Principles of Managing Uncertain Data
Lecture 4: Computing Joins

Introduction

We have learned the concepts of data complexity and combined complexity.

We have seen that CQs can be evaluated in polynomial time under data complexity:
- And that the degree of the polynomial “necessarily” depends on the query (W[1]-hardness)

We have seen that, under combined complexity:
- Boolean CQ evaluation is NP-complete
- CQs cannot be evaluated in polynomial total time, unless P = NP

This Lecture

- In this lecture, we focus on combined complexity
- We have seen an example of a fragment of CQs that can be evaluated in polynomial total time
  - Namely, n-length paths
- We will learn more a general fragment of tractable CQs
  - Acyclic CQs
  - More generally, CQs of a bounded hypertree width
- In addition, we will learn size bounds on (projection-free) joins, and a matching (worst-case optimal) algorithm

Table of Contents

1. Introduction
2. Preliminaries
3. Acyclic Joins
4. Algorithm for Acyclic Joins (Yannakakis)
5. Joins with Hypertree Decompositions
6. Size Bounds and Worst-Case Optimality

Plan

Previous Lecture

- We have learned the concepts of data complexity and combined complexity
- We have seen that CQs can be evaluated in polynomial time under data complexity
- And that the degree of the polynomial “necessarily” depends on the query (W[1]-hardness)
- We have seen that, under combined complexity:
  - Boolean CQ evaluation is NP-complete
  - CQs cannot be evaluated in polynomial total time, unless P = NP
Recalling Conjunctive Queries

- Recall that a Conjunctive Query (CQ) has the form

\[ Q(x) := \varphi_1(x, y), \ldots, \varphi_m(x, y) \]

where each \( \varphi_i \) is an atomic formula, \( x \) and \( y \) are disjoint sequences of unique variables.

- An atomic formula has the form \( R(t_1, \ldots, t_n) \) where \( R \) is a \( k \)-ary relation symbol and each \( t_i \) is either a variable (in \( x \) or \( y \)) or a constant term.

- \( Q(x) \) is the head, \( \varphi_1(x, y), \ldots, \varphi_m(x, y) \) is the body, and each \( \varphi_i(x, y) \) is a body atom.

- We require every variable in the head to occur at least once in the body.

Result of a CQ

- Let \( Q(x) := \varphi_1(x, y), \ldots, \varphi_m(x, y) \) and \( I \) be a CQ and an instance, respectively (over the same signature).

- A homomorphism from \( Q \) to \( I \) is a function \( \mu \) that maps every variable of \( Q \) to a constant, such that \( \mu(\varphi_i(x, y)) \) is a fact of \( I \) for every \( i = 1, \ldots, m \).

- \( \mu(\varphi_i(x, y)) \) is the fact that is obtained by replacing every variable \( z \) with the constant \( \mu(z) \).

- If \( \mu \) is a homomorphism from \( Q \) to \( I \), then \( \mu \) is the restriction of \( \mu \) to the variables of \( x \).

- The result of evaluating \( Q \) over \( I \), denoted \( Q(I) \), is the set \( \{ \mu \mid \mu \text{ is a homomorphism from } Q \text{ to } I \} \).

Numerical Reductions

Reduction 1: from Clique

- To understand the difficulty of joins, we will recall the proof of
-\[ \text{NP-hardness, and see a new one} \]

- In the first reduction (that we have seen already), we generated a CQ with a single binary relation, repeating many times.

- In the second reduction, we generate a CQ with many ternary relation symbols, but none of them appears more than once in \( Q \); in addition, each relation has precisely seven tuples.

- A CQ without repeated relation symbols is called non-repeating or self-join free.

Reduction 2: from 3-SAT

- Given a propositional formula \( \psi = \varphi_1 \land \cdots \land \varphi_m \) over the variables \( x_1, \ldots, x_n \), where each \( \varphi_i \) is a disjunction of three atomic formulas (each has the form \( x_i \) or \( \neg x_i \)), determine whether \( \psi \) is satisfiable.
Reduction

- Given \( \psi = \varphi_1 \land \cdots \land \varphi_m \) we construct:
  - A relation symbol \( R_i \) for each \( \varphi_i \)
  - An atomic formula \( \phi_i = R_i(x, y, z) \) where \( x, y \) and \( z \) are the variables that appear in \( \varphi_i \)
  - \( Q(x_1, \ldots, x_n) = \phi_1 \land \cdots \land \phi_m \)
    - The instance \( I \) has in the relation \( R_i \) all \( 7 \) tuples \((b_1, b_2, b_3) \in \{(0,1),(1,0)\}\) that satisfy \( \phi_i \)
  - That’s it!

