Graphs, Hypergraphs, and the Complexity of Conjunctive Database Queries

Dániel Marx

Institute for Computer Science and Control,
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

ICDT Invited Lecture 2017, Venice, Italy
March 23, 2017

Conjunctive queries

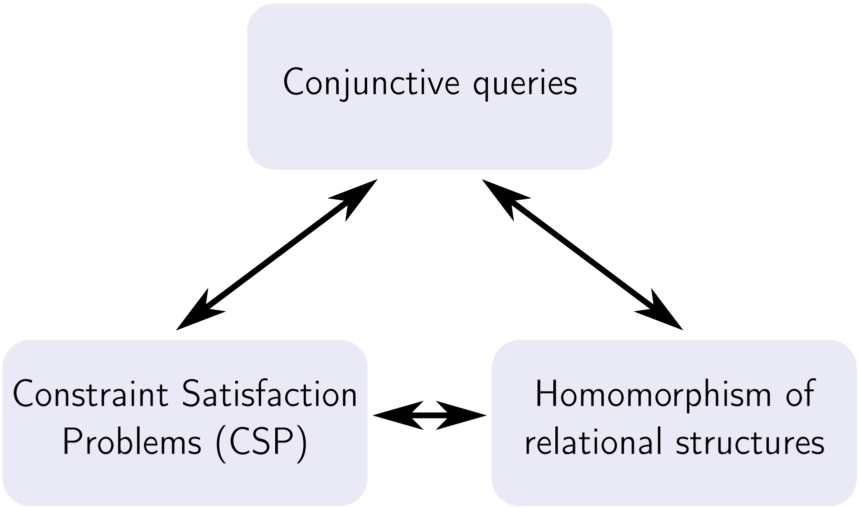
Evaluating conjunctive queries is a fundamental problem.

$$Q = R(A, B, C) \wedge S(C, D) \wedge T(B, C, E)$$

Formally defined as:

$$Q = \{(a, b, c, d, e) \mid (a, b, c) \in R, (c, d) \in S, (b, c, e) \in T\}$$

- Compute the answer relation Q.
- Decide if the relation Q is empty.
- Compute the size of Q.
- ...


Conjunctive queries
Constraint Satisfaction Problems (CSP)
Homomorphism of relational structures

Constraint Satisfaction Problems (CSP)

$$Q = R(A, B, C) \wedge S(C, D) \wedge T(B, C, E)$$

CSP lingo:

- variables A, B, C, D, E
- constraints R, S, T
- find an assignment (a, b, c, d, e) to the variables that satisfies every constraint.

Tasks:

Compute the answer relation.		List the satisfying assignments.
Decide if Q is empty.	$\Leftrightarrow$	Decide if the CSP is satisfiable.
Compute the size of Q.		Count the sat. assignments.

Goal

Goal: understand how efficiently a particular query can be evaluated.

- Worst-case setting: we know the query, but the database relations can be arbitrary.
- Different levels of efficiency: polynomial time, fixed-parameter tractability, linear time.

Goal

Goal: understand how efficiently a particular query can be evaluated.

- Worst-case setting: we know the query, but the database relations can be arbitrary.
- Different levels of efficiency: polynomial time, fixed-parameter tractability, linear time.

Important message:

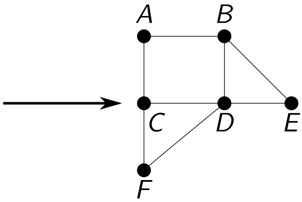
> "Treelikeness" is very helpful!

...because it allows bottom-up dynamic programming.

First: binary relations only

If every relation is binary (i.e., only two variables), then the structure of the query can be described by the **primal graph.**

$$R(A, B) \land R(A, C) \land$$
$$R(B, D) \land R(C, D) \land$$
$$R(B, E) \land R(D, E) \land$$
$$R(C, F) \land R(D, F)$$

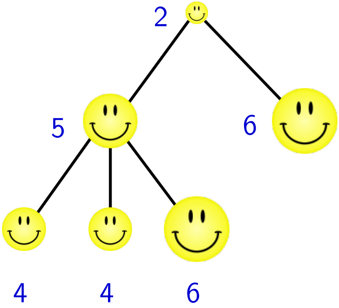


Goal: understand what graph-theoretic properties allow efficient query evaluation.

The Party Problem

Party Problem

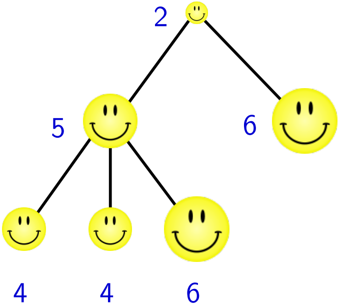
Problem:	Invite some colleagues for a party.
Maximize:	The total fun factor of the invited people.
Constraint:	Everyone should be having fun.



The Party Problem

Party Problem

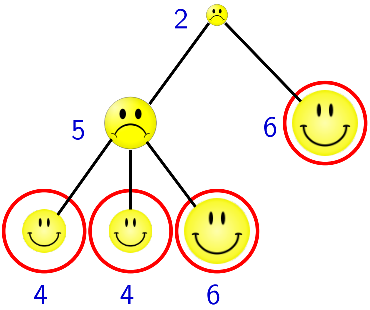
Problem:	Invite some colleagues for a party.
Maximize:	The total fun factor of the invited people.
Constraint:	Everyone should be having fun.
	Do not invite a colleague and his direct boss at the same time!



The Party Problem

Party Problem

Problem:	Invite some colleagues for a party.
Maximize:	The total fun factor of the invited people.
Constraint:	Everyone should be having fun.
	Do not invite a colleague and his direct boss at the same time!

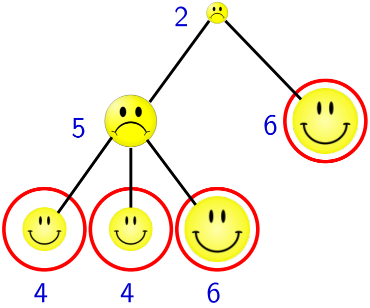


- **Input:** A tree with weights on the vertices.
- **Task:** Find an independent set of maximum weight.

The Party Problem

Party Problem

Problem: Invite some colleagues for a party.
Maximize: The total fun factor of the invited people.
Constraint: Everyone should be having fun.
Do not invite a colleague and
his direct boss at the same time!



- **Input:** A tree with weights on the vertices.
- **Task:** Find an independent set of maximum weight.

Solving the Party Problem

Dynamic programming paradigm:
We solve a large number of subproblems that depend on each other. The answer is a single subproblem.

Subproblems:

T_v: the subtree rooted at v.
$A[v]$: max. weight of an independent set in T_v
$B[v]$: max. weight of an independent set in T_v that does not contain v

Goal: determine $A[r]$ for the root r.

