Online-FAST: An Online Fairness Assured Service Recommendation Strategy Considering Service Capacity Constraints

2020 Fall CS222 Group Project Presentation

Litao Zhou, Hongbo Yang, Shiwen Dong

IEEE Class F1803016
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University

January 7, 2021

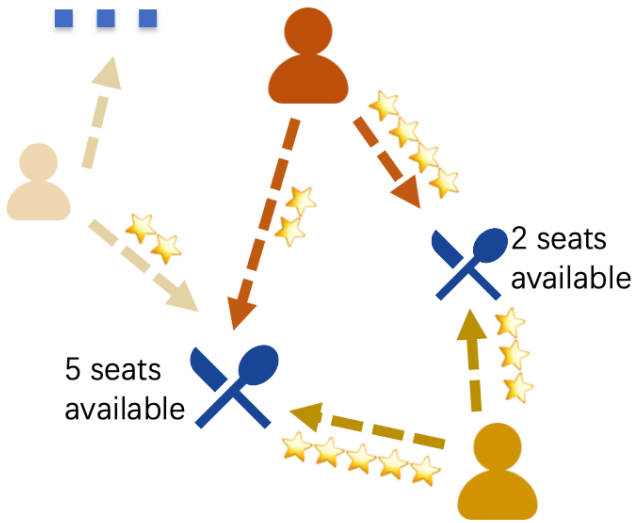
SHANGHAI JIAO TONG UNIVERSITY

Background

Recommendation Fairness

Consider a service recommendation scenario with capacity constraints, some customers may not able to get satisfactory service quality.

(a) Shared Bike Recommendation (A previous project at EI333)



(b) Restaurant Recommendation

We need to (1) meet the capacity constraint, (2) ensure individual fairness, (3) maintain recommendation quality.

Background[1]

FAST: Fairness Assured Service Recommendation Strategy

Significance

1. Establishing a metric for fairness;
2. Giving an algorithm to adjust recommendation results to ensure Fairness among users in N rounds while maintaining a high recommendation quality

Shortcoming

1. High computational complexity with at least a computation of $O(nlogn)$ per request;
2. The proposed algorithm can only calculate a global recommendation plan after all users' information has been gathered

[1] Yao Wu, **Jian Cao**, and Guandong Xu. "FAST: A Fairness Assured Service Recommendation StrategyConsidering Service Capacity Constraint". In: Service-Oriented Computing, (ICSOC 2020)

Background

Shortcoming of FAST

1. High computational complexity with at least a computation of $O(nlogn)$ per request;
2. The proposed algorithm can only calculate a global recommendation plan after all users' information has been gathered
3. The analysis of FAST assumes a **fixed** user set.

These Shortcomings hinder FAST from online deployment.
Our Goal: Establish an Online FAST algorithm.

1. Lower its computing complexity
2. Give a recommended service whenever it gets a request
3. Allow users come with a probability

Terminologies

Notions	Represents
$S = \{s_1, s_2, \cdots, s_m\}$	Set of recommended services.
$C = \{c_1, c_2, \ldots, c_m\}$	Set of services' capacity constraints.
$U = \{u_1, u_2, \ldots, u_n\}$	Set of recommended users.
$\{U^1, U^2, \ldots, U^T\}$	Set of arriving users in T^{th} round.
$\{U_1, U_2, \ldots, U_j\}$	Set of users recommended for service j by the original algorithm
$R = [r_{1,1}, r_{1,2}, \ldots, r_{n,m}]$	Predicted rating matrix produced by the original recommendation algorithm of the system; Each entry r_{ij} denotes the relevant rating of user u_i to service s_j.
$L = \{l_1, l_2, \ldots, l_n\}$	Original recommendation lists based on R.
$L^T = \{l_1^T, l_2^T, \ldots, l_n^T\}$	Recommendation lists finally returned to users in T^{th} round recommendation.
δ_i^T	Variable to indicate whether user u_i uses the recommender system in T^{th} recommendation or not, where $\delta_i^T = 1$ stands for yes and $\delta_i^T = 0$ stands for no.
$Q = \{Q_1, Q_2, \ldots, Q_t\}$ $Q_i = \{q_{i1}, q_{i2}, \ldots, q_{ij}\}$	Request Lists for every round, where q_{ij} indicates the user index of the j^{th} request in the i^{th} round
$A = \{a_1, a_2, \ldots, a_n\}$	The occurrence probability for user u_i in a round.