Example

- \( \psi: (x \lor y \lor z) \land (\neg x \lor y \lor w) \land (x \lor \neg z \lor \neg w) \)
- \( Q(x, y, z, w): R_1(x, y, z), R_2(x, y, z), R_3(x, z, w) \)

\[
\begin{array}{ccc}
I = & R_1 & R_2 & R_3 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

Reduction 3: from 3-Coloring

**Problem Def. (3-Coloring)**

Given a (directed) graph \( G = (V, E) \), determine whether we can assign a color from \{\( r, g, b \}\) to each node, so that no two neighbors get the same color.

From CQs to Joins

- It is sometimes more comfortable to work with RA joins (and projection) instead of CQs.
- Given a CQ \( Q \) and an instance \( I \) over a schema \( S \), we can easily construct a schema \( T \), an RA expression \( \alpha \) over \( T \) and an instance \( J \) over \( T \) such that:
  - \( \alpha \) has the form \( \pi_{y_1, \ldots, y_k}(T_1 \bowtie \cdots \bowtie T_m) \) where the \( T_i \) are distinct relation symbols
  - \( \alpha(J) \) and \( Q(I) \) are “the same”
  - That is, there is a straightforward translation between the two
- For example, how would you translate the following CQ?
  - \( Q(x, y) = R(x, y, \text{Avia}), R(y, z, x), S(x, x) \)

Translation

- Let \( Q(x) = \varphi_1(x, y), \ldots, \varphi_m(x, y) \) and \( I \) be over \( S \)
- Each variable becomes an attribute
- Each body atom \( \varphi_i \) becomes a unique relation schema \( T_i \) with the attributes (variables) that appear in \( \varphi_i \) (in any order)
- Each head variable becomes a projection attribute
- In \( J \), the relation \( T_i \) is obtained by evaluating \( \varphi_i \) over \( I \) as if \( \varphi_i \) is a CQ with all variables in the head
- Example: \( Q(x, y) = R(x, y, \text{Avia}), R(y, z, x), S(x, x) \)
  \[\Rightarrow \pi_{x,y}(T_1(x, y, \text{Avia}), T_2(x, y, z) \bowtie T_3(x))\]
In the remainder of this lecture, a CQ expression is an RA expression of the form
\[ \pi_A(R_1 \bowtie \cdots \bowtie R_k) \]

- Every \( R_i \) is a distinct relation symbol (of any arity)
- \( A \) is a sequence of attributes among the \( A_i \)s
- If projection \( \pi \) is redundant, it may be omitted

A hypergraph is a pair \((V, H)\), where \( V \) is a finite set of nodes, and \( H \) is a set of subsets of \( V \), called hyperedges (and sometimes just edges)

If \( H \) is a hypergraph, then we denote by
- \( \text{nodes}(H) \) the set of nodes of \( H \)
- \( \text{edges}(H) \) the set of hyperedges of \( H \)

Let \( \alpha = \pi_A(R_1 \bowtie \cdots \bowtie R_k) \) be a CQ expression
- The hypergraph of \( \alpha \), denoted \( H_\alpha \), has:
  - The attributes in \( \alpha \) as the set of nodes
  - A hyperedge \( e_i \) for each \( R_i \), containing the attributes of \( R_i \)

A join tree of a hypergraph \( H \) is a tree \( T \) with the following properties
- The nodes of \( T \) are the hyperedges of \( H \)
  - In notation, \( \text{nodes}(T) = \text{edges}(H) \)
  - For every \( e \in \text{nodes}(H) \), the nodes of \( T \) that contain \( e \) form a connected subtree of \( T \)
- Example:

An ear of a hypergraph \( H \) is a hyperedge \( e \) of \( H \) such that
- \( e \) is disjoint from all other hyperedges or
- there exists another hyperedge \( e' \) where \( e \setminus e' \) is disjoint from all other hyperedges

An ear removal on \( H \) is the operation of obtaining a new hypergraph \( H' \) by removing an ear \( e \) of \( H \)
- \( \text{nodes}(H') = \text{nodes}(H) \) and \( \text{edges}(H') = \text{edges}(H) \setminus \{e\} \)

Example:
Proposition

Let $H$ be a hypergraph. The following are equivalent:

1. $H$ has a join tree.
2. By repeatedly applying ear removal (in any order), one can eliminate all the hyperedges of $H$.

