The Complexity of Relational Queries: A Personal Perspective

Moshe Y. Vardi

Rice University

http://www.cs.rice.edu/∼vardi

Relational Query Theory in 1980

- Codd, 1972: FO=RA
- Chandra&Merlin, 1977: basic theory of conjunctive queries
- Aho&Ullman, 1979: RA is not expressive enough, fixpoint (recursion) needed
- Chandra&Harel, 1979: computable queries
- Chandra&Harel, 1980: structure and complexity of Relational Queries – hierarchy of relational query languages
  - FO $<$ FP $<$ SO

V., 1980: "The theory of relational queries is fully developed."

Complexity of Relational Queries

Observation: Mismatch in Chandra&Harel, 1979

- First-order queries are complete for the polynomial hierarchy ("above" NP!)

- Fixpoint queries are in PTIME.

V., 1981: "Perhaps the theory of relational queries is not fully developed".

- *Required*: complexity theory for relational queries.

Failure of Standard Complexity Theory

Standard Complexity Analysis – *Scaling Behavior*:

- Focus on decision (yes/no) problems to eliminate dependence on output size.

- Measure how run time/memory usage grows as function of input size

Database Context:

- Focus on Boolean (yes/no) quries to eliminate dependence on output size.

- Input size: database size plus query size.

Difficulty:

- Typical input size is $10^9 + 100$

- Which size is more challenging? $2 \cdot 10^9 + 100$ or $10^9 + 200$?

Intuition: Database size and query size play very different roles! This is not reflected in standard complexity theory.

Relational Complexity Theory – 1982

Basic Principle: Separate the influences of data and query on complexity

- *Influence of Query*: Fix data
- *Influence of Data*: Fix query

Real-Life Motivation:

- *Census Data Analysis*: Data fixed for 10 years, multiple queries
- *Technical Trading*: price-arbitrage fixed query, data changes momentarily.

Relational Complexity Theory –1982

A Tale of Two Complexities:

- *Query Complexity* of L: Fix $\mathbf{B}$

$$\{Q \in L : Q(\mathbf{B}) \text{ is nonempty}\}$$

- *Data Complexity* of L: Fix $Q \in L$

$$\{\mathbf{B} : Q(\mathbf{B}) \text{ is nonempty}\}$$

Observation:

- Data complexity is insensitive to syntax of queries, as queries are fixed.

- Query complexity is highly sensitive to syntax of queries (e.g., $R \times R \times R \times R \times R \times R \times R$ vs. R^{111}).

Conclusion: Change "Query Complexity" to "Expression Complexity".

Data vs Expression Complexity

Basic phenomenon: exponential gap!

Query Lang.	Data Comp.	Expression Comp.
FO	LOGSPACE	PSPACE
FP	PTIME	EXPTIME
$\exists$SO	NP	NEXPTIME
PFP	PSPACE	EXPSPACE

Theory Justifies Intuition: Characteristics of queries matter much more than size of data!

Relational Complexity Theory – 1995

Question: Why is expression complexity so high?

Intuitive Answer: Large intermediate results!

• *For example*: $R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \bowtie R_5$ can be empty, even when $R_1 \bowtie R_2 \bowtie R_3$ is very large.

Question: Can we formalize this intuition?

Answer: Variable-confined queries

Example: Compare

- $\pi_{A,B,C}(R_1 \bowtie R_2)$ to
- $\pi_{A,B}(R_1) \bowtie \pi_{A,C}(R_2)$

Observations:

• Pushing projections in RA corresponds to variable re-use in FO.

• Bounding width of intermediate relations corresponds to bounding number of variables.

Variable-Confined Queries

Definition: L^k consists of formulas of logic L with at most k variables.

Example: Formula in FO^2

- $(\exists x)((\exists y)(R(x,y) \wedge (\exists x)R(y,x))$ – exists path of length 2.

Key Result: Variable-confined queries have lower expression complexity!

Query Lang.	Data Compl.	Expression Comp.	VC Expr. Comp.
FO	LOGSPACE	PSPACE	PTIME
FP	PTIME	EXPTIME	NP$\cap$ co-NP
$\exists$SO	NP	NEXPTIME	NP
PFP	PSPACE	EXPSPACE	PSPACE

Conclusion: Expnential gaps shrinks or vanishes for variable-confined queries.

Question: Find smallest k such that given query Q in is L^k.

Answer: Undecidable!

Conjunctive Queries

Conjunctive Query: First-order logic without $\forall, \vee, \neg$; written as a rule

$$Q(X_1, \ldots, X_n) \; :- \; R(X_3, Y_2, X_4), \ldots, S(X_2, Y_3)$$

Significance: most common SQL queries (*Select-Project-Join*)

Example:

$$GrandParent(X, Y) \; :- Parent(X, Z), Parent(Z, Y)$$

Equivalently:

$$(\exists Z)(Parent(X, Z) \wedge Parent(Z, Y))$$

Complexity of Conjunctive Queries

Chandra&Merlin, 1977: Expression complexity of CQ is NP-complete.

Precise Complexity Analysis: $||B|^{||Q||}$.

Yannakakis, 1995: $||B||^{||Q||}$ is much worse than $c^{||Q||} \cdot ||B|^d$ for fixed c, d, which is *fixed-parameter tractable* (FPT) – **parameterized complexity analysis**

Papadimitriou&Yannakakis, 1997: CQ evaluation is W[1]-complete – unlikely to be FPT.

Variable-Confined CQ

V., 1995: CQ^k – CQ using at most k variables.

• If Q is in CQ^k then query can be evaluated over database B in time $||Q|| \cdots ||B||^d$ - FPT!