Figure: Basic Notions

Definition

Definition 1: Ideal Probability

$$p_j^T = \frac{\sum_{u_i \in U_j} \sum_{t=0}^{T} \delta_i^t \cdot \text{Is\_In}\left(s_j, l_i^t, N\right)}{\sum_{u_i \in U_j} \sum_{t=0}^{T} \delta_i^t} \tag{1}$$

Definition 2: Actual Probability

$$p_{i,j}^T = \frac{\sum_{t=0}^{T} \delta_i^t \cdot \text{Is\_In}\left(s_j, l_i^t, N\right)}{\sum_{t=0}^{T} \delta_i^t} \tag{2}$$

where δ_i^t indicates the possibility that user i is the owner of a request in the t^{th} round.

where

$$\text{Is\_In}\left(s_j, \mathit{list}, N\right) = \begin{cases} 0 \text{ if } s_j \text{ is not in the top } N \text{ sub-list of list} \\ 1 \text{ if } s_j \text{ is in the top } N \text{ sub-list of list} \end{cases}$$

Definition

Definition 3: Service Fairness Degree

Fairness degree of user u_i on service s_j up to T^{th} round recommendation:

$$F_{i,j}^{T} = \frac{p_{i,j}^{T} - p_{j}^{T}}{p_{j}^{T}} \tag{4}$$

This definition gives a metric for fairness. The larger $|F|$ is, the less fairness the recommendation has.
And we can define the Service Fairness Degree among the Top-N list of one user.

Definition 4: Top-N Overall Fairness Degree

Overall fairness degree of user u_i up to T^{th} round recommendation:

$$F_{i}^{T} = \sum_{s_j \in I(N)_i} F_{i,j}^{T} \tag{5}$$

Recommendation Quality Metric

Definition 5: Recommendation Quality Metric

Quality of outputted recommendation list l_i^T of user u_i on T^{th} round recommendation:

$$q_i^T = \frac{\sum_{s_j \in l(N)_i \cap l(N)_i^t} \frac{r_{i,j}}{\log_2\left(p_{i,j}^T + 1\right)}}{r_{i,l(N)_i[0]}} \tag{6}$$

where $l(N)_i[0]$ represents the subscript index of the service appearing at the top position of $l(N)_i$, and $p_{i,j}^T$ is the position of service s_j in $l(N)_i$

1. Considering the quality of $l(N)_i$, namely the top-N list;
2. Normalizing the quality score with the highest rating of $l(N)_i$ as the denominator.

Fairness Assured Recommendation Online Problem for Services with Capacity Constraints

Based on the previous metrics, we have following definition of our problem.

Definition 6: Online FAST

Given

1. Relevance rating matrix R
2. Original recommendation lists $\{L\}$
3. Set of services' capacity constraint C
4. $N \in Z_+$
5. Set of ordered lists of recommendation requests Q where q_{ij} indicates the j^{th} request in i^{th} round from a set of users U.

Definition 6 cont'd

Objective

Finding a top-N recommendation list for each request to minimize $D(F_i^T)$, where $D\left(F_i^T\right)$ represents the variance among Top-N Fairness of all users.

Constraint

1. Capacity: keeping $\forall c_j \in C, c_j \geqslant \sum_{u_i \in U} \delta_i^T \cdot \text{Is\_In}\left(s_j, l_i^T, N\right)$
2. Recommendation Quality: preserving $\sum_{u_i \in U} q_i^T$ at a high level
3. Online Property: The recommendation for q_{ij} should not be affected by $q_{i'j'}$ where $i' > i$ or $j' > j$ & $i' = i$.