If $H$ satisfies the above conditions, then $H$ is said to be acyclic.

You will prove the proposition in a home assignment.

In particular, you will show how to build a join tree for a given $H$ via ear removal efficiently!

(This will be used later in this lecture)

When $H$ is a graph (i.e., every hyperedge has exactly two nodes), acyclicity is the usual notion of graph acyclicity (forest).

In other words, graph acyclicity and hypergraph acyclicity are the same on graphs.

A CQ expression $\alpha$ is acyclic if its associated hypergraph $H_\alpha$ is acyclic.

Which of the following is acyclic?

\[
\begin{align*}
\left( \bigwedge_{i<j} R_{ij}(x_i, x_j) \right) \\
\left( \bigvee_{i<j} R_{ij}(x_i, x_j) \right) \bowtie S(x_1, \ldots, x_n)
\end{align*}
\]

Which of the above can be solved in polynomial total time?

In this part we describe the algorithm of Mihalis Yannakakis [Yan81] for computing acyclic CQs.

The algorithm terminates in polynomial total time.

Recall: polynomial time in the combined size of the input and the output.

Input: CQ expression $\alpha = \pi_A \left( R_1 \bowtie \cdots \bowtie R_k \right)$, instance $I$

1. Compute a join tree $T$ for $H_\alpha$.
2. Apply a full reduction to $I$ according to $T$.
   
   More specifically, replace source relations with semijoins.
3. Compute $\alpha(I)$ in leaf-to-root order according to $T$, projecting on only relevant variables.

And eliminating every redundant/irrelevant variable.
Computing a Join Tree

- This can be done (in polynomial time) by the ear-removal procedure.
- We will view the join tree as directed and ordered by:
  - Selecting an arbitrary root that all nodes are reachable from.
  - This action determines all directions.
  - Selecting an arbitrary order among every set of siblings.
- In the next slides, denote this (directed & ordered) tree by $T$.

Intuition on Full Reduction (1)

- The left semijoin of two relations $r$ and $s$, denoted $r \leftarrow s$, is the relation that is obtained from $r$ by selecting only the tuples that have a matching tuple in $s$ (i.e., are joinable with) $s$.
- In RA:
\[ r \leftarrow s \overset{\text{def}}{=} \pi_A(r \bowtie s) \]
where $A$ is the attribute sequence of $r$.
- For example, what is $r \leftarrow s$ if:
  - $r$ and $s$ have the same set of attributes?
  - $r$ and $s$ have disjoint sets of attributes?

Referenced Notation

- For each node $v$ of $T$, let:
  - $R_v$ be the relation symbol that corresponds to $v$.
  - $r_v$ be the relation of $I$ over $R_v$.
- Example: $\pi_A(B(x, y, z) \bowtie S(x, u) \bowtie T(y, z, w))$.

Left Semijoin
On the other hand, after the full reduction, there are no tuples that contribute to the overall result of the join; why so?

For each node $v$, let $T_v$ be the subtree of $T$ rooted at $v$.

We compute the correct result for all nodes $v$ of $T$ in leaf-to-root order.

1. For each node $v$ of $T$, let $c_v$ be the set of projected attributes that appear in $v$.
2. For each node $v$ of $T$, let $T_v$ be the set of attributes shared by $v$ and its parent (empty if $v$ is the root).
3. For each node $v$ of $T$, let $r_v$ be the parent of $v$.
4. For each node $v$ of $T$, let $p_v$ be the root of the subtree $T_v$.
5. For each node $v$ of $T$, let $c_v$ be the children of $v$.
6. We apply the join as follows:
   - For all nodes $v$ of $T$ in leaf-to-root order do
     - For all modes $c$ of $T_v$ in root-to-leaf order do
       - if $c$ is not the root of $T_v$ then
         - $\text{result}(c) = \emptyset$
       - if $c$ is the root of $T_v$ then
         - $\text{result}(c) = \text{result}(c_v)$
     - The result is $\text{result}(\text{root}(T))$
Tree Decomposition of a Graph

