Computing Online Average Happiness Maximization Sets over Data Streams

Zhiyang Hao[1] and Jiping Zheng[1,2](✉)

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
{haozhiyang,jzh}@nuaa.edu.cn
[2] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

Abstract. Finding a small subset representing a large dataset is an important functionality in many real applications such as data mining, recommendation and web search. The average happiness maximization set problem also known as the average regret minimization set problem was recently proposed to fulfill this task and it can additionally satisfy users on average with the representative subset. In this paper, we study the online average happiness maximization set (Online-AHMS) problem over data streams where each data point should be decided to be accepted or discarded when it arrives, and the discarded data points will never be considered. We provide an efficient online algorithm named GreedyAT with theoretical guarantees for the Online-AHMS problem which greedily selects data points based on the adaptive thresholds strategy. Experimental results on the synthetic and real datasets demonstrate the efficiency and effectiveness of our GreedyAT algorithm.

Keywords: Happiness maximization set · Online algorithm · Adaptive threshold

1 Introduction

In many real applications such as data mining [27], recommendation [18,30] and web search [29], an important functionality is to select a succinct subset from a large dataset to meet the requirements of various users. To fulfill this task, three popular tools are proposed in the last three decades, namely the top-k query [17], the skyline query [5,11] and the happiness maximization set (also known as the regret minimization set, or the k-regret) query [21,35]. The top-k query is to use the concept of utility function to quantify a user's preference on different attributes and the top-k data points with the largest utilities are returned. However, the weakness of the top-k query is that the utility function is often unclear or only vaguely known which limits the applicability of this tool. By utilizing the concept of *domination*: a point p dominates a point q iff p is as good as q on all attributes and strictly better than q on at least one attribute, the skyline query returns all data points not dominated by other data points. Though no specific utility functions are needed, the skyline query does not effectively reduce the solution size over high-dimensional datasets [5].

To avoid the limitations of the top-k and skyline queries, the Happiness Maximization Set (HMS) query (called the k-regret query when first proposed [21]) was proposed

B. Li et al. (Eds.): APWeb-WAIM 2022, LNCS 13423, pp. 19–33, 2023.
https://doi.org/10.1007/978-3-031-25201-3_2

to simultaneously have the merits of both top-k and skyline queries, resulting in many studies in the database community [1,3,7,10,13,19,22,24,26,28,32,35–39]. Specifically, the happiness ratio is defined to quantify the happiness level of a user with a utility function for a set of data points, compared with when s/he has seen the entire dataset. In HMS, a subset of size k is chosen from the dataset such that the minimum happiness ratio of any possible utility function between the best data point in the selected subset and the best point in the whole dataset is maximized. The happiness maximization set query is to select a size-k subset maximizing the minimum happiness ratio. We can see that the minimum happiness ratio may unfairly prioritize the least satisfied users [19,26,38,39]. Instead, the Average Happiness Maximization Set (AHMS) query [38,39] is provided to maximize the users' happiness ratios on average, which is more proper to satisfy the majority of users. Note that the Average Happiness Maximization Set query is identical to the Average Regret Minimization Set query, but the average happiness ratio function of the AHMS query shows the property of submodularity which allows for the deviation of stronger theoretical results. For the AHMS query [19,26,38,39], the users' utility functions are not just uniformly distributed but follow a probability distribution which can be obtained from users' historical preferences and feedback provided in real applications [12,16,27,31]. Since the average happiness ratio over the utility function distribution can be approximated by sampling, the AHMS query selects a subset to maximize the average happiness ratio on a sample of M utility functions instead of on the distribution of the utility functions.

The HMS query and its variants show their NP-hardness on any dataset when the dimensionality d is larger than 2, i.e., $d \geq 3$, and many promising algorithms for the happiness maximization set query (or the regret minimization set query) exhibit their advantages to solve the related problems [1,3,7,10,13,19,22,24,26,36–39]. However, instead of considering the streaming setting where the data points arrive one by one, the existing algorithms almost aim at the static setting of the dataset where full access to the entire dataset is always available. Wang et al. [32] and Zheng et al. [42] considered the dynamic environment of datasets where data points are inserted and deleted dynamically and efficient algorithms, namely FD-RMS and DynCore were proposed to solve this problem, respectively. Ma et al. [20] assumed the data points are only valid in a sliding window and an efficient coreset-based algorithm was proposed to answer the regret minimization set query. However, these researches do not consider the scenario that an immediate decision should be made for each on-arriving data point in a data stream, i.e., whether the current data point should be included in the result set or not, and the discarded data points will never be considered. Our model is also distinct from the existing streaming setting [30] where a small portion of the data points can be buffered and added to the solution later. Moreover, the scenario in the online setting occurs in many real applications, such as news recommendation, football player recruitment, job seeking, etc. when a news is published by a news agency, a news portal (e.g., Sina News), should make an immediate decision on whether the news is to be included in the fixed-size headlines; otherwise, other news portals will have the chance to release the news at the earliest time. The same scenario happens for a football team recruiting a player, or a company employing a staff. If they miss the chance to recruit the player or employ the job-seeker, they may have no chance to consider her/him again, because the player or job-seeker may be recruited or employed by other football teams or companies.

In this paper, we study the Online Average Happiness Maximization Set (Online-AHMS) problem. Since we have no foreknowledge of future data points in the online setting, we define the online average happiness ratio function as a reasonable measure. The goal of the Online-AHMS query is to return a subset with k data points while maximizing the online average happiness ratio at any time in the online setting. To solve this problem, we provide an efficient online-selection algorithm named GreedyAT where *adaptive thresholds* are set to filter the data points. In addition, by utilizing the property that our online average happiness ratio function is submodular, we can guarantee a constant competitive ratio of the GreedyAT algorithm. To sum up, the main contributions of this paper are listed as follows:

– We study the online average happiness maximization set problem which selects the data points irrevocably on the arriving of them, and an online algorithm GreedyAT is proposed based on the adaptive thresholds strategy.
– We perform theoretical analysis of our GreedyAT algorithm and the competitive ratio of the GreedyAT algorithm is provided based on the submodularity property of our online average happiness ratio function.
– Extensive experiments on the synthetic and real datasets are conducted to verify the efficiency, effectiveness and applicability of the GreedyAT algorithm in the online setting.

