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

Litao Zhou, Hongbo Yang, Shiwen Dong

January 7, 2021

1 Introduction

1.1 Problem Background

A good recommendation system is of great significance to the trade of commodity. Traditional measure of the performance of a system is recommendation accuracy. Now the fairness of algorithms is also attracting more and more attention with the wide application of recommendation systems. Today, a good recommendation system should not only learn and predict users' preferences to items accurately, but also avoid the unfairness between different users resulting from the recommendation.

Consider a service recommendation scenario with capacity constraints, we will determine how many customers can get satisfactory service quality. For example, restaurants usually have a capacity limitation representing the number of customers that can be served during meal times. If too many customers come to the restaurant during the peak hour, the supply of the restaurant can't meet the needs of so many people. Therefore, when the number of potential users exceeds the service capacity, we have to sacrifice the interests of some users by not providing them the best recommendation in exchange for the overall capacity feasibility. As a matter of fact, if the decision is too premature, there will be great unfairness. How to ensure the fairness of recommendation while maintaining users' actual experience is a problem we try to cope with. Unlike accuracy, which can be quantified clearly, fairness is abstract and relatively difficult to define.

1.2 Related Work

Currently, researchers describe the fairness of a recommendation system from multiple perspectives, which can be divided into three categories considering different stakeholders [1]: customer-fairness, provider-fairness [2] and CP-fairness (both of customer and provider). Customer-fairness contains group fairness [3] and individual fairness [4][5]. The former focuses on the balance between the superior groups and the disadvantaged groups, which requires that these two kinds of groups should have the same opportunities for a particular service so that recommendation results won't be dominated by some specific group. More microscopically, on the other hand, the individual fairness emphasizes the internal consistency within the group, requiring that similar users should get similar treatment. This fairness can be embodied in the prediction accuracy or the similarity of actual recommendation results and the ideal recommendation results.

In this paper, we will focus on the individual fairness of the customer, following the definition of the fairness in a research [6] about the recommendation methods with the capacity constraint of services. The FAST algorithm (Fairness Assured service recommendation Strategy) proposed in that research is designed to guarantee the long term fairness of multi-round recommendations. By experimenting results on both real and synthetic data-sets, FAST has proved to be able to achieve higher fairness than other existing baseline methods. Additionally, it also maintains a reasonably high recommendation quality.

1.3 Motivation

The significance of FAST system is that it proposes a formal metric of individual fairness under service capacity. The metric has proved to be useful when evaluating the fairness of recommendation results under capacity constraints. However, as for the optimization of that metric, the proposed algorithm has a few deficiencies that can't be ignored when it comes to practical deployment.

The first deployment issue with the original algorithm lies in its high computational complexity. A global sorting on fairness degree is required for every assignment in order to find the user who should be given priority to in selecting services. For an online implementation, even if we can implement the sorting process with a priority queue, since the sorting is required for every assignment until the full top-N list is collected, the time required for a single user's request is still in linear proportion to the size of the user set. The computation of $O(n \log n)$ per request can be a large overhead for the server to be deployed.

Another thing to note is that the proposed algorithm can only calculate a global recommendation plan after all users' information has been gathered. However, in practice, users must come sequentially. A static recommendation may not have the desired effect in practice. In fact, from an economist's point of view, Ho[7] pointed out that although recommendation quality may improve over the course of a session, the probability of user to consider and accept a given recommendation will degrade over the course of the session. Therefore, recommendations with an adaptive feature will always outstrip the static recommendations in quality. As for the fairness problem, even if the original FAST algorithm yields a better solution on the proposed recommendation quality metrics, an individual user in reality may be less likely to accept the recommendation results due to the system's response latency.

1.4 Major Work and Contributions

Considering the shortcomings of FAST algorithm above, it is necessary to put forward an online algorithm, which can better adapt to the requirements of actual deployment, improve the efficiency of processing a single request, and ensure the fairness of multi-round allocation. We will show in this paper that such an online algorithm can be constructed. In addition to effectively resolving the above problems, our online algorithm is able to support streamed computing and distributed architecture, which is very common in modern real-time recommendation systems[8]. This is because the recommendation is dependent on heuristics rather than global sorting results. This improvement can make our algorithm better coupled with other recommendation algorithms, so as to realize the fairness of the recommendation system in a broader range of applications.

According to the theory of online algorithms[9], an online algorithm is forced to make decisions that may later turn out not to be optimal, which is also the case in our problem, since we don't know the coming order of future users. In our work, such discrepancy is mainly introduced in the heuristic which ensures that later coming users may still have the chance to be allocated a better service. However, we will show that such a sacrifice would not result in a huge loss of overall performance. This idea will be captured by competitive analysis. We will compare our online algorithm with the original algorithm by calculating the competitive ratio in fairness metric, recommendation quality metric and convergence speed, through theoretical analysis and experiments.

2 Definitions

In order to correctly quantify the fairness of the allocation system and effectively compare the performance of the algorithm, we will propose some formal definitions in this section. We will first clarify the notations that will be used through the paper, and then propose indicators to measure the fairness of the system and the quality of the recommended results respectively.

2.1 Assumptions and Justifications

We assume that there exists a recommendation algorithm that can provide a rating matrix R and provide a recommendation list L_i for all users based on R. We also assume that there exists a capacity constraint on every service, and that the recommendation is executed with respect to rounds. At the end of every round, the capacity will be emptied.

