# Top-k Queries over Uncertain Scores

Qing Liu, Debabrota Basu, Talel Abdessalem, Stéphane Bressan

NUS National University of Singapore

# Introduction

- Modern recommendation systems leverage some forms of collaborative user (crowd) sourced collection of information.

# Introduction

- Modern recommendation systems leverage some forms of collaborative user (crowd) sourced collection of information.
- Crowdsourcing Platforms
  - easily announce their needs to the crowd / get access to the information they need
  - choose the highest quality / most competitively priced

# Introduction

- Modern recommendation systems leverage some forms of collaborative user (crowd) sourced collection of information.
- Crowdsourcing Platforms
  - easily announce their needs to the crowd / get access to the information they need
  - choose the highest quality / most competitively priced
- Examples: TripAdvisor

# Introduction

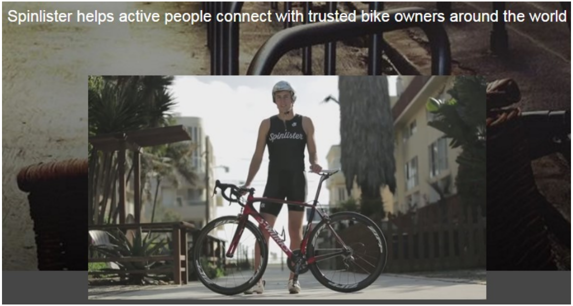
- Modern recommendation systems leverage some forms of collaborative user (crowd) sourced collection of information.
- Crowdsourcing Platforms
  - easily announce their needs to the crowd / get access to the information they need
  - choose the highest quality / most competitively priced
- Examples: TripAdvisor
  - collaborative user or crowdsourced collection of information, e.g., user generated ratings and reviews, to recommend travel plans and hotels, vacation rentals and restaurants.

# Introduction

- Crowdsourcing and Collaborative Economy:
  - communities or crowds rent, share, sell products or services

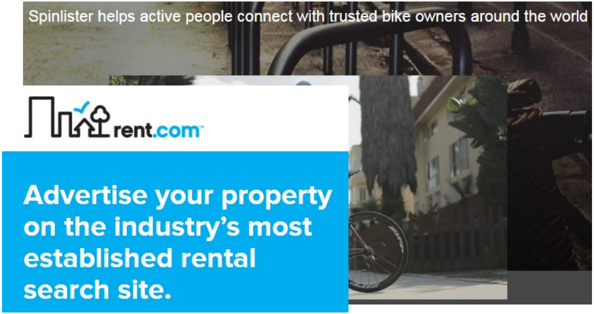
# Introduction

- Crowdsourcing and Collaborative Economy:
  - communities or crowds rent, share, sell products or services



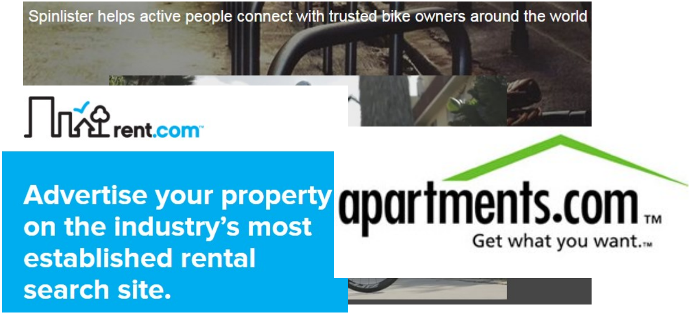
# Introduction

- Crowdsourcing and Collaborative Economy:
  - communities or crowds rent, share, sell products or services



# Introduction

- Crowdsourcing and Collaborative Economy:
  - communities or crowds rent, share, sell products or services



# Introduction

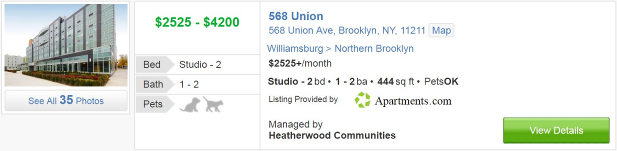
- Independent collection of information $\rightarrow$ uncertainty and diversity.

