# Logical Algorithmics: From Theory to Practice

**Moshe Y. Vardi**
Rice University
vardi@cs.rice.edu

# First-Order Logic on Graphs

- "node $x$ has at least two distinct neighbors"

$$(\exists y)(\exists z)(\neg(y = z) \wedge E(x, y) \wedge E(x, z))$$

*Concept*: $x$ is *free* in the above formula, which expresses a property of nodes. Truth definition requires a *binding* $\alpha$. $G \models_\alpha \varphi(x)$

- "each node has at least two distinct neighbors"

$$(\forall x)(\exists y)(\exists z)(\neg(y = z) \wedge E(x, y) \wedge E(x, z))$$

*Concept*: The above is a *sentence*, that is, a formula with no free variables; it expresses a property of graphs. $G \models \varphi$

# From Model Theory to Relational Databases

- A sentence $\psi$ is either true or false on a given graph $\mathbf{G}$. In particular, sentences specify classes of graphs: $\mathsf{models}(\psi) = \{\mathbf{G} : \mathbf{G} \models \psi\}$

  **Model Theory**: Logic provides a metatheory for mathematical modeling.

- E.F. Codd, 1970: Formulas $\varphi(x_1, \ldots, x_k)$ define *queries*:
  $\varphi(\mathbf{G}) = \{\langle \alpha(x_1), \ldots, \alpha(x_k)\rangle : \mathbf{G} \models_\alpha \varphi(x_1, \ldots, x_k)\}$
  - Example: $(\exists y)(\exists z)(\neg(y = z) \wedge E(x, y) \wedge E(x, z))$ – "List nodes that have at least two distinct neighbors"
  - Sentences express Boolean (yes/no) queries.

# Relational Databases

**Codd's Two Fundamental Ideas** [Turing Award, 1981]:

- *Tables are finite relations*: a *row* in a table is just a *tuple* in a relation; order of rows/tuples does not matter!

- *Formulas are queries*: they specified the *what* rather then the *how* – declarative programming!

**Novel insight**: Intimate connection between logic and algorithms.

# Algorithmic Problems in First-Order Logic

**Satisfiability Problem**: Given a first-order formula $\psi$, *is there* a (finite) graph $\mathbf{G}$ and binding $\alpha$, such that $\mathbf{G} \models_\alpha \psi$?

**Truth-Evaluation Problem (Model Checking)**: Given a first-order formula $\varphi(x_1, \ldots, x_k)$, a *finite* graph $\mathbf{G}$, and a binding $\alpha$, does $\mathbf{G} \models_\alpha \varphi(x_1, \ldots, x_k)$?

**Facts**:

- Satisfiability is *undecidable* [Church-Turing, 1936, Trakhtenbrot, 1950]
- Truth evaluation, which is query evaluation, is *decidable*, because finite.

# Logical Algorithmics

**A standard algorithmic problem**: Does an input structure, e.g., *an undirected graph* $G$, have a *certain* property $P$, e.g., *perfect matching*.

**Logical Perspective**: Specify property $P$ via a sentence $\varphi$ in a logic $L$ – decidable on finite structures via a *uniform "meta-algorithm"* –

$$QE_L(\varphi, G) = 1 \text{ iff } G \models \varphi$$

# Logic and Complexity Theory

**Insight**: Many interesting graph-theoretic properties cannot be expressed in FoL, e.g., 3-Colorability: existential quantification on coloring is required.

- Existential Second-Order Logic (ESO) – $(\exists R_1), \ldots, (\exists R_k)\varphi(R_1, \ldots, R_k)$, where $\varphi$ is first -order.

Fagin's Theorem, 1974: ESO expresses *precisely* NPTIME – every property in NPTIME can be described by a sentence in ESO, and vice versa.

- No explicit notion of computation
- No explicit notion of polynomial
- *Descriptive Complexity Theory*

# Descriptive-Complexity Theory

**Descriptive complexity**: a branch of computational complexity theory that characterizes complexity classes by the type of logic needed to express the languages in them.

**Consequence of Fagin's Theorem**: PH is precisely the class of languages expressible in 2nd-order logic.

**Deep Insight**: There is a deep connection beween the computational complexity of problems and their decriptive complexity [Immerman, 1999].