To deal with the capacity constraint, directly filtering out the overcrowded services from the recommendation list of certain users will have a great chance to break the fairness. To solve the problem, we try to design a strategy to adjust L and generate new recommendation lists L^T which can make users' recommendation as fair as possible without breaking capacity constraints while still preserving recommendation quality. The results will be in the form of a Top-N recommendation list for every user, where N is a constant and services will be ordered according to its rating matrix.

A single assignment from users to service is clearly not enough to ensure fairness, so our problem will focus on the accumulating fairness throughout multiple rounds. We also assume that our system focuses on a certain

"active" user set, who are likely to participate in most of the recommendation rounds, so that the users will not vary very much through multiple rounds, ensuring the steadiness of the fairness scope. We also assume that the participated users will only come once in every round, which follows a given occurrence probability A. The user occurrence in the actual coming requests input into the algorithm should conform to the distribution A.

2.2 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 recommendation 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.

Table 1: Basic Notions of Our Work

The notions we'll be using are based on the framework of FAST[6], but with a few extra notations in order to formulate the online property. The notations are given in Table 1.

We should note that although the measures are with respect to each big round, the input of users are not coming in a pack. Our algorithm should read q_{ij} one after another and return the result without knowing any information of the later requests except for the prior probability A of the users who haven't arrived yet.

2.3 Fairness Metrics

To quantify the level of fairness, we need to formulate metrics to measure it properly. Since we consider the Top-N services, we first need to define whether a service is in one's Top-N sub-list.

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

And we define two types of probability below to formulate a user's individual probability of receiving a service with the probability of all users.

Definition 1 (Ideal Probability) The probability of a service s_j appearing in the recommendation lists of all users in $U_j = \{u_i | u_i \text{ owns } Q_{Tk}, \forall k\}$ up to T^{th} round recommendation is:

$$p_j^T = \frac{\sum_{u_i \in U_j} \sum_{t=0}^{T} \delta_i^t \cdot \text{Is\_In}\left(s_j, l_i^t, N\right)}{\sum_{u_i \in U_j} \sum_{t=0}^{T} \delta_i^t} \tag{2}$$

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

Definition 2 (Actual Probability) The probability of a service s_j appearing in the recommendation lists of user u_i up to T^{th} request recommendation in Q is:

$$p_{i,j}^T = \frac{\sum_{t=0}^{T} \delta_i^t \cdot \text{Is\_In}\left(s_j, l_i^t, N\right)}{\sum_{t=0}^{T} \delta_i^t} \tag{3}$$

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

Simply speaking, the Ideal Probability describes an "average" while the Actual Probability is of individuals. Also notice that if u_i gets fair recommendation on s_j, the actual and ideal possibilities for s_j to appear in the top-N list of u_i are equal. Conversely, we can define the difference between them to be a measure for (un)fairness.

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. If $F_{i,j}^T$ is greater than zero, it means service s_j is allocated to u_i 's recommendation lists more frequently than others in U_j; If $F_{i,j}^T$ is less than zero, service s_j appears in his recommendation lists with fewer opportunities than others; If $F_{i,j}^T$ is equal to zero, it means u_i gets fair recommendation on s_j. Then for all top-N services, we can give another definition to show the fairness just among users.

Definition 4 (Top-N Overall Fairness Degree) Overall fairness degree of user u_i up to T^{th} round recommendation:

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

As has been argued in the FAST system[6], in the fairest case, the variance of Top-N fairness among users should be equal to zero, where all users have the same actual probability in receiving a service. It is also valid to take the variance among users' Top -N Fairness $D(F_i^T)$ as a measure of the fairness of recommending system, because F_i^T exactly captures the notion of the accumulating extent of a user's fairness throughout multiple rounds. The smaller the variance is, the more fairness a recommending system has.

2.4 Recommendation Quality Metrics

Recap that L is the list generated by a conventional recommending algorithm, and L^T is generated by the one which takes fairness into consideration. Notice that the quality of recommendations may decrease due to our work to improve fairness.Since we also hope to maintain the quality while improving fairness, we should also introduce a measure for recommending quality.

We only consider the quality of $l(N)_i$, namely the top-N list. In the new list l_i^T, the quality declines when a service is removed, and we use the service predicted rating score to measure the extent of the decline. Therefore, the quality of a new recommendation list l_i^T will be the sum of rating scores of all the services belonging to $l(N)_i$ and l_i^T at the same time. Notice that users may give their rating in some inclination, we should make a normalization to eliminate this noise factor.

We use the highest rating of $l(N)_i$ as the denominator to normalize the quality score. The positions of services $p_{i,j}^T$ in their recommendation lists also reflect their importance to a user.($p_{i,j}^T$ is not possibility) The quality measurement can be extended by giving each service a logarithmic discount based on its position in $l(N)_i$. The quality of recommendation lists of the entire system consists of the recommendation list quality of each user by adding them all.

Definition 5 (Quality of Recommendation List) 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 $l(N)_i$

3 Problem Formulation

Based on the previous metrics, we can get the formal definition of fairness assured online recommendation problem for services with capacity constraints.

Definition 6 (Fairness Assured Recommendation Online Problem for Services with Capacity Constraints.) **Given**, Relevance rating matrix R and original recommendation lists $\{L\}$ generated by a conventional recommendation algorithm, the set of services' capacity constraint C, a positive integer N, an set of ordered lists of recommendation requests Q where q_{ij} is from a set of users U, indicating the j^{th} request in i^{th} round.