# Introduction

- Independent collection of information → uncertainty and diversity.
- Objects (services, vacation rentals and restaurants...) have uncertain scores (quality, price...).

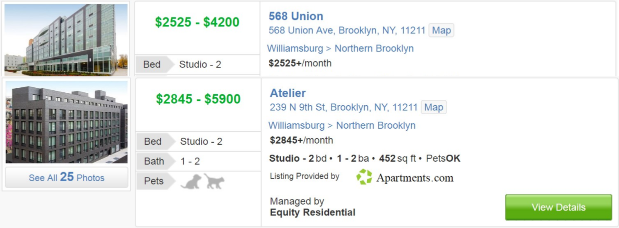
# Introduction

- Independent collection of information → uncertainty and diversity.
- Objects (services, vacation rentals and restaurants...) have uncertain scores (quality, price...).



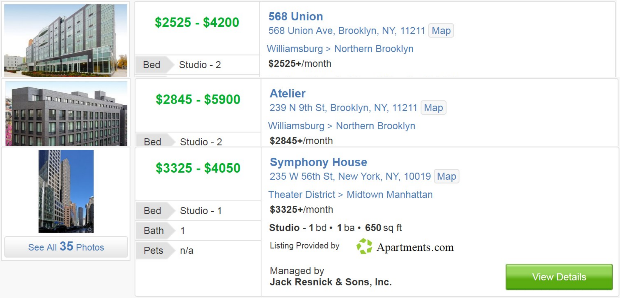
# Introduction

- Independent collection of information $\rightarrow$ uncertainty and diversity.
- Objects (services, vacation rentals and restaurants...) have uncertain scores (quality, price...).



# Introduction

- Independent collection of information $\rightarrow$ uncertainty and diversity.
- Objects (services, vacation rentals and restaurants...) have uncertain scores (quality, price...).

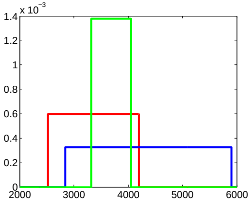


# Introduction

- Ranking is one of the building blocks of recommendation.
- A **top-k query** returns the sequence of the k objects with the highest scores, given a database of objects ranked by their scores for the feature of interest.

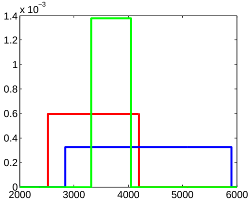
# Introduction

- Ranking is one of the building blocks of recommendation.
- A **top-k query** returns the sequence of the k objects with the highest scores, given a database of objects ranked by their scores for the feature of interest.
- Price of the apartments.



# Introduction

- Ranking is one of the building blocks of recommendation.
- A **top-k query** returns the sequence of the k objects with the highest scores, given a database of objects ranked by their scores for the feature of interest.
- Price of the apartments.




- With uncertain scores, a top-k query can only return an uncertain result.

# Related Work

- Soliman, Hyas and Ben-David [Soliman and Ilyas, 2009] study top-$k$ queries over objects with uncertain scores given as probability distributions.

# Related Work

- Soliman, Hyas and Ben-David [Soliman and Ilyas, 2009] study top-$k$ queries over objects with uncertain scores given as probability distributions.
- In this paper, we consider **probabilistic top-$k$ queries** under the top-k semantics as in [Soliman and Ilyas, 2009].

