A Unified Approach to Ranking in Probabilistic Databases

Jian Li, Barna Saha, Amol Deshpande

University of Maryland, College Park, USA

Probabilistic Databases

- Motivation: Increasing amounts of uncertain data
  - Sensor Networks; Information Networks
    - Noisy input data; measurement errors; incomplete data
    - Prevalent use of probabilistic modeling techniques
  - Data Integration and Information Extraction
    - Need to model reputation, trust, and data quality
    - Increasing use of automated tools for schema mapping etc.
  - ...
- Probabilistic databases
  - Annotate *tuples* with existence probabilities, and *attribute values* with probability distributions
  - Propagate probabilities through query execution
  - Interpretation according to the "possible worlds semantics"

Possible World Semantics

ID	Score	Prob
t_1	200	0.2
t_2	150	0.8
t_3	100	0.4

ProbDB

pw1

ID	Score
t_1	200
t_2	150
t_3	100

w.p. 0.064

pw2

ID	Score
t_1	200
t_2	150

w.p. 0.096

pw3

ID	Score
t_2	150
t_3	100

w.p. 0.256

Top-k Query Processing

Score values are used to rank the tuples in every *pw*.

Assume tuples are already sorted in an non-increasing score order.

ID	Score	Prob
t_1	200	0.2
t_2	150	0.8
t_3	100	0.4

ProbDB

pw1

ID	Score
t_1	200
t_2	150
t_3	100

w.p. 0.064

pw2

ID	Score
t_1	200
t_2	150

w.p. 0.096

pw3

ID	Score
t_2	150
t_3	100

w.p. 0.256

The top-1 answer for each possible world

Outline

- Prior Proposals for Top-k Queries over ProbDB
- Parameterized Ranking Functions
- Computing PRF
  - Independent Tuples
  - Computing PRF^e(®)
  - Probabilistic And/Xor Trees
- Approximating Ranking Functions
- Other Results
- Experiments

Top-*k* Queries: Many Prior Proposals

- U-top-k [Soliman et al. '07]
  - Returns the most probable top-*k* answer
- U-rank-k [Soliman et al. '07]
  - At rank *i*, return the tuple with max prob of being rank *i*
- Probabilistic Threshold (PT-k) [Hua et al. '08]
  - Return all tuples *t* s.t. $Pr(r(t) \leq k) \geq \mu$
- Global-top-k [Zhang et al. '08]
  - Return *k* tuples *t* with the largest $Pr(r(t) \leq k)$ values.
- Expected Score
  - Return k tuples *t* with the highest *score(t)Pr(t)*
- Expected Rank [Cormode et al. '09]
  - Return *k* tuples *t* with smallest $\sum Pr(pw) r_{pw}(t)$ ($r_{pw}(t) = |pw|+1$ if $t \notin pw$)

Top-*k* Queries

- Which one should we use???
- Comparing different ranking functions

Normalized Kendall Distance: #reversals and #mismatch elements

lies in [0,1], 0: Same answers, 1: Disjoint answers

	E-Score	PT/GT	U-Rank	E-Rank	U-Top
E-Score	----	0.124	0.302	0.799	0.276
PT/GT	0.124	----	0.332	0.929	0.367
U-Rank	0.302	0.332	-----	0.929	0.204
E-Rank	0.799	0.929	0.929	----	0.945
U-Top	0.276	0.367	0.204	0.945	----

Real Data Set: 100,000 tuples, Top-100

Top-*k* Queries

- Which one should we use???
- Comparing different ranking functions

Normalized Kendall Distance: #reversals and #mismatch elements

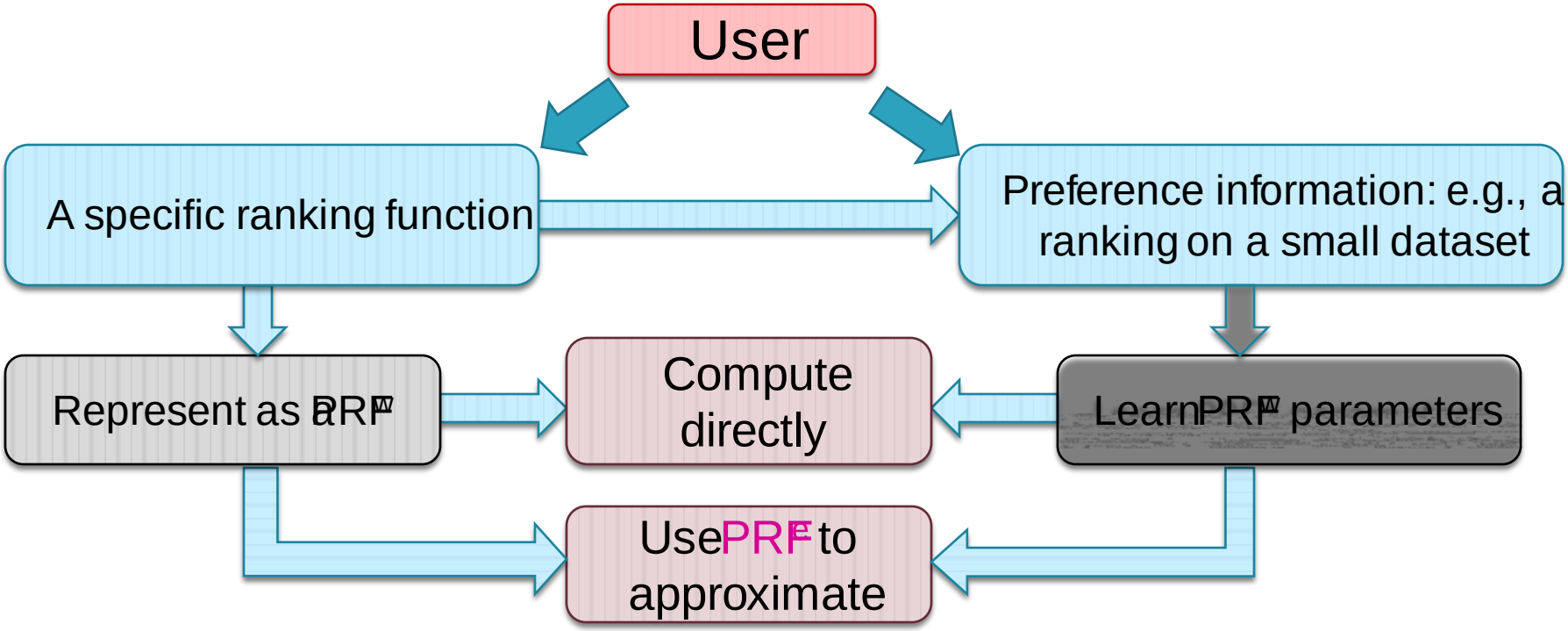
lies in [0,1], 0: Same answers, 1: Disjoint answers