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 $j' > j$ & $i' = i$.

The improvement in our formulation compared with the original FAST system is that our problem regulates that the requests should be taken sequentially. Our problem also takes a user coming probability A instead of the fixed user set as input. These two differences are what we believe to be of the greatest help for online implementation.

4 A Fairness Guaranteed Service Recommendation Online System (Online-FAST)

In the original FAST system, the recommendation algorithm is shown in Algorithm 1, which assumes that the requests from users come in groups and that the system can make use of the global information (the users participating in the round) to return a fairness guaranteed recommendation result while ensuring the recommendation quality.

Algorithm 1: Offline Fairness Assured Service Recommendation Algorithm (FAST) for Fixed Users

Input: N: Parameter Top-N;
$l_1, l_2, \ldots, l_n$: Original recommendation list of users;
$l(N)_1, l(N)_2, \ldots, l(N)_n$: Original top-$N$ recommendation list of users;
R: Rating matrix;
$c_1, c_2, \ldots, c_m$: Capacity constraints of services;
$U_1, U_2, \ldots, U_m : U_j$: list of all items;
$F_1^{T-1}, F_2^{T-1}, \ldots, F_n^{T-1}$: Top-N Fairness of all users up to last recommendation.
Output: $l_1^T, l_2^T, \ldots, l_n^T$: Recommendation list for all the users in T^{th} round;

```
for time = 0 → n × N − 1 do
    Sort users according to F_i^{T−1} from lowest to highest ;
    rec_user ← 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^{T−1}_{rec_user} ;
        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, ..., l_n^T
```

As has been argued in Section 1.3, the need of online deployment requires that the algorithm should not depend on future requests when processing a user's request. In addition, users are allowed to appear or not

appear with a certain probability in each round. In order to resolve these two requirements, we present a new algorithm Online-FAST in Algorithm 2 based on the framework of FAST(Algorithm 1).

Algorithm 2: Online Fairness Assured Service Recommendation Algorithm (Online-FAST) for Mutable Users with Arriving Probability

Input: N: Parameter Top-N;
$l_1, l_2, \ldots, l_n$: Original recommendation list of users;
$l(N)_1, l(N)_2, \ldots, l(N)_n$: Original top- N recommendation list of users;
R: Rating matrix;
$c_1, c_2, \ldots, c_m$: Capacity constraints of services;
$F_1^{T-1}, F_2^{T-1}, \ldots, F_n^{T-1}$: Top-N Fairness of all users up to last recommendation;
$a_1, a_2, \ldots, a_n$: The occurrence probability of users in T^{th} round;
$q_{i1}, \ldots, q_{ik}$: The coming request of T^{th} round.
Output: $l_1^T, l_2^T, \ldots, l_n^T$: Recommendation list for all the users in T^{th} round;

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
return

In an online scenario, since we don't know the future requests, many scheduling or recommendation systems will use the preemption strategy, which always assigns the available service to first-arriving users until the capacity of the service is full. Certainly, such strategy ensures a relatively optimal result under an online setting. However, it fails to ensures the fairness of users throughout multiple rounds. It might be the case that a large number of users' requests arrive almost at the same time. However, due to network speed, server routing, backend mechanism and other problems, they are identified as a series of successive requests by the server and are processed in an preemptive approach which should not exist.

In order to eliminate the potential unfairness risks in preemption algorithms, we need to design a strategy where those who arrive first can reserve a portion of his best services for those who need it later, and this waiver mechanism should not unduly affect the quality of the recommendation. In Online-FAST, we introduce a utility function called `IsFeasible` to determine whether the current user should be recommended or give up a service.

The `IsFeasible` function will estimate the number of users that may appear in the future, for whom assigning the services can leads to a reduce in the global difference of fairness degree, and then compare it with the capacity of the current service. It will return false if the job should not be assigned to the current user and be prepared for future users, and return true if the assignment of the service will not cause future unfairness.

Compared to FAST, the `IsFeasible` function can be viewed as a "smaller" effort towards reducing un-

fairness since it makes an under-approximation of future unfairness. Therefore, the algorithm should converge slower than FAST. We will later show by experiments that the online-FAST can still ensure the convergence of fairness degree, and will maintain the convergence rate at a practical level.

5 Analysis

5.1 Correctness

In order to ensure the functional correctness of Online-FAST, we claim two important properties, and relevant proofs can refer to Appendix A.

Theorem 7 (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.

Theorem 7 indicates that when considering arriving probability, the fairness degree should be calculated and maintained globally, no matter whether the user actually comes in the current round, and that the arriving probabilities for the users should be the same, otherwise our proposed fairness degree will lose the converge-to-zero property, thus failing to accurately describe the fairness of the recommendation system.

This property can be further illustrated by the two figures in 1. We compare experiments on different strategies on users with the same arriving probability against users with diverse arriving probability. If the users have a different arriving probability, the preempt algorithm will not keep the average of fairness degree to zero.

