CMSC828E
Ranking in Probabilistic Databases

Jian Li

University of Maryland, College Park, USA

Sep, 2010

Outline

- Ranking in probabilistic databases is hard
  - Many proposals
  - Consensus answers
- The overview of our approach
- Parameterized Ranking Function (PRF)
  - Relation with other semantics
- Algorithm (The generating function method)
- Approximation
  - using a single PRFe
  - using a linear combination of PRFe
- Learning the weight function

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

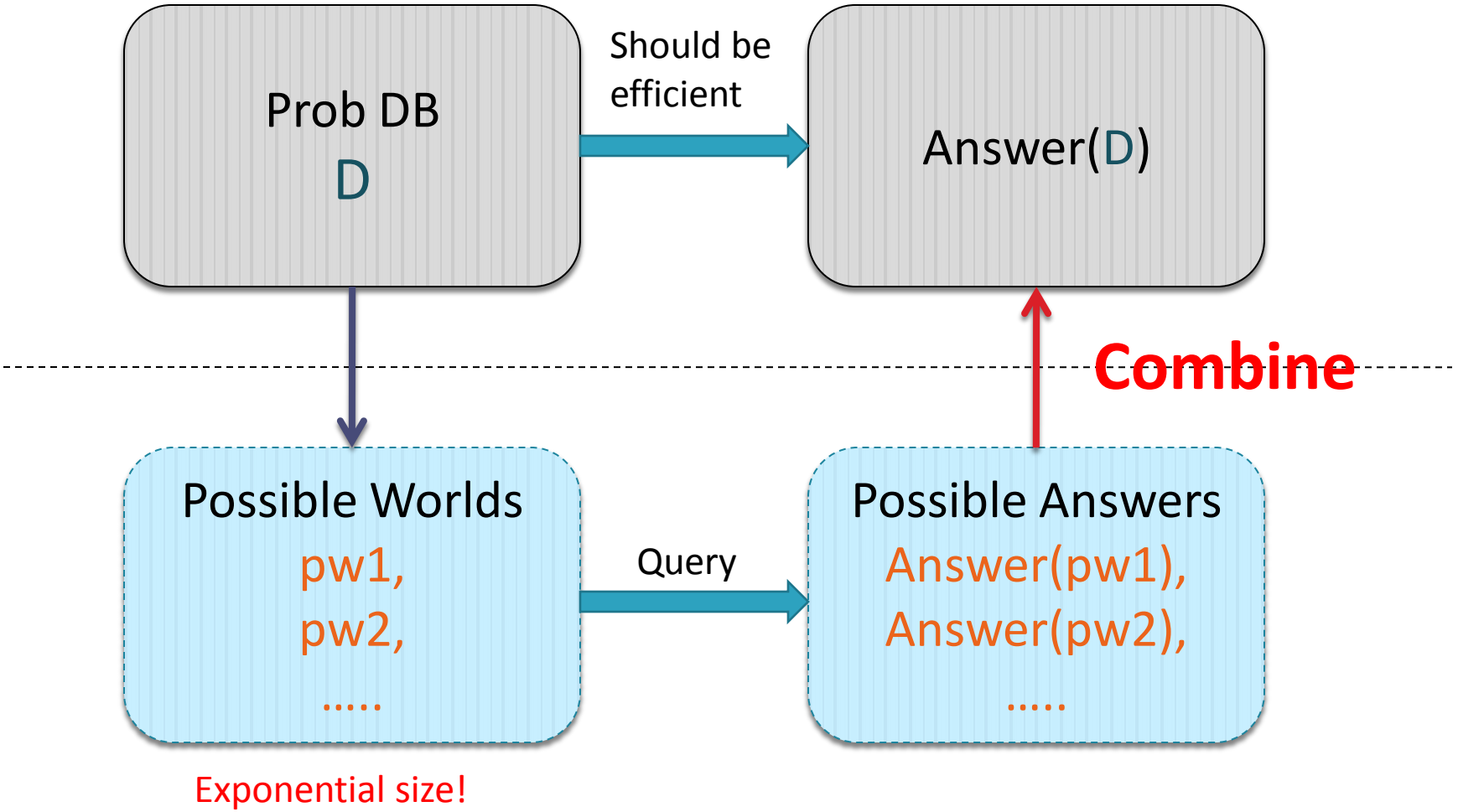
w.p. 0.096

pw3

ID	Score
t_2	150
t_3	100

w.p. 0.256

Semantics of Query Processing



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.

How to combine all possible top-k answers??

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

- Ranking in probabilistic databases is hard
  - Many proposals
  - Consensus answers
- The overview of our approach
- Parameterized Ranking Function (PRF)
  - Relation with other semantics
- Algorithm (The generating function method)
- Approximation
  - using a single PRFe
  - using a linear combination of PRFe
- Learning the weight function

Some Proposals

- Expected Score
  - Return k tuples *t* with the highest *score(t)Pr(t)*
- 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/GT-k) [Hua et al. '08] [Zhang et al. '08]
  - Return all tuples *t* s.t. $Pr(r(t) \leq k) \geq \theta$
- Expected Rank [Cormode et al. '09]
  - Return *k* tuples *t* with smallest $E[r(t)] = \sum Pr(pw)\, r_{pw}(t)$

Some Proposals

- Expected Score
  - Return k tuples *t* with the highest *score(t)Pr(t)*

ID	Score	Prob	Exp-score
t_1	200	0.2	100
t_2	150	0.8	120
t_3	100	0.4	40

Ranking: t_2, t_1, t_3

Some Proposals

- U-top-k [Soliman et al. '07]
  - Returns the most probable top *k*-answer

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

Possible worlds	Prob
t_1, t_2, t_3	0.064
t_1, t_2	0.096
t_1, t_3	0.016
t_2, t_3	0.265
t_1	0.024
t_2	0.384
t_3	0.064
Φ	0.096

K=2

Top2	Prob
t_1, t_2	0.160
t_1, t_3	0.016
t_2, t_3	**0.265**
t_1	0.024
t_2	0.384
t_3	0.064

Top-2: t_2 ,t_3

Some Proposals

- U-rank-k [Soliman et al. '07]
  - At rank *i*, return the tuple with max prob of being rank *i*