Solving the Party Problem

Subproblems:

T_v: the subtree rooted at v.

$A[v]$: max. weight of an independent set in T_v

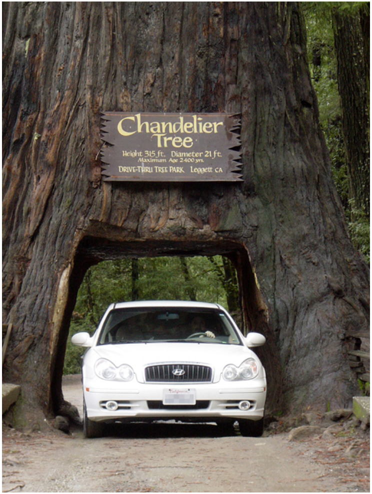
$B[v]$: max. weight of an independent set in T_v that does not contain v

Recurrence:

Assume $v_1, \ldots, v_k$ are the children of v. Use the recurrence relations

$$\boxed{\begin{array}{l} B[v] = \sum_{i=1}^{k} A[v_i] \\ A[v] = \max\{B[v]\ ,\ w(v) + \sum_{i=1}^{k} B[v_i]\} \end{array}}$$

The values $A[v]$ and $B[v]$ can be calculated in a bottom-up order (the leaves are trivial).


Chandelier Tree
Height 315 ft. Diameter 21 ft.
Maximum Age 2400 yrs.
DRIVE-THRU TREE PARK Leggett CA

Treewidth

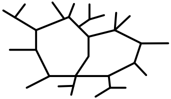
Generalizing trees

How could we define that a graph is "treelike"?

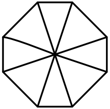
Generalizing trees

How could we define that a graph is "treelike"?

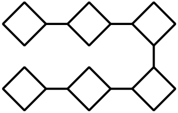
1. Number of cycles is bounded.



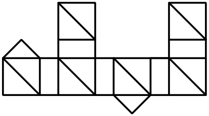
good



bad



bad

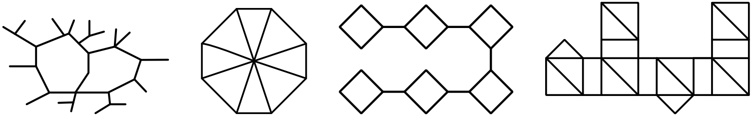


bad

Generalizing trees

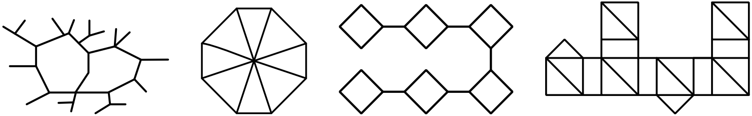
How could we define that a graph is “treelike”?

1. Number of cycles is bounded.



good bad bad bad

2. Removing a bounded number of vertices makes it acyclic.

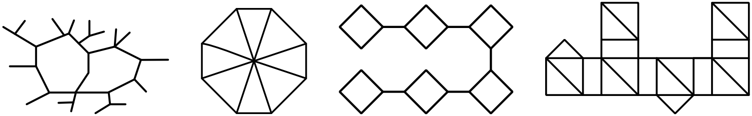


good good bad bad

Generalizing trees

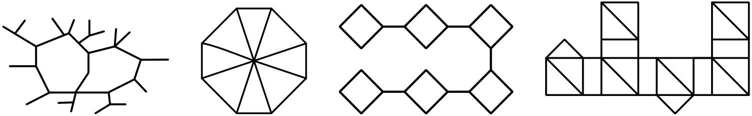
How could we define that a graph is “treelike”?

1. Number of cycles is bounded.



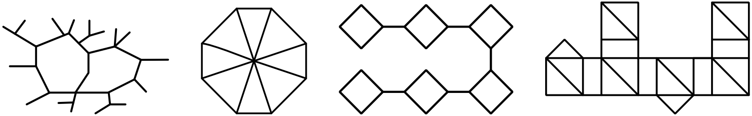
good bad bad bad

2. Removing a bounded number of vertices makes it acyclic.



good good bad bad

3. Bounded-size parts connected in a tree-like way.



bad bad good good

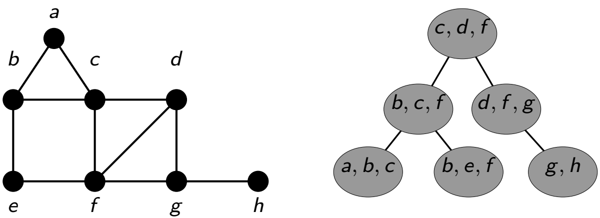
Treewidth — a measure of "tree-likeness"

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. For any edge uv, there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.



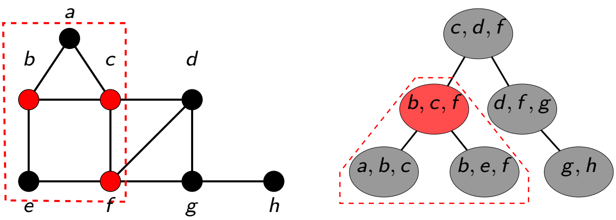
Treewidth — a measure of "tree-likeness"

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. For any edge uv, there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.



A subtree communicates with the outside world only via the root of the subtree.

WEIGHTED MAX INDEPENDENT SET and treewidth

Theorem

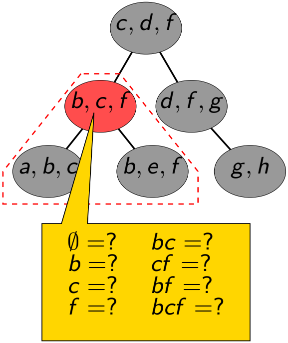
Given a tree decomposition of width w, WEIGHTED MAX INDEPENDENT SET can be solved in time $2^w \cdot w^{O(1)} \cdot n$.

B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.

Generalizing our solution for trees:

Instead of computing 2 values $A[v]$, $B[v]$ for each vertex of the tree, we compute $2^{|B_x|} \leq 2^{w+1}$ values for each bag B_x.

$M[x, S]$:
the max. weight of an independent set $I \subseteq V_x$ with $I \cap B_x = S$.



WEIGHTED MAX INDEPENDENT SET and treewidth

Theorem

Given a tree decomposition of width w, WEIGHTED MAX INDEPENDENT SET can be solved in time $2^w \cdot w^{O(1)} \cdot n$.

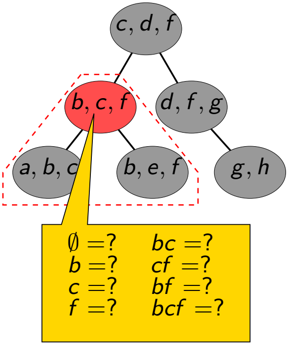
B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.