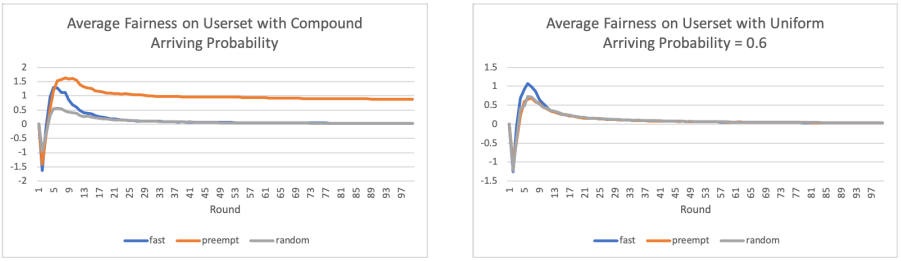


Figure 1: An intuitive interpretation of Theorem 7

Based on Theorem 7, we can argue the functional correctness of our online-FAST algorithm. Claim 8 indicates F-FAST can assure long-term fairness for multi-round recommendations in an online scenario where users' arriving probabilities are given.

Claim 8 (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.

The reasoning in Claim 8 depends heavily on the assumption that the arrival order of users conforms to uniform distribution. Furthermore, in the process of reasoning, we used some intuitive observations to reach conclusions. Therefore, we need to further illustrate the convergence of the algorithm through experiments.

6 Experiments

6.1 Datasets and Metrics

We conduct experiments on two datasets. One is a real world dataset. In order to evaluate the strategy in a more comprehensive way, we also perform experiments on a synthetic dataset. Details of the processing of the datasets can be referred to the Appendix B.

6.1.1 Yelp dataset

The data is provided by Yelp Dataset Challenge, which mainly deals with review and business data to get user ratings for businesses and the city where business is located.

6.1.2 Synthetic datasets

We also generate a synthetic dataset to test the performance of the algorithms under different parameter settings. For this purpose, we generate 4 synthetic datasets with different situations of capacity conflicts when N is set to 5. Dataset 1 represents a situation where there is almost no capacity conflicts. Dataset 2 is a very balanced dataset. Dataset 3 represents a situation where the capacity conflicts are pretty prominent. Dataset 4 is a situation with serious conflicts. Each dataset contains 800 users and 50 services.

6.1.3 Metrics

We measure the total quality of recommendations of each user and variance among Top-N Fairness of all users at the same time.

6.2 Strategy Comparison

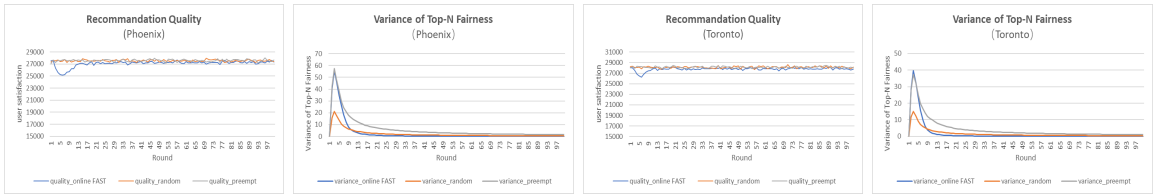
In our experiments, we compare our approach against the following two baseline methods.

Preempt Strategy In this approach, each time a user comes, we check first N services in its original recommendation list one by one until a service with enough capacity is found. Then we allocate this service into the output recommendation list. This method is a kind of first-come first-served (FCFS) process scheduling algorithm which doesn't take the fairness into consideration.

Random Strategy In this approach, each time a user comes, we check first N services in its original recommendation list one by one. For each service, we find all users in the dataset whose top-N original recommendation list contains this service. Then it will be randomly distributed to one of these users. This strategy can also assure the Top-N Fairness in the long run.

6.3 Results on Yelp Dataset

We conduct 100 rounds of recommendations on a dynamic user set on the Yelp dataset, which contains the data of Phoenix and Toronto. In the experiment we set N to 5 and Arrival probability to 0.6, and Figure2 shows the results.



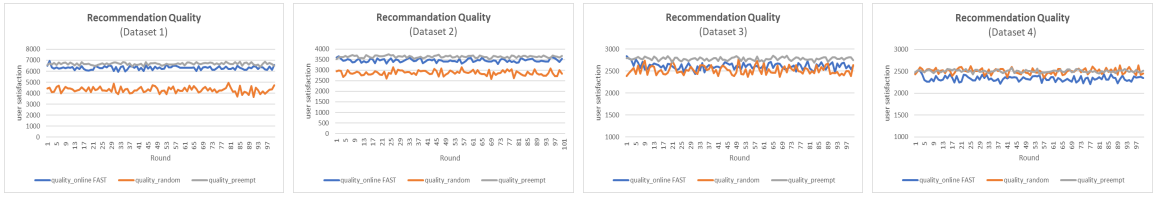
(a) Quality of Recommendation, Phoenix (b) Variance of Top-N Fairness,Phoenix (c) Quality of Recommendation, Toronto (d) Variance of Top-N Fairness, Toronto

Figure 2: Recommendation Quality under Different Levels of Capacity Constraints

It can be seen from Figure2 that our online-FAST's variance of Top-N Fairness approaches zero faster than other two strategy. What's more, its variance of fairness at the stable status is the least. When it comes to the quality of recommendations, we can see that these three strategies can reach the similar result.

