$r\alpha$Hit: Achieving α-Approximation with Highest Probability under Uncertain Scoring Function

Xingxing Xiao
Harbin Institute of Technology
Harbin, China
Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
xiaoxx@hit.edu.cn

Jianzhong Li
Harbin Institute of Technology
Harbin, China
Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
lijzh@hit.edu.cn

Zemin Chao
Harbin Institute of Technology
Harbin, China
chaozm@hit.edu.cn

ABSTRACT

For multi-criteria decision-making, the user preference is usually modeled by a scoring function. However, it is difficult for a user to specify the scoring function accurately, and a single function cannot represent the preferences of multiple users. Instead, the probability distribution of scoring function, called the preference distribution, can be used and easily obtained. Inspired by this, we propose the $r\alpha$Hit query ($\alpha < 1$). Using a given preference distribution, it returns a set of r tuples that maximizes the probability of containing a high-scoring tuple. We say a tuple is high-scoring if its score is at least an α-approximation of the top score. In 2D space, we introduce an exact algorithm 2DRAH for $r\alpha$Hit. We prove that $r\alpha$Hit is NP-hard for $d \geq 3$. In MD space, we first show a sampling-based approximation algorithm MDRAH with a guaranteed output quality for any dimension. Further, we improve it and propose an algorithm based on preference space partitioning, MDRAH$^+$, with a smaller time complexity. Extensive experiments demonstrate the performance of our proposed algorithms.

PVLDB Reference Format:
Xingxing Xiao, Jianzhong Li, and Zemin Chao. $r\alpha$Hit: Achieving α-Approximation with Highest Probability under Uncertain Scoring Function. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

For multi-criteria decision-making, the preference of a specific user over tuples in a huge database is commonly expressed by a scoring function. The function assigns a score to each tuple, and a high score means that the tuple is favored by the user. Most frequently, the function is in the form of a weighted sum of tuple attributes, i.e., $\sum w_i A_i$. For example, find the top NBA players based on a weighted sum of performance criteria such as points and assists, or the best cars based on horsepower and miles per gallon.

For an individual user, it is not reasonable to expect her/him to precisely determine the scoring function, even though s/he may roughly know what to look for. For a group of users, due to the wide diversity of preferences, it is not feasible to use a single function to represent the preferences of all users. In both cases, the probability distribution of scoring function, i.e., the *preference distribution*, can be used instead [23, 36]. It describes the fuzzy scoring function of a single user or the overall picture of preferences of many users. Finding the preference distribution is a typical learning problem and has been widely studied in the fields of data mining and machine learning [4, 9, 11, 15, 25–27]. For example, Bayesian learning models [11, 15] discuss how to use the Bayes' rule to build a statistical model to recover the distribution from various user information such as online ratings, reviews and search logs.

Combined with the preference distribution, Peng et al. [23] proposed the rHit query. rHit aims to find an r-size subset S of a given dataset D to maximize the probability of S containing the tuple with largest score. They assumed that a user is only interested in her/his absolute top choice, and S is the set of r tuples interesting to majority of users. However, even for a moderate size r, such S may not exist. It is too stringent a requirement to contain the top-scoring tuple. In experiments, rHit's solution just covers a small range of users, and many more are not. The situation is worse for anti-correlated data or a large number of attributes. From the perspective of many uncovered users, rHit is less convincing and lacks competitiveness. Furthermore, in general, users do not know the exact top-scoring tuple, and are still attracted to a tuple with a sufficiently high score.

In this paper, consider a natural relaxation of rHit from the top score, and introduce a more general definition, the $r\alpha$Hit query, which has not been studied before. Given a threshold $\alpha < 1$, we say a tuple is *high-scoring*, if its score is at least α times the top score of tuples in D. $r\alpha$Hit returns a size r set S to maximize the probability of containing a high-scoring tuple. Note that the value of α is usually very close to 1, such as 0.99 or 0.999. For instance, an online shop could perform an $r\alpha$Hit query on a database of all available products to select a set of r candidates to place on its homepage, from which as many customers as possible can find at least one good choice (imperceptibly close to the top choice). Compared with rHit, $r\alpha$Hit allows returning a representative set covering much more users with $\alpha < 1$, which has been verified

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

in experiments. It pursues not the excellent performance on a few individuals, but the good performance on the majority of users. In $r\alpha$Hit, users can flexibly control the satisfaction criterion by adjusting α. It is a common setting in [2, 22, 37, 38] to relax from score and assume that the tuples with close-to-top scores are good enough. Specifically, it is said that a user is $x\%$ *happy* with a high-scoring tuple (whose score is at least $x\%$ of the top score, by setting $x = 100 \times \alpha$). Accordingly, $r\alpha$Hit returns a small set S that makes the most users happy at a given level.