Generalizing our solution for trees:

Instead of computing 2 values $A[v]$, $B[v]$ for each vertex of the tree, we compute $2^{|B_x|} \leq 2^{w+1}$ values for each bag B_x.

$M[x, S]$:
the max. weight of an independent set $I \subseteq V_x$ with $I \cap B_x = S$.

Claim: We can determine $M[x, S]$ if all the values are known for the children of x.



3-Coloring and tree decompositions

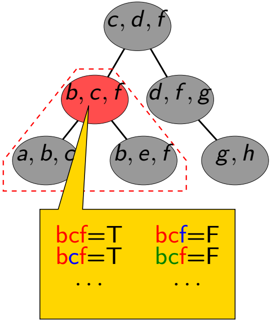
Theorem

Given a tree decomposition of width w, 3-Coloring can be solved in time $3^w \cdot w^{O(1)} \cdot n$.

B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.

For every node x and coloring $c : B_x \to \{1, 2, 3\}$, we compute the Boolean value $E[x, c]$, which is true if and only if c can be extended to a proper 3-coloring of V_x.

Claim:
We can determine $E[x, c]$ if all the values are known for the children of x.



Coloring as a CSP

We can interpret 3-coloring as a CSP:

- vertices $\Leftrightarrow$ variables
- domain $D = \{r, g, b\}$
- edges $\Leftrightarrow$ inequality constraints

$$R = \{(x, y) \in D \times D \mid x \neq y\}$$

Straightforward generalization to higher number of colors:

Theorem

Given a tree decomposition of width w, c-Coloring can be solved in time $c^{w+1} \cdot w^{O(1)} \cdot n$.

Coloring as a CSP

We can interpret 3-coloring as a CSP:

- vertices $\Leftrightarrow$ variables
- domain $D = \{r, g, b\}$
- edges $\Leftrightarrow$ inequality constraints

$$R = \{(x, y) \in D \times D \mid x \neq y\}$$

Straightforward generalization to arbitrary binary CSPs:

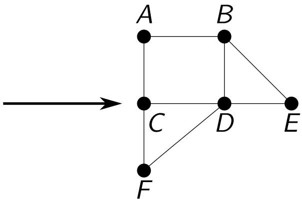
Theorem

Given a tree decomposition of width w, binary CSP over domain D can be solved in time $|D|^{w+1} \cdot w^{O(1)} \cdot n$.

Coloring as a database query

- vertices $\Leftrightarrow$ variables
- edges $\Leftrightarrow$ relation $R = \{rg, rb, gr, gb, br, bg\}$

$$R(A,B) \wedge R(A,C) \wedge$$
$$R(B,D) \wedge R(C,D) \wedge$$
$$R(B,E) \wedge R(D,E) \wedge$$
$$R(C,F) \wedge R(D,F)$$



Straightforward generalization to arbitrary binary queries:

Theorem

Given a tree decomposition of width w, a Boolean Conjunctive Query where every variable allows at most N different values can can be solved in time $N^{w+1} \cdot |Q|^{O(1)}$.

Projections

Projecting the relation $R(A, B, C, D)$ to $\{A, B\}$:

$$R_{|AB} = \{(a, b) \mid \exists c, d : (a, b, c, d) \in R\}$$

Projection of the query to a set S: projecting every relation.

$$Q = R(A, B, C) \wedge S(C, D) \wedge T(B, C, E)$$

$$\begin{aligned} Q_{|AB} &= R_{|AB}(A, B, C) \wedge S_{|AB}(C, D) \wedge T_{|AB}(B, C, E) \\ &= R_{|AB}(A, B, C) \wedge T_{|B}(B, C, E) \end{aligned}$$

Easy: If $(a, b, c) \in Q$, then $(a, b) \in Q_{|AB}$, but not necessarily the other way around!

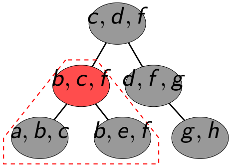
Boolean Conjunctive Queries and tree decompositions

Theorem

Given a tree decomposition of width w, a Boolean Conjunctive Query where every variable allows at most N different values can can be solved in time $N^{w+1} \cdot |Q|^{O(1)}$.

B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.

For every node x and tuple $t \in Q_{|B_x}$, we compute the Boolean value $E[x, t]$, which is true if and only if t can be extended to a tuple of $Q_{|V_x}$.



Claim:
We can determine $E[x, t]$ if all the values are known for the children of x.

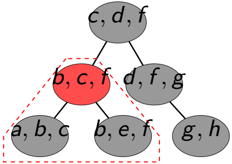
Boolean Conjunctive Queries and tree decompositions

Theorem

Given a tree decomposition of width w, a Boolean Conjunctive Query where every variable allows at most N different values can can be solved in time $N^{w+1} \cdot |Q|^{O(1)}$.

B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.

For every node x and tuple $t \in Q_{|B_x}$, we compute the Boolean value $E[x, t]$, which is true if and only if t can be extended to a tuple of $Q_{|V_x}$.



Running time:
Dominating factor is the size of $Q_{|B_x}$, which can be bounded by $N^{|B_x|} \leq N^{w+1}$.

Tractable classes

We have seen that for every fixed bound on the treewidth, BCQ is polynomial-time solvable in the size of the database.

Are there other properties that make the problem polynomial-time solvable?

Tractable classes

Formally:

If $\mathcal{G}$ is a class of graphs with bounded treewidth, then BCQ restricted $\mathcal{G}$ (we call it $\mathrm{BCQ}(\mathcal{G})$) is polynomial-time solvable.
Are there other such classes?

Tractable classes

Formally:

If $\mathcal{G}$ is a class of graphs with bounded treewidth, then BCQ restricted $\mathcal{G}$ (we call it BCQ($\mathcal{G}$)) is polynomial-time solvable.
Are there other such classes?

An equally interesting question: we can relax polynomial time and allow arbitrary dependence on the length of the query.

$\Rightarrow$ Fixed-parameter tractability

Fixed-parameter tractability

Main definition

A parameterized problem is **fixed-parameter tractable (FPT)** if there is an $f(k)n^c$ time algorithm for some constant c.

Main goal of parameterized complexity: to find FPT problems.

Fixed-parameter tractability

Main definition

A parameterized problem is **fixed-parameter tractable (FPT)** if there is an $f(k)n^c$ time algorithm for some constant c.

Main goal of parameterized complexity: to find FPT problems.

Examples of NP-hard problems that are FPT:

- Finding a vertex cover of size k.
- Finding a path of length k.
- Finding k disjoint triangles.
- Drawing the graph in the plane with k edge crossings.
- Finding disjoint paths that connect k pairs of points.
- ...

W[1]-hardness

Negative evidence similar to NP-completeness. If a problem is **W[1]-hard,** then the problem is not FPT unless FPT=W[1].

Some W[1]-hard problems:

- Finding a clique/independent set of size k.
- Finding a dominating set of size k.
- Finding k pairwise disjoint sets.
- ...

