PrDB: Increasing the Representational Power and Scaling Reasoning in Probabilistic Databases

Amol Deshpande, University of Maryland

(joint work w/ Prof. Lise Getoor, Bhargav Kanagal, Jian Li, and Prithviraj Sen)

Motivation

- Increasing amounts of real-world uncertain data
  - Sensor networks, Scientific databases, Social networks…
    - Noisy, error-prone observations
    - Imprecise data, data with confidence or accuracy bounds
    - Widespread use of statistical and probabilistic models
      - … for *entity resolution, link prediction, function prediction* etc.
  - Automatically constructed knowledge-bases
    - Noisy data sources, automatically derived schema mappings
    - Reputation/trust/staleness issues
    - Automatically extracted knowledge from text
- Need to develop database systems for efficiently representing and managing uncertainty

Probabilistic Databases

- Several approaches proposed in recent years in DB literature
  - Typically based on *probability theory*
    - Annotate tuples with probabilities of existence *(tuple-existence uncertainty)*
    - Specify a *pdf* over possible values of an attribute (*attribute-value uncertainty)*
  - Focus on SQL query evaluation, but inference also considered
  - *Strong independence assumptions; limited attribute uncertainty support*

- **PrDB Goals:**
  - Increase representationl power to support:
    - Correlations among the data items
    - Uncertainties at different abstraction levels and granularities
  - Scale reasoning and querying to large-scale uncertain data while supporting the above

An Example Probabilistic Database

- Example from Dalvi and Suciu [2004]
- Assume independent tuples

S

	A	B	*prob*
s1	'm'	1	0.6
s2	'n'	1	0.5

T

	B	C	*prob*
t1	1	'p'	0.4

Interpret as a distribution over a set of deterministic possible worlds

Possible worlds

instance	probability
{s1, s2, t1}	0.12
{s1, s2}	0.18
{s1, t1}	0.12
{s1}	0.18
{s2, t1}	0.08
{s2}	0.12
{t1}	0.08
{}	0.12

*p(s1) * p(t1) * (1-p(s2))*
*= 0.6 * 0.4 * 0.5*
= 0.12

Query Processing Semantics

- Evaluate on each possible world and combine results
- Example Query: $\pi_C(S \bowtie_B T)$

select C
from S, T
where S.B = T.B

S

	A	B	*prob*
s1	'm'	1	0.6
s2	'n'	1	0.5

T

	B	C	*prob*
t1	1	'p'	0.4

$\bowtie$

	A	B	C
i1	'm'	1	'p'
i2	'n'	1	'p'

π_C

	C
r1	'p'

Query Processing Semantics

- Evaluate on each possible world and combine results
- Example Query: $\pi_C(S \bowtie_B T)$

S

	A	B	*prob*
s1	'm'	1	0.6
s2	'n'	1	0.5

T

	B	C	*prob*
t1	1	'p'	0.4

<u>*Possible worlds*</u> <u>*Query Result*</u>

instance	prob	result
{s1, s2, t1}	0.12	{'p'}
{s1, s2}	0.18	{}
{s1, t1}	0.12	{'p'}
{s1}	0.18	{}
{s2, t1}	0.08	{'p'}
{s2}	0.12	{}
{t1}	0.0	
{}	0.1	

	C	*prob*
r1	'p'	0.32

Not clear how to do this in general

e.g. ranking ??

Consensus Answers [PODS'09]

Outline

- Probabilistic Databases: Overview, Limitations
- PrDB: Overview
- PrDB: Some Details
  - Instance-optimal query execution
  - Inference with Shared Factors
  - Indexing Structures for Correlated Databases
- Ongoing and Future Work

1. Correlations in Uncertain Data

- Most application domains generate correlated data
  - Data Integration
    - Conflicting information best captured using “mutual exclusivity”
    - Data from the same source may all be valid or may all be invalid
  - Information extraction
    - Annotations on consecutive text segments strongly correlated
  - Social networks; Sensor networks
    - Attributes of neighboring nodes often highly correlated
    - Predicted links, class labels, extracted events likely to be correlated
- Even if base data exhibits independence..
  - Correlations get introduced during query processing

2. Shared Uncertainties and Correlations

- Uncertainties and correlations often specified for groups of tuples rather than for individual tuples
- Necessary when trying to model and reason about uncertainty in large populations

AdID	Model	Color	Price
1	Honda	?	$9,000
2	?	Beige	$8,000
3	?	?	$6,000
…	…	…	…
…	…	…	…
…	…	…	…
1000000	?	?	$10,000

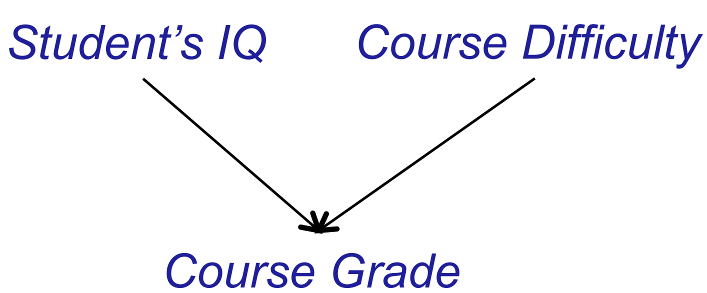
Model	Pr(M)
Honda	0.2
Mazda	0.1
…	…

Model	Color	Pr(C|M)
Honda	Beige	0.1
Honda	Red	0.2
…	…	…
Mazda	Beige	0.02

A Used Car Ads Database

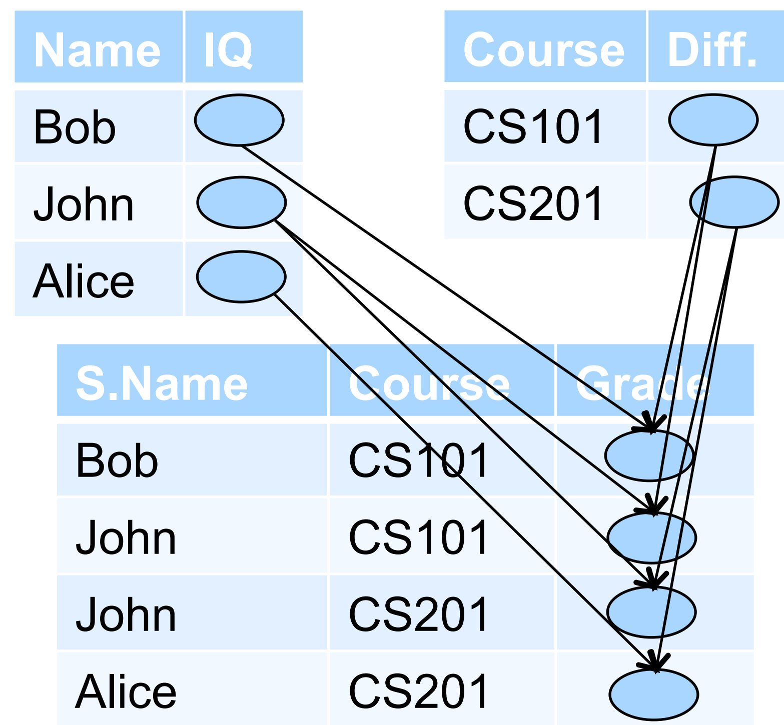
3. Schema-level Uncertainties

- Often we have probabilistic knowledge at the schema level (learned from a deterministic database) that we are trying to transfer
  - Using Probabilistic Relational Models (PRMs), Relational Markov networks (RMNs), Markov Logic Networks (MLNs) etc.



