Une étude expérimentale de la largeur d’arbre de données graphe du monde réel

Silviu Maniu  Pierre Senellart  Suraj Jog

22 Octobre 2018
BDA 2018
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- **Width**: maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one

Trees have treewidth 1
Cycles have treewidth 2
$k$-cliques and $(k-1)$-grids have treewidth $k-1$
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one

Trees have treewidth 1
Cycles have treewidth 2
\(k\)-cliques and \((k-1)\)-grids have treewidth \(k\)
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- **Width**: maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- **Width**: maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- **Width**: maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one

Trees have treewidth 1
Cycles have treewidth 2
$k$-cliques and $(k-1)$-grids have treewidth $k$
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- **Width**: maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- **Width:** maximum size of a separator minus one

Trees have treewidth 1
Cycles have treewidth 2
$k$-cliques and $(k-1)$-grids have treewidth $k$
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- **Width**: maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- **Width:** maximum size of a separator minus one
Treewidth: Informal Definition

- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- **Width:** maximum size of a separator minus one

- Trees have treewidth 1
- Cycles have treewidth 2
- $k$-cliques and $(k - 1)$-grids have treewidth $k - 1$
Tree decomposition

Definition (Tree decomposition)

A tree decomposition of a graph \((V, E)\) is a pair \((T, B)\) where \(T = (I, F)\) is a tree and \(B : I \to 2^V\) is a labeling of the nodes of \(T\) by subsets of \(V\) (called bags), with:

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

Definition (Tree decomposition)

A tree decomposition of a graph \((V, E)\) is a pair \((T, B)\) where \(T = (I, F)\) is a tree and \(B : I \to 2^V\) is a labeling of the nodes of \(T\) by subsets of \(V\) (called bags), with:

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 \(T\).
Tree decomposition

Definition (Tree decomposition)

A tree decomposition of a graph \((V, E)\) 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:

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

Definition (Treewidth)

The **width** of a tree decomposition is the maximum size of a bag in it, minus one. The **treewidth** of a graph is the minimum width of a tree decomposition of this graph.
Treewidth: Formal Definition

Definition (Treewidth)

The width of a tree decomposition is the maximum size of a bag in it, minus one. The treewidth of a graph is the minimum width of a tree decomposition of this graph.

In databases:

- Readily usable notion for graph databases (treewidth of the underlying graph)
- Treewidth of a relational database: that of its Gaifman graph (the graph where data values are nodes, and two data values are connected if they co-occur in the same tuple)
Tree Decompositions of Relational Data

Instance:

\[
\begin{array}{ccc}
N & & \\
a & b & \\
b & c & \\
c & d & \\
d & e & \\
e & f & \\
S & & \\
\end{array}
\]
Tree Decompositions of Relational Data

Instance:

N
\[ \begin{array}{cccc}
  a & b \\
  b & c \\
  c & d \\
  d & e \\
  e & f \\
\end{array} \]

S
\[ \begin{array}{cccc}
  a & c \\
  b & e \\
\end{array} \]

Gaifman graph:

\[ \begin{array}{cccc}
  a & f \\
  b & e \\
  c & d \\
\end{array} \]
Tree Decompositions of Relational Data

Instance:

\[
\begin{array}{cccc}
N & a & b \\
 & b & c \\
c & d \\
ds & e \\
e & f \\
\end{array}
\]

Gaifman graph:

Tree decomposition:
Outline

Treewidth

Motivation

Treewidth Computation

Treewidth of Real-World Data

Conclusion
Complex Query Evaluation is Hard!

- query evaluation of Boolean monadic second-order (MSO) queries is hard for every level of the polynomial hierarchy (Ajtai et al., 2000);
- unless $P = NP$, there is no polynomial-time counting or enumeration algorithm for first-order (FO) queries with free second-order variables (Saluja et al., 1995; Durand and Strozecki, 2011);
- computing the probability of conjunctive queries (CQs) over tuple-independent databases is $\#P$-hard (Dalvi and Suciu, 2007);
- unless $P = NP$, there is no polynomial-time algorithm to construct a deterministic decomposable negation normal form (d-DNNF) representation of the Boolean provenance of some CQ (Dalvi and Suciu, 2007; Jha and Suciu, 2013).
Low Treewidth Makes Things Easy!