Tractable classes

Theorem [Grohe, Schwentick, Segoufin 2001]

Let $\mathcal{G}$ be a computable class of graphs. Then assuming FPT $\neq W[1]$, the following are equivalent:

- BCQ($\mathcal{G}$) is polynomial-time solvable.
- BCQ($\mathcal{G}$) is FPT.
- $\mathcal{G}$ has bounded treewidth.

Tractable classes

Theorem [Grohe, Schwentick, Segoufin 2001]

Let $\mathcal{G}$ be a computable class of graphs. Then assuming FPT $\neq W[1]$, the following are equivalent:

- BCQ($\mathcal{G}$) is polynomial-time solvable.
- BCQ($\mathcal{G}$) is FPT.
- $\mathcal{G}$ has bounded treewidth.

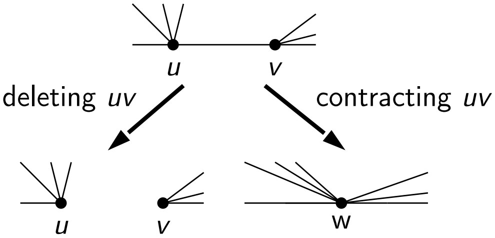
Two surprises:

- Treewidth-based algorithms already solve every polynomial-time solvable case.
- FPT does not give us extra power over polynomial time.

Minors

Definition

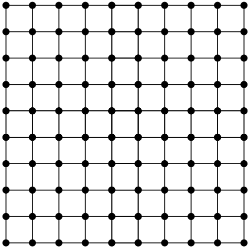
Graph H is a **minor** of G ($H \le G$) if H can be obtained from G by deleting edges, deleting vertices, and contracting edges.



Excluded Grid Theorem

Theorem [Chuzhoy 2016] [Chekuri and Chuzhoy 2014]

Every graph with treewidth at least $k^{19}\text{polylog}(k)$ has a $k \times k$ grid minor.



The $k \times k$ grid has treewidth exactly k.

Tractable classes

Theorem [Grohe, Schwentick, Segoufin 2001]

Let $\mathcal{G}$ be a computable class of graphs. Then assuming $\mathsf{FPT} \neq W[1]$, the following are equivalent:

- $\mathsf{BCQ}(\mathcal{G})$ is polynomial-time solvable.
- $\mathsf{BCQ}(\mathcal{G})$ is FPT.
- $\mathcal{G}$ has bounded treewidth.

Tractable classes

Theorem [Grohe, Schwentick, Segoufin 2001]

Let $\mathcal{G}$ be a computable class of graphs with unbounded treewidth. Then assuming FPT $\neq W[1]$, BCQ($\mathcal{G}$) is not FPT.

- Assuming FPT $\neq W[1]$, k-Clique is not FPT.
- k-Clique can be simulated by a BCQ whose primal graph is a $k \times k$ grid.
- $\mathcal{G}$ has unbounded treewidth
  $\Rightarrow$ Excluded Grid Theorem
  $\Rightarrow$ $\mathcal{G}$ contains graphs with a $k \times k$ grid minor
  $\Rightarrow$ BCQ($\mathcal{G}$) can simulate BCQ's with $k \times k$ grid structure.

Can you beat treewidth?

We have seen that treewidth-based algorithms discover every polynomial time solvable class.

- Is there a class $\mathcal{G}$ where we can be significantly faster than the treewidth-based algorithm? E.g., running time $N^{\sqrt{\mathrm{tw}(Q)}}$ or $N^{(\mathrm{tw}(\mathrm{Q}))^{1/100}}$ or $N^{(\log \log \mathrm{tw}(\mathrm{Q}))}$.

Can you beat treewidth?

We have seen that treewidth-based algorithms discover every polynomial time solvable class.

- Is there a class $\mathcal{G}$ where we can be significantly faster than the treewidth-based algorithm? E.g., running time $N^{\sqrt{\mathrm{tw}(Q)}}$ or $N^{(\mathrm{tw(Q)})^{1/100}}$ or $N^{(\log\log \mathrm{tw(Q)})}$.

Theorem [M. 2007]

Let $\mathcal{G}$ be a computable class of graphs. Assuming the Exponential-time Hypothesis, there is no algorithm for $\mathrm{BCQ}(\mathcal{G})$ with running time $f(Q)N^{o(\mathrm{tw}(Q)/\log \mathrm{tw(Q)})}$.

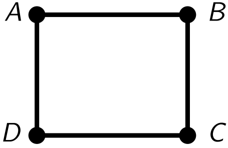
Exponential-time Hypothesis:
There is no $2^{o(n)}$ time algorithm for n-variable 3SAT.

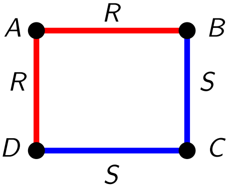
Proof requires a tighter combinatorial understanding of what large treewidth means.

Homomorphisms

The primal graph loses information if some relation appears more than once in the query.

$$Q = R(A, B) \wedge S(B, C) \wedge R(A, D) \wedge S(D, C)$$

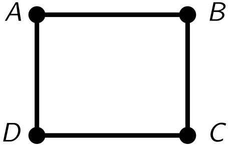


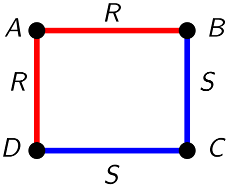


Homomorphisms

The primal graph loses information if some relation appears more than once in the query.

$$Q = R(A, B) \wedge S(B, C) \wedge R(A, D) \wedge S(D, C)$$





This is empty if and only if

$$Q' = R(A, B) \wedge S(B, C)$$

is empty!

Homomorphisms

A **homomorphism** from Q to Q' is a mapping ϕ of the variables of Q to the variables of Q' such that if $R(A, B)$ appears in Q, then $R(\phi(A), \phi(B))$ appears in Q'.

Observation:

- If there is a homomorphism $Q \to Q'$ and Q' is nonempty, then Q is nonempty as well.
- If there is a homomorphism from Q to a subquery Q', then Q is empty $\Leftrightarrow$ Q' is empty.

Homomorphisms

A **homomorphism** from Q to Q' is a mapping ϕ of the variables of Q to the variables of Q' such that if $R(A, B)$ appears in Q, then $R(\phi(A), \phi(B))$ appears in Q'.

Observation:

- If there is a homomorphism $Q \to Q'$ and Q' is nonempty, then Q is nonempty as well.
- If there is a homomorphism from Q to a subquery Q', then Q is empty $\Leftrightarrow$ Q' is empty.

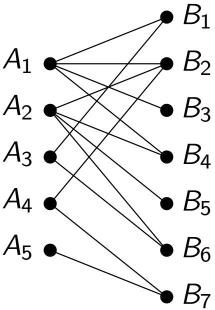
Fact: Every query Q has a unique (up to isomorphism) smallest subquery Q' with a homomorphism $Q \to Q'$. This is the **core** of Q.

