

# An Experimental Study of Treewidth of Real-World Graph Data

Silviu Maniu, Pierre Senellart, Suraj Jog



ICDT 2019, Lisbon, March 26th 2019

# Why Treewidth?

Many **computational tasks are hard** (when considering **data complexity**) on graph-like structures, but are easy on tree structures

But just how **tree-like** is the data? And **how useful is having this tree-likeness measure?**

# Why Treewidth?

**Treewidth:** a measure on how tree-like data is [Robertson and Seymour, 1984]

For data that has **bounded treewidth**, one can obtain efficient algorithms:

1. **query evaluation of MSO queries is linear time** over **bounded-treewidth** data structures [Courcelle, 1990], also applies to **counting** and **enumeration**
2. **computing probabilities of MSO queries over TID probabilistic databases** which are **bounded-treewidth** is linear time [Amarilli et al., 2015]

# Treewidth: Intuition

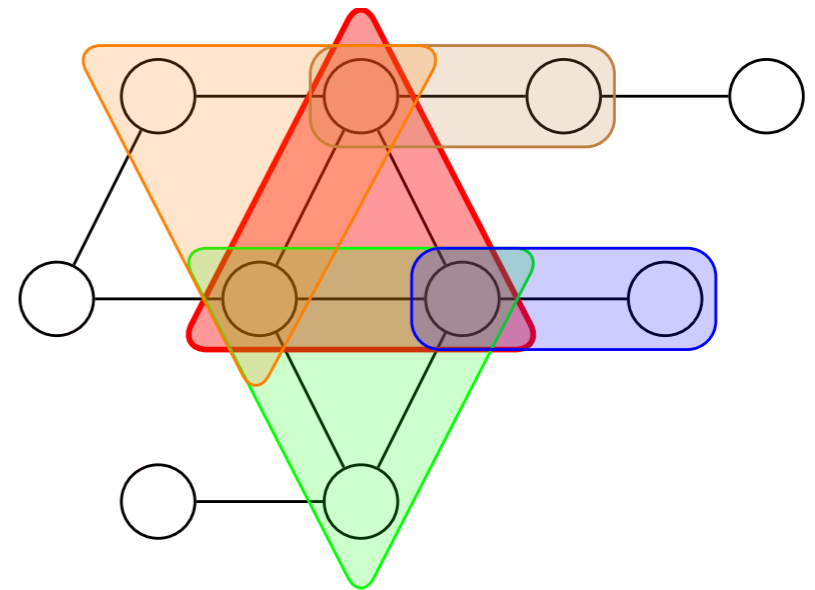
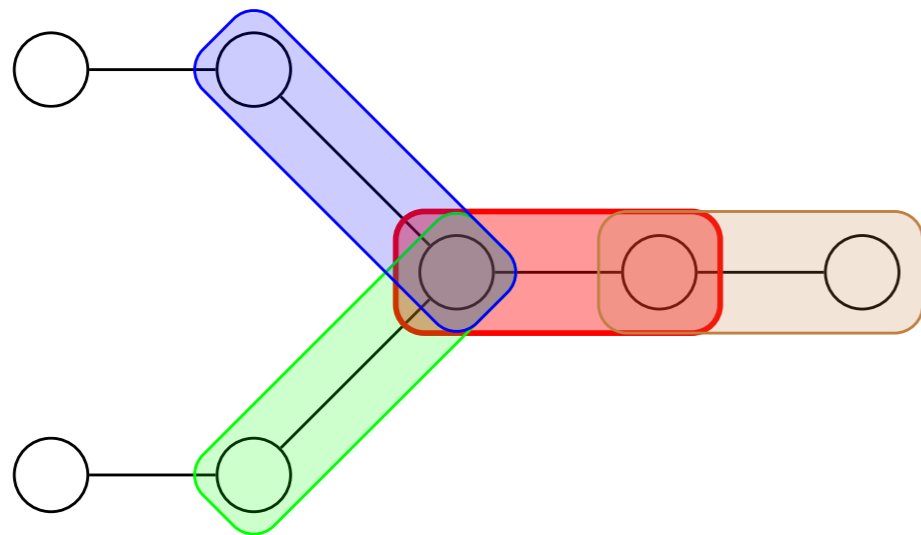
**Treewidth:** a measure on how tree-like data is [Robertson and Seymour, 1984]