Assume we know that the databases we work with have treewidth less than some fixed constant $k$. Then:

- **query evaluation** of MSO queries is linear-time (Courcelle, 1990; Flum et al., 2002);
- **counting** (Arnborg et al., 1987) and **enumeration** (Bagan, 2006; Amarilli et al., 2017) of MSO queries is linear-time;
- **computing the probability** of MSO queries over a bounded-treewidth tuple-independent database is linear-time assuming constant-time rational arithmetic (Amarilli et al., 2015);
- a **d-DNNF** representation of the provenance of any MSO query can be computed in linear time (Amarilli et al., 2016).
Low Treewidth Makes Things Easy!

Assume we know that the databases we work with have treewidth less than some fixed constant $k$. Then:

- **query evaluation** of MSO queries is **linear-time** (Courcelle, 1990; Flum et al., 2002);
- **counting** (Arnborg et al., 1987) and **enumeration** (Bagan, 2006; Amarilli et al., 2017) of MSO queries is **linear-time**;
- **computing the probability** of MSO queries over a bounded-treewidth tuple-independent database is **linear-time** assuming constant-time rational arithmetic (Amarilli et al., 2015);
- a **d-DNNF** representation of the provenance of any MSO query can be computed in **linear time** (Amarilli et al., 2016).

(These algorithms are hiding a non-elementary dependency in $k$, so only feasible for very low values of $k$.)
Low Treewidth: Only Hope?

- In some cases, there are other ways to have low complexity: 
  Query evaluation of MSO queries is linear-time over bounded-cliquewidth databases. (Courcelle et al., 2000)

- But in others, there are none!
  There exists an FO-query $Q$ such that for any unbounded-treewidth family of databases $D$, probabilistic query evaluation of $Q$ over $D$ is $\#P$-hard under RP reductions (assuming arity is 2, and some technical condition). (Amarilli et al., 2016)
### Practical Implications?

- If data has low treewidth, plenty of *efficient algorithms*
Practical Implications?

- If data has low treewidth, plenty of efficient algorithms
- Exploiting low treewidth is the only way to have efficient probabilistic query evaluation for arbitrary queries
Practical Implications?

- If data has low treewidth, plenty of efficient algorithms
- Exploiting low treewidth is the only way to have efficient probabilistic query evaluation for arbitrary queries
- Are real-world databases low-treewidth?
Practical Implications?

- If data has low treewidth, plenty of efficient algorithms
- Exploiting low treewidth is the only way to have efficient probabilistic query evaluation for arbitrary queries
- Are real-world databases low-treewidth?
- If not, can we still do something with them?
Even **computing** the treewidth is hard (Arnborg et al., 1987)

But we can find **upper bounds** (Bodlaender and Koster, 2010) and **lower bounds** (Bodlaender and Koster, 2011) on treewidth relatively efficiently

When we have a bound on the treewidth, we can find a tree decomposition in **linear-time** (Bodlaender, 1996)...

but this algorithm is **too costly in practice**. Better use upper bound algorithms that also provide a tree decomposition
Upper Bound Algorithms (Bodlaender and Koster, 2010)

- General strategy:
  - Choose an ordering strategy between nodes (e.g., start with nodes with low degree)
  - Eliminate nodes in this order
  - As nodes are eliminated, put remaining neighbors in a bag and add edges between them so that they form a clique

- The resulting procedure constructs a tree decomposition of the graph

- Algorithms differ by their choice of ordering strategy:
  - minimum degree first
  - minimum fill-in first (# edges to add)
  - combination of both
Lower Bound Algorithms (Bodlaender and Koster, 2011)

- Use a **proxy** that is proved to be always lower than the treewidth:
  - Second lowest degree
  - Second lowest degree in a *subgraph* of the graph
  - Second lowest degree in a *minor* of the graph

