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

Prob DB

→

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

Prob DB

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 $\text{PRF}^{e}(\alpha)$
  - 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. $Pr(r(t) \leq k) \geq \theta$
- 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

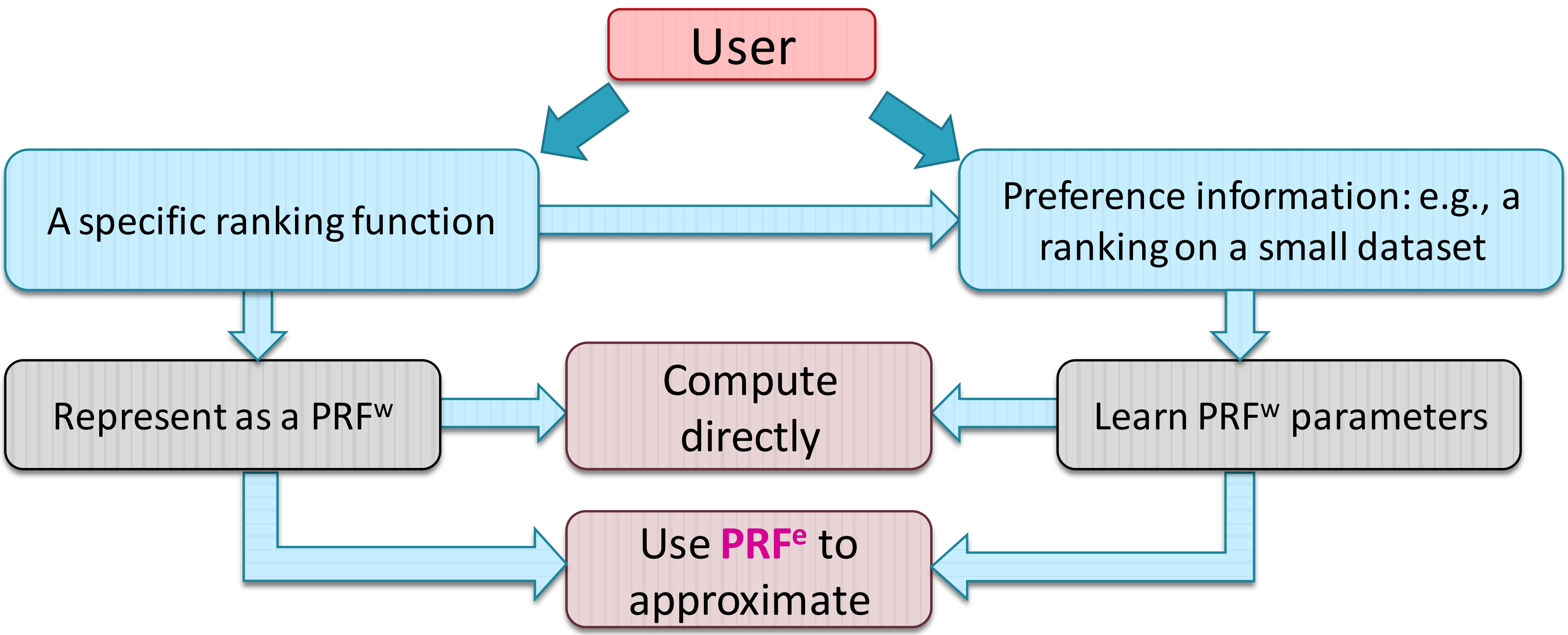
Normalized Kendall Distance: #reversals and #mismatch elements
lies in [0,1], 0: Same answers; 1: Disjoint answers

	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 functions
  - PRF^e much more efficient to evaluate (than PRF^w)



Outline

- Prior Proposals for Top-*k* Queries over ProbDB
- Parameterized Ranking Functions
- Computing PRF
  - Independent Tuples
  - Computing $\mathrm{PRF}^e(\alpha)$
  - Probabilistic And/Xor Trees
- Approximating Ranking Functions
- Other Results
- Experiments

Parameterized Ranking Function

- Weight Function: $\omega : \mathsf{T} \times \mathsf{N} \rightarrow \mathsf{C}$
- Parameterized Ranking Function (PRF)

$$
\begin{aligned}
\Upsilon_{\omega}(t) &= \sum_{pw:t\in pw} \omega(t, r_{pw}(t)) \cdot \mathsf{Pr}(pw) \\
&= \sum_{pw:t\in pw} \sum_{i>0} \omega(t,i) \mathsf{Pr}(pw \wedge r_{pw}(t) = i) \\
&= \sum_{i>0} \omega(t,i) \cdot \mathsf{Pr}(r(t) = i).
\end{aligned}
$$

Return *k* tuples with the highest $|\Upsilon_{\omega}|$ values.

Parameterized Ranking Function

- $\omega(t,i) = 1$: Rank the tuples by probabilities
- $\omega(t,i) = score(t)$: E-Score
- **PRF$^{\omega}$(h):** $\omega(t,i) = \omega(i)$ and $\omega(i) = 0 \ \forall i > h$
  - PT/GT-k : $\omega(i) = \begin{cases} 1, & i \leq k \\ 0, & \text{otherwise} \end{cases}$
  - *U-Rank*: $\omega_j(i) = \begin{cases} 1, & \text{i=j} \\ 0, & \text{otherwise} \end{cases}$

    The tuple with the largest Υ_{ω_j} value is the rank-j answer.
- **PRF^{e}(α):** $\omega(i) = \alpha^i$ where α *can be a real or a complex number*

Outline

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

Computing PRF: Assuming tuple Independence

T_{i-1}: the set of tuples with scores higher than t_i
σ : Boolean indicator vector

$$\begin{aligned}
\Pr(r(t_i) = j) &= \Pr(t_i) \sum_{pw:|pw \cap T_{i-1}|=j-1} \Pr(pw) \\
&= \Pr(t_i) \sum_{\sigma:\sum_{l=1}^{i-1} \sigma_l = j-1} \prod_{l<i:\sigma_l=1} \Pr(t_l) \prod_{l<i:\sigma_l=0} (1 - \Pr(t_l))
\end{aligned}$$

- Generating Function Method

$$\mathcal{F}(x) = \prod_{i=1}^{n} (a_i + b_i x)$$

  - The coefficient of x^k : $\sum_{\beta:\sum_{i=1}^{n} \beta_i = k} \prod_{i:\beta_i=0} a_i \prod_{i:\beta_i=1} b_i$

Computing PRF: Assuming tuple Independence

T_{i-1}: { t_1, t_2, … … , t_{i-1} }

- Generating Function Method

$$\mathcal{F}^i(x) = \left(\prod_{t \in T_{i-1}} \left(1 - \Pr(t) + \Pr(t) \cdot x\right) \right) \left(\Pr(t_i) \cdot x\right)$$

