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

Silviu Maniu *Pierre Senellart* Suraj Jog

22 Octobre 2018

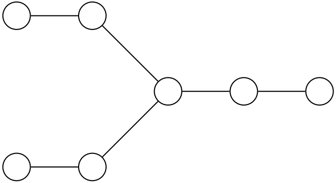
BDA 2018

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

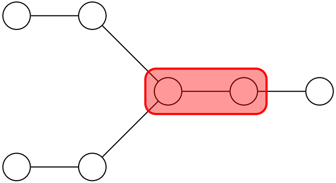
Treewidth: Informal Definition

- 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



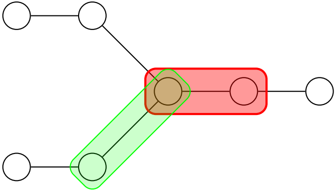
Treewidth: Informal Definition

- 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



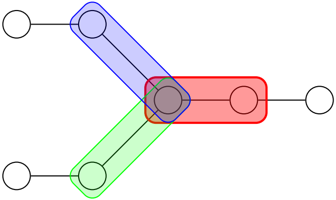
Treewidth: Informal Definition

- 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



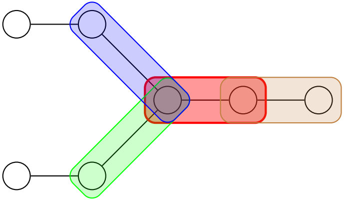
Treewidth: Informal Definition

- 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



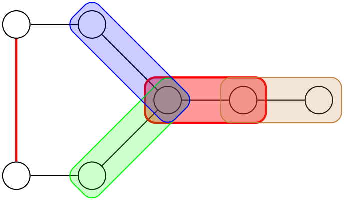
Treewidth: Informal Definition

- 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



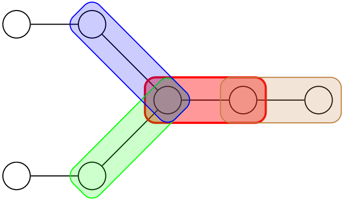
Treewidth: Informal Definition

- 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



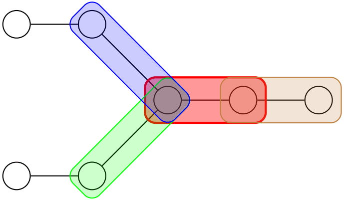
Treewidth: Informal Definition

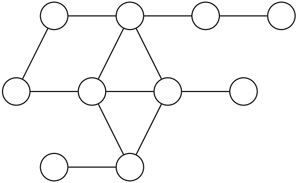
- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one



Treewidth: Informal Definition

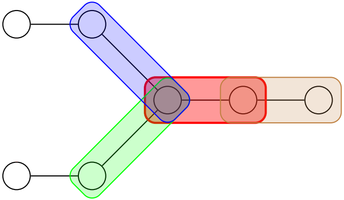
- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one

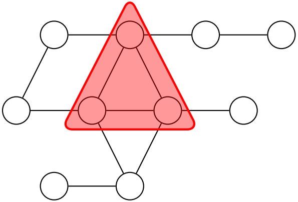




Treewidth: Informal Definition

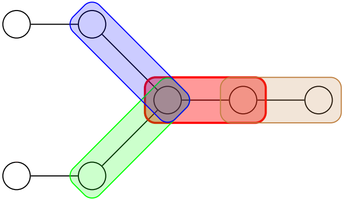
- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one

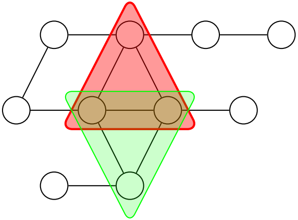




Treewidth: Informal Definition

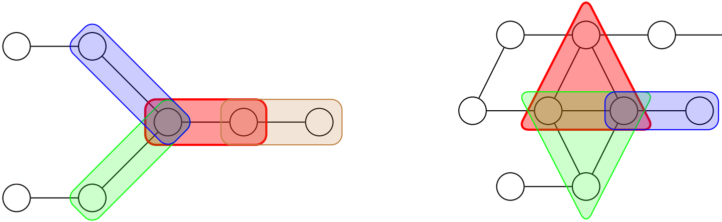
- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one





