Approximate Lifted Inference with Probabilistic Databases

Wolfgang Gatterbauer, Dan Suciu

Sept 3, VLDB 2015

Carnegie Mellon Tepper SCHOOL OF BUSINESS

University of Washington Computer Science & Engineering

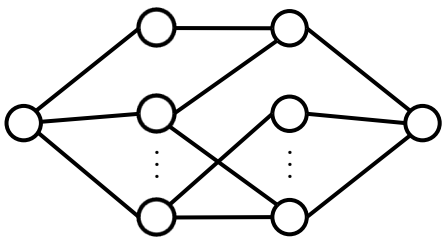
Why probabilistic relational data?

- Relational data is increasingly probabilistic Russell [CACM'15]
  - DeepDive (>7M tuples) Niu et al. [ICDM'12], Shin et al. [VLDB'15]
  - NELL machine reading (>50M tuples) Carlson et al. [AAAI'10]
  - Google Knowledge Vault (>2BN tuples) Dong et al. [KDD'14, VLDB'15]
- Data is inferred from unstructured information using statistical models
  - Learned from the web, large text corpora, ontologies, etc.
  - The learned/extracted data is relational
- Extraction rules usually specified in first-order form

$$\text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y), \text{ weight } = 3$$

Grounding

FOL → grounded graph



Inference #P hard ☹
in practice: sampling

Dichtomy results, e.g.:

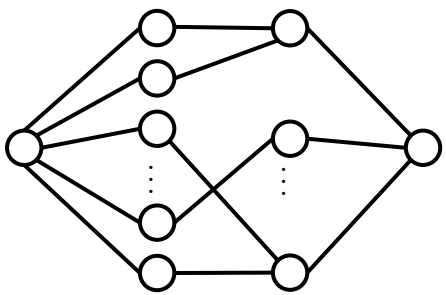
Dalvi, Suciu [VLDB'04, JACM'12]

Re, Suciu [VLDBJ'09]

Fink, Olteanu [PODS'14]

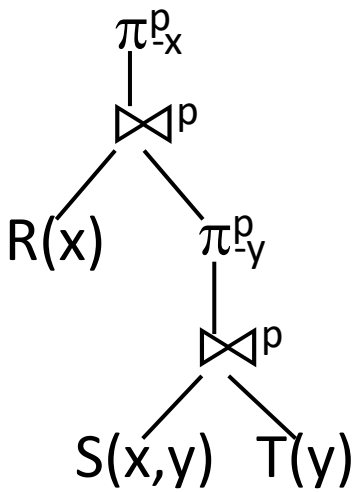
Lifted inference

Certain structural properties



liftable: inference scales well with data size

PTIME ☺



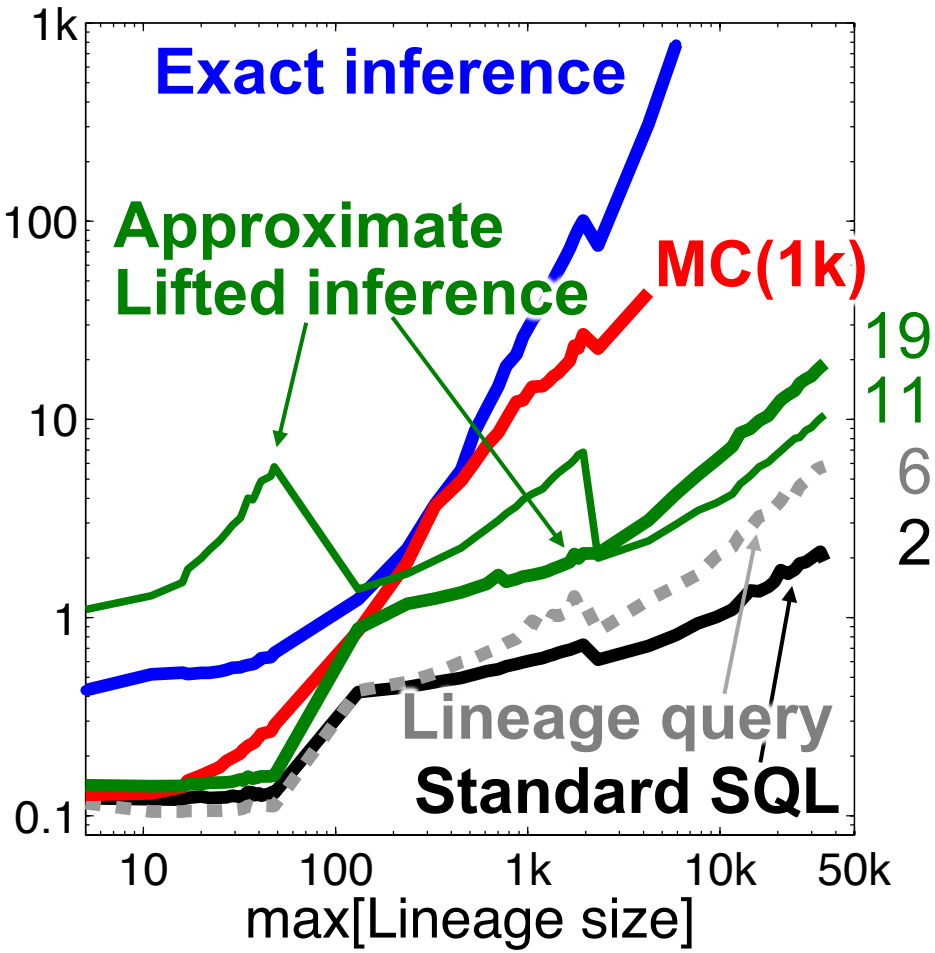
no grounding necessary!
→ super fast

Can we perform approximate inference without first grounding?

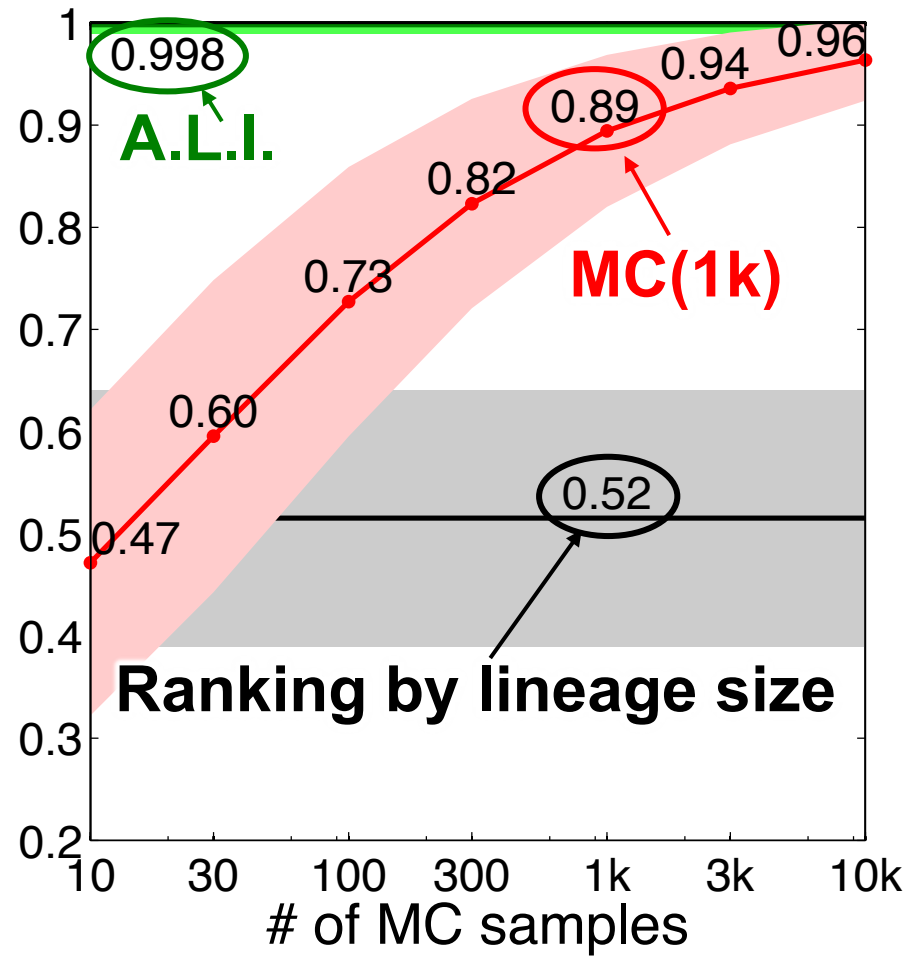
Focus: <u>self-join free conjunctive queries (sf-CQs), ranking answers</u>

Experiments: ranking query results on TPC-H

Time (sec)



Ranking quality (Average Precision)

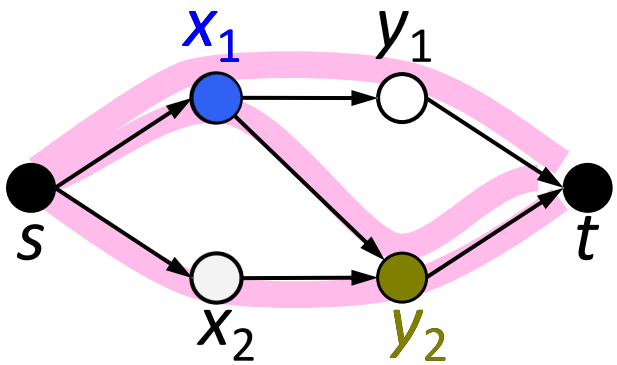


1. Theory: Upper bounds on the probability of monotone DNFs

2. Practise: Approximate lifted inference with probabilistic databases

Network Reliability

f=true iff s & t connected



$\mathbf{P}[x_i] = p_i$, $\mathbf{P}[y_j] = q_j$

Boolean Functions

f = path 1 ∨ path 2 ∨ path 3

paths 1 & 2 not independent!