# Problem Definition

- $\mathcal{O}$: a set of $n$ objects;

# Problem Definition

- $\mathcal{O}$: a set of $n$ objects;
- $s(o_i)$: the score of an object $o_i \in \mathcal{O}$;

# Problem Definition

- $\mathcal{O}$: a set of $n$ objects;
- $s(o_i)$: the score of an object $o_i \in \mathcal{O}$;
- $X_i$: a random variable, equals to $s(o_i)$;

# Problem Definition

- $\mathcal{O}$: a set of $n$ objects;
- $s(o_i)$: the score of an object $o_i \in \mathcal{O}$;
- $X_i$: a random variable, equals to $s(o_i)$;
- $f_i$: bounded continuous probability density function of $X_i$;

# Problem Definition

- $\mathcal{O}$: a set of $n$ objects;
- $s(o_i)$: the score of an object $o_i \in \mathcal{O}$;
- $X_i$: a random variable, equals to $s(o_i)$;
- $f_i$: bounded continuous probability density function of $X_i$;
- $\pi^{(k)} = [o_1, \cdots, o_k]$: sequence of $k$ objects in $\mathcal{O}$;

# Problem Definition

- $\mathcal{O}$: a set of $n$ objects;
- $s(o_i)$: the score of an object $o_i \in \mathcal{O}$;
- $X_i$: a random variable, equals to $s(o_i)$;
- $f_i$: bounded continuous probability density function of $X_i$;
- $\pi^{(k)} = [o_1, \cdots, o_k]$: sequence of $k$ objects in $\mathcal{O}$;
- $Pr(\pi^{(k)})$: probability of $\pi^{(k)}$ be the top-$k$ sequence;

$$Pr(\pi^{(k)}) = \int_{-\infty}^{\infty} \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_k} f_1(x_1) \cdots f_n(x_n)\, dx_n \cdots dx_1 \tag{1}$$

# Problem Definition

- $\mathcal{O}$: a set of $n$ objects;
- $s(o_i)$: the score of an object $o_i \in \mathcal{O}$;
- $X_i$: a random variable, equals to $s(o_i)$;
- $f_i$: bounded continuous probability density function of $X_i$;
- $\pi^{(k)} = [o_1, \cdots, o_k]$: sequence of $k$ objects in $\mathcal{O}$;
- $Pr(\pi^{(k)})$: probability of $\pi^{(k)}$ be the top-$k$ sequence;

$$Pr(\pi^{(k)}) = \int_{-\infty}^{\infty} \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_k} f_1(x_1) \cdots f_n(x_n)\, dx_n \cdots dx_1 \tag{1}$$

- (Objective:) **Probabilistic top-$k$ sequence**: the $\pi^{(k)}$ that maximizes $Pr(\pi^{(k)})$.

# Solutions

- Naive: calculate $Pr(\pi^{(k)})$ for every possible sequence $\pi^{(k)}$ and returning the $\pi^{(k)}$ with the highest $Pr(\pi^{(k)})$.
  - $\frac{n!}{(n-k)!}$ possible sequences to examine.

# Solutions

- Naive: calculate $Pr(\pi^{(k)})$ for every possible sequence $\pi^{(k)}$ and returning the $\pi^{(k)}$ with the highest $Pr(\pi^{(k)})$.
  - $\frac{n!}{(n-k)!}$ possible sequences to examine.
- Branch-and-Bound [Soliman et al., 2010]: Prune some $\pi^{(k)}$s.
  - Worst case: $\frac{n!}{(n-k)!}$ possible sequences to examine.

# Solutions

- Naive: calculate $Pr(\pi^{(k)})$ for every possible sequence $\pi^{(k)}$ and returning the $\pi^{(k)}$ with the highest $Pr(\pi^{(k)})$.
  - $\frac{n!}{(n-k)!}$ possible sequences to examine.
- Branch-and-Bound [Soliman et al., 2010]: Prune some $\pi^{(k)}$s.
  - Worst case: $\frac{n!}{(n-k)!}$ possible sequences to examine.
- Soliman's Algorithm [Soliman et al., 2010]: searches the space of candidate probabilistic top-$k$ sequences using a Markov chain Monte Carlo algorithm.

