DATABASE THEORY

Lecture 7: Tree-Like Conjunctive Queries

Markus Krötzsch
Knowledge-Based Systems

TU Dresden, 14th May 2018
Review

Conjunctive queries (CQs) are simpler than FO-queries:
- NP combined and query complexity (instead of PSpace)
- data complexity remains in AC$^0$

CQs become even simpler if they are tree-shaped:
- GYO algorithm defines acyclic hypergraphs
- acyclic hypergraphs have join trees
- join trees can be evaluated in P with Yannakakis’ Algorithm

This time:
- Find more general conditions that make CQs tractable
  \( \sim \) “tree-like” queries that that are not really trees
- Play some games
Is Yannakakis’ Algorithm Optimal?

We saw that tree queries can be evaluated in polynomial time, but we know that there are much simpler complexity classes:

\[ \text{NC}^0 \subset \text{AC}^0 \subset \text{NC}^1 \subset \text{L} \subset \text{NL} \subset \text{AC}^1 \subset \ldots \subset \text{NC} \subset \text{P} \]
Is Yannakakis’ Algorithm Optimal?

We saw that tree queries can be evaluated in polynomial time, but we know that there are much simpler complexity classes:

$$\text{NC}^0 \subset \text{AC}^0 \subset \text{NC}^1 \subset L \subset \text{NL} \subset \text{AC}^1 \subset \ldots \subset \text{NC} \subset \text{P}$$

Indeed, tighter bounds have been shown:

**Theorem 7.1 (Gottlob, Leone, Scarcello: J. ACM 2001):** Answering tree BCQs is complete for LOGCFL.

LOGCFL: the class of problems LogSpace-reducible to the word problem of a context-free language:

$$\text{NC}^0 \subset \text{AC}^0 \subset \text{NC}^1 \subset L \subset \text{NL} \subset \text{LOGCFL} \subset \text{AC}^1 \subset \ldots \subset \text{NC} \subset \text{P}$$

$\sim$ highly parallelisable
In practice, many queries are tree queries, but even more queries are “almost” tree queries, but not quite . . .

How can we formalise this idea?

Several attempts to define “tree-like” queries:

- Treewidth: a way to measure tree-likeness of graphs
- Query width: towards tree-like query graphs
- Hypertree width: adoption of treewidth to hypergraphs
How to recognise trees . . .

. . . from quite a long way away:
How to recognise trees . . .

. . . from quite a long way away:
Tree Decompositions

Idea: if we can group the edges of a graph into bigger pieces, these pieces might form a tree structure

**Definition 7.2:** Consider a graph $G = \langle V, E \rangle$. A tree decomposition of $G$ is a tree structure $T$ where each node of $T$ is a subset of $V$, such that:

- The union of all nodes of $T$ is $V$.
- For each edge $(v_1 \rightarrow v_2) \in E$, there is a node $N$ in $T$ such that $v_1, v_2 \in N$.
- For every vertex $v \in V$, the set of nodes of $T$ that contain $v$ form a subtree of $T$; equivalently: if two nodes contain $v$, then all nodes on the path between them also contain $v$ (*connectedness condition*).

Nodes of a tree decomposition are often called **bags**
(not related to the common use of “bag” as a synonym for “multiset”)
Tree Decompositions: Example
Tree Decompositions: Example
The treewidth of a graph defines how “tree-like” it is:

**Definition 7.3:** The width of a tree decomposition is the size of its largest bag minus one.

The treewidth of a graph $G$, denoted $\text{tw}(G)$, is the smallest width of any of its tree decompositions.

Simple observations:

- If $G$ is a tree, then we can decompose it into bags that contain only one edge
  $\Rightarrow$ trees have treewidth 1

- Every graph has at least one tree decomposition where all vertices are in one bag
  $\Rightarrow$ maximal treewidth = number of vertices − 1
Tree decomposition of width 3
Treewidth: Example

\[ \sim \text{ tree decomposition of width 2} = \text{treewidth of the example graph} \]
What is the treewidth of the following graphs?
Treewidth and Conjunctive Queries

Treewidth is based on graphs, not hypergraphs
Treewidth and Conjunctive Queries

Treewidth is based on graphs, not hypergraphs
\[ \leadsto \text{treewidth of } CQ = \text{treewidth of primal graph of query hypergraph} \]

Query graph and corresponding primal graph:

\[ \leadsto \text{Treewidth 3} \]

Observation: acyclic hypergraphs can have unbounded treewidth!
Queries of low treewidth can be answered efficiently:

**Theorem 7.4 (Dechter/Chekuri+Rajamaran ’97/Kolaitis+Vardi ’98/Gottlob & al. ’98):**
Answering BCQs of treewidth $k$ is possible in time $O(n^k \log n)$, and thus in polynomial time if $k$ is fixed.
The problem is also complete for LOGCFL.

Checking for low treewidths can also be done efficiently:

**Theorem 7.5 (Bodlaender ’96):** Given a graph $G$ and a fixed number $k$, one can check in linear time if $\text{tw}(G) \leq k$, and the corresponding tree decomposition can also be found in linear time.

**Warning:** neither CQ answering nor tree decomposition might be practically feasible if $k$ is big.
Treewidth via Games

Seymour and Thomas [1993] gave an alternative characterisation of treewidth:

The Cops-and-Robber Game

- The game is played on a graph $G$  
- There are $k$ cops and one robber, each located at one vertex  
- In each turn:
  - the cops can fly to an arbitrary vertex in the graph
  - the robber can run along the edges of the graph, as far as she likes, as long as she does not pass through any vertex that was occupied by a cop before or after the turn
    (the robber can run to a place where a cop was before the turn, but not pass through such a place)
- The goal of the cops is to catch the robber; the goal of the robber is never to be caught
Cops and Robbers: Example
Cops and Robbers: Example
Cops and Robbers: Example
Cops and Robbers: Example
Cops and Robbers: Example
Cops and Robbers: Example
Cops and Robbers: Example
Cops and Robbers: Example

Caught!
Theorem 7.6 (Seymour and Thomas): A graph $G$ is of treewidth $\leq k - 1$ if and only if $k$ cops have a winning strategy in the cops & robber game on $G$.

Intuition: the cops together can block even the widest branch and still move in on the robber
Kolaitis and Vardi [1998] gave a logical characterisation of treewidth

Bounded treewidth CQs correspond to certain FO-queries:

- We allow FO-queries with $\exists$ and $\land$ as only operators
- But operators can be nested in arbitrary ways (unlike in CQs)
- Theorem: A query can be expressed with a CQ of treewidth $k$ if and only if it can be expressed in this logic using a query with at most $k + 1$ distinct variables

**Intuition:** variables can be reused by binding them in more than one $\exists$

$\Rightarrow$ Apply a kind of “inverted prenex-normal-form transformation”

$\Rightarrow$ Variables that occur in the same atom or in a “tightly connected” atom must use different names

$\Rightarrow$ minimum number of variables $\Leftrightarrow$ treewidth (+1)
Summary and Outlook

Treewidth has Pros and Cons:

**Advantages:**
- Bounded treewidth is easy to check
- Bounded treewidth CQs are easy to answer

**Disadvantages:**
- Even families of acyclic graphs may have unbounded treewidth
- Loss of information when using primal graph
  (cliques might be single hyperedges – linear! –
  or complex query patterns – exponential!)

**Open questions:**
- Are there better ways to capture “tree-like” queries?