For Boolean Conjunctive Queries, it is only the core of the query that matters!

Homomorphisms

What is the core of

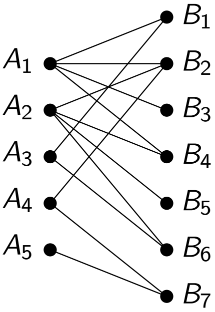
$$Q = R(A_1, B_1) \wedge R(A_1, B_2) \wedge R(A_2, B_2) \wedge R(A_1, B_3) \wedge R(A_1, B_4) \wedge R(A_2, B_4) \wedge R(A_2, B_5) \wedge R(A_2, B_6) \wedge R(A_3, B_1) \wedge R(A_3, B_6) \wedge R(A_4, B_2) \wedge R(A_4, B_7) \wedge R(A_5, B_7)?$$



Homomorphisms

What is the core of

$$Q = R(A_1, B_1) \wedge R(A_1, B_2) \wedge R(A_2, B_2) \wedge R(A_1, B_3) \wedge R(A_1, B_4) \wedge R(A_2, B_4) \wedge R(A_2, B_5) \wedge R(A_2, B_6) \wedge R(A_3, B_1) \wedge R(A_3, B_6) \wedge R(A_4, B_2) \wedge R(A_4, B_7) \wedge R(A_5, B_7)?$$



It is just $R(A_1, B_1)$! (As the graph is bipartite.)

Homomorphisms

Theorem [Grohe 2003]

Let $\mathcal{Q}$ be a computable class of queries with binary relations. Then assuming FPT $\neq W[1]$, the following are equivalent:

- BCQ restricted to queries $\mathcal{Q}$ is is polynomial-time solvable.
- BCQ restricted to queries $\mathcal{Q}$ is FPT.
- The primal graph of the core of every query in $\mathcal{Q}$ has bounded treewidth.

Homomorphisms

Theorem [Grohe 2003]

Let $\mathcal{Q}$ be a computable class of queries with binary relations. Then assuming $\text{FPT} \neq W[1]$, the following are equivalent:

- BCQ restricted to queries $\mathcal{Q}$ is is polynomial-time solvable.
- BCQ restricted to queries $\mathcal{Q}$ is FPT.
- The primal graph of the core of every query in $\mathcal{Q}$ has bounded treewidth.

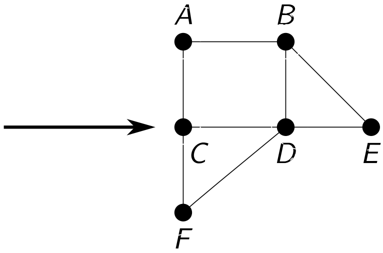
Theorem [M. 2007]

Let $\mathcal{Q}$ be a computable class of queries with binary relations. Assuming the Exponential-time Hypothesis, there is no algorithm for BCQ restricted to $\mathcal{Q}$ with running time $f(Q)N^{o(\text{ctw}(Q)/\log \text{ctw}(Q))}$, where $\text{ctw}(Q)$ is the treewidth of the core of the primal graph of Q.

Next: relations of arbitrary arity

Primal graph: vertices are the variables, two vertices are adjacent if they appear in a common relation of the query.

$R(A, B) \wedge R(A, C) \wedge$
$R(B, D, E) \wedge R(C, D, F)$

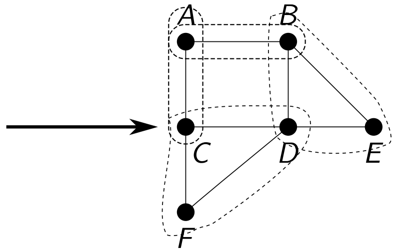


Most of the theoretical results go through for fixed constant arity.

Next: relations of arbitrary arity

Primal graph: vertices are the variables, two vertices are adjacent if they appear in a common relation of the query.

$$R(A,B) \wedge R(A,C) \wedge$$
$$R(B,D,E) \wedge R(C,D,F)$$



Most of the theoretical results go through for fixed constant arity.

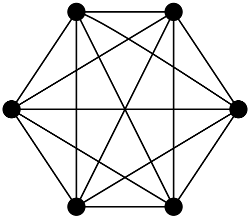
But for undbounded arities we need to look at the **hypergraph** of the query!

Primal graph vs. hypergraphs

The primal graph loses a lot of information if arity is unbounded.

$$Q_1 = \bigwedge_{i \neq j} R(A_i, A_j)$$

$$Q_2 = R(A_1, \ldots, A_k)$$



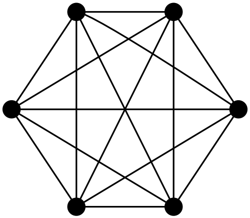
- Queries of the form Q_1 are hard: binary relations with large treewidth.
- Queries of the form Q_2 are trivial: N tuples to consider.

Primal graph vs. hypergraphs

The primal graph loses a lot of information if arity is unbounded.

$$Q_1 = \bigwedge_{i \neq j} R(A_i, A_j)$$

$$Q_2 = R(A_1, \dots, A_k) \wedge S(A_2, A_3, A_5) \wedge T(A_3, A_8) \dots$$



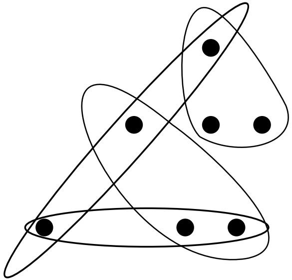
- Queries of the form Q_1 are hard: binary relations with large treewidth.
- Queries of the form Q_2 are trivial: N tuples to consider.

What do we know about bounding the size of the answer?

(…and enumerating all solutions)

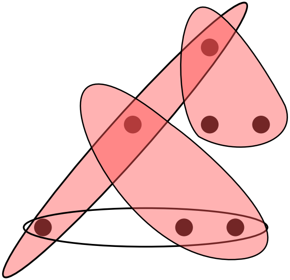
Upper bound

Observation: If the hypergraph has edge cover number ρ and every relation has size at most N, then there are at most N^{ρ} tuples in the answer.



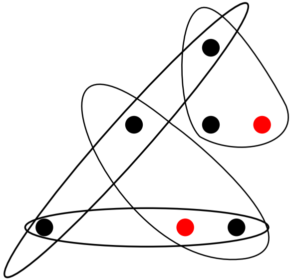
Upper bound

Observation: If the hypergraph has edge cover number ρ and every relation has size at most N, then there are at most N^{ρ} tuples in the answer.



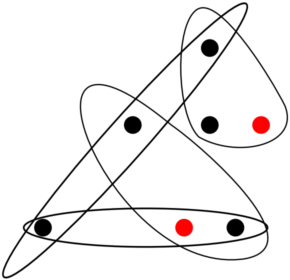
Lower bound

Observation: If the hypergraph has independence number α, then one can construct an instance where every relation has size N at the answer has size N^{α}.

