Anytime Approximation in Probabilistic Databases via Scaled Dissociations

SIGMOD 2019

Maarten Van den Heuvel
U of Antwerp

Peter Ivanov
Northeastern U

Wolfgang Gatterbauer*
Northeastern U

Floris Geerts
U of Antwerp

Martin Theobald
U of Luxemburg

Team

Maarten Van den Heuvel

Peter Ivanov

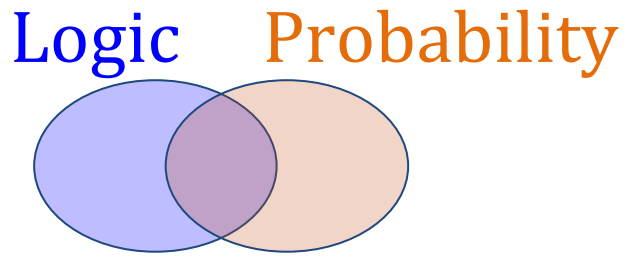
Wolfgang Gatterbauer*

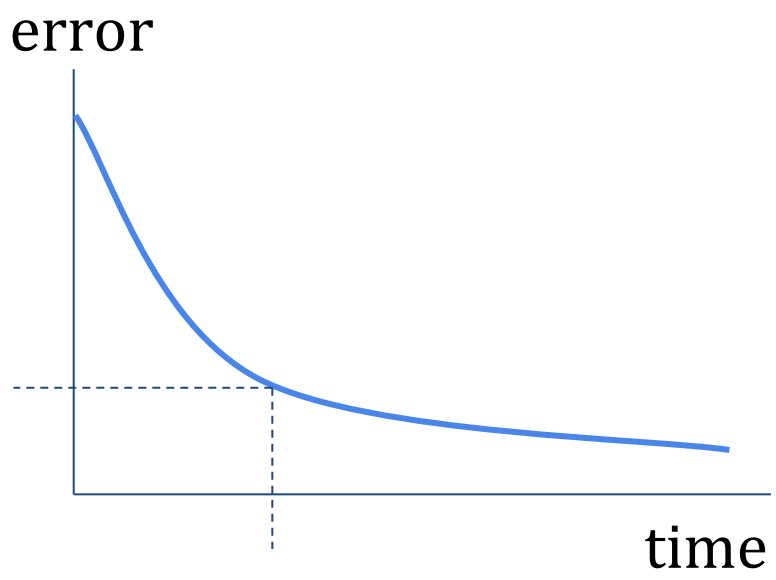
Floris Geerts

Martin Theobald

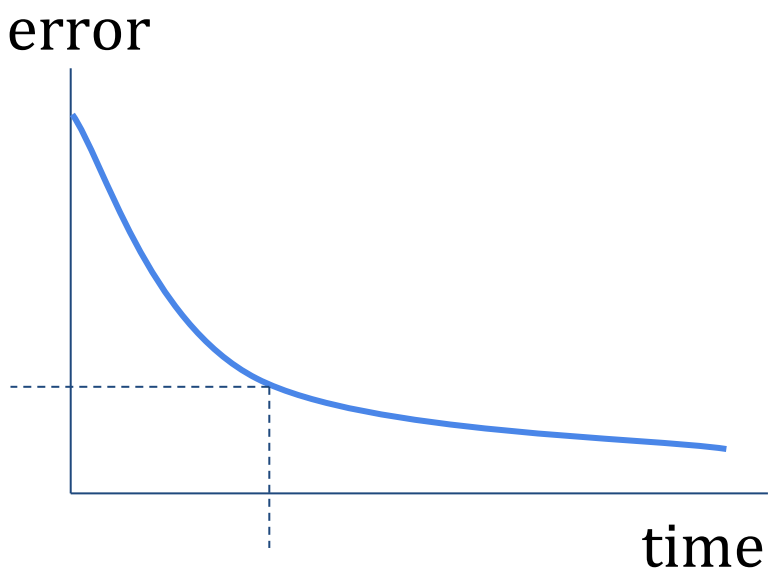
Probabilistic inference

- key algorithmic problem in probabilistic AI
  - e.g. Probabilistic Graphical Models (PGMs)
  - e.g. Statistical Relational Learning (SRL)
  - e.g. Probabilistic Databases (PDBs)
- ☹ well-known to be #P-hard
  - either identify tractable cases
  - or find approximations
- ☺ Anytime approximation framework
  - create more and more refined bounds
  - until you get a certain error guarantee





Probabilistic inference



- Deterministic anytime approximation
  - returning guaranteed upper and lower bounds
- Good bounds are important
  - prior work uses model-based (MB) bounds
- ? Problem
  - exponentially many bounds to choose from (they are quite different)
  - how to choose bounds (perhaps even better ones)?

- Our approach: scaled dissociations
  - Embed the combinatorial model-based space of lower bounds within a continuous enlarged space, then use continuous optimization

Agenda

1. Probabilistic inference
   - Boolean formulas, anytime approximations

2. Better bounds
   - how to find better bounds than "model-based" bounds

3. Experiments & Take-aways

Not discussed (please see paper or stop by at the poster)
- Probabilistic databases → Lineage
- How to select variable to decompose, and leaves to expand
- Various technical details

Probabilistic inference: when it is easy

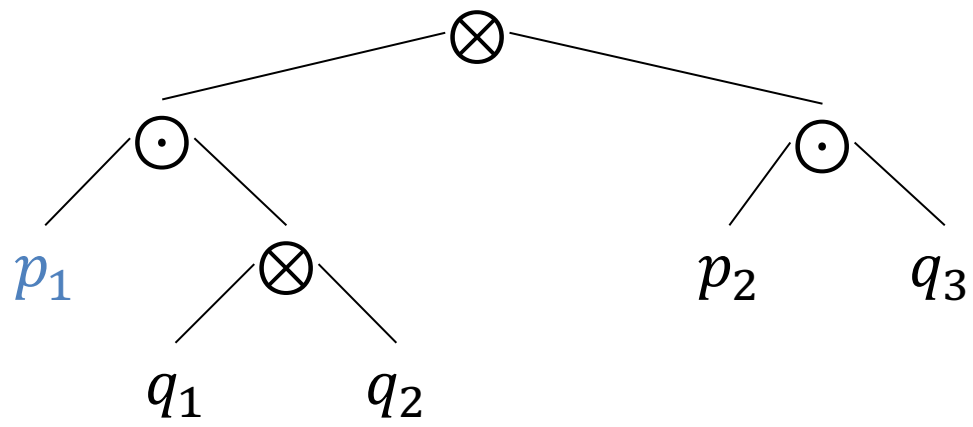
$$\varphi = x_1y_1 \vee x_1y_2 \vee x_2y_3$$

$$= \underbrace{x_1(y_1 \vee y_2)}_{\varphi_1} \vee \underbrace{x_2y_3}_{\varphi_2}$$

$\varphi_1 \perp \varphi_2$

$$\mathbb{P}[\varphi] = \mathbb{P}[x_1(y_1 \vee y_2)] \otimes \mathbb{P}[x_2y_3]$$

$$= (p_1 \odot (q_1 \otimes q_2)) \otimes (p_2 \odot q_3)$$



Read-once (RO) expression

$$\Rightarrow \mathbb{P}[\varphi_1 \vee \varphi_2] = 1-(1-\mathbb{P}[\varphi_1])(1-\mathbb{P}[\varphi_2])$$

$$= \mathbb{P}[\varphi_1] \otimes \mathbb{P}[\varphi_2]$$

"independent-or"

$\mathbb{P}[x_i] = p_i \quad \mathbb{P}[y_j] = q_j$

parse tree

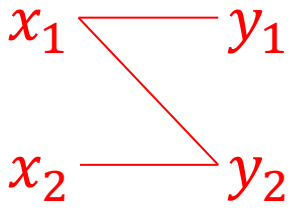
expression that allows us to calculate $\mathbb{P}[\varphi]$

5 leaf nodes (5 variables)