Improvement of our formulation:

1. Regulating that the requests should be taken sequentially;
2. Taking a user coming probability A instead of the fixed user set as input;

Online-FAST

Function `Fast()`:
- **for** $time = 0 \to n \times N - 1$ **do**
  - Sort users according to F_i^{T-1} from lowest to highest ;
  - rec_user $\leftarrow$ user with the lowest F_i^{T-1} ;
  - **for** s_j *in* $l(N)_{rec\_user}$ **do**
    - **if** $c_j > 0$ **then**
      - $c_j = c_j - 1$;
      - Update $F_{\text{rec\_user}}^{T-1}$;
    - **end**
  - **end**
- **end**
- Fill l_i^T whose positions are larger than N in l_i in sequence ;

return $l_1^T, l_2^T, \ldots, l_n^T$

Function `IsFeasible`(*rec_service, priority_users*):
- $expected\_capacity = \sum_{u \in \text{priority\_users}} a_u \cdot I(rec\_service \in l(N)_u)$;

return $c_{rec\_service} - 1 > expected\_capacity$

Function `OnlineFast()`:
- served = {};
- **foreach** *request* $q_{ij} \in Q_i$ **do**
  - rec_user = q_{ij}.user;
  - rec_service = $l_{\text{rec\_user}}[0]$;
  - **while** *rec_user's top-N list is not filled* **do**
    - priority_users = $\left\{u \in U/served | F_u^{T-1} < F_{\text{rec\_user}}^T\right\}$;
    - **if** `IsFeasible`(*rec_service,priority_users*) **then**
      - Assign rec_service to rec_user ;
      - $c_j = c_j - 1$;
      - Update $F_{\text{rec\_user}}^{T-1}$;
    - **end**
    - rec_service = $l_{\text{rec\_user}}$.next(rec_service);
  - **end**
  - send the recommendation result to user;
  - add user to served set;
- **end**

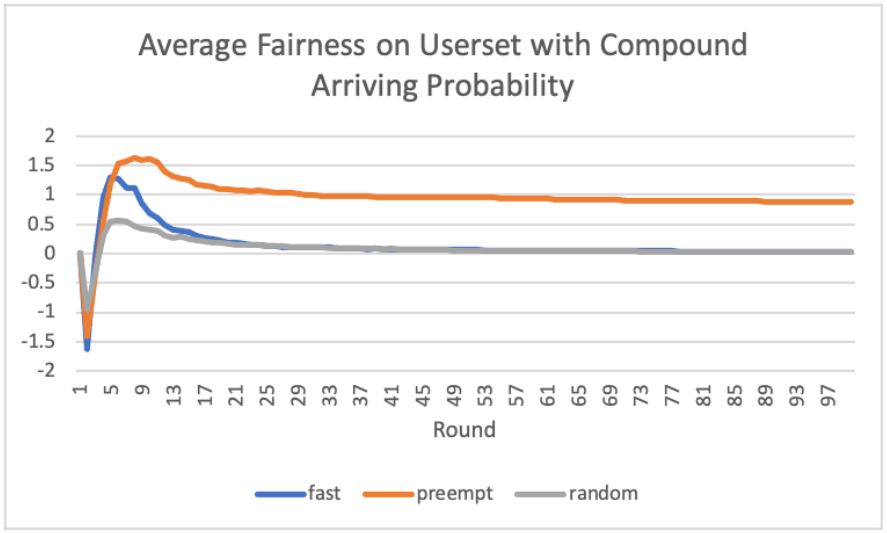
Figure: Comparing 2 algorithms

By using `IsFeasible` as a heuristic, we avoid the high computational cost of sorting and eliminate the dependency on future requests in Offline-FAST.

Analysis of Online-FAST

Theorem 1: Fairness Degree Converges to Zero

Given the arriving probability A for the user set, the sum of Top-N Fairness Degree of all the users in each round will approach to zero (i.e. $\lim_{T\to\infty} \sum_{u_i \in U} F_i^T = 0$) if and only if the users share a same arriving probability.