The rest of the paper is organized as follows. We introduce the related work in Sect. 2. Related concepts are provided and we formally define our Online-AHMS problem in Sect. 3. In Sect. 4, we propose our GreedyAT algorithm based on the adaptive thresholds strategy and further provide the theoretical analysis to obtain the competitive ratio of the GreedyAT algorithm. We conduct experiments on the synthetic and real datasets to evaluate our GreedyAT algorithm in Sect. 5. Lastly, we conclude our work in Sect. 6.

2 Related Work

To avoid the drawbacks of the top-k query (needs users to provide exact utility functions) and the skyline query (the output size is uncontrollable), the happiness maximization set query has been investigated in the last decade which was called the k-regret query first proposed by Nanongkai et al. [21]. Due to the NP-hardness of the k-regret query [10,35], many promising methods are proposed and useful variants are introduced. Geometry-based methods, such as GeoGreedy [24] and Sphere [36] are proposed to improve the efficiency of the k-regret query. To achieve the same goal, various techniques, e.g., ϵ-kernel [1,7], hitting set [1], discretized matrix [3] are borrowed to answer the k-regret query efficiently. As useful variants of the k-regret query, the kRMS query [10], the interactive regret minimization query [22,34,40], the regret minimization query on nonlinear utility functions [13,25], the rank-based regret minimization query [4,33] and the regret minimization query with approximation guarantees [41] are proposed to meet different situations. As a completely identical concept to the k-regret query, the happiness maximization set query was recently studied in [19,26,37] where if the happiness ratio is defined as hr then the regret ratio rr is one minus hr, i.e., $rr = 1 - hr$.

The researches above related to the happiness maximization set query (or the k-regret query) focused on maximizing the minimum happiness ratio (or minimizing the maximum regret ratio) and optimizing the worst-case scenario. To address the issue that the happiness maximization set query may prioritize the least satisfied users, the average regret minimization query was proposed by Zeighami et al. [38]. They introduced the concept of the average regret ratio to measure the user's regret ratio in the average case which is more reasonable since it considers the expectations of different users. Moreover, Zeighami et al. [39] proved the average regret minimization set problem is NP-hard and an efficient algorithm Greedy-Shrink was proposed. Luenam et al. [19] showed that the happiness maximization set query can admit stronger approximation guarantees than the regret minimization set query.

Although the massive volume and the real-time generation of data points imply a growing need for non-static algorithms, few researches considered the this setting of the happiness maximization set query or the k-regret query. Wang et al. [32] studied the k-regret query on dynamic datasets where data points were arbitrarily inserted or deleted. They provided the FD-RMS algorithm which transformed the fully-dynamic k-regret query to a dynamic set cover problem and constantly maintained the result with theoretical guarantees. Further, Zheng et al. [42] also considered the same problem and a more efficient method named DynCore was proposed to achieve a better regret ratio bound with a lower time complexity. Ma et al. [20] assumed there was a sliding window on a data stream and the data points inside the window were valid. They provided a coreset-based method to continuously maintain the result efficiently. In the online setting, however, upon the arrival of a data point, we need to decide whether to accept it or not immediately as an online selection which usually occurs in many real applications. The above researches do not consider this scenario because the data points are buffered without immediate decisions. In this paper, we study the Online-AHMS problem which is very promising in real-world applications.

3 Problem Definition

In this section, we formally define the online average happiness maximization set (Online-AHMS) problem. Some useful concepts such as utility function, happiness ratio, average happiness ratio and their online versions are introduced before we state our problem. Let D be a d-dimensional dataset containing n data points where each data point $p =< p[1], p[2], \ldots, p[d] >\in D$ is described by d numerical attributes which are normalized in the range $[0, 1]$. We assume that a larger value in each dimension is preferable to all users.

Definition 1 (Utility Function). *A utility function f is a mapping $f\colon \mathbb{R}^d_+ \rightarrow \mathbb{R}_+$ that assigns a non-negative utility $f(p)$ to each data point $p \in D$ which shows how satisfied the user is with the data point p.*

Following [10,21], we assume the form of the utility functions to be linear, i.e., f is represented as a d-dimensional vector $u =< u[1], u[2], \ldots, u[d] >\in \mathbb{R}^d_+$ where $u[i]$ denotes the importance of the i-th dimension in user's happiness. W.l.o.g., we assume that u is normalized such that $||u||_1 = \sum_{i=1}^{d} u[i] = 1$. Thus, the utility of a data point $p \in D$ w.r.t. f can be expressed as $f(p) = u \cdot p = \sum_{i=1}^{d} u[i]p[i]$.

Definition 2 (Happiness Ratio). *Given a subset $S \subseteq D$ and a user's utility function f, the happiness ratio of S over D for f, denoted by $hr_{D,f}(S)$, is defined to be $\frac{\max_{p \in S} f(p)}{\max_{p \in D} f(p)}$.*

Intuitively, we have $\max_{p \in S} f(p) \leq \max_{p \in D} f(p)$, so the happiness ratio of a user ranges from 0 to 1. The happiness ratio measures how happy a user is if s/he sees the subset S instead of the entire dataset D. The happiness ratio of a user is closer to 1 which indicates that the user feels happier with the selected subset S.

However, the users are generally difficult or not willing to provide their utility functions explicitly. We thus assume that all users' utility functions belong to a utility function class, denoted by $\mathcal{FC}$. Therefore, $\mathcal{FC}$ is the set of all possible linear utility functions, i.e., $\mathcal{FC} = \{f | f(p) = u \cdot p\}$. We focus on that the users' utility functions in $\mathcal{FC}$ follow a probability distribution Θ. Unless specified otherwise, we consider arbitrary type of distribution. Let $\eta(f)$ be the pobability density function for utility functions f in $\mathcal{FC}$ corresponding to Θ. Next we formally define the average happiness ratio.