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

Possible worlds	Prob
t_1, t_2, t_3	0.064
t_1, t_2	0.096
t_1, t_3	0.016
t_2, t_3	0.265
t_1	0.024
t_2	0.384
t_3	0.064
Φ	0.096

	1	2	3
t_1	0.2	0	0
t_2	0.649	0.16	0
t_3	0.064	0.281	0.064

Positional probabilities

Ranking: t_2, t_3, t_3

Some Proposals

- Probabilistic Threshold (PT) [Hua et al. '08]
  - Return all tuples *t* s.t. $Pr(r(t) \leq k) \geq \theta$

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

Possible worlds	Prob
t_1, t_2, t_3	0.064
t_1, t_2	0.096
t_1, t_3	0.016
t_2, t_3	0.256
t_1	0.024
t_2	0.384
t_3	0.064
Φ	0.096

K=2

ID	Prob(r(t)≤2)
t_1	0.2
t_2	0.8
t_3	0.336

Ranking: t_2, t_3, t_1

Some Proposals

- Expected Rank [Cormode et al. '09]
  - Return k tuples t with smallest $E[r(t)] = \sum Pr(pw)\, r_{pw}(t)$ (if t is not in pw, $r_{pw}(t) = |pw| + 1$.

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

Possible worlds	Prob
t_1, t_2, t_3	0.064
t_1, t_2	0.096
t_1, t_3	0.016
t_2, t_3	0.256
t_1	0.024
t_2	0.384
t_3	0.064
Φ	0.096

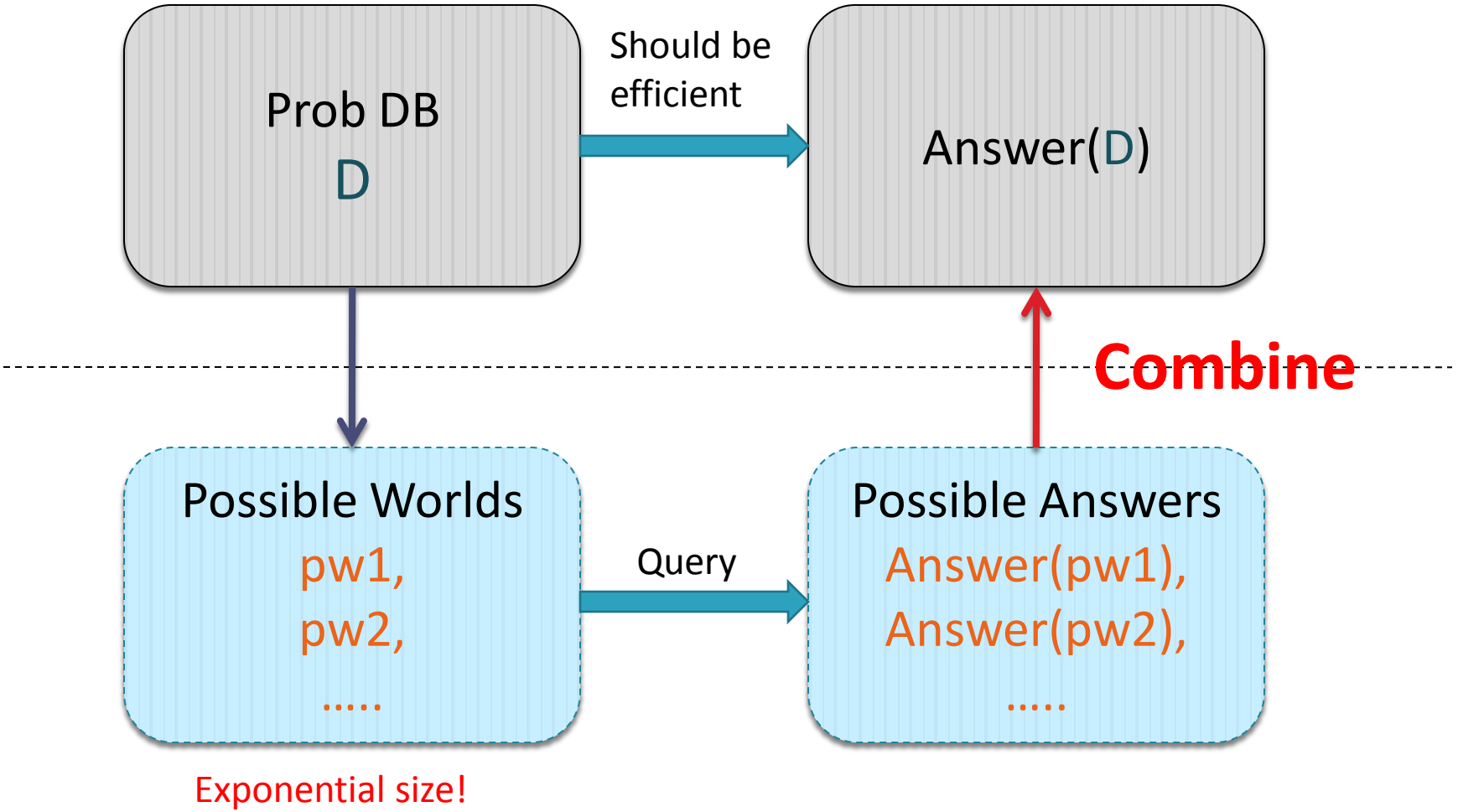
K=2

ID	E[r(t)]
t_1	1.96
t_2	1.28
t_3	2

Ranking: t_2, t_1, t_3

$E(r(t_1)) = (0.064+0.096+0.016+0.024+0.096)*1+(0.384+0.064)*2+0.256*3=1.96$

Semantics of Query Processing



Outline

- Ranking in probabilistic databases is hard
  - Many proposals
  - Consensus answers
- The overview of our approach
- Parameterized Ranking Function (PRF)
  - Relation with other semantics
- Algorithm (The generating function method)
- Approximation
  - using a single PRFe
  - using a linear combination of PRFe
- Learning the weight function

Consensus Answers

- Think of each possible answers as a point in the space.
- Suppose d() is a distance metric between answers.
- **Consensus Answers:**