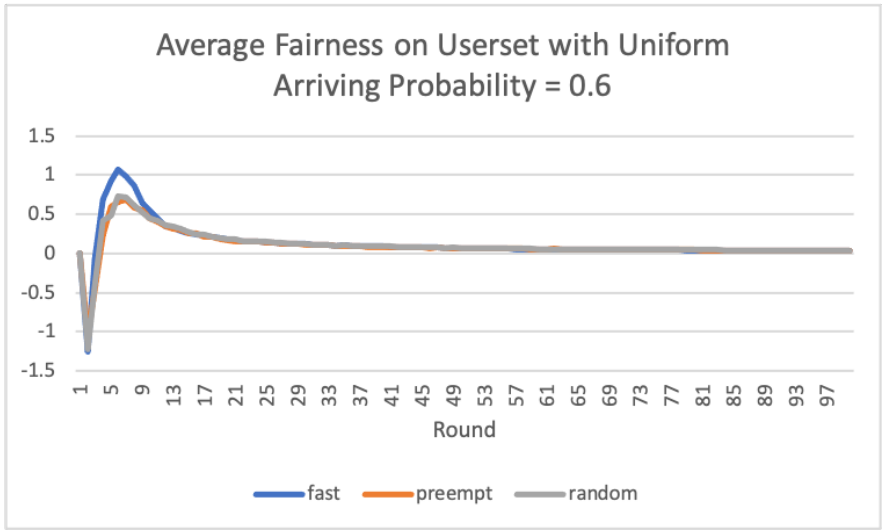


Figure: An intuitive interpretation of Theorem 1

Analysis of Online-FAST

Based on the result of Theorem 1, we have the following claim:

Claim: Variance Convergence

For a group of users with the same arriving probability, assume the arrival of them are uniformly distributed. The variance among the Top-N fairness of them $D\left(F_i^T\right)$ converges with the recommended round T.

1. This claim is based on intuitive inference and assumption of randomness;
2. Its validness and versatility can be checked with experiments in real cases.

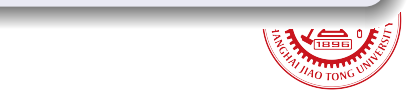
Experiment: Dataset Preparation

Dataset

1. Yelp Dataset: real world dataset
   1. Phoenix
   2. Toronto
2. Synthetic dataset, generated by different param-settings
   1. Dataset 1 : Slack capacity constraints
   2. Dataset 2 : balanced capacity constraints
   3. Dataset 3 : prominent capacity conflicts
   4. Dataset 4 : serious capacity conflicts

Evaluation Metric

1. Total quality of recommendations of each user
2. Variance among Top-N Fairness of all users



Experiment: Constrast Algorithms

We compare our algorithm against the following two baseline methods

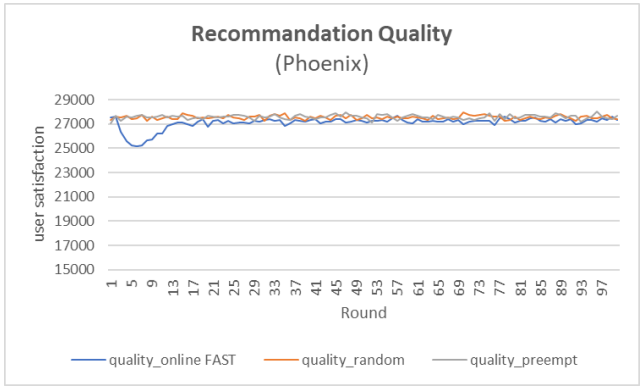
Preempt Strategy

A kind of first-come first-served (FCFS) process scheduling algorithm which doesn't take fairness into consideration

Random Strategy

Assign Services to users randomly, which can also assure Top-N Fairness in the long run.

Experiment on Yelp Dataset



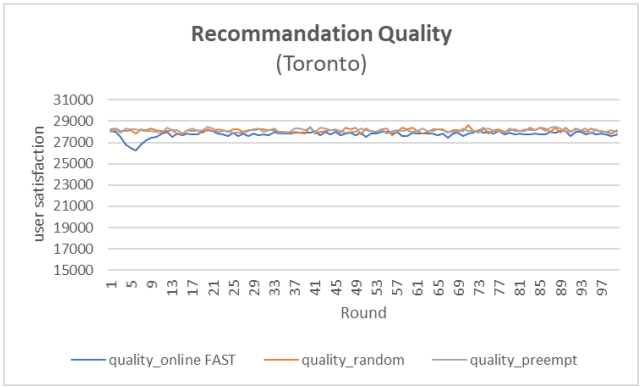
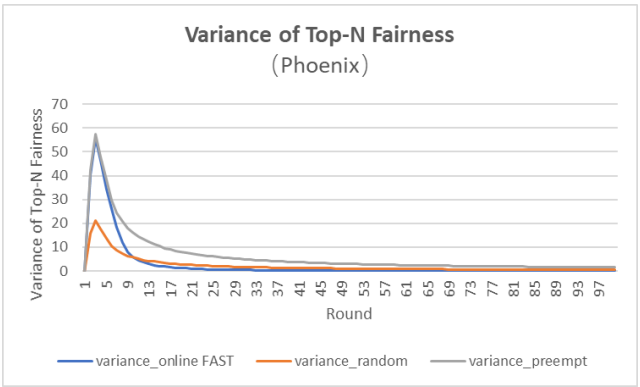