Probabilistic inference: when it is hard

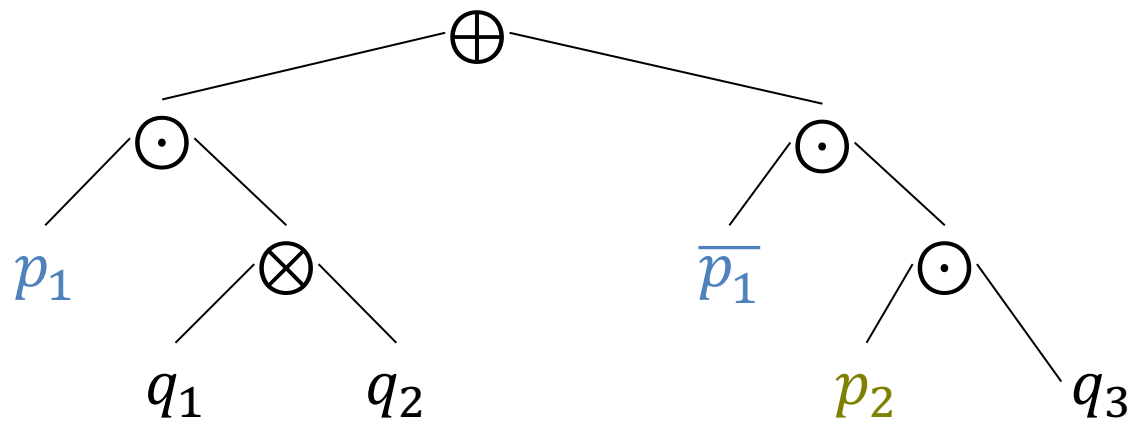
$$\varphi = x_1y_1 \vee x_1y_2 \vee x_2y_2$$
$$= x_1y_1 \vee y_2(x_1 \vee x_2)$$

P4



$$\mathbb{P}[\varphi] = p_1 \odot \mathbb{P}[\varphi[1/x_1]] \oplus \overline{p_1} \odot \mathbb{P}[\varphi[0/x_1]]$$
$$= p_1 \odot (q_1 \otimes q_2) \oplus \overline{p_1} \odot (p_2 \odot q_3)$$

"Decomposition"
with Shannon expansion
(total probability theorem)

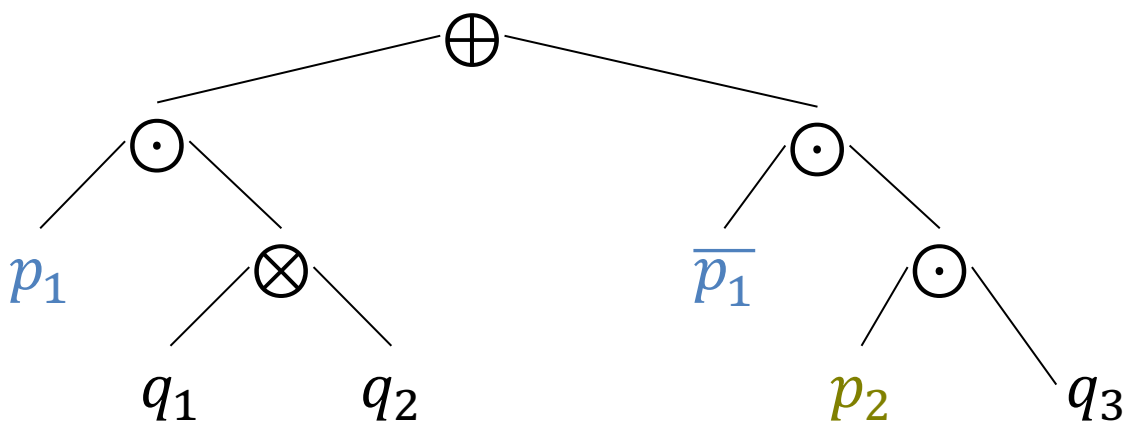


parse tree

expression that allows us to calculate $\mathbb{P}[\varphi]$

6 leaf nodes (4 variables)

Anytime compilation of Boolean formulas



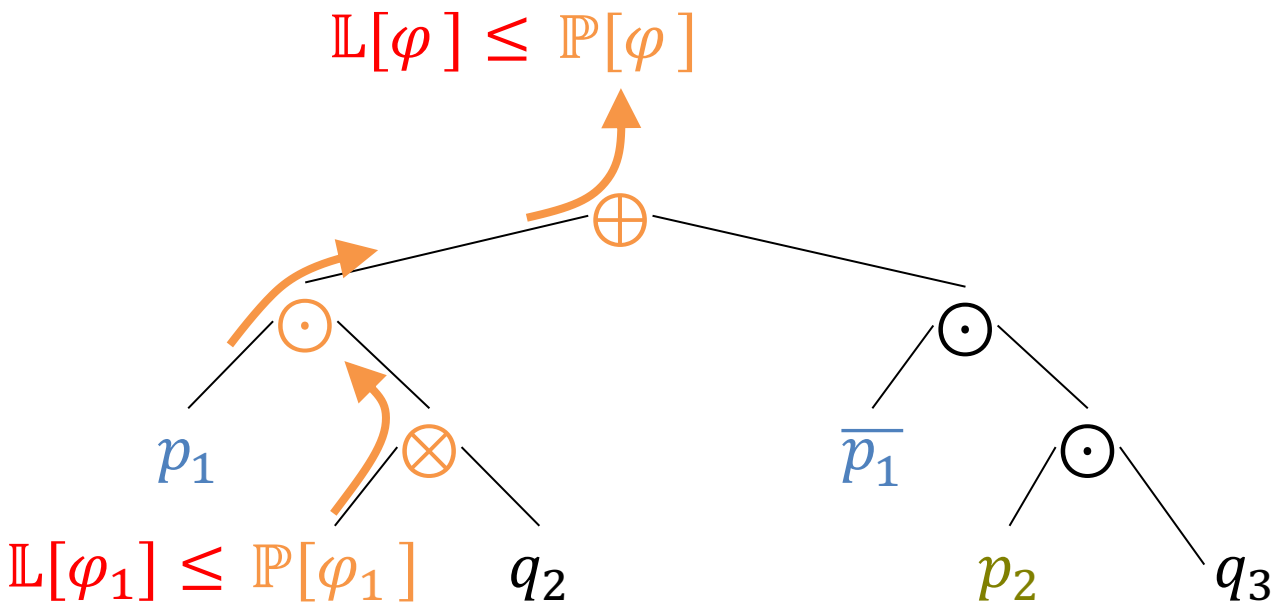
parse tree

Olteanu, Huang, Koch [ICDE'10]

Fink, Olteanu [ICDT'11]

Fink, Huang, Olteanu [VLDBJ'13]

Anytime compilation of Boolean formulas



1. parse tree is monotone

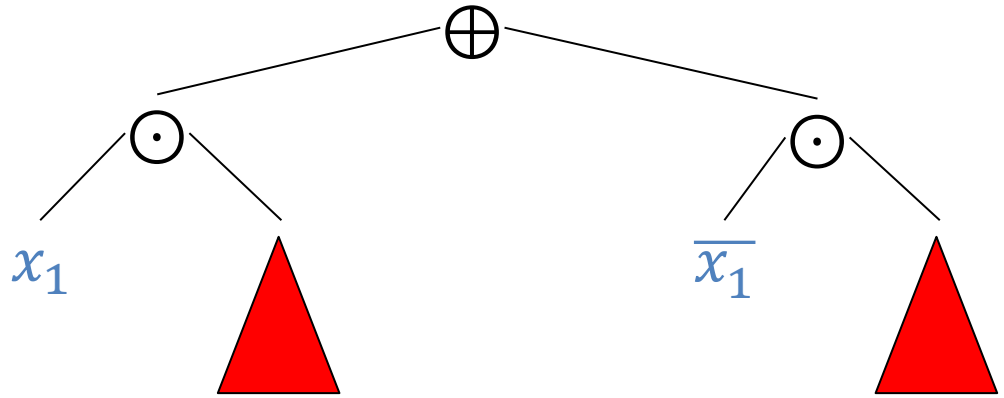
   ⇒ lower and upper bounds propagate to the root

Olteanu, Huang, Koch [ICDE'10]

Fink, Olteanu [ICDT'11]