  A single deterministic answer

$$\tau = \arg\min_{\tau' \in} \{\mathbb{E}[d(\tau', \tau_{pw})]\}$$

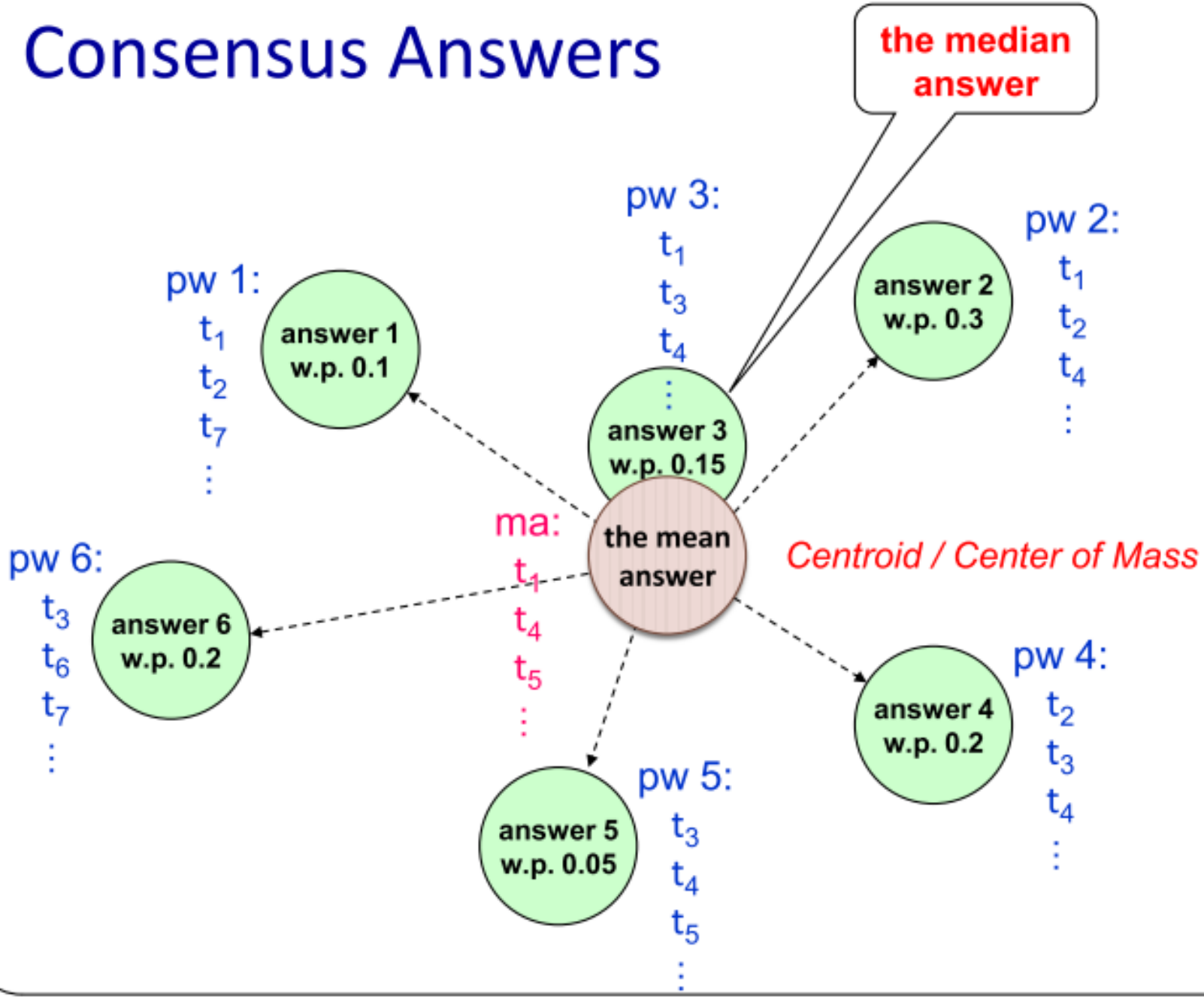
  where τ_{pw} is the answer for the possible world pw

  Mean answer: Ω is the set of all feasible answer

  Median answer: Ω is the set of all possible answer

Consensus Answers for Queries over Probabilistic Databases.
Jian Li and Amol Deshpande. PODS 2009

Consensus Answers



Consensus Top-k Answer

- Distance function: Symmetric difference

$$d(\tau_1, \tau_2) = |\tau_1 \Delta \tau_2| = |(\tau_1 \setminus \tau_2) \cup (\tau_2 \setminus \tau_1)|$$

- Spearman footrule distance

- Kendall's tau distance

Consensus Answer

- The idea also applies to other queries.
- E.g. aggregate queries:

  SELECT groupname, count(*) FROM R GROUP BY groupname

x-tuple

Key	groupname	Prob
1	1	0.5
1	2	0.5
2	2	0.4
2	3	0.6
…	…	…

groupname	count
1	1
2	1
3	0
…	…

Possible answer 1

groupname	count
1	1
2	0
3	1
…	…

Possible answer 2

.....

- Distance: Square vector distance
- Mean answer: Trivial. Take average for each group.
- Median answer: More involved.

Consensus Answer

- E.g. Clustering

- Distance function: Consensus clustering distance.

  $d(\tau_1, \tau_2)$= # (pairs that are clustered together in one solution but separated in another).