	E-Score	PT/GT	U-Rank	E-Rank	U-Top
E-Score	----	0.864	0.890	0.004	0.925
PT/GT	0.864	----	0.395	0.864	0.579
U-Rank	0.890	0.395	-----	0.890	0.316
E-Rank	0.004	0.864	0.890	----	0.926
U-Top	0.925	0.579	0.316	0.926	----

Synthetic Dataset: 100,000 tuples, Top-100

Our Approach

- Define two *parameterized* ranking functions: PRF^w; PRF^e
  - .. that can simulate or approximate a variety of ranking functi
  - PRF^e much more efficient to evaluate (than PRF^w)



Parameterized Ranking Function

- ! *(t,i)= 1*: Rank the tuples by probabilities
- ! *(t,i)=score(t)*: E-Score
- PRF F(h):
  - PT/GT k: $\omega(i) = \begin{cases} 1, & i \leq k \\ 0, & \text{otherwise} \end{cases}$
  - *U-Rank*: $\omega(i) = \begin{cases} 1, & i=j \end{cases}$ [illegible]

The tuple with the largest Υ_{ω_j} value is the rank-j answer.

- PRF E(®): ! *(i)=®i* where ® *can be a real or a complex number*

Computing PRF: Assuming tuple Independence

T_{i-1}: { t_1, t_2, ... $_{i-1}$}.. , t

- Generating Function Method
  - The coefficient of x^k : *Pr(r(t_i)=k)*
- Algorithm:
  - Fori=1 to n
    - Construct
    - *Expand*
    -

Expand from scratch $O(n^2)$

$O(n^3)$ overall

Computing PRF: Assuming tuple Independence

T_{i-1}: { t_1, t_2, ... $_{i-1}$}.. , t

- Generating Function Method
  - The coefficient of x^k : *Pr(r(t_i)=k)*
- Algorithm:
  - Fori=1 to n
    - Construct
    - Expand
    -

Can be improved to O(n)

$O(n^2)$ overall

Outline

- Prior Proposals for Top-k Queries over ProbDB
- Parameterized Ranking Functions
- Computing PRF
  - Independent Tuples
  - Computing PRF^e(®)
  - Probabilistic And/Xor Trees
- Approximating Ranking Functions
- Other Results
- Experiments

Computing PRF(®): Assuming uple Independence

- Recall: (j)=®j
- Generating Function Method
  -
  -

No need to expand the polynomial !!

- Therefore:
- Morevoer: $\mathcal{F}^i(\alpha) = \frac{\Pr(t_i)}{\Pr(t_{i-1})}\mathcal{F}^{i-1}(\alpha)\Big(1 - \Pr(t_{i-1}) + \Pr(t_{i-1})\alpha\Big)$

O(1)