A "Schema-level" Dependence

An Instantiation



PrDB Framework

- Flexible uncertainty model (based on probabilistic graphical models)
  - Support for representing rich correlation structures [ICDE'07]
  - Support for specifying uncertainty at multiple abstraction levels [DUNE'07]
- Declarative constructs for interacting with the database
  - Manipulating and updating uncertainty as a first class citizen
- Rich querying semantics
  - SQL queries; Inference, reasoning, and what-if queries
- New techniques for scaling reasoning and query processing
  - Inference techniques to exploit the structure in the data [VLDB'08, UAI'09]
  - Index structures for handling large volumes of data [SIGMOD'09,'10]
  - Efficient algorithms for ranking queries, consensus answers [VLDB'09,PODS'09]

Outline

- Probabilistic Databases:Overview, Limitations
- PrDB: Overview
- PrDB: Some Details
  - Instance-optimal query execution
  - Inference with Shared Factors
  - Indexing Structures for Correlated Databases
- Ongoing and Future Work

A Simple Example

- Represent the uncertainties and correlations *graphically* using small functions called *factors*
  - Concepts borrowed from the *graphical models* literature

S	A	B	*prob*
s1	‘m’	1	0.6
s2	‘n’	1	0.5

T	B	C	*prob*
t1	1	‘p’	0.4

A Simple Example

- Represent the uncertainties and correlations *graphically* using small functions called *factors*
  - Concepts borrowed from the *graphical models* literature

0 = Tuple does not exist
1 = Tuple exists

S

	A	B	*prob*
s1	'm'	1	0.6
s2	'n'	1	0.5

T

	B	C	*prob*
t1	1	'p'	0.4

s1	***f_1(s1)***
0	0.4
1	0.6

Often not probability distributions
Values can be > 1

s2	***t1***	***f_2(s2, t1)***
0	0	0.1
0	1	0.5
1	0	0.4
1	1	0

s2 and t1 mutually exclusive

A Simple Example

- Represent the uncertainties and correlations *graphically* using small functions called *factors*
  - Concepts borrowed from the *graphical models* literature

S

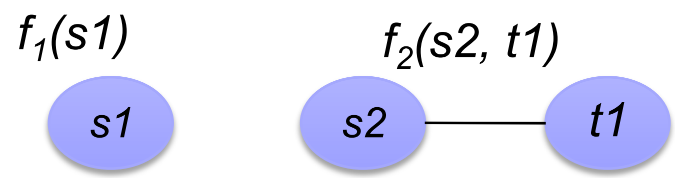
	A	B	prob
s1	'm'	1	0.6
s2	'n'	1	0.5

T

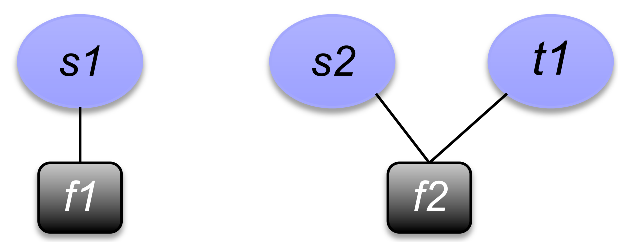
	B	C	prob
t1	1	'p'	0.4

s1	$f_1(s1)$
0	0.4
1	0.6

s2	*t1*	$f_2(s2, t1)$
0	0	0.1
0	1	0.5
1	0	0.4
1	1	0



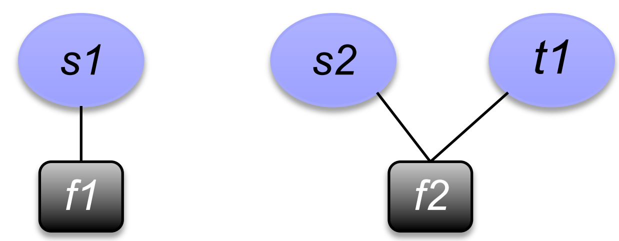
Markov network representation



Factor graphs

Probabilistic Graphical Models

- A PGM can compactly represent a joint probability distribution over a large number of random variables with complex correlations
- Specified completely by:
  - A set of random variables
  - A set of factors over the random variables
- Joint pdf obtained by multiplying all the factors and normalizing
- An *Inference* task: Finding a marginal prob. distribution over subset of variables
  - e.g. $Pr(t_1)$



$Pr(s_1, s_2, t_1) \propto f_1(s_1)\, f_2(s_2, t_1)$

<u>For example:</u>

$$Pr(s_1 = 0, s_2 = 0, t_1 = 0) = \frac{1}{Z} f_1(s_1 = 0)\, f_2(s_2 = 0, t_1 = 0)$$

Normalizing Constant

PrDB: Representation and Storage

- Underlying representation essentially a factor graph
  - Tuples and factors stored separately in different tables
- Factors can be inserted on any set of random variables
  - Corresponding to tuple existences or attribute values
- **Semantics**: the joint pdf over the random variables is obtained by multiplying all the factors and normalizing
  - No special care taken right now to ensure this is correct
- Allows specifying ***shared factors*** *that apply to groups of tuples, or to all tuples of a relation (schema-level)*

PrDB: Representation and Storage

insert into *S* ***values*** *('s1', 'm', 1)* ***uncertain****('f 0.2; t 0.8');*

insert into *T* ***values*** *('t1', uncertain, 'p');*
insert factor *'f 2 0.2; f 3 0.8; t 2 0.9; t 3 0.1'* ***in*** *S, T* ***on*** *'s1.e', 't1.B';*

S

tid	A	B	e
s1	'm'	1	*Π*
s2	'n'	1	*Π*

T

tid	B	C	e
t1	*Π*	'p'	***t***

fid	rv	pos
f1	s1.e	1
f2	s2.e	1
f3	s1.e	1
f3	t1.B	2

fid	funcid
f1	φ1
f2	φ1
f3	φ2

funcid	func
φ1	{[0] : 0.2, [1] : 0.8}
φ2	{[0, 2] : 0.2, [1, 3] : 0.8, [0, 2] : 0.9, [1, 3] : 0.1}

<u>**Data Tables**</u>

<u>**Uncertainty Parameters (factors)**</u>

PrDB: Representation and Storage

insert into *S* ***values*** *('s1', 'm', 1)* ***uncertain****('f 0.2; t 0.8');*

insert into *T* ***values*** *('t1', uncertain, 'p');*
insert factor *'f 2 0.2; f 3 0.8; t 2 0.9; t 3 0.1'* ***in*** *S, T* ***on*** *'s1.e', 't1.B';*

S

tid	A	B	e
s1	'm'	1	***Π***
s2	'n'	1	***Π***

T

tid	B	C	e
t1	***Π***	'p'	***t***

fid	rv	pos
f1	s1.e	1
f2	s2.e	1
f3	s1.e	1
f3	t1.B	2

fid	funcid
f1	φ1
f2	φ1
f3	φ2

funcid	func
φ1	{[0] : 0.2, [1] : 0.8}
φ2	{[0, 2] : 0.2, [1, 3] : 0.8, [0, 2] : 0.9, [1, 3] : 0.1}

Data Tables

Uncertainty Parameters (factors)

PrDB: Query Processing Overview

- Inference queries
  - Find marginal or conditional probability distributions over subsets of attributes

- Declarative SQL queries
  - PrDB supports a fairly large subset of SQL queries, including:
    - Select-project-join queries
    - Aggregates
    - Set operations (union, difference)

PrDB: Query Processing Overview

No Index on the Data

Load the base PGM into memory

Construct an augmented PGM [ICDE'07]

Use exact or approximate *lifted* inference [VLDB'08, UAI'09]

INDSEP Indexes Present

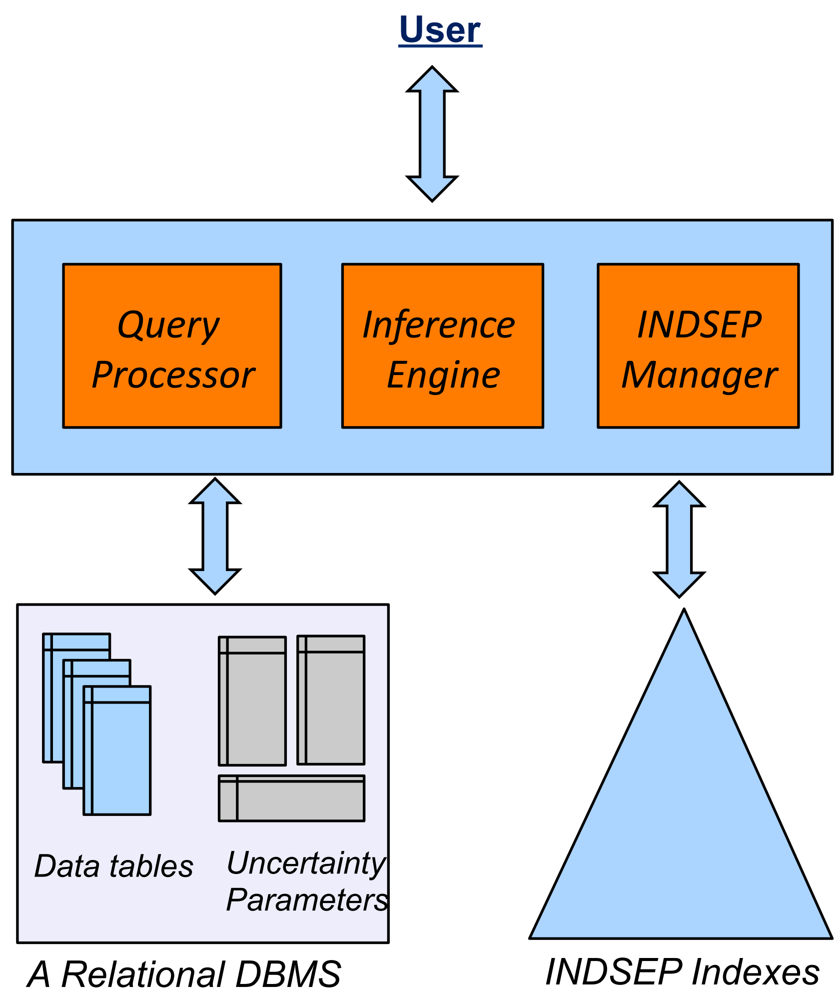
Aggregation or inference queries: Use index directly [SIGMOD'09]