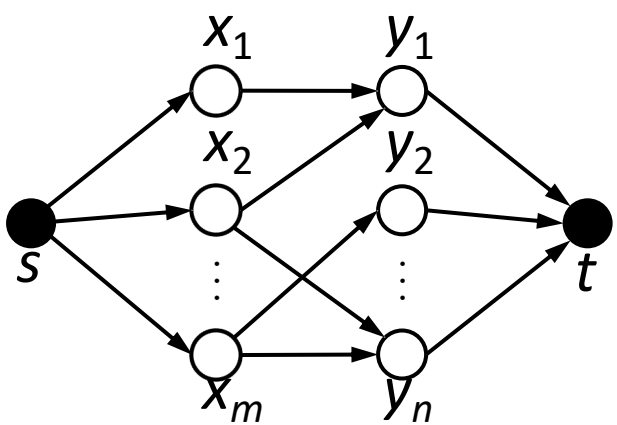
$$f = x_1 y_1 \vee x_1 y_2 \vee x_2 y_2$$

$$\mathbf{P}[f] = \mathbf{P}[x_1]\mathbf{P}[y_1 \vee y_2] + \mathbf{P}[\overline{x_1}]\mathbf{P}[x_2 y_2]$$

$$= p_1 \underbrace{\overline{\overline{q_1}\,\overline{q_2}}}_{1-(1-q_1)(1-q_2)} + \overline{p_1}\; p_2 q_2$$

"independent or"

More general:

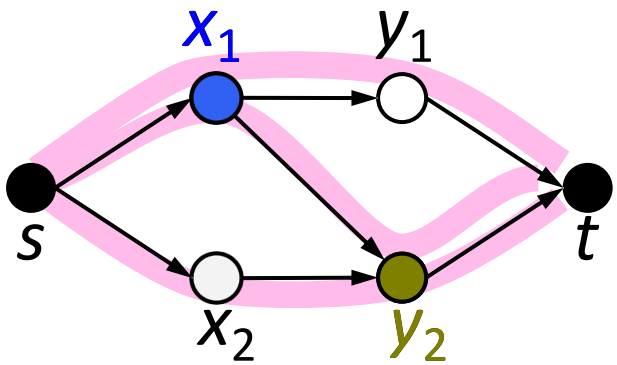


Calculating $\mathbf{P}[f]$ is #P hard ☹

Provan, Ball [SICOMP'83]

Dissociation of Monotone Boolean Functions (Intuition)

Original network

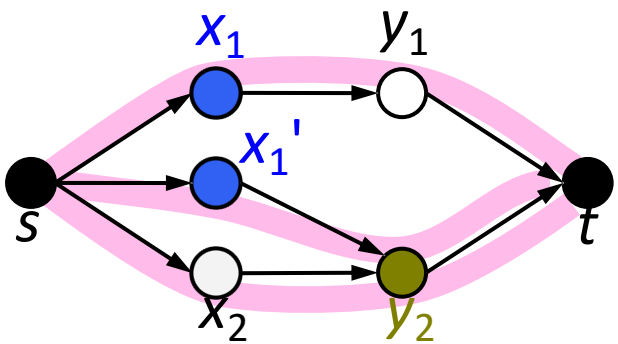


$$f = x_1 y_1 \vee x_1 y_2 \vee x_2 y_2$$

$$\mathbf{P}[f] = \mathbf{P}[x_1]\mathbf{P}[y_1 \vee y_2] + \mathbf{P}[\overline{x_1}]\mathbf{P}[x_2 y_2]$$

$$= p_1 \quad \overline{\overline{q_1}\,\overline{q_2}} \quad + \quad \overline{p_1} \quad p_2 q_2$$

"Dissociated" network



$$f' = x_1 y_1 \vee x_1' y_2 \vee x_2 y_2$$

$$= x_1 y_1 \vee (x_1' \vee x_2) y_2$$

$$\mathbf{P}[f'] = \overline{\overline{p_1 q_1}} \quad \overline{\overline{\overline{p_1'} \quad \overline{p_2}\, q_2}}$$

Read-once formula

Serial-parallel graph

Calculating $\mathbf{P}[f]$ is in PTIME ☺

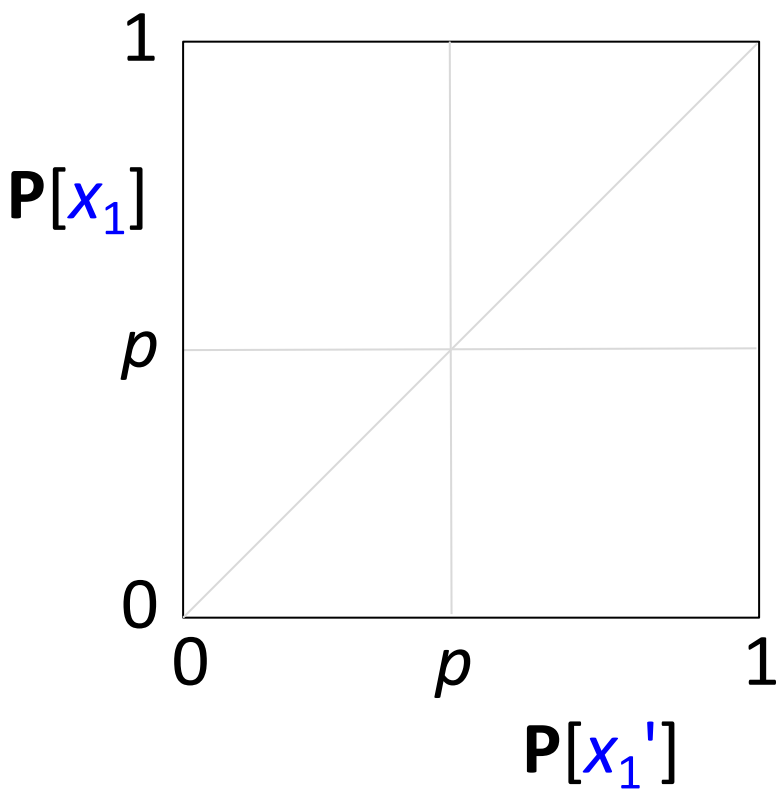
Now, how to choose p and p' ?

Gurvich [1977]

Oblivious Bounds for disjunctive Dissociations

Now, how to choose p and p' ?

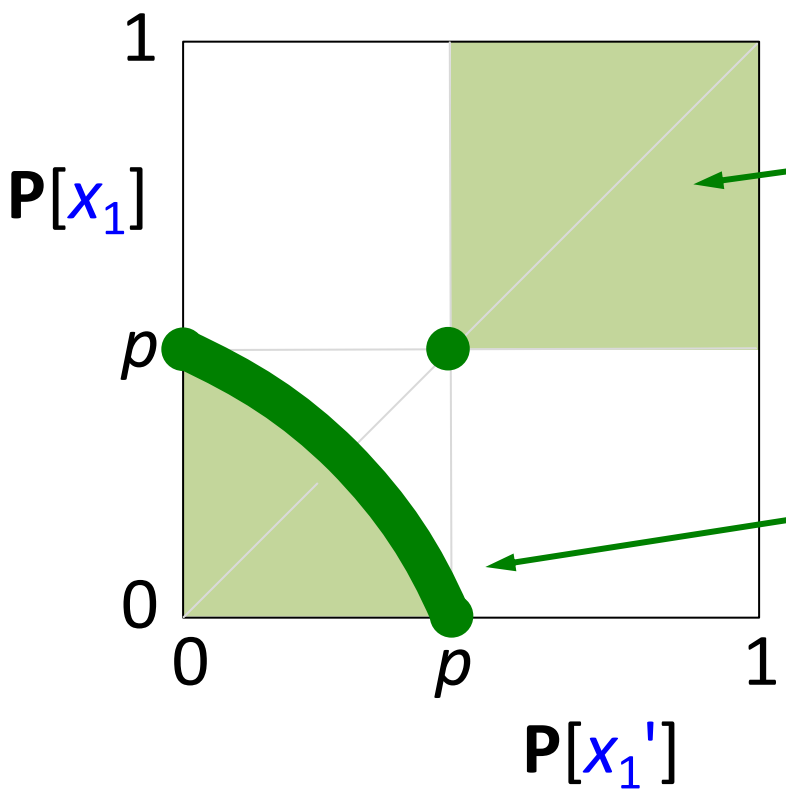
$$f' = x_1 y_1 \vee x_1' y_2 \vee x_2 y_2$$



Oblivious Bounds for disjunctive Dissociations (an algebraic framework)

Now, how to choose p and p' ?

$f' = x_1 y_1 \vee x_1' y_2 \vee x_2 y_2$



Oblivious upper bound

$p_1 \geq p,\ p_1' \geq p$

Oblivious lower bound

$\overline{p}_1 \cdot \overline{p}_1' \geq \overline{p}$

G,Suciu [TODS'14]

1. Theory: Upper bounds on the probability of monotone DNFs

2. Practise: Approximate lifted inference with probabilistic databases

Conjunctive Queries & Probabilistic Databases (PDBs)

Query in SQL

```
SELECT distinct T.C
FROM   R,S,T
WHERE  R.A=S.A
  and  S.B=T.B
```

Join

Schema

R(A)

S(A,B)

T(B,C)

Query in Datalog

Q(z):- R(x), S(x,y), T(y,z)

Instance

R	A
0.5	a
0.7	d

S	A	B
0.7	a	b
0.8	a	c
0.7	d	d

T	B	C
0.7	b	e
0.8	c	e
0.8	d	f

Results

Q	C
0.41	e
0.39	f

Promise of PDBs: ranking of output, due to uncertainty of input

	DBs	PDBs
Query complexity	NP-hard	≥#P hard
Data complexity	**PTIME** ☺	**#P hard** ☹

Problem of PDBs: ranking is hard

Vardi [STOC'82]

Dalvi, Suciu [VLDB'04]

Background: Evaluating Probabilistic Queries

Q_1:- R(x), S(x,y), T(x,y)

R	A	S	A	B	T	A	B
p_1	a	p_3	a	b	p_6	a	b
p_2	d	p_4	a	c	p_7	a	c
		p_5	d	c	p_8	d	c

$$\mathbf{P}[Q] = p_1p_3p_6 \vee p_1p_4p_7 \vee p_2p_5p_8$$
$$= p_1(p_3p_6 \vee p_4p_7) \vee p_2(p_5p_8)$$

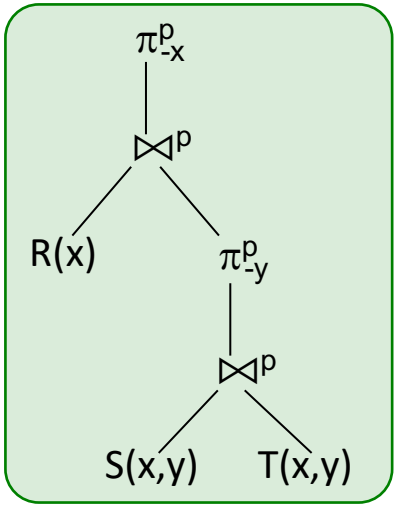
Read-Once formula

PTIME ☺

"hierarchical"

	x	y
R	○	
S	○	○
T	○	○

Incidence matrix



probabilistic query plan

Q_2:- R(x), S(x,y), T(y)

R	A	S	A	B	T	B
p_1	a	p_3	a	b	p_6	b
p_2	d	p_4	a	c	p_7	c
		p_5	d	c		

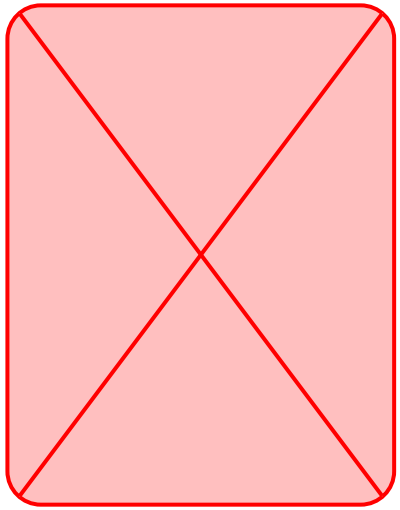
$$\mathbf{P}[Q] = p_1p_3p_6 \vee p_1p_4p_7 \vee p_2p_5p_7$$

NO Read-Once formula

#P hard ☹

not "hierarchical"

	x	y
R	○	
S	○	○
T		○



Dalvi, Suciu [VLDB'04]

The idea: Dissociation

Q_1:- R(x), S(x,y), T(x,y)

