

$$P(x, x_1, \ldots, x_k) = \sum_{i=1}^{n} \sum_{(n_1,\ldots,n_k)\in\{0,1\}^k} a_{i;n_1,\ldots,n_k}\, x^i x_1^{n_1} \ldots x_k^{n_k}$$

  where $a_{i;n1,\ldots,nk}$ is the number of independent sets of size i such that some vertices are labeled j iff $n_j = 1$.
- $P(x,x_1,\ldots,x_k)$ is computed for subgraphs of G induced by subtrees bottom up.
- The polynomial I(x) is obtained from $P(x,x_1,\ldots,x_k)$ by:

$$I(x) = P(x, 1, \ldots, 1) = \sum_{i=1}^{n} \sum_{(n_1,\ldots,n_k)\in\{0,1\}^k} a_{i,n_1,\ldots,n_k} x^i$$

Computation of $P(x,x_1,\ldots,x_k)$

- Compute $P(x,x_1,\ldots,x_k)$ bottom up.
- $m\langle i_1,\ldots,i_j\rangle$: $$1+\sum_{\ell=1}^{m}\binom{m}{\ell}x^{\ell}x_{i_1}\cdots x_{i_j}=1+((1+x)^m-1)x_{i_1}\cdots x_{i_j}.$$
- $\eta_{i,j}$: Delete all monomials containing $x_i x_j$.
- $\rho_{i\to S}$: First replace x_i by $x_{i_1}\cdots x_{i_j}$ for $S=\{i_1,\ldots,i_j\}$. Then replace x_j^2 by x_j for all j.
- $\oplus$: First, multiply the two polynomials.
  Then replace x_j^2 by x_j for all j.
- At the end: Delete all x_i.
- The indepenent set polynomial is in FPT.

Summary

The width paramete, mcw has these two advantages:

- It generalizes tree-width without an exponential explosion.
- For some interesting applications, the running time is the same function of the (sometimes exponentially smaller) multi-clique-width as of the clique-width.

Open Problems

- Complexity of computing or approximating multi-clique-width?
- For which problems are multi-clique-width based algorithms much faster?
- How often is the clique-width much larger than the multi-clique-width?

Thank you!