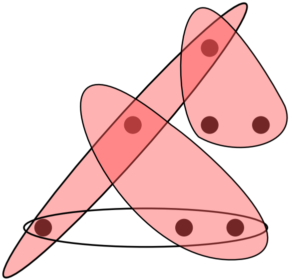


Definition of the relations:

- If variable A is in the independent set, then it can take any value in $[N]$.
- Otherwise it is forced to 1.



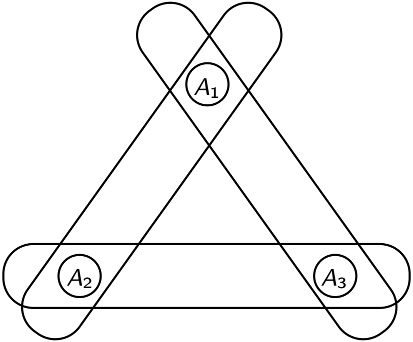
N^2



N^3

Which is tight: the upper bound or the lower bound?

Example: triangles



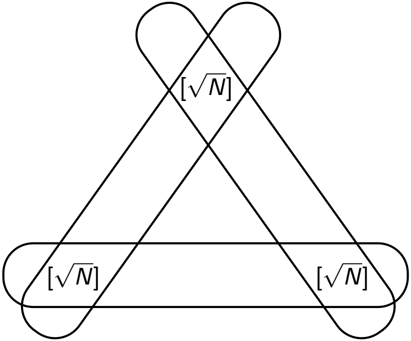
Upper bound

Two kind of values for A_1:

- Light: can be extended to at most $\sqrt{N}$ ways to A_2.
  $\Rightarrow \leq N \cdot \sqrt{N}$ answers with light A_1
- Heavy: can be extended to at least $\sqrt{N}$ ways to A_2.
  $\Rightarrow \leq \sqrt{N}$ heavy values $\Rightarrow \leq \sqrt{N} \cdot N$ answers with heavy A_1

$\Rightarrow$ At most $2 \cdot N^{3/2}$ answers.

Example: triangles



Lower bound

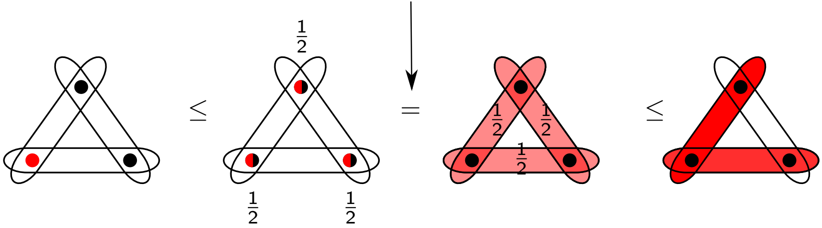
Allow every variable to be any value from $[\sqrt{N}] \Rightarrow N^{3/2}$ answers.

The correct bound $N^{3/2}$ is between
$N^{\alpha} = N^1$ and $N^{\rho} = N^2$.

Fractional values

- α: independence number
- α^*: fractional independence number
  (max. weight of vertices s.t. each edge contains weight ≤ 1)
- ρ^*: fractional edge cover number
  (min. weight of edges s.t. each vertex receives weight ≥ 1)
- ρ: edge cover number

LP duality!



$\alpha = 1$ $\qquad$ $\alpha^* = 3/2$ $\qquad$ $\rho^* = 3/2$ $\qquad$ $\rho = 2$

Tight bound

Theorem [Atserias, Grohe, M. 2008]

Consider a query with fractional edge cover number ρ^*.

- If every relation has size at most N, there are at most N^{ρ^*} answers.
- For every N, one can construct relations of size $\leq N$ such that there are $\approx N^{\rho^*}$ answers.

Upper bound

Follows from classic combinatorial/probabilistic/geometric results (Shearer's Lemma, Submodularity of Entropy, Loomis-Whitney, ...)

Tight bound

Theorem [Atserias, Grohe, M. 2008]

Consider a query with fractional edge cover number ρ^*.

- If every relation has size at most N, there are at most N^{ρ^*} answers.
- For every N, one can construct relations of size $\leq N$ such that there are $\approx N^{\rho^*}$ answers.

Lower bound

Let f be a max. fractional independent set. Allow variable A to have any value from $[N^{f(A)}]$.

Size of relation R:

$$\prod_{A \text{ in } R} N^{f(A)} = N^{\sum_{A \in a(R)} f(A)} \leq N^1$$

Answer size:

$$\prod_{A} N^{f(A)} = N^{\sum_A f(A)} = N^{\alpha^*} = N^{\rho^*}$$

Enumerating all solutions

Can we find all solutions in time roughly N^{ρ^*}?

Possible approaches:

- Join plan
- Join-Project plan
- Something else

Join-Project plans

$Q_i = Q_{|A_1,\dots,A_i}$ — projection to the first i variables.

Observation 1:
$\rho^*(Q_i) \leq \rho^*(Q)$, so the N^{ρ^*} upper bound holds for every Q_i.

Join-Project plans

$Q_i = Q_{|A_1,\ldots,A_i}$ — projection to the first i variables.

Observation 1:
$\rho^*(Q_i) \leq \rho^*(Q)$, so the N^{ρ^*} upper bound holds for every Q_i.

Observation 2:
Q_i can be computed from Q_{i-1} in time N^{ρ^*+1}:

$$Q_i = ((\ldots(Q_{i-1} \bowtie R_{1|A_1,\ldots,A_i})) \bowtie R_{2|A_1,\ldots,A_i})\ldots \bowtie R_{m|A_1,\ldots,A_i}$$

$\Rightarrow$ Simple Join-Project plan in N^{ρ^*+1} time.

Join-Project plans

$Q_i = Q_{|A_1,\ldots,A_i}$ — projection to the first i variables.

Observation 1:
$\rho^*(Q_i) \leq \rho^*(Q)$, so the N^{ρ^*} upper bound holds for every Q_i.

Observation 2:
Q_i can be computed from Q_{i-1} in time N^{ρ^*+1}:

$$Q_i = ((\ldots(Q_{i-1} \bowtie R_{1|A_1,\ldots,A_i})) \bowtie R_{2|A_1,\ldots,A_i}) \ldots \bowtie R_{m|A_1,\ldots,A_i}$$

$\Rightarrow$ Simple Join-Project plan in N^{ρ^*+1} time.

- Do we need projections?
- Can we get rid of the $+1$?

Example

Our "favorite hypergraph": $2m$ relations, $\binom{2m}{m}$ variables, each contained in exactly m relations.