Definition 3 (Average Happiness Ratio). *Given a subset $S \subseteq D$ and a utility function class $\mathcal{FC}$ with the probability density function $\eta(.)$ corresponding to a probability distribution Θ, the average happiness ratio of S is defined as*

$$ahr_{D,\mathcal{FC}}(S) = \int_{f \in \mathcal{FC}} hr_{D,f}(S) \cdot \eta(f) df. \tag{1}$$

Unfortunately, the set of all possible linear utility functions $\mathcal{FC}$ is uncountable, so evaluating the above average happiness ratio needs to compute an integral over $\mathcal{FC}$, which is very time-consuming. Hence, we utilize a sampling technique from [39] to compute the average happiness ratio in Definition 3 with a theoretical bound. Specifically, we sample M utility functions according to the distribution Θ and obtain a sampled utility function class $\mathcal{FC}_M = \{f_1, f_2, \ldots, f_M\}$ where $|\mathcal{FC}_M| = M$. With $\mathcal{FC}_M$, we compute the estimated average happiness ratio by averaging the happiness ratio of the M sampled utility functions, as follows,

$$ahr_{D,\mathcal{FC}_M}(S) = \frac{1}{M} \sum_{f \in \mathcal{FC}_M} \frac{\max_{p \in S} f(p)}{\max_{p \in D} f(p)}. \tag{2}$$

It holds that $ahr_{D,\mathcal{FC}_M}(S) \in [0, 1]$, and when $ahr_{D,\mathcal{FC}_M}(S)$ is close to 1, it indicates that the subset S satisfies the majority of users with utility functions in $\mathcal{FC}_M$. We claim that this estimated average happiness ratio $ahr_{D,\mathcal{FC}_M}(S)$ differs from the exact average happiness ratio $ahr_{D,\mathcal{FC}}(S)$ by an error $\epsilon \in [0, 1]$ (Theorem 4 in [39]). In the following, we ignore the error ϵ and only focus on maximizing $ahr_{D,\mathcal{FC}_M}(S)$ instead of $ahr_{D,\mathcal{FC}}(S)$ and further provide the average happiness maximization set problem based on the sampled utility function class $\mathcal{FC}_M$.

Definition 4 (Average Happiness Maximization Set, AHMS). *Given a dataset $D \in \mathbb{R}^d_+$ with n data points, a user-specified positive integer k, and the sampled utility function class $\mathcal{FC}_M$, we want to find a set $S \subseteq D$ containing at most k data points such that $ahr_{D,\mathcal{FC}_M}(S)$ is maximized, i.e.,*

$$S = \underset{S' \subseteq D : |S'| \leq k}{\arg\max}\ ahr_{D,\mathcal{FC}_M}(S'). \tag{3}$$

In this paper, we focus on solving the Online-AHMS problem. We first present some descriptions of our online model, then we give the formal definition of the Online-AHMS problem based on the model.

In the online setting, the data points of D are revealed one by one in an online fashion. At each timestamp $t \in \{1, 2, \ldots, n\}$, a data point $p_t \in D$ is revealed, the algorithm must immediately decide whether to accept p_t into the solution set S_{t-1}, which is initially empty.

Moreover, we relax the model by allowing the algorithm to preempt previously accepted data points. That is, when the algorithm adds the newly revealed data point p_t into the current solution S_{t-1}, a data point may be removed from S_{t-1}, as long as this preemption can improve the solution quality. We cannot reconsider those data points that have been rejected from the solution or have been discarded because of preemption.

In the online setting, since at any timestamp $t \in \{1, 2, \ldots, n\}$, there are only a portion of the data points $D_t = \{p_1, p_2, \ldots, p_t\}$ revealed. We have no foreknowledge of the future data points and only select a subset S from D_t, so we would like to find a practical method to compute $\max_{p \in D_t} f(p)$ for each utility function $f \in \mathcal{FC}_M$. Note that this actually calculates the maximum utility among all data points in D_t for f, the natural idea is to maintain an auxiliary variable $m_{f,t}$ which holds the current maximum utility for each utility function f after the point p_t is revealed. Further, we utilize $m_{f,t}$ to calculate the happiness ratio of S over D_t for f, i.e., $hr_{D_t,f}(S) = \frac{\max_{p \in S} f(p)}{m_{f,t}}$.

With the above analysis, we next reformalize the average happiness ratio in the online setting and define the online average happiness ratio.

Definition 5 (Online Average Happiness Ratio). *Given a portion of the revealed data points $D_t = \{p_1, p_2, \ldots, p_t\}$ at timestamp $t \in \{1, 2, \ldots, n\}$, a sampled utility function class $\mathcal{FC}_M = \{f_1, f_2, \ldots, f_M\}$ and a subset $S \subseteq D_t$, the online average happiness ratio is defined as*

$$\widehat{ahr}_{\mathcal{FC}_M,t}(S) = \frac{1}{M} \sum_{f \in \mathcal{FC}_M} \frac{\max_{p \in S} f(p)}{m_{f,t}}, \tag{4}$$

where $m_{f,t}$ is an auxiliary variable which holds the current maximum utility for the utility function $f \in \mathcal{FC}_M$ over D_t at timestamp t.

Obviously, the value of $\widehat{ahr}_{\mathcal{FC}_M,t}(S)$ does not depend on the entire unknown data stream D, but on the current maximum utility $m_{f,t}$ for each f and the subset S, which makes it an ideal objective function for selecting representative data points in the online setting. In fact, the online average happiness ratio defined above still satisfies submodularity which we will prove in Sect. 4.1, thus our model is reasonable in practice.

Definition 6 (Online Average Happiness Maximization Set, Online-AHMS). *Given an unknown data stream D with n data points, a user-specified positive integer k, and the sampled utility function class $\mathcal{FC}_M = \{f_1, f_2, \ldots, f_M\}$. The Online-AHMS problem returns a feasible selected set S_t with $|S_t| \le k$ over the currently revealed data points $D_t \subseteq D$ such that the online average happiness ratio $\widehat{ahr}_{\mathcal{FC}_M,t}(S_t)$ is maximized at any timestamp $t \in \{1, 2, \ldots, n\}$, i.e.,*

$$S_t = \underset{S' \subseteq D_t : |S'| \le k}{\arg\max} \ \widehat{ahr}_{\mathcal{FC}_M,t}(S'). \tag{5}$$

Unfortunately, according to the existing results in [39], the Online-AHMS problem is NP-hard for any $d \geq 3$. Thus, it is unlikely to have a polynomial-time algorithm that solves the Online-AHMS problem optimally, unless P=NP. Hence, we focus on designing the approximate algorithm for the Online-AHMS problem in this paper.