  - The coefficient of x^k : *Pr(r(t_i)=k)*

- **Algorithm:**
  - For i=1 to n
    - Construct $\mathcal{F}^i(x)$
    - *Expand* $\mathcal{F}^i(x) = \sum_{j=1}^{n} \Pr(r(t_i) = j) x^j$
    - $\Upsilon(t_i) = \sum_{j=1}^{n} \omega(t_i, j) \Pr(r(t_i) = j)$

Expand from scratch
$O(n^2)$

$O(n^3)$ overall

Computing PRF: Assuming tuple Independence

T_{i-1}: { t_1, t_2, … … , t_{i-1} }

- Generating Function Method

$$\mathcal{F}^i(x) = \left(\prod_{t \in T_{i-1}} \left(1 - \Pr(t) + \Pr(t) \cdot x\right) \right) (\Pr(t_i) \cdot x)$$

  - The coefficient of x^k : *Pr(r(t_i)=k)*

- **Algorithm:**
  - For i=1 to n
    - Construct $\mathcal{F}^i(x)$
    - Expand $\mathcal{F}^i(x) = \sum_{j=1}^{n} \Pr(r(t_i) = j) x^j$
    - $\Upsilon(t_i) = \sum_{j=1}^{n} \omega(t_i, j) \Pr(r(t_i) = j)$

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 $PRF^e(\alpha)$
  - Probabilistic And/Xor Trees
- Approximating Ranking Functions
- Other Results
- Experiments

Computing PRF$^e(\alpha)$: Assuming tuple Independence

- Recall $\omega(j)=\alpha^j$
- Generating Function Method
  - $\mathcal{F}^i(x) = \sum_{j=1}^{n} \mathsf{Pr}(r(t_i) = j)x^j$
  - $\Upsilon(t_i) = \sum_{i=1}^{n} \mathsf{Pr}(r(t_i) = j)\omega(i) = \sum_{i=1}^{n} \mathsf{Pr}(r(t_i) = j)\alpha^j$

$$\boxed{\Upsilon(t_i) = \mathcal{F}^i(\alpha)}$$

No need to expand the polynomial !!

- Therefore: $\mathcal{F}^i(\alpha) = \left(\prod_{t \in T_{i-1}} \Big(1 - \mathsf{Pr}(t) + \mathsf{Pr}(t) \cdot \alpha\Big) \right) (\mathsf{Pr}(t_i) \cdot \alpha)$
- Morevoer: $\mathcal{F}^i(\alpha) = \frac{\mathsf{Pr}(t_i)}{\mathsf{Pr}(t_{i-1})} \mathcal{F}^{i-1}(\alpha)\Big(1 - \mathsf{Pr}(t_{i-1}) + \mathsf{Pr}(t_{i-1})\alpha\Big)$

O(1)

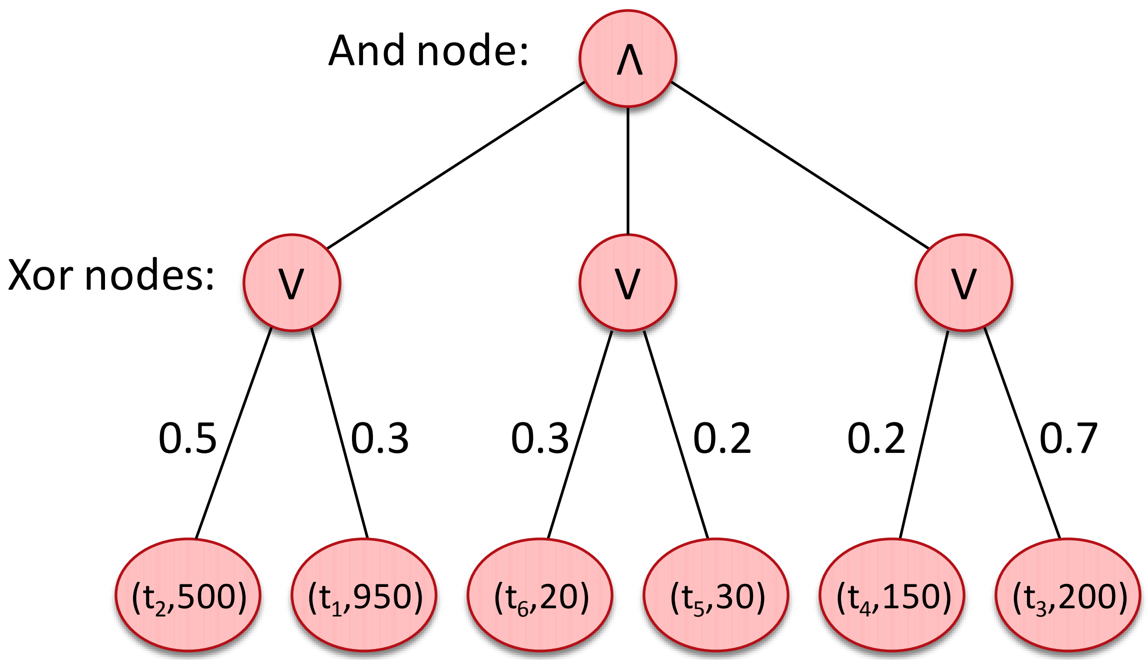
O(n) overall

Outline

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

Computing PRF: Probabilistic And/Xor Trees

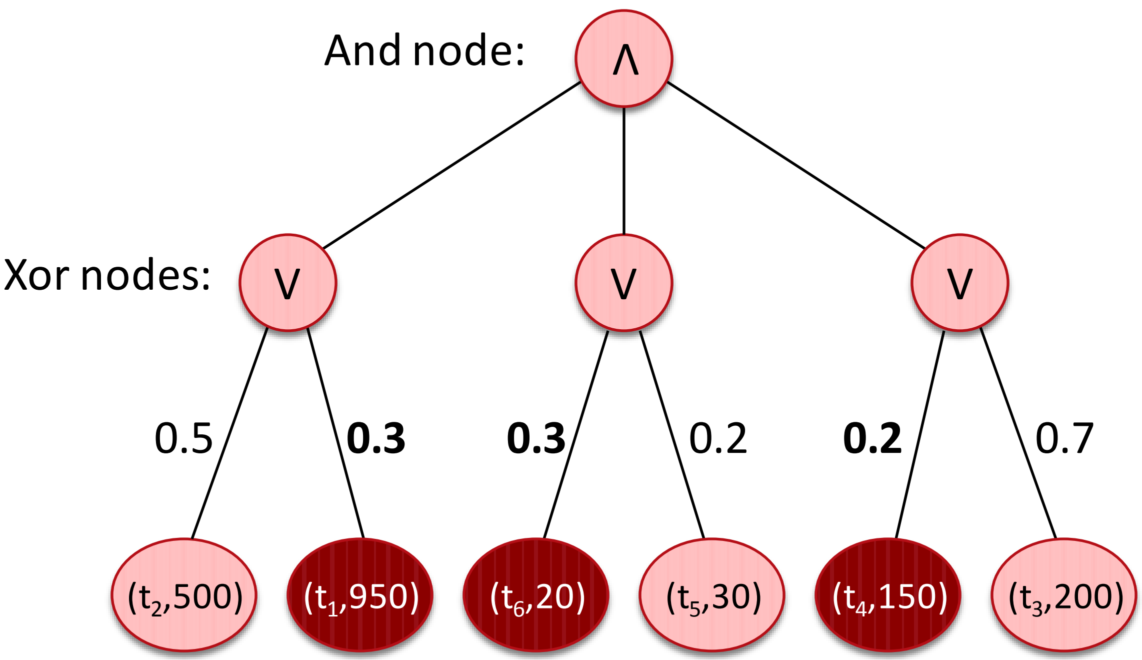
- Capture two types of correlations: mutual exclusivity and coexistence.