# The Query-Evaluation Meta-Algorithm

$QE_L(\varphi, G) = 1$ iff $G \models \varphi$

**Question** [V., 1982]: What is the computational complexity of $QE_L$?

Observations:

- It surely depends on the logic $L$, e.g., FoL vs ESO.
- $QE_L$ has two input variables: $\varphi$ and $G$.
  - Fagin's Theorem tells us that for each *fixed* $\varphi \in$ESO, the complexity of $QE(\varphi, .)$ is in NPTIME.
  - What about *variable* $\varphi$?

# What Logics to Consider?

**Surely**: FoL and ESO. Anything else?

- Aho&Ullman, 1979: FoL is not expressive enough – cannot express reachability, induction needed
- Chandra&Harel, 1980: FP=FO(Ind)
- FoL $<$ FP $<$ ESO

What logics to consider? FoL, FP, ESO

# Multivariate Complexity Theory

**Standard Complexity Analysis** – *Scaling Behavior*:

- measures how run time/memory usage *scales* as function of *input size*.

**QE Context**:

- *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.

**Later development**: Parameterized Complexity Theory [Downey&Fellows]

# Complexity Theory for Logical Algorithmics, I

**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.

# Complexity Theory for Logical Algorithmic, II

**A Tale of Three Complexities**:

- *Expression Complexity* of $L$: Fix $\mathbf{B}$ – $\{\varphi \in L : QE(\varphi, \mathbf{B}) = 1\}$
- *Data Complexity* of $L$: Fix $\varphi \in L$– $\{\mathbf{B} : QE(\varphi, \mathbf{B}) = 1\}$
- *Combined Complexity* of $L$: $\{\langle \varphi, \mathbf{B} \rangle : \varphi \in L$ and $QE(\varphi, \mathbf{B}) = 1\}$

# Data vs Expression Complexity

**Basic phenomenon**: exponential gap! [V., 1982]

| Logic | Data Comp. | Expression Comp. |
|---|---|---|
| FO | LOGSPACE | PSPACE-c |
| FP | PTIME-c | EXPTIME-c |
| ESO | NPTIME-c | NEXPTIME-c |

**Theory justifies intuition**:

- Characteristics of queries matter much more than size of data!
- Query optimization is important!
- Distinction between data and queries has *very broad* applicability.
- *Multivariate Complexity Theory* $\rightsquigarrow$ *Parametrized Complexity Theory* [Abrahamson et al, 1989]

# Back to Descriptive Complexity Theory

| Logic | Data Comp. | Query Comp. |
|---|---|---|
| FP | PTIME | EXPTIME |
| ESO | NPTIME | NEXPTIME |

**Fagin's Theorem**: ESO $\equiv$ NPTIME

**Question**: What about FP and PTIME?

**Immerman-V.'s Theorem, 1982**: FP $\equiv$ PTIME over ordered structures!

- **Major Open Question** [Chandra&Harel, Gurevich]: What about PTIME on *unordered* structures?

# Constraint-Satisfaction Problem (CSP)

**Input:** $(V, D, C)$:

- A finite set $V$ of *variables*
- A finite set $D$ of *values*
- A finite set $C$ of *constraints* restricting the values that tuples of variables can take.

**Constraint:** $(t, R)$

- $t$: a tuple of variables over $V$
- $R$: a relation over $D$ of arity $|t|$

**Solution:** $h : V \to D$ s.t. $h(t) \in R$ for all $(t, R) \in C$

**Question:** Does $(V, D, C)$ have a solution? I.e., is there an assignment of values to the variables such that all constraints are satisfied?

# Constraint Satisfaction

**Applications**:

- belief maintenance
- machine vision
- natural language processing
- planning and scheduling
- temporal reasoning
- type reconstruction
- bioinformatics
- ...

# Homomorphisms

**Homomorphism**: Let $\mathbf{A} = (A, R_1^{\mathbf{A}}, \ldots, R_m^{\mathbf{A}})$ and $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$ be two relational structures over same vocabulary.

$h : A \to B$ is a *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ if for $i = 1, \ldots, m$ and every tuple $(a_1, \ldots, a_n) \in A^n$,

$$R_i^{\mathbf{A}}(a_1, \ldots, a_n) \implies R_i^{\mathbf{B}}(h(a_1), \ldots, h(a_n)).$$

**The Homomorphism Problem:** Given relational structures $\mathbf{A}$ and $\mathbf{B}$, is there a homomorphism $h : \mathbf{A} \to \mathbf{B}$?

**Example:** An undirected graph $\mathbf{A} = (V, E)$ is 3-colorable $\iff$ there is a homomorphism $h : \mathbf{A} \to K_3$, where $K_3$ is the *3-clique*.