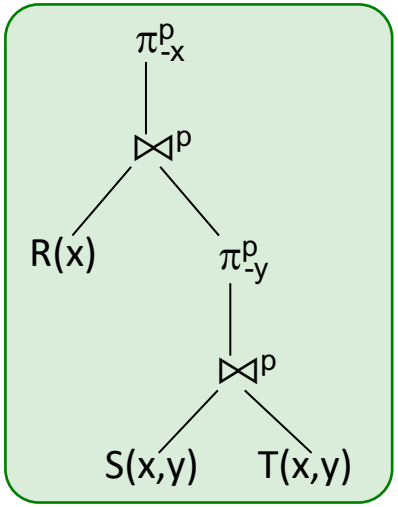
R	A	S	A	B	T	A	B
p_1	a	p_3	a	b	p_6	a	b
p_2	d	p_4	a	c	p_7	a	c
		p_5	d	c	p_8	d	c

$$\mathbf{P}[Q] = p_1p_3p_6 \vee p_1p_4p_7 \vee p_2p_5p_8$$
$$= p_1(p_3p_6 \vee p_4p_7) \vee p_2(p_5p_8)$$

PTIME ☺

"hierarchical"

	x	y
R	○	
S	○	○
T	○	○



Q_2:- R(x), S(x,y), T(y)

R	A	S	A	B	T	B
p_1	a	p_3	a	b	p_6	b
p_2	d	p_4	a	c	p_7	c
		p_5	d	c		

$$\mathbf{P}[Q] = p_1p_3p_6 \vee p_1p_4p_7 \vee p_2p_5p_7$$

	x	y
R	○	
S	○	○
T		○

The idea: Dissociation

Q_1:- R(x), S(x,y), T(x,y)

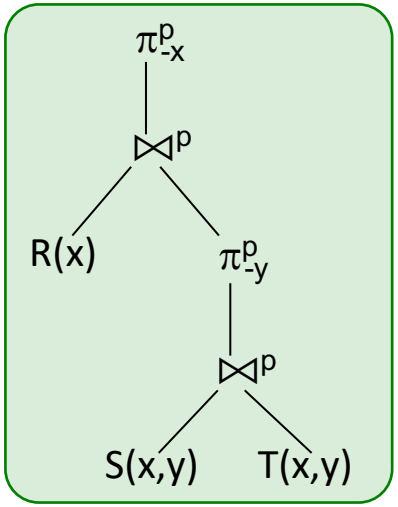
R	A	S	A	B	T	A	B
p_1	a	p_3	a	b	p_6	a	b
p_2	d	p_4	a	c	p_7	a	c
		p_5	d	c	p_8	d	c

$$\mathbf{P}[Q] = p_1p_3p_6 \vee p_1p_4p_7 \vee p_2p_5p_8$$
$$= p_1(p_3p_6 \vee p_4p_7) \vee p_2(p_5p_8)$$

PTIME ☺

"hierarchical"

	x	y
R	○	
S	○	○
T	○	○



Q^Δ_2:- R(x), S(x,y), T(x,y)

R	A	S	A	B	T	A	B
p_1	a	p_3	a	b	p_6	a	b
p_2	d	p_4	a	c	p_7	a	c
		p_5	d	c	p_7'	d	c

Query Dissociation

$$\mathbf{P}[Q^\Delta] = p_1p_3p_6 \vee p_1p_4p_7 \vee p_2p_5p_7'$$
$$= p_1(p_3p_6 \vee p_4p_7) \vee p_2p_5p_7'$$

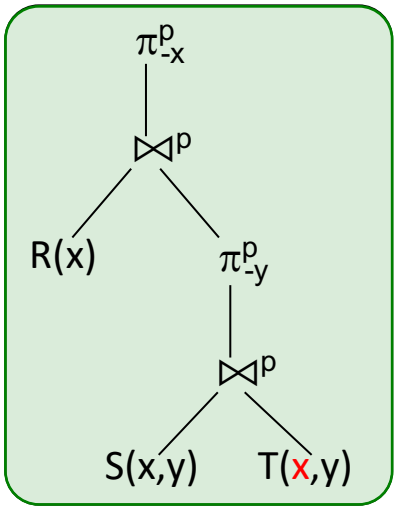
Read-Once formula ☺

dissociation of tuples

PTIME ☺

"hierarchical"

	x	y
R	○	
S	○	○
T	●	○



Can be evaluated with a DMBS

The idea: Dissociation

$Q_2^{\Delta'}$:- R(x,y), S(x,y), T(y)

2

PTIME ☺

"hierarchical"

R	A	B
p_1	a	b
p_1'	a	c
p_2	d	c

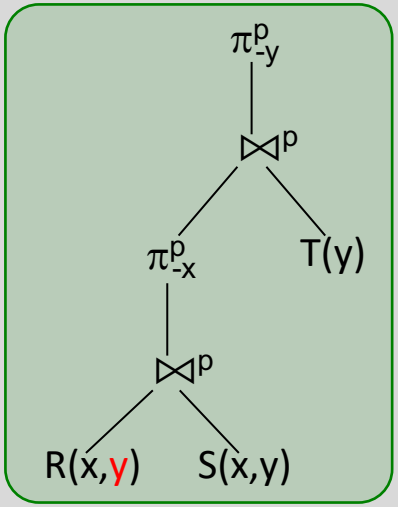
S	A	B
p_3	a	b
p_4	a	c
p_5	d	c

T	B
p_6	b
p_7	c

Query Dissociation

	x	y
R	○	●
S	○	○
T		○

$$\mathbf{P}[Q^{\Delta'}] = p_1p_3p_6 \vee p_1'p_4p_7 \vee p_2p_5p_7$$
$$= p_1p_3p_6 \vee (p_1'p_4 \vee p_2p_5)p_7$$



Q_2^{Δ}:- R(x), S(x,y), T(x,y)

1

PTIME ☺

"hierarchical"

R	A
p_1	a
p_2	d

S	A	B
p_3	a	b
p_4	a	c
p_5	d	c

T	A	B
p_6	a	b
p_7	a	c
p_7'	d	c

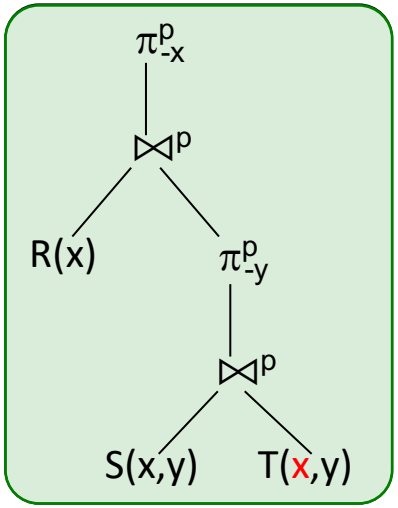
Query Dissociation

	x	y
R	○	
S	○	○
T	●	○

$$\mathbf{P}[Q^{\Delta}] = p_1p_3p_6 \vee p_1p_4p_7 \vee p_2p_5p_7'$$
$$= p_1(p_3p_6 \vee p_4p_7) \vee p_2p_5p_7'$$

Read-Once formula ☺

dissociation of tuples



Can be evaluated with a DMBS

Partial Dissociation Order and Propagation

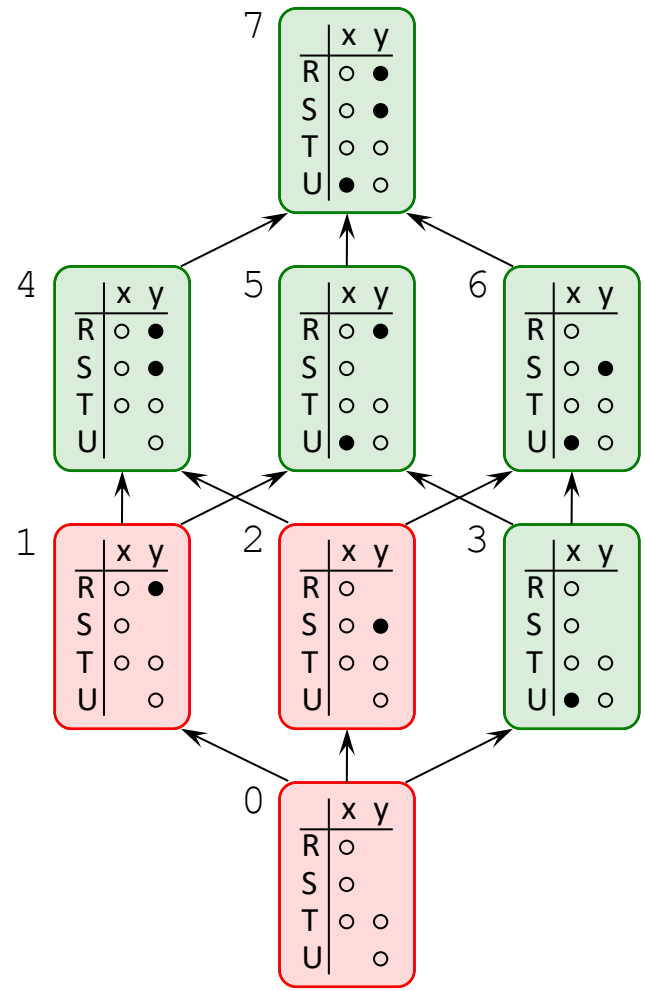
Q_3:- R(x), S(x), T(x,y), U(y)

Def. "Partial dissociation order" $\leq$:

$\Delta \leq \Delta' \;\Leftrightarrow\; \forall$relations R : $Var(R^{\Delta}) \supseteq Var(R^{\Delta'})$

Theorem 1:

$\boxed{\Delta \leq \Delta' \;\Leftrightarrow\; \mathbf{P}[Q^{\Delta}] \leq \mathbf{P}[Q^{\Delta'}]}$



Def. "PTIME dissociation":

Δ is PTIME $\Leftrightarrow$ Q^{Δ} is PTIME

Def. "Propagation score":

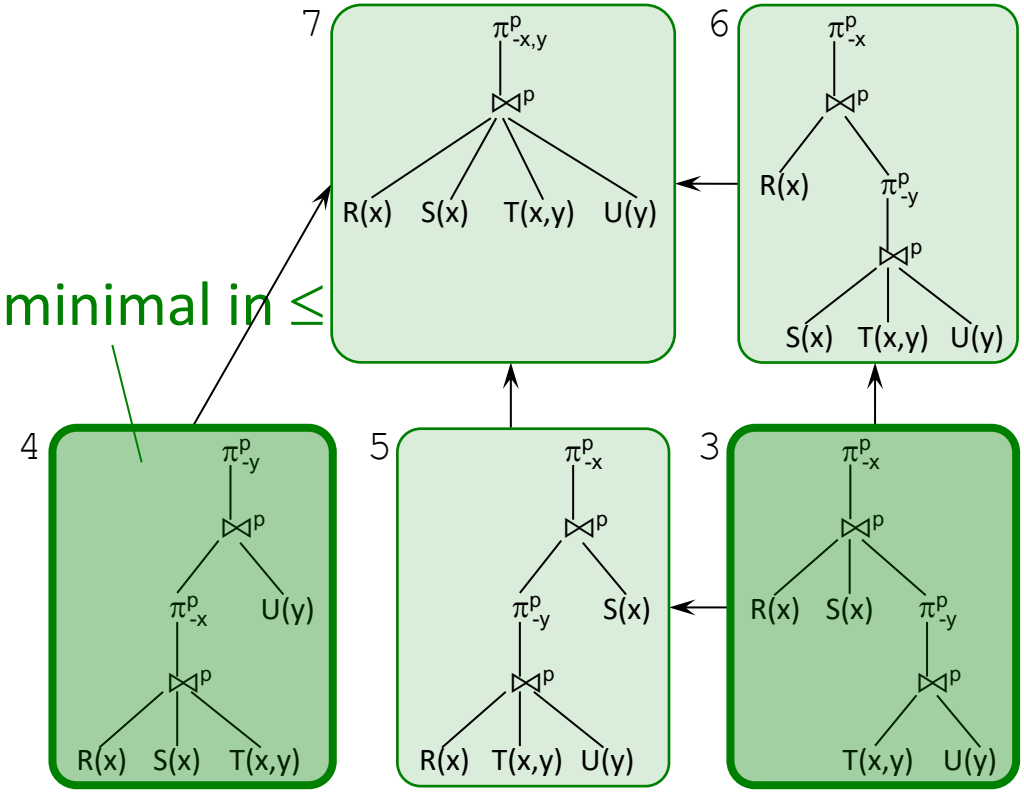
minimum prob. of all PTIME dissociations