$m = 2$:

$$R_1(A_{12}, A_{13}, A_{14}) \wedge R_2(A_{12}, A_{23}, A_{24}) \wedge$$
$$R_3(A_{13}, A_{23}, A_{34}) \wedge R_4(A_{14}, A_{24}, A_{34})$$

Example

Our "favorite hypergraph": $2m$ relations, $\binom{2m}{m}$ variables, each contained in exactly m relations.

$$
\begin{aligned}
m = 3\text{:}\quad & R_1(A_{123}, A_{124}, A_{125}, A_{126}, A_{134}, A_{135}, A_{136}, A_{145}, A_{146}, A_{156}) \wedge \\
& R_2(A_{123}, A_{124}, A_{125}, A_{126}, A_{234}, A_{235}, A_{236}, A_{245}, A_{246}, A_{256}) \wedge \\
& R_3(A_{123}, A_{134}, A_{135}, A_{136}, A_{234}, A_{235}, A_{236}, A_{245}, A_{246}, A_{256}) \wedge \\
& R_4(A_{124}, A_{134}, A_{145}, A_{146}, A_{234}, A_{245}, A_{246}, A_{345}, A_{346}, A_{456}) \wedge \\
& R_5(A_{125}, A_{135}, A_{145}, A_{156}, A_{235}, A_{245}, A_{256}, A_{345}, A_{356}, A_{456}) \wedge \\
& R_6(A_{126}, A_{136}, A_{146}, A_{156}, A_{236}, A_{246}, A_{256}, A_{346}, A_{356}, A_{456})
\end{aligned}
$$

Example

Our "favorite hypergraph": $2m$ relations, $\binom{2m}{m}$ variables, each contained in exactly m relations.

$m = 3$: $R_1(A_{123}, A_{124}, A_{125}, A_{126}, A_{134}, A_{135}, A_{136}, A_{145}, A_{146}, A_{156}) \wedge$
$R_2(A_{123}, A_{124}, A_{125}, A_{126}, A_{234}, A_{235}, A_{236}, A_{245}, A_{246}, A_{256}) \wedge$
$R_3(A_{123}, A_{134}, A_{135}, A_{136}, A_{234}, A_{235}, A_{236}, A_{245}, A_{246}, A_{256}) \wedge$
$R_4(A_{124}, A_{134}, A_{145}, A_{146}, A_{234}, A_{245}, A_{246}, A_{345}, A_{346}, A_{456}) \wedge$
$R_5(A_{125}, A_{135}, A_{145}, A_{156}, A_{235}, A_{245}, A_{256}, A_{345}, A_{356}, A_{456}) \wedge$
$R_6(A_{126}, A_{136}, A_{146}, A_{156}, A_{236}, A_{246}, A_{256}, A_{346}, A_{356}, A_{456})$

Edge cover number

$\rho = m + 1$: if you pick e.g., $R_1, \ldots, R_m$, then $A_{m+1,\ldots,2m}$ is not covered.

Fractional edge cover number

$\rho^* = 2$: weight $1/m$ for every relation, every variable is in m relations.

Example

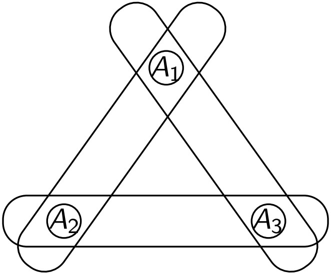
Our "favorite hypergraph": $2m$ relations, $\binom{2m}{m}$ variables, each contained in exactly m relations.

$$
\begin{aligned}
m = 3:\ & R_1(A_{123}, A_{124}, A_{125}, A_{126}, A_{134}, A_{135}, A_{136}, A_{145}, A_{146}, A_{156}) \wedge \\
& R_2(A_{123}, A_{124}, A_{125}, A_{126}, A_{234}, A_{235}, A_{236}, A_{245}, A_{246}, A_{256}) \wedge \\
& R_3(A_{123}, A_{134}, A_{135}, A_{136}, A_{234}, A_{235}, A_{236}, A_{245}, A_{246}, A_{256}) \wedge \\
& R_4(A_{124}, A_{134}, A_{145}, A_{146}, A_{234}, A_{245}, A_{246}, A_{345}, A_{346}, A_{456}) \wedge \\
& R_5(A_{125}, A_{135}, A_{145}, A_{156}, A_{235}, A_{245}, A_{256}, A_{345}, A_{356}, A_{456}) \wedge \\
& R_6(A_{126}, A_{136}, A_{146}, A_{156}, A_{236}, A_{246}, A_{256}, A_{346}, A_{356}, A_{456})
\end{aligned}
$$

Join plans

- There is a point where we have joined roughly $m/2$ relations, say, $R_1 \wedge \ldots \wedge R_{m/2}$.
- This hypergraph has an independent set of size $m/2$: variables $A_{i,m+1,\ldots,2m}$ are independent for $1 \le i \le m/2$.
- One can use this to make sure that there are $N^{m/2}$ solutions.

Join-Project plans are suboptimal



$$R = ([N/2] \times [1]) \cup ([1] \times [N/2])$$

Join-Project plan first joins two relations:

$$R(A_1, A_2) \bowtie R(A_2, A_3) = ([N/2] \times 1 \times [N/2]) \cup (1 \cup \times [N/2] \cup 1)$$

Has size $\Omega(N^2)$ — but the upper bound is $N^{3/2}$.

Optimal join algorithms

We can get rid of the $+1$ in the exponent, but these are not Join-Project algorithms.

- Ngo, Porat, Ré, and Rutra [PODS 2012]
- Veldhuizen [ICDT 2014]
- Ngo and Rudra [Sigmod Record 13]

Back to Boolean Conjunctive Queries

We have seen that treewidth of the primal graph is not a good measure of the complexity of BCQ with unbounded arities.

Tree decomposition $+$ Size bounds $=$?

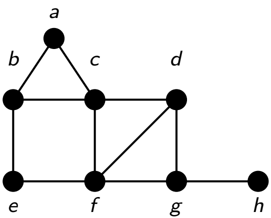
Treewidth — a measure of “tree-likeness”

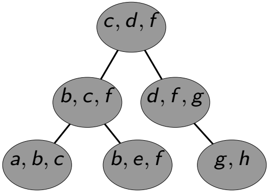
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. For any hyperedge e, there is a bag containing e.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.





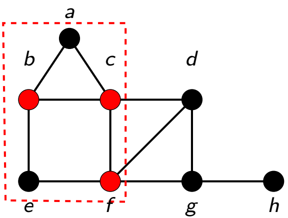
Treewidth — a measure of "tree-likeness"

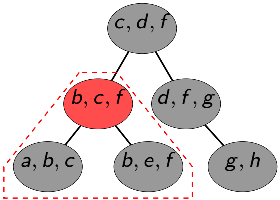
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. For any hyperedge e, there is a bag containing e.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.





A subtree communicates with the outside world only via the root of the subtree.