- Let $G$ be a graph
- A Tree Decomposition (TD) of $G$ is a pair $(T, \chi)$ with the following properties:
  - $T$ is a tree
  - $\chi$ is a function that maps every node $t$ of $T$ to a subset (called bag) of $\text{nodes}(G)$, so that:
    - For every edge $e \in \text{edges}(G)$ there is a node $t$ of $T$ such that $e \subseteq (\chi(t))$
    - Every node $v$ of $G$ occurs in a connected subtree of $T$; that is, the set $\{t \in \text{nodes}(T) \mid v \in \chi(t)\}$ induces a connected subtree of $T$
  - The width of $T$ is $\max \{|\chi(t)| \mid t \in \text{nodes}(T)\} - 1$
  - The treewidth of $G$ is the minimal width over all TDs of $G$

Tree Decomposition of a Hypergraph

- Definitions of this part taken from Gottlob et al. [GGM'05]
- Let $\mathcal{H}$ be a hypergraph
- A Tree Decomposition (TD) of $\mathcal{H}$ is a pair $(T, \chi)$ with the following properties:
  - $T$ is a tree
  - $\chi$ is a function that maps every node $t$ of $T$ to a subset (called bag) of $\text{nodes}(\mathcal{H})$, so that:
    - For every hyperedge $e \in \text{edges}(\mathcal{H})$ there is a node $t$ of $T$ such that $e \subseteq (\chi(t))$
    - Every node $v$ of $\mathcal{H}$ occurs in a connected subtree of $T$; that is, the set $\{t \in \text{nodes}(T) \mid v \in \chi(t)\}$ induces a connected subtree of $T$
  - Note: if $(T, \chi)$ is a TD of $\mathcal{H}$, then $T$ is a join tree over the bags
Quality?

- Every hypergraph has a TD!
- In what sense is a TD “good”?
- Depends on the context!
- In our case, we would like be able to efficiently compute the part of the join that corresponds to each bag
- This could be achieved if each bag could be covered by a small number of relations
- Just intuition... Later we show how exactly that helps to get complexity bounds

Generalized Hypertree Width

- The generalized hypertree width (ghw) of a hypergraph \( H \) is the minimum of the widths of all GHDs of \( H \)
- The ghw of a CQ expression \( \alpha \) is the ghw of \( H_{\alpha} \)
- Claim (easy to prove): \( \alpha \) (or \( H \)) is acyclic if and only if its ghw is 1

Utilizing Bounded ghw

We now show how a small (bounded) ghw can be used for efficiently computing a join

CQ Evaluation with a GHD (1)

- Let \( H \) be a hypergraph
- A Generalized Hypertree Decomposition (GHD) of \( H \) is a triple \((T, \chi, \lambda)\) such that:
  - \((T, \chi, \lambda)\) is a tree decomposition of \( H \)
  - \(\lambda\) is a function that maps every node \( t \) of \( T \) to a subset of edges of \( H \) that covers \( \chi(t) \); that is, \( \chi(t) \subseteq \bigcup \lambda(t) \)
  - \( \bigcup \lambda(t) \) means \( \bigcup_{e \in \lambda(t)} e \)
- The width of a GHD \((T, \chi, \lambda)\) is the maximal number of hyperedges needed for covering a node; that is \( \max \{ |\lambda(t)| \ | t \in \text{nodes}(T)\} \)

Notation

- For each hyperedge \( e \) of \( H_{\alpha} \), let:
  - \( R_e \) be the relation symbol that corresponds to \( e \)
  - \( r_e \) be the relation of \( I \) over \( R_e \)
- Given an instance \( I \), we can compute \( \alpha(I) \) as follows
- For each node \( t \) of \( T \) compute the relation
  \[ r(t) := \pi_{\alpha(t)} \left( \bigotimes_{e \in \lambda(t)} r_e \right) \]
- Next, for each relation \( r_i \) find a node \( t \) such that \( \chi(t) \) contains all the attributes of \( R_i \) and set:
  \[ r(t) = r(t) \otimes r_i \]
- That is, delete from \( r(t) \) every tuple that cannot be joined with any tuple from \( r_i \)
Now we have the following:
- $M_{t_1, t_2} = \pi_A(selfjoin(T)) r(t)$
- $\pi_A(M_{t_1, t_2}) = \pi_A(M_{\text{hyperjoin}(T)} r(t))$
- $\pi_A(M_{\text{hyperjoin}(T)} r(t))$ is an acyclic CQ expression
- Apply Yannakakis’s to compute $\pi_A(M_{\text{hyperjoin}(T)} r(t))$
- That’s it!