Example: Contrast

$$(\exists x, y, z)((R(x, y) \wedge R(y, z)$$

and

$$(\exists x)((\exists y)(R(x, y) \wedge (\exists x)R(y, x))$$

Conclusion: The critical parameter is number of variables, not size of query!

Question: Characterize smallest k such that a given conjunctive query Q is in CQ^k.

CQs and Treewidth

Treewidth: basic concept in graph theory

- A tree has treewidth 1.
- A cycle has treewith 2.
- An $m \times m$ grid has treewidth m.

Query Graph: graph of a conjunctive query

- *Nodes*: variables
- *Edges*: connect nodes that co-occur in an atom

Definition: $treewidth(Q)$ is $treewidth(graph(Q))$.

Kolaitis&V., 1998: Q is in CQ^k iff $treewidth(Q) < k$.

Corollary: Bounded treewidth CQs are fixed-parameter tractable.

Grohe, Schwentick&Segoufin, 2000: Optimal!

Gottlob, Leone&Scarcelloo, 1999: Not optimal!

Theory and Practice

Question: Can theory be used to optimize CQs?

Partial Answer: Not easily!

- Finding treewidth of a graph is NP-hard!

CQ Evaluation

Hard problem for databases: evaluation of large conjunctive queries

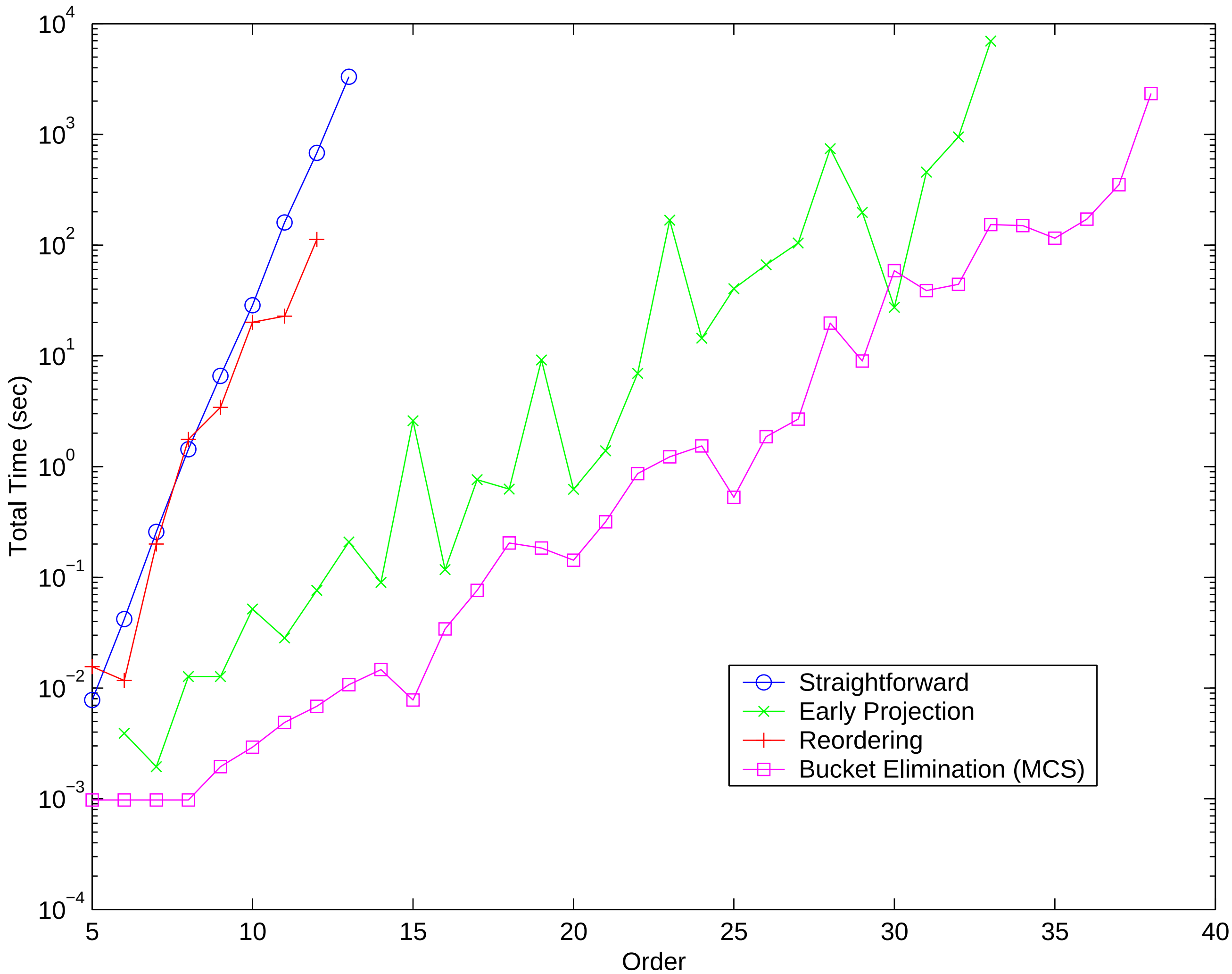
- Corresponds to evaluating a sequence of joins and projections.
- Many possible evaluation orders possible
- Query optimizer has to search a very large space

An Alternative Approach: (McMahan&V., 2004)

- Consider the problem as a constraint-satisfaction problem (CSP).
- Apply CSP heuristics for constraint propagation.
- Focus on minizing the size of intermediate relations.
- Essentialy, minimize number of variables heuristically.

Question: Does it work?

Experimental Results



Answer: Exponential improvement for large CQs.

In Conclusion

Role of Theory:

- Clarify conceptual framework
- Suggest experimental possibilties

Paradigmatic Example: Relational model

Thank You!