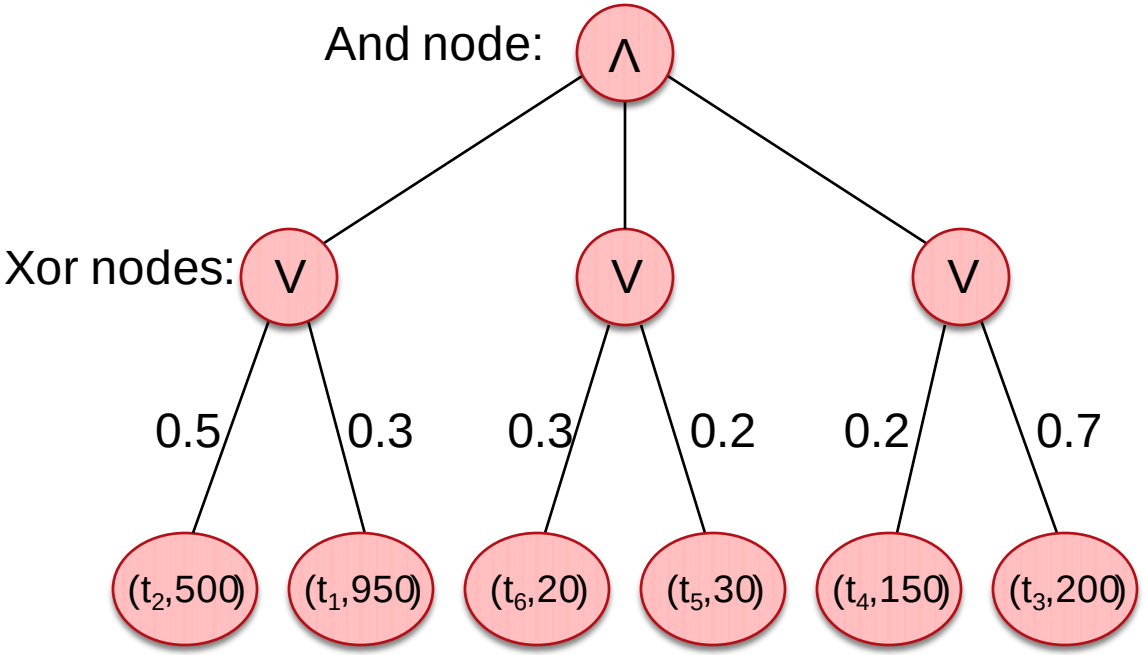
O(n)overall

Outline

- Prior Proposals for Top-k Queries over ProbDB
- Parameterized Ranking Functions
- Computing PRF
  - Independent Tuples
  - Computing PRF^e(®)
  - Probabilistic And/Xor Trees
- Approximating Ranking Functions
- Other Results
- Experiments

Computing PRF: Probabilistic And/Xor Trees

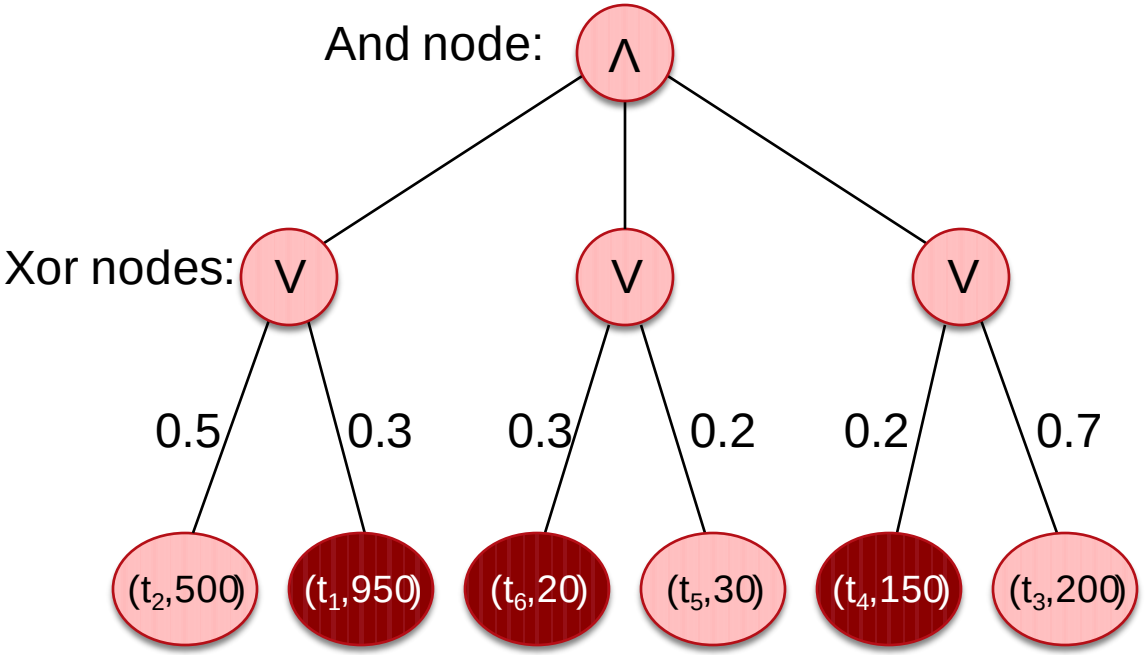
- Capture two types of correlations: mutual exclusitivity and coexistence.



Possible Worlds	Pr
t_4	0.02
t_3	0.08
é é .	
t_1, t_4, t_6	0.03
t_1, t_3, t_6	0.018
é é .	

Computing PRF: Probabilistic And/Xor Trees

- Capture two types of correlations: mutual exclusitivity and coexistence.



Possible Worlds	Pr
t_4	0.02
t_3	0.08
é é .	
t_1, t_4, t_6	**0.018**
t_1, t_3, t_6	0.012
é é .	