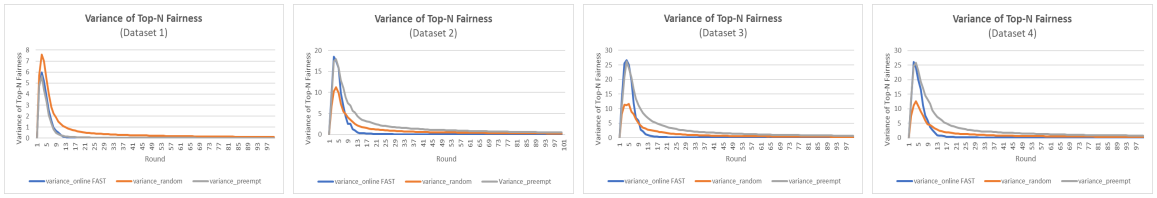
6.4 Results on Synthetic Dataset

Comparisons between Different Levels of Capacity Constraints. We conduct 100 rounds of recommendations on a dynamic user set on four synthetic datasets to compare the performance of algorithms in the cases of different levels of capacity conflicts. In four groups of experiments, N is uniformly set to 5 and probability of arrival is uniformly set to 0.6. Figure 3 and 4 show the results of quality and fairness of recommendations, respectively. It can be seen from Figure 3 that as capacity conflict being more intensive, quality of recommendations tends to decrease. The reason is when capacity conflict becomes more intensive, users have less chances to be assigned top-N services in his original list, which in turn leads to a decrease in quality. The less intensive the capacity conflict is, the better quality of recommendations our approach can achieve compared with the random strategy. Figure 4 shows that online-FAST arrives the stable status of fairness faster than preempt strategy and random strategy in all scenarios. What's more, its variance of fairness at the stable status is the least.



(a) Synthetic Dataset 1 (b) Synthetic Dataset 2 (c) Synthetic Dataset 3 (d) Synthetic Dataset 4

Figure 3: Recommendation Quality under Different Levels of Capacity Constraints



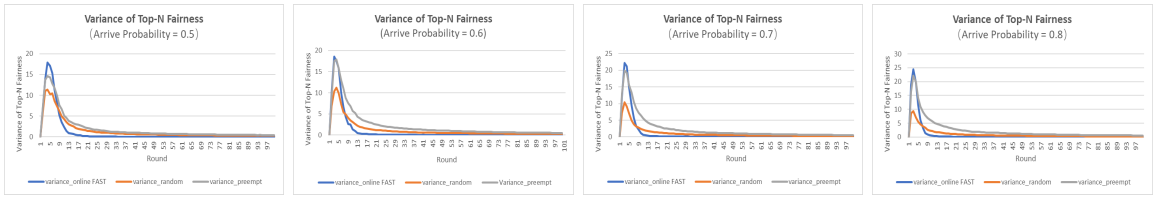
(a) Synthetic Dataset 1 (b) Synthetic Dataset 2 (c) Synthetic Dataset 3 (d) Synthetic Dataset 4

Figure 4: Variance of Top-N Fairness under Different Levels of Capacity Constraints

Comparisons between Different Probability of Arrival. Figure 5 and 6 show the performance of algorithms under different arrival probability. This experiment is performed on Synthetic Dataset 2 on a dynamic user set, and a total of 100 rounds of recommendations are carried out for each experiment. As can be seen from the figure5, as the arrival probability rises, overall recommendation quality improves, but the rate of growth in our online-FAST is less than that of the random strategy. So the smaller the arrival probability is, the more advantage online-FAST have over them with respect to recommendation quality. Figure 6 shows that online-FAST arrives the stable status of fairness faster than preempt strategy and random strategy in all scenarios. What's more, its variance of fairness at the stable status is the least.

(a) Arrive Probability=0.5 (b) Arrive Probability=0.6 (c) Arrive Probability=0.7 (d) Arrive Probability=0.8

Figure 5: Recommendation Quality with Different Arriving Probability

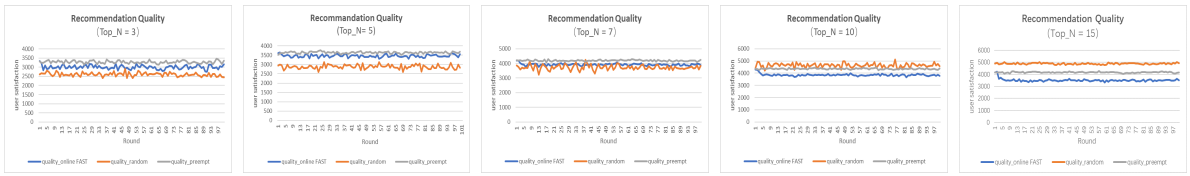


(a) Arrive Probability=0.5 (b) Arrive Probability=0.6 (c) Arrive Probability=0.7 (d) Arrive Probability=0.8

Figure 6: Variance of Fairness with Different Arrive Probability

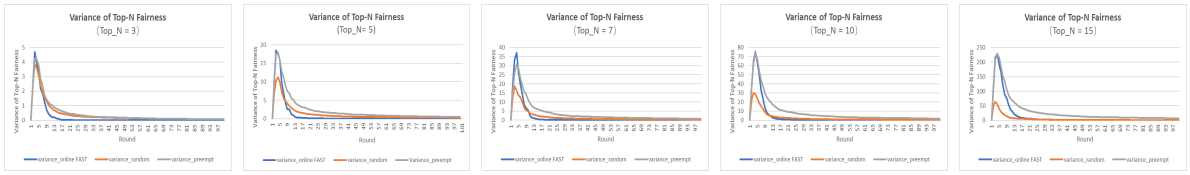
Comparisons between Different N. Figure 7 and 8 show the performance of algorithms under different N. This experiment is performed on Synthetic Dataset 2 on a dynamic user set, and a total of 100 rounds of recommendations are carried out for each experiment. As can be seen from the figure7, as N rises, overall recommendation quality improves at first, but when N becomes large enough, it starts to decrease. The reason is the impacts of the services in the lower positions in the original recommendation list are smaller.