Figure: Online FAST keeps Recommendation Quality at a high level



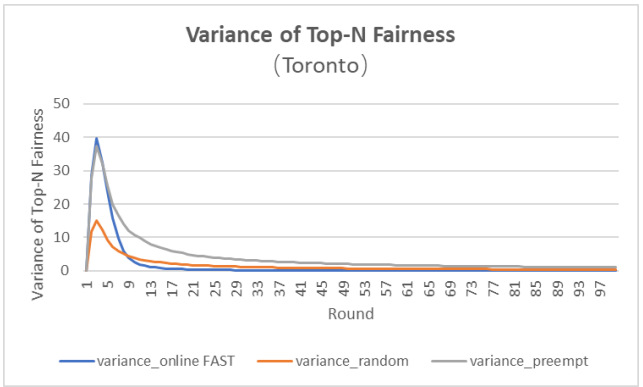
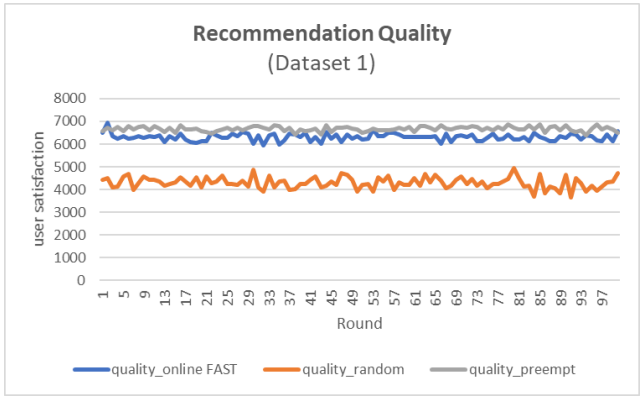
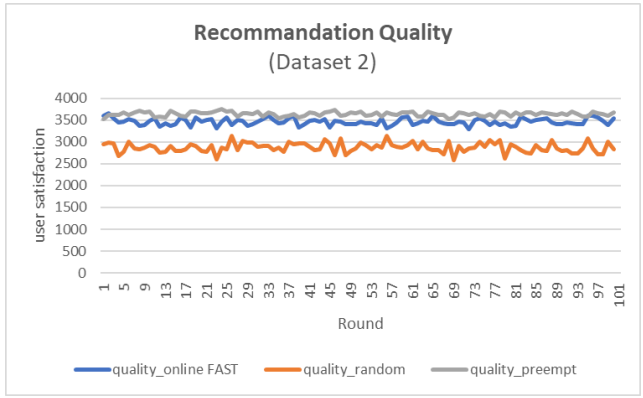


Figure: Online FAST shows better convergence property (Arrival Probability = 0.8, N = 5)

