Fair Regret Minimization Queries

Yuan Ma[1] and Jiping Zheng[1,2](✉)

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
{mayuancs,jzh}@nuaa.edu.cn

[2] Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China

Abstract. When facing a database containing numerous tuples, users may be only interested in a small but representative subset. Unlike top-k and skyline queries, the k-regret query is a tool which does not need users to provide preferences but returns a representative subset of specified size by users with the minimum regret. However, existing regret-based approaches cannot answer the k-regret query on the dataset which is divided into groups and the result set contains fixed-size tuples in each group, which can be viewed as a metric of *fairness*. For this scenario, in this paper we generalize the k-regret query to its fair form, *i.e.,* the fair regret minimization query. Moreover, we provide an efficient algorithm named α-GREEDY which does not need to access the whole dataset at each greedy step with the help of a layer structure. We conduct experiments to verify the efficiency of the proposed algorithm on both synthetic and real datasets.

Keywords: Regret minimization · Fairness · Set cover

1 Introduction

Returning a representative subset from a large dataset[1] is an important functionality for multi-criteria decision making. Top-k, skyline and k-regret queries are three important tools to address this problem. The k-regret query proposed by Nanongkai *et al.* [21] which integrates the merits of top-k and skyline queries has attracted great attention in the last decade. For the k-regret query, the notion of the *regret ratio* is provided to quantify how regretful if a user gets the best tuple in the selected subset but not the best tuple among all tuples in the database. In vein of the k-regret query, extensions and variants are proposed to solve different problems in real applications [29]. Existing approaches to answer regret minimization queries focus on the datasets where each tuple in them is treated equally.

However, in real applications datasets might have sensitive attributes and be grouped, *e.g.,* the dataset is partitioned into several groups according to some

[1] We use the terms "database","dataset" and "tuple","point" interchangeably in the paper.

H. Yin et al. (Eds.): IDEAL 2021, LNCS 13113, pp. 511–523, 2021.
https://doi.org/10.1007/978-3-030-91608-4_51

attributes such as *gender*, *race* and *ethnicity*. For election, all candidates may be divided into two groups according to their gender or into several groups by race. Under these settings, no existing regret minimization approaches can solve the problem since previous researches only concern the overall returned k tuples and the extracted representative subset would be biased towards minor attributes or groups. For example, when users search the term "CEO" on Google Images [18], the percentage of women in top-100 results is only 11%, significantly lower than the truth of 27% (the ratio of women to men for CEO is 27/73). To solve the problem, it is essential to introduce the concept of fairness to the regret minimization query. Since fairness has different interpretations under different contexts [3,25], in this paper, we take the notion of *fairness* as the input constraint and define the *fairness* constraint as the group cardinality constraints [8,27], *i.e.,* for each group G_i, there exists a cardinality constraint k_i. That is, to be fair, we must select k_i tuples in group G_i. To speed up the process of our fair regret minimization query, we first transform our problem to the constrained set cover problem. Moreover, to avoid the inefficiency of existing greedy approach which needs traversing the whole dataset at each greedy step, we propose an approximation algorithm named α-Greedy. With the help of a layer structure, α-Greedy only accesses part of the dataset thus processes the fair regret minimization query efficiently. Also, the proposed α-Greedy algorithm is with theoretical guarantee. The main contributions of this paper are summarized as follows

- We provide a formal definition of *the fair regret minimization query.*
- An algorithm named α-Greedy is proposed transforming the fair regret minimization query to *the constrained set cover problem*, which has approximation ratio to the overall cardinality of the returned subset.
- We evaluate the performance of our proposed algorithm along with the baselines on both real and synthetic datasets.

2 Related Work

The k-regret query first proposed by Nanongkai *et al.* [21] has been investigated in the last decade [4,11,15,20–22,28,30,31,34] to avoid the drawbacks of top-k queries and skyline queries. For the k-regret query, Nanongkai *et al.* [21] propose the greedy algorithm which we call RDP-Greedy where the tuple which contributes the most is added to the current solution in each selection. Peng *et al.* [22] interpret RDP-Greedy from the geometric aspect and devise a geometric approach whose regret ratios are the same as RDP-Greedy's but is more efficient. Cube is proposed in [21] which has a known upper bound, but its empirical performance is quite poor; Xie *et al.* [30] improve the algorithm by the geometric properties of tuples.