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	----

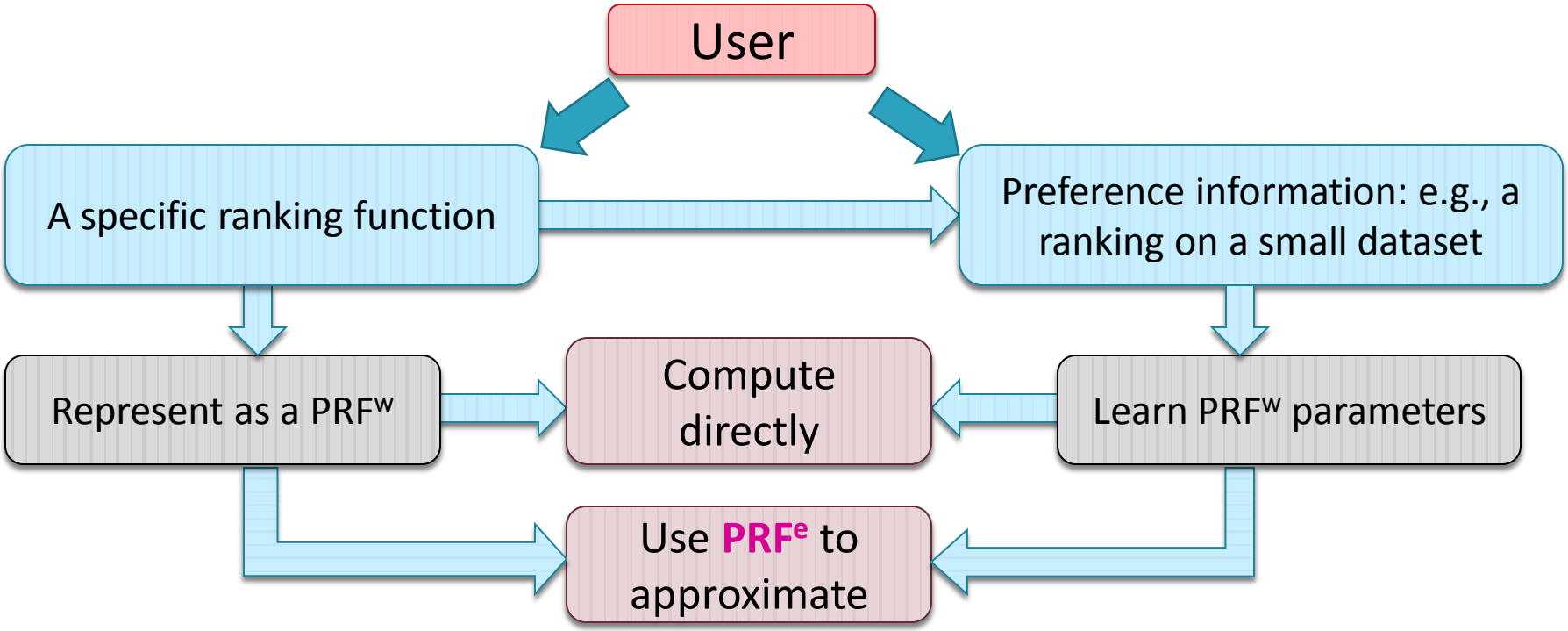
Synthetic Dataset: 100,000 tuples, Top-100

Outline

- Ranking in probabilistic databases is hard
  - Many proposals
  - Consensus answers
- The overview of our approach
- Parameterized Ranking Function (PRF)
  - Relation with other semantics
- Algorithm (The generating function method)
- Approximation
  - using a single PRFe
  - using a linear combination of PRFe
- Learning the weight function

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

- Ranking in probabilistic databases is hard
  - Many proposals
  - Consensus answers
- The overview of our approach
- Parameterized Ranking Function (PRF)
  - Relation with other semantics
- Algorithm (The generating function method)
- Approximation
  - using a single PRFe
  - using a linear combination of PRFe
- Learning the weight function

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.
  - *Exp-Rank:* $\omega(i) = n - i$ (if there is no tuple uncertainty)
- **PRF^{e}(α)**: $\omega(i)=\alpha^i$ where α *can be a real or a complex number*

PRF and Consensus Top-k Answer

- Distance function: Symmetric difference

$$d(\tau_1, \tau_2) = |\tau_1 \Delta \tau_2| = |(\tau_1 \setminus \tau_2) \cup (\tau_2 \setminus \tau_1)|$$

- Probabilistic threshold (PT-k): Find k tuples with largest *Pr(r(t)≤k).*
  - PRFw with weight function *ω(i)=1* if *i≤k* and *ω(i)=0* o.w.

- **THM:** The mean answer under symmetric difference metric is equivalent to PT-k.

PRF and Consensus Top-k Answer

- Distance function: Weighted Symmetric difference

$$d_\omega(\tau_1, \tau_2) = \sum_{i=1}^{k} \omega(i)\delta(\tau_2(i) \notin \tau_1).$$

ω: 9 8 7 6 5 4

τ_1 = A B C D E F
T_2 = K G C D E F
$d(\tau_1, \tau_2)$ = 9+8 = 17

τ_1 = A B C D E F
τ_2 = A B K E F G
$d(\tau_1, \tau_2)$ = 7+4 = 11

- PRFw with weight function *ω().*

- **THM:** The mean answer under weighted symmetric difference metric is equivalent to PRFw.

Outline

- Ranking in probabilistic databases is hard
  - Many proposals
  - Consensus answers
- The overview of our approach
- Parameterized Ranking Function (PRF)
  - Relation with other semantics
- Algorithm (The generating function method)
- Approximation
  - using a single PRFe
  - using a linear combination of PRFe
- Learning the weight function

Computing PRF: Independent Tuples

Computing Pr(r(t_i) = j), for a given tuple t_i and a given rank j

$T_{i-1}=\{t_1,t_2,\ldots,t_{i-1}\}$ i.e., the set of tuples with scores higher than t_i

$$\begin{aligned}\Pr(r(t_i)=j) &= \Pr(t_i) \sum_{pw:j-1 \text{ tuples in } T_{i-1} \text{ exists}} \Pr(pw) \\ &= \Pr(t_i) \sum_{\substack{S\subseteq T_{i-1} \\ |S|=j-1}} \prod_{t\in S} \Pr(t) \prod_{t\in T_{i-1}\setminus S} (1-\Pr(t))\end{aligned}$$

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^j : *Pr(r(t_i)=j)*

Computing PRF: Independent Tuples

- **Algorithm:**
  - For i=1 to n
    - Expand $\mathcal{F}^i(x) = \sum_{j=1}^{n} \Pr(r(t_i) = j)x^j$
    - $\Upsilon(t_i) = \sum_{j=1}^{n} \omega(t_i, j)\Pr(r(t_i) = j)$
  - Return k tuples with largest $|\Upsilon_\omega|$ values

Can be improved to **O(n)**

O(n) time

O(n²) overall

F^i and F^{i-1} differ in two multiplicative terms

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

Computing PRF$^e(\alpha)$: Independent Tuples

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

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

No need to expand the polynomial !!