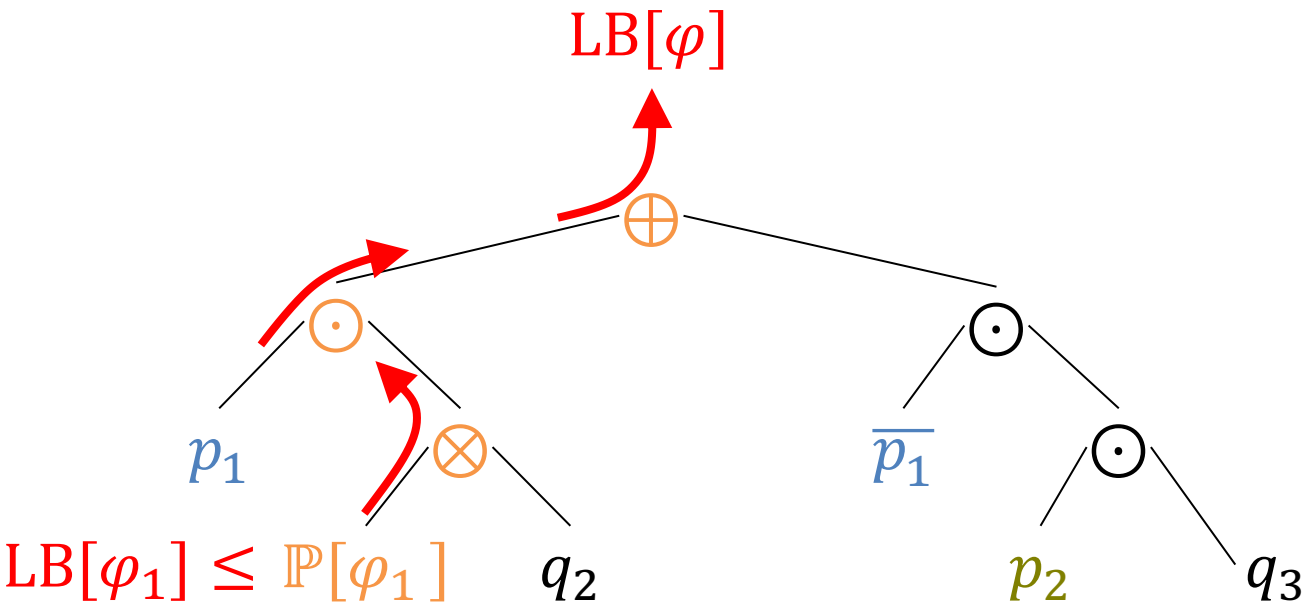
Fink, Huang, Olteanu [VLDBJ'13]

Anytime compilation of Boolean formulas



2. grow partial "d-trees" ("decomposition tree")

   ⇒ try to bound early; continue if too lose



1. parse tree is monotone

   ⇒ lower and upper bounds propagate to the root

Olteanu, Huang, Koch [ICDE'10]

Fink, Olteanu [ICDT'11]

Fink, Huang, Olteanu [VLDBJ'13]

But how do we get the bounds?

That's where the magic happens.

Model-based bounds (MBs)

- Model-based bounds (MBs):
  - Intuition: replace repeated variables with 0 or 1 to make φ read-once

$$\varphi = x_1y_1 \vee x_1y_2 \vee x_2y_2$$ e.g., replace 2nd instance of x_1 with 1 (True)

$$\varphi_U = x_1y_1 \vee y_2 \vee \cancel{x_2y_2}$$ result is simpler and upper bound

- Remaining problem
  - How to choose from d^n options? Each may lead to very different bounds.
    - assuming n variables repeated, each with d repetitions
    - E.g. we encountered formulas with $n = 1225$ and $\text{AVG}(d) \approx 5.6$
  - Prior work chooses randomly

Fink, Olteanu [ICDT'11]

Fink, Huang, Olteanu [VLDBJ'13]

Agenda

1. Probabilistic inference
   - Boolean formulas, anytime approximations

2. Better bounds
   - how to find better bounds than "model-based" bounds

3. Experiments & Take-aways

Oblivious Bounds for Monotone Boolean functions

PROBLEM

G., Suciu [TODS'14]

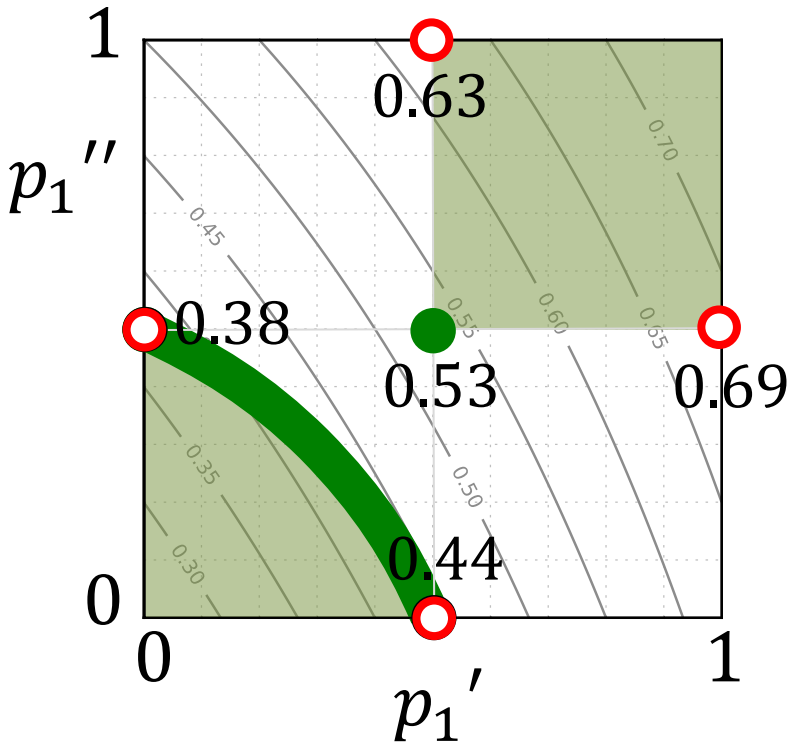
Given: $\varphi = \varphi_1 \vee \varphi_2$

Replace it with: $\varphi' = \varphi_1[x'/x] \vee \varphi_2[x''/x]$ (let's call it dissociation)

How to choose p' and p'' s.t. we get a lower bound $\mathbb{P}[\varphi'] \leq \mathbb{P}[\varphi]$ (or upper bound)

RESULT

Opt. Oblivious bounds:	Upper	$p'=p, p''=p$
	Lower	$(1-p')(1-p'')=1-p$
Model-based bounds:	Upper	$p'=p, p''=1$
	Lower	$p'=p, p''=0$

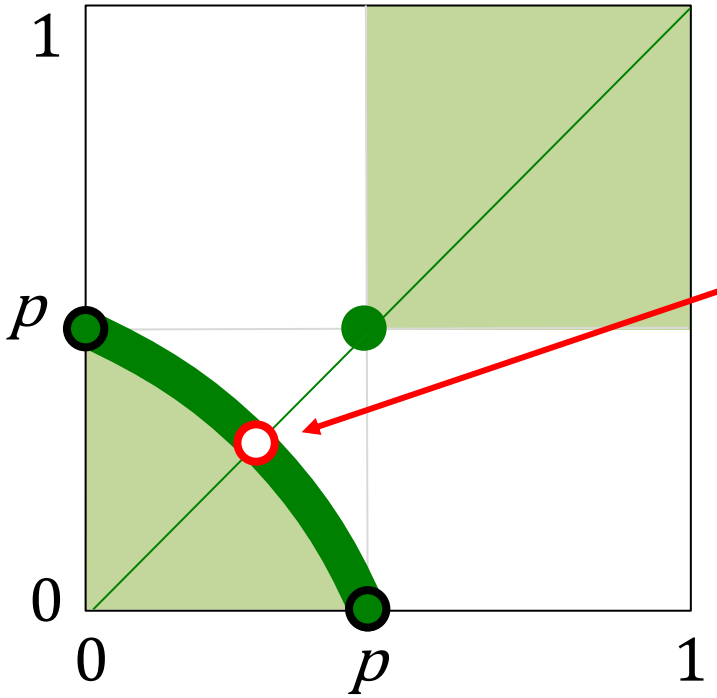


EXAMPLE

$\varphi = x_1'y_1 \vee x_1''y_2 \vee x_2y_2$

and all probabilities are 0.5 Then $\mathbb{P}[\varphi]=0.5$

Lower bounds by default are not good



used "symmetric dissociations" (SDs)