# Homomorphism Problems

**Examples:**

- $k$-Clique: $K_k \stackrel{h}{\to} (V, E)$?
- Hamiltonian Cycle: $(V, C_{|V|}, \neq) \stackrel{h}{\to} (V, E, \neq)$?
- Subgraph Isomorphism: $(V, E, \overline{E}) \stackrel{h}{\to} (V', E', \overline{E'})$?
- s-t Connectivity: $(V, E, \{\langle s, t\rangle\}) \stackrel{h}{\not\to} (\{0, 1\}, =, \neq)$?

Feder&V., 1993: CSP=Homomorphism Problem

- Trivial reductions in both directions

# Uniform CSP

**Uniform CSP:** $\{(\mathbf{A}, \mathbf{B}) : \exists \text{ homomorphism } h : \mathbf{A} \to \mathbf{B}\}$

*Complexity of Uniform CSP*: NP-complete [Levin, 1973]

**Multivariate-Complexity View**: $\mathbf{A}$ – query, $\mathbf{B}$ – data.

**Note**: Uniform problem corresponds to combined complexity.

# Non-Uniform CSP, I

**Fixed-Source CSP:** Fix query $\mathbf{A}$

$$\mathrm{CSP}(\mathbf{A}, .) = \{\mathbf{B} : \exists \text{ homomorphism } h : \mathbf{A} \to \mathbf{B}\}$$

*Data Complexity of Non-Uniform CSP*: LOGSPACE

# Non-Uniform CSP, II

**Fixed-Target CSP:** Fix data $\mathbf{B}$

$$\mathrm{CSP}(., \mathbf{B}) = \{\mathbf{A} : \exists \text{ homomorphism } h : \mathbf{A} \to \mathbf{B}\}$$

*Expression Complexity of Non-Uniform CSP*: Depends on target $\mathbf{B}$

- $\mathrm{CSP}(., K_2)$ is in PTIME (2-Colorability)
- $\mathrm{CSP}(., K_3)$ is NP-complete (3-Colorability)

# Complexity of Non-Uniform CSP

**Research Program:** Identify the tractable cases of fixed-target CSP

- **Goal**: Understand complexity of a large class of problems in NP.

**Dichotomy Conjecture:** (Feder&V., 1993): For a structure $\mathbf{B}$,

- either $\mathrm{CSP}(., \mathbf{B})$ is in PTIME
- or $\mathrm{CSP}(., \mathbf{B})$ is NP-complete.

**Recall**: $P \neq NP \Rightarrow NP - NPC - P \neq \emptyset$ (Ladner, 1975)

# The Product Operation

**Definition**: Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. The *product* of these graphs is the graph $G_1 \times G_2 = (V_1 \times V_2, E_1 \times E_2)$, where $(\langle u, u' \rangle, \langle v, v' \rangle) \in E_1 \times E_2$ iff $(u, v) \in E_1$ and $(u', v') \in E_2$.

**Note**: This definition can be extended to pairs of relational structures.

# Polymorphisms

**Definition**: Let $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$ be a relational structure. A $k$-ary *polymorphism* is a homomorphism $f : \mathbf{B}^k \to \mathbf{B}$ (*closure condition*).

Feder&V., 93: Study $Poly(\mathbf{B})$ – set of polymorphisms of $\mathbf{B}$

**Definition**: A $k$-ary *near-unanimity operation* is a $k$-ary function $f$ such that $f(x_1, x_2, \ldots, x_k) = a$ whenever at least $k-1$ of the $x_i$'s equal $a$.

**Example**: 3-Majority is a near-unanimity operation.

**Theorem**: [Feder&V., 1993]
If $Poly(\mathbf{B})$ contains a near-unanimity function, then $CSP(., \mathbf{B})$ can be solved in PTIME.

# The Polymorphism Program

**Observation**: Polymorphisms are a major topic of study in *Universal Algebra* – a branch of *Model Theory*.

**Theorem**: [Jeavons et al., 1995-8]
$Poly(\mathbf{B_1}) = Poly(\mathbf{B_2}) \Rightarrow CSP(., \mathbf{B_1}) \equiv_p CSP(., \mathbf{B_2})$.