Figure 8 shows that online-FAST arrives the stable status of fairness faster than preempt strategy and random strategy in all scenarios. What's more, its variance of fairness at the stable status is the least. As N rises, the variance of Top-N Fairness rises instead. This shows that a longer top-N recommendation list will not bring benefits to the recommendation quality but impair the fairness.



(a) Top-N = 3 (b) Top-N = 5 (c) Top-N = 7 (d) Top-N = 10 (e) Top-N = 15

Figure 7: Recommendation Quality under Different N



(a) Top-N = 3 (b) Top-N = 5 (c) Top-N = 7 (d) Top-N = 10 (e) Top-N = 15

Figure 8: Variance of Top-N Fairness under Different N

7 Conclusion

In this paper, we further expand the notion of individual fairness by improving FAST algorithm. To be specific, we first introduce user sets with arrival probability into the original FAST framework, and formally prove the soundness of fairness degree. Next, we 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. Its online property will also make the decision process independent of future requests, thus making the algorithm more practical.

Through theoretical analysis and experiments, our work further enhances the practical value and application range of the fairness degree adopted in the FAST algorithm. It is shown that under dynamic user sets and online scenarios, FAST framework can still achieve the purpose of assuring the fairness among users, without losing the recommendation quality. Further work that can be done on the FAST framework includes research

into user requests with dynamic arriving probability, or cases where the original recommendation algorithm gives non-uniform recommendation results in different rounds.

A Proof of Properties for 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.

Proof. The sum of Top-N Fairness at the T^{th} round can be formulated as:

$$\sum_{u_i\in U}F_i^T=\sum_{u_i\in U}\sum_{s_j\in l(N)_i}\frac{p_{i,j}^T-p_j^T}{p_j^T} \tag{7}$$

We can reformulate Equation 7 as:

$$\sum_{u\in U}F_i^T=\sum_{s_j\in S}\sum_{u_i\in U_j}\left(\frac{p_{i,j}^T}{p_j^T}-1\right) \tag{8}$$

For each addend in Equation 8, by substituting $p_{i,j}$ and p_j with their definitions, we can get:

$$\begin{aligned}
\frac{p_{i,j}^T}{p_j^T}&=\frac{\frac{\sum_{t=0}^T\delta_i^t\cdot\text{Is_In}\left(s_j,l_i^t,N\right)}{\sum_{t=0}^T\delta_i^t}}{\frac{\sum_{u_k\in U_j}\sum_{t=0}^T\delta_k^t\cdot\text{Is_In}\left(s_j,l_k^t,N\right)}{\sum_{u_k\in U_j}\sum_{t=0}^T\delta_k^t}}\\
&=\frac{\sum_{u_k\in U_j}\sum_{t=0}^T\delta_k^t}{\sum_{t=0}^T\delta_i^t}\cdot\frac{\sum_{t=0}^T\delta_i^t\cdot\text{Is_In}\left(s_j,l_i^t,N\right)}{\sum_{u_k\in U_j}\sum_{t=0}^T\delta_k^t\cdot\text{Is_In}\left(s_j,l_k^t,N\right)}\\
&\overset{T\to\infty}{\longrightarrow}\frac{\sum_{u_k\in U_j}a_k}{a_i}\cdot\frac{\sum_{t=0}^T\delta_i^t\cdot\text{Is_In}\left(s_j,l_i^t,N\right)}{\sum_{u_k\in U_j}\sum_{t=0}^T\delta_k^t\cdot\text{Is_In}\left(s_j,l_k^t,N\right)}\\
&\overset{T\to\infty}{\longrightarrow}\frac{\sum_{u_k\in U_j}a_k}{a_i}\cdot\frac{a_i\cdot\sum_{t=0}^T\text{Is_In}\left(s_j,l_i^t,N\right)}{\sum_{u_k\in U_j}a_k\cdot\sum_{t=0}^T\text{Is_In}\left(s_j,l_k^t,N\right)}\\
&=\frac{\left(\sum_{u_k\in U_j}a_k\right)\left(\cdot\sum_{t=0}^T\text{Is_In}\left(s_j,l_i^t,N\right)\right)}{\sum_{u_k\in U_j}\left(a_k\cdot\sum_{t=0}^T\text{Is_In}\left(s_j,l_k^t,N\right)\right)}
\end{aligned} \tag{9}$$

To make the second $\overset{T\to\infty}{\longrightarrow}$ derivation hold, we specify that $\text{Is\_In}\left(s,l_i^T,N\right)=0$ if $u_i\notin U^T$ for any service s and list length N, which is compatible with the definition and the algorithm implementation.