Can be obtained using a **tree decomposition** (or a hierarchy of separators)

The treewidth is the **smallest obtainable size of a separator** (minus one)

# Treewidth: Intuition

The treewidth is the **smallest obtainable size of a separator** (minus one)



# Treewidth: Intuition

The **treewidth** is the **smallest obtainable size of a separator** (minus one)

- **trees** have treewidth 1
- **cycles** have treewidth 2
- **$k$ -cliques** have treewidth  $k-1$

# Treewidth: Formal Definition

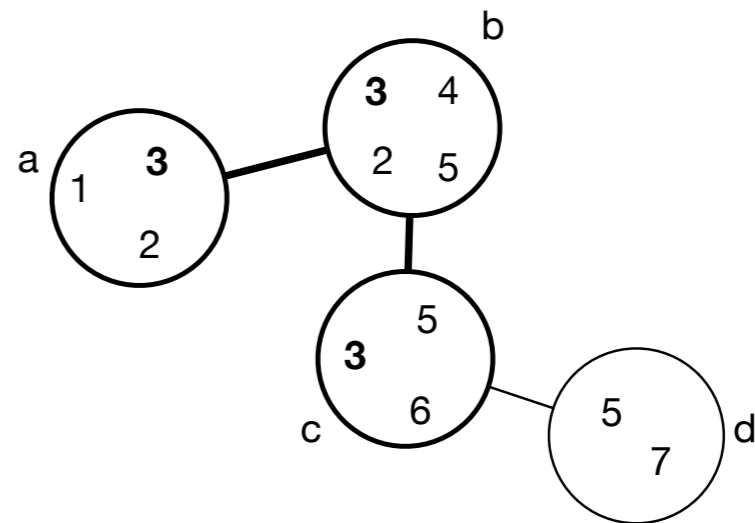
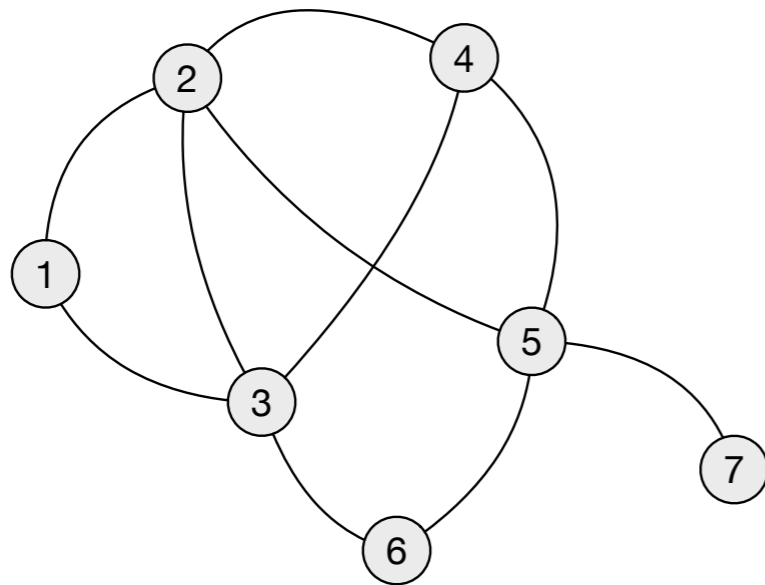
## Definition

Given an *undirected graph*  $G = (V, E)$ , where  $V$  represents the set of vertices (or nodes) and  $E \subseteq V \times V$  the set of edges, a *tree decomposition* is a pair  $(T, B)$  where  $T = (I, F)$  is a tree and  $B : I \rightarrow 2^V$  is a labeling of the nodes of  $T$  by subsets of  $V$  (called *bags*), with the following properties:

1.  $\bigcup_{i \in I} B(i) = V$ ;
2.  $\forall (u, v) \in E, \exists i \in I$  s.t.  $\{u, v\} \subseteq B(i)$ ; and
3.  $\forall v \in V, \{i \in I \mid v \in B(i)\}$  induces a subtree of  $T$ .