SQL SPJ Queries [SIGMOD'10]

Gather a minimal set of correlations & uncertainties using the index

Use exact or approximate inference

In some cases, solve using the index



PrDB Overview

PrDB: Query Processing

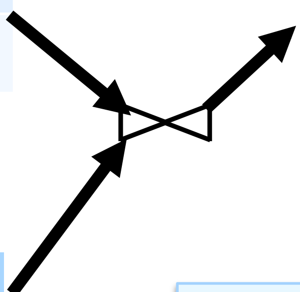
- During query processing, add new deterministic factors (hard constraints) corresponding to intermediate tuples
  - Encode the dependencies between base tuples and intermediate tuples
- Example query: $\pi_C(S \bowtie_B T)$

S

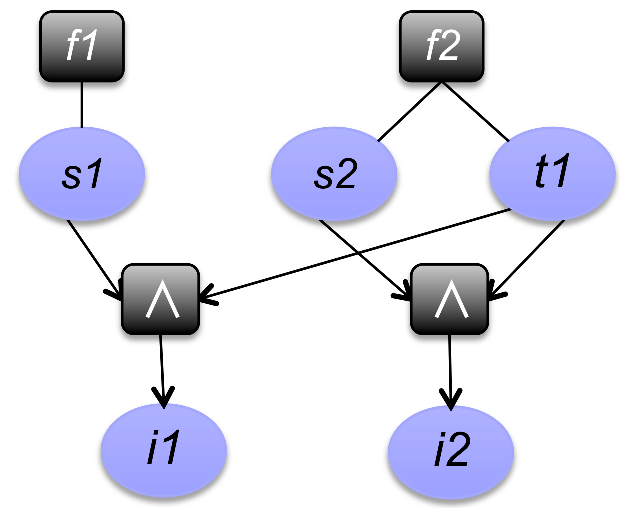
	A	B
s1	'm'	1
s2	'n'	1

T

	B	C
t1	1	'p'



	A	B	C
i1	'm'	1	'p'
i2	'n'	1	'p'



IF:	*s1 and t1 are 1*
THEN:	$Pr(i1 = 1) = 1, Pr(i1 = 0) = 0$
ELSE:	$Pr(i1 = 1) = 0, Pr(i1 = 0) = 1$

PrDB: Query Processing

- During query processing, add new deterministic factors (hard constraints) corresponding to intermediate tuples
  - Encode the dependencies between base tuples and intermediate tuples
- Example query: $\pi_C(S \bowtie_B T)$

S

	A	B
s1	'm'	1
s2	'n'	1

T

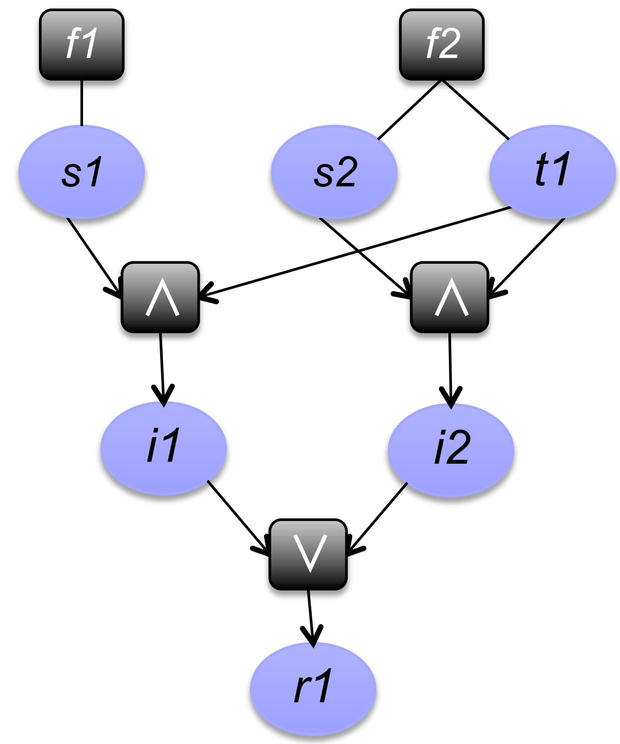
	B	C
t1	1	'p'

$\bowtie$

	A	B	C
i1	'm'	1	'p'
i2	'n'	1	'p'

π_C

	C
r1	'p'



PrDB: Query Processing

- Query evaluation ≡ Find the result tuple probabilities ≡ Inference !!
  - Can use standard techniques like *variable elimination, junction trees (exact), message passing, loopy Belief propagation, Gibbs Sampling (approx)*

S

	A	B
s1	'm'	1
s2	'n'	1

T

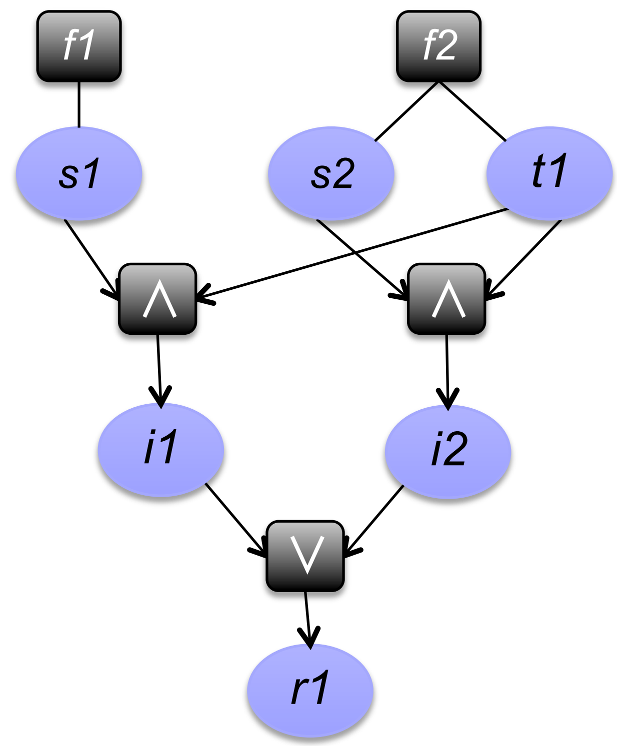
	B	C
t1	1	'p'

$\bowtie$

	A	B	C
i1	'm'	1	'p'
i2	'n'	1	'p'

π_C

	C
r1	'p'