4 The GreedyAT Algorithm

In this section, we will first introduce *monotonicity* and *submodularity* properties of the online average happiness ratio function and show how these properties can be used in designing our approximation algorithm. Then, we present our GreedyAT algorithm for the Online-AHMS problem.

4.1 Properties

Given a dataset D, a non-negative set function g: $2^D \rightarrow \mathbb{R}_+$ and a subset $S \subseteq D$, a set function g is naturally associated with a marginal gain $\varDelta g(p|S) := g(S \cup \{p\}) - g(S)$, which represents the increase of $g(S)$ when adding a data point $p \in D \backslash S$ to S.

Definition 7 (Monotonicity). *A set function g is monotone if and only if for any $S \subseteq D$ and $p \in D \backslash S$, it holds that $\varDelta g(p|S) \geq 0$.*

Definition 8 (Submodularity). *A set function g is submodular if and only if for any $S \subseteq T \subseteq D$ and $p \in D \backslash S$, it holds that $\varDelta g(p|S) \geq \varDelta g(p|T)$.*

A set function is submodular if the gain of adding a data point p to a set S is always no less than the gain of adding the same data point to a superset of S. In [38], it was shown that the estimated average regret ratio $arr_{\mathcal{FC}_M}(S)$ is monotonically decreasing and supermodular. Since $ahr_{\mathcal{FC}_M}(S) = 1 - arr_{\mathcal{FC}_M}(S)$, it trivially follows that $ahr_{\mathcal{FC}_M}(S)$ is monotonically increasing and submodular. Next, we show the online average happiness ratio function $\widehat{ahr}_{\mathcal{FC}_M,t}(S)$ satisfies above two properties.

Lemma 1. *$\widehat{ahr}_{\mathcal{FC}_M,t}(S)$ is a monotonically increasing function.*

Proof. At timestamp $t \in \{1, 2, \ldots, n\}$, the data point p_t is revealed, $D_t \subseteq D$ is the set containing all currently revealed data points. $\widehat{ahr}_{\mathcal{FC}_M,t}(S) = \frac{1}{M} \sum_{f \in \mathcal{FC}_M} \frac{\max_{p \in S} f(p)}{m_{f,t}}$. For each utility function $f \in \mathcal{FC}_M$, the denominator $m_{f,t}$ is a fixed value, because it represents the maximum utility of all data points in D_t on f at this timestamp t. Let S and T be subsets of D_t, where T is a superset of S. It clearly yields that $\max_{p \in S} f(p) \leq \max_{p \in T} f(p)$ as the additional data points in $T \backslash S$ can only contribute larger utility for each f. So, we can easily have $\widehat{ahr}_{\mathcal{FC}_M,t}(S) \leq \widehat{ahr}_{\mathcal{FC}_M,t}(T)$.

Lemma 2. *$\widehat{ahr}_{\mathcal{FC}_M,t}(S)$ is a submodular function.*

Proof. Similar to Lemma 1, at timestamp $t \in \{1, 2, \ldots, n\}$, the data point p_t is revealed, and $D_t \subseteq D$ is the set that contains all currently revealed data points. We need to show that for all $S \subseteq T \subseteq D$ and for any data point $p \in D_t \backslash T$,

$\Delta\widehat{ahr}_{\mathcal{FC}_M,t}(p|S) \geq \Delta\widehat{ahr}_{\mathcal{FC}_M,t}(p|T)$. To do so, consider a data point $p \in D_t \backslash T$. There are two possibilities depending on whether p is the best data point in S for any user with utility function $f \in \mathcal{FC}_M$ or not. If p is not the best data point in $S \cup \{p\}$ (and consequently, since $S \subseteq T$, p is not the best data point in $T \cup \{p\}$ either) for any utility function, $\Delta\widehat{ahr}_{\mathcal{FC}_M,t}(p|S)$ and $\Delta\widehat{ahr}_{\mathcal{FC}_M,t}(p|T)$ are both zero which proves the result in this case.

Otherwise, if p is the best data point in $S \cup \{p\}$ for some utility functions, then by the definition of the online average happiness ratio, we easily have $\Delta\widehat{ahr}_{\mathcal{FC}_M,t}(p|S) = \frac{1}{M}\sum_{f \in F} \frac{\max_{p \in S \cup \{p\}} f(p) - \max_{p \in S} f(p)}{m_{f,t}}$, where F is the set of utility functions whose best data points change when p is added to S. Similarly, we have $\Delta\widehat{ahr}_{\mathcal{FC}_M,t}(p|T) = \frac{1}{M}\sum_{f \in F} \frac{\max_{p \in T \cup \{p\}} f(p) - \max_{p \in T} f(p)}{m_{f,t}}$ holds for the same reason. And if the best data point of a user changes when p is added to T, the utility function must be in F since S is a subset of T. We can show that $\Delta\widehat{ahr}_{\mathcal{FC}_M,t}(p|S)$ is larger than or equal to $\Delta\widehat{ahr}_{\mathcal{FC}_M,t}(p|T)$, which implies that $\widehat{ahr}_{\mathcal{FC}_M,t}(S)$ is a submodular function.

4.2 The Algorithm

In this section, we present our greedy online-selection algorithm with adaptive thresholds, i.e., GreedyAT for the Online-AHMS problem.

According to the lemmas in Sect. 4.1, $\widehat{ahr}_{\mathcal{FC}_M,t}(S)$ is a monotonically increasing and submodular function. Thus, we can transform the Online-AHMS problem to an online submodular maximization problem under a cardinality constraint, which has been recently studied extensively [6,8,9]. The current state-of-the-art solution is the online algorithm *Preemption* [6] with a competitive ratio (approximation ratio for online algorithms) of at least $1/4$. Initially, Preemption accepts the first k revealed data points sequentially and gets the solution set S_k. When the (k+1)-th data point is revealed, the Preemption algorithm needs to find a swap which replaces some data point in S_k. However, the swap happens only when the increased value of the solution, i.e., $\widehat{ahr}_{\mathcal{FC}_M,k+1}(S_k \cup \{p_{k+1}\} - \{p_i\}) - \widehat{ahr}_{\mathcal{FC}_M,k}(S_k)$ is large enough to pass a given threshold $c \cdot \widehat{ahr}_{\mathcal{FC}_M,k}(S_k)/k$ where $p_i \in S_k$ and c is a given positive constant, i.e., $c > 0$. According to Corollary 4.3 in [6], it is known that Preemption provides a $\frac{c}{(c+1)^2}$-competitive ratio, and the best competitive ratio is 1/4 when $c = 1$. However, once the parameter c is fixed, it can be obviously observed that after several swaps, the value of the solution set will reach a high value and consequent swaps have less chance to increase this value. Thus, the solution quality is not satisfactory.