The recently proposed rkHit query [36] relaxes rHit from the perspective of rank. For rkHit, the size r set S is chosen to maximize the probability of containing at least one of the k largest scoring tuples, i.e., the tuples with ranks no larger than k. rkHit measures the attractiveness of a tuple t through the rank, which is based on comparing the score of t to the scores of all other tuples. Except for the top score, the comparisons with other scores are redundant. The score is a direct measure of tuple quality, and the relaxation should be from the best quality/score. Therefore, $r\alpha$Hit directly considers the score ratio between t and the best tuple. In $r\alpha$Hit, a high-scoring tuple has a score ratio of at least α, and is an α-approximation of the top-scoring tuple. In contrast, for rkHit, a rank of k (even if $k = 2$) can have a ratio much smaller than α.

In addition, rkHit suffers from unique shortcomings absent in $r\alpha$Hit. First, rkHit is not *stable*: adding/removing unimportant tuples (i.e., not optimal for any scoring function) can alter its solution. The instability makes rkHit susceptible to manipulation, allowing crafty sellers to influence its output. Sellers strategically add unimportant products to the database in an attempt to showcase their main products. Second, rkHit's vulnerability to manipulation extends further: sellers adopt a simpler strategy by introducing a single product to the market as k different products, just with k names. Then any product ranked behind them must be out of the first k. In this way, rkHit favors the sellers' products. Conversely, $r\alpha$Hit exhibits stability and is oblivious to such manipulations.

There is a lot of previous work [1, 17, 18, 22, 29, 32, 35, 39] on finding r tuples to best represent the full dataset. Lin et al. [17] chose r tuples to dominate the maximum number of tuples. [29] selected tuples to collectively cover the largest area of the data space. Tao et al. [32] considered r-center clustering of tuples to minimize the maximum distance of a tuple in the dataset to the selected subset. [18] defined a measure that combines two quantities called significance and diversity, where the relative weight of each must be specified in advance. [1, 22, 35, 39] selected tuples to approximate the 1-st or k-th ranked tuple in dataset for any scoring function. None of the above considered preference distributions. Compared to them, the definition of $r\alpha$Hit is more meaningful, when the distribution is known. It ignores impossible scoring functions and finds a representative set to fulfill the most users' needs.

In this paper, we present several theoretical and practical results for $r\alpha$Hit in 2D and MD (multi-dimensional) spaces. We define the high-scoring space of a tuple t, that is, the set of weight vectors under which t is high-scoring. It is a $(d-1)$-dimensional convex polyhedron, and is a line segment when $d = 2$, leading to the design of algorithm 2DRAH. Combined with dynamic programming, 2DRAH returns the optimal solution for $r\alpha$Hit. In the case of $d \geq 3$, $r\alpha$Hit is NP-hard and the exact calculation of hit probability involves a complex multiple integral. Thus we estimate it by sampling, and then the problem is converted into an instance of the maximum coverage problem [14]. This results in algorithm MDRAH, which, through a greedy strategy, returns a solution close to an $(1 - 1/e)$-approximation. Afterwards, we improve MDRAH and propose algorithm MDRAH$^+$ with smaller time complexity. It partitions the preference space into a manageable number of cells to approximately describe the high-scoring spaces. As result, fewer samples are required for estimation, and the problem is changed into finding the weighted maximum coverage [14] on a set system of fixed size. By an exact solver for it, MDRAH$^+$ has no relative error of $1/e$. A practical adaptation of MDRAH$^+$ is considered by replacing the exact solver with a greedy approximate solver. Further, we consider accelerating MDRAH and MDRAH$^+$ through preprocessing and indexing (relying entirely on dataset D).

The main contributions of this paper are listed below.

- We propose the $r\alpha$Hit query, which aims to find a size r representative set that attracts the most user attention.
- We claim that $r\alpha$Hit is NP-hard for $d \geq 3$. We show that $r\alpha$Hit is stable, and removing the tuples outside the skyline [5] has no effect on $r\alpha$Hit.
- In 2D space, we provide an $O(n \log s + rs)$ time exact algorithm 2DRAH for $r\alpha$Hit, where s is the skyline size.
- In MD space, we first design an $O(n \log^{d-2} s + rs \log s)$ time sampling-based algorithm MDRAH for $r\alpha$Hit to approach an $(1 - 1/e)$-approximation.
- We improve MDRAH and propose an $O(n \log^{d-2} s)$ time approximate algorithm based on preference space partitioning, MDRAH$^+$.
- We extend $r\alpha$Hit to two new definitions, αHit Permutation and Continuous $r\alpha$Hit, suitable for special scenarios. Our algorithms remain applicable to them.
- Comprehensive experiments on real and synthetic data validate the efficiency and effectiveness of our algorithms.

2 DEFINITION & PROPERTIES

2.1 Problem Definition

Let D be a dataset of n tuples with d numeric attributes. Assume that the dimensionality d is a fixed constant in this paper. Given a tuple $t \in D$, $t[i]$ denotes its value on i-th attribute. W.l.o.g., for each attribute, a larger value is preferred and the range is normalized to $[0, 1]$. Given an integer $i > 0$, let $[i]$ denote the set $\{1, 2, \cdots, i\}$. For each $i \in [d]$, there is a tuple t in D with $t[i] = 1$. In this paper, the terms "tuple" and "point" are used interchangeably.