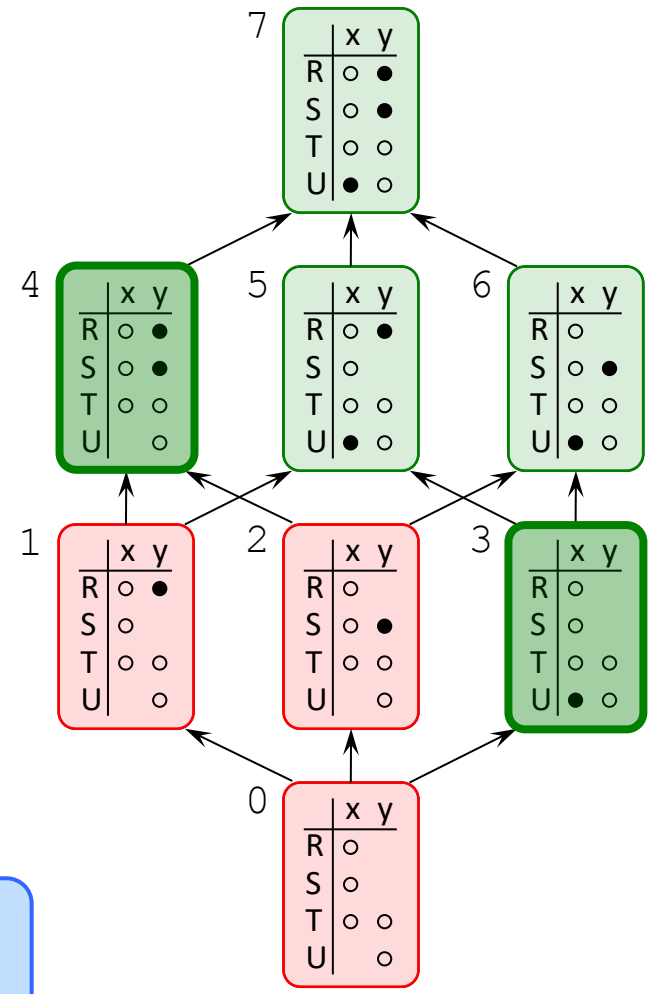
$\rho[Q] := \mathrm{MIN}_{\Delta\,:\,\mathrm{safe}}\; \mathbf{P}[Q^{\Delta}]$

Partial Dissociation Order and Propagation

Q_3:- R(x), S(x), T(x,y), U(y)

Theorem 2: Isomorphism b/w PTIME dissociations and probabilistic query plans: $\mathbf{P}[Q^\Delta] = \mathbf{P}[P^\Delta]$



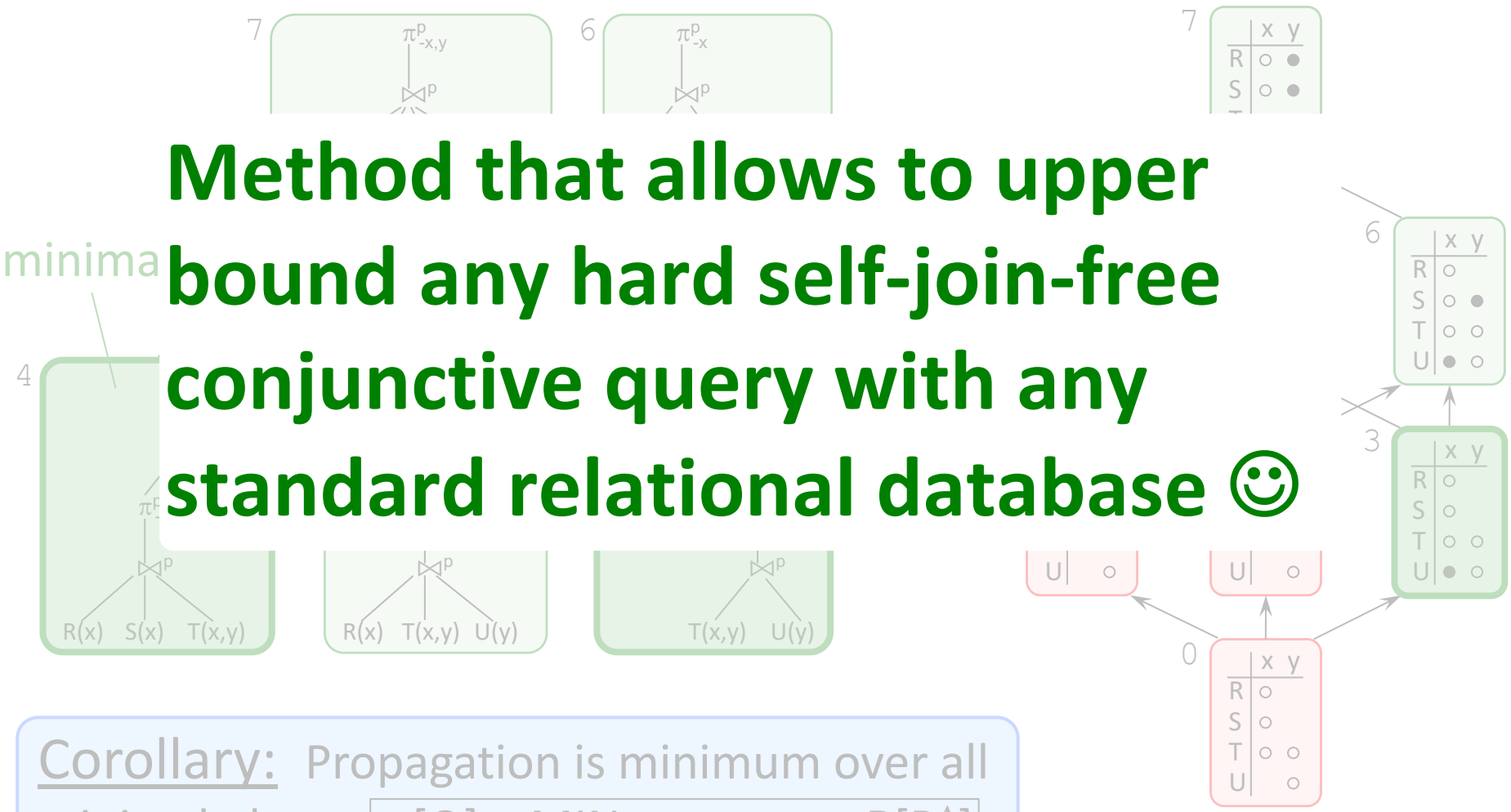


Corollary: Propagation is minimum over all minimal plans: $\rho[Q] = \mathrm{MIN}_{\Delta\,:\,\text{minimal in}\,\leq}\ \mathbf{P}[P^\Delta]$

Partial Dissociation Order and Propagation

Q_3:- R(x), S(x), T(x,y), U(y)

Theorem 2: Isomorphism b/w safe dissociations and probabilistic query plans: $\mathbf{P}[Q^\Delta] = \mathbf{P}[P^\Delta]$



Method that allows to upper bound any hard self-join-free conjunctive query with any standard relational database ☺

Corollary: Propagation is minimum over all minimal plans: $\rho[Q] = \mathrm{MIN}_{\Delta\,:\,\text{minimal in}\,\leq}\ \mathbf{P}[P^\Delta]$

What else you will find in the papers and TR

- All proofs
- Multi-query Optimizations
- Deterministic tables and FDs
  - Conservative extension of all PTIME queries
  - Conservative extension of standard SQL
- Detailed Evaluations on synthetic TPC-H data
- Generalizes network propagation to hypergraphs

Take-aways

Problems

- ☹ Ranking over uncertain databases is #P-hard
- ☹ No native support of uncertainty in databases today

Proposition: Approximate lifted inference w/ DBs ☺ PTIME

- works fast & accurate (recall experiments at the beginning)
- upper bound: efficient ranking & "filtering"
- can be implemented in any existing DBMS
- strict generalization of all "safe queries" for sf-CQs
- deterministic approximation: reproducible

Current work:

- extending this to CQs with self-joins

Please come to our poster tonight. Thanks!

BACKUP

Algorithm for enumerating all minimal query plans

Q(x) :- R(y,u,v), S(x,y,z,u), T(x,z)

Step 1: project on all minimal sets of **EVar**(q) that disconnect the query: **TVar**(q)