Now we can further reformulate each addend in Equation 8 as follows

$$\begin{aligned}
\sum_{u_i\in U_j}\left(\frac{p_{i,j}^T}{p_j^T}-1\right)&\overset{T\to\infty}{\longrightarrow}\frac{\sum_{u_i\in U_j}\left(\sum_{u_k\in U_j}a_k\right)\left(\sum_{t=0}^T\text{Is_In}\left(s_j,l_i^t,N\right)\right)}{\sum_{u_k\in U_j}\left(a_k\cdot\sum_{t=0}^T\text{Is_In}\left(s_j,l_k^t,N\right)\right)}-|U_j|\\
&=\frac{|U_j|\cdot\sum_{u_i\in U_j}\left[\left(\frac{\sum_{u_k\in U_j}a_k}{|U_j|}-a_i\right)\left(\sum_{t=0}^T\text{Is_In}\left(s_j,l_i^t,N\right)\right)\right]}{\sum_{u_i\in U_j}\left(a_i\cdot\sum_{t=0}^T\text{Is_In}\left(s_j,l_i^t,N\right)\right)}\\
&=\frac{|U_j|\cdot\sum_{u_i\in U_j}\left(\bar{a}-a_i\right)\left(\sum_{t=0}^T\text{Is_In}\left(s_j,l_i^t,N\right)\right)}{\sum_{u_i\in U_j}\left(a_i\cdot\sum_{t=0}^T\text{Is_In}\left(s_j,l_i^t,N\right)\right)}
\end{aligned} \tag{10}$$

Since $\sum_{u\in U}F_i^T$ is a finite sum of $\sum_{u_i\in U_j}\left(\frac{p_{i,j}^T}{p_j^T}-1\right)$, we only need to consider the property of convergence to zero for every addend.

In Equation 10, $\text{Is\_In}\left(s_j, l_i^t, N\right)$ depends on the dynamic recommendation strategy, as T grows, it will become a large number. Note that both the denominator and the numerator have this term, so itself alone will not cause $\sum_{u\in U} F_i^T$ to converge or diverge. We only need to consider the relationship of a_i and $\bar{a}$, the total average arriving probability of the user set. It can be observed directly that the derived term in Equation 10 will be zero if and only if the arriving probabilities are identical, which finishes the proof. □

Claim 2 (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.

Proof. First, the variance can be formulated as:

$$D\left(F_i^T\right) = \frac{\sum_{u_i\in U}\left(F_i^T - E\left(F_i^T\right)\right)^2}{n} \tag{11}$$

where $E\left(F_i^T\right)$ represents the mean of Top-N Fairness. According to Theorem 7, we can get $\lim_{T\to\infty} E\left(F_i^T\right) = 0$. Since U is a finite set, we have that:

$$\lim_{T\to\infty} D\left(F_i^T\right) = \lim_{T\to\infty} \frac{\sum_{u_i\in U}\left(F_i^T\right)^2}{n} \tag{12}$$

Since 0 is an lower bound for the term, to prove convergence, we only need to prove $\sum_{u_i\in U}\left(F_i^T\right)^2 \geqslant \sum_{u_i\in U}\left(F_i^{T+1}\right)^2$

According to Equation 4, we know that:

$$\sum_{u_i\in U}\left(F_i^T\right)^2 = \sum_{u_i\in U}\left(\sum_{s_j\in l(N)_i}\frac{p_{i,j}^T - p_j^T}{p_j^T}\right)^2 \tag{13}$$

The assumption that the arrivals of users are uniformly distributed implies the following properties.

1. For service s_j that is unlikely to cause capacity conflicts,

$$c_j \geqslant \sum_{u_i\in U_j} a_i \tag{14}$$

since the arrivals of users are uniformly distributed, it can be assigned to every user in its U_j for most of the rounds. So the ratio of p_j^T and $p_{i,j}^T$ will remain steady. They won't cause much influence on the fluctuation of the variance.

2. For service s_j that is likely to cause capacity conflicts,

$$c_j < \sum_{u_i\in U_j} a_i \tag{15}$$

p_j can be considered to be a constant that doesn't change with respect to the round. Since each service s_j will always be assigned to c_j users, so p_j^T will be a constant less than 1, and we call it Const_j

$$p_j^T = c_j / \operatorname{len}\left(U_i\right) = \ \text{Const}_j < 1 \tag{16}$$

Then, Equation 13 will be:

$$\sum_{u_i\in U}\left(F_i^T\right)^2 = \sum_{u_i\in U}\left[\sum_{s_j\in l_i^N}\left(\frac{p_{i,j}^T}{\text{Const}_j} - 1\right)\right]^2 \tag{17}$$

The only variable in Equation 17 is $p_{i,j}^t$, and according to Equation (11), we can get:

$$\begin{aligned} p_{i,j}^T &= \frac{\sum_{t=0}^{T} Is_{-} \operatorname{In}\left(s_j, l_i^t, N\right)}{T} \\ p_{i,j}^{T+1} &= \frac{\sum_{t=0}^{T} Is_{-} \operatorname{In}\left(s_j, l_i^t, N\right)}{T+1} \\ &+ \frac{Is_{-} \operatorname{In}\left(s_j, l_i^{T+1}, N\right)}{T+1} \end{aligned} \tag{18}$$

According to Theorem 7, we can divide users into two parts, users with low Top-N Fairness $\left(F_i^T < 0\right)$ and users with high Top-N Fairness $\left(F_i^T \geqslant 0\right)$

For users with low Top-N Fairness, addends with $p_{i,j}^T < p_j$ occupies the main influence factor in the summation formula of Top-N Fairness in this situation. As designed in our strategy, users with low Top-N Fairness will usually get allotted to better services that have been reserved from previous requests, so that in most cases:

$$\begin{aligned} 1 > \text{Const}_j > P_{i,j}^{T+1} &= \frac{\sum_{t=0}^{T} Is_{-} \operatorname{In}\left(s_j, l_i^t, N\right) + 1}{T+1} \\ &> \frac{\sum_{t=0}^{T} I_{-} \operatorname{In}\left(s_j, l_i^t, N\right)}{T} = P_{i,j}^T \end{aligned} \tag{19}$$