Beside the researches of the k-regret minimization query, there are several variants in the literature. Chester *et al.* [11] propose the concept of kRMS which relaxes the maximum regret ratio to the kth maximum regret ratio. Zeighami *et al.* [34] raise the average regret minimization problem which aims at minimizing the average regret ratio instead of the maximum regret ratio. [1,31] study the problem that minimizes the output size when given a specified regret ratio ϵ, *i.e.,* they return a subset whose maximum regret ratio is at most ϵ and the size is as small as possible. A fully dynamic algorithm is proposed by Wang *et al.* [28] which can answer the k-regret query while the dataset is changing, *i.e.,* with insertions and deletions of database tuples. [20] introduces *interaction* to minimize the maximum regret ratio. All existing researches related to the regret minimization problem focus on the cardinality constraint, *i.e.,* the output size is k and do not consider the fairness constraint.

The fairness constraint is ubiquitous in our society and there is a broad literature on fairness, incorporating it as a metric [2,3,13,17,23,25,26,33]. [19,27] introduce fairness into submodular maximization problems. [12,14,32] study fairness in machine learning and classification. Also, fairness is attracting more attentions in recommendation [5,7,9,10,24], subset selection [27], and data summarization [8], etc.

3 Preliminaries

Denote D as a set of n d-dimensional points, and $p[i]$ as the value on the ith dimension for each point $p \in D$. The concepts of utility function and regret ratio are given before we define our problem.

Utility Function [21]. A utility function f is a mapping f: $\mathbb{R}_+^d \rightarrow \mathbb{R}_+$. The utility of a user with f is $f(p)$ for any point p which shows how satisfied the user is with the point. Denote $f(S)$ the highest utility among $p \in S$ with function f, *i.e.,* $f(S) = \max_{p \in S} f(p)$.

Following [4,21,30], we restrict the class of utility functions to linear utility functions, *i.e.,* for a linear function f, there exists a d-dimensional vector v, satisfies $f(p) =< v, p >= \sum_{i=1}^{d} v[i] \cdot p[i]$ for any $p \in D$ where $p[i]$ and $v[i]$ are non-negative real values.

Regret Ratio [21]. Given a dataset D, a subset $S \subseteq D$ and a set of utility functions F. The regret ratio of S with a utility function $f \in F$, represented as $rr_D(S, f)$, is defined to be

$$rr_D(S, f) = 1 - \frac{f(S)}{f(D)}$$

Moreover, we use $mrr_D(S, F) = \max_{f \in F} rr_D(S, f)$ to represent the maximum regret ratio of S under a set of utility functions F.

Fairness Constraint [24,27]. The fairness constraint is decribed as follows: assign a cardinality constraint k_i to each group G_i and ensure that $\sum_{i=1}^{l} k_i = k$. The value for each k_i can be determined by users. We can assign $k_i = \frac{|G_i|}{|D|} \cdot k$ to represent the proportion of each group in the dataset, or set $k_i = \frac{k}{l}$ to achieve a balanced representation of each group.

Problem Definition. Given a dataset D divided into l groups $G_1, \ldots, G_l$, a set of utility functions F, and a set of positive integers $k_1, \cdots, k_l$, the fair regret minimization query is to find a subset S of size k where $k = \sum_{i=1}^{l} k_i$ from D such that $mrr_D(S, F)$ is minimized and $|S \cap G_i| = k_i$ for each G_i.

4 The α-Greedy Algorithm

Inspired by [4,28,31], the equivalence between *the fair regret minimization query* and *the constrained set cover problem* is revealed. Equation 1 shows the connection between them.

$$minimize\ \epsilon \quad s.t. \quad S\ covers\ F\ \ \&\ \ |S \cap G_i| = k_i, 1 \leq i \leq l \tag{1}$$

For the constrained set cover problem, the aim is to minimize ϵ while satisfying the following two constraints: *coverage* and *fairness*. First, we define the constraint of *coverage*: if a tuple p_i and a utility function f_j satisfy $1 - \frac{f_j(p_i)}{\max_{p \in D} f_j(p)} \leq \epsilon$, *i.e.,* the regret ratio of p_i on f_j is no more than ϵ, we say f_j is covered by p_i or p_i covers f_j, and ϵ is the threshold of the coverage. For a subset $S \subseteq D$, if there exists at least one tuple $p \in S$ covering any arbitrary function f in F, then S covers F. Based on this definition, we want to find a subset S which can cover F and satisfy the fairness constraint. It means that S is a fair subset and with the mrr no more than ϵ. This reveals the equivalence between the constrained set cover problem and the fair regret minimization query under a specified threshold ϵ. A collection $\mathcal{D}$ corresponds to the dataset D and each set $P \in \mathcal{D}$ corresponds to a point $p \in D$ which is composed of utility functions covered by p. Hence, processing the regret query is equal to find a solution for the corresponding set cover problem.