# Treewidth: Formal Definition

Given a graph  $G = (V, E)$  the *width* of a tree decomposition  $(T, B)$  is equal to  $\max_{i \in I} (|B(i)| - 1)$ . The *treewidth* of  $G$ ,  $w(G)$ , is equal to the minimal width of all tree decompositions of  $G$ .





# Treewidth

Readily usable for **graphs**, for relational data the Gaifman graph can be used

Only **low treewidth** makes things easy: even linear time algorithms hide a non-elementary dependency in treewidth

In some cases, **treewidth is the only hope to have polynomial time algorithms!** [Amarilli et al., 2016]

# In Practice

If data has **low treewidth**, we have plenty of efficient algorithms (based on tree decompositions)

**Question 1:** Are real-world data low treewidth?

# Computing Treewidth

Computing the treewidth exactly is hard [Arnborg et al., 1987]

We can compute **upper bounds** [Bodlaender and Koster, 2010], which also give a **tree decomposition**

Also can obtain **lower bounds** efficiently [Bodlaender and Koster, 2011]

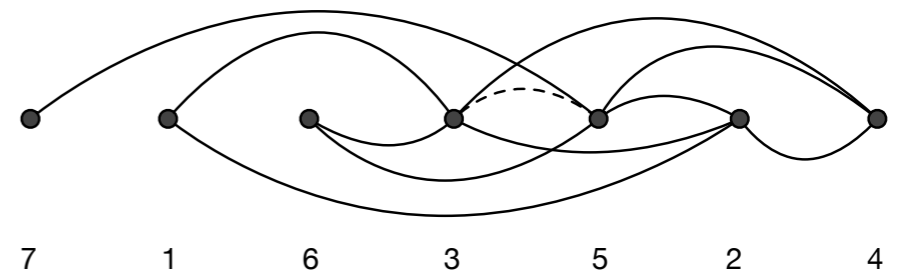
# Upper Bound Algorithms

General algorithm, [generating a tree decomposition](#):

1. generate an **ordering of the nodes**
2. for each node in the ordering, create a **bag** containing it and its neighbors
3. remove the node from the graph, create a **clique between the neighbors**
4. continue until no nodes are left

Details differ in [how the ordering is generated](#):

- **minimum degree first** (MINDEGREE)
- **minimum fill-in** (MINFILLIN)
- combination thereof



# Lower Bound Algorithms

Computing other proxy measures, which are **lower bounds** on treewidth:

1. second lowest degree in the **graph** (DELTA2D)
2. second lowest degree in a **subgraph of the graph** (MMD)
3. second lowest degree in a **minor of the graph** (MMD+)

Algorithms differ in how the subgraphs and minors are explored (usually in a **greedy** fashion)