lower bounds were very bad

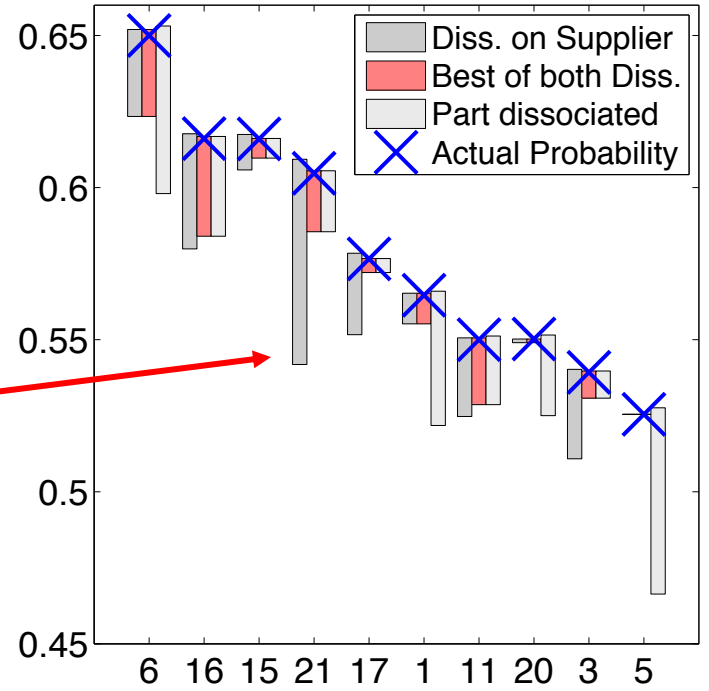


Figure from G., Suciu [TODS'14]

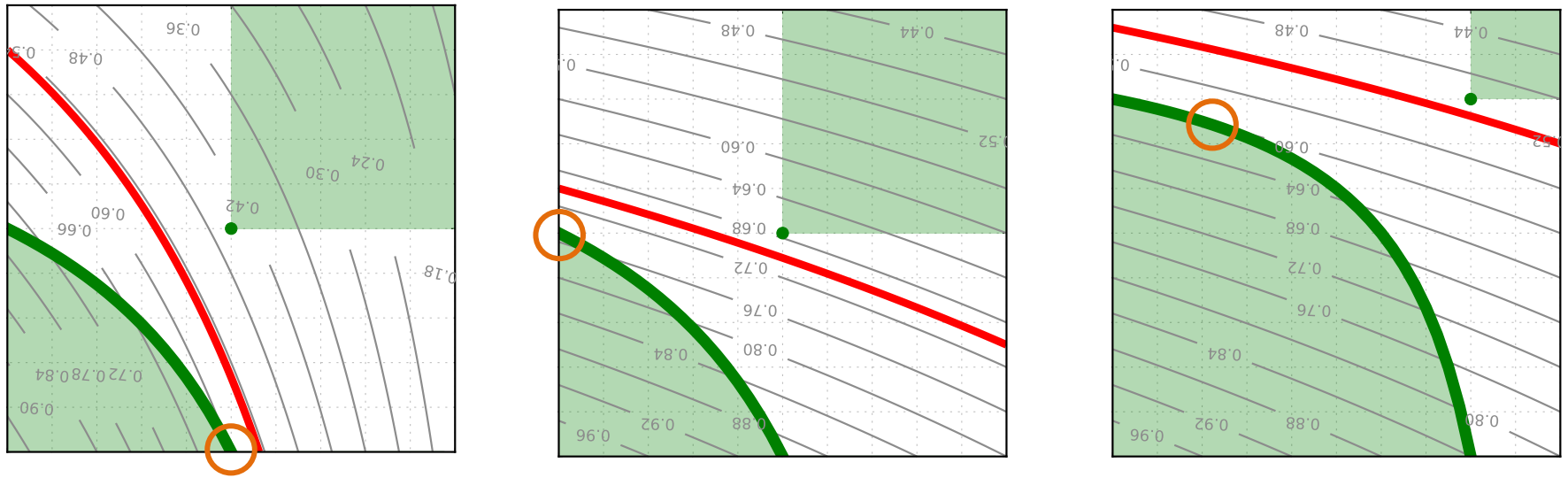
Conclusion at the time was to just use upper bounds and ignore the lower bounds

G., Suciu [VLDB'15]

G., Suciu [VLDBJ'17]

Definition scaled dissociation (informal)

Scaled dissociation (informally): Find the maximal lower bounds among all that fulfill the constraints.



Then use this bound as lower bound for anytime approximation

Finding scaled dissociations is not trivial

- Optimization problem is not nice ☹
  - non-linear objective function
  - non-convex constraint set
  - This makes it difficult to apply optimization methods
- What we are going to do ☺
  - We perform a change of variables s.t. we can instead solve a non-linear optimization problem over convex sets
  - Then apply known gradient-descent (GD) methods

Reduction to convex constraint set

DETAILS

We observe that we can **reformulate** the constraints

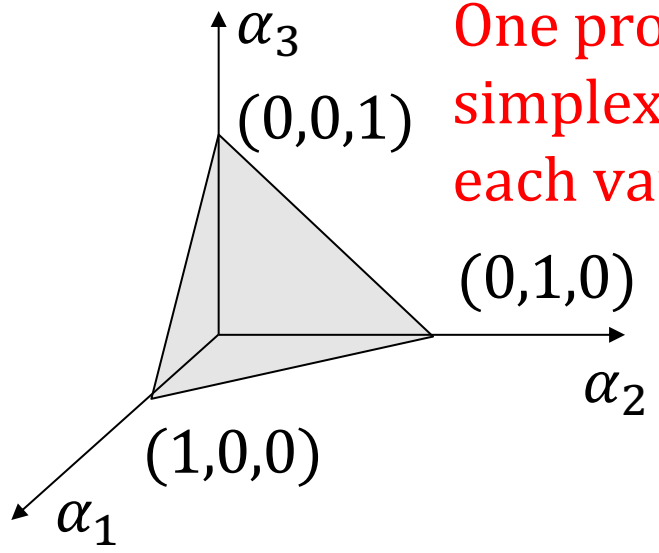
$$\max f(q_1, q_2)$$
$$1 - \mathbb{P}[x] = (1 - q_1)(1 - q_2)$$
$$q_j \in [0,1]$$

$$= (1 - \mathbb{P}[x])^{\alpha_1}(1 - \mathbb{P}[x])^{\alpha_2} = a^{\alpha_1} b^{\alpha_2}$$

$$\max g(\alpha_1, \alpha_2)$$
$$\alpha_1 + \alpha_2 = 1$$
$$\alpha_j \in [0,1]$$

More generally, for d>2

$$\sum_{j \in [d]} \alpha_j = 1$$
$$\boldsymbol{\alpha} \in [0,1]^d$$



One probability simplex Δ_i for each variable i

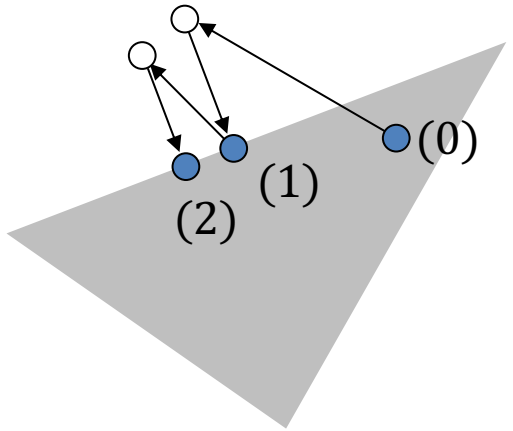
Optimization problem over a set of convex probability simplexes

$$\boldsymbol{\alpha}_{\text{OPT}} = \arg\max\{g(\langle \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots \boldsymbol{\alpha}_n \rangle) | \boldsymbol{\alpha}_i \in \Delta_i, i \in [n]\}$$