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

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
…….	
t_1, t_4, t_6	**0.018**
t_1, t_3, t_6	0.012
…….	

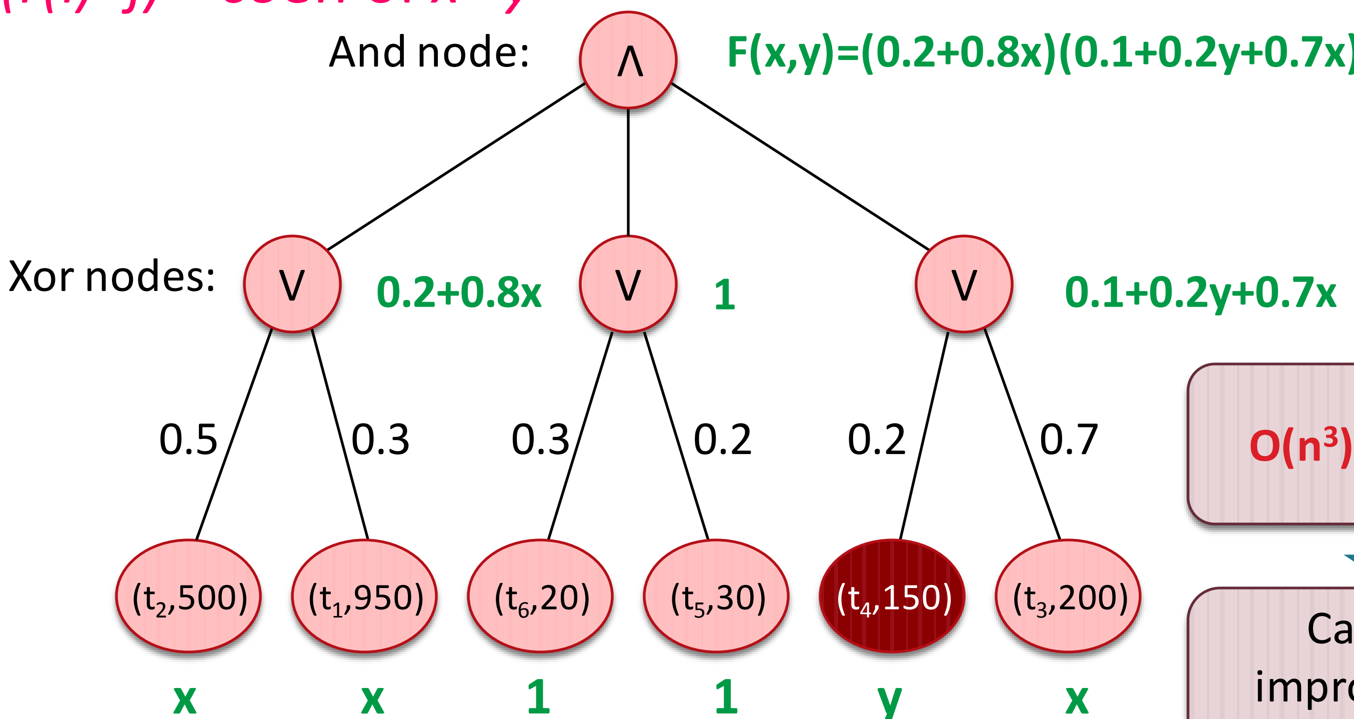
Pr=0.3*0.3*0.2=0.018

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 $x^{j-1}y$



$O(n^3)$ overall

Can be improved to $O(n^2 \log^2 n)$

Computing PRF$^e(\alpha)$: Probabilistic And/Xor Trees

Generating Function: $\mathcal{F}^i(x,y) = \sum_j c'_j x^j + (\sum_j c_j x^{j-1})y$

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

Therefore: $\Upsilon(t_i) = \sum_{j=1}^{n} \alpha^j c_j$

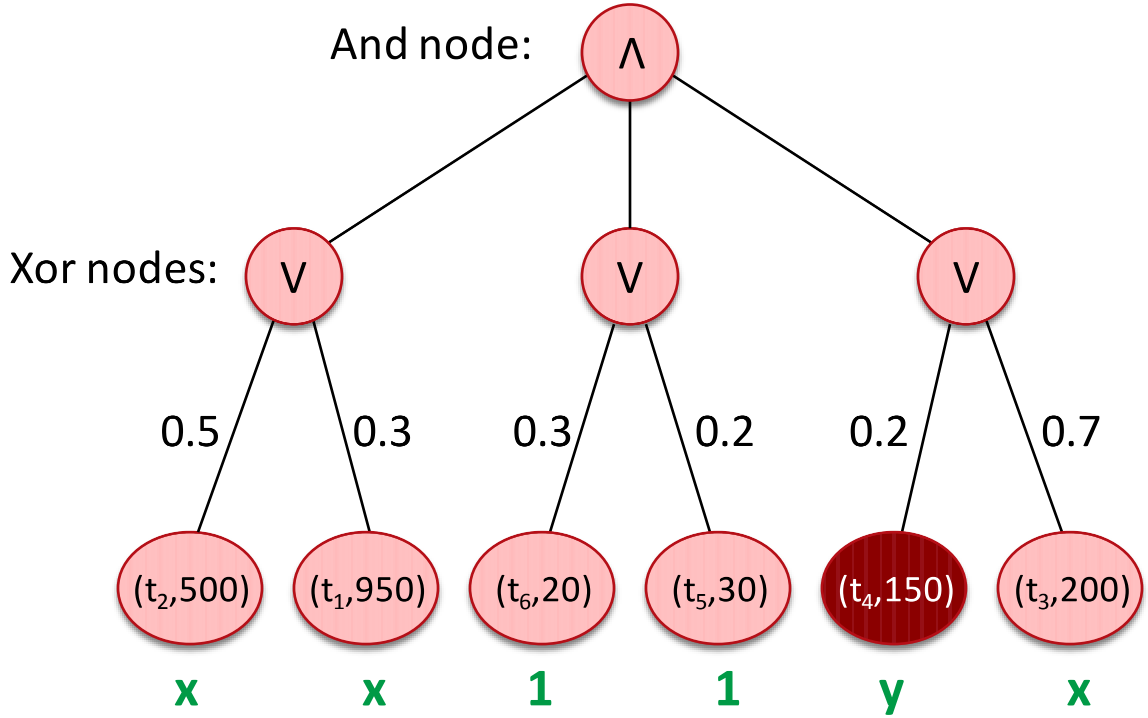
$$\begin{aligned} \Upsilon(t_i) &= \sum_j c_j \alpha^j = \Big(\sum_j c'_j \alpha^j + (\sum_j c_j \alpha^{j-1})\alpha\Big) - \sum_j c'_j \alpha^j \\ &= \mathcal{F}^i(\alpha,\alpha) - \mathcal{F}^i(\alpha,0). \end{aligned}$$

No need to expand the polynomial !!