- Algorithms differ in the way they **explore** subgraphs or minors (usually **greedily**):
  - by removing nodes of smallest degree
  - by removing nodes of smallest degree except for a fixed node, and trying all such fixed nodes
  - by contracting edges incident to nodes of smallest degree
Outline

Treewidth

Motivation

Treewidth Computation

Treewidth of Real-World Data

Conclusion
Experimental Setup

- 25 datasets from 8 different domains
- All tests ran on a machine with 32GB RAM, Intel Xeon 1.70GHz CPU
- Up to two weeks of computation time before termination
<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>nodes</th>
<th>edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>infrastructure</td>
<td>CA</td>
<td>1,965,206</td>
<td>2,766,607</td>
</tr>
<tr>
<td></td>
<td>PA</td>
<td>1,088,092</td>
<td>1,541,898</td>
</tr>
<tr>
<td></td>
<td>Tx</td>
<td>1,379,917</td>
<td>1,921,660</td>
</tr>
<tr>
<td></td>
<td>Bucharest</td>
<td>189,732</td>
<td>223,143</td>
</tr>
<tr>
<td></td>
<td>HongKong</td>
<td>321,210</td>
<td>409,038</td>
</tr>
<tr>
<td></td>
<td>Paris</td>
<td>4,325,486</td>
<td>5,395,531</td>
</tr>
<tr>
<td></td>
<td>London</td>
<td>2,099,114</td>
<td>2,588,544</td>
</tr>
<tr>
<td></td>
<td>Stif</td>
<td>17,720</td>
<td>31,799</td>
</tr>
<tr>
<td></td>
<td>USPowerGrid</td>
<td>4,941</td>
<td>6,594</td>
</tr>
<tr>
<td>social</td>
<td>Facebook</td>
<td>4,039</td>
<td>88,234</td>
</tr>
<tr>
<td></td>
<td>Enron</td>
<td>36,692</td>
<td>183,831</td>
</tr>
<tr>
<td></td>
<td>WikiTalk</td>
<td>2,394,385</td>
<td>4,659,565</td>
</tr>
<tr>
<td></td>
<td>CitHeph</td>
<td>34,546</td>
<td>420,877</td>
</tr>
</tbody>
</table>
### Datasets (2/2)

<table>
<thead>
<tr>
<th>Category</th>
<th>Dataset</th>
<th>Treewidth</th>
<th>Motivation</th>
<th>Treewidth Computation</th>
<th>Treewidth of Real-World Data</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>social</td>
<td>Stack-TCS</td>
<td>25 232</td>
<td></td>
<td></td>
<td>69 026</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stack-Math</td>
<td>1 132 468</td>
<td></td>
<td></td>
<td>2 853 815</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LiveJournal</td>
<td>3 997 962</td>
<td></td>
<td></td>
<td>34 681 189</td>
<td></td>
</tr>
<tr>
<td>web</td>
<td>Wikipedia</td>
<td>252 335</td>
<td></td>
<td></td>
<td>2 427 434</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Google</td>
<td>875 713</td>
<td></td>
<td></td>
<td>4 322 051</td>
<td></td>
</tr>
<tr>
<td>communication</td>
<td>Gnutella</td>
<td>65 586</td>
<td></td>
<td></td>
<td>147 892</td>
<td></td>
</tr>
<tr>
<td>hierarchy</td>
<td>Royal</td>
<td>3 007</td>
<td></td>
<td></td>
<td>4 862</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Math</td>
<td>101 898</td>
<td></td>
<td></td>
<td>105 131</td>
<td></td>
</tr>
<tr>
<td>ontology</td>
<td>Yago</td>
<td>2 635 315</td>
<td></td>
<td></td>
<td>5 216 293</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DbPedia</td>
<td>7 697 211</td>
<td></td>
<td></td>
<td>30 622 392</td>
<td></td>
</tr>
<tr>
<td>database</td>
<td>Tpch</td>
<td>1 381 291</td>
<td></td>
<td></td>
<td>79 352 127</td>
<td></td>
</tr>
<tr>
<td>biology</td>
<td>Yeast</td>
<td>2 284</td>
<td></td>
<td></td>
<td>6 646</td>
<td></td>
</tr>
</tbody>
</table>
Lower and Upper Bounds (Absolute)