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

O(1)

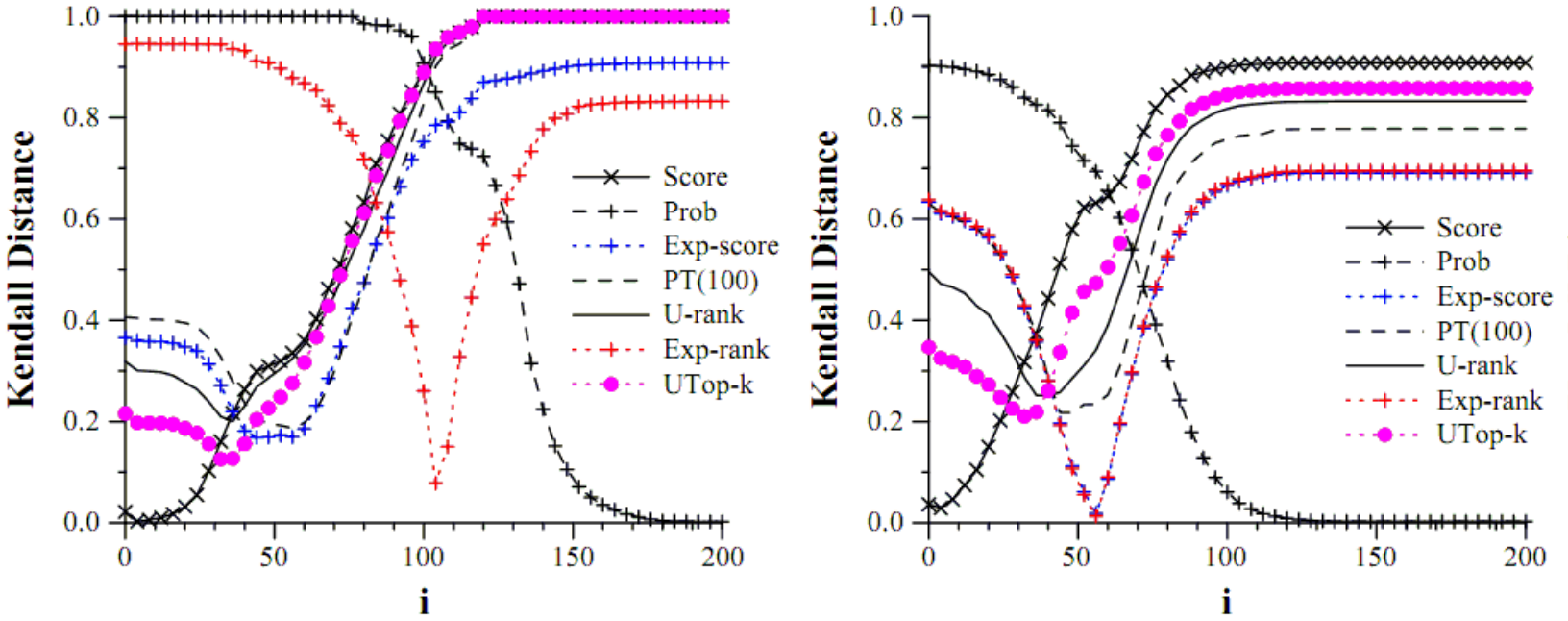
O(n) overall

Outline

- Ranking in probabilistic databases is hard
  - Many proposals
  - Consensus answers
- The overview of our approach
- Parameterized Ranking Function (PRF)
  - Relation with other semantics
- Algorithm (The generating function method)
- Approximation
  - using a single PRFe
  - using a linear combination of PRFe
- Learning the weight function

Approximating Other Ranking Functions Using PRFe

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

Outline

- Ranking in probabilistic databases is hard
  - Many proposals
  - Consensus answers
- The overview of our approach
- Parameterized Ranking Function (PRF)
  - Relation with other semantics
- Algorithm (The generating function method)
- Approximation
  - using a single PRFe
  - using a linear combination of PRFe
- Learning the weight function

Approximating Ranking Functions

Approximating PRF^{ω} using linear combination of 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

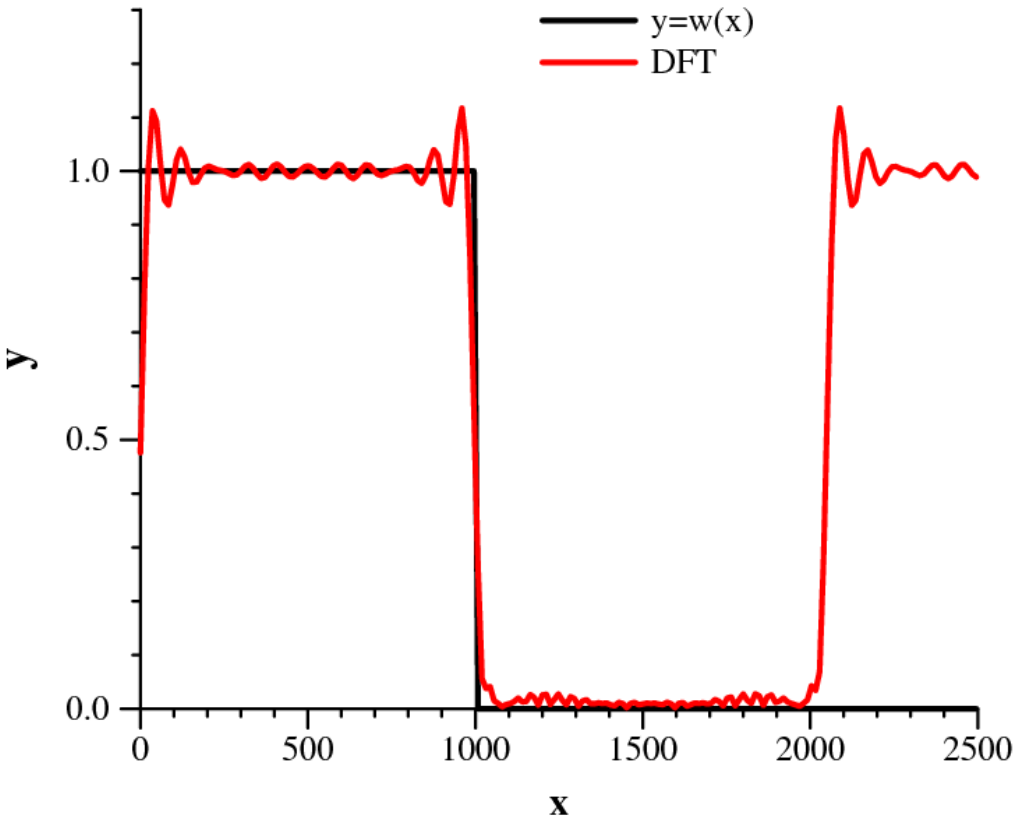
$$\omega(i) = \frac{1}{N} \sum_{k=0}^{N-1} \psi(k) e^{\frac{2\pi J}{N} ki} \qquad i = 0, \ldots, N-1.$$