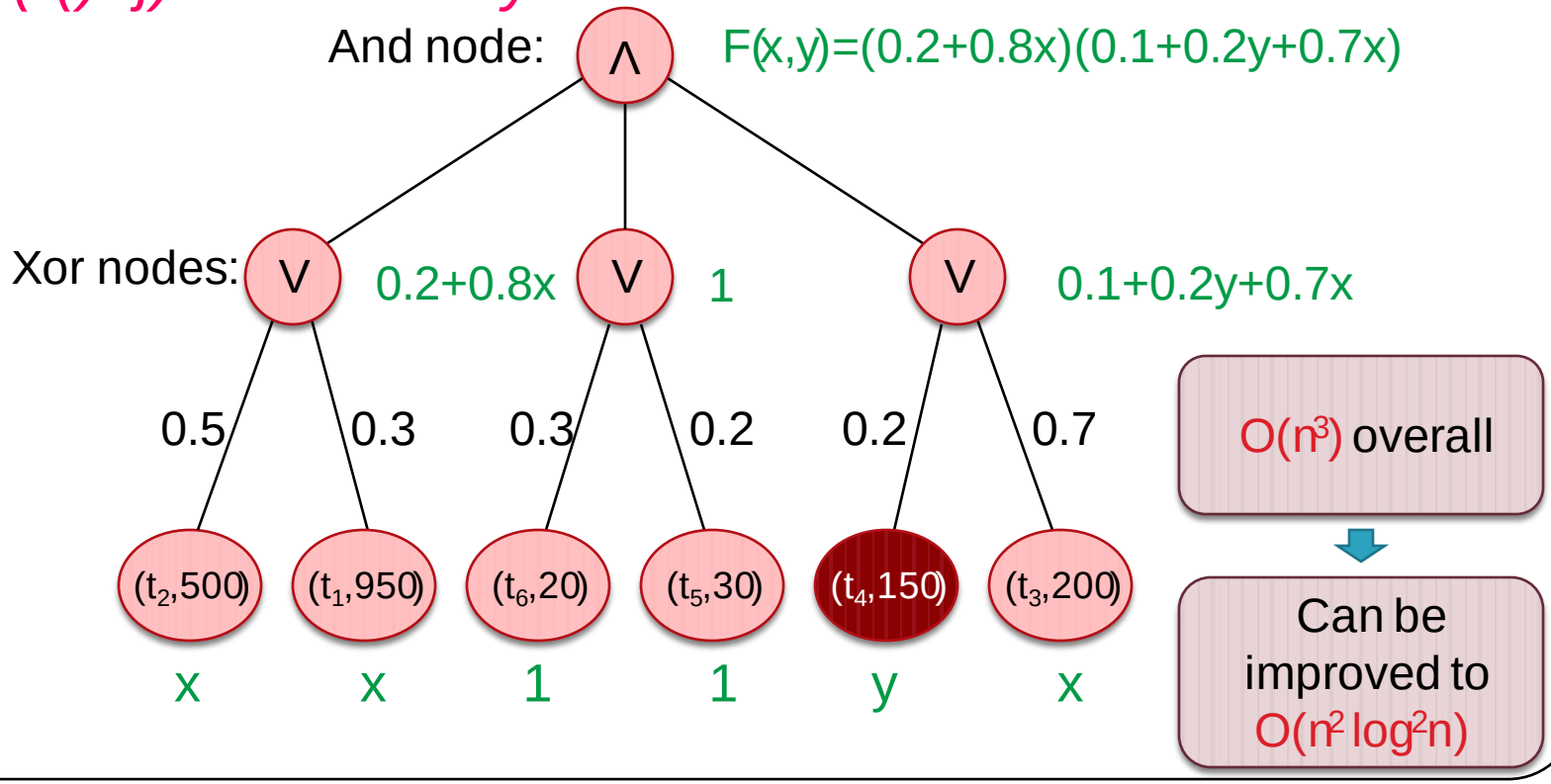
Pr=0.3*0.3*0.2=0.018

Computing PRF: Probabilistic And/Xor Trees

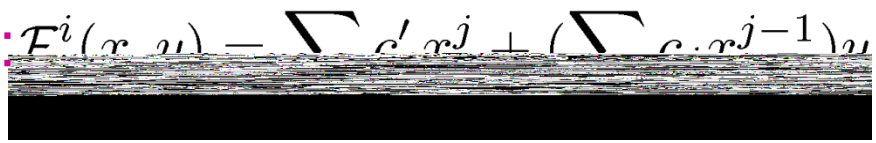
r(i)=j if and only if (1) *j-1* tuples with higher scores appear

(2) tuple *i* appears

Pr(r(i)=j) = coeff of $x^{j-1}y$



Computing PRF(®): Probabilistic And/Xor Trees

Generating Function: 

The coeff of $x^{j-1}y$: $c_j = Pr(r(t_i)=j)$

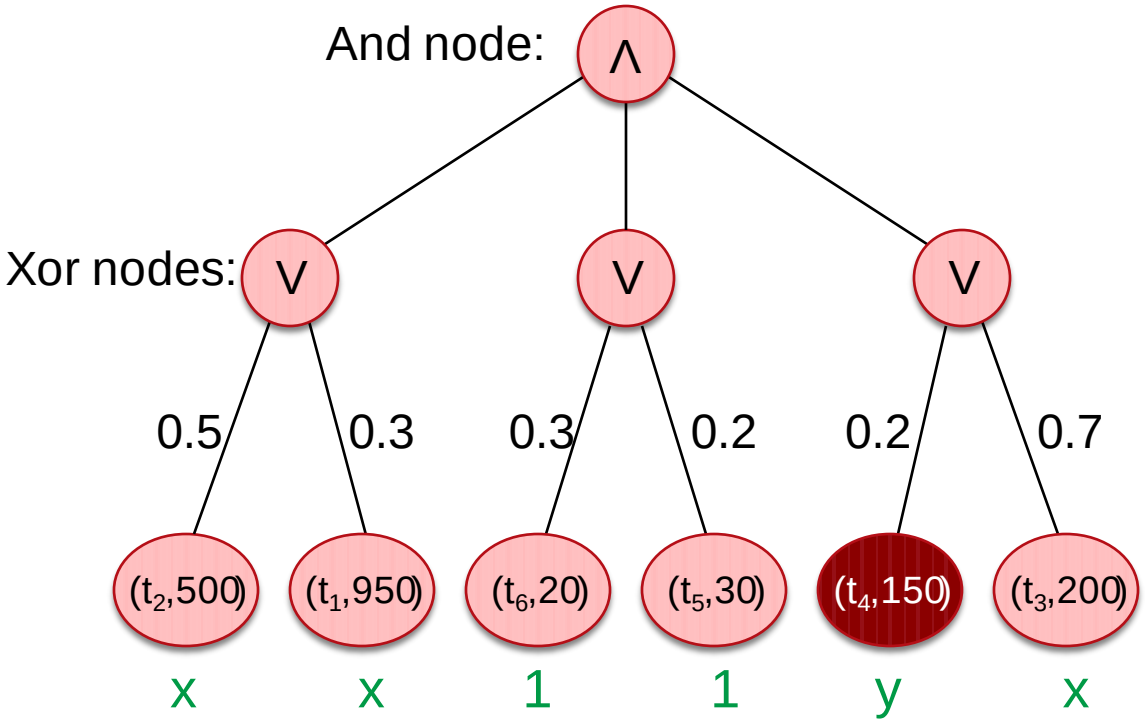
Therefore:

No need to expand the polynomial !!

Computing PRF: Probabilistic And/Xor Trees

We maintain only the numerical values of $F_i(\alpha,\alpha)$ and $F_i(\alpha,0)$ at each node.

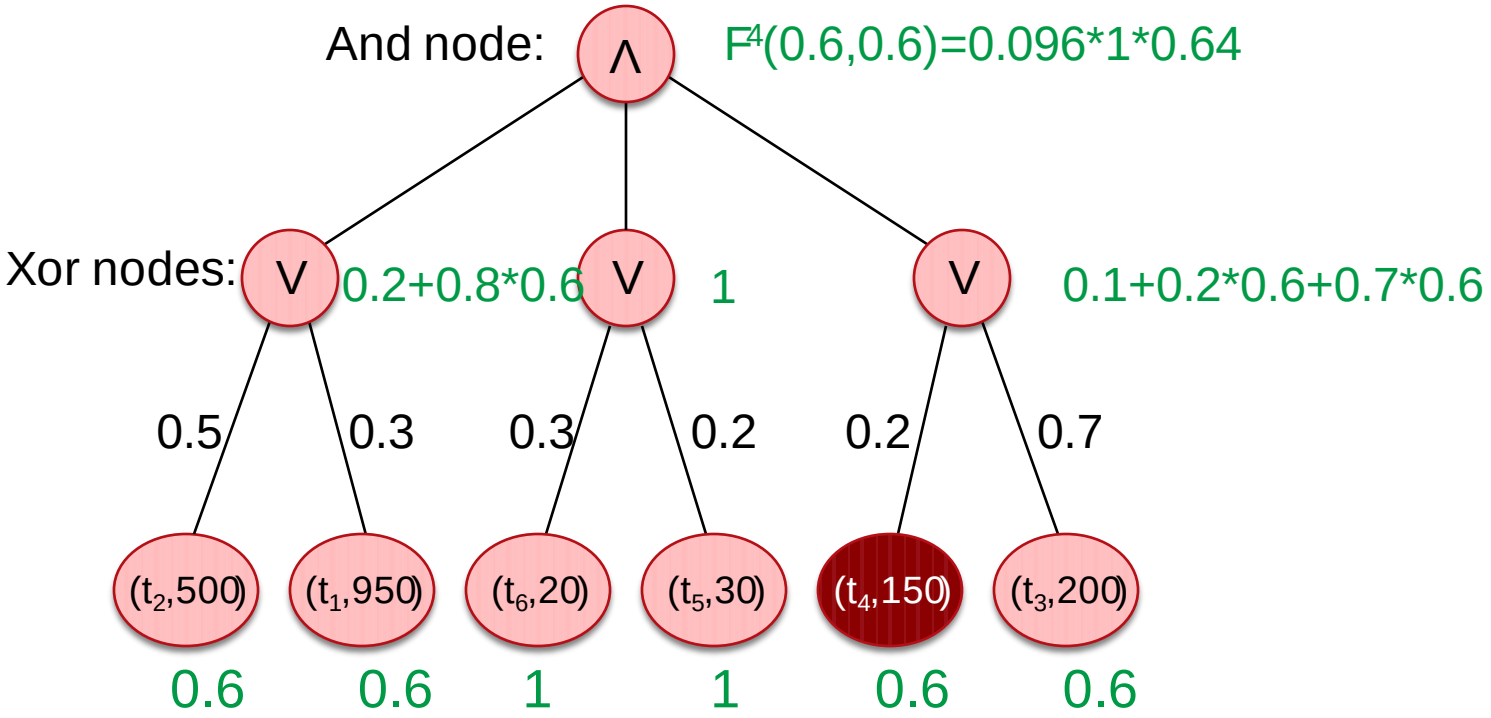
E.g. $\alpha=0.6$. We are computing $F_i(0.6,0.6)$



Computing PRF(®): Probabilistic And/Xor Trees

We maintain only the numerical values of $F^i(®,®)$ and $F^i(®,0)$ at each node.