Gradient Descent methods

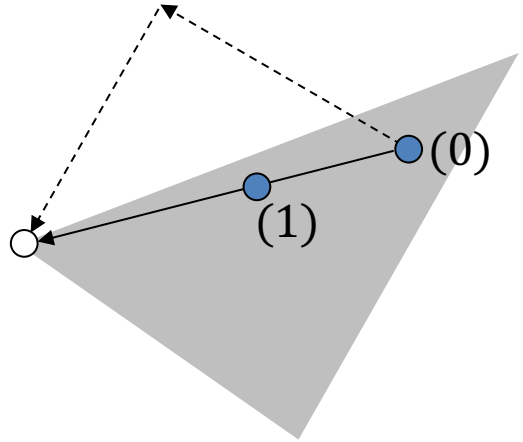
Optimization problem over convex probability simplexes

Projected GD (PGD)



1. Move in the direction of the gradient
2. Project back into Δ

Conditional GD (CGD)



1. Move in the direction of the optimal point in Δ assuming a linearized approximation

... with "some" step size

More details in the paper about making this fast

DETAIL

- Gradient can be calculated efficiently
  - we have read-once formulas, connection to influence of a variable

Kanagal, Li, Deshpande [SIGMOD'11]

- Evaluate $g(\boldsymbol{\alpha})$ and $\nabla g(\boldsymbol{\alpha})$ only once per optimization step
  - To guarantee convergence of to local optimum by PGD and CGD, we would have to re-evaluate $g(\boldsymbol{\alpha})$ and $\nabla g(\boldsymbol{\alpha})$ multiple times per step
  - But we are able to bound differences between gradients in different points in Δ. Thus no need to re-evaluate

Instantiations of anytime approximation framework

- a general framework that allows combinations of instantiation

Procedure	Decisions	Choices
1. Find bounds	Method	MB, SD, PGD, CGD
	# Steps	1, 10
	Strategy	local, global
2. Decompose	Variable selection	Occmax, Imax, etc.

grayed out please see paper

Agenda

1. Probabilistic inference
   - Boolean formulas, anytime approximations
2. Better bounds
   - how to find better bounds than "model-based" bounds
3. Experiments & Take-aways

How does it perform in practice?

- Tried on 4800 lineages
  - Obtained as lineages of hard queries (synthetic, TPC-H, Yago3)
- Compared 39 instantiations of the anytime approximation framework
  - with model-based (MB), symmetric lower bound, scaled dissociation (PGD, CGD)
  - Including various node and variable selection strategies

Take-away message

- our gradient descent (GD) methods perform overall the best.
- Improves prior model-based methods (MB), sometimes quite a lot.
- GD methods should not do too many steps (no need to wait for convergence)

- Details are in the paper. We illustrate next with one synthetic example

Experiment: time needed to reach fixed error guarantee

Data

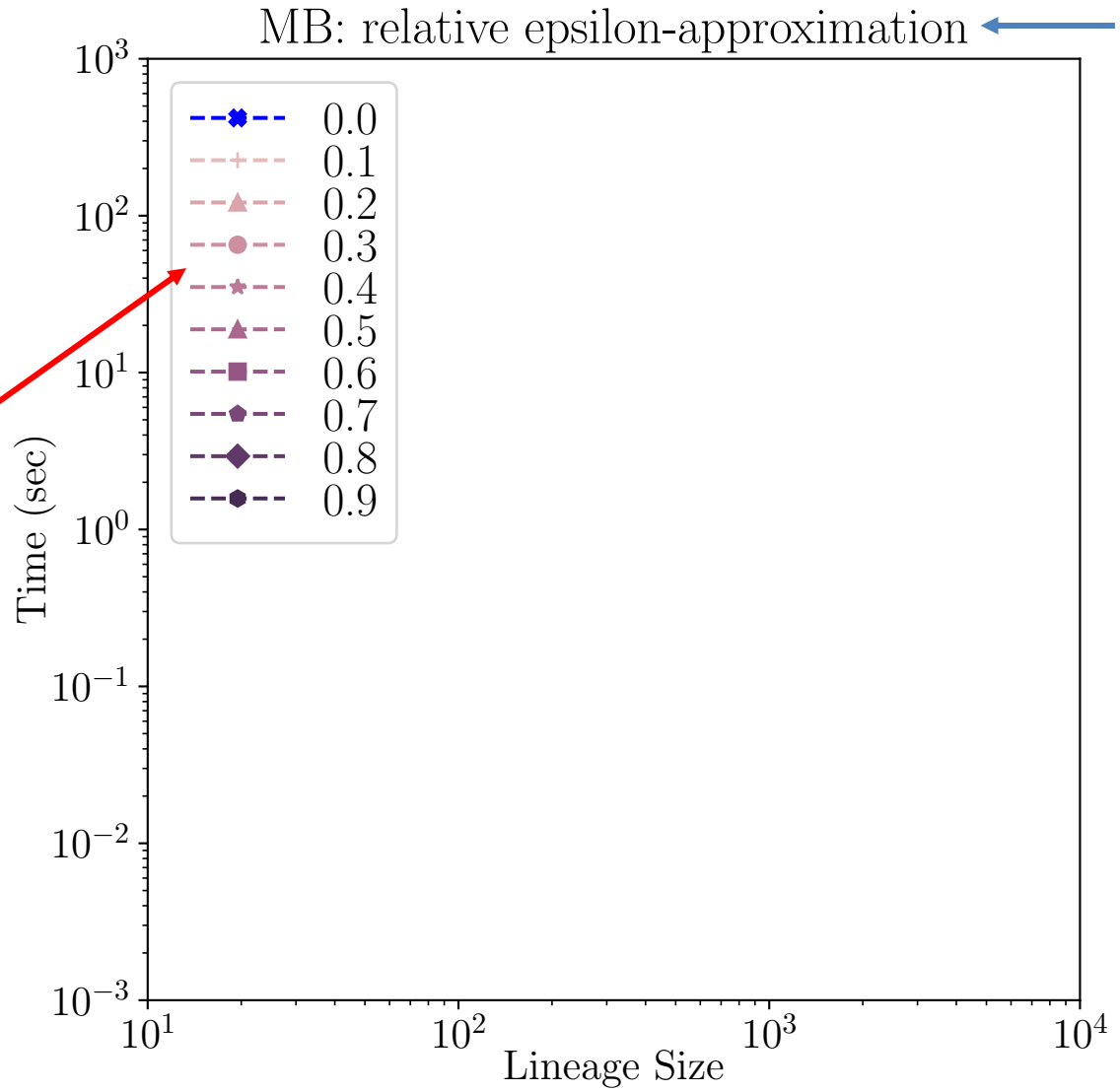
- Boolean chain query R(x)S(x,y)T(y)
- Tuples randomly sampled from domain with size prop to relation size
- probabilities in [0,0.1]

Error guarantees

- Calculate **relative ε-approx.** from bounds U and L

ratio $\frac{U}{L}$	ε
3	0.5
1.5	0.2
1.22	0.1
1	0

Olteanu+[ICDE'10]



MB (prior): model-based 10 random bounds

notice the log scale!