**Conclusion**: $Poly(\mathbf{B})$ characterizes the complexity of $CSP(., \mathbf{B})$.

# FOCS 2017: Conjecture Resolved!

- Andrei A. Bulatov: *A Dichotomy Theorem for Nonuniform CSPs*
- Dmitriy Zhuk: *The Proof of CSP Dichotomy Conjecture*

**Techniques**: Universal Algebra

**Also**: It is decidable, for a given $B$, whether $CSP(., \mathbf{B})$ is in PTIME or NP-complete.

# Semi-Uniform CSP

**Recall**: **Fixed-Source CSP** – $\mathrm{CSP}(\mathbf{A}, .)$ is in LOGSPACE.

**Question**: What about $\mathrm{CSP}(\mathcal{A}, .)$, for a class $\mathcal{A}$ of structures?

**Definition**: The *core* of a structure is its (unique) minimal homomorphic substructure. $\mathcal{C}_k$ consists of structures with cores of treewidth at most $k$.

**Theorem** [Dalmau&Kolaitis&V., 2002]: $\mathrm{CSP}(\mathcal{C}_k, .)$ is tractable.

**Theorem**: [Grohe, 2005]: Assume $FPT \neq W[1]$. Then $\mathrm{CSP}(\mathcal{A}, .)$ is tractable iff $\mathcal{A} \subseteq \mathcal{C}_k$.

# Relational Complexity Theory – 1995

**Question**: Why is expression complexity so high? How do databases evaluate queries in practice?

**Intuitive Answer**: Large intermediate results!

- How much is $1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 8 \times 9 \times 0$?

- *Example*: $R_1 \ltimes R_2 \ltimes R_3 \ltimes R_4 \ltimes R_5$ can be empty, even when $R_1 \ltimes R_2 \ltimes R_3$ is very large.

**Question**: Can we formalize this intuition?

**Answer**: *Variable-confined (VC) queries*

# Projection Pushing

**Example**: Compare two equi-joins of *ternary* relations

- $\pi_{1,6}(R_1 \ltimes_{3=1} R_2)$ – 6-ary intermediate relation
- $\pi_{1,4}(\pi_{1,3}(R_1) \ltimes_{2=1} \pi_{1,3}(R_2))$ – 4-ary intermediate relation

**Observations**:

- *Projection pushing* 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**: "There exists a path of length 2"

- FO$^3$: $(\exists x)((\exists y)(\exists z)(R(x,y) \wedge R(y,z))$
- FO$^2$: $(\exists x)((\exists y)(R(x,y) \wedge (\exists x)R(y,x))$

**Key Result** [V.'95]: VC queries have lower expression complexity!

| Query Lang. | Data Compl. | Expression Comp. | VC Expr. Comp. |
|---|---|---|---|
| FO | LOGSPACE | PSPACE | PTIME |
| FP | PTIME | EXPTIME | PTIME |
| ∃SO | NP | NEXPTIME | NP |

# Variable-Confined Queries Are Easier

**Conclusion**: Exponential gap between data complexity and expression complexity *shrinks or vanishes* for variable-confined queries.

**Optimization Problem**: Find smallest $k$ such that given FO query $Q$ in is $\mathsf{FO}^k$.

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

**Observation**: [Kolaitis+V., 1998] CQ evaluation=CSP.

# Complexity of Conjunctive Queries

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

**Precise Complexity Analysis**: $||B|^{||Q||}$, for evaluating query $Q$, over database $B$.

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: $\mathsf{CQ}^k$ – CQ using at most $k$ variables.

- If $Q$ is in $\mathsf{CQ}^k$, then query can be evaluated over database $B$ in time $||Q|| \cdot ||B||^k$ - FPT!