It is NP-complete to decide whether a given hypergraph $H$ has a ghw at most $k$ for any constant $k \geq 3$ [GMS09]

Nevertheless, there is a restricted variant of a GHD, called hypertree decomposition, which can be found in polynomial time for every fixed $k$.
- Basically, it is a GHD with an additional requirement
- We do not discuss hypertree decompositions here, but still:
- We define the Hypertree Width of a hypergraph $H$ as the minimal width over all hypertree decompositions of $H$

Fact: A hypergraph is acyclic if and only if its hypertree width (and ghw) is 1

For every constant $k$, CQ expressions with hypertree width at most $k$ can be evaluated in polynomial total time.*

*In fact, polynomial delay [KS06]
In this part, we focus on a projection-free join query:

\[ Q \leftarrow R_1 \bowtie \cdots \bowtie R_k \]

- For \( i = 1, \ldots, k \), denote by \( \text{Att}(R_i) \) the attribute set of \( R_i \).
- Denote by \( \text{Att}(Q) \) the union \( \bigcup_{i=1}^k \text{Att}(R_i) \).
- A database \( D \) consists of the relation \( r_i \) over each \( R_i \).
- We denote by \( |r_i| \) the number of tuples in \( r_i \).

**Rough Bound**

- Recall that \( Q = R_1 \bowtie \cdots \bowtie R_k \).
- Suppose that \( R_{i_1}, \ldots, R_{i_l} \) contain all (i.e., cover the) attributes in \( \text{Att}(Q) \)
- Then, each tuple \( t \in Q(D) \) is the combination of tuples from \( r_{i_1}, \ldots, r_{i_l} \) that agree on the common attributes
  - And some combinations may not be tuples, due to the other relations
  - Q: How many such combinations can there be?
  - A: At most \( |r_{i_1}| \times \cdots \times |r_{i_l}| \)
  - Hence, \( |Q(D)| \leq |r_{i_1}| \times \cdots \times |r_{i_l}| \)

**Fractional Edge Cover**

- An edge cover of \( Q \) is a sequence \((a_1, \ldots, a_k) \in \{0, 1\}^k \) such that each \( A \in \text{Att} \) occurs in at least one \( R_i \) with \( a_i = 1 \)
- A fractional edge cover of \( Q \) is a sequence \((w_1, \ldots, w_k) \) in \([0, 1]^k \) such that for every \( A \in \text{Att} \) we have
  \[ \sum_{i \in \text{Att}(R_i)} w_i \geq 1 \]
- A fractional edge cover \((w_1, \ldots, w_k) \) of \( Q \) is optimal if \( \sum_{i \in \text{Att}(R_i)} w_i \) is minimal
- Denote by \((w_1^*, \ldots, w_k^*)\) an optimal edge cover of \( Q \)

**Warm-Up Discussion**

- How many answers can be for the following queries, in terms of \([r_1], \ldots, [r_k]\)?
  - \( R_1(A) \bowtie R_2(B) \bowtie R_3(C) \)
  - \( R_1(A, B) \bowtie R_2(B, C) \bowtie R_3(C, A) \bowtie R_4(A, B) \)
  - \( R_1(A, B) \bowtie R_2(B, C) \bowtie R_3(C, A) \bowtie R_4(A, B, C) \)

**AGM Bound**

**Theorem**

- For every fractional edge cover \((w_1, \ldots, w_k) \) of \( Q \) we have
  \[ Q(D) \leq \prod_{i=1}^k |r_i|^{w_i} \]
- For every \( N_0 \in \mathbb{N} \) there is a database \( D \) with \( N > N_0 \) tuples such that
  \[ Q(D) \geq \prod_{i=1}^k |r_i|^{w_i^*} \]
  and \( |r_i^*| = |r_i| \) whenever \( w^*_i, w_i > 0 \).


Worst-Case Optimality

- An algorithm for computing $Q$ is worst-case optimal if its running time is $O(f(|r_1|, \ldots, |r_n|))$ where $f(n_1, \ldots, n_k)$ is the maximal $O(|D|)$ over all databases $D$ with $|r_i| = n_i$.
- Starting with Ngo et al. [NPRR12], in recent years several worst-case optimal join algorithms have been devised [Vel14, KNRR15, KEK17].
- In particular, the running time of these algorithms does not exceed the AGM bound.

(The algorithms themselves are beyond the scope of the course.)

References


References


References


References