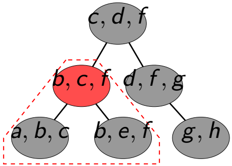
Boolean Conjunctive Queries and tree decompositions

Theorem

Given a tree decomposition of width w, a Boolean Conjunctive Query where every variable allows at most N different values can can be solved in time $N^{w+1} \cdot |Q|^{O(1)}$.

B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.

For every node x and tuple $t \in Q_{|B_x}$, we compute the Boolean value $E[x,t]$, which is true if and only if t can be extended to a tuple of $Q_{|V_x}$.



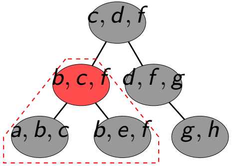
Claim:
We can determine $E[x,t]$ if all the values are known for the children of x.

Boolean Conjunctive Queries and tree decompositions

Theorem

Given a tree decomposition of width w, a Boolean Conjunctive Query where every variable allows at most N different values can can be solved in time $N^{w+1} \cdot |Q|^{O(1)}$.

B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.



For every node x and tuple $t \in Q_{|B_x}$, we compute the Boolean value $E[x, t]$, which is true if and only if t can be extended to a tuple of $Q_{|V_x}$.

Running time:
Dominating factor is the size of $Q_{|B_x}$, which can be bounded by $N^{|B_x|} \leq N^{w+1}$.

Fractional hypertree width

Fractional hypertree width: every bag has fractional edge cover number at most k.

Theorem [Grohe and M. 2006]

Given a fractional hypertree decomposition of width k, a Boolean Conjunctive Query where every variable allows at most N different values can can be solved in time $N^k \cdot |Q|^{O(1)}$.

Fractional hypertree width

Fractional hypertree width: every bag has fractional edge cover number at most k.

Theorem [Grohe and M. 2006]

Given a fractional hypertree decomposition of width k, a Boolean Conjunctive Query where every variable allows at most N different values can can be solved in time $N^k \cdot |Q|^{O(1)}$.

Generalized hypertree width: every bag has edge cover number at most k.

Hypertree width: same as generalized hypertree width, with an additional "special condition."

Acyclic hypergraphs: hypetree width $=$ generalized hypertree width $= 1$.

Finding decompositions

- If we want fixed-parameter tractability, then we can find an optimal decomposition in time $f(H)$.
- For polynomial-time algorithms, we need to find good decompositions in polynomial time.

Finding decompositions

- If we want fixed-parameter tractability, then we can find an optimal decomposition in time $f(H)$.
- For polynomial-time algorithms, we need to find good decompositions in polynomial time.

Treewidth

- optimal decomposition in time n^k [Robertson and Seymour].
- optimal decomposition in time $2^{O(k^3)} \cdot n$ [Bodlaender 1996].
- 5-approximate decomposition in time $2^{O(k)} \cdot n$ [Bodlaender et al. 2013].
- $O(\sqrt{\log k})$-approximation in polynomial time [Feige, Hajiaghayi, Lee 2008].

Finding decompositions

- If we want fixed-parameter tractability, then we can find an optimal decomposition in time $f(H)$.
- For polynomial-time algorithms, we need to find good decompositions in polynomial time.

Hypertree width

- optimal decomposition in time n^k [Gottlob, Leone, and Scarcello 2002]
- W[1]-hard $\Rightarrow$ no FPT algorithm.

Finding decompositions

- If we want fixed-parameter tractability, then we can find an optimal decomposition in time $f(H)$.
- For polynomial-time algorithms, we need to find good decompositions in polynomial time.

Generalized hypertree width

- NP-hard even for $k \geq 3$ [Gottlob, Miklós, Schwentick PODS 2007] and for $w = 2$ [Fischl, Gottlob, and Pichler 2016]
- But $ghw \leq hw \leq 3 \cdot ghw \Rightarrow$ Hypertree width gives a 3-approximation!

Finding decompositions

- If we want fixed-parameter tractability, then we can find an optimal decomposition in time $f(H)$.
- For polynomial-time algorithms, we need to find good decompositions in polynomial time.

Fractional hypertree width

- For every $k \geq 1$, there is a polynomial-time algorithm computing a decomposition of width $O(k^3)$ [M. 2009].

Theorem

If class $\mathcal{H}$ has bounded fractional hypertree width, then $\mathrm{BCQ}(\mathcal{H})$ can be solved in polynomial time.

- NP-hard for every $k \geq 2$ [Fischl, Gottlob, and Pichler 2016]

Better decompositions?

Fractional hypertree decomposition is the **best possible** tree decomposition in a formal sense.

Observation: If a tree decomposition guarantees that the projection to every bag has at most N^w solutions, then the decomposition has fractional hypertree width at most w.

(If a bag has fractional edge cover number ρ^*, we can construct an instance where it has N^{ρ^*} solutions.)

Better decompositions?

Fractional hypertree decomposition is the **best possible** tree decomposition in a formal sense.

How can we move beyond fractional hypertree decompositions?

- **Idea 1:** Look at the database, and choose a decomposition based on that (not only on the query).
- **Idea 2:** Branch and partition the solution space (e.g., light-heavy) and choose different decompositions.

Submodular width

Theorem [M. 2010]

Let $\mathcal{H}$ be a computable class of hypergraphs. Assuming the Exponential-Time Hypothesis, the following are equivalent:

- BCQ($\mathcal{H}$) is fixed-parameter tractable (solvable in time $f(Q) \cdot N^{O(1)}$).
- $\mathcal{H}$ has bounded submodular width.

Definition: H has submodular width $\leq w$ if for any function $f : 2^{V(H)} \to \mathbb{R}^+$ that is

- **monotone** ($f(X) \geq f(Y)$ for any $X \supset Y$),
- **submodular** ($f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$), and
- **edge dominated** ($f(e) \leq 1$ for any edge $e \in E(H)$)

there is a tree decomposition of H with $f(B) \leq w$ for every bag B.

Submodular width

Definition: H has submodular width $\leq w$ if for any function $f : 2^{V(H)} \to \mathbb{R}^+$ that is

- **monotone** ($f(X) \geq f(Y)$ for any $X \supset Y$),
- **submodular** ($f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$), and
- **edge dominated** ($f(e) \leq 1$ for any edge $e \in E(H)$)

there is a tree decomposition of H with $f(B) \leq w$ for every bag B.

Intuitive algorithmic idea: we imagine

$$f(X) \approx \frac{\log \#\text{ solutions in } Q_{|X}}{\log N}$$

Then there is a decomposition where $f(B) \leq w$ for every bag, so $|Q_{|B}| \leq N^w$.

Conclusions

Messages

- Treelike decompositions can make the problem easy.
- You may want to look at the data and choose a decomposition based on that.
- You may want to branch and choose different decompositions in the different branches.

Topics not covered: counting, enumeration, quantification, functional dependencies, parallel algorithms ...