To improve the quality of the solution set when swapping the on-arriving data point p_t with the data point in the current solution set S_{t-1}, we adaptively set the thresholds based on the fact that n is known because it is obvious that in the scenarios in Sect. 1, the numbers of the candidate players, employees and coming news can be predicated or are known in advance. Our adaptive thresholds strategy is that we partition the stream into two parts and the first part is set with a larger threshold according to the parameter c_1 and the second part with a smaller threshold corresponding to the parameter c_2 which is smaller than c_1, i.e., $c_2 < c_1$. The strategy is simple but empirically effective. To do this, we introduce a balancing parameter $\beta \in (0, 1)$, such that the first part is with the points $\{p_{k+1}, \ldots, p_{\lfloor \beta n \rfloor}\}$ while the second part contains the points $\{p_{\lceil \beta n \rceil}, \ldots, p_n\}$ (or $\{p_{\lceil \beta n \rceil + 1}, \ldots, p_n\}$ when $\lceil \beta n \rceil$ is an integer). When the data points in the first part $\{p_{k+1}, \ldots, p_{\lfloor \beta n \rfloor}\}$ are revealed, we use a higher threshold with parameter c_1 to determine the happening of the swap while when the data points in the second part $\{p_{\lceil \beta n \rceil}, \ldots, p_n\}$ are revealed, we use a lower threshold with parameter c_2 to decide whether to swap or not. When the swap happens, we greedily select the point p_i in S_{t-1} to be replaced to maximize the increased value and p_i will be discarded forever. Based on the above idea, we provide our greedy online-selection algorithm with adaptive thresholds, GreedyAT as shown in Algorithm 1.

Algorithm 1: GreedyAT

```
Input: Data stream $D = \{p_1, p_2, \ldots, p_n\}$, sampled utility function class
    $\mathcal{FC}_M = \{f_1, f_2, \ldots, f_M\}$, user-specified positive integer $k$, balancing parameter
    $\beta \in (0, 1)$, and threshold parameters $c_1 \geq c_2 > 0$
Output: The solution set $S_t$ at timestamp $t$
1  $S_0 \leftarrow \emptyset$ ;
2  for $t \leftarrow 1, \ldots, n$ do
3      if $t \leq k$ then
4          $S_t \leftarrow S_{t-1} \cup \{p_t\}$ ;
5      else
6          if $t \leq \lfloor \beta n \rfloor$ then
7              $c \leftarrow c_1$ ;
8          else if $t \geq \lceil \beta n \rceil$ then
9              $c \leftarrow c_2$ ;
10         Let $p_i$ be the point in $S_{t-1}$ maximizing $\widehat{ahr}_{\mathcal{FC}_M,t}(S_{t-1} \cup \{p_t\} \setminus \{p_i\})$;
11         if $\widehat{ahr}_{\mathcal{FC}_M,t}(S_{t-1} \cup \{p_t\} \setminus \{p_i\}) - \widehat{ahr}_{\mathcal{FC}_M,t}(S_{t-1}) \geq c \cdot \widehat{ahr}_{\mathcal{FC}_M,t}(S_{t-1})/k$ then
12             $S_t \leftarrow S_{t-1} \cup \{p_t\} \setminus \{p_i\}$ ;
13         else
14             $S_t \leftarrow S_{t-1}$ ;
15 return $S_t$;
```

In Algorithm 1, when $t \leq k$, it is obvious that the data points revealed are all selected into the solution set (Lines 3–4). Otherwise, two thresholds along with parameters c_1, c_2 are set for the data points in $\{p_{k+1}, \ldots, p_{\lfloor \beta n \rfloor}\}$ and $\{p_{\lceil \beta n \rceil}, \ldots, p_n\}$, respectively (Lines 6–10). We greedily online select the point p_i in S_{t-1} with the maximum increased value after the swap (Line 10). If the increased value overpasses the threshold, we replace p_i with the current revealed data point p_t (Lines 11–12). Note that when the data point p_t is revealed, we calculate the utility of p_t for f, i.e., $f(p_t)$, and update the current maximum utility $m_{f,t}$ by $m_{f,t} \leftarrow \max\{m_{f,t}, f(p_t)\}$ which is used to compute $\widehat{ahr}_{\mathcal{FC}_M,t}(S)$.

Our GreedyAT algorithm improves the Preemption algorithm due to the fact that after first several swaps, the online average happiness ratio will reach a high value (almost equal to 1 in most cases) and a smaller threshold will improve the solution quality inevitably.

4.3 Theoretical Analysis

In this section, we provide the competitive ratio of the GreedyAT algorithm.

Theorem 1. *The competitive ratio of GreedyAT is at least* $\min\{\frac{c_1}{(c_1+1)^2}, \frac{c_2}{(c_2+1)^2}\}$.

Proof. Let S_t^* denote the subset of size at most k that maximizes the online average happiness ratio at timestamp $t \in \{1, 2, \ldots, n\}$, i.e., the optimal solution with the value $OPT = \widehat{ahr}_{\mathcal{FC}_M,t}(S_t^*)$. Buchbinder et al. [6] proved that their algorithm which compares each preemption against a threshold determined by a fixed parameter $c > 0$ without the adaptive thresholds strategy, yields a $\frac{c}{(c+1)^2}$-competitive ratio. At any timestamp $t \in \{1, \ldots, p_{\lfloor \beta n \rfloor}\}$, the GreedyAT algorithm outputs a selected subset S_t such that $|S_t| \leq k$ and $\widehat{ahr}_{\mathcal{FC}_M,t}(S_t) \geq \frac{c_1}{(c_1+1)^2} \cdot OPT$. And at any timestamp $t \in \{p_{\lceil \beta n \rceil}, \ldots, n\}$, the GreedyAT algorithm outputs a selected subset S_t such that $|S_t| \leq k$ and $\widehat{ahr}_{\mathcal{FC}_M,t}(S_t) \geq \min\{\frac{c_1}{(c_1+1)^2}, \frac{c_2}{(c_2+1)^2}\} \cdot OPT$. Hence, the competitive ratio of the GreedyAT algorithm is at least $\min\{\frac{c_1}{(c_1+1)^2}, \frac{c_2}{(c_2+1)^2}\}$.