**Example**: Contrast

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

and

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

# Hardness of CQs

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

**Question**: Characterize smallest $k$ such that a given conjunctive query $Q$ is in $\mathsf{CQ}^k$.

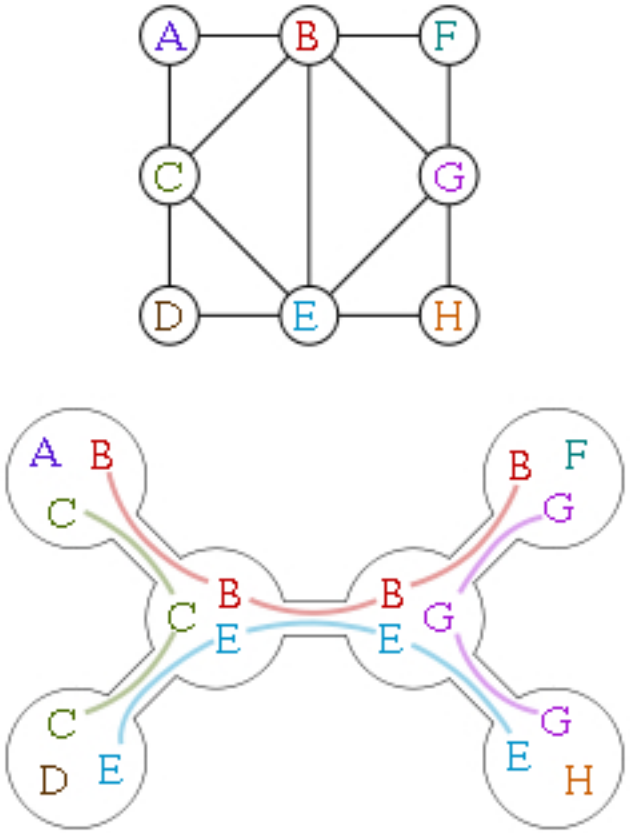



Figure 1: Tree Decomposition of Width 2

# CQs Treewidth

**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: $Q$ can be rewritten as $\mathsf{CQ}^k$ iff $treewidth(Q) < k$.

**Corollary**: Bounded treewidth CQs are fixed-parameter tractable.

# Theory and Practice

**Question**: Can this theory be used to optimize CQs?

**Partial Answer**: Not easily!

- Finding treewidth of a graph is NP-hard!

# CQ Evaluation I

**Hard problem for databases**: evaluation of large conjunctive queries

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

# CQ Evaluation II

**An Alternative Approach**: (McMahan&V., EDBT'04)

- Consider the problem as a constraint-satisfaction problem (CSP).
- Apply CSP heuristics for constraint propagation.
- Minimize the size of intermediate relations via treewdith minimization
- Essentialy, minimize number of variables, *heuristically*.