ψ(0),…,ψ(N-1) is the DFT of ω(0),…,ω(N-1)

- Use the largest *L* DFT coefficients

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

Approximating Ranking Functions



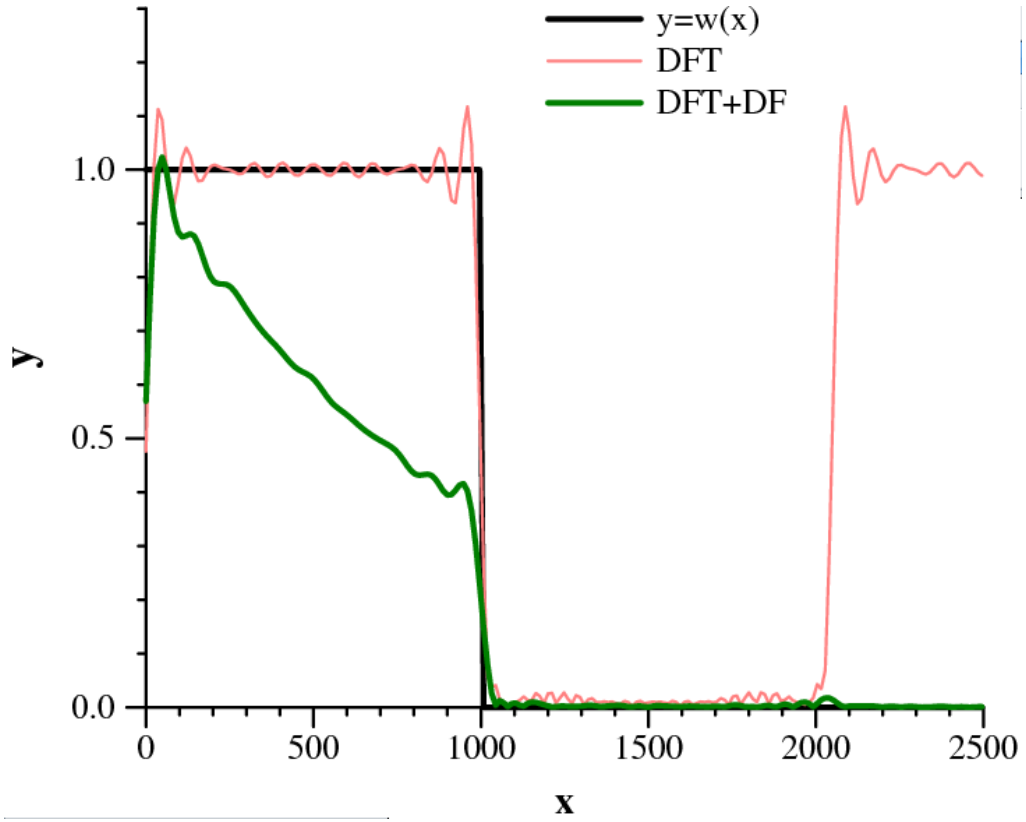
y=w(x)

DFT : Periodic !

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

Approximating Ranking Functions

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



y=w(x)

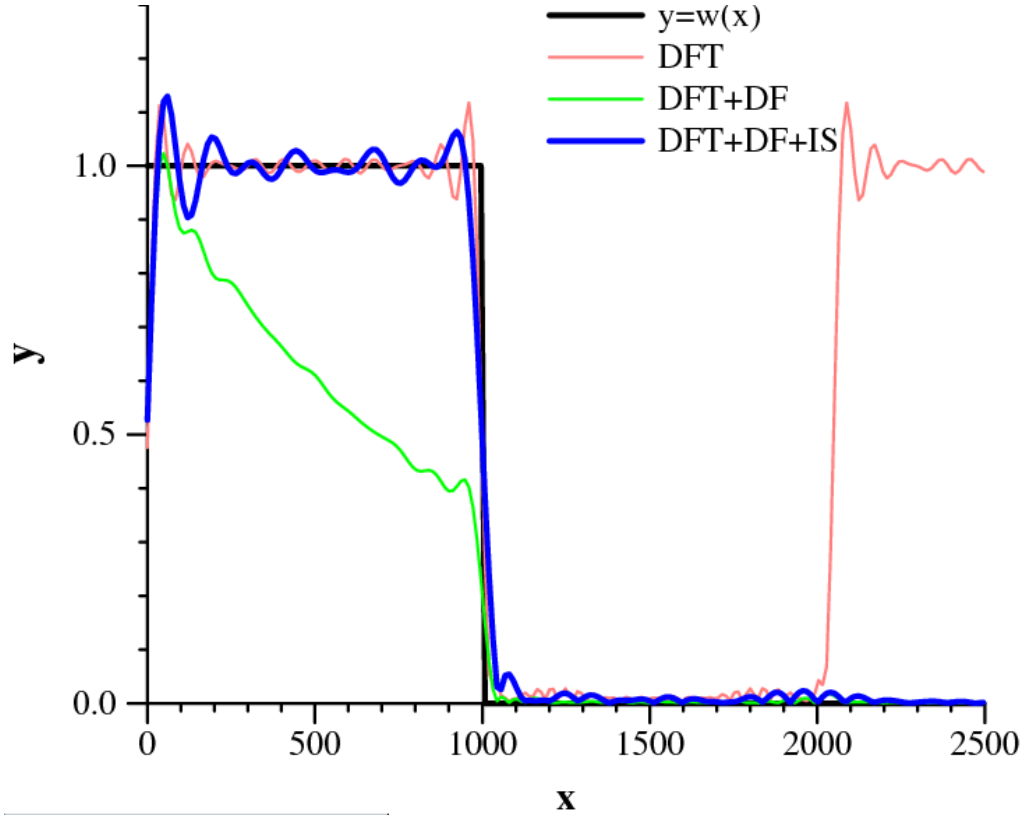
DFT : Periodic !