Besides the competitive ratio derived (Theorem 1), our GreedyAT algorithm also has the advantages of current graceful online submodular maximization algorithms [6, 8, 9]. GreedyAT only needs one pass over the data stream and the space complexity is $O(k)$ ($O(1)$ if k is a constant).

5 Experimental Evaluation

In this section, we experimentally evaluate the performance of our proposed algorithm GreedyAT for solving the Online-AHMS problem on both synthetic and real datasets. We first introduce the experimental setup in Sect. 5.1. Then, we present the experimental results in Sect. 5.2.

5.1 Setup

All algorithms were implemented in C++. The experiments were conducted on a machine running Ubuntu 16.04 with an Intel Core i7-5500 CPU and an 8GB RAM.

Datasets. Due to space limitation, we run our experiments only on 1 synthetic dataset and 1 real dataset which are both popular in the literature [1, 2, 21, 36, 37]. The datasets used in our experiments are listed as follows:

- **Anti-correlated**. The Anti-correlated dataset is a synthetic dataset generated by the synthetic dataset generator [5]. The dataset contains 10,000 random data points with 4 anti-correlated attributes.
- **Tweet**. The Tweet dataset is a real dataset for streaming applications and we adapted it for our comparison. The dataset was obtained from an archive website[1]. In the Tweet dataset, the data points describe tweets delivered during a certain period and the goal is to select the most popular tweets of a fixed size. After pre-processing, we

[1] https://web.archive.org/.

selected 12,197 tweets described by 7 attributes, namely *TweetID*, *UserID*, *FollowerCount*, *FollowingCount*, *ReplyCount*, *LikeCount* and *RetweetCount*. Among them, the first two attributes identify a specific tweet, while the last five attributes characterize the popularity of the tweet and are used to conduct our experiments. The detailed meaning of each attribute is: (1) *TweetID* is the tweet identifier, (2) *UserID* is the anonymized user identifier, (3) *FollowerCount* is the number of accounts following the user, (4) *FollowingCount* is the number of accounts followed by the user, (5) *ReplyCount* is the number of tweets replying to this tweet, (6) *LikeCount* is the number of likes that this tweet received, (7) *RetweetCount* is the number of retweets that this tweet received.

For these two datasets, all attributes of the data points are normalized into [0,1] after pre-processing. We considered the entire dataset as a data stream and processed the data points in the order.

Algorithms. The algorithms compared are listed as follows:

- **GREEDY**: The GREEDY algorithm is a classical algorithm for the offline submodular maximization problem under a cardinality constraint which achieves a $(1-1/e)$ approximation factor [23]. GREEDY iterates k times over the entire dataset and greedily selects the data point with the largest marginal gain in each iteration. Although GREEDY is not an online algorithm, we used GREEDY as the benchmark algorithm providing typically the best solution quality, which allows us to compare the relative performance of other algorithms.
- **Random**: A randomized algorithm with replacement via Reservoir Sampling [14]. Originally, Random is a random sampling of k data points as a solution. We adapted it in the online setting by unconditionally accepting the first k data points and using Reservoir Sampling [14] to randomly swap a data point in the current solution. For constraint maximization problems, it cannot return a solution with theoretical guarantees. In spite of this, we still considered the empirical performance of Random as a simple baseline.
- **StreamGreedy**: A variation of GREEDY for the streaming cardinality-constraint submodular maximization problem in [15]. In the online setting, StreamGreedy unconditionally accepts the first k data points and replaces the newly revealed data point with any data point in the solution if the replacement can improve the current solution by a constant threshold $\eta > 0$.
- **Preemption**: Current state-of-the-art algorithm for the online submodular maximization problem in [6]. Preemption achieves a competitive ratio of $\frac{c}{(c+1)^2}$, where $c > 0$ is a fixed parameter. It works similarly to StreamGreedy, but instead of using a constant threshold η, it uses a more suitable threshold, which depends on the current solution. Unless stated explicitly, we set the value of parameter c in Preemption to 1, such that Preemption can achieve the best competitive ratio.
- **GreedyAT**: Our Online-AHMS algorithm based on the preemption and adaptive thresholds strategy proposed in Sect. 4.

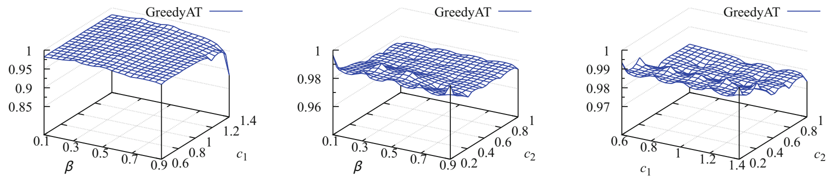
To evaluate the online average happiness ratio function of a solution for each algorithm above, we only considered that utility functions used are linear and the learned probability distribution of the utility functions is uniform. By default, the sample size

M of the utility functions is set to $1,000$, i.e., $M = 1,000$. The algorithms are evaluated from two perspectives, namely the average happiness ratio (AHR) and the CPU time.

5.2 Experimental Results

We first evaluate the performance of the GreedyAT algorithm when varying the parameters, by changing two values of the balanced parameter β and the threshold parameters c_1 and c_2. Since the change of parameter values in GreedyAT only affects the quality of the solution, we ignore the results for the CPU time of GreedyAT. Moreover, we only show the experimental results on the real dataset Tweet due to space limitation, and fix the solution size k to 20.