# Experimental Setup

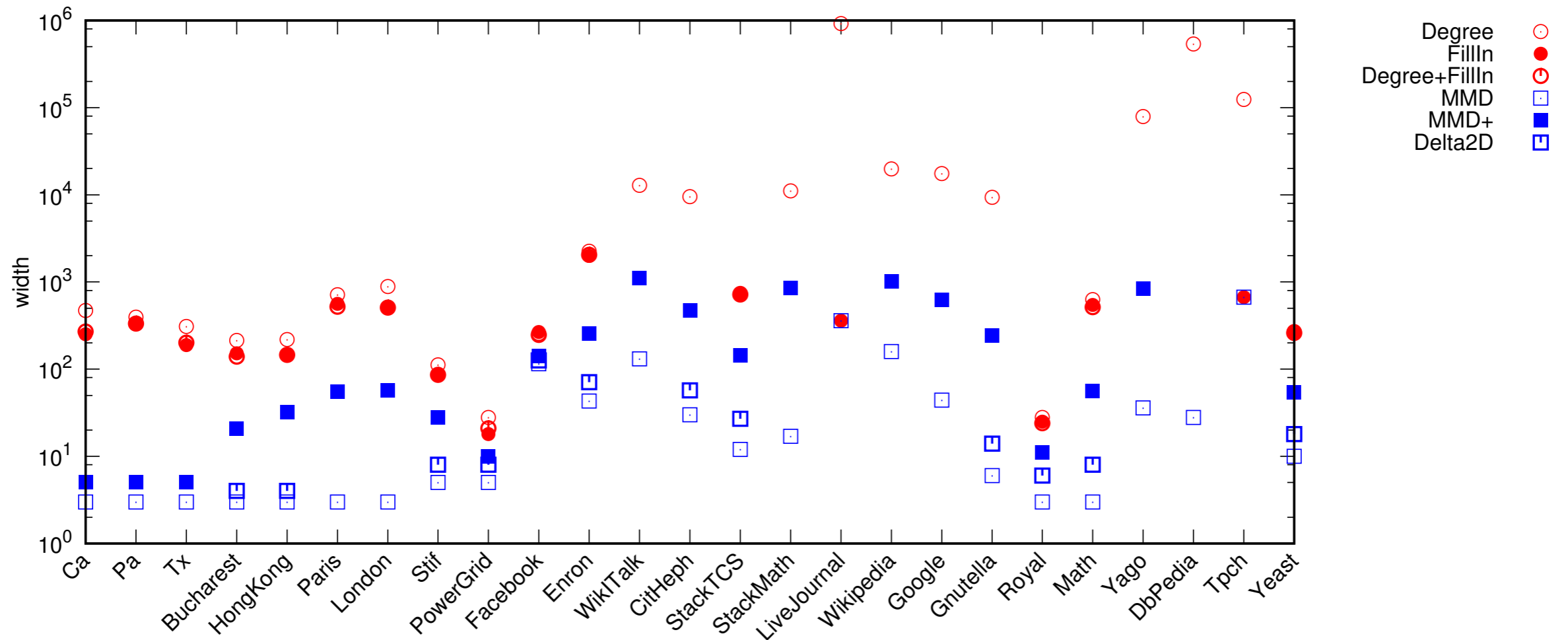
25 **graph datasets** from 8 different **domains**:  
infrastructure, social networks, Web, communication,  