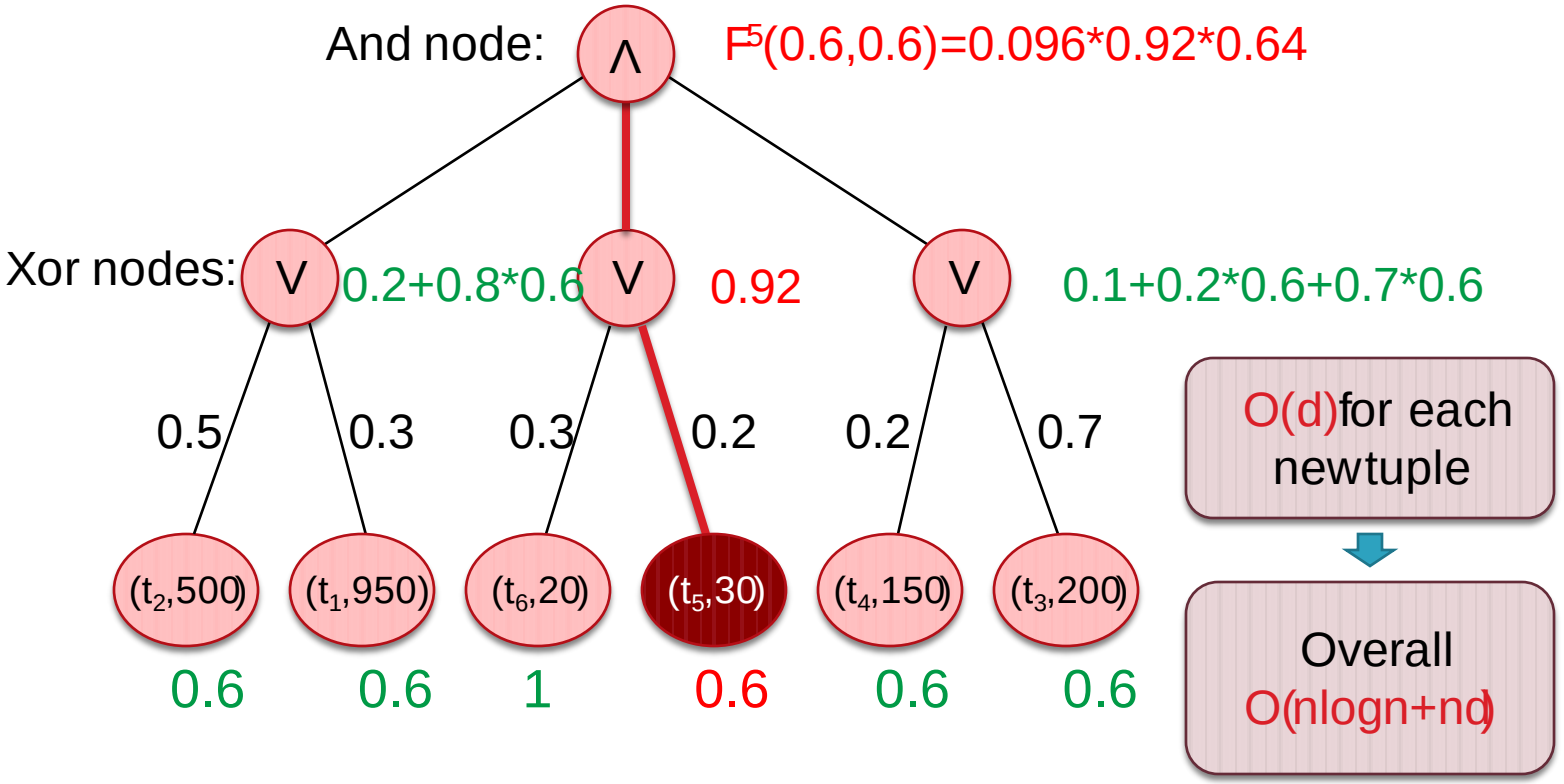
E.g. ®=0.6. We are computing $F^4(0.6,0.6)$



Computing PRF: Probabilistic And/Xor Trees

We maintain only the numerical values of $F^i(®,®)$ and $F^i(®,0)$ at each node.

E.g. ®=0.6. Now we want to compute $F^5(0.6,0.6)$



Outline

- Prior Proposals for Top-k Queries over ProbDB
- Parameterized Ranking Functions
- Computing PRF
  - Independent Tuples
  - Computing PRF^e(®)
  - Probabilistic And/Xor Trees
- Approximating Ranking Functions
- Other Results
-

Approximating Ranking Functions

- How to approximate () by a linear combination of exponentials?
- A scheme based on Discrete Fourier Transformation