![Diagram showing lower and upper bounds for various datasets]
Lower and Upper Bounds (Relative)
Partial Tree Decompositions

If a database has high-treewidth, possible to:

- Isolate a part of **low treewidth**
- Process this part with **efficient techniques**
- Process the high-treewidth part ( + whatever is needed to keep track of the low-treewidth part) with **other techniques** (e.g., approximation algorithms)
- **Combine** results in a well-founded manner
Partial Tree Decomposition Results

OpenStreetMaps Paris
(5m road segments)

Google Web graph fragment
(4m hyperlinks)
Example Application: Probability of Connectedness

(Maniu et al., 2017)

- Partial tree decomposition with:
  - tendrils of low-treewidth
  - a root node of high-treewidth
Example Application: Probability of Connectedness

(Maniu et al., 2017)

- Partial tree decomposition with:
  - tendrils of low-treewidth
  - a root node of high-treewidth
- Algorithm for probabilistic query evaluation for the connectedness query:
  - Process the tree decomposition bottom-up, keeping track of the provenance of connectedness between exported nodes
  - Add virtual edges with this provenance as annotation
  - When one reaches the core, use Monte-Carlo sampling to approximate the probability
Performance for Connectedness \textit{(Maniu et al., 2017)}
Outline

Treewidth

Motivation

Treewidth Computation

Treewidth of Real-World Data

Conclusion
Summary

• Treewidth is never low (<10) 😞
Summary

- Treewidth is never low (<10) 😞
- Infrastructure network have treewidth lower than other kind of networks: $\mathcal{O}(\sqrt[3]{n})$?
Summary

- Treewidth is never low (<10) 😞
- Infrastructure network have treewidth lower than other kind of networks: $O(3\sqrt{n})$?
- Partial tree decompositions can be very effective
Summary

• Treewidth is never low (<10) 😞
• Infrastructure network have treewidth lower than other kind of networks: $O(\sqrt[3]{n})$?
• Partial tree decompositions can be very effective
• Big gap between upper and lower bounds on treewidth
Summary

- Treewidth is **never low** (<10) 😞
- Infrastructure network have treewidth **lower** than other kind of networks: $O(\sqrt[3]{n})$?
- Partial tree decompositions can be very **effective**
- Big **gap** between upper and lower bounds on treewidth
- Also in this work:
Summary

- Treewidth is never low (<10) 😞
- Infrastructure network have treewidth lower than other kind of networks: $O(\sqrt[3]{n})$?
- Partial tree decompositions can be very effective
- Big gap between upper and lower bounds on treewidth
- Also in this work:
  - More experimental results
Summary

- Treewidth is never low (<10) 😞
- Infrastructure network have treewidth lower than other kind of networks: $O(3^{\sqrt{n}})$?
- Partial tree decompositions can be very effective
- Big gap between upper and lower bounds on treewidth
- Also in this work:
  - More experimental results
  - Comparative running time of different upper and lower bound algorithms
Summary

- Treewidth is never low (<10) 😞
- Infrastructure network have treewidth lower than other kind of networks: $O(\sqrt[3]{n})$?
- Partial tree decompositions can be very effective
- Big gap between upper and lower bounds on treewidth
- Also in this work:
  - More experimental results
  - Comparative running time of different upper and lower bound algorithms
  - Partial tree decompositions of synthetic graph models
Open Questions and Future Work

- Can we formally prove results on complexity of complex query answering based on parameters of partial tree decompositions?
Open Questions and Future Work

• Can we formally prove results on complexity of complex query answering based on parameters of partial tree decompositions?

• Can we extend the connectedness algorithm on partial tree decompositions to more interesting query languages (regular path queries)? To more general notions of provenance?
Open Questions and Future Work

- Can we formally prove results on complexity of complex query answering based on parameters of partial tree decompositions?
- Can we extend the connectedness algorithm on partial tree decompositions to more interesting query languages (regular path queries)? To more general notions of provenance?
- Can we apply all of this to a real-world problem? Routing in public transport networks with a model of uncertainty on schedules?
References

References I


References II


References III


