Approximation of Conjunctive Queries
Possible Approaches

...to address the challenges raised by the volume of big data

- **Scale Independence** – find queries that can be answered regardless of scale

- **Replace** the query with one that is much faster to execute
Minimizing Conjunctive Queries

- Database theory has developed principled methods for optimizing CQs:
  - Find an equivalent CQ with minimal number of atoms (the core)
  - Provides a notion of “true” optimality

\[
\begin{align*}
Q(x) &:= R(x,y), R(x,b), R(a,b), R(u,c), R(u,v), S(a,c,d) \\
\{y \rightarrow b\} & \quad \downarrow \\
Q(x) &:= R(x,b), R(a,b), R(u,c), R(u,v), S(a,c,d) \\
\{v \rightarrow c\} & \quad \downarrow \\
Q(x) &:= R(x,b), R(a,b), R(u,c), S(a,c,d)
\end{align*}
\]

minimal query
Minimizing Conjunctive Queries

• But, a minimal equivalent CQ might not be easier to evaluate – query evaluation remains NP-hard

• However, we know “good” classes of CQs for which query evaluation is tractable (in combined complexity):
  – Graph-based
  – Hypergraph-based
(Hyper)graph of Conjunctive Queries

Q : - R(x,y,z), R(z,u,v), R(v,w,x)
“Good” Classes of Conjunctive Queries

- Graph-based:
  - CQs of **bounded treewidth** – their graph has bounded treewidth

- Hypergraph-based:
  - CQs of **bounded hypertree width** – their hypergraph has bounded hypertree width
  - **Acyclic** CQs – their hypergraph has hypertree width 1
Treewidth of a Graph

- A **tree decomposition** of a graph $G = (V,E)$ is a labeled tree $T = (N,F,\lambda)$, where $\lambda : N \rightarrow 2^V$ such that:
  1. For each node $u \in V$ of $G$, there exists $n \in N$ such that $u \in \lambda(n)$
  2. For each edge $(u,v) \in E$, there exists $n \in N$ such that $\{u,v\} \subseteq \lambda(n)$
  3. For each node $u \in V$ of $G$, the set $\{n \in N \mid u \in \lambda(n)\}$ induces a connected subtree of $T$

{\begin{itemize}
  \item[1] 6 7
  \item[2] 4 8
  \item[3] 5
  \item[4] 2
\end{itemize}}

{\begin{itemize}
  \item[5] {4,6}
  \item[6] {4,5}
  \item[7] {3,4,6}
  \item[8] {4,6,8}
  \item[9] {2,5}
  \item[10] {1,3,6}
  \item[11] {6,7,8}
\end{itemize}}
Treewidth of a Graph

- A tree decomposition of a graph $G = (V,E)$ is a labeled tree $T = (N,F,\lambda)$, where $\lambda : N \rightarrow 2^V$ such that:
  1. For each node $u \in V$ of $G$, there exists $n \in N$ such that $u \in \lambda(n)$
  2. For each edge $(u,v) \in E$, there exists $n \in N$ such that $\{u,v\} \subseteq \lambda(n)$
  3. For each node $u \in V$ of $G$, the set $\{n \in N \mid u \in \lambda(n)\}$ induces a connected subtree of $T$
Treewidth of a Graph

- A tree decomposition of a graph $G = (V,E)$ is a labeled tree $T = (N,F,\lambda)$, where $\lambda : N \to 2^V$ such that:
  1. For each node $u \in V$ of $G$, there exists $n \in N$ such that $u \in \lambda(n)$
  2. For each edge $(u,v) \in E$, there exists $n \in N$ such that $\{u,v\} \subseteq \lambda(n)$
  3. For each node $u \in V$ of $G$, the set $\{n \in N \mid u \in \lambda(n)\}$ induces a connected subtree of $T$

```
1----6----7
|     |     |
3----4----8
      \\
5       \\
2
```

```
{4,6}
/{4,5}  {3,4,6}  {4,6,8}
/{2,5}  {1,3,6}  {6,7,8}
```
Treewidth of a Graph

- A **tree decomposition** of a graph $G = (V, E)$ is a labeled tree $T = (N, F, \lambda)$, where $\lambda : N \rightarrow 2^V$ such that:
  1. For each node $u \in V$ of $G$, there exists $n \in N$ such that $u \in \lambda(n)$
  2. For each edge $(u, v) \in E$, there exists $n \in N$ such that $\{u, v\} \subseteq \lambda(n)$
  3. For each node $u \in V$ of $G$, the set $\{n \in N \mid u \in \lambda(n)\}$ induces a **connected** subtree of $T$

- The **width** of a tree decomposition $T = (N, F, \lambda)$ is $\max_{n \in N} \{|\lambda(n)| - 1\}$

  -1 so that the treewidth of a tree is 1

- The **treewidth** of $G$ is the minimum width over all tree decompositions of $G$
CQs of Bounded Treewidth

**Theorem:** For a fixed $k \geq 0$, $\text{BQE}(\text{CQTW}_k)$ is in PTIME

\[ \{Q \in \text{CQ} \mid \text{the treewidth of } G(Q) \text{ is at most } k \} \]