hierarchies, ontologies, relational databases, biology

Tests ran machines on having 32GB RAM, Intel Xeon  
1.7GHz CPUs

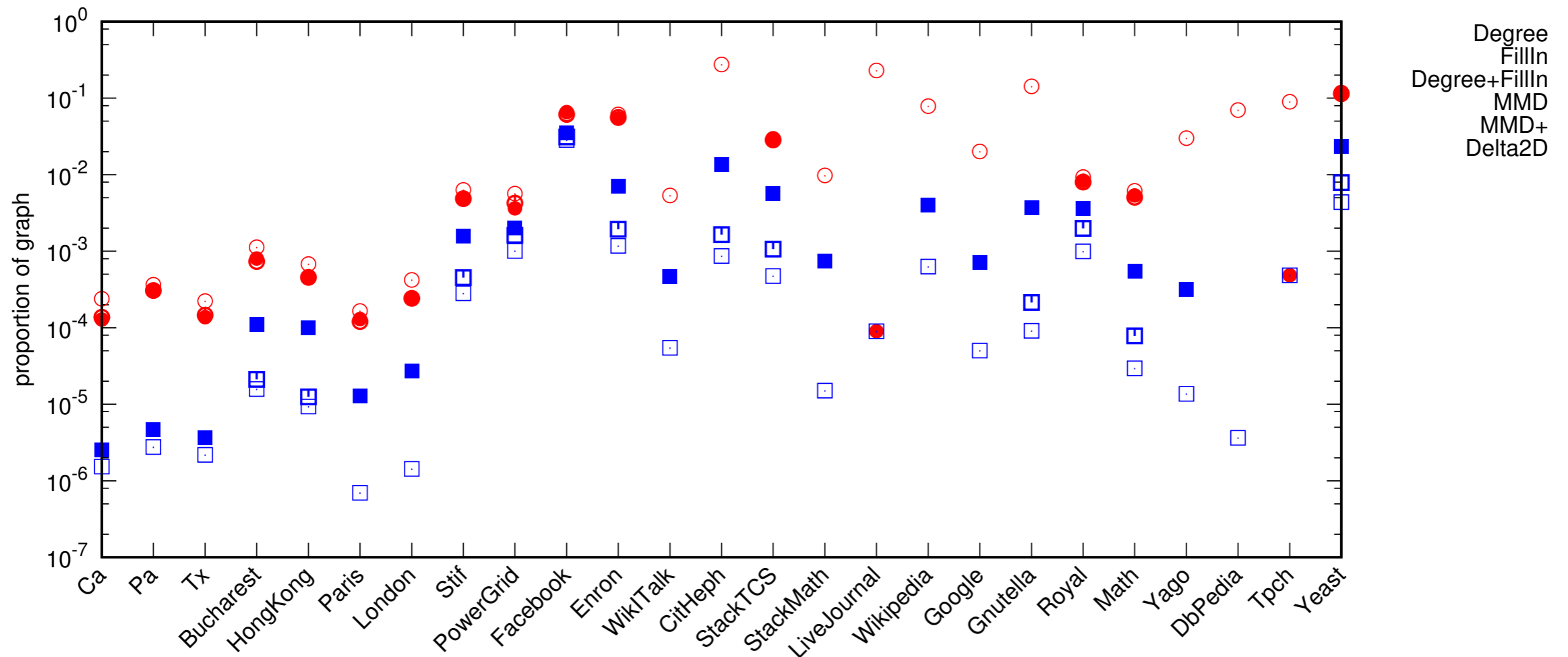
The **maximal running time allowed** was *two weeks*