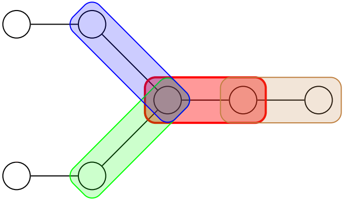
Treewidth: Informal Definition

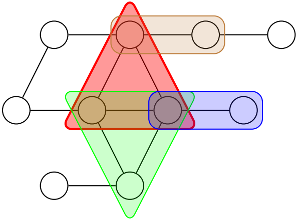
- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one



Treewidth: Informal Definition

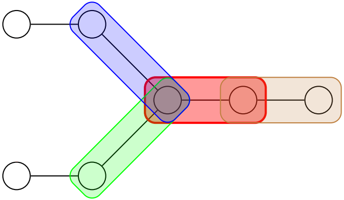
- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one

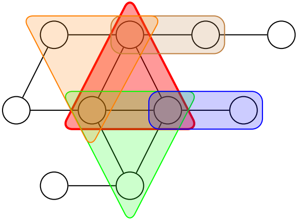




Treewidth: Informal Definition

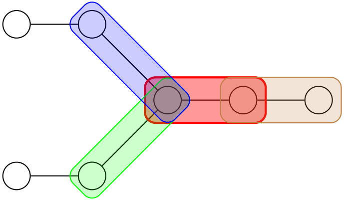
- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one

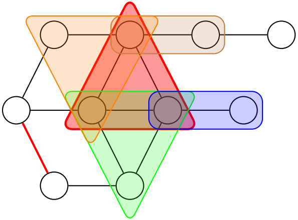




Treewidth: Informal Definition

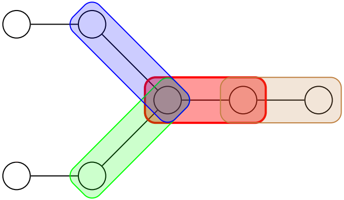
- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one

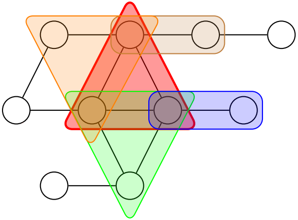




Treewidth: Informal Definition

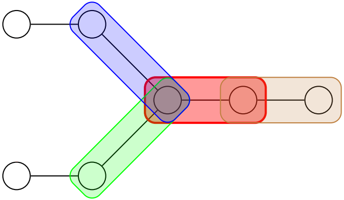
- Graph-theoretic measure of how close to a tree a graph is
- Computed as the minimum width of a tree decomposition, i.e., a way to build a hierarchy of separators
- Width: maximum size of a separator minus one

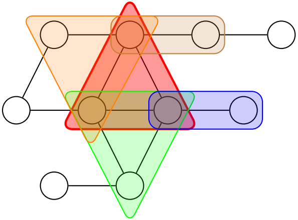




Treewidth: Informal Definition

- 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:

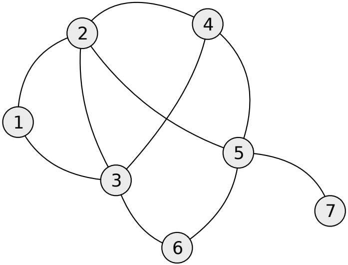
1. $\bigcup_{i \in I} B(i) = V$;
2. $\forall (u, v) \in E$, $\exists i \in I$ s.t. $\{u, v\} \subseteq B(i)$; and
3. $\forall v \in V$, $\{i \in I \mid v \in B(i)\}$ induces a subtree of T.

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.

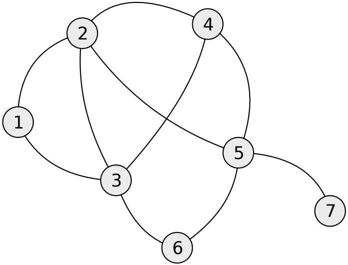


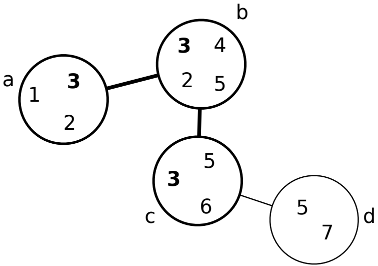
Tree decomposition

Definition (Tree decomposition)

A tree decomposition of a graph (V, E) is a pair (T, B) where $T = (I, F)$ is a tree and $B : I \to 2^V$ 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.