Experiment: time needed to reach fixed error guarantee

Data

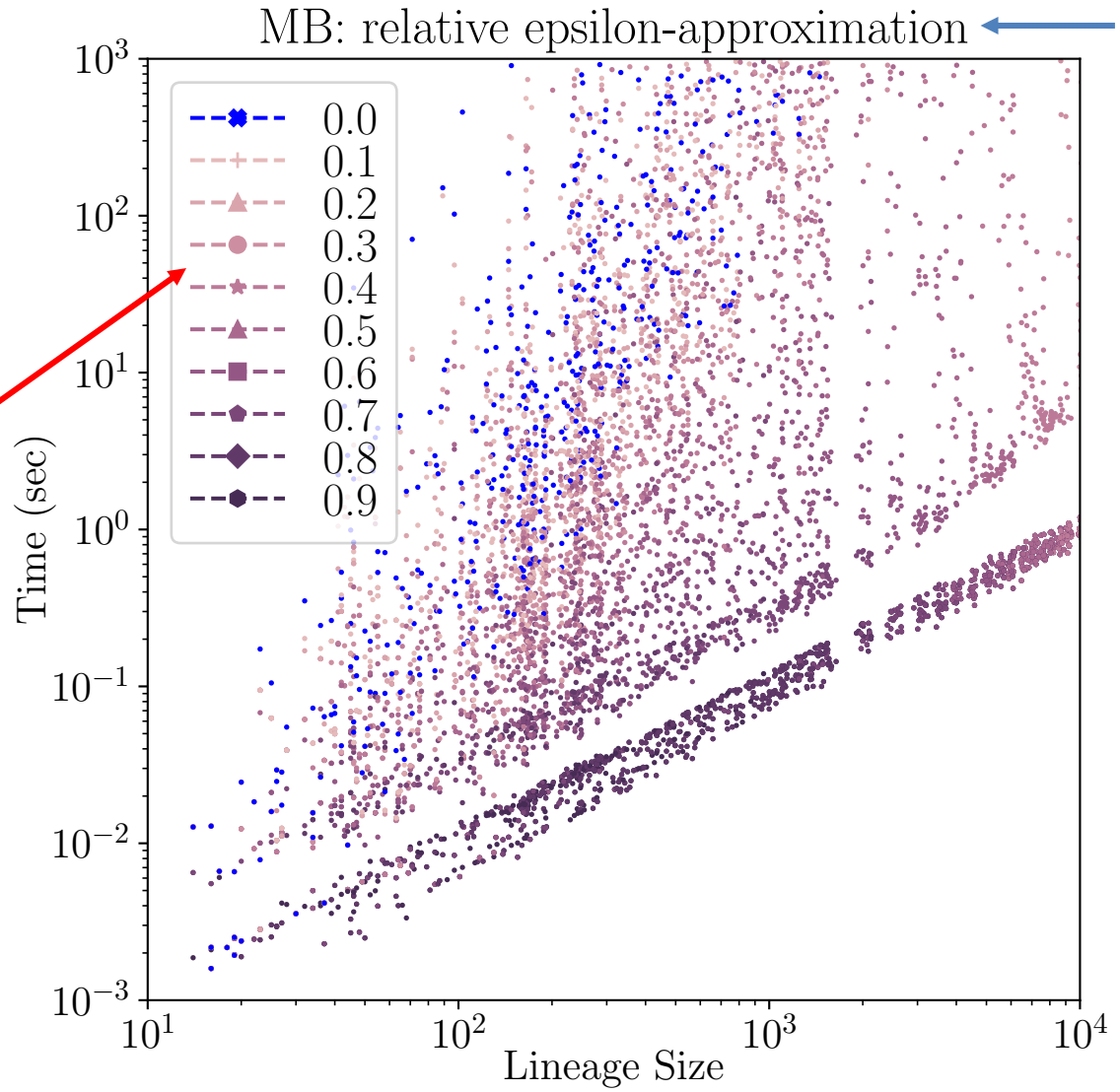
- Boolean chain query R(x)S(x,y)T(y)
- Tuples randomly sampled from domain with size prop to relation size
- probabilities in [0,0.1]

Error guarantees

- Calculate **relative ε-approx.** from bounds U and L

ratio $\frac{U}{L}$	ε
3	0.5
1.5	0.2
1.22	0.1
1	0

Olteanu+[ICDE'10]



MB (prior): model-based
10 random bounds

notice the log scale!

Experiment: time needed to reach fixed error guarantee

Data

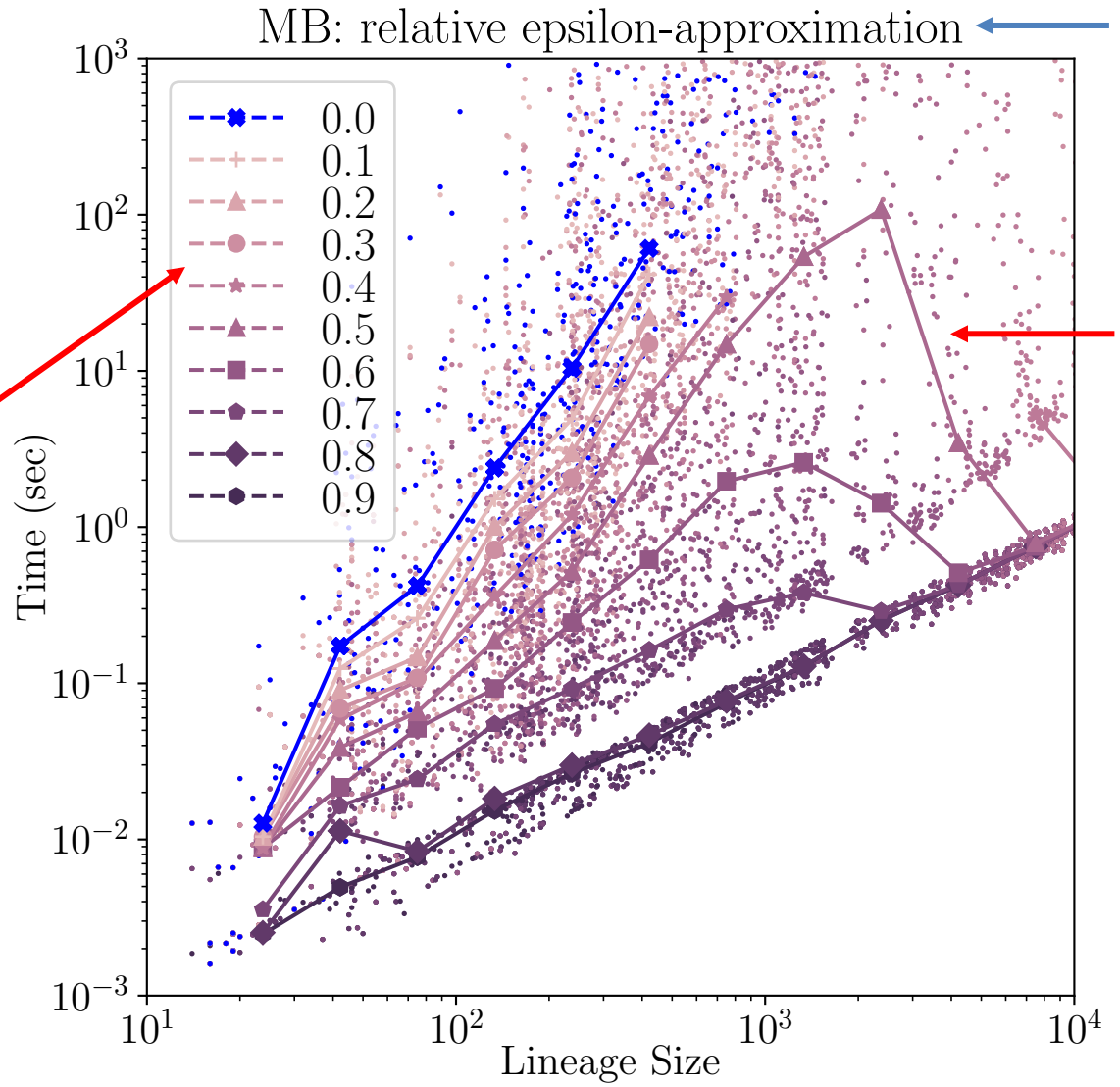
- Boolean chain query R(x)S(x,y)T(y)
- Tuples randomly sampled from domain with size prop to relation size
- probabilities in [0,0.1]

Error guarantees

- Calculate **relative ε-approx.** from bounds U and L

ratio $\frac{U}{L}$	ε
3	0.5
1.5	0.2
1.22	0.1
1	0

Olteanu+[ICDE'10]



MB (prior): model-based 10 random bounds

Median time to reach a certain error guarantee for fixed lin. size

notice the log scale!