Computing PRF$^e(\alpha)$: Probabilistic And/Xor Trees

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

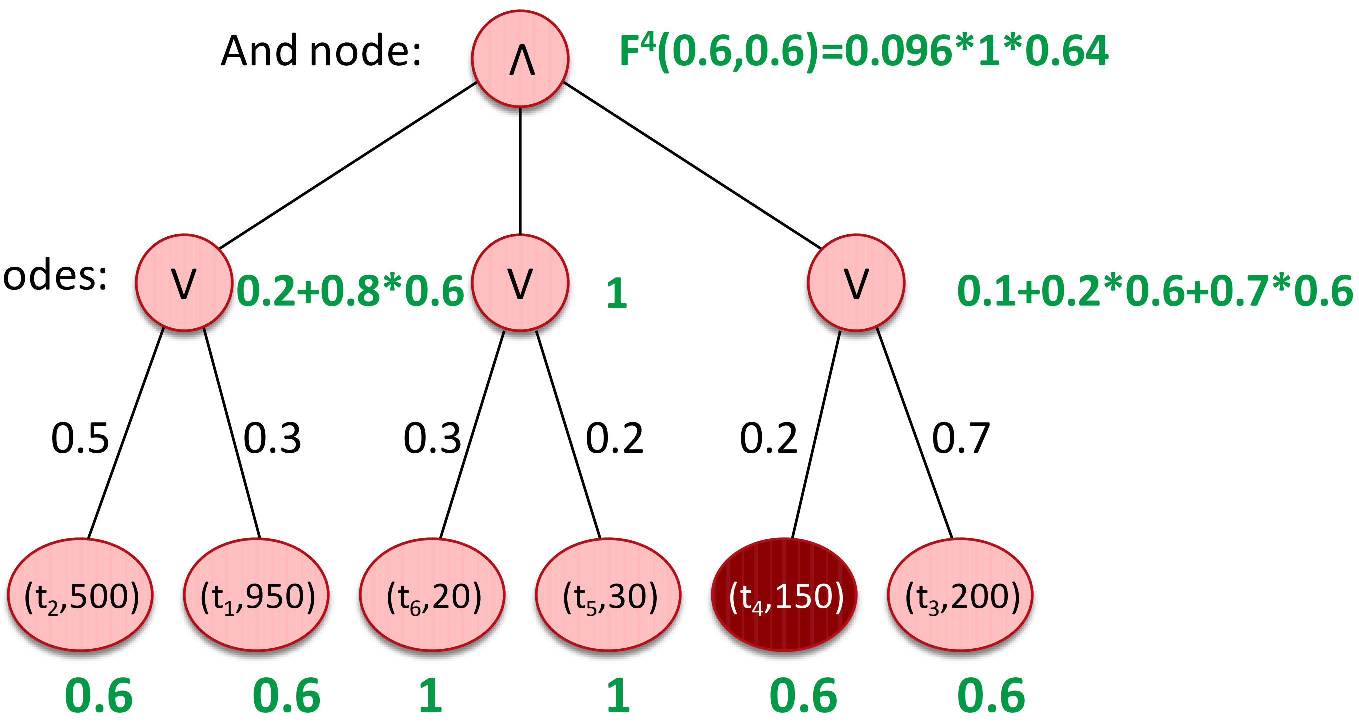
E.g. α=0.6. We are computing $F^4(0.6,0.6)$



Computing PRF$^e(\alpha)$: Probabilistic And/Xor Trees

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

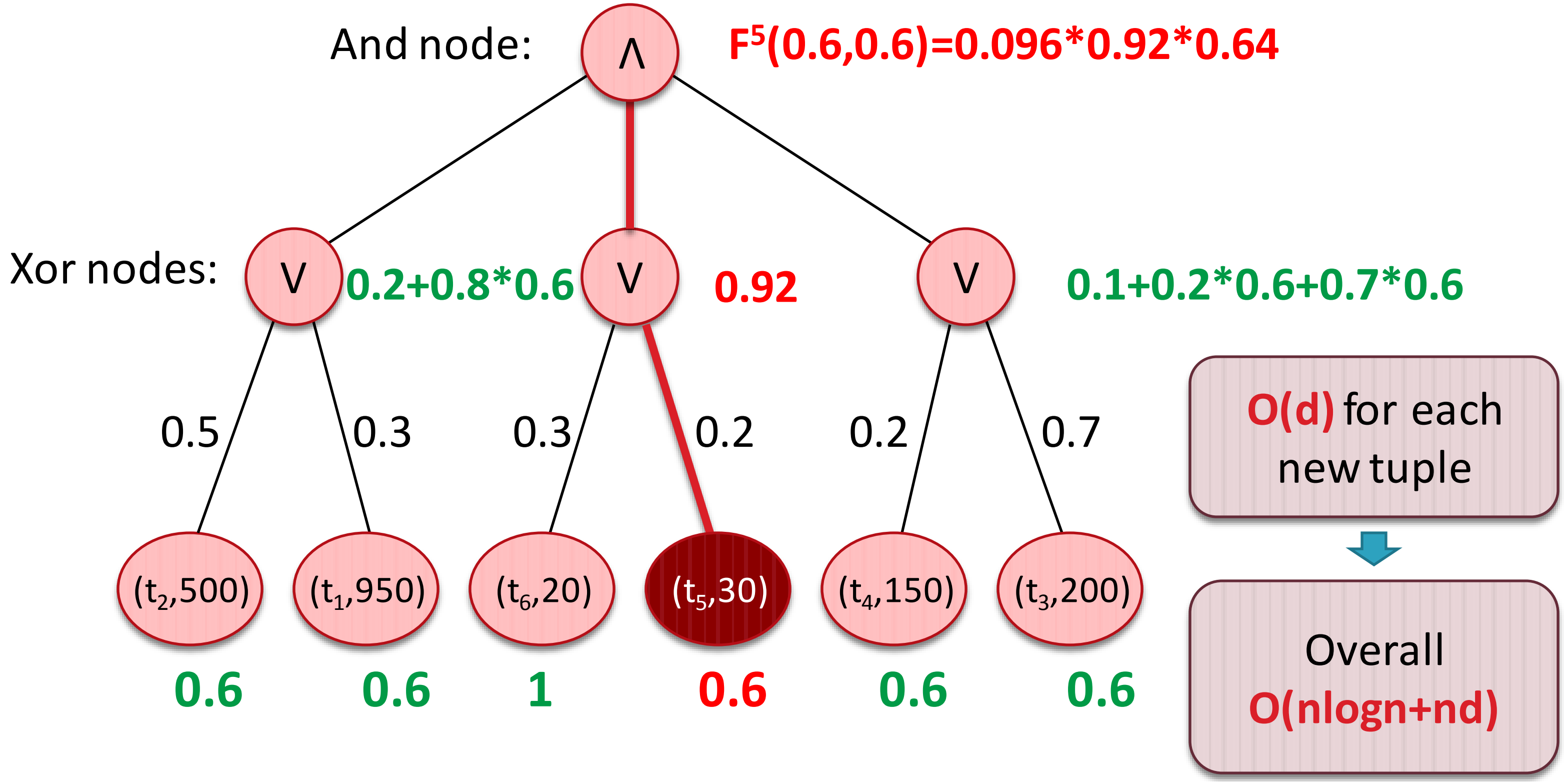
E.g. α=0.6. We are computing $F^4(0.6,0.6)$