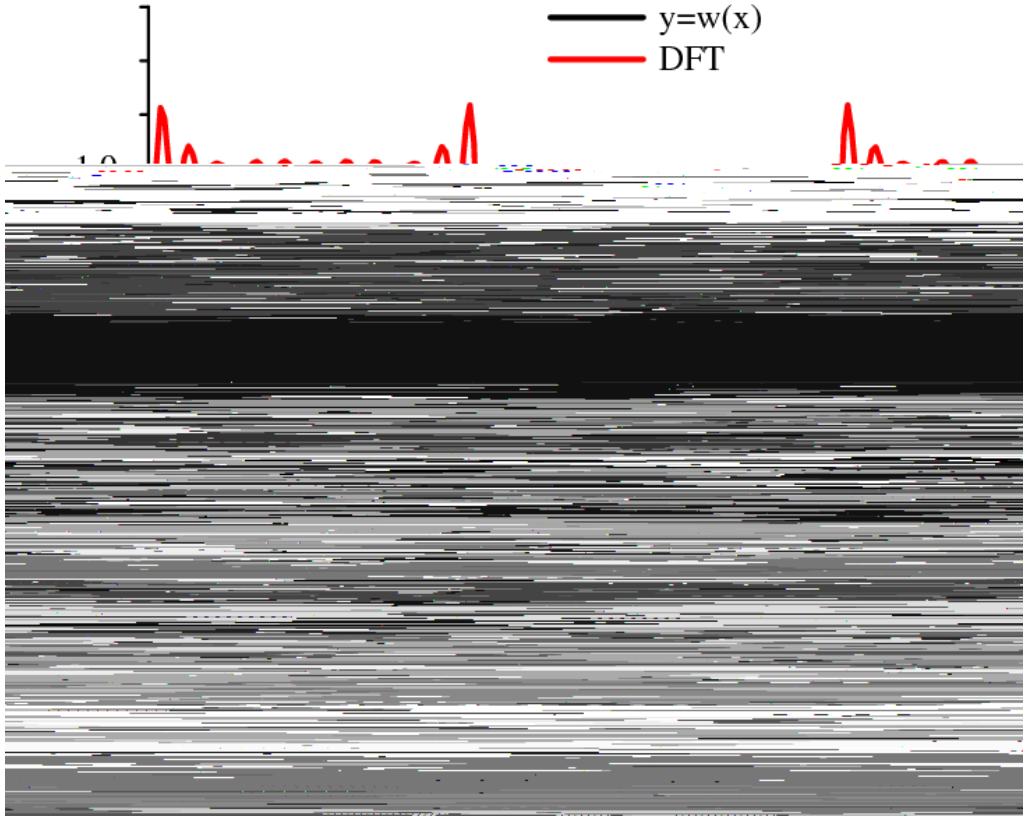
$$\omega(i) = \frac{1}{N} \sum_{k=0}^{N-1} \psi(k) e^{\frac{2\pi \jmath}{N} ki} \qquad i = 0, \ldots, N-1.$$

Ã(0)Ã(N-1), is the DFT of(0)! (N-1),

- Use the largest L DFT coefficients

$$\tilde{\omega}^{DFT}(i) = \frac{1}{N} \sum_{k=0}^{L-1} \psi(k) e^{\frac{2\pi \jmath}{N} ki} \qquad i = 0, \ldots, N-1.$$

Approximating Ranking Functions



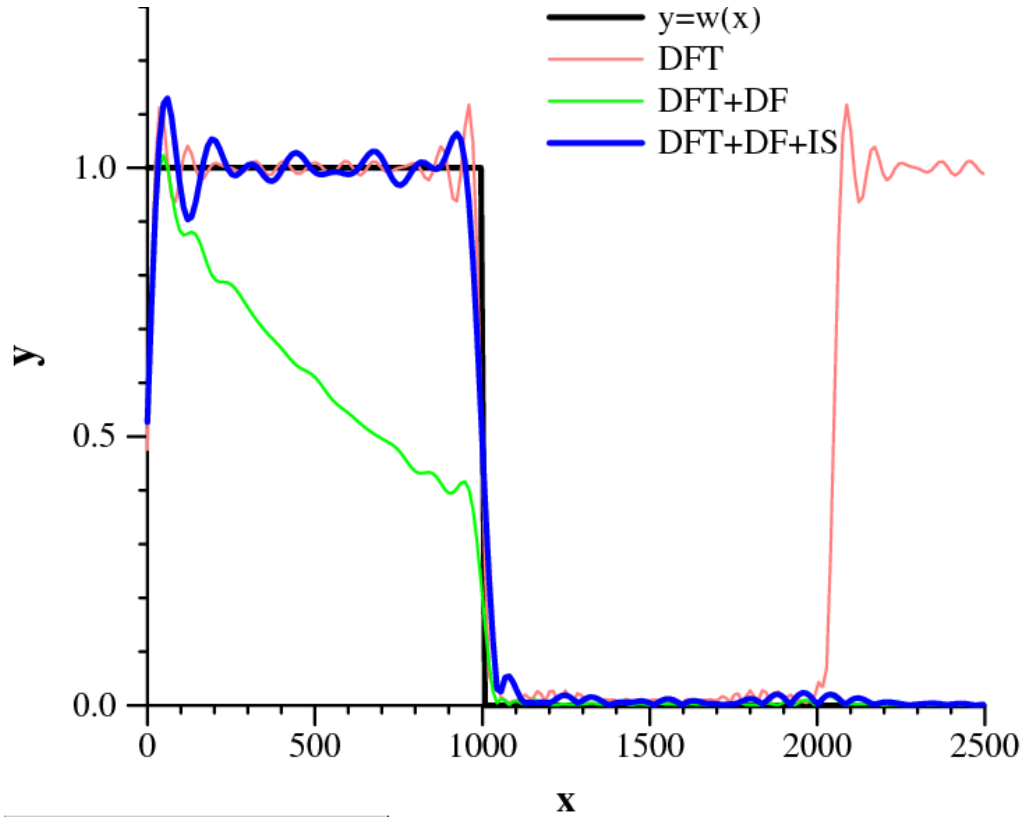
y=w(x)

DFT: Periodic !

Approximating w(x)=1 if x<=1000, 0 if x>1000

Approximating Ranking Functions

Initial Scaling: Perform DFT on a scaled sequence $\acute{}^{-i}!\ (i)\quad i = 0;:::;N\ ¡\ 1$



y=w(x)

DFT : Periodic !