Experiment: time needed to reach fixed error guarantee

Data

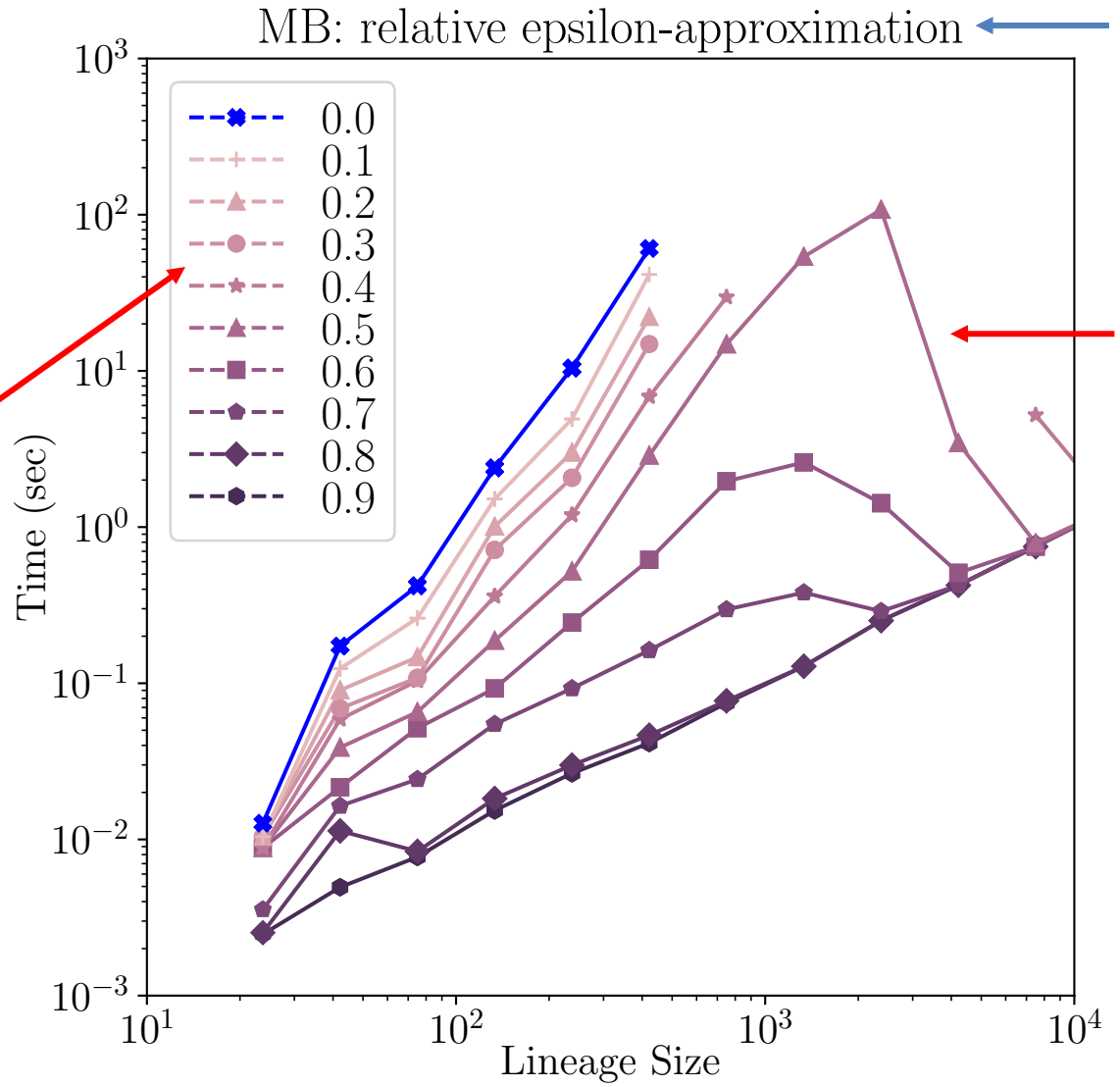
- Boolean chain query R(x)S(x,y)T(y)
- Tuples randomly sampled from domain with size prop to relation size
- probabilities in [0,0.1]

Error guarantees

- Calculate **relative ε-approx.** from bounds U and L

ratio $\frac{U}{L}$	ε
3	0.5
1.5	0.2
1.22	0.1
1	0

Olteanu+[ICDE'10]



MB (prior): model-based 10 random bounds

Median time to reach a certain error guarantee for fixed lin. size

notice the log scale!

Experiment: time needed to reach fixed error guarantee

Data

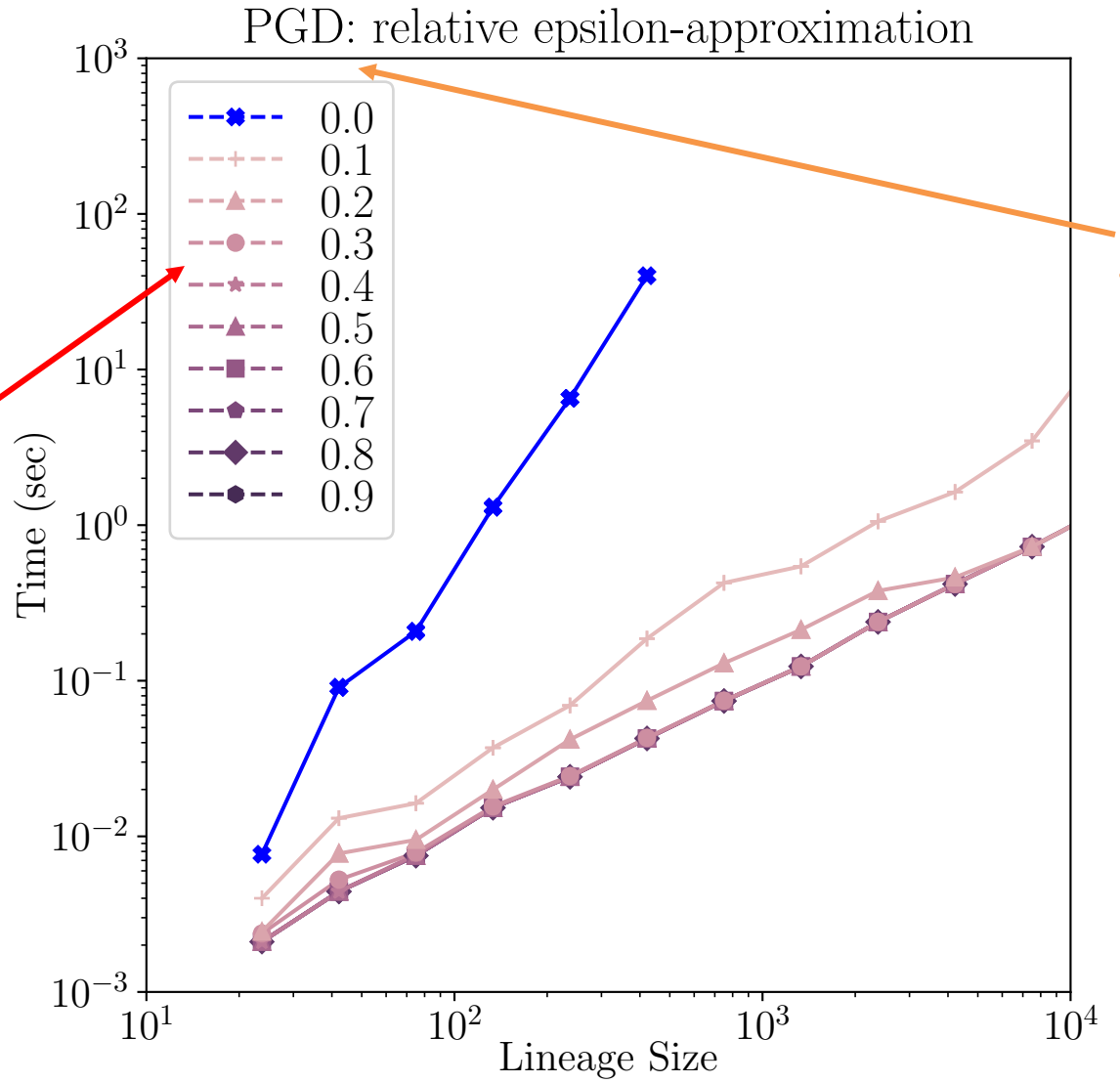
- Boolean chain query R(x)S(x,y)T(y)
- Tuples randomly sampled from domain with size prop to relation size
- probabilities in [0,0.1]

Error guarantees

- Calculate **relative ε-approx.** from bounds U and L

ratio $\frac{U}{L}$	ε
3	0.5
1.5	0.2
1.22	0.1
1	0

Olteanu+[ICDE'10]



PGD (our): projected gradient descent

notice the log scale!