Computing PRF$^e(\alpha)$: 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. α=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 $\mathrm{PRF}^{e}(\alpha)$
  - Probabilistic And/Xor Trees
- Approximating Ranking Functions
- Other Results
- Experiments

Approximating Ranking Functions

Approximating PRF^{ω} using PRF^{e}

- Suppose $\omega(i) \approx \sum_{l=1}^{L} u_l \alpha_l^i$

$$\Upsilon(t) = \sum_i \omega(i) \Pr(r(t) = i) \approx \sum_{l=1}^{L} u_l \left(\sum_i \alpha_l^i \Pr(r(t) = i) \right)$$

- Reduce to L individual PRF^{e} computations
- Running time : **O(nlogn+nL)** (as opposed to $O(n^2)$)

Approximating Ranking Functions

- How to approximate $\omega()$ by a linear combination of exponentials? $\omega(i) \approx \sum_{l=1}^{L} u_l \alpha_l^i$
- A scheme based on Discrete Fourier Transformation

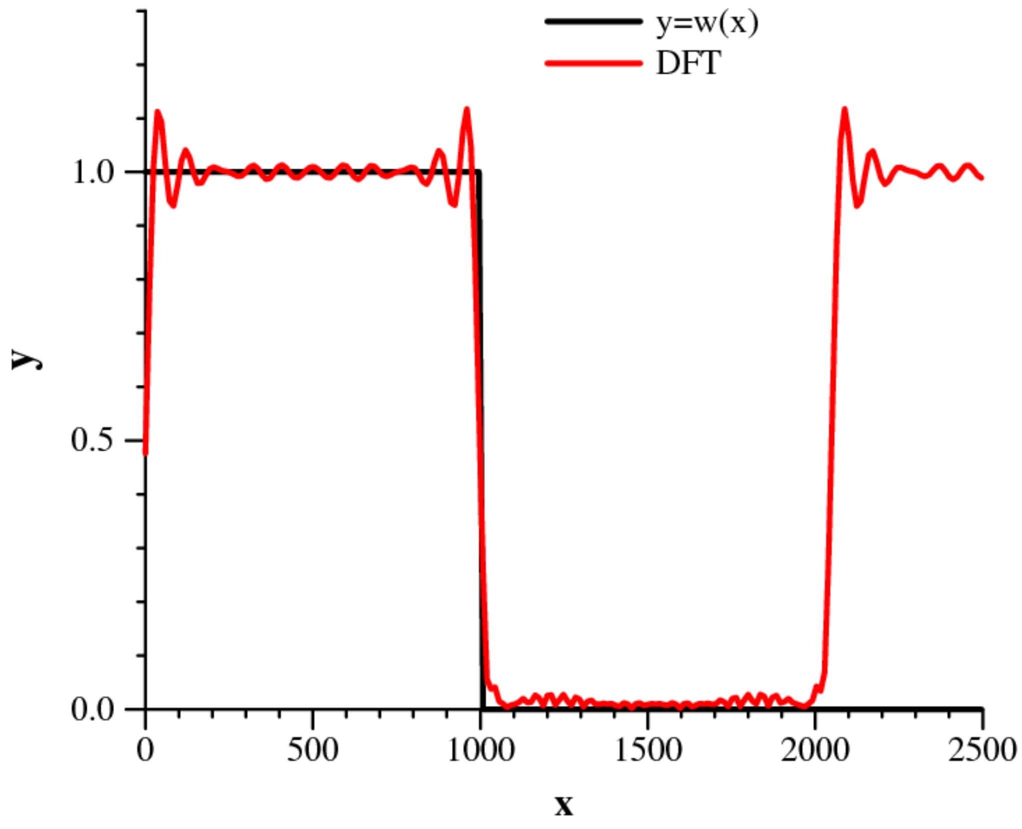
$$\omega(i) = \frac{1}{N} \sum_{k=0}^{N-1} \psi(k) e^{\frac{2\pi \jmath}{N} ki} \qquad i = 0, \ldots, N-1.$$