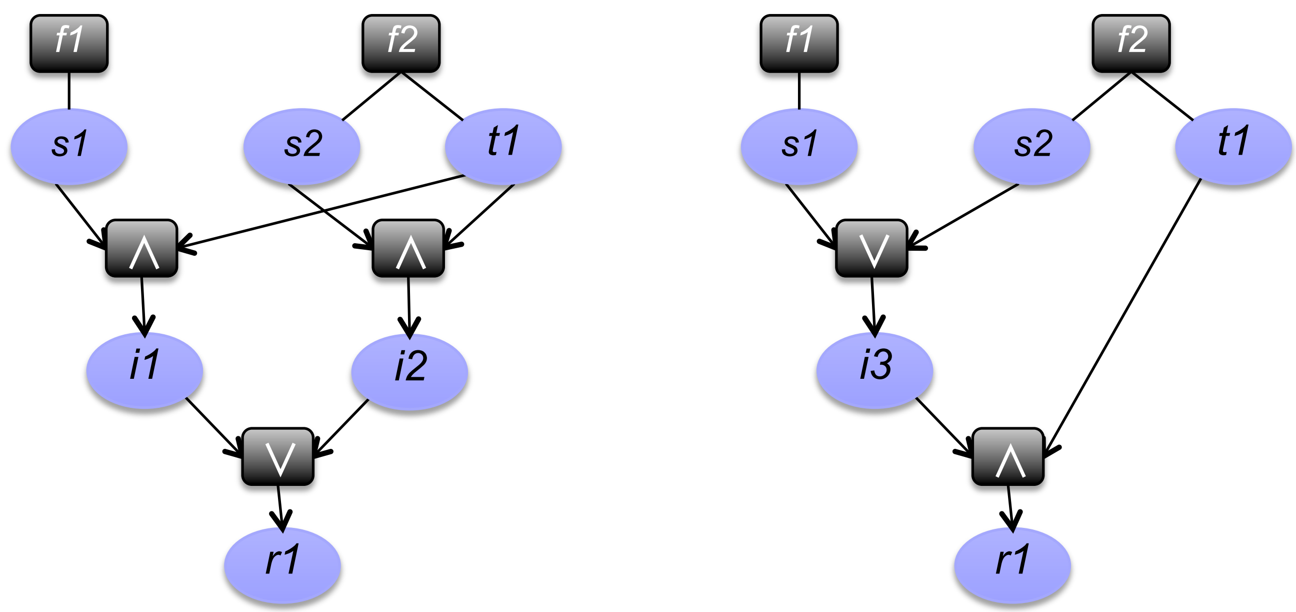


Outline

- Probabilistic Databases:Overview, Limitations
- PrDB: Overview
- PrDB: Some Details
  - Instance-optimal query execution
  - Inference with Shared Factors
  - Indexing Structures for Correlated Databases
- Ongoing and Future Work

1. Instance-optimal Query Execution

- AND and OR factors enable reorganization of the network
  - Complexity of the generated network depends on the query plan
    - "Safe plans" always generate tree networks – enabling extensional evaluation
  - But a reorganization may not necessarily correspond to a traditional query plan
    - Benefits in looking for optimal reorganization for a given query and dataset
  - We designed an efficient algorithm to find such reorganizations during query execution in some cases, but many problems still open [VLDB'10]



2. Inference with Shared Factors

AdID	Model	Color	Price
1	Honda	?	$9,000
2	?	Beige	$8,000
3	?	?	$6,000
…	…	…	…
…	…	…	…
…	…	…	…
1000000	?	?	$10,000

Model	Pr(M)
Honda	0.2
Mazda	0.1
…	…

Model	Color	Pr(C|M)
Honda	Beige	0.1
Honda	Red	0.2
…	…	…
Mazda	Beige	0.02

Query: How many "red" cars are for sale ?

- Option 1: "Ground out" (propositionalize) the random variables, and use standard techniques
- Option 2: Directly operate on the shared factors