**Code and datasets available online**  
<https://github.com/smaniu/treewidth>

# Treewidth: Absolute Values



# Treewidth: Relative Values





# Main Takeaways

**Negative result:** treewidth is never very low ( $<10$ )

However, for **infrastructure networks**, treewidth is **relatively low** (similar to the  $\mathcal{O}(\sqrt[3]{n})$  bound conjectured in [Dibbelt et al, 2016])

**Other graphs exhibit high treewidth values**, outside practical usefulness: social networks, Web graphs, knowledge graphs

# Is Treewidth Only of Theoretical Relevance?

**The second question:** can we still find practical use cases for treewidth (and tree decompositions)?

# Partial Tree Decompositions

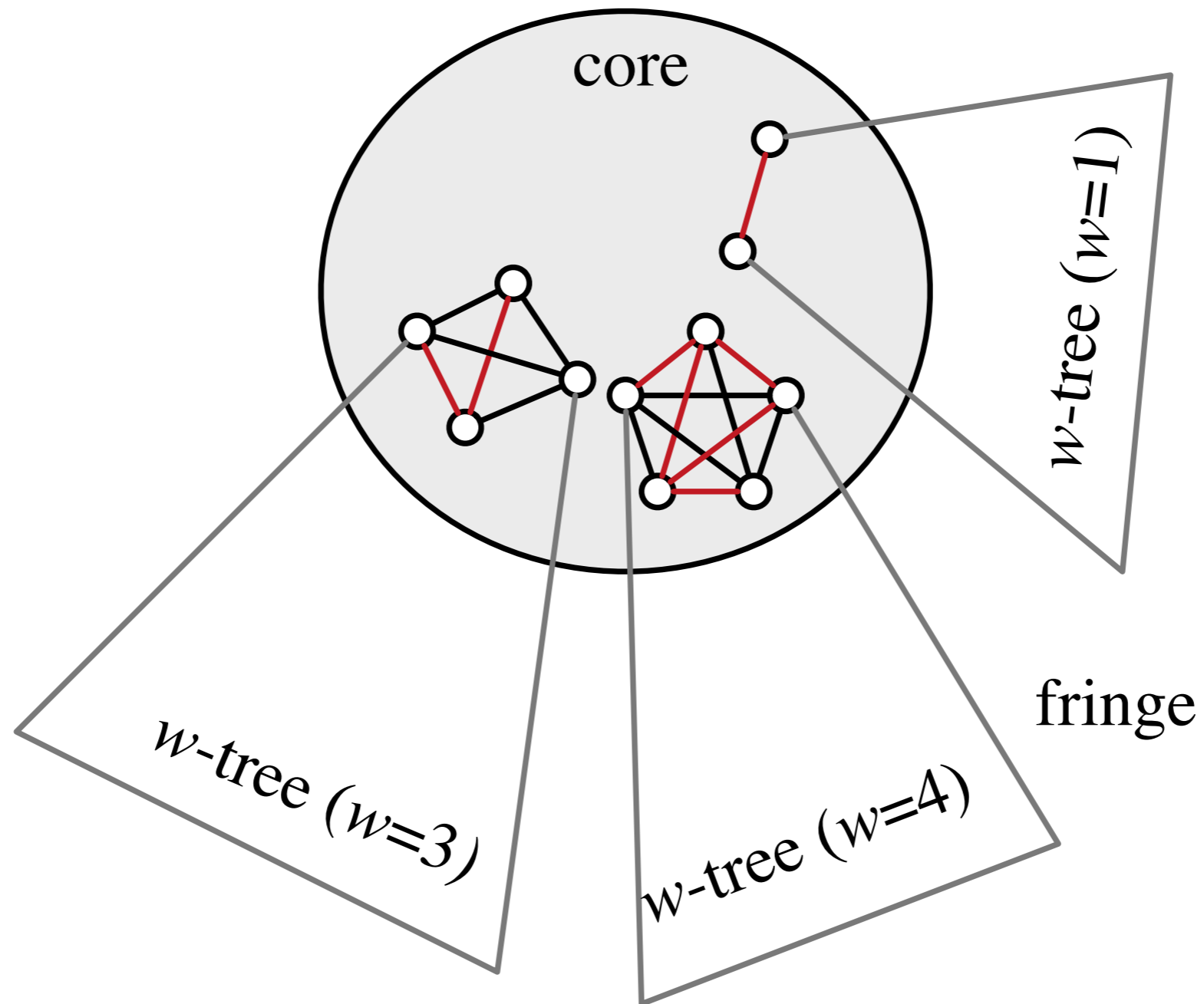
Instead of computing the full tree width estimation, **what if we stop sometime during the decompositions process?**