$\psi(0),\ldots,\psi(N-1)$ is the DFT of $\omega(0),\ldots,\omega(N-1)$

- 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



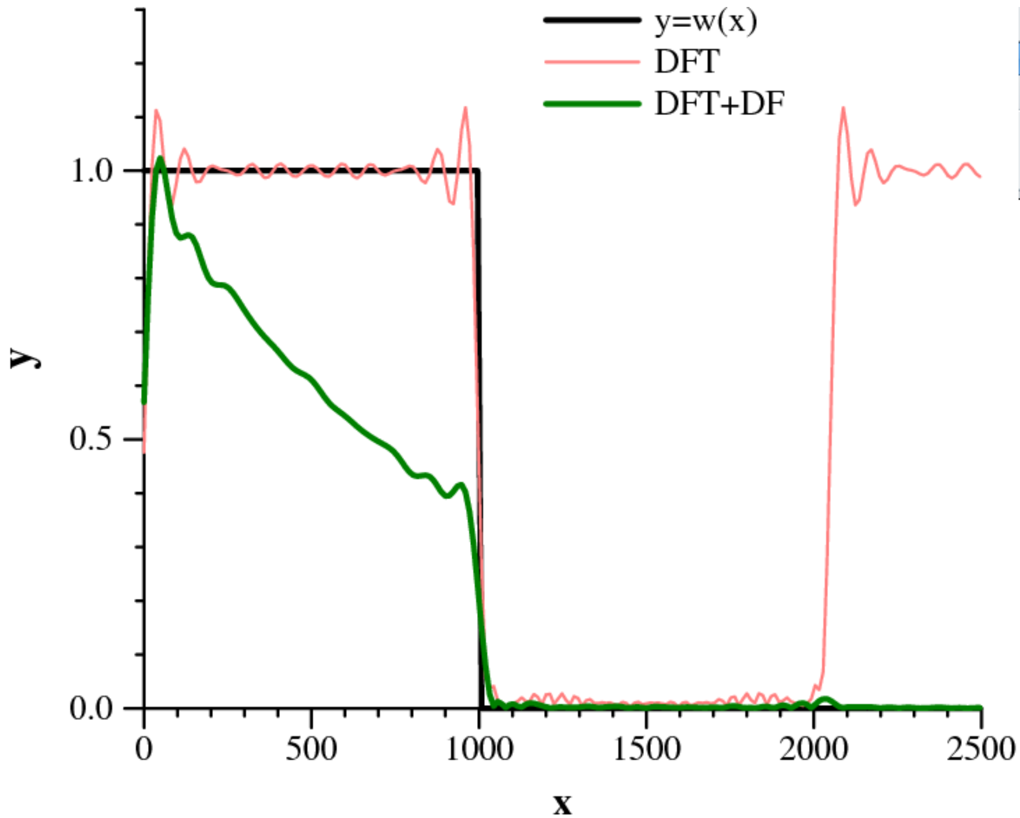
y=w(x)

DFT : Periodic !

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

Approximating Ranking Functions

Adding a **damping factor** η : $\tilde{\omega}^{DFT+DF}(i) = \eta^i \frac{1}{N} \sum_{k=0}^{L-1} \psi(k)(\eta e^{\frac{2\pi \jmath}{N}k})^i$



y=w(x)

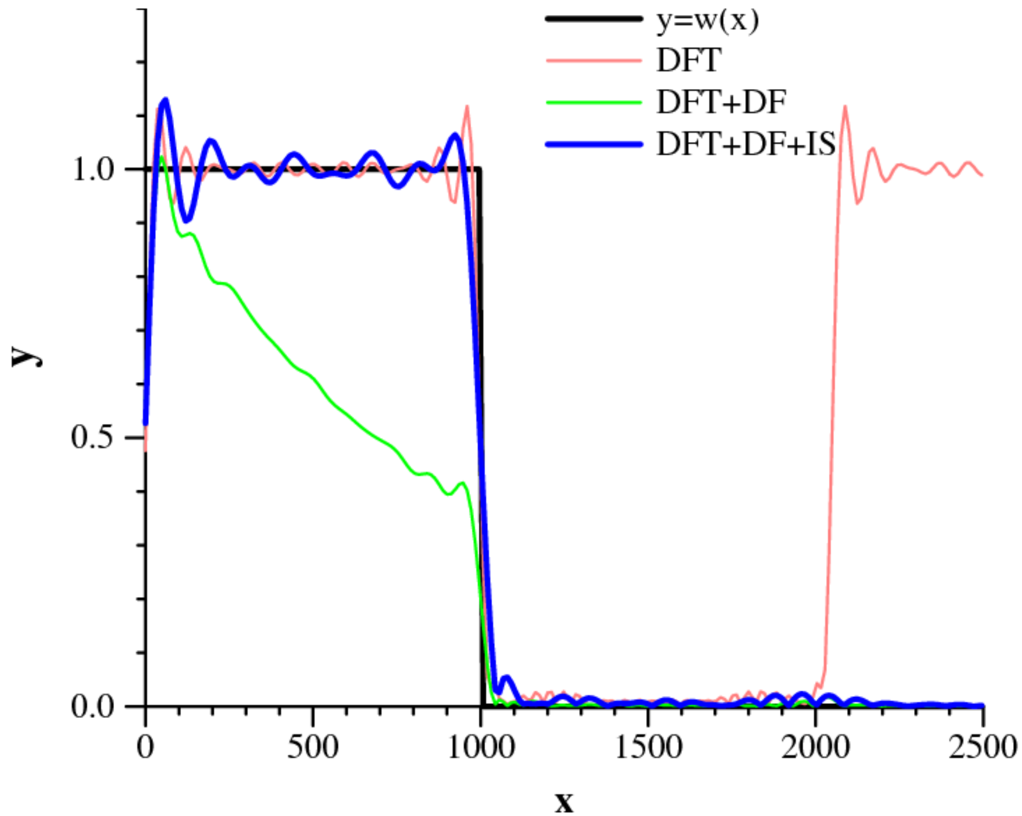
DFT : Periodic !

DFT+**Damping Factor**: Biased approximation

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

Approximating Ranking Functions

Initial Scaling : Perform DFT on a scaled sequence $\eta^{-i}\omega(i) \quad i = 0, \ldots, N-1$



y=w(x)

DFT : Periodic !

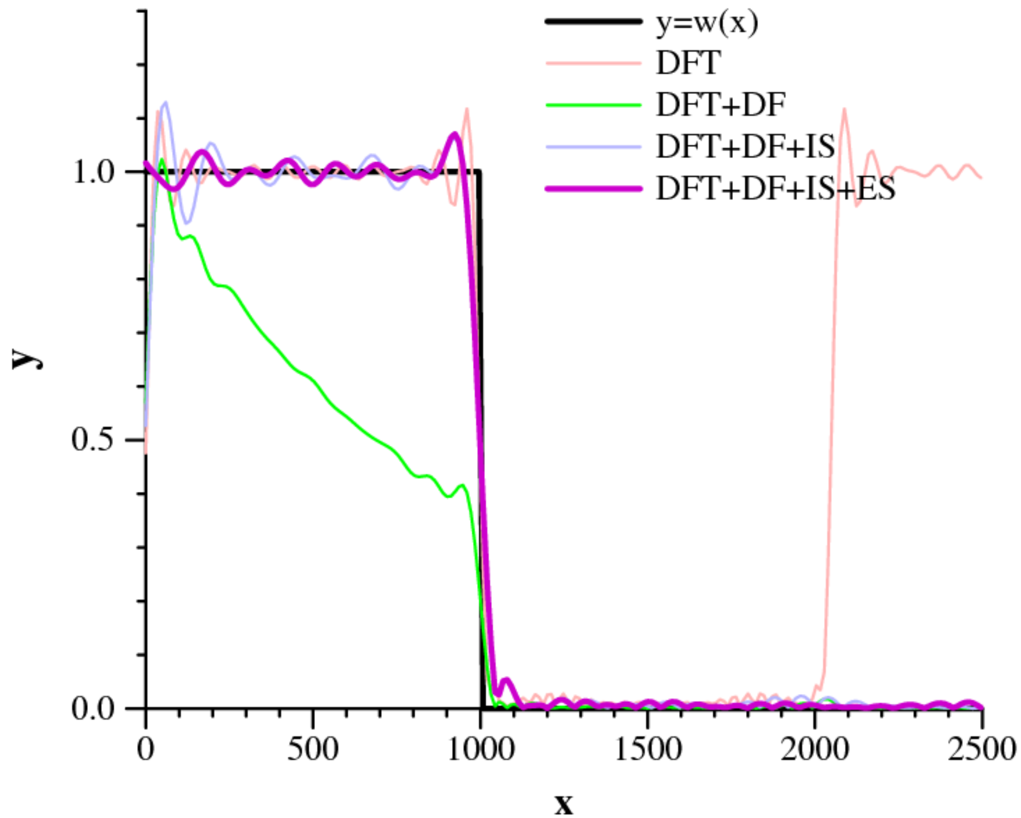
DFT+Damping Factor : Biased approximation

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 !