2. Inference with Shared Factors

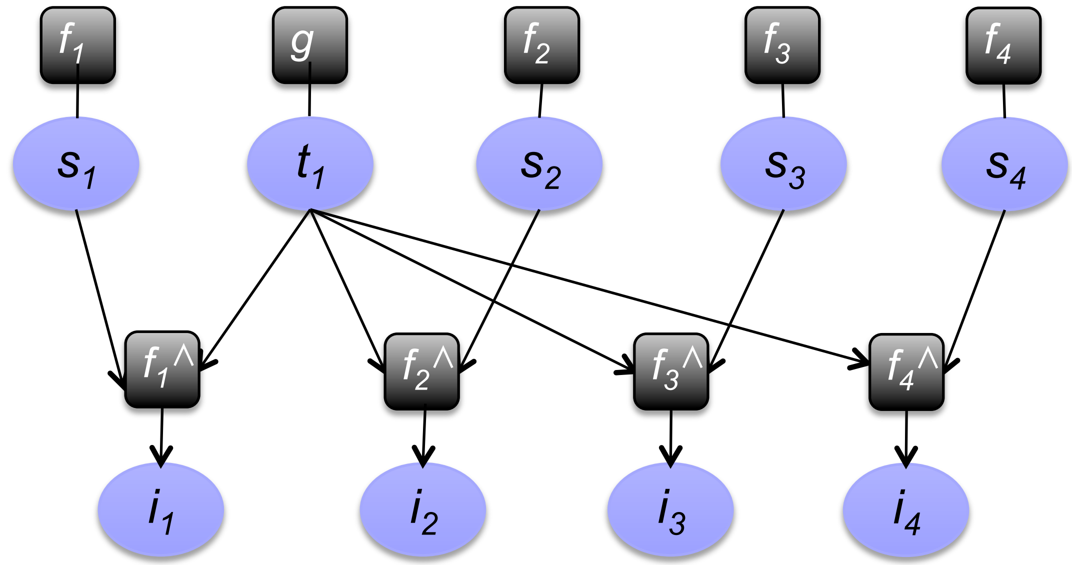
s1	*f_1(s1)*
0	0.2
1	0.8

s2	*f_2(s2)*
0	0.2
1	0.8

s3	*f_3(s3)*
0	0.4
1	0.6

s4	*f_4(s4)*
0	0.21
1	0.79

t1	*g(t1)*
0	0.5
1	0.5



2. Inference with Shared Factors

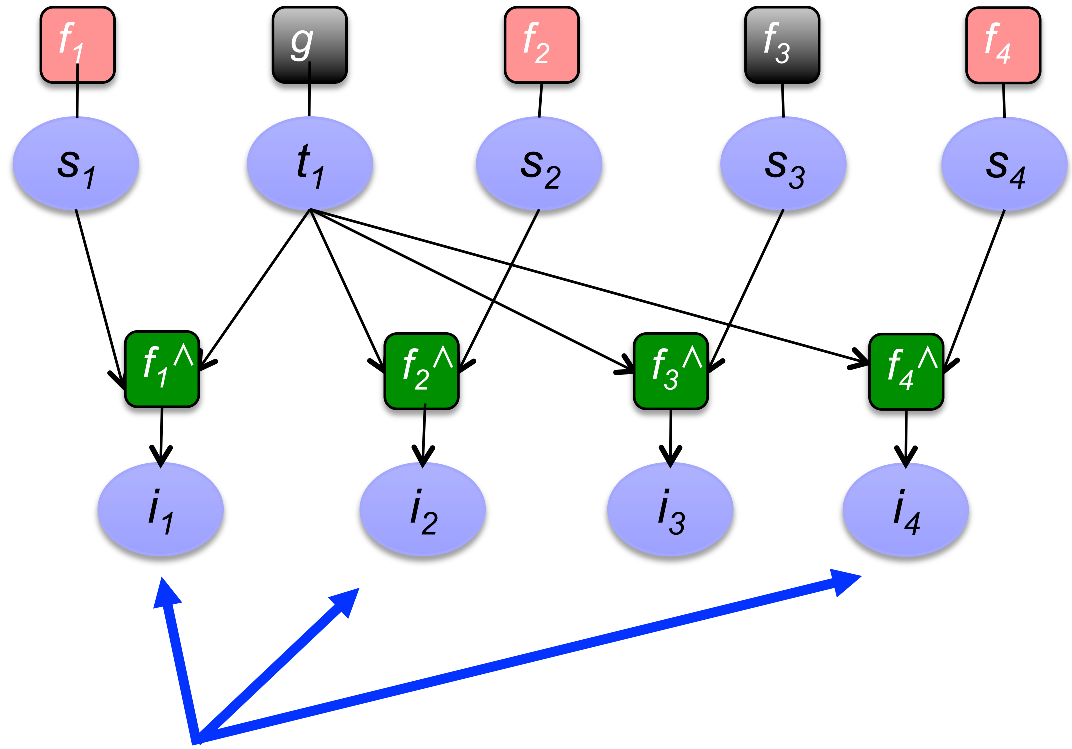
s1	f_1(s1)
0	0.2
1	0.8

s2	f_2(s2)
0	0.2
1	0.8

s3	f_3(s3)
0	0.4
1	0.6

s4	f_4(s4)
0	0.21
1	0.79

t1	g(t1)
0	0.5
1	0.5



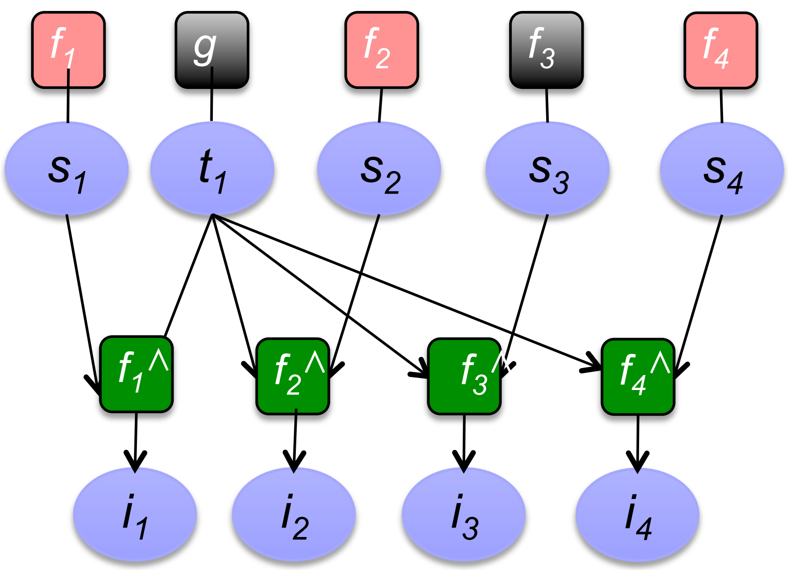
(Near-)identical answers because of the symmetry

How to identify such opportunities in general ?

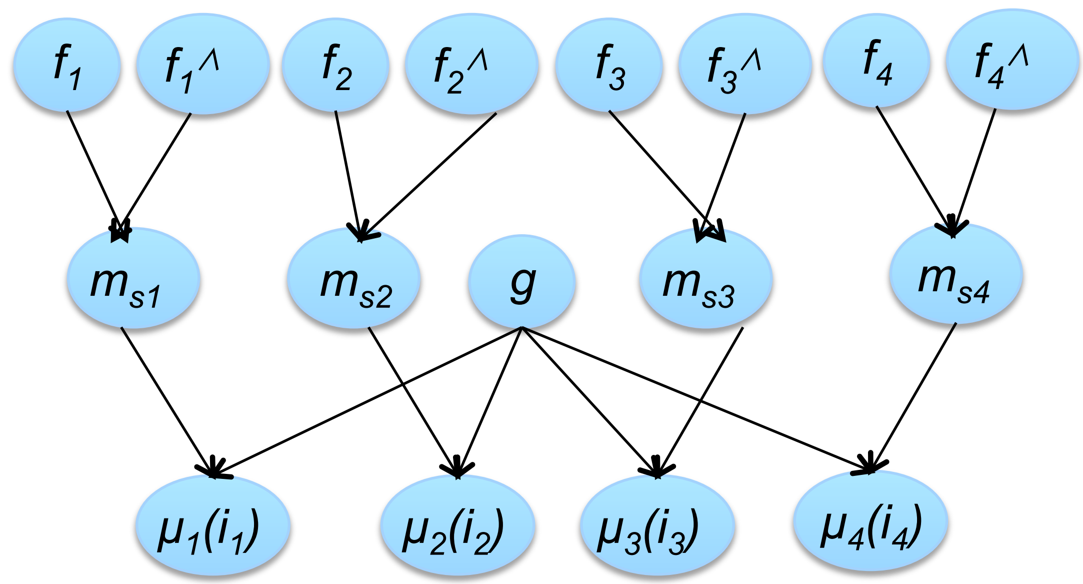
2. Bisimulation-based Lifted Inference

Step 1: Capture a (simulated) run of variable elimination as a graph

Graphical Model



RV-Elim Graph



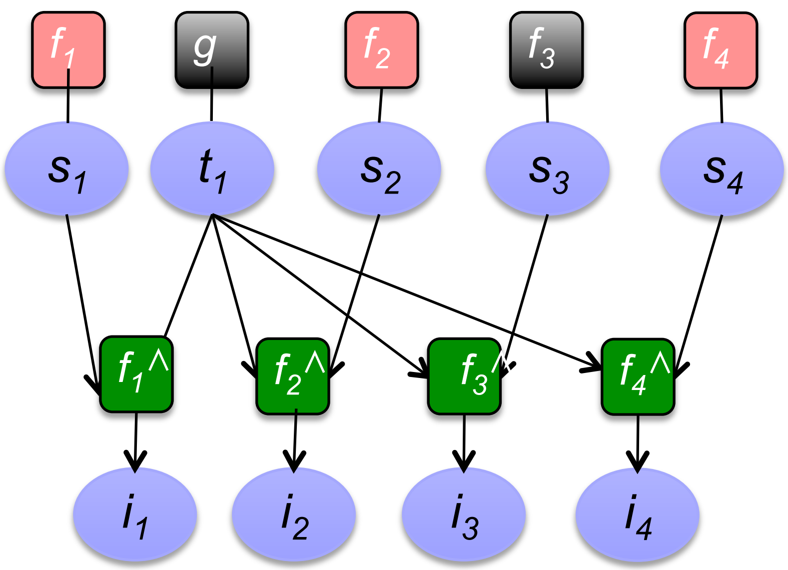
Elimination Order:

s1, s2, s3, s4, t1

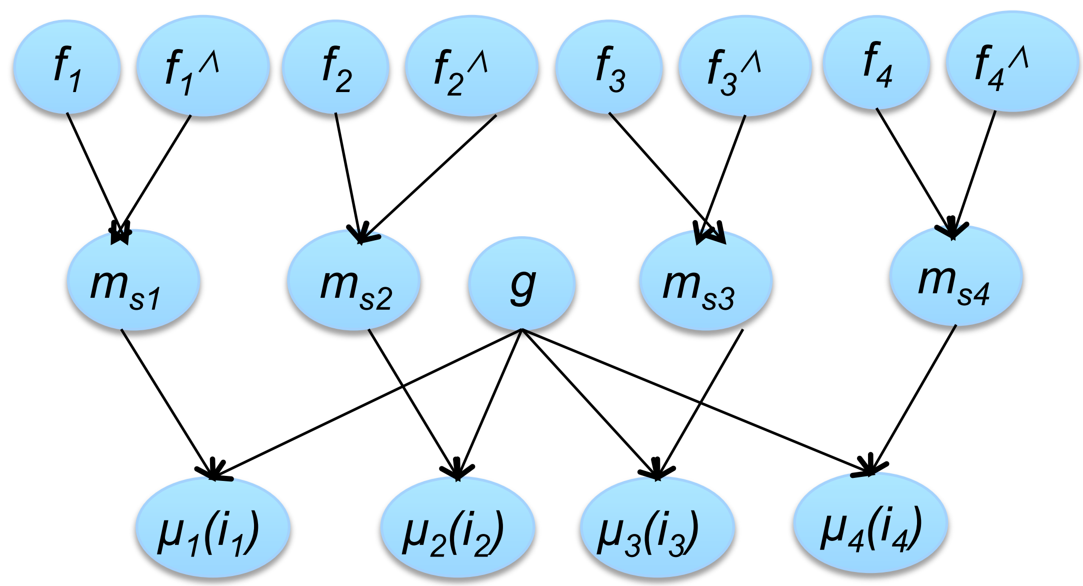
2. Bisimulation-based Lifted Inference

Step 2: Run *bisimulation* on the RV-Elim graph to identify symmetries

Graphical Model



RV-Elim Graph

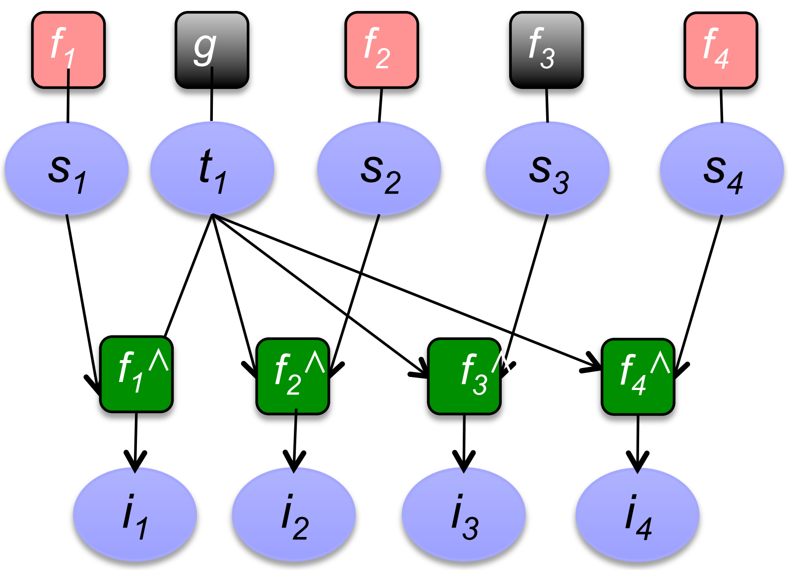


Intuitively, two nodes are bisimilar if
(1) they represent identical factors, and
(2) their parents are identically colored