By utilizing the form of Eq. 1, we can find a suitable value of ϵ by performing a binary search which is also adopted by [4,28,31]. When a solution with k tuples is returned, we can find the exact ϵ. Thus, the problem is transformed to the decision version of a set cover problem, *i.e.,* whether the returned k tuples are enough to cover all elements (utility functions). Here, k tuples can be seen as k sets for the set cover problem.

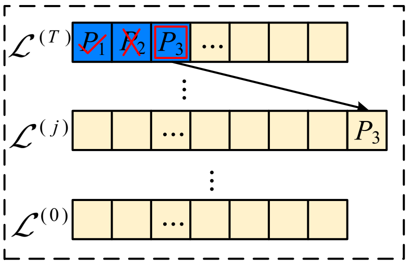

Fig. 1. The layer structure: P_1 is selected, P_2 is discarded and P_3 is moved to layer $\mathcal{L}^{(j)}$

The main idea of α-Greedy is to adopt a layer structure shown in Fig. 1 to avoid traversing all the sets at each greedy step to find the set contributing the most to the result set. Initially, all sets $\{P_1, \ldots, P_n\}$ in $\mathcal{D}$ are put into different levels of the layer structure according to their sizes, *i.e.,* the set P is put into $\mathcal{L}^{(j)}$ if $\alpha^j \leq |P| < \alpha^{j+1}$. The layers are verified in the top-down manner. A set in jth layer would be selected if the number of new elements it covers is no smaller than α^j. At the beginning, since the covered set C is empty, we directly select the first set P_1 in the top level $\mathcal{L}^{(T)}$ ($|P_1 \backslash C| = |P_1 \backslash \emptyset| \geq \alpha^T$). The covered set $C = C \cup P_1 = P_1$ is updated after the selection. Then we check some following set P_i. P_i is discarded if one of the two cases occurs: P_i cannot cover any new element ($|P_i \backslash C| = 0$) or selecting P_i would violate the fairness constraint. Otherwise, we calculate the number of new elements P_i covers, *i.e.,* $|P_i \backslash C|$. If $|P_i \backslash C| < \alpha^T$ and $|P_i \backslash C|$ lies in the interval $[\alpha^j, \alpha^{j+1})$ ($j \geq 0$), we move P_i to the end of $\mathcal{L}^{(j)}$. In Fig. 1, we first directly select P_1 to the result set. Then P_2 is discarded for no new elements added. P_3 is moved to Layer $\mathcal{L}^{(j)}$ and appended to the end of it for $|P_3 \backslash C|$ is in the interval of this layer. Once we check the sets at the ith level ($i < T$), it means the above levels are empty since all the sets in them are selected (*e.g.,* P_1) or discarded (*e.g.,* P_2) or moved to some lower level (*e.g.,* P_3, moved to $\mathcal{L}^{(j)}$). When all elements are covered or all the layers have been accessed, the check process is terminated. In most cases, we need not access all the layers which can improve efficiency.

The pseudocode of α-Greedy is shown in Algorithm 1. We set the upper bound and lower bound of ϵ to be 1 and 0, respectively (Line 1). Then we use the binary search to find the most suitable value of ϵ, *i.e.,* the lowest coverage threshold for the decision version of the set cover problem (Lines 2–9). We construct the set system $\Sigma = (\mathcal{D}, F)$ according the coverage along with ϵ: taking all utility functions as the ground set $F = \{f_1, \ldots, f_m\}$; for each point p_i and function f_j, if $rr_D(p_i, f_j) \leq \epsilon$, we put f_j to the set P_i, hence P_i is a set of utility functions; lastly, all P_is make up the collection $\mathcal{D} = \{P_1, \ldots, P_n\}$. The *Solver* procedure is adopted to solve the constrained set cover problem. If *Solver* returns an empty set, it implies the algorithm cannot find a feasible solution with the threshold ϵ, and the value of ϵ should be enlarged. Otherwise, we can find a feasible solution with the threshold ϵ, but we don't know whether the threshold ϵ can be lowered or not. Thus, we lower its value and try again. When the difference between ϵ_l and ϵ_h is within a parameter ξ (small enough), we use the latest feasible collection as the result, and determine its corresponding points in dataset D instead of the sets in $\mathcal{D}$. Finally, Lines 11–12 ensure the solution is fair for each group.