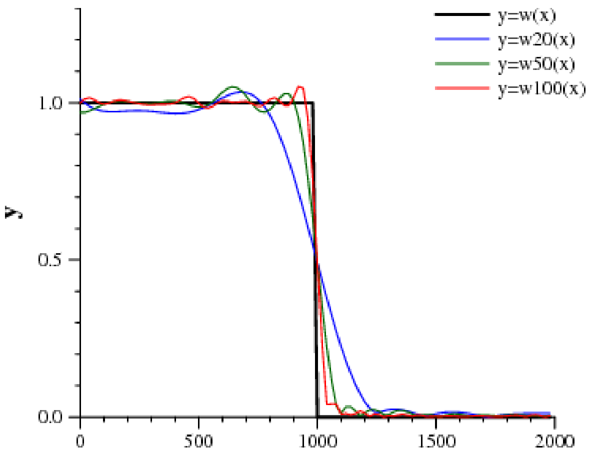
DFT+Damping Factor : Biased approximation

DFT+DF+ **Initial Scaling** : Unbiased

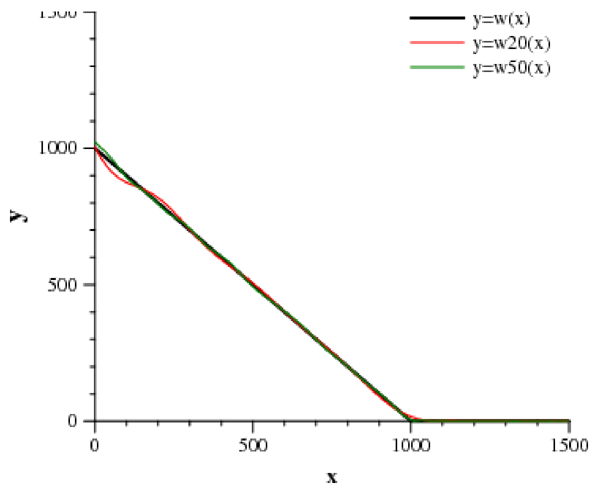
DFT+DF+IS+ **Extending&Shifting**: Unbiased and better quality around origin

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

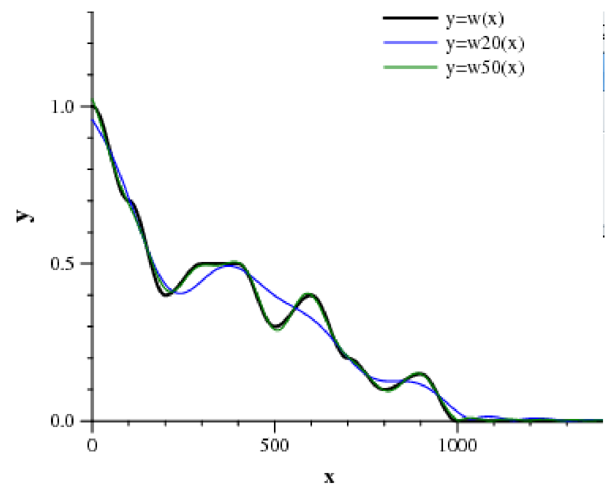
Approximating Ranking Functions



Approximations of the step function



Approximating w(i)=1000-i for i=0...1000, w(i)=0 for i>1000



Approximating an arbitrary smooth function

Outline

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

Other Results

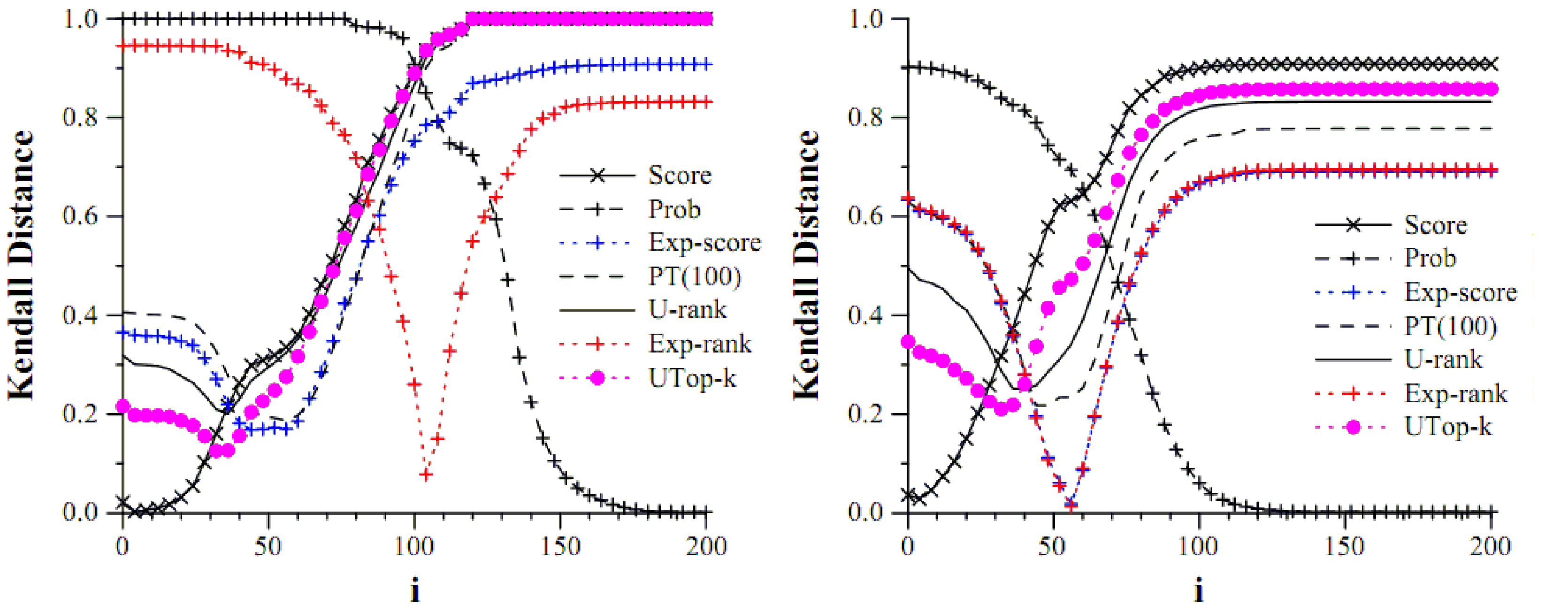
- Learning the weight function for PRF$^{\omega}$ based user preferences
- A binary search-like heuristic for learning PRF$^{e}(\alpha)$
- PRF computation on graphical models
  - A polynomial time algorithm when the junction tree has bounded treewidth
    – A nontrivial dynamic program combined with the generating function method.