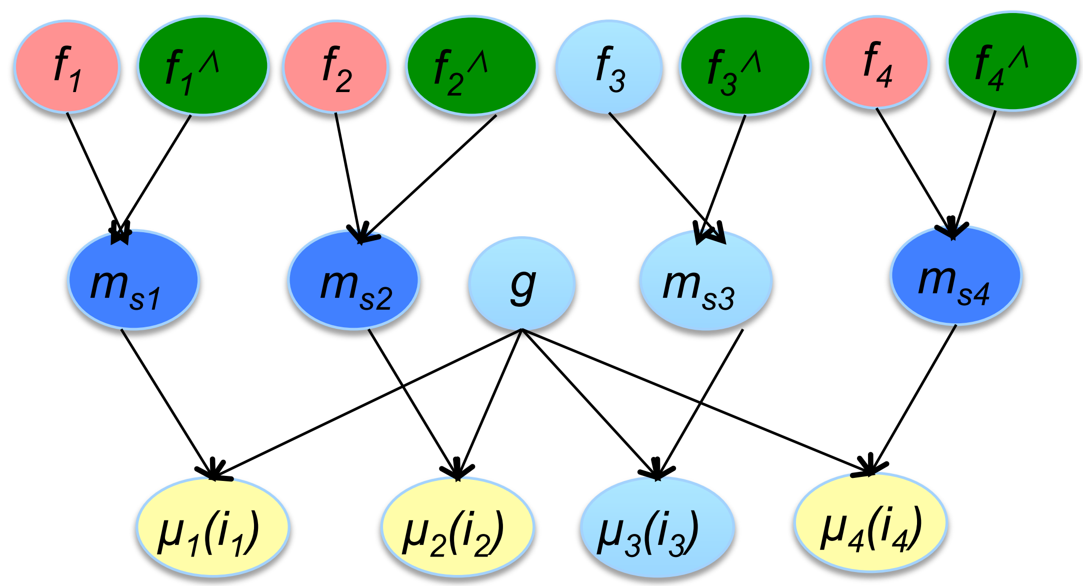
2. Bisimulation-based Lifted Inference

Step 2: Run *bisimulation* on the RV-Elim graph to identify symmetries

Graphical Model



RV-Elim Graph

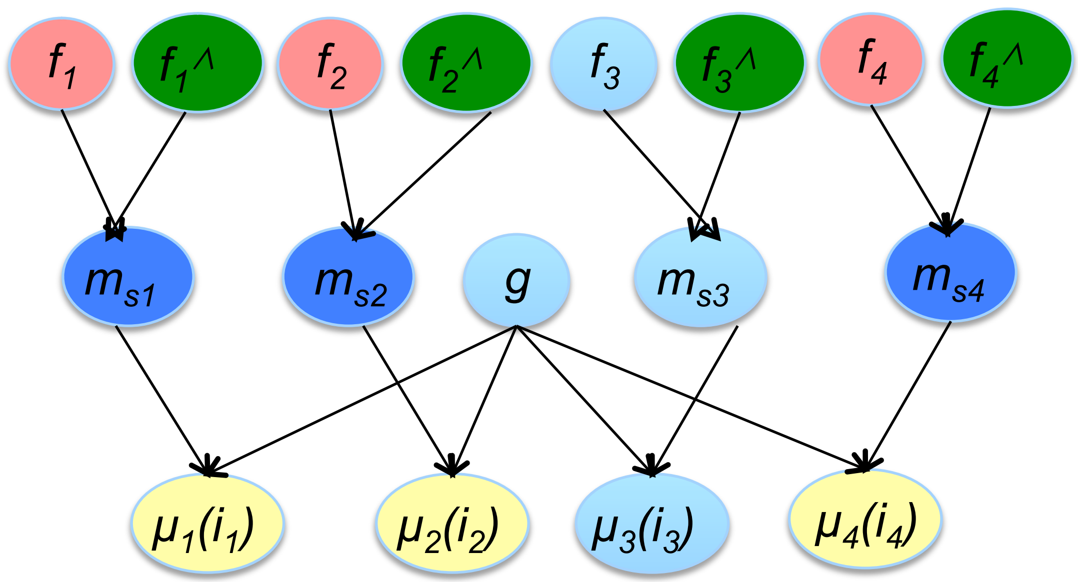


Intuitively, two nodes are bisimilar if
(1) they represent identical factors, and
(2) their parents are identically colored