Actually, if $G(Q)$ has treewidth $k \geq 0$, then $Q$ can be evaluated in time $O(|D|^k)$ + time to compute a tree decomposition for $G(Q)$ of optimal width, which is feasible in linear time.
“Good” Classes of Conjunctive Queries

• Graph-based
  – CQs of bounded treewidth – their graph has bounded treewidth
    ▪ Evaluation is feasible in polynomial time

• Hypergraph-based:
  – CQs of bounded hypertree width – their hypergraph has bounded hypertree width
  – Acyclic CQs – their hypegraph has hypertree width 1
Acyclic Hypergraphs

- A join tree of a hypergraph \( H = (V,E) \) is a labeled tree \( T = (N,F,\lambda) \), where \( \lambda : N \rightarrow E \) such that:
  1. For each hyperedge \( e \in E \) of \( H \), there exists \( n \in N \) such that \( e = \lambda(n) \)
  2. For each node \( u \in V \) of \( H \), the set \( \{ n \in N \mid u \in \lambda(n) \} \) induces a connected subtree of \( T \)
Acyclic Hypergraphs

• A join tree of a hypergraph $H = (V,E)$ is a labeled tree $T = (N,F,\lambda)$, where $\lambda : N \rightarrow E$ such that:
  1. For each hyperedge $e \in E$ of $H$, there exists $n \in N$ such that $e = \lambda(n)$
  2. For each node $u \in V$ of $H$, the set $\{n \in N \mid u \in \lambda(n)\}$ induces a connected subtree of $T$

• **Definition:** A hypergraph is acyclic if it has a join tree

![Prime example of a cyclic hypergraph](image-url)
Acyclic Hypergraphs

- A join tree of a hypergraph $H = (V,E)$ is a labeled tree $T = (N,F,\lambda)$, where $\lambda : N \rightarrow E$ such that:
  1. For each hyperedge $e \in E$ of $H$, there exists $n \in N$ such that $e = \lambda(n)$
  2. For each node $u \in V$ of $H$, the set $\{n \in N \mid u \in \lambda(n)\}$ induces a connected subtree of $T$

- **Definition:** A hypergraph is **acyclic** if it has a join tree

![Diagram of hypergraph and join tree]
Acyclic CQs

**Theorem:** \( \text{BQE}(\text{ACQ}) \) is in PTIME

\( \{ Q \in \text{CQ} | H(Q) \text{ is acyclic} \} \)

Actually, if \( H(Q) \) is acyclic, then \( Q \) can be evaluated in time \( O(|D| \cdot |Q|) \), i.e., *linear time* in the size of \( D \) and \( Q \).
“Good” Classes of Conjunctive Queries: Recap

- **Graph-based**
  - CQs of bounded treewidth – their graph has bounded treewidth
    - Evaluation is feasible in *polynomial time*

- **Hypergraph-based:**
  - CQs of bounded hypertree width – their hypergraph has bounded hypertree width
    - Evaluation is feasible in *polynomial time*
  - Acyclic CQs – their hypergraph has hypertree width 1
    - Evaluation is feasible in *linear time*

\[
\text{CQHTW}_1 = \text{ACQ} \\
\text{CQHTW}_k \\
\text{CQTW}_1 \\
\text{ACQ} \\
\text{CQTW}_k
\]
Replace a given CQ with one that is much faster to execute

or

Replace a given CQ with one that falls in “good” class of CQs

preferably, with an acyclic CQ

since evaluation is in linear time
**Definition:** A CQ $Q$ is **semantically acyclic** if there exists an acyclic CQ $Q'$ such that $Q \equiv Q'$

\[
\begin{align*}
Q(x,z) & \iff R(x,y), R(y,z), R(x,w), R(w,z) \\
\{w \rightarrow y, z \rightarrow y\} & \downarrow \downarrow \downarrow \\
Q(x,z) & \iff R(x,y), R(y,z)
\end{align*}
\]
Semantic Acyclicity

**Theorem:** A CQ $Q$ is semantically acyclic iff its core is acyclic

**Theorem:** Deciding whether a CQ $Q$ is semantically acyclic is NP-complete

**Proof idea (upper bound):**
- We can show the following: if $Q$ is semantically acyclic, then there exists an acyclic CQ $Q'$ such that $|Q'| \leq |Q|$ and $Q \equiv Q'$
- Then, we can guess in polynomial time:
  - An acyclic CQ $Q'$ such that $|Q'| \leq |Q|$
  - A mapping $h_1 : \text{terms}(Q) \rightarrow \text{terms}(Q')$
  - A mapping $h_2 : \text{terms}(Q') \rightarrow \text{terms}(Q)$
- And verify in polynomial time that $h_1$ is a query homomorphism from $Q$ to $Q'$ (i.e., $Q' \subseteq Q$), and $h_2$ is a query homomorphism from $Q'$ to $Q$ (i.e., $Q \subseteq Q'$)
Semantic Acyclicity