Outline

- Prior Proposals for Top-*k* Queries over ProbDB
- Parameterized Ranking Functions
- Computing PRF
  - Independent Tuples
  - Computing $\mathrm{PRF}^{e}(\alpha)$
  - Probabilistic And/Xor Trees
- Approximating Ranking Functions
- Other Results
- Experiments

Experiments

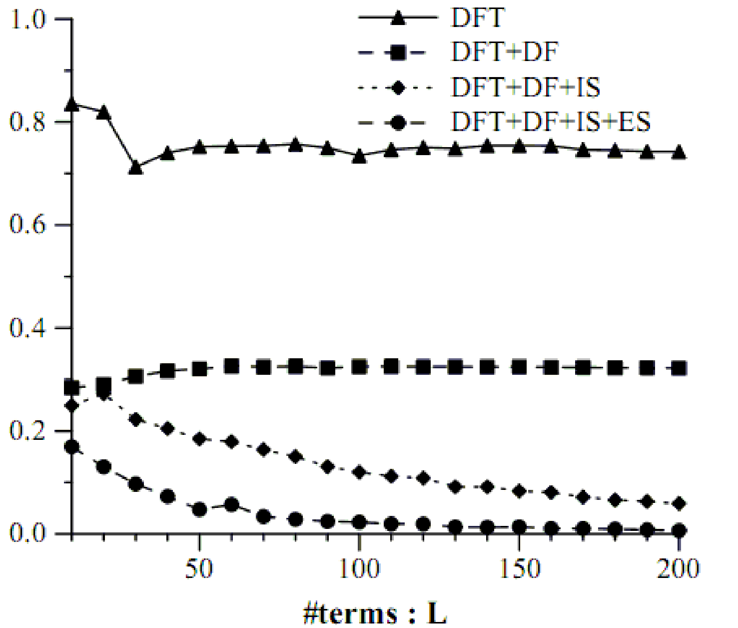
Comparing PRF^e with other ranking functions for varying value of α



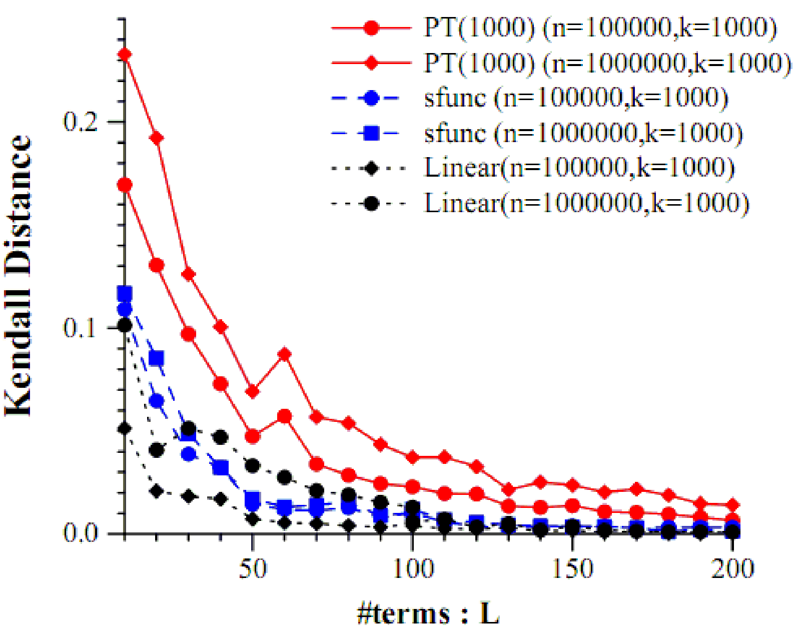
Approximating with PRF-e (a=1-0.9^i): (i) IIP-100000, k=100; (ii) Syn-IND-1000, k=100

Experiments

Approximation quality for approximating different ranking functions using PRF^e functions



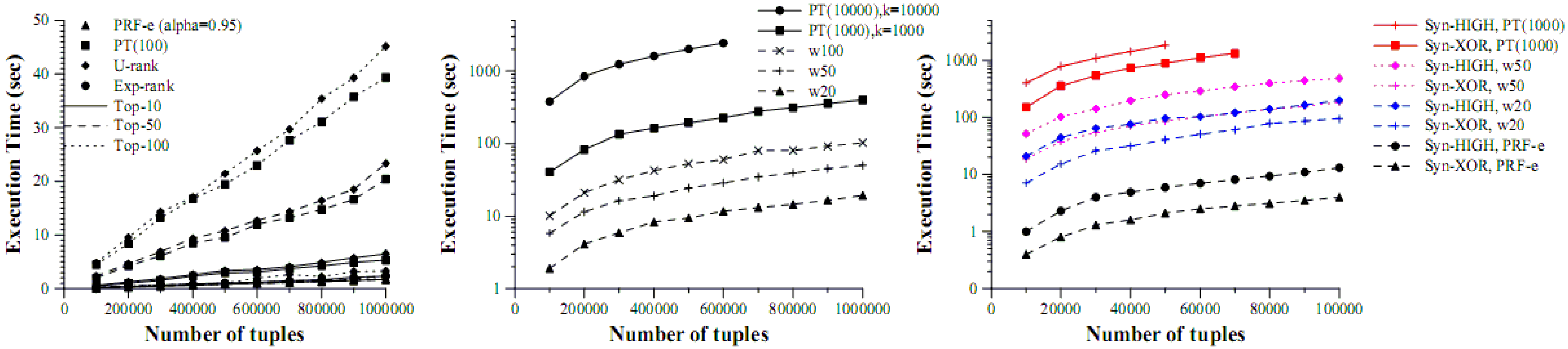
Approximating PT(1000)-1000 (n=100000)



No. of Terms vs Approximation Quality

Experiments

Execution Time



(i) (ii) (iii) k=1000

Conclusions

- Proposed a unifying framework for ranking over probabilistic databases through:
  - *Parameterized* ranking functions
  - Incorporation of user feedback
- Designed highly efficient algorithms for computing PRF and PRF^e
- Developed novel approximation techniques for approximating PRF^w with PRF^e