In the procedure of *Solver*, the solution $\mathcal{S}$ and the covered set C are initialized to be empty sets. All non-empty sets (empty sets are discarded directly) are put into appropriate layers (Line 16). We check the layers in the top-down manner and determine the operation on the set in each layer (Lines 19–27). When the loop breaks, *Solver* decides whether the solution $\mathcal{S}$ is feasible or not according to $F \backslash C$ (Line 28). $|F \backslash C| = 0$ implies that all elements are covered by $\mathcal{S}$ under the fairness constraint. Note that the α value is set to be only slightly larger than

1 to make sure that the P_is lie in different layers. When α approaches 1, there will be a large amount of layers with no sets. For this situation, when checking the P_is, we directly ignore these empty layers.

Algorithm 1: α-GREEDY

```
Input: A dataset of $n$ points $D = \{p_1, p_2, \cdots, p_n\}$ divided into $l$ groups
       $G_1, \ldots, G_l$, a set $F = \{f_1, \cdots, f_m\}$, fairness constraint
       $FC : k_1, \cdots, k_l \in \mathbb{Z}^+$, and a real-valued parameter $\alpha > 1$.
Output: A result set $S$ of the fair regret minimization query.
1  $\epsilon_l = 0$, $\epsilon_h = 1$;
2  while $\epsilon_h - \epsilon_l \geq \xi$ do
3      $\epsilon = (\epsilon_l + \epsilon_h)/2$;
4      Construct set system $\Sigma = (\mathcal{D}, F)$;
5      $temp \leftarrow Solver(\mathcal{D}, F, FC, \alpha)$;
6      if $temp == \emptyset$ then $\epsilon_l = \epsilon$;
7      else
8          $\mathcal{S} \leftarrow temp$;
9          $\epsilon_h = \epsilon$;
10 $S \leftarrow$ points in $D$ corresponding to $\mathcal{S}$;
11 while $\exists i \in [1, l] : |S \cap G_i| < k_i$ do
12     Add points in $G_i$ to $S$ until $|S \cap G_i| = k_i$;
13 return $S$;
14 Function $Solver(\mathcal{D}, F, FC, \alpha)$
15     $\mathcal{S}, C \leftarrow \emptyset$;
16     Assign each set $P_i \in \mathcal{D}$ to level $\mathcal{L}^{(t)}$ if $\alpha^t \leq |P_i| < \alpha^{t+1}$;
17     Let $T$ be the largest $t$ with non-empty $\mathcal{L}^{(t)}$;
18     for $(j = T; j \geq 0; j--)$ do
19         for each set $P_i$ in $\mathcal{L}^{(j)}$ do
20             $G_o \leftarrow \{G : P_i \in G_o, 1 \leq o \leq l\}$;
21             if $|\mathcal{S} \cap G_o| == k_o$ or $|P_i \backslash C| == 0$ then
22                 $\mathcal{L}^{(j)} \leftarrow \mathcal{L}^{(j)} \backslash \{P_i\}$;
23                 continue;
24             if $|P_i \backslash C| \geq \alpha^j$ then
25                 $\mathcal{S} \leftarrow \mathcal{S} \cup \{P_i\}, C \leftarrow C \cup P_i$;
26                 if $|F \backslash C| == 0$ then break;
27             else $P_i \leftarrow P_i \backslash C$, push $P_i$ into $\mathcal{L}^{(j')}$: $\alpha^{j'} \leq |P_i| < \alpha^{j'+1}$;
28     if $|F \backslash C| > 0$ then $\mathcal{S} = \emptyset$;
29     return $\mathcal{S}$;
```

Lemma 1. *Solver is a $1/(\alpha + 1)$-approximation algorithm for the maximum coverage problem with the fairness constraint.*

Proof. The maximum coverage problem is to select k subsets from F such that their union has the maximum cardinality [16]. For our problem, we only prove the special case, *i.e.,* each group's constraint k_i equals 1. For $k_i > 1$, we make k_i copies of group G_i and the fairness constraint to each copy of G_i can also be 1.