# Solutions

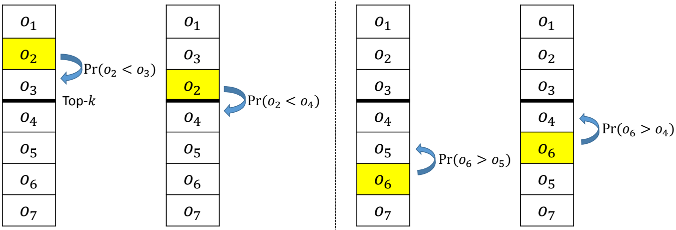
- Naive: calculate $Pr(\pi^{(k)})$ for every possible sequence $\pi^{(k)}$ and returning the $\pi^{(k)}$ with the highest $Pr(\pi^{(k)})$.
  - $\frac{n!}{(n-k)!}$ possible sequences to examine.
- Branch-and-Bound [Soliman et al., 2010]: Prune some $\pi^{(k)}$s.
  - Worst case: $\frac{n!}{(n-k)!}$ possible sequences to examine.
- Soliman's Algorithm [Soliman et al., 2010]: searches the space of candidate probabilistic top-$k$ sequences using a Markov chain Monte Carlo algorithm.
- In this paper, we explore the variants of Markov chain Monte Carlo algorithms.

# Markov chain Monte Carlo Algorithms

- Soliman's Algorithm
  - Initial state: a rank over the $n$ objects

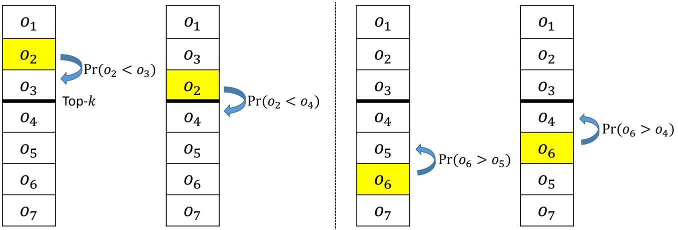
# Markov chain Monte Carlo Algorithms

- Soliman's Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:



# Markov chain Monte Carlo Algorithms

- Soliman's Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:



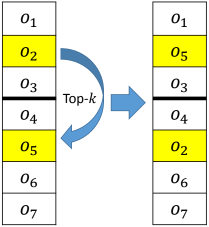

  - Acceptance Probability: $\alpha = \min(\frac{Pr(\pi_{t+1}^{(k)}) \cdot Pr(\pi_t | \pi_{t+1})}{Pr(\pi_t^{(k)}) \cdot Pr(\pi_{t+1} | \pi_t)}, 1)$

# Markov chain Monte Carlo Algorithms

- `Swap` and `SwapEXP` Algorithm
  - Initial state: a rank over the $n$ objects

# Markov chain Monte Carlo Algorithms

- `Swap` and `SwapEXP` Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:



# Markov chain Monte Carlo Algorithms

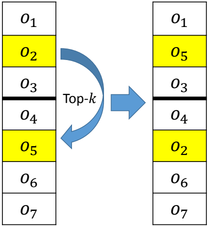
- Swap and SwapEXP Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:
  - Acceptance Probability:

    Swap: $\alpha = \min\left(\frac{Pr(\pi_{t+1}^{(k)}) \cdot \frac{1}{kn}}{Pr(\pi_t^{(k)}) \cdot \frac{1}{kn}} = \frac{Pr(\pi_{t+1}^{(k)})}{Pr(\pi_t^{(k)})}, 1\right)$

    SwapEXP:

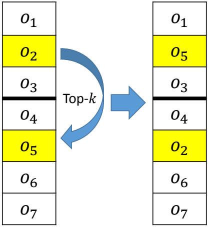
    $\alpha = \min\left(\frac{\widehat{Pr}(\pi_{t+1}^{(k)})}{\widehat{Pr}(\pi_t^{(k)})} = \exp(\beta(Pr(\pi_{t+1}^{(k)}) - Pr(\pi_t^{(k)}))), 1\right)$