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 !

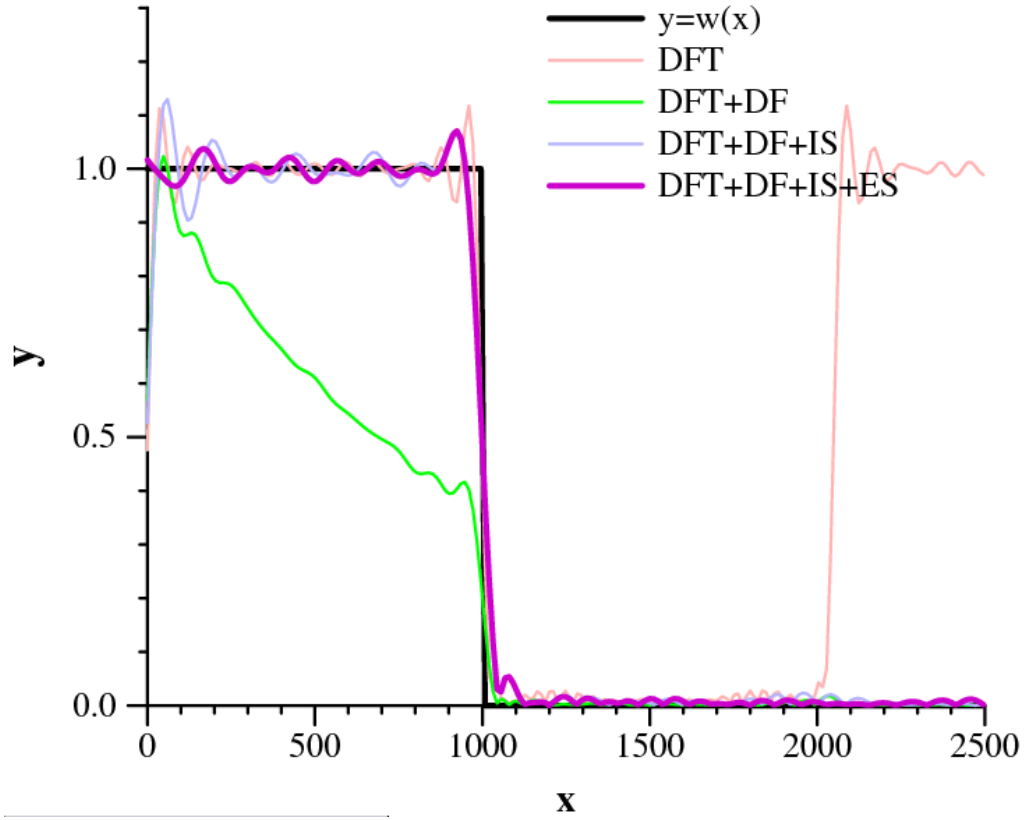
DFT+Damping Factor : Biased approximation

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 !

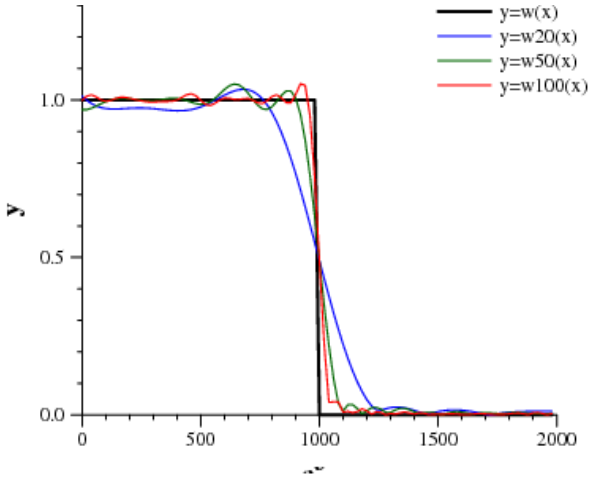
DFT+Damping Factor : Biased approximation

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

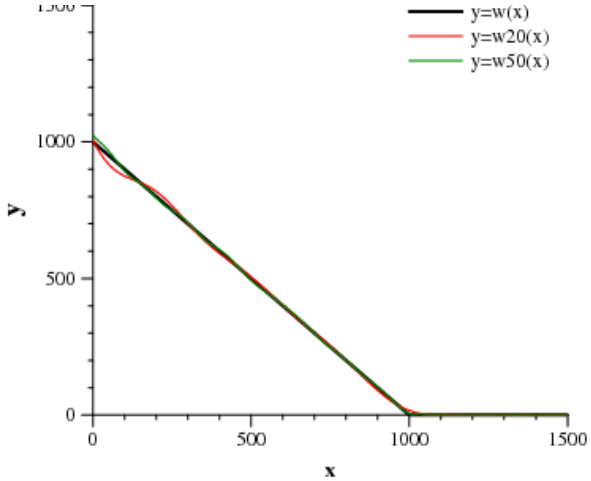
DFT+DF+IS+ **Extending&Shifting**: Unbiased and better quality around origin