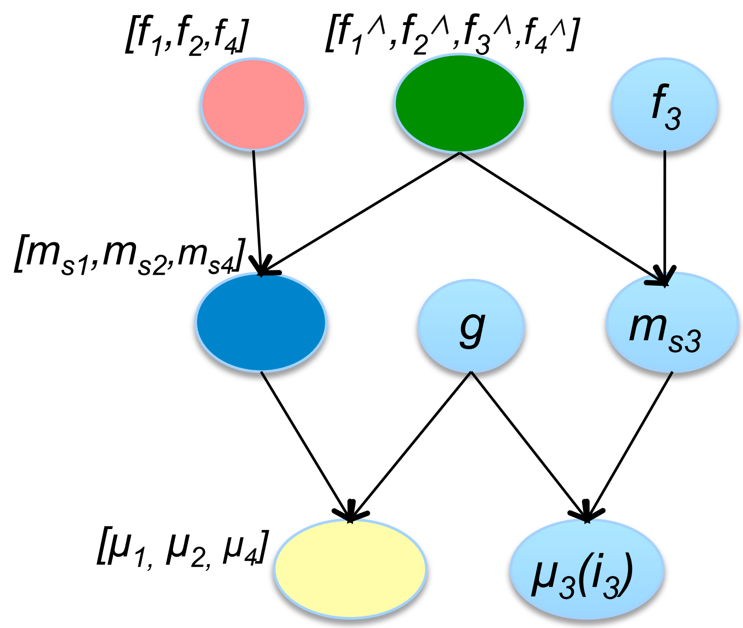
2. Bisimulation-based Lifted Inference

Step 3: Compress the RV-Elim graph; run inference on compressed graph

RV-Elim Graph

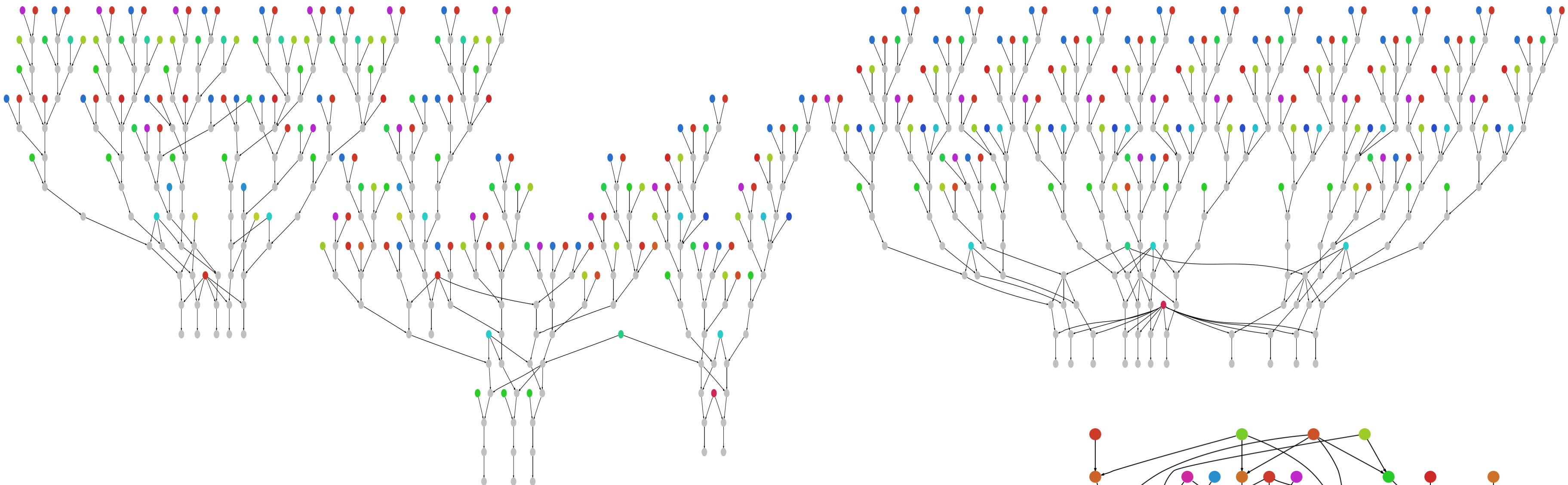


Compressed RV-Elim Graph



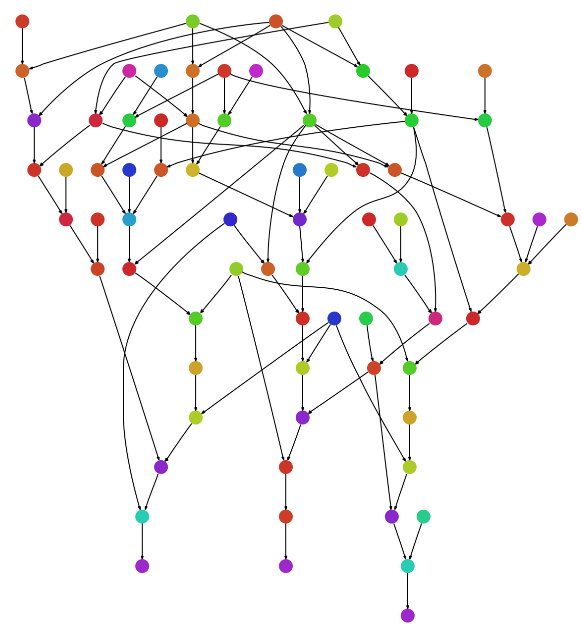
2. Example

[[3 relation join with 3 tuples each, attribute and tuple uncertainty]]



Original RV-Elim graph, 1170 vertices

Compressed RV-Elim graph, 78 vertices



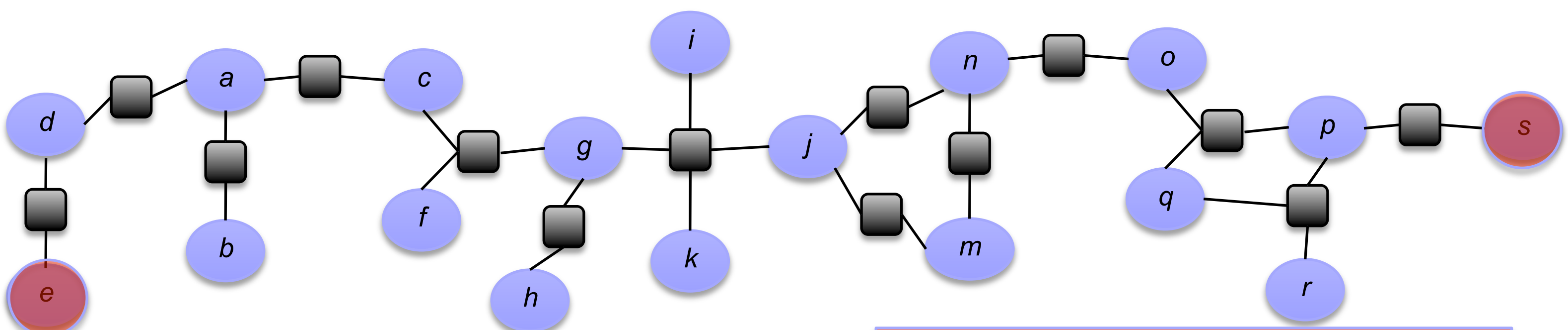
2. Bisimulation-based Lifted Inference

- Orders of magnitude performance improvements with symmetry
- Bisimulation can be done in linear time on DAGs
  - Somewhat more involved here
    - Need to keep track of the order in which factors were multiplied
    - Must construct labels on-the-fly as opposed to standard bisimulation
  - $O(|E| \log(D) + |V|)$
- Choice of elimination order crucial
  - Dictates the amount of compression possible
  - We choose it by running bisimulation on the graphical model itself
- Our technique works on the ground (propositionalized) model
  - Enables approximations: e.g. allow approximate matches on factors [UAI'09]
- Many open challenges in effectively exploiting symmetry and first order representations

3. Querying Very Large CPDBs

- Base representation of PGMs can't handle large datasets
  - Queries may only reference a small set of variables
    - Still may need to touch the entire dataset
  - Infeasible to load into memory and operate upon the full PGM

An example PGM



Queries of interest

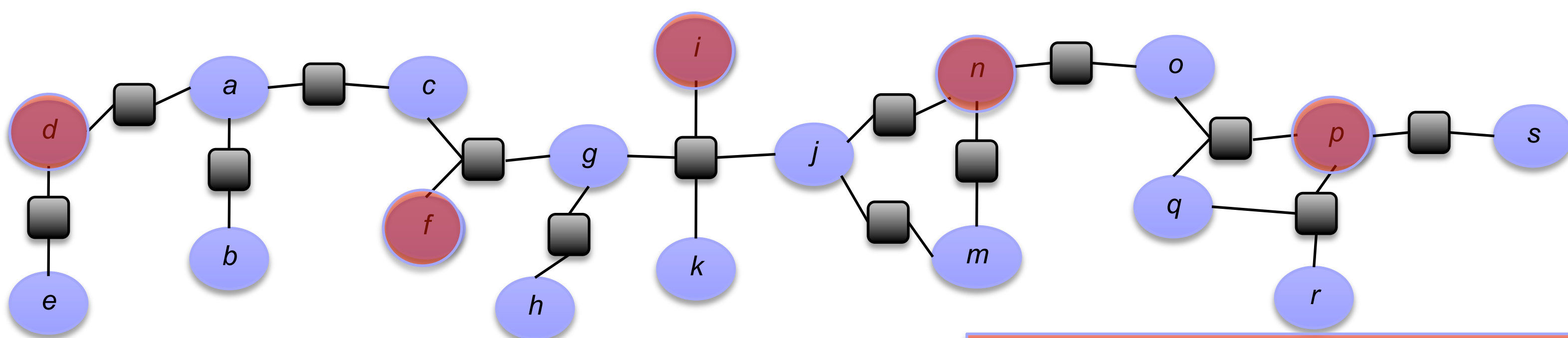
Q1: Need to do an inference operation involving nearly all variables

Q1: How does the value of "s" affect the value "e" ?

3. Querying Very Large CPDBs

- Base representation of PGMs can't handle large datasets
  - Queries may only reference a small set of variables
    - Still may need to touch the entire dataset
  - Infeasible to load into memory and operate upon the full PGM

An example PGM



Q2: Must compute a potentially large probability distribution: Pr(d, i, f, n, p)

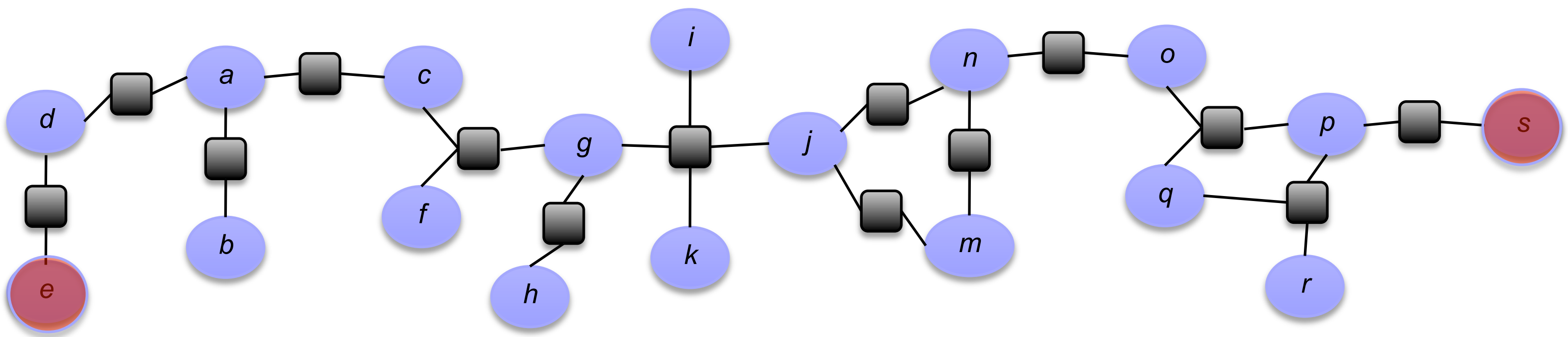
Queries of interest

Q1: How does the value of "s" affect the value "e" ?