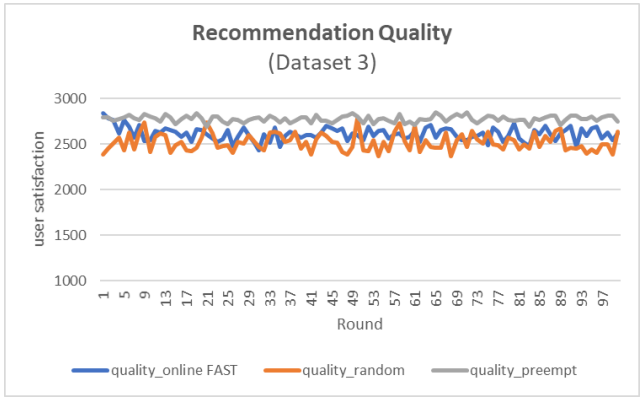
Experiment on Synthetic Dataset



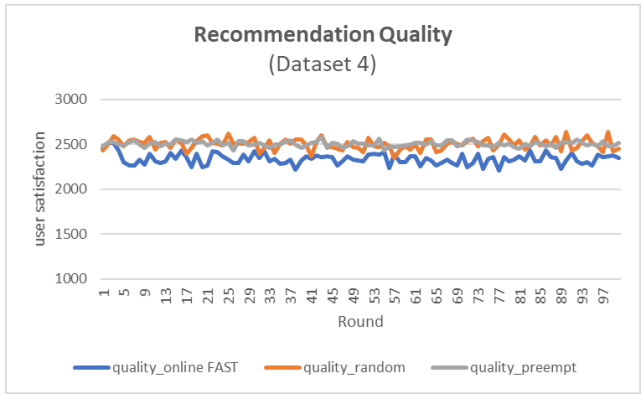
(a) Dataset 1



(b) Dataset 2



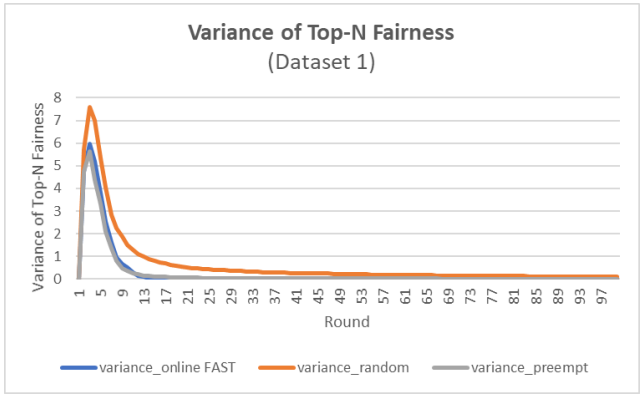
(c) Dataset 3



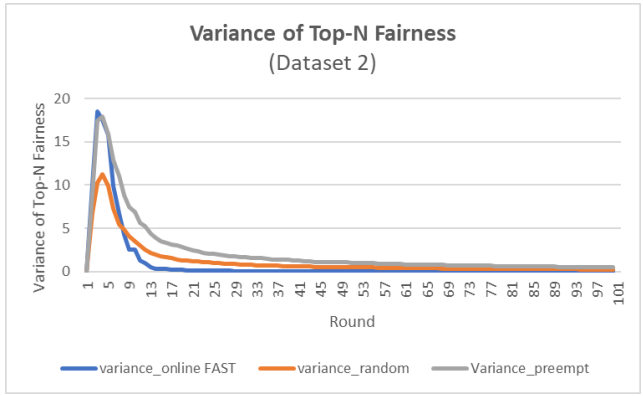
(d) Dataset 4

Figure: Recommendation Quality of Online-FAST is steady under different levels of capacity constraints (Arrival Probability $= 0.6$, N $= 5$)