Experiment: time needed to reach fixed error guarantee

Data

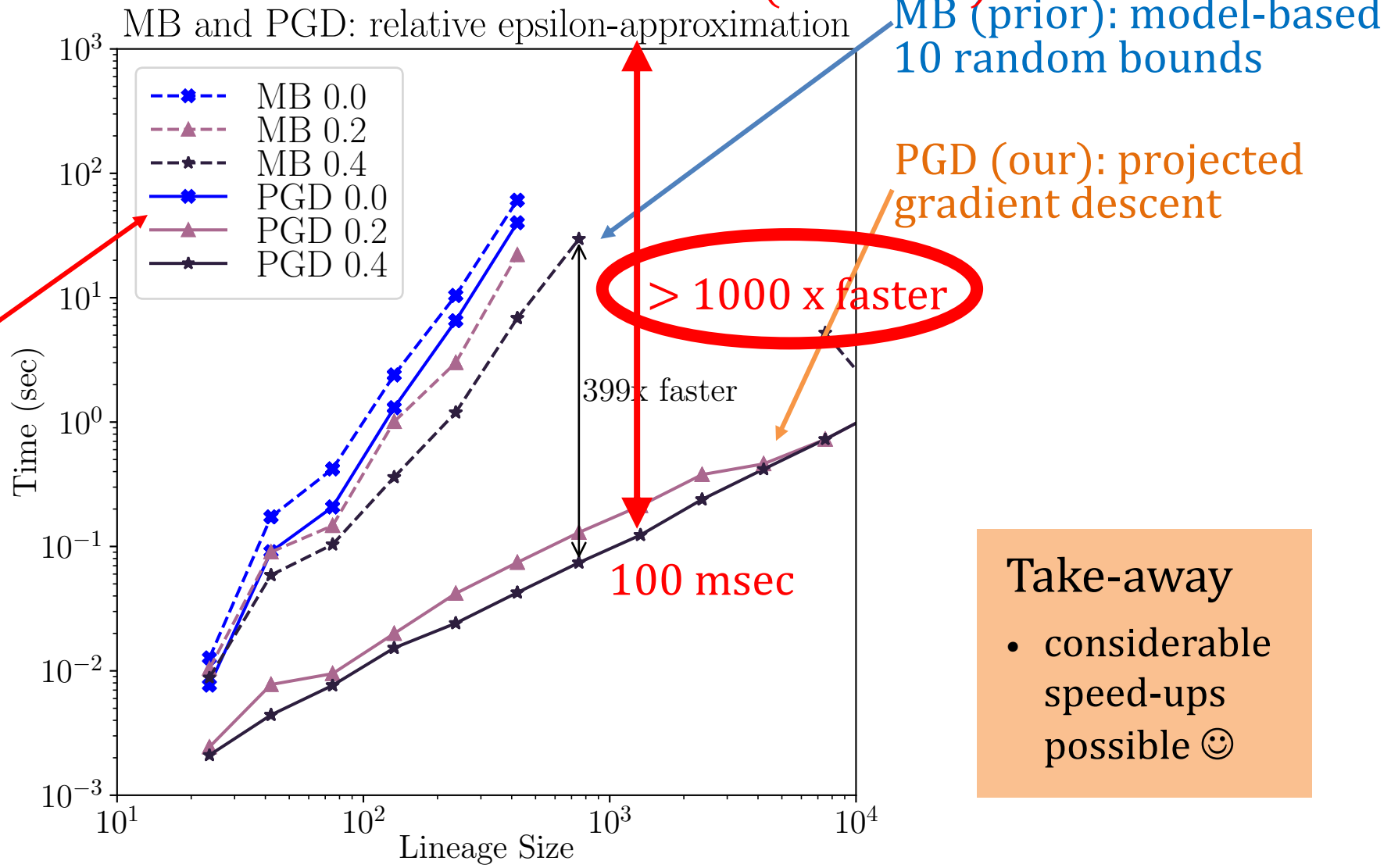
- Boolean chain query R(x)S(x,y)T(y)
- Tuples randomly sampled from domain with size prop to relation size
- probabilities in [0,0.1]

Error guarantees

- Calculate **relative ε-approx.** from bounds U and L

ratio $\frac{U}{L}$	ε
3	0.5
1.5	0.2
1.22	0.1
1	0

Olteanu+[ICDE'10]



Take-away

- considerable speed-ups possible ☺

Take-aways and open points

Take-aways & open points scaled dissociations

Problem: anytime approximations for probabilistic inference

- need to choose from exponentially many model-based approximations (MB)*
- How to get good bounds fast?

Our solution: scaled dissociations

- Replace exponentially many UBs with one single better one
- Embed the combinatorial model-based space of LBs within a continuous enlarged space. Then use gradient-descent (GD) methods (with some tweaks, see paper)

Result:

- consistent speed-ups, at times considerable

Thanks! ☺

Yet to understand:

- Properties of optimization: When is finding the best LB hard, when easy?
- Iterative update methods that work better (but convergence...)
- Is there a principled, perhaps optimization-based approach, to selecting variables for Shannon expansion with the goal of reducing error of approximation?

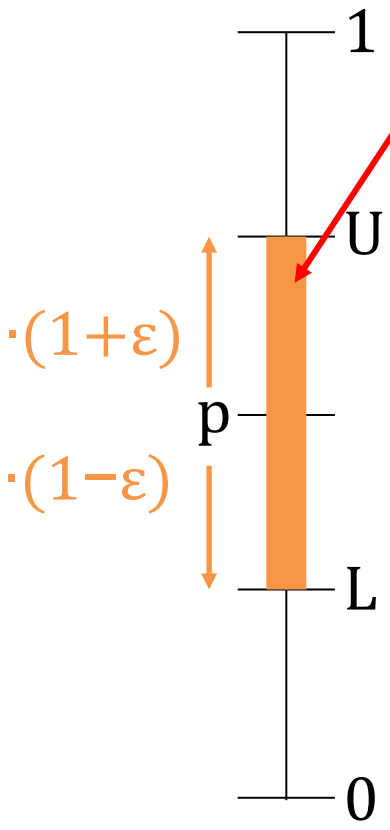
* exponential in number of repeated variables

BACKUP

Relative ε-approximation

Olteanu, Huang, Koch[ICDE'10]

ε-valid interval iff $\frac{U}{1+\varepsilon} \leq \frac{L}{1-\varepsilon}$



Numerical example

U=0.6

then ε=0.2

return 0.5

L=0.4

$$\varepsilon = \frac{\frac{U}{L} - 1}{1 + \frac{U}{L}}$$

ε	ratio $\frac{U}{L}$
0.5	3
0.2	1.5
0.1	1.22
0.05	1.1
0	1

Anytime Approximation in Probabilistic Databases via Scaled Dissociations

Maarten V.d. Heuvel
U of Antwerp

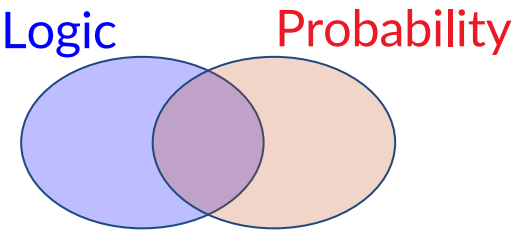
Peter Ivanov
Northeastern U

Wolfgang Gatterbauer
Northeastern U

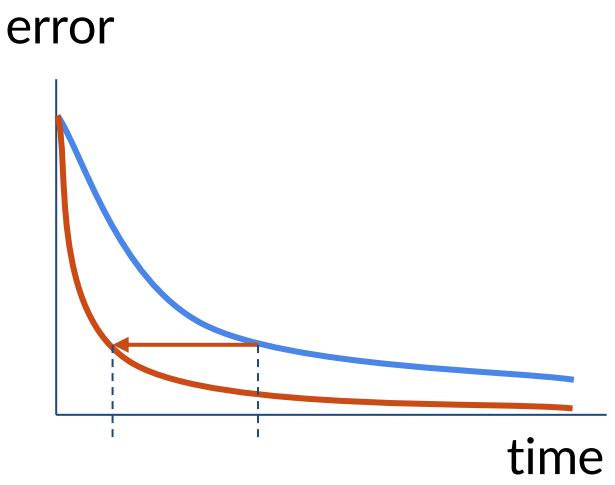
Floris Geerts
U of Antwerp

Martin Theobald
U of Luxemburg

Probabilistic inference
key algorithmic problem in various areas such as Probabilistic Databases, AI, and Statistical Relational Learning. Well known to be hard ☹



Branch & Bound type **Anytime Algorithms**
✅ approximations with flexible accuracy/time trade-off
❌ but how to choose among combinatorially many bounds?



Key idea:
💡 embed combinatorial space of bounds within a continuous space

Results:
– continuous optimization problem (gradient descent) + better bounds
– considerable speed-ups across various data sets

at times **>100x faster**