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:

N	
a	b
b	c
c	d
d	e
e	f

S	
a	c
b	e

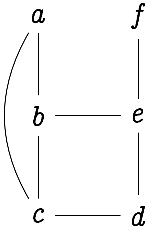
Tree Decompositions of Relational Data

Instance:

N	
a	b
b	c
c	d
d	e
e	f

S	
a	c
b	e

Gaifman graph:



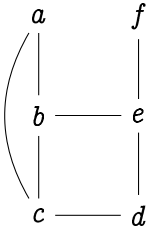
Tree Decompositions of Relational Data

Instance:

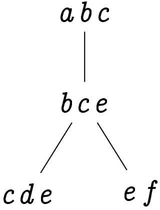
N	
a	b
b	c
c	d
d	e
e	f

S	
a	c
b	e

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 $\mathcal{D}$, probabilistic query evaluation of Q over $\mathcal{D}$ 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?

Outline

Computing the Treewidth

- 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

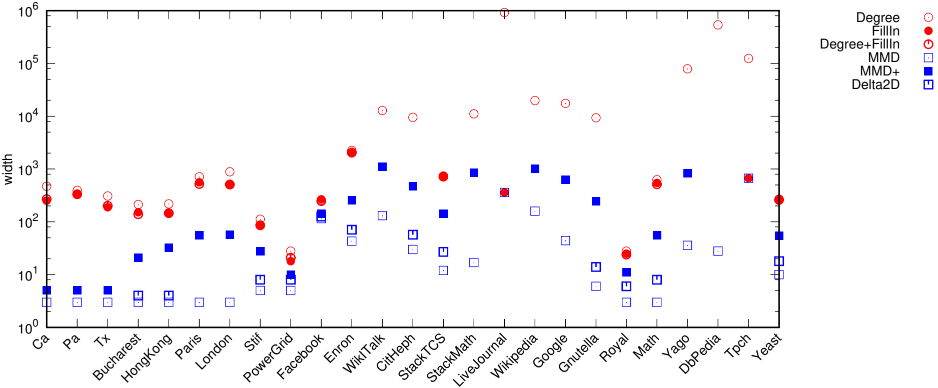
Datasets (1/2)

type	name	nodes	edges
infrastructure	CA	1 965 206	2 766 607
	PA	1 088 092	1 541 898
	TX	1 379 917	1 921 660
	BUCHAREST	189 732	223 143
	HONGKONG	321 210	409 038
	PARIS	4 325 486	5 395 531
	LONDON	2 099 114	2 588 544
	STIF	17 720	31 799
	USPOWERGRID	4 941	6 594
social	FACEBOOK	4 039	88 234
	ENRON	36 692	183 831
	WIKITALK	2 394 385	4 659 565
	CITHEPH	34 546	420 877

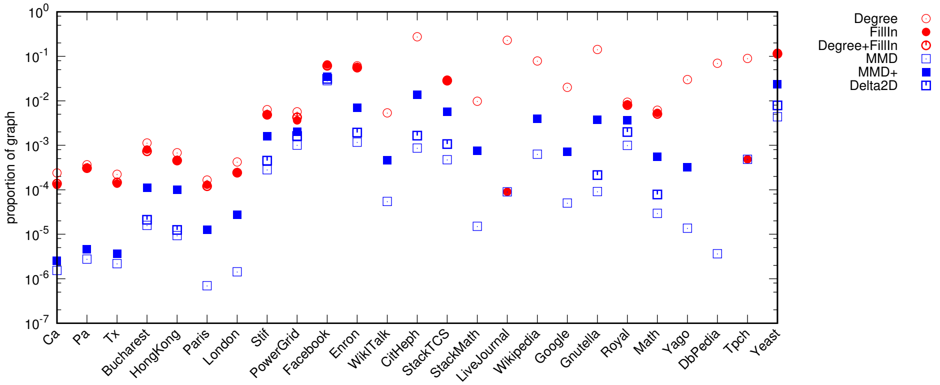
Datasets (2/2)

social	Stack-TCS	25 232	69 026
	Stack-Math	1 132 468	2 853 815
	LiveJournal	3 997 962	34 681 189
web	Wikipedia	252 335	2 427 434
	Google	875 713	4 322 051
communication	Gnutella	65 586	147 892
hierarchy	Royal	3 007	4 862
	Math	101 898	105 131
ontology	Yago	2 635 315	5 216 293
	DbPedia	7 697 211	30 622 392
database	Tpch	1 381 291	79 352 127
biology	Yeast	2 284	6 646

Lower and Upper Bounds (Absolute)



Lower and Upper Bounds (Relative)