Reorder the groups so that *Solver* picks a set from G_i just in the ith iteration, *i.e.,* $\mathcal{S} = \{S_1, \dots, S_k\}$. Let O denote the optimal solution, and we reorder the sets in O to ensure that S_i and O_i are both selected in the group G_i in the ith iteration. Let $S = \bigcup_{i=1}^{k} S_i$, $O = \bigcup_{i=1}^{k} O_i$, and denote $\Delta_i = S_i - \bigcup_{h=1}^{i-1} S_h$ as the set of new elements added by *Solver* in the ith iteration. When *Solver* picks S_i, the set O_i can also be picked. Since $\frac{\alpha^i}{\alpha^{i+1}} = \frac{1}{\alpha}$, the new elements S_i covers are at least $\frac{1}{\alpha}$ of the set M_i where M_i is the set which adds the largest number of new elements in group G_i. Therefore, $|\Delta_i| \geq \frac{1}{\alpha}|M_i - \bigcup_{h=1}^{i-1} S_h| \geq \frac{1}{\alpha}|O_i - \bigcup_{h=1}^{i-1} S_h| \geq \frac{1}{\alpha}|O_i - S|$. We have

$$|S| = \sum_{i=1}^{k} |\Delta_i| \geq \sum_{i=1}^{k} \frac{1}{\alpha}|O_i - S| \overset{(*)}{\geq} \frac{1}{\alpha}(|\bigcup_{i=1}^{k} O_i| - |S|) = \frac{1}{\alpha}(|O| - |S|)$$

The derivation of the inequality $(*)$ is as follows.

$$\begin{aligned}
&\sum_{i=1}^{k} |O_i - S| = |O_1 - S| + |O_2 - S| + \sum_{i=3}^{k} |O_i - S| \\
&\geq |O_1 + O_2 - S| + \sum_{i=3}^{k} |O_i - S| \geq |O_1 + O_2 + \cdots + O_k - S| \\
&= |\bigcup_{i=1}^{k} O_i - S| \geq |\bigcup_{i=1}^{k} O_i| - |S|
\end{aligned}$$

Hence, $|S| \geq \frac{1}{\alpha+1}|O|$, the lemma is proved.

Theorem 1. *Algorithm 1 has* $1 + \log_{1+\frac{1}{\alpha}} m$*-approximation with total cardinality constraint (*i.e., k*) for the fair regret minimization query.*

Proof. We assume the size of optimal cover set is k. Since the *Solver* procedure picks out k sets at each step, by Lemma 1 we have that the number of uncovered elements is not more than $(1 - \frac{1}{1+\alpha})|F| = \frac{\alpha}{1+\alpha}m$. After $1 + \log_{1+\frac{1}{\alpha}} m$ greedy steps, the number of uncovered elements is less than $(\dfrac{\alpha}{1+\alpha})^{1+\log_{1+\frac{1}{\alpha}} m} \cdot m = \dfrac{\alpha}{1+\alpha} < 1$, so there does not exist any uncovered element after these greedy steps. The α-Greedy algorithm which violates the cardinality constraint has the better performance than the optimal solution.

5 Empirical Evaluation

We first elaborate the experimental setup before we analyze the quality and efficiency of our proposed algorithm.

Datasets. We run the algorithms on both synthetic and real datasets. The synthetic dataset was created using the data generator [6], and three real-world datasets which are widely adopted in the literature [1,21,30,31]. The information of datasets is summarized as the following.

- Anti-correlated. The syntheic dataset is a 6-dimensional anti-correlated dataset of 10,000 points.
- NBA[2]. NBA is a 6-dimensional dataset with 21,961 points for each player/season combination from year 1946 to 2009. Six attributes represent the performance of each player, *e.g., total scores* and *blocks*, etc.
- Household[3]. The household dataset contains 1,048,576 family tuples with seven attributes, showing economic characteristics of each family, *e.g., annual property insurance cost*, and *selected monthly owner costs.*
- El Nino[4]. The El Nino dataset contains 178,080 tuples with 5 oceanographic attributes taken at the Pacific Ocean, such as *relative humidity*, *sea surface temperature*, and *air temperature.*

Following [4,21,30], we also preprocessed the datasets to only contain skyline points to answer the query. The processed anti-correlated dataset has 5,120 skyline points and there are 130, 57, 1183 skyline points for the NBA, Household and El Nino datasets, respectively. For convenience, we divide the datasets in sequence into l groups. F is consisted of 1,000 uniform sampled utility functions. All the algorithms are performed on a Core-i7 machine running Ubuntu 18.04 LTS, and implemented in C++.