According to Equation 17, we can know that

$$\left|F_i^{T+1}\right| < \left|F_i^T\right|, \left(F_i^{T+1}\right)^2 < \left(F_i^T\right)^2 \tag{20}$$

For users with high Top-N Fairness, addends with $P_{i,j}^T \geqslant p_j$ occupies the main influence factor in the summation formula of TopN Fairness in this situation. Also, according to our recommendation strategy, these users will always be allotted to unsatisfactory services in order to reserve better services to users with a lower F_i^T so that:

$$\begin{aligned} \text{Const}_j \leqslant p_{i,j}^{T+1} &= \frac{\sum_{t=0}^{T} Is_{-} \operatorname{In}\left(s_j, l_i^t, N\right)}{T+1} \\ &< \frac{\sum_{t=0}^{T} I_{-} \operatorname{In}\left(s_j, l_i^t, N\right)}{T} = P_{i,j}^T \end{aligned} \tag{21}$$

According to Equation 17, we can also get:

$$\left|F_i^{T+1}\right| < \left|F_i^T\right|, \left(F_i^{T+1}\right)^2 < \left(F_i^T\right)^2 \tag{22}$$

since in both cases, $\left(F_i^T\right)^2$ is likely to become smaller as the round of recommendation increases, thus Claim 8 is true.

□

B Processing of Datasets

B.1 Yelp Dataset

This dataset contains the businesses information of two cities, Phoenix and Toronto. The dataset of Phoenix contains 11,252 users, 3774 businesses and 194,188 reviews. The Toronto dataset contains 8867 users, 3,505 businesses, and 1,190,64 reviews. Using the known scores on the extracted data, a collaborative filtering recommendation algorithm based on matrix factorization is implemented to obtain a complete preference score matrix. Then, based on the preference score matrix, the user's original recommendation list l_i, the U_j list of businesses, and other required information are obtained. Since the dataset does not provide capacity information, in this experiment, the capacity constraint of each business is set to a random number between 50 to 100. For online algorithm, we add the user requests list Q_i for each round, which is generated randomly based on the number of users and the arrival probability we have set.

B.2 Synthetic datasets

The capacity constraint of each service can be divided into four levels which are determined by the ratio of the number of users in U_j to its capacity:

- *Very Popular Services*: the number of users in U_j is more than 2 times its capacity.
- *Popular Services*: the number of users in U_j is 1-2 times its capacity.
- *Ordinary Services*: the number of users in U_j is 0.9-1.0 times its capacity.
- *Unpopular Services*: the number of users in U_j is 0.9 times its capacity.

Based on the capacity constraints, 4 datasets are generated. **Dataset 1** represents a situation where there is almost no capacity conflict with half of its services are *Ordinary Services* and the other half are *Unpopular Services*. **Dataset 2** is a very balanced dataset, and it contains services with all capacity conflict levels, and each level accounts for a quarter of the total. **Dataset 3** represents a situation where the capacity conflict is pretty prominent. Half of its services are *Very Popular Services*, and the remaining half is divided equally among the other three levels. **Dataset 4** is a situation with serious conflict, three-quarters of the services are *Very Popular Services*, and the remaining quarter uniformly contains the other three levels. The capacity of each item is a random number from 50 to 70. The user's original recommendation list l_i, the U_j list of businesses, the user requests list Q_i and other required information are obtained in the same way as Yelp Dataset.

References

[1] Burke Robin. "Multisided Fairness for Recommendation". In: *arXiv preprint arXiv:1707.00093* (2017) (cit. on p. 1).

[2] Ashudeep Singh Thorsten Joachims. "Fairness of Exposure in Rankings". In: *arXiv preprint arXiv:1802.07281* (2018) (cit. on p. 1).

[3] Doshi Tulsee Qian Hai Wei Li Wu Yi Heldt Lukasz Zhao Zhe Hong Lichan Chi Ed H Beutel Alex Chen Jilin. "Fairness in Recommendation Ranking through Pairwise Comparisons". In: *arXiv preprint arXiv:1903.00780* (2019) (cit. on p. 1).

[4] Gerhard Weikum Asia J. Biega Krishna P. Gummadi. "Equity of attention: Amortizing individual fairness in rankings". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM, 405–414* (2018) (cit. on p. 1).

[5] Krishnaram Kenthapadi Sahin Cem Geyik Stuart Ambler. "FairnessAware Ranking in Search & Recommendation Systems with Application to LinkedIn Talent Search". In: *arXiv preprint arXiv:1905.01989* (2019) (cit. on p. 1).

[6] Yao Wu, Jian Cao, and Guandong Xu. "FAST: A Fairness Assured Service Recommendation Strategy Considering Service Capacity Constraint". In: *Service-Oriented Computing*. Ed. by Eleanna Kafeza et al. Cham: Springer International Publishing, 2020, pp. 287–303. ISBN: 978-3-030-65310-1 (cit. on pp. 1, 3, 4).

[7] Shuk Ho, David Bodoff, and Kar Tam. "Timing of Adaptive Web Personalization and Its Effects on Online Consumer Behavior". In: *Information Systems Research - ISR* 21 (Jan. 2010) (cit. on p. 2).

[8] Yanxiang Huang et al. "TencentRec: Real-Time Stream Recommendation in Practice". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD '15. Melbourne, Victoria, Australia: Association for Computing Machinery, 2015, pp. 227–238 (cit. on p. 2).

[9] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005 (cit. on p. 2).