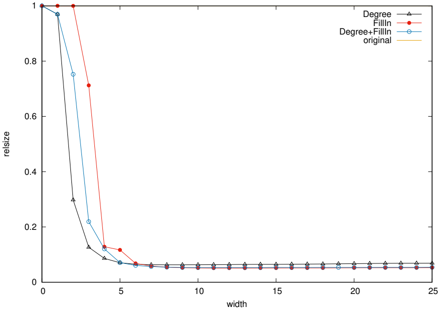


Partial Tree Decompositions

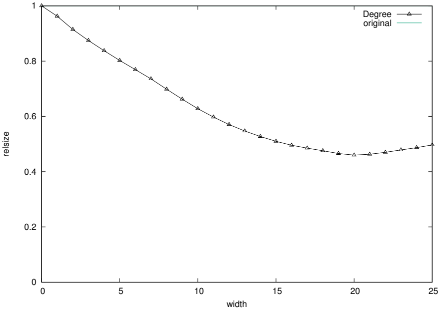
If a database has high-treewidth, possible to:

- 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)

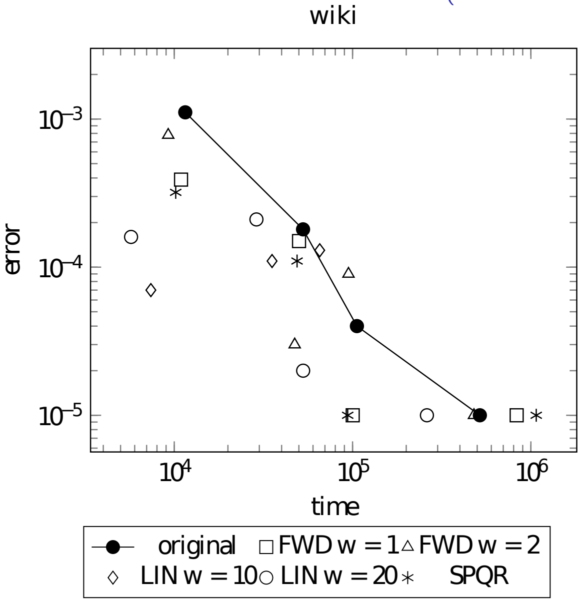
- Partial tree decomposition with:
  - tendrils of low-treewidth
  - a root node of high-treewidth

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 (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: $O(\sqrt[3]{n})$?

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

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: $O(\sqrt[3]{n})$?
- 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

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 I

Ajtai, M., Fagin, R., and Stockmeyer, L. J. (2000). The closure of monadic NP. *JCSS*, 60(3).

Amarilli, A., Bourhis, P., Jachiet, L., and Mengel, S. (2017). A circuit-based approach to efficient enumeration. In *ICALP*.

Amarilli, A., Bourhis, P., and Senellart, P. (2015). Provenance circuits for trees and treelike instances. In *ICALP*.

Amarilli, A., Bourhis, P., and Senellart, P. (2016). Tractable lineages on treelike instances: Limits and extensions. In *PODS*.

Arnborg, S., Corneil, D. G., and Proskuworski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2).

References II

Bagan, G. (2006). MSO queries on tree decomposable structures are computable with linear delay. In *CSL*, volume 4207.

Bodlaender, H. L. (1996). A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6).

Bodlaender, H. L. and Koster, A. M. C. A. (2010). Treewidth computations I. Upper bounds. *Information and Computation*, 208(3).

Bodlaender, H. L. and Koster, A. M. C. A. (2011). Treewidth computations II. Lower bounds. *Information and Computation*, 209(7).

Courcelle, B. (1990). The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1).

References III

Courcelle, B., Makowsky, J. A., and Rotics, U. (2000). Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2).

Dalvi, N. N. and Suciu, D. (2007). The dichotomy of conjunctive queries on probabilistic structures. In *PODS*.

Durand, A. and Strozecki, Y. (2011). Enumeration complexity of logical query problems with second-order variables. In *CSL*.

Flum, J., Frick, M., and Grohe, M. (2002). Query evaluation via tree-decompositions. *J. ACM*, 49(6).

Jha, A. K. and Suciu, D. (2013). Knowledge compilation meets database theory: Compiling queries to decision diagrams. *Theory Comput. Syst.*, 52(3).

References IV

Maniu, S., Cheng, R., and Senellart, P. (2017). An indexing framework for queries on probabilistic graphs. *ACM Trans. Database Syst.*, 42(2).

Saluja, S., Subrahmanyam, K., and Thakur, M. (1995). Descriptive complexity of #P functions. *JCSS*, 50(3).