    $(\widehat{Pr}(\pi^{(k)}) = C_\beta^{-1} \exp(\beta Pr(\pi^{(k)})))$



# Markov chain Monte Carlo Algorithms

- Swap and SwapEXP Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:




  - Acceptance Probability:

    Swap: $\alpha = \min(\frac{Pr(\pi_{t+1}^{(k)})\cdot\frac{1}{kn}}{Pr(\pi_t^{(k)})\cdot\frac{1}{kn}} = \frac{Pr(\pi_{t+1}^{(k)})}{Pr(\pi_t^{(k)})}, 1)$

    SwapEXP:

    $$\alpha = \min(\frac{\widehat{Pr}(\pi_{t+1}^{(k)})}{\widehat{Pr}(\pi_t^{(k)})} = \exp(\beta(Pr(\pi_{t+1}^{(k)}) - Pr(\pi_t^{(k)}))), 1)$$

    $$(\widehat{Pr}(\pi^{(k)}) = C_\beta^{-1}\exp(\beta Pr(\pi^{(k)})))$$

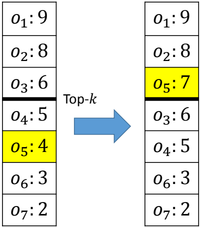
- SwapEXP is more likely to reject the "worse" candidate state.

# Markov chain Monte Carlo Algorithms

- ReSample and ReSampleEXP Algorithm
  - Initial state: a rank over the $n$ objects

# Markov chain Monte Carlo Algorithms

- `ReSample` and `ReSampleEXP` Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:



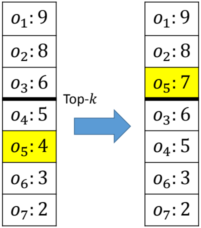
# Markov chain Monte Carlo Algorithms

- `ReSample` and `ReSampleEXP` Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:
  - Acceptance Probability:

`ReSample`: $\alpha = \min(\frac{Pr(\pi_{t+1}^{(k)}) \cdot Pr(\pi_t | \pi_{t+1})}{Pr(\pi_t^{(k)}) \cdot Pr(\pi_{t+1} | \pi_t)}, 1)$

`ReSampleEXP`:

$\alpha = \min(\frac{Pr(\pi_t | \pi_{t+1})}{Pr(\pi_{t+1} | \pi_t)} \cdot \exp(\beta(Pr(\pi_{t+1}^{(k)}) - Pr(\pi_t^{(k)})))\ , 1).$

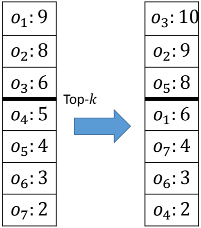


# Markov chain Monte Carlo Algorithms

- `ReSampleAll` Algorithm
  - Initial state: a rank over the $n$ objects

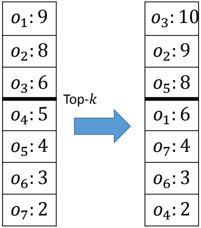
# Markov chain Monte Carlo Algorithms

- `ReSampleAll` Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:



# Markov chain Monte Carlo Algorithms

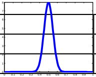
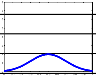
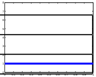
- `ReSampleAll` Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:
  - Acceptance Probability: `ReSample`: $\alpha = 1$



# Performance Evaluation

- Datasets: synthetic datasets

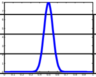
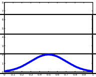
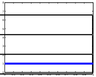
Table: Distributions

| | Setting 1 | | Setting 2 | | Setting 3 | |
|---|---|---|---|---|---|---|
| | |  | |  | |  |
| median score | $G(0.5, 0.05)$ | | $G(0.5, 0.2)$ | | $U[0, 1]$ | |
| width | $G(0.5, 0.05)$ | | $G(0.5, 0.2)$ | | $U[0, 1]$ | |