**Question**: Does it work?

**Answer**: Exponential improvement for large CQs.

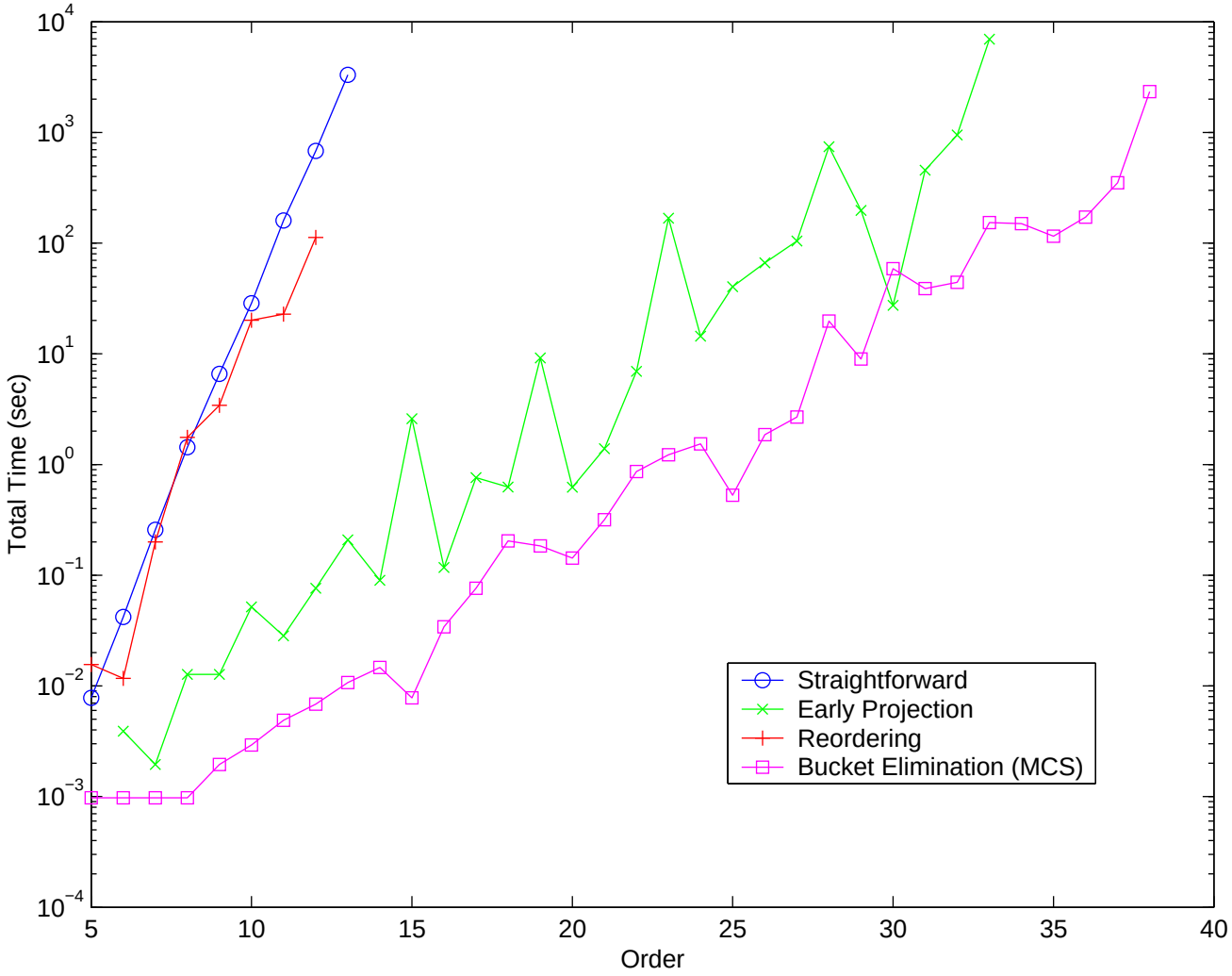



Figure 2: Experimental Results

# Back to Tree Decomposition

**Key Observation**: Computing optimal tree decompositions is NP-hard, but there has been huge progress since 2004 in computing "good" tree decomspoitions, including anytime solvers.

**Not yet done**: *Practical* CQ optimization vs tree decomposition.

- Relationship to "real" query optimization
- Hypertree decomposition [Gottlob-Leone-Scarcello, 1999, Ghionna-Granata-Greco-Scarcello, 2006]

# Boolean CSP

Domain of $B = \{0, 1\}$,

2-SAT – $\mathbf{B}$:

$x \vee y$:

| | |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

$\neg x \vee y$:

| | |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 1 | 1 |

$\neg x \vee \neg y$:

| | |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |

**3-SAT**: A 3-clause $C$ over Boolean variables $x, y, x$ corresponds to a 7-tuple relation $R_C$ over $x, y, x$.

# Boolean CSP and Conjunctive Queries

**SAT Instance**: $f = C_1 \wedge \ldots C_k$

**Fact**: $models(f) = R_{C_1} \bowtie \ldots \bowtie R_{C_k}$

**But**: $models(f)$ is exponentially large.

# Symbolic Representation: Binary Decision Diagrams

**BDDs**: efficient way to manipulate Boolean functions

- directed acyclic graph ("folded decision tree")
- internal nodes correspond to Boolean variables
- all paths lead to one of the two terminal vertices labeled by the values 0 and 1

**Properties**:

- compact representation of sets of truth assignments
- canonical representation for a given variable ordering
- easy equivalence check, polynomial Boolean operations

**Idea**: Evaluate Boolean conjunctive queries symbolically – *not competitive against modern SAT solvers* (except for special cases) [Pan+V., 2005].

# Quantitative Boolean Reasoning

- **Model counting (#SAT)**: computing number of satisfying assignments of Boolean formula
- **Complexity**: #P-complete (Valiant, 1979)
- **Numerous applications**: especially in probabilistic reasoning

**Weighted Model Counting**: Assignments are *weighted*, e.g., literal weighting:

- Each literal has a weight.
- Weight of assignment = product of literal weights
- **Task**: Compute sum of weights of satisfying assignments

# Algebraic Decision Diagrams

**ADDs**: Efficient way to manipulate *pseudo-Boolean* functions

- directed acyclic graph ("folded decision tree")
- internal nodes correspond to Boolean variables
- terminal nodes labeled with real numbers

**Properties**:

- canonical representation for a given variable ordering
- polynomial plus/times operations