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

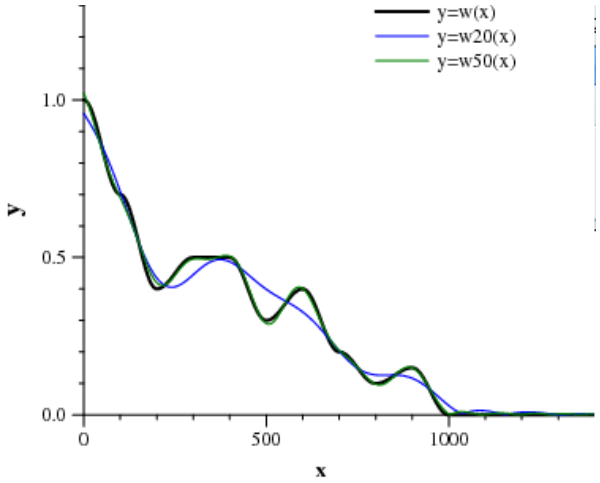
Approximating Ranking Functions



Approximations of the step function



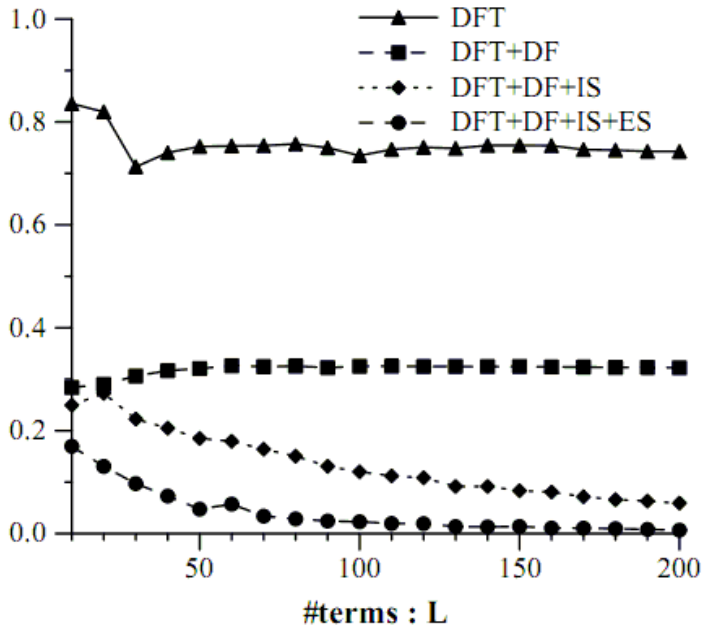
Approximating w(i)=1000-i for i=0...1000, w(i)=0 for i>1000



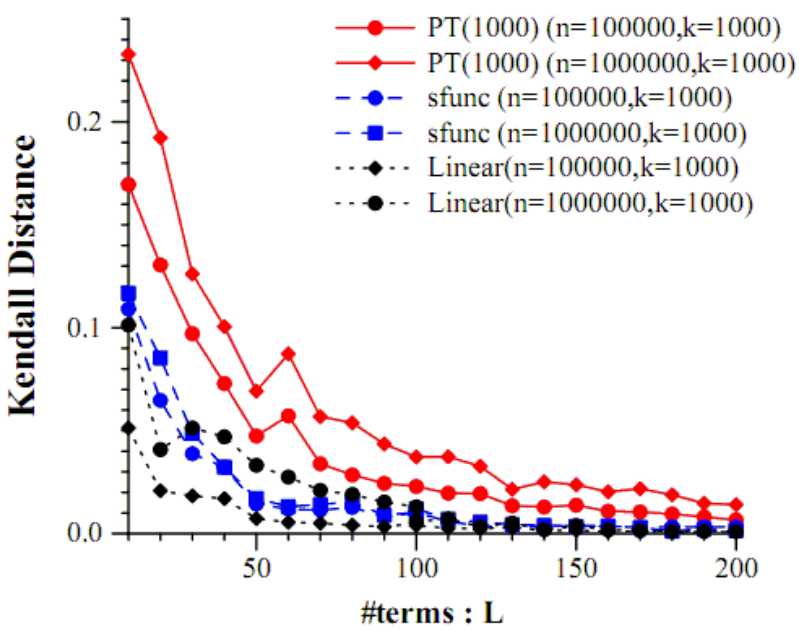
Approximating an arbitrary smooth function

Approximating Ranking Functions

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



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



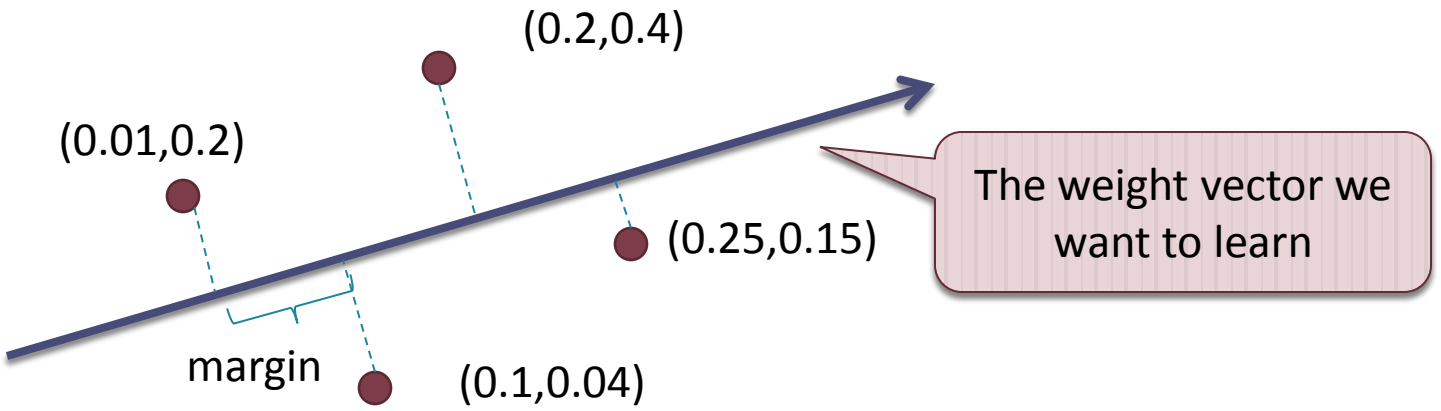
No. of Terms vs Approximation Quality

Outline

- Ranking in probabilistic databases is hard
  - Many proposals
  - Consensus answers
- The overview of our approach
- Parameterized Ranking Function (PRF)
  - Relation with other semantics
- Algorithm (The generating function method)
- Approximation
  - using a single PRFe
  - using a linear combination of PRFe
- Learning the weight function

Learn the Weight Function

- We can learn the weight from user feedbacks.
  - A feedback can be a total or partial ordering of the tuples.
- Use the positional probabilities as the features.
  - Feature vector: *{Pr(r(t)=1), Pr(r(t)=2), Pr(r(t)=3),…}.*
- Use Ranking-SVM to learn the weight.
  - Maximize the margin.



Note

- Texpoint 3.2.1.