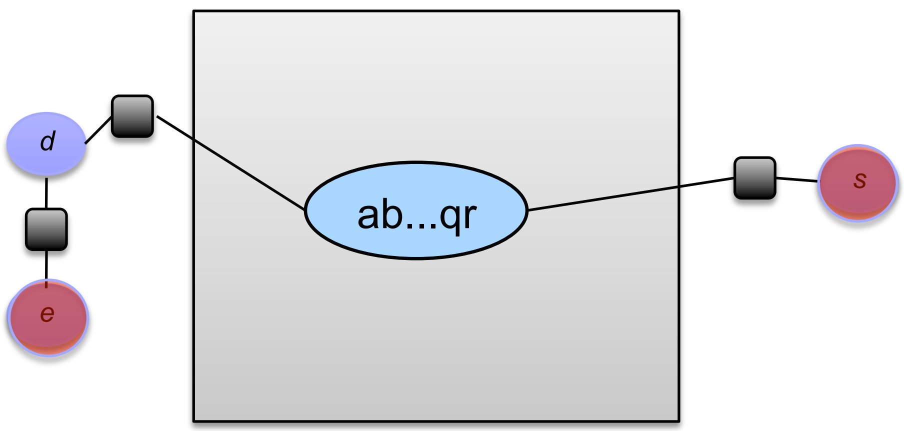
Q2: Compute probability distribution of "d + i + f + n + p"

3. Key Insight

Original PGM



What if we could "shortcut" the in-between nodes ?

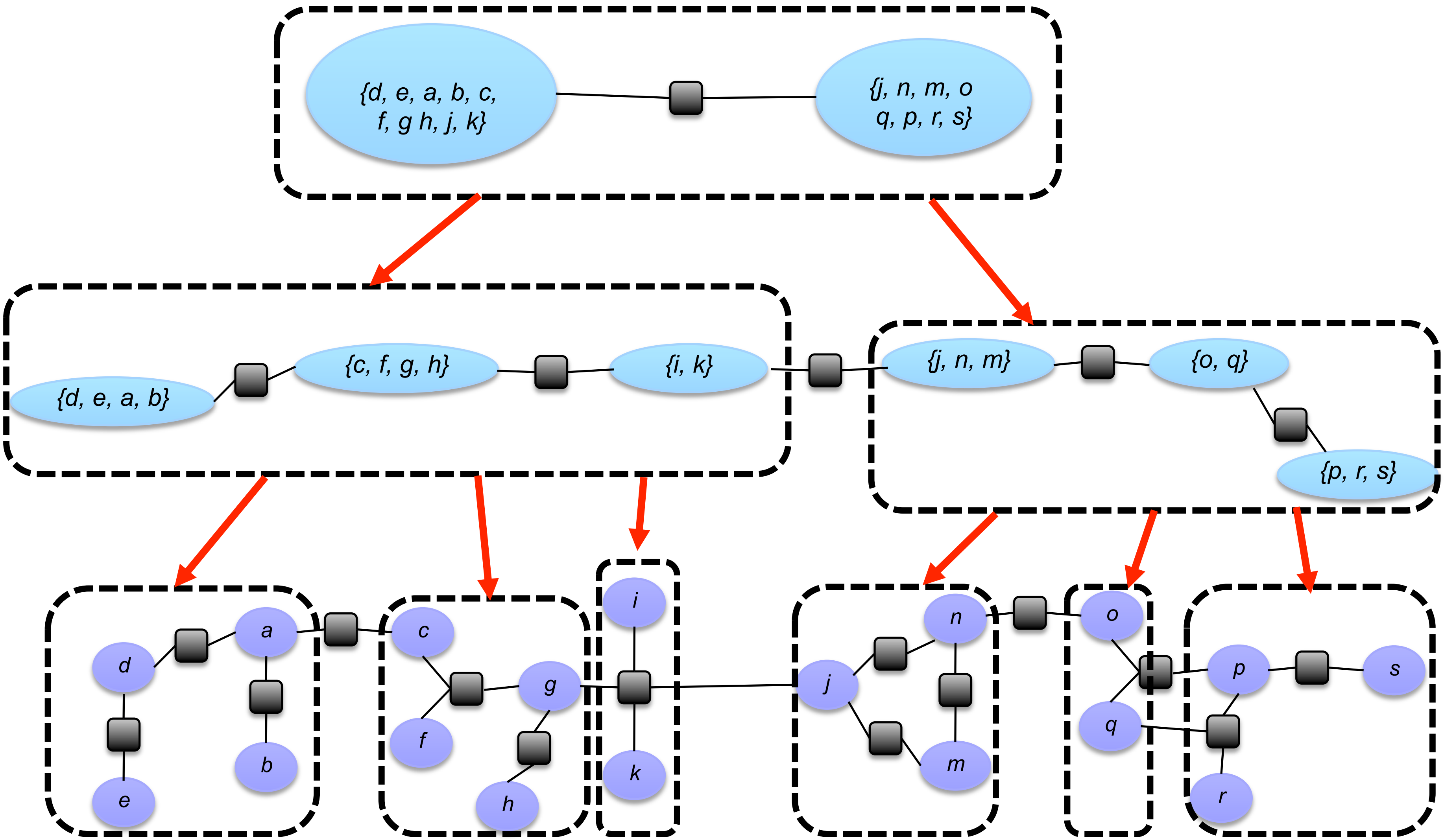


Many fewer computations

Can do inference much faster

3. INDSEP

- INDSEP is a hierarchical data structure based on this idea



3. INDSEP: Overview

- Unclear how to do this on the graphical model directly
- Instead we work with a *junction tree* of the model
  - Caveat: Inherit the limitations of the junction tree approach – only works for models with *bounded treewidth*
- Very large speedups for *inference queries,* and for *decomposable aggregate functions (like SUM, MAX)*
  - Evaluating boolean formulas trickier, but still significant benefits
- Supports a lazy approach for updates
  - Future queries inherit the burden of updating the index

Outline

- Probabilistic Databases:Overview, Limitations
- PrDB: Overview
- PrDB: Some Details
  - Instance-optimal query execution
  - Inference with Shared Factors
  - Indexing Structures for Correlated Databases
- Ongoing and Future Work

Ongoing Work and Open Problems

- Better connections with the work in the ML community
  - Many ML problems and application domains ideal use cases for probabilistic databases
    - Need to scale to large (relational) databases
    - Need support for rich querying over uncertain data
  - Significant overlap in the tools and techniques being developed
  - But many important differences
    - Learning and knowledge transfer equally important there
    - Not much work in the probabilistic database community

Ongoing Work and Open Problems

- Language constructs and semantics
  - Flexibility in specifying uncertainties at different abstraction levels results in significant interpretation issues
  - *How to resolve conflicting uncertainties ?*
  - *How to keep the semantics simple enough that users can make sense of it ?*

- Efficient algorithms for lifted inference
  - Much work in recent years, but many interesting open problems remain

Ongoing Work and Open Problems

- Querying very large correlated probabilistic databases
  - Our indexing structures inherit the limitations of junction trees
    - Can only handle datasets or queries with low treewidths
  - *How to incorporate approximations into the framework ?*
  - *Lineage formula probability computation especially hard*
    - Computing probabilities of *read-once* lineages easy with tuple independence, but #P-Hard for simplest of correlations

- Uncertain graph data
  - Shared correlations prevalent in settings like social networks, biological networks
  - Compact models of correlations required

Thank You !!

- More details at:

  http://www.cs.umd.edu/~amol/PrDB