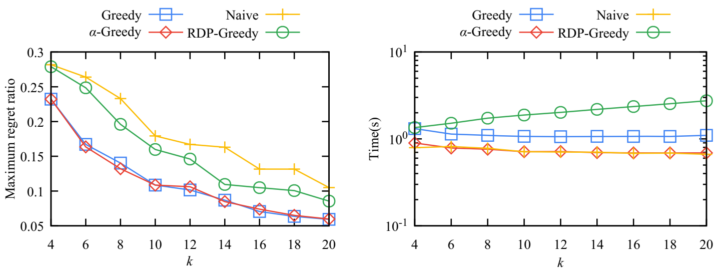

Fig. 2. Vary k on the anti-correlated dataset

[2] https://www.basketball-reference.com/.
[3] http://www.ipums.org.
[4] http://archive.ics.uci.edu/ml/datasets/El+Nino.

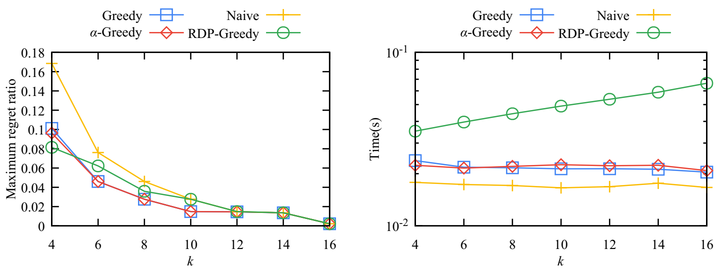

Fig. 3. Vary k on the NBA

In our experiments, we compare the following algorithms: Greedy, α-Greedy, Naive and RDP-Greedy. Greedy and Naive are mentioned in Sect. 4. RDP-Greedy is a heuristic algorithm adapting the greedy algorithm in [21] for the fair regret minimization query. The parameter ξ is set to be 10^{-4}. For α-Greedy, we set $\alpha = 1.001$. We compare all the algorithms from two aspects: maximum regret ratio (mrr) and running time.