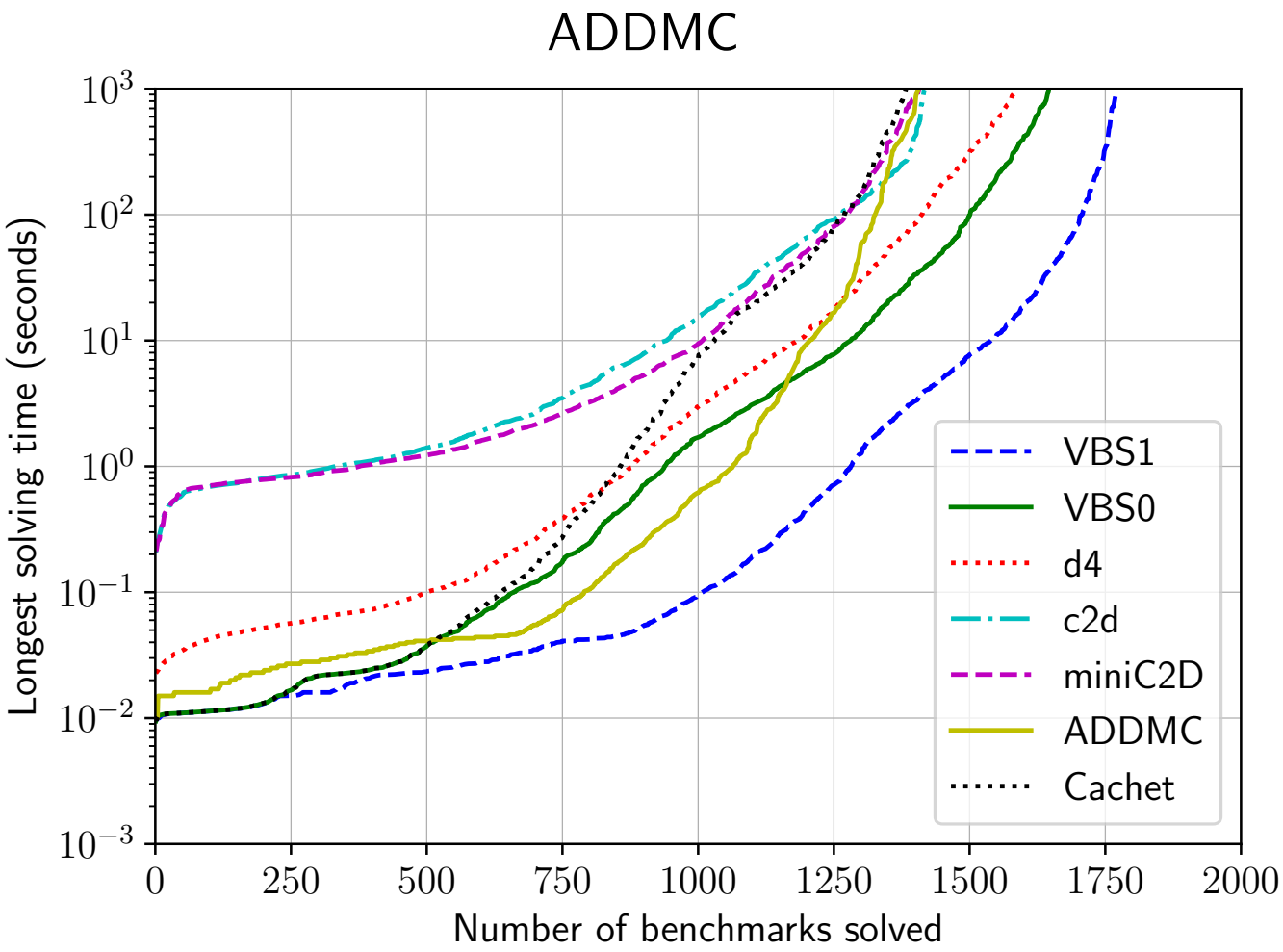
# Weighted MC with ADDs

**Basic Idea**:

- Construct BDD $B_\varphi$ for input formula $\varphi$.
- Combine with literal weights to construct ADD $A_\varphi : 2^{Prop(\varphi)} \rightarrow \mathbb{R}$
- Project all propositions using $\Sigma_p(A) = A[p \mapsto 0] + A[p \mapsto 1]$.

**Problem**: $B_\varphi$ blows up.

**ADDMC**: [Dudek, Phan, and V., 2020] – Use CSP heuristics to minimize number of intermediate variables – tied for 1st place of weighted track in 2020 Model-Counting Competition.