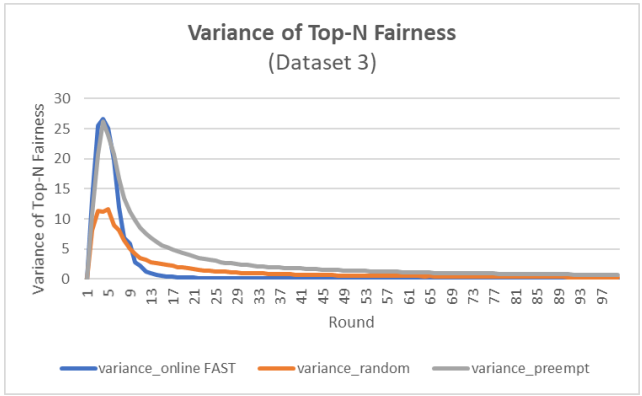
Experiment on Synthetic Dataset



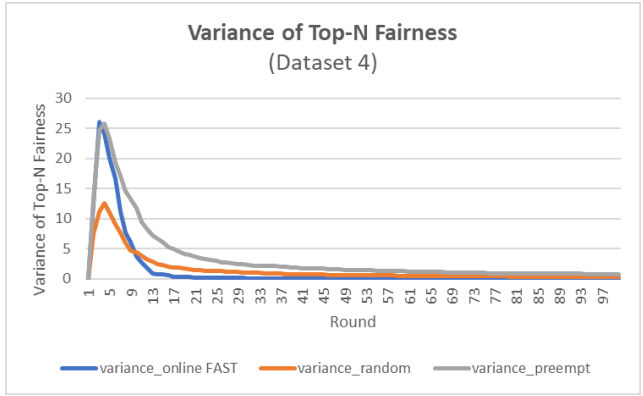
(a) Dataset 1



(b) Dataset 2



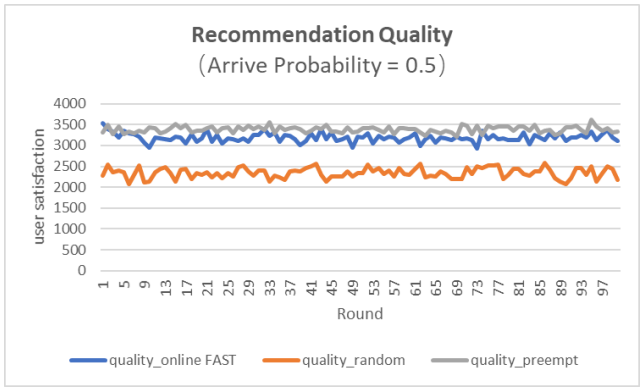
(c) Dataset 3



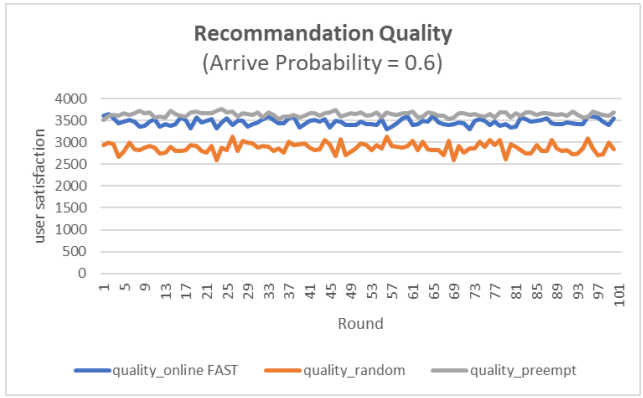
(d) Dataset 4

Figure: Variance of Top-N Fairness can converge under different levels of capacity constraints (Arrival Probability = 0.6, N = 5)

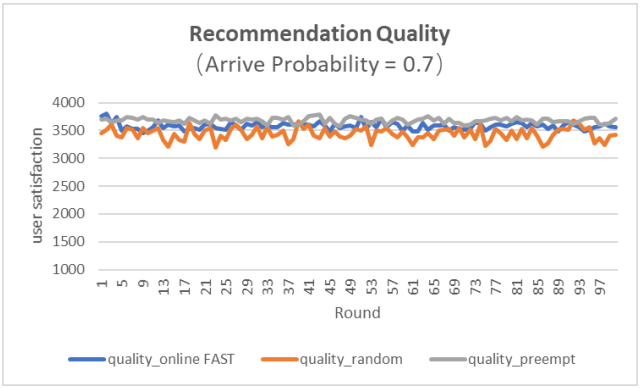
Experiment on Synthetic Dataset



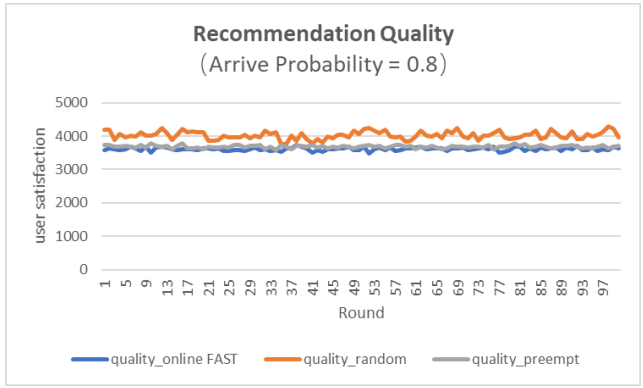
(a) P=0.5



(b) P=0.6



(c) P=0.7



(d) P=0.8

Figure: Recommendation Quality is steady under different arriving probability on Balanced Capacity Constraints Dataset, N = 5)

Conclusion

In this paper, we further expand the concept and application of individual fairness by improving FAST algorithm. To be specific, we first

Our Contributions

1. Introduce user sets with arrival probability into the original FAST framework, and formally prove the soundness of fairness degree.
2. Make the FAST algorithm more suitable for online deployment by designing an online algorithm with
   - $O(N)$ cost per request, compared with $O(N \log N)$ in Offline-FAST
   - Making recommendations independent of future requests

Future Work

1. Recommendation for users with diverse arriving probability
2. Dynamic recommendation results per round from the original algorithm