	x	y	z	u	v
R		○		○	○
S	○	○	○	○	
T	○		○		

HVar(q) = {x}

EVar(q) = {y,z,u,v}

Head, **E**xistential

	x	y	u	z	v
R		○	○		○
S	○	○	○	○	
T	○	●	●	○	

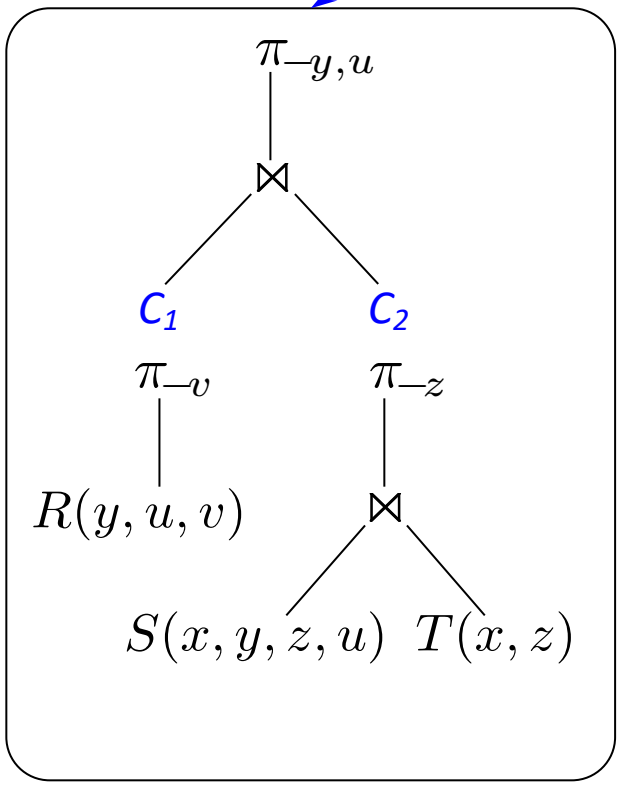
Step 2: join disconnected components

C_1	y	u	v
R	○	○	○

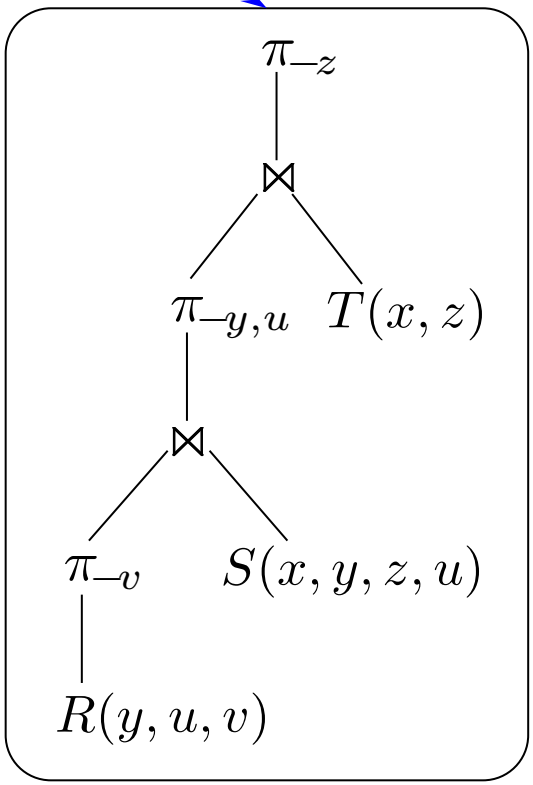
C_2	x	y	u	z
S	○	○	○	○
T	○	●	●	○

Top

TVar$_1$(q) = {y,u}



TVar$_2$(q) = {z}



Optimization 1

Q_5:- R(x,z), S(y,u), T(z), U(u), M(x,y,z,u)

$$\rho(q) = \min \begin{bmatrix} \pi^p_{-z} \bowtie^p [T(z)\ \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]]]] \\ \pi^p_{-z} \bowtie^p [T(z)\ \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]]]] \\ \pi^p_{-z} \bowtie^p [T(z)\ \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]]]] \\ \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-z} \bowtie^p [T(z), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]]]] \\ \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-z} \bowtie^p [T(z), \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]]]] \\ \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-z} \bowtie^p [T(z), \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]]]] \end{bmatrix}$$

1: Pushing min operator into roots

$$\rho(q) = \min \begin{bmatrix} \pi^p_{-z} \bowtie^p [T(z)\ \min \begin{bmatrix} \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]]] \\ \pi^p_{-u} \bowtie^p [U(u)\ \min \begin{bmatrix} \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]] \\ \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]] \end{bmatrix}] \end{bmatrix}] \\ \pi^p_{-u} \bowtie^p [U(u), \min \begin{bmatrix} \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-z} \bowtie^p [T(z), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]]] \\ \pi^p_{-z} \bowtie^p [T(z), \min \begin{bmatrix} \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]] \\ \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]] \end{bmatrix}] \end{bmatrix}] \end{bmatrix}$$

Optimization 2

$$V_1(x,z,u) = \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]$$

$$V_2(y,z,u) = \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]$$

$$V_3(z,u) = \min \begin{bmatrix} \pi^p_{-x} \bowtie^p [R(x,z), V_1(x,z,u)] \\ \pi^p_{-y} \bowtie^p [S(y,u), V_2(y,z,u)] \end{bmatrix}$$

$$\rho(q) = \min \begin{bmatrix} \pi^p_{-z} \bowtie^p [T(z), \min \begin{bmatrix} \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-u} \bowtie^p [U(u), V_1(x,z,u)]] \\ \pi^p_{-u} \bowtie^p [U(u), V_3(z,u)] \end{bmatrix}] \\ \pi^p_{-u} \bowtie^p [U(u), \min \begin{bmatrix} \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-z} \bowtie^p [T(z), V_2(y,z,u)]] \\ \pi^p_{-z} \bowtie^p [T(z), V_3(z,u)] \end{bmatrix}] \end{bmatrix}$$

2: Re-using common subplans

$$\rho(q) = \min \begin{bmatrix} \pi^p_{-z} \bowtie^p [T(z), \min \begin{bmatrix} \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]]] \\ \pi^p_{-u} \bowtie^p [U(u), \min \begin{bmatrix} \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]] \\ \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]] \end{bmatrix}] \end{bmatrix}] \\ \pi^p_{-u} \bowtie^p [U(u), \min \begin{bmatrix} \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-z} \bowtie^p [T(z), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]]] \\ \pi^p_{-z} \bowtie^p [T(z), \min \begin{bmatrix} \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]] \\ \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]] \end{bmatrix}] \end{bmatrix}] \end{bmatrix}$$

Optimization 3: Deterministic semi-join reduction

$$BV(x, y, z, u, p_R, p_S, p_T, p_U, p_M) :-$$
$$R(x, z, p_R), S(y, u, p_S), T(z, p_T), U(u, p_U), M(x, y, z, u, p_M)$$

$$R^*(x, z, p_R) :- BV(x, y, z, u, p_R, p_S, p_T, p_U, p_M)$$
$$S^*(y, u, p_S) :- BV(x, y, z, u, p_R, p_S, p_T, p_U, p_M)$$
$$T^*(z, p_T) :- BV(x, y, z, u, p_R, p_S, p_T, p_U, p_M)$$
$$U^*(u, p_U) :- BV(x, y, z, u, p_R, p_S, p_T, p_U, p_M)$$
$$M^*(x, y, z, u, p_M) :- BV(x, y, z, u, p_R, p_S, p_T, p_U, p_M)$$

Deterministic join allows database to find optimal query plan for reducing the number of tuples prior to probabilitic evaluation

Using further schema knowledge: Baseline

$$q :- R(x,z), S(y,u), T(z), U(u), M(x,y,z,u)$$

	x	z	y	u
M	○	○	○	○
R	○	○		
T		○		
S			○	○
U				○

TVar(q) = {{z},{u}}

$$\rho(q) = \min \begin{array}{r} \{z\} \\ \\ \\ \\ \\ \{u\} \end{array} \left[\begin{array}{l}
\pi^p_{-z} \bowtie^p [T(z), \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]]]] \\
\pi^p_{-z} \bowtie^p [T(z), \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]]]] \\
\pi^p_{-z} \bowtie^p [T(z), \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]]]] \\
\pi^p_{-u} \bowtie^p [U(u), \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-z} \bowtie^p [T(z), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]]]] \\
\pi^p_{-u} \bowtie^p [U(u), \pi^p_{-z} \bowtie^p [T(z), \pi^p_{-x} \bowtie^p [R(x,z), \pi^p_{-y} \bowtie^p [S(y,u), M(x,y,z,u)]]]] \\
\pi^p_{-u} \bowtie^p [U(u), \pi^p_{-z} \bowtie^p [T(z), \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-x} \bowtie^p [R(x,z), M(x,y,z,u)]]]]
\end{array} \right]$$

4: Functional Dependencies: preprocessing step

$$q :- R(x, \underline{z}), S(y, \underline{u}), T(z), U(u), M(x, y, z, u)$$

FDs: $z \to x,\ u \to y$

	x	z	y	u
M	○	○	○	○
R	○	○		
T	⋆	○		
S			○	○
U			⋆	○

TVar(q) = {{x,z}{y,u}}

Sound and complete adaptation: at each recursion of the previous algorithm, dissociate eagerly all implied FDs

Stars in matrix do not change query reliability

$$\rho(q) = \min \begin{bmatrix} \pi^p_{-x,z} \bowtie^p [R(x,z), T(z), \pi^p_{-y,u} \bowtie^p [S(y,u), U(u), M(x,y,z,u)]] \\ \pi^p_{-y,u} \bowtie^p [S(y,u), U(u), \pi^p_{-x,z} \bowtie^p [T(z), R(x,z), M(x,y,z,u)]] \end{bmatrix}$$

Projection on 2 variables at once

Functional dependencies simplify and reduce number of query plans

5: Deterministic Tables: adapted step 1

$$q :- R^d(x,z), S(y,u), T^d(z), U(u), M(x,y,z,u)$$

	x	z	y	u
M	○	○	○	○
S			○	○
U				○
R^d	○	○	⋆	⋆
T^d	⋆	○	⋆	⋆

Sound and complete adaptation of step 1:
if $x \in$ all prob tables $\Rightarrow$ it must be in **TVar**

TVar(q) = {{u}}

$$\rho(q) = \pi^p_{-u} \bowtie^p [U(u), \pi^p_{-y} \bowtie^p [S(y,u), \pi^p_{-x,z} \bowtie^p [T^d(z), R^d(x,z), M(x,y,z,u)]]]$$

Here, deterministic tables reduce to one simplified query plan

4 & 5: Natural extension of all known safe queries

$$q :- R^d(x, \underline{z}), S(y, \underline{u}), T^d(z), U(u), M(x, y, z, u)$$

	x	z	y	u
M	○	○	○	○
S			○	○
U			★	○
R^d	○	○	★	★
T^d	★	○	★	★

1 single algorithm for enumerating all minimal query plans !!!

TVar(q) = {{u,y}}

Query is safe iff query enumeration algorithm returns one single query

$$\rho(q) = \pi^p_{-y,u} \bowtie^p [U(u), S(y, u), \pi^p_{-x,z} \bowtie^p [T^d(z), R^d(x, z), M(x, y, z, u)]]$$

SQL translation Variant 1

Deterministic SQL

```
select distinct s_nationkey
from Supplier, Partsupp, Part
where s_suppkey = ps_suppkey
and ps_partkey = p_partkey
and p_name like $2
and ps_suppkey <= $1;
```

```
drop table if exists rus;
create table rus as
    select s_nationkey, iOR(Q3.P) as P
    from
        (select s_nationkey, P.P*Q2.P as P
        from Part P,
            (select Q1.s_nationkey, Q1.ps_partkey, iOR(Q1.P) as P
            from
                (select s_nationkey, s_suppkey, ps_partkey, S.P*PS.P as P
                from Supplier as S, Partsupp as PS
                where s_suppkey = ps_suppkey
                and s_suppkey <= $1) as Q1
            group by Q1.s_nationkey, Q1.ps_partkey) as Q2
        where p_partkey = Q2.ps_partkey
        and p_name like $2) as Q3
    group by Q3.s_nationkey;
```

```
drop table if exists rus;
create table rus as
    select s_nationkey, iOR(Q3.P) as P
    from
        (select s_nationkey, P.P*Q2.P as P
        from Part P,
            (select Q1.s_nationkey, Q1.ps_partkey, iOR(Q1.P) as P
            from
                (select s_nationkey, s_suppkey, ps_partkey,S.P*PS.P as P
                from Supplier as S, Partsupp as PS
                where s_suppkey = ps_suppkey
                and s_suppkey <= $1) as Q1
            group by Q1.s_nationkey, Q1.ps_partkey) as Q2
        where p_partkey = Q2.ps_partkey
        and p_name like $2) as Q3
    group by Q3.s_nationkey;
```

```
select rup.s_nationkey, least(rup.p, rus.p) ru
from rup, rus
where rup.s_nationkey = rus.s_nationkey
order by least(rup.p, rus.p) desc;
```