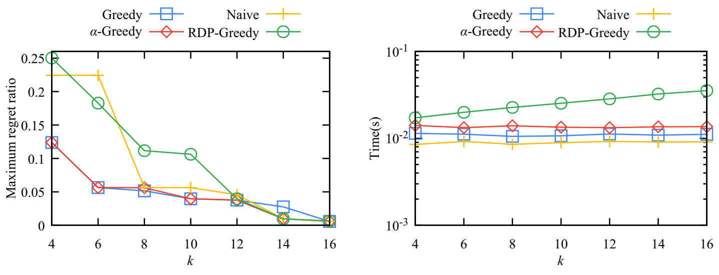

Fig. 4. Vary k on the household

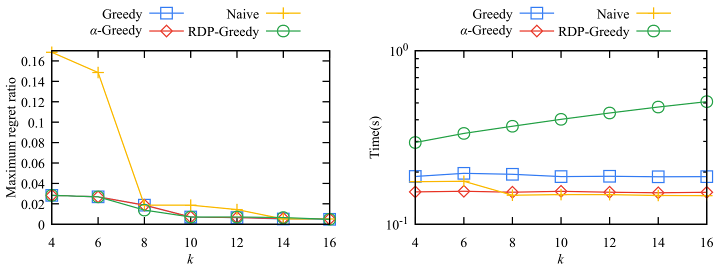

Fig. 5. Vary k on the El Nino

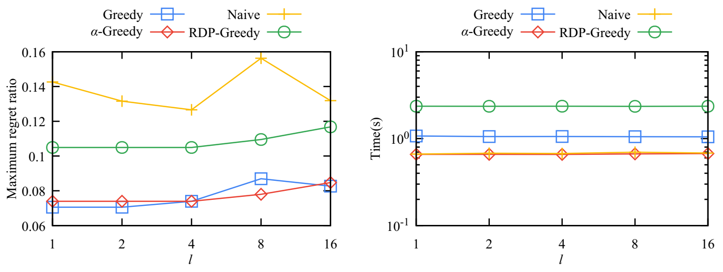

Fig. 6. Vary l on the anti-correlated dataset

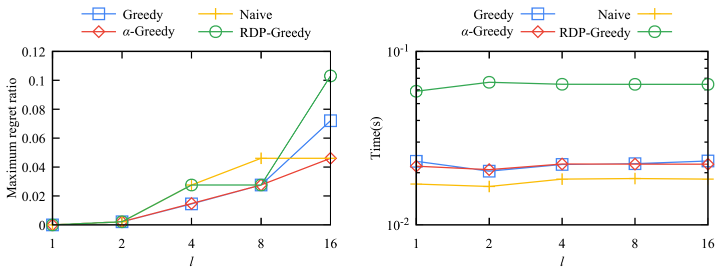

Fig. 7. Vary l on the NBA

Results. As Figs. 2, 3, 4, 5 show, Greedy and α-Greedy achieve better performance in mrr than Naive and RDP-Greedy in most cases. α-Greedy obtains the similar quality in mrr, but α-Greedy is much more efficient on the large datasets, such as the Anti-correlated dataset with 5,120 skyline points. Since the skyline sizes of the NBA and the household datasets are only 130, 57 respectively, α-Greedy does not have the advantage in running time and may be even worse. This is because α-Greedy needs time to put each set into the suitable level of the layer structure. Naive is with the best performance in running time due to its selection strategy not needing to search the whole dataset, which is similar to α-Greedy. But Naive has the worst performance in the maximum regret ratio. RDP-Greedy performs well on the NBA and El Nino datasets, but it is time-consuming to compute mrr of current solution before each selection, and the time cost increases significantly with the increase of k.

Figures 6, 7 show the results when the size of groups varies. We set the group size to be 1, 2, 4, 8, and 16, respectively. Note that, when $l = 1$ the fair regret minimization query is degraded into the k-regret query, and the lowest regret ratio is achieved due to the largest searching space for selection. As the group size l increases, the maximum regret ratio with the same output size increases. This can be viewed as the cost of fairness, *i.e.,* sacrificing some quality in mrr to ensure fairness. When the output sizes of the algorithms are set to a same value, *i.e.,* $k = 16$, the time costs of the algorithms almost remain unchanged under different group sizes.

6 Conclusion

In this paper, we propose *the fair regret minimization query* which combines the fairness constraint with the regret minimization query. To speed up the process of the fair regret minimization query, we first transform the query to the constrained set cover problem. Then, the α-Greedy algorithm is proposed with theoretical guarantees to answer the fair regret minimization query efficiently by utilizing a layer structure. Our experiments verify that our proposed algorithm is effective and scales well on different datasets or under various constraints.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China under grant U1733112 and the Fundamental Research Funds for the Central Universities under grant NS2020068.

References

1. Agarwal, P.K., Kumar, N., Sintos, S., Suri, S.: Efficient algorithms for k-regret minimizing sets. In: Proceedings of International Symposium on Experimental Algorithms (SEA), pp. 7:1–7:23 (2017)
2. Ajtai, M., Aspnes, J., Naor, M., Rabani, Y., Schulman, L.J., Waarts, O.: Fairness in scheduling. J. Algorithms **29**(2), 306–357 (1998)
3. Asudeh, A., Jagadish, H.: Fairly evaluating and scoring items in a data set. In: Proceedings of the VLDB Endowment (VLDB), pp. 3445–3448 (2020)
4. Asudeh, A., Nazi, A., Zhang, N., Das, G.: Efficient computation of regret-ratio minimizing set: a compact maxima representative. In: Proceedings of International Conference on Management of Data (SIGMOD), pp. 821–834 (2017)
5. Beutel, A., et al.: Fairness in recommendation ranking through pairwise comparisons. In: Proceedings of International Conference on Knowledge Discovery & Data Mining (SIGKDD), pp. 2212–2220 (2019)
6. Börzsöny, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of International Conference on Data Engineering (ICDE), pp. 421–430 (2001)
7. Celis, L.E., Huang, L., Vishnoi, N.K.: Multiwinner voting with fairness constraints. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), pp. 144–151 (2018)
8. Celis, L.E., Keswani, V., Straszak, D., Deshpande, A., Kathuria, T., Vishnoi, N.: Fair and diverse DPP-based data summarization. In: Proceedings of International Conference on Machine Learning (ICML), pp. 716–725 (2018)
9. Celis, L.E., Straszak, D., Vishnoi, N.K.: Ranking with fairness constraints. In: International Colloquium on Automata, Languages, and Programming (ICALP), vol. 107, pp. 28:1–28:15 (2018)
10. Celis, L.E., Vishnoi, N.K.: Fair personalization. arXiv preprint arXiv:1707.02260 p. 7 (2017)
11. Chester, S., Thomo, A., Venkatesh, S., Whitesides, S.: Computing k-regret minimizing sets. In: Proceedings of the VLDB Endowment (VLDB), pp. 389–400 (2014)
12. Chouldechova, A., Roth, A.: A snapshot of the frontiers of fairness in machine learning. Commun. ACM **63**(5), 82–89 (2020)
13. Dash, A., Shandilya, A., Biswas, A., Ghosh, K., Ghosh, S., Chakraborty, A.: Summarizing user-generated textual content: motivation and methods for fairness in algorithmic summaries. In: Proceedings of the ACM on Human-Computer Interaction, vol. 3(CSCW), pp. 1–28 (2019)

14. Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R.: Fairness through awareness. In: Proceedings of Innovations in Theoretical Computer Science Conference, pp. 214–226 (2012)
15. Faulkner, T.K., Brackenbury, W., Lall, A.: k-regret queries with nonlinear utilities. In: Proceedings of the VLDB Endowment (VLDB), pp. 2098–2109 (2015)
16. Feige, U.: A threshold of ln n for approximating set cover. J. ACM (JACM) **45**(4), 634–652 (1998)
17. Feige, U.: On allocations that maximize fairness. In: Proceedings of Symposium on Discrete Algorithms (SODA), vol. 8, pp. 287–293 (2008)
18. Kay, M., Matuszek, C., Munson, S.A.: Unequal representation and gender stereotypes in image search results for occupations. In: Proceedings of Conference on Human Factors in Computing Systems, pp. 3819–3828 (2015)
19. Kazemi, E., Zadimoghaddam, M., Karbasi, A.: Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints. In: Proceedings of International Conference on Machine Learning (ICML), pp. 2544–2553 (2018)
20. Nanongkai, D., Lall, A., Das Sarma, A., Makino, K.: Interactive regret minimization. In: Proceedings of International Conference on Management of Data (SIGMOD), pp. 109–120 (2012)
21. Nanongkai, D., Sarma, A.D., Lall, A., Lipton, R.J., Xu, J.: Regret-minimizing representative databases. In: Proceedings of the VLDB Endowment (VLDB), pp. 1114–1124 (2010)
22. Peng, P., Wong, R.C.W.: Geometry approach for k-regret query. In: Proceedings of International Conference on Data Engineering (ICDE), pp. 772–783 (2014)
23. Rabin, M.: Incorporating fairness into game theory and economics. Am. Econ. Rev. **83**(5), 1281–1302 (1993)
24. Serbos, D., Qi, S., Mamoulis, N., Pitoura, E., Tsaparas, P.: Fairness in package-to-group recommendations. In: Proceedings of International Conference on World Wide Web (WWW), pp. 371–379 (2017)
25. Singh, A., Joachims, T.: Fairness of exposure in rankings. In: Proceedings of International Conference on Knowledge Discovery & Data Mining (SIGKDD), pp. 2219–2228 (2018)
26. Stoyanovich, J., Yang, K., Jagadish, H.: Online set selection with fairness and diversity constraints. In: Proceedings of International Conference on Extending Database Technology (EDBT), pp. 241–252 (2018)
27. Wang, Y., Fabbri, F., Mathioudakis, M.: Fair and representative subset selection from data streams. In: Proceedings of The Web Conference (WWW), p. 11 (2021)
28. Wang, Y., Li, Y., Wong, R.C.W., Tan, K.L.: A fully dynamic algorithm for k-regret minimizing sets. In: Proceedings of International Conference on Data Engineering (ICDE), p. 12 (2021)
29. Xie, M., Wong, R.C.W., Lall, A.: An experimental survey of regret minimization query and variants: bridging the best worlds between top-k query and skyline query. VLDB J. **29**, 147–175 (2020)
30. Xie, M., Wong, R.C.W., Li, J., Long, C., Lall, A.: Efficient k-regret query algorithm with restriction-free bound for any dimensionality. In: Proceedings of International Conference on Management of Data (SIGMOD), pp. 959–974 (2018)
31. Xie, M., Wong, R.C.W., Peng, P., Tsotras, V.J.: Being happy with the least: achieving α-happiness with minimum number of tuples. In: Proceedings of International Conference on Data Engineering (ICDE), pp. 1009–1020 (2020)

32. Zafar, M.B., Valera, I., Rogriguez, M.G., Gummadi, K.P.: Fairness constraints: mechanisms for fair classification. In: Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 962–970 (2017)
33. Zehlike, M., Yang, K., Stoyanovich, J.: Fairness in ranking: A survey. arXiv preprint arXiv:2103.14000 p. 58 (2021)
34. Zeighami, S., Wong, R.C.: Minimizing average regret ratio in database. In: Proceedings of International Conference on Management of Data (SIGMOD), pp. 2265–2266 (2016)