Effect of Parameters β, c_1, and c_2. Figure 1(a) shows the effect of parameters β and c_1 on GreedyAT. We report the average happiness ratios (AHR) of GreedyAT for $c_2 = 0.4$ by varying $\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $c_1 \in \{0.6, 0.8, 1.0, 1.2, 1.4\}$. First of all, for any c_1, the quality of solutions is better when the value of β is 0.3 or 0.5, but degrades if β is too small or too large. This shows that the larger or smaller value of β will lead to the malfunction of our adaptive thresholds strategy, because the effectiveness of GreedyAT with the thresholds under larger or smaller β value is same as that there is only one threshold c_1 or c_2. On the other hand, when β is fixed, the AHR decreases as c_1 becomes larger. This is due to the fact that we have discarded some important data points when c_1 is large.



(a) varying β and c_1 (b) varying β and c_2 (c) varying c_1 and c_2

Fig. 1. Performance of GreedyAT under different adaptive thresholds ($k = 20$)

Figure 1(b) shows the effect of parameters β and c_2 on GreedyAT. We report the AHR of GreedyAT for $c_1 = 1.0$ by varying $\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $c_2 \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$. Similarly, we can observe that the AHR decreases when c_2 is larger. This illustrates the marginal gains of the remaining data points could be small when a portion of the data stream is accessed. Moreover, if the value of β is moderate, the solution quality will increase, which shows that it is pathological for β to be too large or too small.

Figure 1(c) shows the effect of parameters c_1 and c_2 on GreedyAT. We report the AHR of GreedyAT for $\beta = 0.5$ by varying $c_1 \in \{0.6, 0.8, 1.0, 1.2, 1.4\}$ and $c_2 \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$. When fixing c_2, for larger c_1, the solution quality of GreedyAT remains relatively stable. But if c_1 is fixed, the AHR becomes smaller by increasing c_2.

Thus, considering the theoretical guarantee and practical performance, the values of three parameters (β, c_1, and c_2) in GreedyAT will be properly decided in the remaining experiments.

Effect of Solution Size k. We report the performance of all the algorithms on synthetic and real datasets by varying $k \in \{10, 15, 20, 25, 30\}$. We present two groups of experiments with different parameter settings. In the first group of the experiments, we set the parameter $\eta = 0.1$ in StreamGreedy, $c = 1.0$ in Preemption, and $\beta = 0.5, c_1 = 1.0, c_2 = 0.4$ in GreedyAT. While in the second group of the experiments, we set the parameter $\eta = 0.05$ in StreamGreedy, $c = 0.8$ in Preemption, and $\beta = 0.7, c_1 = 1.2, c_2 = 0.2$ in GreedyAT.

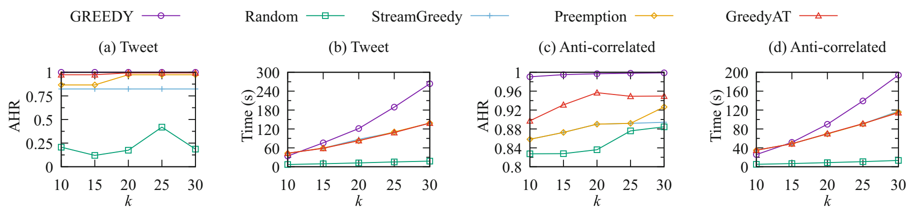


Fig. 2. The performance of all algorithms with varying k ($\eta = 0.1$, $c = 1.0$ and $\beta = 0.5, c_1 = 1.0, c_2 = 0.4$)

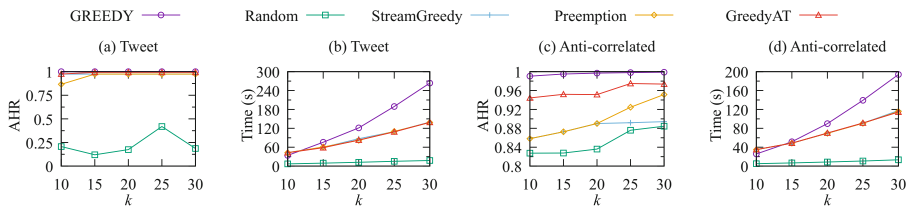


Fig. 3. The performance of all algorithms by varying k ($\eta = 0.05$, $c = 0.8$ and $\beta = 0.7, c_1 = 1.2, c_2 = 0.2$)

As shown in Figs. 2 and 3, generally the average happiness ratio and the query time of each algorithm grow with the increase of k. We can see that Random is the fastest algorithm, but Random shows the weakest quality, especially on the real dataset, Tweet. Only the AHR of Random decreases when k increases (i.e., $k = 30$). Compared with GREEDY, GreedyAT takes less CPU time except in the case of $k = 10$ on the Tweet and Anti-correlated datasets, and the superiority is more obvious when k is larger. Therefore, GreedyAT shows higher efficiency than GREEDY, especially for larger k. On the other hand, the only difference between the GreedyAT algorithm and the Preemption and StreamGreedy algorithms is the use of the adaptive thresholds strategy, so the CPU time is almost the same for these algorithms.

In terms of the solution quality (i.e., AHR) as depicted in Figs. 2 and 3, the quality of GREEDY is the best among all algorithms. This is obvious for GREEDY has accessed the whole stream many times. Also, this also can be seen as the cost of the online algorithms, i.e., StreamGreedy, Preemption and GreedyAT, which only access the stream once. However, compared with Preemption and StreamGreedy, the solution quality of GreedyAT is generally closer to that of GREEDY for different k. GreedyAT almost has the same solution quality as GREEDY on the Tweet dataset (Figs. 2(a) and 3(a)) and provides solutions of at least 90% quality of GREEDY on the Anti-correlated dataset (Figs. 2(c) and 3(c)). This confirms the effectiveness of the adaptive thresholds strategy we used in GreedyAT. Hence, GreedyAT generally outperforms Preemption and StreamGreedy.

6 Conclusion

In this paper, we studied the online average happiness maximization set (Online-AHMS) problem and formulated our problem as an online submodular maximization problem under the cardinality constraint. Then we provided an efficient online algorithm called GreedyAT based on the adaptive thresholds strategy to solve the Online-AHMS problem. Our proposed GreedyAT has been proved with a constant competitive ratio. Extensive experimental results on the synthetic and real datasets confirmed the efficiency, effectiveness, and applicability of our proposed algorithm.