SQL translation Variant 2

Deterministic SQL

```
select distinct s_nationkey
from Supplier, Partsupp, Part
where s_suppkey = ps_suppkey
and ps_partkey = p_partkey
and p_name like $2
and ps_suppkey <= $1;
```

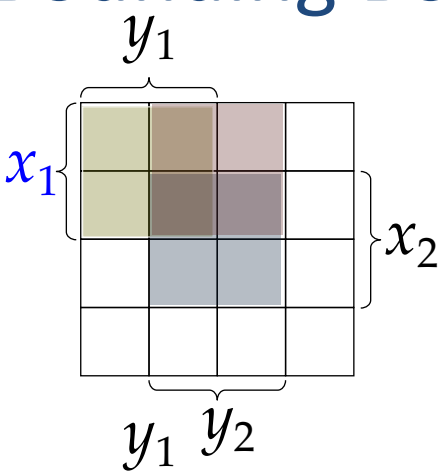
```
drop table if exists PS cascade;
create table PS as
    select ps_suppkey, ps_partkey, PS.P
    from Supplier S, Partsupp PS, Part P
    where s_suppkey = ps_suppkey
    and p_partkey = ps_partkey
    and p_name like $2
    and s_suppkey <= $1
    group by PS.ps_suppkey, PS.ps_partkey, PS.P;

drop table if exists P cascade;
create table P as
    select p_partkey, P.P
    from PS, Part P
    where p_partkey = ps_partkey
    group by P.p_partkey, P.P;
```

```
drop table if exists rus;
create table rus as
    select s_nationkey, iOR(Q3.P) as P
    from
        (select s_nationkey, P.P*Q2.P as P
        from ~~Part~~ P,
            (select Q1.s_nationkey, Q1.ps_partkey, iOR(Q1.P) as P
            from
                (select s_nationkey, s_suppkey, ps_partkey, S.P*PS.P as P
                from Supplier as S, ~~Partsupp as~~ PS
                where s_suppkey = ps_suppkey
                ~~and s_suppkey <= $1~~) as Q1
            group by Q1.s_nationkey, Q1.ps_partkey) as Q2
        where p_partkey = Q2.ps_partkey
        ~~and p_name like $2~~) as Q3
    group by Q3.s_nationkey;
```

```
drop table if exists rus;
create table rus as
    select s_nationkey, iOR(Q3.P) as P
    from
        (select s_nationkey, P.P*Q2.P as P
        from ~~Part~~ P,
            (select Q1.s_nationkey, Q1.ps_partkey, iOR(Q1.P) as P
            from
                (select s_nationkey, s_suppkey, ps_partkey,S.P*PS.P as P
                from Supplier as S, ~~Partsupp as~~ PS
                where s_suppkey = ps_suppkey
                ~~and s_suppkey <= $1~~) as Q1
            group by Q1.s_nationkey, Q1.ps_partkey) as Q2
        where p_partkey = Q2.ps_partkey
        ~~and p_name like $2~~) as Q3
    group by Q3.s_nationkey;
```

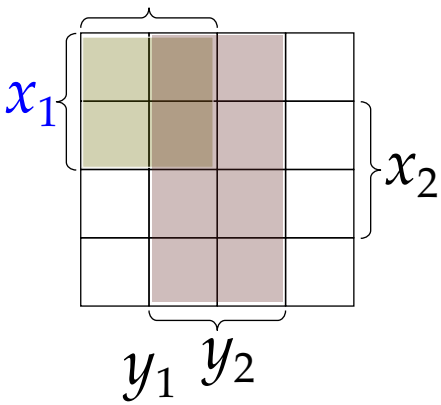
```
select rup.s_nationkey, least(rup.p, rus.p) ru
from rup, rus
where rup.s_nationkey = rus.s_nationkey
order by least(rup.p, rus.p) desc;
```

Bounding Boolean Formulas by Models



$$f = x_1y_1 \vee x_1y_2 \vee x_2y_2$$

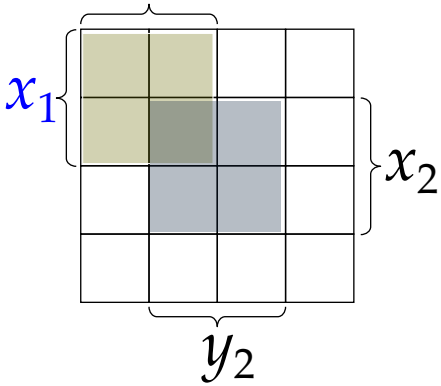
$$f' = x_1y_1 \vee x_1'y_2 \vee x_2y_2$$



Oblivious upper bound: $x_1' = 1$

$$f' = x_1y_1 \vee \quad y_2 \vee \cancel{x_2y_2}$$

Read-once, PTIME ☺



Oblivious lower bound: $x_1' = 0$

$$f' = x_1y_1 \qquad \vee x_2y_2$$

Read-once, PTIME ☺

Oblivous framework: p' is computed only as function of p, thus uses only "local" information, thus can be fast

Oblivious Bounds, Relaxation & Compensation, Models

$d = 2$

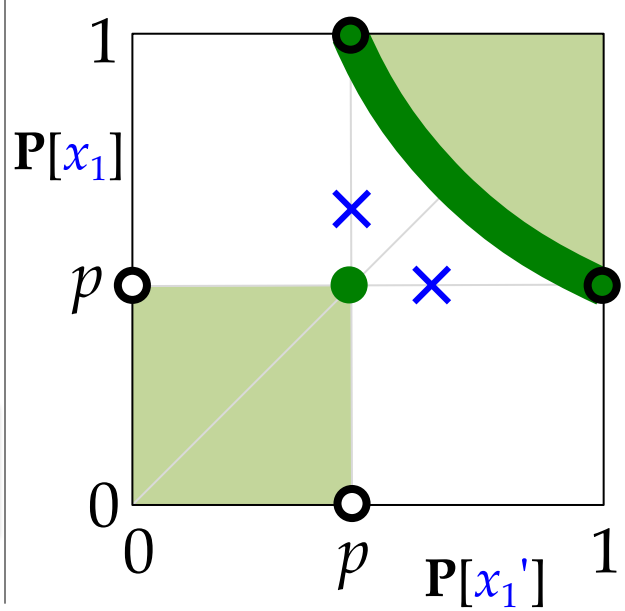
(similar results hold for arbitrary d)

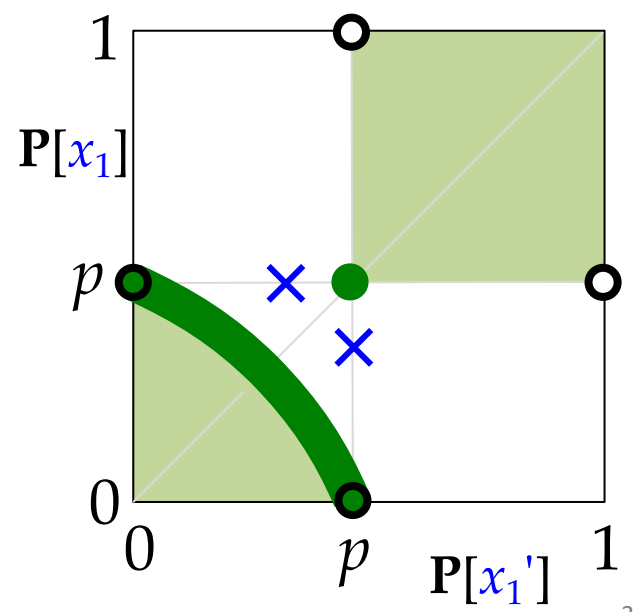
		Conjunctive D.	Disjunctive D.
		$f = f_1 \wedge f_2$ $f' = f_1[x_1'/x] \wedge f_2[x_1'/x]$	$f = f_1 \vee f_2$ $f' = f_1[x_1'/x] \vee f_2[x_1'/x]$
● **Optimal obli-vious bounds**	Upper	$p_1 \cdot p_2 = p$	$p_1 = p,\ p_2 = p$
	Lower	$p_1 = p,\ p_2 = p$	$\overline{p_1} \cdot \overline{p_2} = \overline{p}$
○ **Model-based bounds**	Upper	$p_1 = p,\ p_2 = 1$ (optimal)	$p_1 = p,\ p_2 = 1$ (non-optimal)
	Lower	$p_1 = p,\ p_2 = 0$ (non-optimal)	$p_1 = p,\ p_2 = 0$ (optimal)
× **Relaxation & Comp.**		$p_1 = p,\ p_2 = \mathbf{P}[x \mid f_1]$	$p_1 = p,\ p_2 = \mathbf{P}[x \mid \overline{f_1}]$

● G, Suciu [TODS'14]

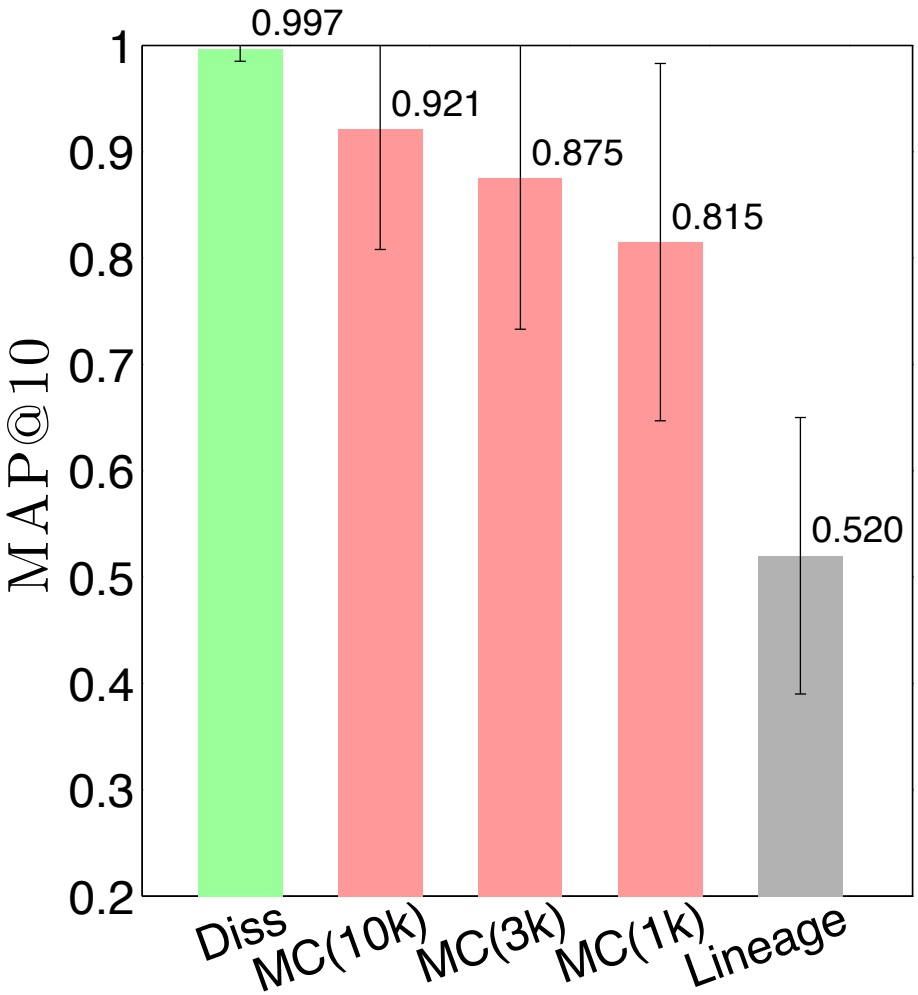
○ Fink, Olteanu [ICDT'11]