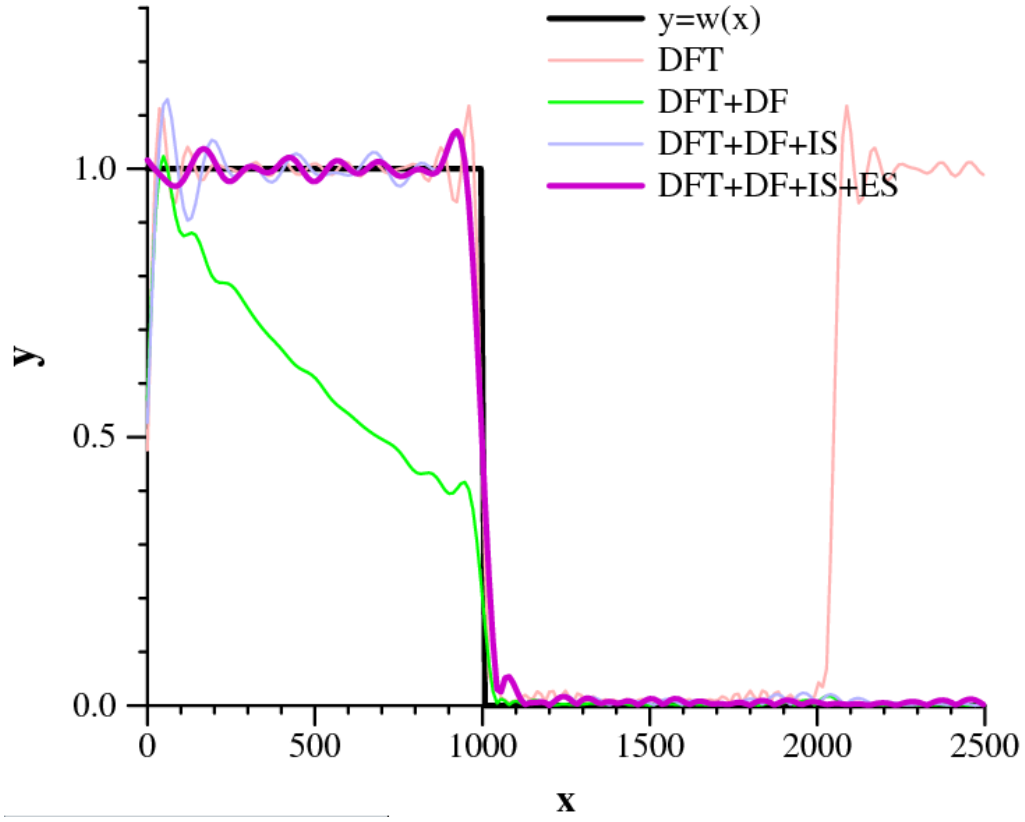
DFT+Damping Factor : Biased approximation

DFT+DF+ Initial Scaling: Unbiased

Approximating w(x)=1 if x<=1000, 0 if x>1000

Approximating Ranking Functions

Extending and Shifting: Particular tailored for optimizing the approximation quality around the origin.



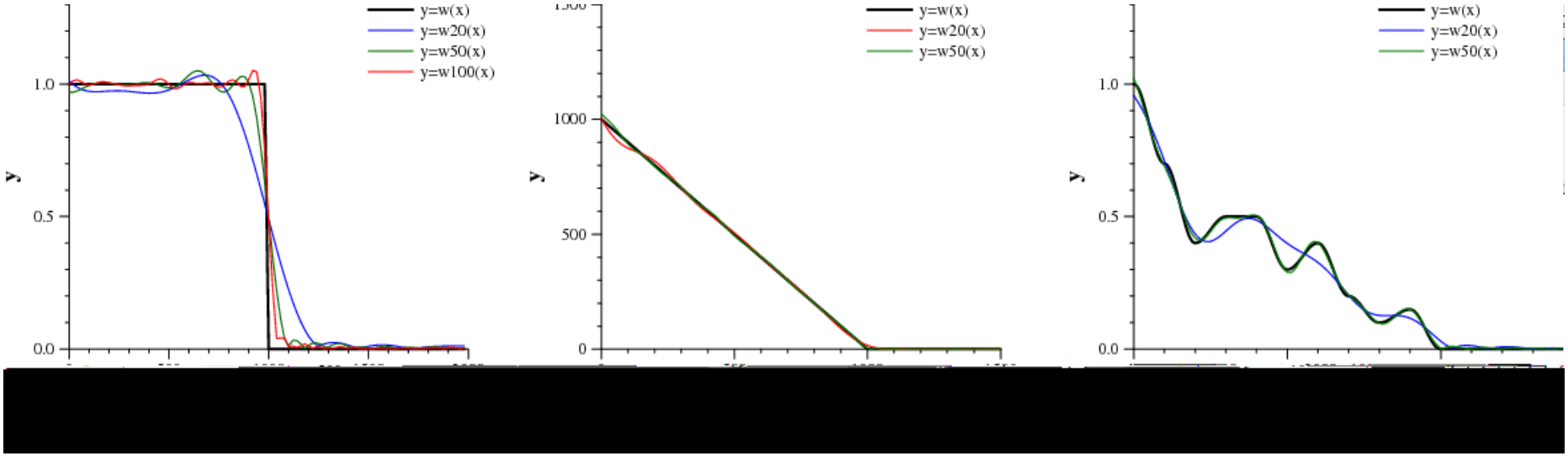
y=w(x)

DFT : Periodic !

DFT+Damping Factor : Biased approximation

DFT+DF+ Initial Scaling: Unbiased

DFT+DF+IS+ Extending&Shifting: Unbiased and better quality around origin

Approximating w(x)=1 if x<=1000, 0 if x>1000

Approximating Ranking Functions