**Theorem:** A CQ $Q$ is semantically acyclic iff its core is acyclic

**Theorem:** Deciding whether a CQ $Q$ is semantically acyclic is NP-complete

But, semantic acyclicity is rather weak:

- Not many CQs are semantically acyclic
  - $\Rightarrow$ consider **acyclic approximations** of CQs

- Semantic acyclicity is not an improvement over usual optimization – both approaches are based on the core
  - $\Rightarrow$ exploit **semantic information** in the form of constraints
Acyclic Approximations of CQs
Acyclic Approximations

If our CQ $Q$ is not semantically acyclic, we may target a CQ that is:

1. Easy to evaluate – acyclic
2. Provides sound answers – contained in $Q$
3. As “informative” as possible – “maximally” contained in $Q$

**Definition:** A CQ $Q'$ is an acyclic approximation of $Q$ if:

1. $Q'$ is acyclic
2. $Q' \subseteq Q$
3. There is no acyclic CQ $Q''$ such that $Q' \subset Q'' \subseteq Q$
Do Acyclic Approximations Exist?

The cyclic CQ

\[ Q \leftarrow R(x,y,z), R(z,u,v), R(v,w,x) \]

has several acyclic approximations

\[ Q_1 \leftarrow R(x,y,z), R(z,u,y), R(y,v,x) \]

\[ Q_2 \leftarrow R(x,y,z), R(z,u,v), R(v,w,x), R(x,z,v) \]

\[ Q_3 \leftarrow R(x,y,x) \]
Existence, Size and Computation

**Theorem:** Consider a CQ $Q$. Then:

1. $Q$ has an acyclic approximation

2. Each acyclic approximation of $Q$ has size polynomial in $Q$

3. An acyclic approximation of $Q$ can be found in time $2^{O(|Q| \cdot \log |Q|)}$

4. $Q$ has at most exponentially many (non-equivalent) acyclic approximations
Evaluating Acyclic Approximations

- Recall that evaluating $Q$ over $D$ takes time $|D|^{O(|Q|)}$

- Evaluating an acyclic approximation $Q'$ of $Q$ over $D$ takes time

\[ 2^{O(|Q| \cdot \log |Q|)} + |D| \cdot |Q|^k \]

  time for computing $Q'$

  time for evaluating $Q'$

  - $|Q'| \leq |Q|^k$

  - Evaluation of an acyclic CQ $Q_A$ is feasible in time $O(|D| \cdot |Q_A|)$

- Observe that $2^{O(|Q| \cdot \log |Q|)} + |D| \cdot |Q|^k$ is dominated by $|D| \cdot 2^{O(|Q| \cdot \log |Q|)}$

  \[ \Rightarrow \text{fixed-parameter tractable} \]
Poor Approximations

\[ Q \; \leftarrow \; E(x,y), \; E(y,z), \; E(z,x) \]

has only one acyclic approximation, that is, \( Q' \; \leftarrow \; E(x,x) \)

**Proposition:** Consider a Boolean CQ \( Q \) that contains a single binary relation \( E(.,.) \). If \( G(Q) \) is not bipartite, then the only acyclic approximation of \( Q \) is \( Q' \; \leftarrow \; E(x,x) \)
Acyclic Approximations: Recap

- Acyclic approximations are useful when the CQ is not semantically acyclic
- Always exist, but are not unique
- Have polynomial size, and can be computed in exponential time
- Can be evaluated “efficiently” (fixed-parameter tractability)
- In some cases, acyclic approximations are not very informative
But, semantic acyclicity is rather weak:

- Not many CQs are semantically acyclic
  ⇒ consider acyclic approximations of CQs

- Semantic acyclicity is not an improvement over usual optimization – both approaches are based on the core
  ⇒ exploit semantic information in the form of constraints
Associated Papers


  Eligible topics include static analysis of approximations


  Semantic acyclicity for CQs


  Complexity of semantic acyclicity for CQs (in a different context)

- Víctor Dalmau, Phokion G. Kolaitis, Moshe Y. Vardi: Constraint Satisfaction, Bounded Treewidth, and Finite-Variable Logics. CP 2002: 310-326

  Evaluation of semantically acyclic CQ (in a different context)
Associated Papers


  A different way of measuring complexity, and its full analysis


  Using tree decompositions to get faster query evaluation


  How to improve performance of relational queries on databases with special properties
Associated Papers

• Minos N. Garofalakis, Phillip B. Gibbons: Approximate Query Processing: Taming the TeraBytes. VLDB 2001

  Approximation techniques that take into account both data and queries


  An in-depth study of acyclicity


  A hierarchy of classes of efficient CQs, the bottom level of which is acyclic queries
Associated Papers

- Martin Grohe, Thomas Schwentick, Luc Segoufin: When is the evaluation of conjunctive queries tractable? STOC 2001: 657-666
  Characterizing efficiency of CQs via the notion of bounded treewidth

  Approximation techniques that take into account both data and queries

  Notion of acyclicity of CQs and fast evaluation scheme based on it