# Performance Evaluation

- Datasets: synthetic datasets

Table: Distributions

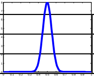
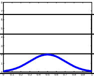
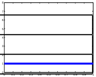
| | Setting 1 | | Setting 2 | | Setting 3 | |
|---|---|---|---|---|---|---|
| median score | $G(0.5, 0.05)$ |  | $G(0.5, 0.2)$ |  | $U[0, 1]$ |  |
| width | $G(0.5, 0.05)$ | | $G(0.5, 0.2)$ | | $U[0, 1]$ | |

  - default: uniform score distributions, median score of $o_i$: $\frac{l_i+u_i}{2}$, width: $u_i - l_i$

# Performance Evaluation

- Datasets: synthetic datasets

Table: Distributions

| | Setting 1 | | Setting 2 | | Setting 3 | |
|---|---|---|---|---|---|---|
| median score | $G(0.5, 0.05)$ |  | $G(0.5, 0.2)$ |  | $U[0, 1]$ |  |
| width | $G(0.5, 0.05)$ | | $G(0.5, 0.2)$ | | $U[0, 1]$ | |

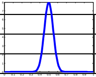
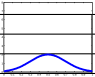
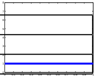
  - default: uniform score distributions, median score of $o_i$: $\frac{l_i+u_i}{2}$, width: $u_i - l_i$

- Metrics

# Performance Evaluation

- Datasets: synthetic datasets

Table: Distributions

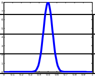
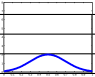
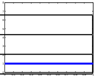
| | Setting 1 |  | Setting 2 |  | Setting 3 |  |
|---|---|---|---|---|---|---|
| median score | $G(0.5, 0.05)$ | | $G(0.5, 0.2)$ | | $U[0, 1]$ | |
| width | $G(0.5, 0.05)$ | | $G(0.5, 0.2)$ | | $U[0, 1]$ | |

  - default: uniform score distributions, median score of $o_i$: $\frac{l_i+u_i}{2}$, width: $u_i - l_i$
- Metrics
  - Probability of the Probabilistic top-$k$ sequence (higher $\rightarrow$ better)

# Performance Evaluation

- Datasets: synthetic datasets

Table: Distributions

| | Setting 1 | | Setting 2 | | Setting 3 | |
|---|---|---|---|---|---|---|
| median score | $G(0.5, 0.05)$ |  | $G(0.5, 0.2)$ |  | $U[0, 1]$ |  |
| width | $G(0.5, 0.05)$ | | $G(0.5, 0.2)$ | | $U[0, 1]$ | |

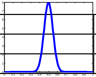
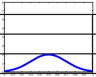
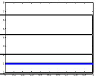
  - default: uniform score distributions, median score of $o_i$: $\frac{l_i+u_i}{2}$, width: $u_i - l_i$

- Metrics
  - Probability of the Probabilistic top-$k$ sequence (higher $\rightarrow$ better)
  - Convergence of the Markov chains (Gelman-Rubin Convergence Diagnostic)

# Performance Evaluation

- Datasets: synthetic datasets

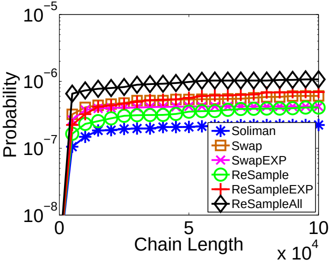
Table: Distributions

| | Setting 1 |  | Setting 2 |  | Setting 3 |  |
|---|---|---|---|---|---|---|
| median score | $G(0.5, 0.05)$ | | $G(0.5, 0.2)$ | | $U[0, 1]$ | |
| width | $G(0.5, 0.05)$ | | $G(0.5, 0.2)$ | | $U[0, 1]$ | |