× Choi, Darwiche [NIPS'09, JSAI-isAI'10]





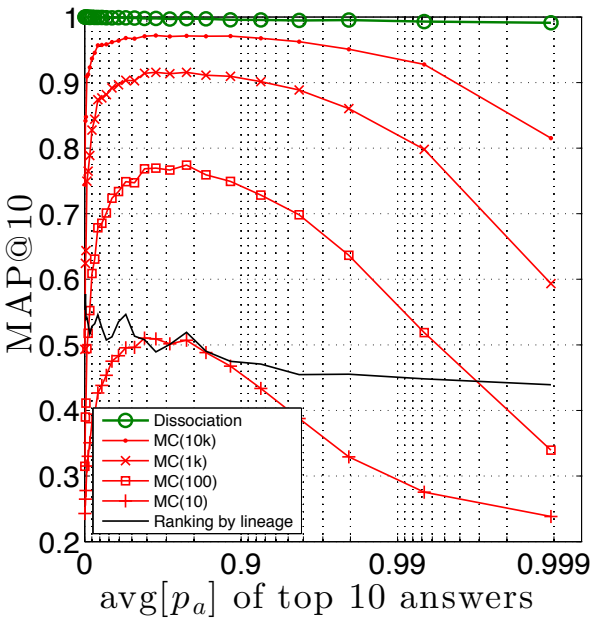
Quality: Big Picture

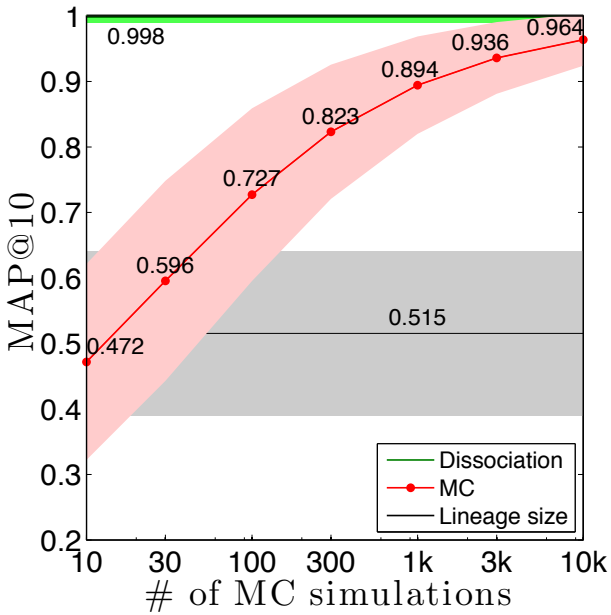


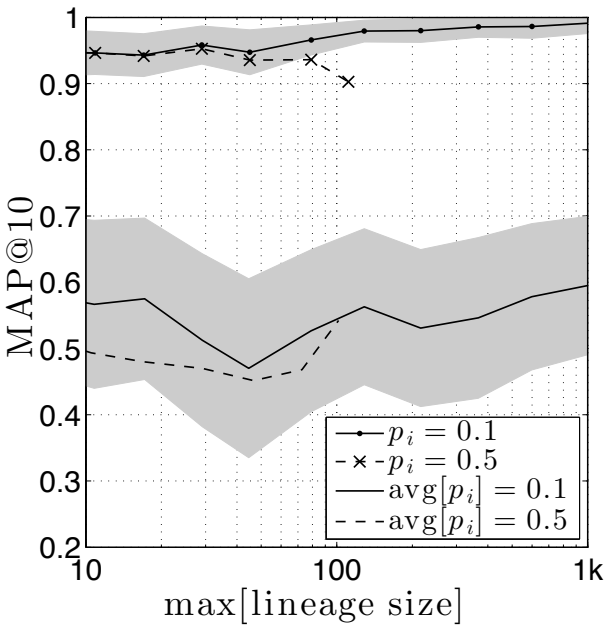
- 20 values for $1: {500, 1000, ..., 10k}
- 28 values for $2: {%red%a, ...}
- 10 values for avg[p_i]: {0.05, ..., 0.5}
- >120k datapoints per method randomized database instances

Random ranking
AP@10 = 0.220

MC quality depends on avg[p_a], Lineage ranking is robust





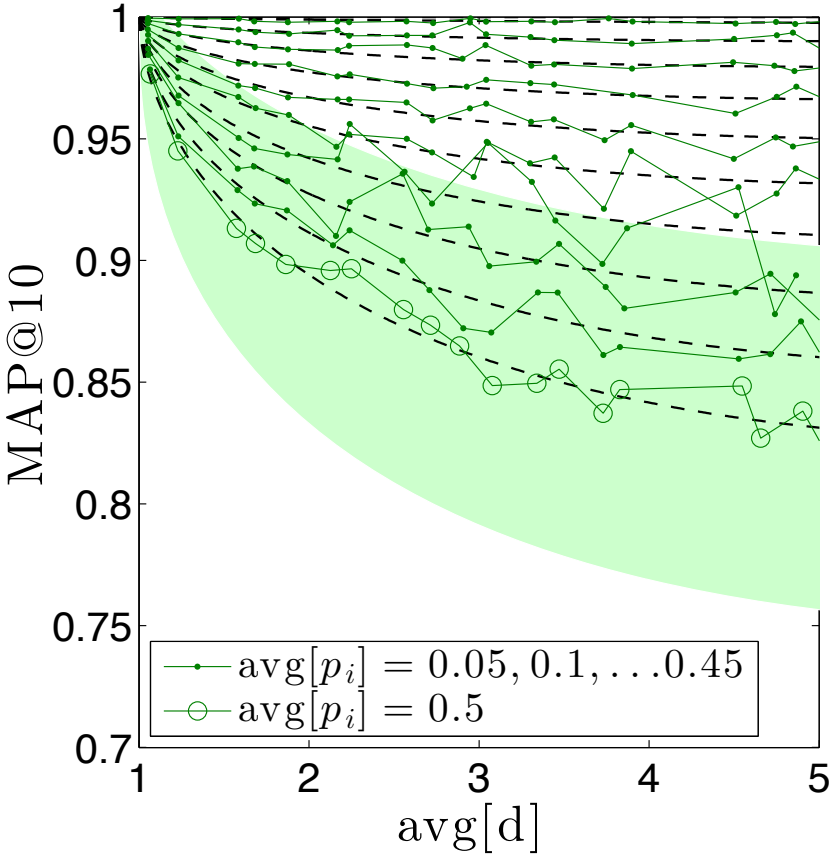


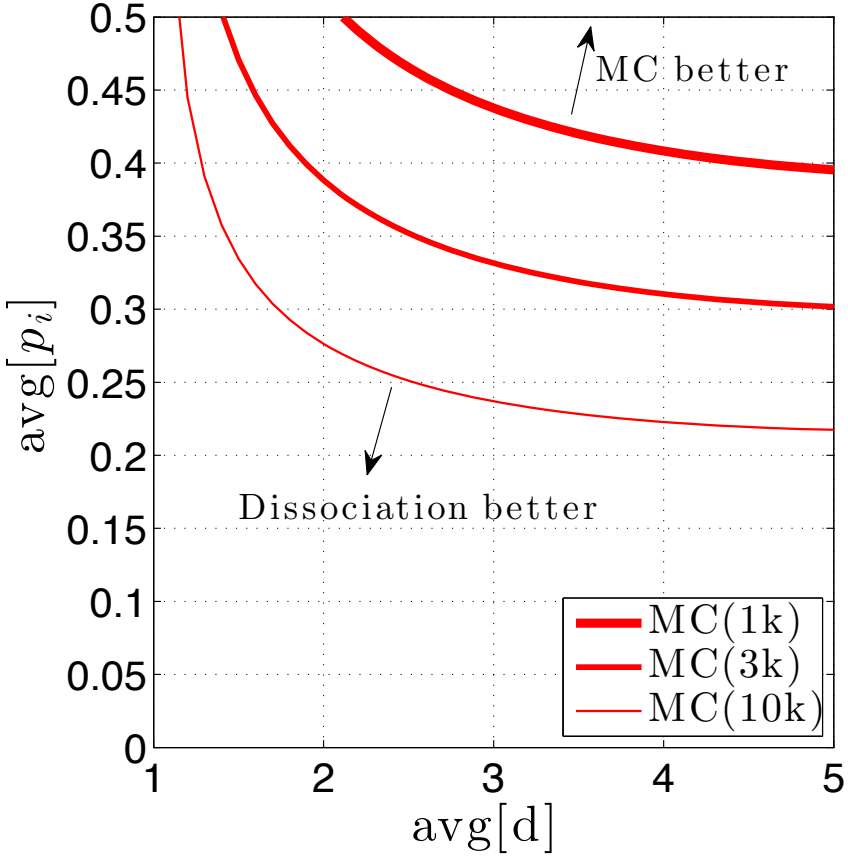
- 4.5k simulations per MC data point

- 65k data points for MC
- 6k data points for D,L
- 0.1 < avg[p_a] < 0.9

- p_a = const
- avg[p_a] = const

Dissociation Quality depends on avg[p_i] and avg[d]