References

1. Agarwal, P.K., Kumar, N., Sintos, S., Suri, S.: Efficient algorithms for k-regret minimizing sets. In: SEA, pp. 7:1–7:23 (2017)
2. Alami, K., Maabout, S.: A framework for multidimensional skyline queries over streaming data. Data Knowl. Eng. **127**, 101792 (2020)
3. Asudeh, A., Nazi, A., Zhang, N., Das, G.: Efficient computation of regret-ratio minimizing set: a compact maxima representative. In: SIGMOD, pp. 821–834 (2017)
4. Asudeh, A., Nazi, A., Zhang, N., Das, G., Jagadish, H.V.: RRR: rank-regret representative. In: SIGMOD, pp. 263–280 (2019)
5. Börzsöny, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430 (2001)
6. Buchbinder, N., Feldman, M., Schwartz, R.: Online submodular maximization with preemption. ACM Trans. Algorithms. **15**(3), 30:1–30:31 (2019)
7. Cao, W., et al.: k-regret minimizing set: Efficient algorithms and hardness. In: ICDT, pp. 11:1–11:19 (2017)
8. Chakrabarti, A., Kale, S.: Submodular maximization meets streaming: matchings, matroids, and more. Math. Program. **154**(1–2), 225–247 (2015)
9. Chan, T.H.H., Huang, Z., Jiang, S.H.C., Kang, N., Tang, Z.G.: Online submodular maximization with free disposal. ACM Trans. Algorithms. **14**(4), 56:1–56:29 (2018)
10. Chester, S., Thomo, A., Venkatesh, S., Whitesides, S.: Computing k-regret minimizing sets. In: VLDB, pp. 389–400 (2014)
11. Chomicki, J., Ciaccia, P., Meneghetti, N.: Skyline queries, front and back. SIGMOD Rec. **42**(3), 6–18 (2013)
12. Chu, W., Ghahramani, Z.: Preference learning with gaussian processes. In: ICML, pp. 137–144 (2005)
13. Faulkner, T.K., Brackenbury, W., Lall, A.: k-regret queries with nonlinear utilities. In: VLDB, pp. 2098–2109 (2015)
14. Feige, U., Mirrokni, V.S., Vondrak, J.: Maximizing non-monotone submodular functions. SIAM J. Comput. (SICOMP) **40**(4), 1133–1153 (2011)
15. Gomes, R., Krause, A.: Budgeted nonparametric learning from data streams. In: ICML, pp. 391–398 (2010)
16. Houlsby, N., Huszar, F., Z. Ghahramani, Z., Hernández-lobato, J.: Collaborative gaussian processes for preference learning. In: NIPS, pp. 2096–2104 (2012)

17. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing techniques in relational database systems. CSUR. **40**(4), 11:1–11:58 (2008)
18. Li, Y., et al.: Hyperbolic hypergraphs for sequential recommendation. In: CIKM, pp. 988–997 (2021)
19. Luenam, P., Chen, Y.P., Wong, R.C.: Approximating happiness maximizing set problems. CoRR abs/2102.03578, pp. 1–13 (2021)
20. Ma, W., Zheng, J., Hao, Z.: A coreset based approach for continuous k-regret minimization set queries over sliding windows. In: WISA, pp. 49–61 (2021)
21. Nanongkai, D., Sarma, A., Lall, A., Lipton, R., Xu, J.: Regret-minimizing representative databases. In: VLDB, pp. 1114–1124 (2010)
22. Nanongkai, D., Lall, A., Das Sarma, A., Makino, K.: Interactive regret minimization. In: SIGMOD, pp. 109–120 (2012)
23. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions-i. Math. Program. **14**(1), 265–294 (1978)
24. Peng, P., Wong, R.C.W.: Geometry approach for k-regret query. In: ICDE, pp. 772–783 (2014)
25. Qi, J., Zuo, F., Samet, H., Yao, J.C.: k-regret queries using multiplicative utility functions. TODS. **43**(2), 10:1–10:41 (2018)
26. Qiu, X., Zheng, J., Dong, Q., Huang, X.: Speed-up algorithms for happiness-maximizing representative databases. In: APWebWAIM DS Workshop, pp. 321–335 (2018)
27. Qu, M., Ren, X., Han, J.: Automatic synonym discovery with knowledge bases. In: KDD, pp. 997–1005 (2017)
28. Storandt, S., Funke, S.: Algorithms for average regret minimization. In: AAAI, pp. 1600–1607 (2019)
29. Stoyanovich, J., Yang, K., Jagadish, H.: Online set selection with fairness and diversity constraints. In: EDBT, pp. 241–252 (2018)
30. Wang, Y., Li, Y., Tan, K.: Efficient representative subset selection over sliding windows. TKDE **31**(7), 1327–1340 (2019)
31. Wang, Y., Mathioudakis, M., Li, Y., Tan, K.: Minimum coresets for maxima representation of multidimensional data. In: PODS, pp. 138–152 (2021)
32. Wang, Y., Li, Y., Wong, R.C.W., Tan, K.L.: A fully dynamic algorithm for k-regret minimizing sets. In: ICDE, pp. 1631–1642 (2021)
33. Xiao, X., Li, J.: Rank-regret minimization. CoRR abs/2111.08563, pp. 1–15 (2021)
34. Xie, M., Wong, R.C.W., Lall, A.: Strongly truthful interactive regret minimization. In: SIGMOD, pp. 281–298 (2019)
35. 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)
36. 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: SIGMOD, pp. 959–974 (2018)
37. 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: ICDE, pp. 1009–1020 (2020)
38. Zeighami, S., Wong, R.C.W.: Minimizing average regret ratio in database. In: SIGMOD, pp. 2265–2266 (2016)
39. Zeighami, S., Wong, R.C.: Finding average regret ratio minimizing set in database. CoRR abs/1810.08047, pp. 1–18 (2018)
40. Zheng, J., Chen, C.: Sorting-based interactive regret minimization. In: APWeb-WAIM, pp. 473–490 (2020)
41. Zheng, J., Dong, Q., Wang, X., Zhang, Y., Ma, W., Ma, Y.: Efficient processing of k-regret minimization queries with theoretical guarantees. Inf. Sci. **586**, 99–118 (2022)
42. Zheng, J., Wang, Y., Wang, X., Ma, W.: Continuous k-regret minimization queries: a dynamic coreset approach. TKDE (2022). https://doi.org/10.1109/TKDE.2022.3166835