— **Partial Tree Decomposition**

Results in a hybrid tree-graph structure:

1. a **low treewidth fringe**, and
2. a (potentially) **high-treewidth core graph**

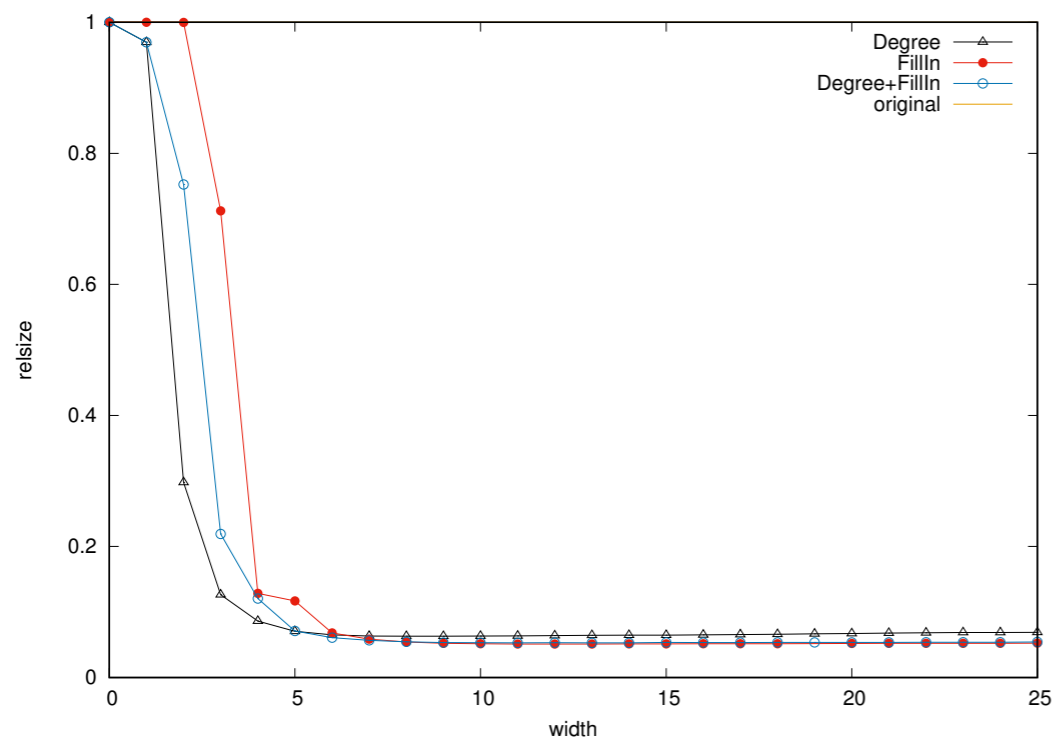
# Partial Tree Decompositions



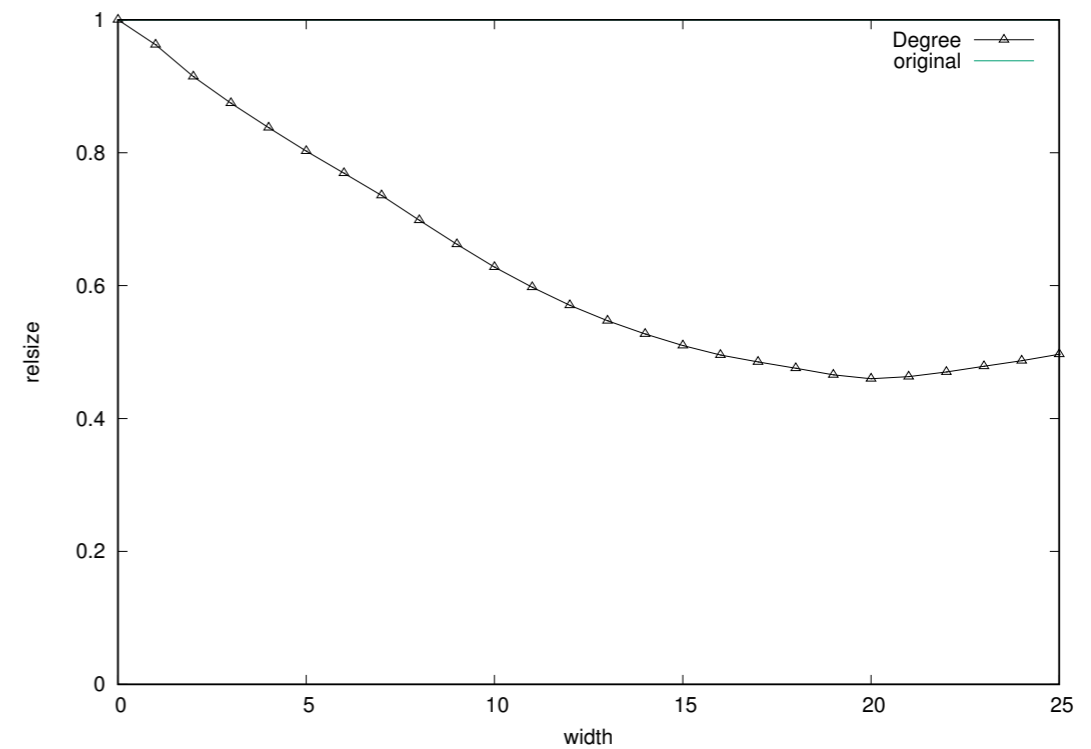
# General Algorithm for Partial Tree Decompositions

1. **Isolate a part of low treewidth** (by stopping the decomposition process early)
2. **(Pre-)process the fringe efficiently** (due to its low treewidth)
3. **Process the core graph using other techniques** (e.g., approximate algorithms)
4. **Combine results**

# Partial Tree Decompositions: When to Stop?



**Paris OpenStreetMaps**  
(4M road segments)



**Google Web graph**  
(875k hyperlinks)

# Applications

**Low treewidth decompositions** were used to compute shortest distances efficiently in graphs [Wei, 2010]

- Might suggest a reason why contraction-based approaches work so well for shortest distances in infrastructure networks

# Applications

In our previous work [Maniu et al, 2017], we used tree decompositions to **compute efficiently reachability probabilities in probabilistic graphs**:

1. **compute exact probabilities in low treewidth areas, and**
2. **combine them in the root graph and use sampling to estimate probabilities**



# Conclusions

- Bounded-treewidth data has nice theoretical properties, but ...
- ... there are no graphs in the real-world having low treewidth.
- However, hope for practical applications remains when using partial tree decompositions!

**Thank you!**