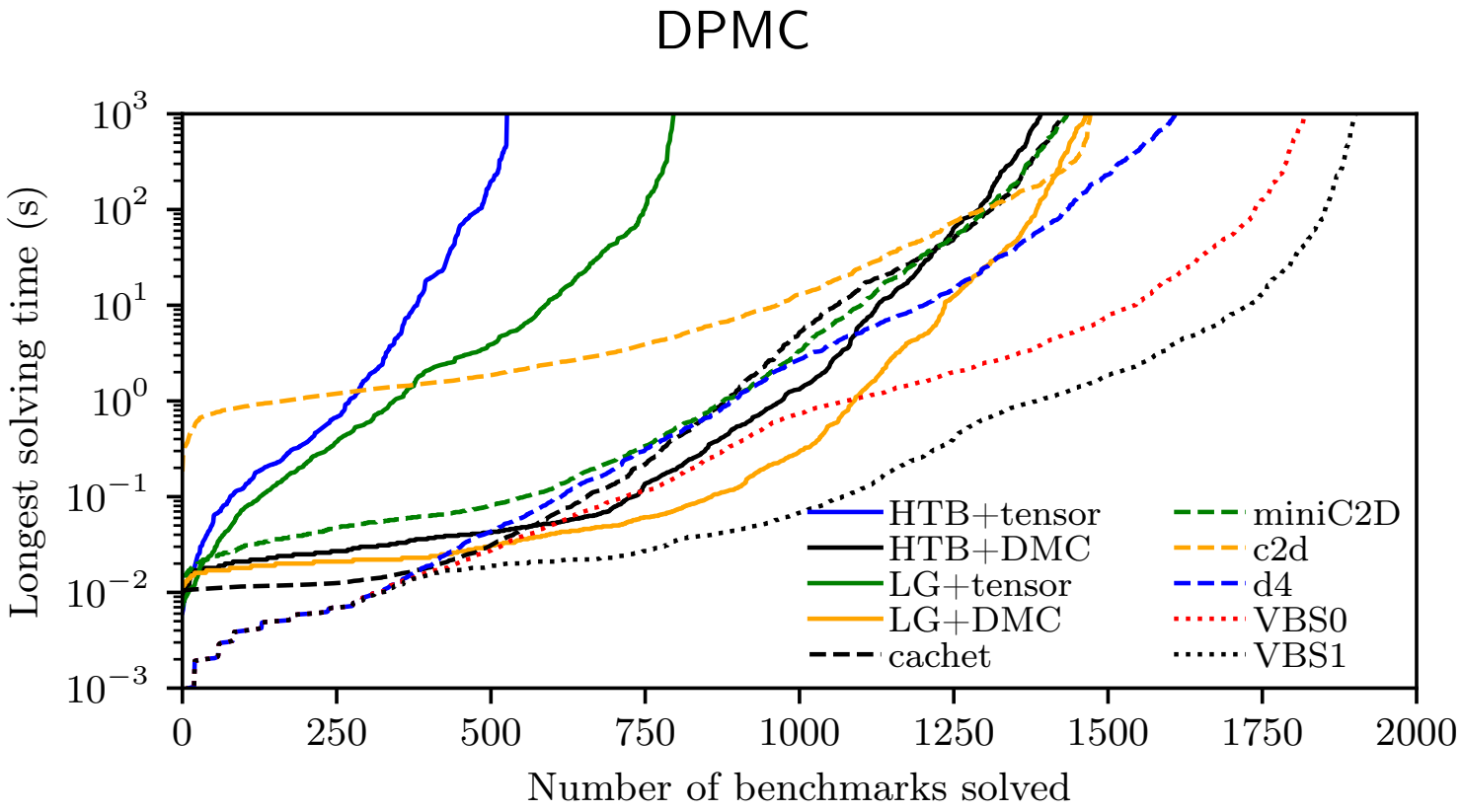

ADDMC
Longest solving time (seconds)
$10^3$
$10^2$
$10^1$
$10^0$
$10^{-1}$
$10^{-2}$
$10^{-3}$
0
250
500
750
1000
1250
1500
1750
2000
Number of benchmarks solved
VBS1
VBS0
d4
c2d
miniC2D
ADDMC
Cachet

# Back to Tree Decompositions

**Key Observation**: Computing optimal tree decompositions is NP-hard, but there has been huge progress since 2005 in computing "good" tree decomspoitions, including *anytime* solvers.

**DPMC**: Query rewriting based on tree decompositions rather than CSP heuristics [Dudek, Phan, and V., 2020].

- **Empirical observation**: DMPC beats ADDMC decisively.


DPMC
Longest solving time (s)
$10^3$
$10^2$
$10^1$
$10^0$
$10^{-1}$
$10^{-2}$
$10^{-3}$
0
250
500
750
1000
1250
1500
1750
2000
Number of benchmarks solved
HTB+tensor
HTB+DMC
LG+tensor
LG+DMC
cachet
miniC2D
c2d
d4
VBS0
VBS1

# Logical Algorithmics – In Conclusion

**Logical Algorithmics** – a logical perspective on algorithms and complexity

- Descriptive complexity theory
- Multivariate complexity theory
- Dichotomy theorems for constraint satisfaction problems
- Effective approach for quantitative Boolean reasoning