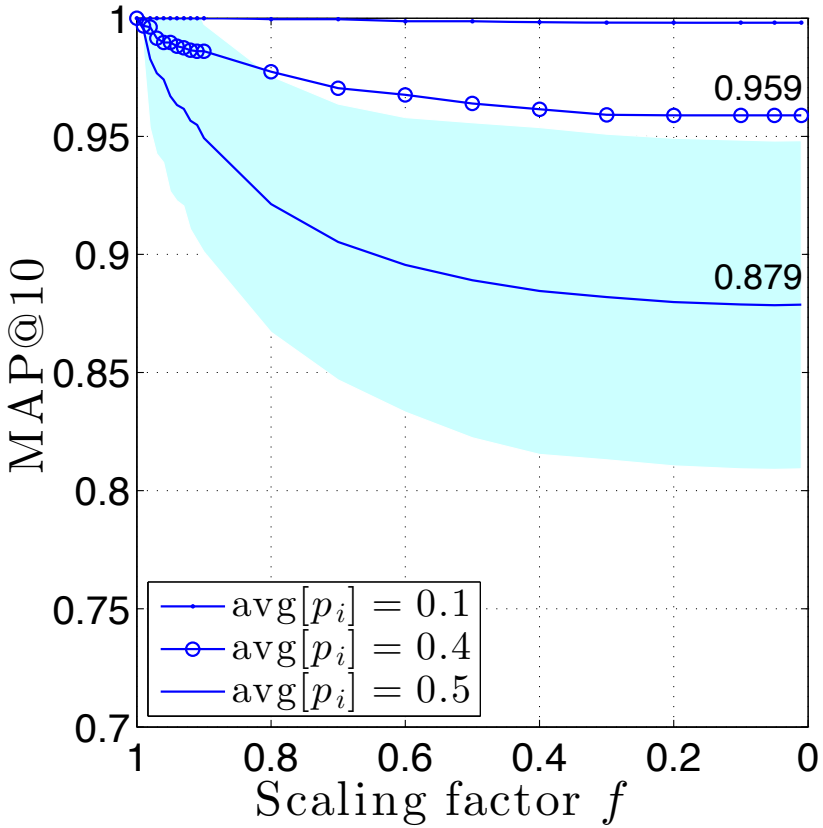


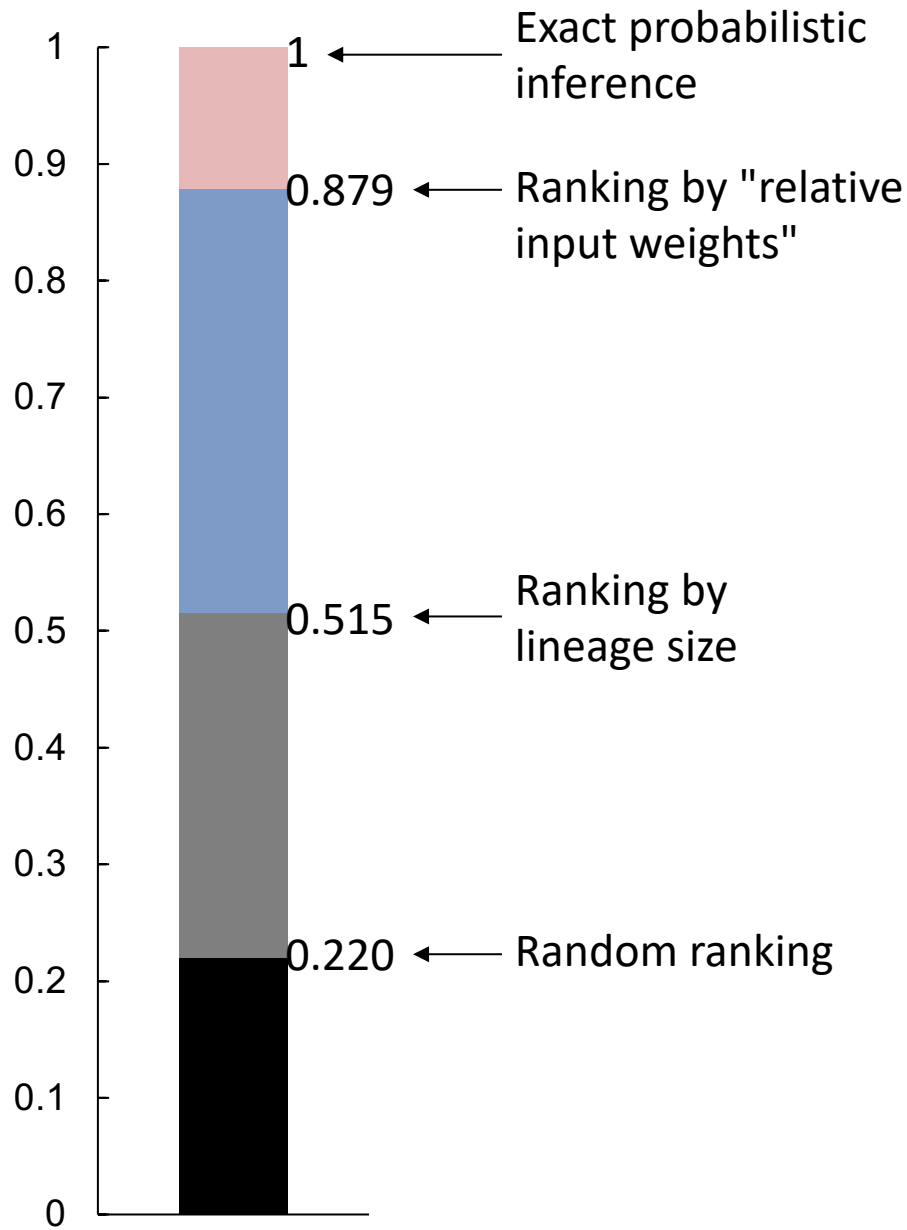


- Evaluate both plans independently
- For all plans avg[d] < 1.1 for best combination

- MC: best regime: 0.1 < avg[p_a] < 0.9

What is the value of exact Probabilistic Inference?



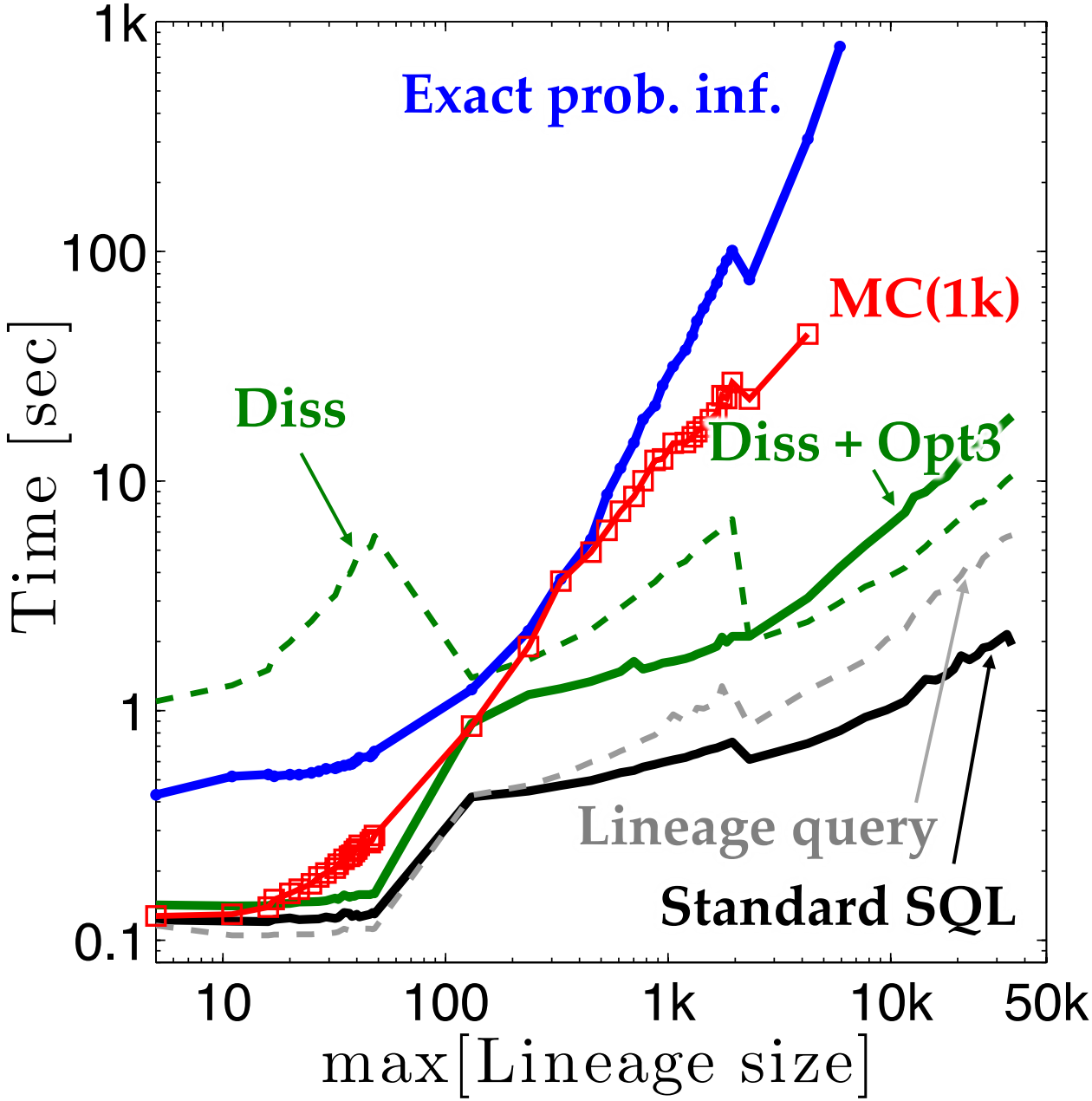


Number of minimal plans (#MP), total plans (#P), and total dissociations for star and chain queries

k-star query				k-chain query			
k	#MP	#P	#Δ	k	#MP	#P	#Δ
1	1	1	1	2	1	1	1
2	2	3	4	3	2	3	4
3	6	13	64	4	5	11	64
4	24	75	4096	5	14	45	4096
5	120	541	$> 10^6$	6	42	197	$> 10^6$
6	720	4683	$> 10^9$	7	132	903	$> 10^9$
7	5040	47293	$> 10^{12}$	8	429	4279	$> 10^{12}$
seq	$k!$	A000670	$2^{k(k-1)}$	seq	A000108	A001003	$2^{(k+1)k}$

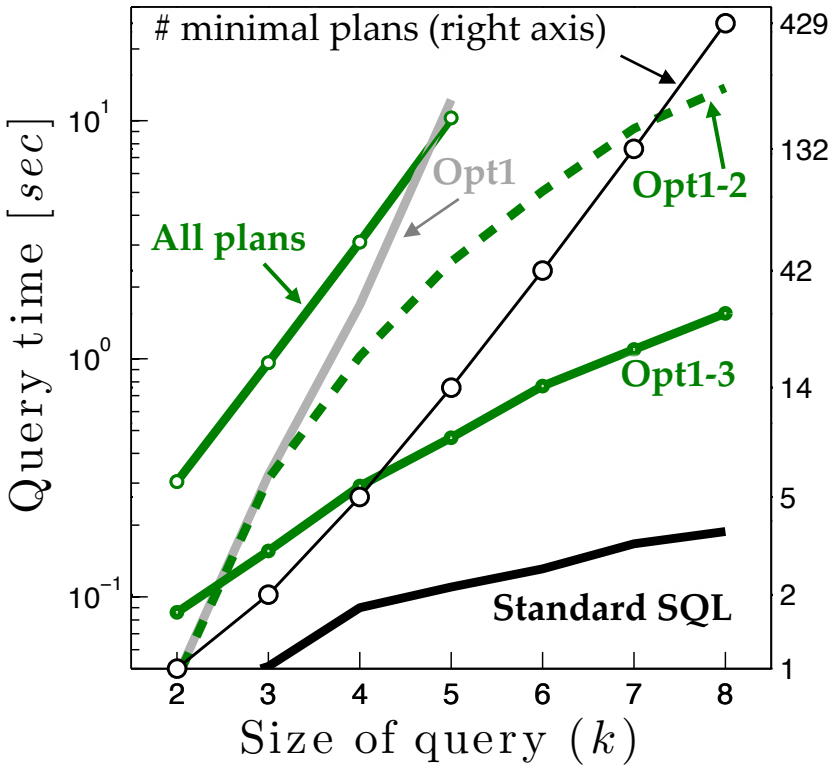
A stands for OEIS sequence numbers (Online Encyclopedia of Integer Numbers)

Timing experiments (Big Picture)



Relation query optimizations

k-chain queries



k-star queries