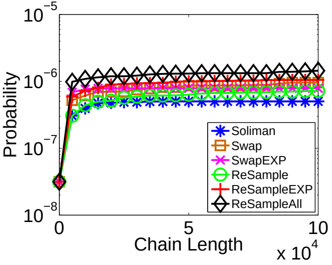
  - default: uniform score distributions, median score of $o_i$: $\frac{l_i+u_i}{2}$, width: $u_i - l_i$

- Metrics
  - Probability of the Probabilistic top-$k$ sequence (higher $\rightarrow$ better)
  - Convergence of the Markov chains (Gelman-Rubin Convergence Diagnostic)
  - Efficiency (Complexity and runtime)

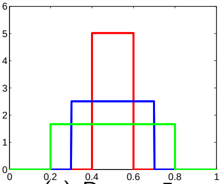
# Effectiveness of Six Algorithms (Probability)



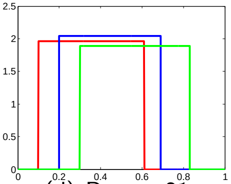

(a) Dataset5




(b) Dataset21

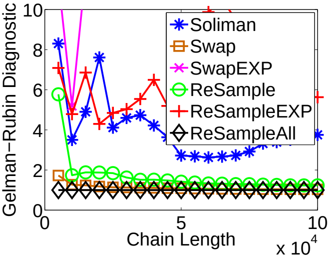


(c) Dataset5

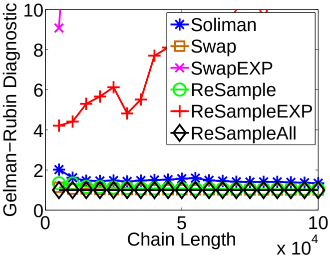


(d) Dataset21

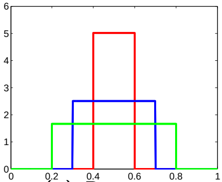
# Convergence of the Markov Chains



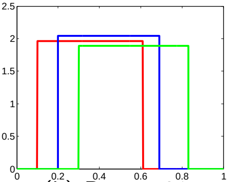

(e) Dataset5




(f) Dataset21




(g) Dataset5




(h) Dataset21

NUS National University of Singapore

# Efficiency

Table: Worst Case Time Complexity of Generating Next State

| | Soliman | Swap(EXP) | ReSample(EXP) | ReSampleAll |
|---|---|---|---|---|
| Time Complexity | $O(nk)$ | $O(1)$ | $O(n)$ | $O(nlogk)$ |

Table: Runtime Per Step of the Algorithms (seconds)

| | Soliman | Swap | SwapEXP | ReSample | ReSampleEXP | ReSampleAll |
|---|---|---|---|---|---|---|
| Runtime Per Step | 0.0058 | 1.9128 | 0.1163 | 0.0523 | 0.0071 | 0.9056 |

# Conclusion

- We explore the design space for Metropolis-Hastings Markov chain Monte Carlo algorithms.

# Conclusion

- We explore the design space for Metropolis-Hastings Markov chain Monte Carlo algorithms.
- We verify through extensive experiments that the proposed algorithms are more effective than the state of the art approach.

# Conclusion

- We explore the design space for Metropolis-Hastings Markov chain Monte Carlo algorithms.
- We verify through extensive experiments that the proposed algorithms are more effective than the state of the art approach.
- `ReSampleAll` is the best, since it samples directly from the target distribution instead of depending on "local" information.

Thank you! Questions?

Top-k Queries

Uncertain Scores

MCMC

liuqing@u.nus.edu

# References I

Soliman, M. A. and Ilyas, I. F. (2009).
Ranking with uncertain scores.
In *ICDE*, pages 317–328.

Soliman, M. A., Ilyas, I. F., and Ben-David, S. (2010).
Supporting ranking queries on uncertain and incomplete data.
*The VLDB Journal*, 19(4):477–501.