Assume that the user preference is measured by an unknown scoring function that maps each tuple in D to a non-negative score. The higher the score, the more desirable the tuple is. Following [3, 10, 21–23, 31, 36, 38], focus on the widely used linear scoring functions of the form $f_w(t) = w \cdot t = \sum_{i=1}^{d} w[i]t[i]$, where $w = \langle w[1], w[2], \cdots, w[d] \rangle$ is a d-dimensional weight vector and $w[i]$ quantifies the significance of i-th attribute. W.l.o.g., assume that $\forall i \in [d]$, $w[i] \geq 0$ and $\sum_{j=1}^{d} w[j] = 1$. The class of linear scoring functions corresponds to the standard $(d-1)$-simplex

$$\Delta^{d-1} = \{w \in \mathbb{R}_+^d \mid \sum\nolimits_{i=1}^{d} w[i] = 1\}.$$

For $d = 2$, Δ^1 is a line segment. For $d = 3$, Δ^2 is an equilateral triangle, and so on. In the following, refer $f_w(\cdot)$ by the weight vector w and use them interchangeably.

A tuple t is preferred over another tuple t', if t has a higher score than t', i.e., $f_w(t) > f_w(t')$. Let $f_w(D)$ be the greatest score of tuples in D. The user is interested in tuples with scores close to $f_w(D)$. Given a user-specified threshold $\alpha \in [0, 1]$, a *high-scoring* tuple t is one whose score is at least an α-approximation of the top score, i.e., $f_w(t) \geq \alpha f_w(D)$. The value of α is close to 1, such as 0.99 and 0.999. Let $\mathcal{D}_\alpha(w)$ be the set of high-scoring tuples, i.e.,

$$\mathcal{D}_\alpha(w) = \{t \in D \mid f_w(t) \geq \alpha f_w(D)\}.$$

Assume that there is a probability distribution Θ on weight vectors, called the preference distribution. Let $\eta(\cdot)$ be the corresponding probability density function with $\int_{\Delta^{d-1}} \eta(w)dw = 1$. For a single user, $\eta(w)$ denotes the probability that $f_w(\cdot)$ is adopted by the user. For a user group, $\eta(w)$ is the proportion of users utilizing $f_w(\cdot)$. In what follows, $\eta(w)$ defaults to the former meaning. Obtaining $\eta(\cdot)$ is a typical learning problem extensively studied in the data mining and machine learning fields [4, 9, 11, 15, 25–27]. Consider the scenario where the preference distribution Θ is given and continuous. This paper can be easily extended to the discrete case.

We say a set $S \subseteq D$ *hits* a weight vector w, if S contains a high-scoring tuple w.r.t. w, i.e., $S \cap \mathcal{D}_\alpha(w) \neq \emptyset$. It is equivalent to the top score of S being at least α times that of D, i.e., $f_w(S) \geq \alpha f_w(D)$. Given the preference distribution Θ, define the α-hit probability.

Definition 1. *Given a set $S \subseteq D$, the α-hit probability of S, denoted by $\mathcal{H}_\alpha(S)$, is the probability of S hitting the weight vector of the user, i.e.,*

$$\mathcal{H}_\alpha(S) = \int_{w \in \Delta^{d-1}} \mathbb{I}(S \cap \mathcal{D}_\alpha(w) \neq \emptyset) \times \eta(w)dw,$$

where $\mathbb{I}(\cdot)$ is the indicator function.

In this paper, we propose the (r, α)-hit ($r\alpha$Hit) problem.

Problem 1 ((r, α)-Hit Problem). *Given a size bound $r \geq 1$ and a threshold $\alpha \in [0, 1]$, compute an r-sized subset of D that maximizes the α-hit probability, i,e., return a set*

$$S^* = \arg\max_{S \subseteq D : |S| \leq r} \mathcal{H}_\alpha(S).$$

Peng et al. [23] proposed the rHit problem to maximize the probability of containing the top-scoring tuple. In this paper, $r\alpha$Hit relaxes the requirement by setting $\alpha < 1$ and considers the α-approximation of largest score. The rkHit problem [36] generalizes rHit from another perspective and maximizes the probability of including one of k largest scoring tuples with $k > 1$. The weak points of rHit and rkHit relative to $r\alpha$Hit have been introduced in the introduction. The algorithms in [23, 36] do not work for $r\alpha$Hit.

2.2 Hardness

rHit is proved to be NP-hard for $d = O(n)$ with an approximation bound of $(1-1/e)$ [36]. It also holds for $r\alpha$Hit, since rHit is a special case with $\alpha = 1$. In Section 4, we propose a polynomial-time exact 2D algorithm, implying that $r\alpha$Hit is in P for $d = 2$. In the following, we present the hardness result of $r\alpha$Hit for a constant $d \geq 3$.

Theorem 1. *$r\alpha$Hit is NP-hard for $d \geq 3$.*

Proof. Cao et al. [7] proved that it is NP-hard for $d = 3$ to determine if there is a size r subset S of D that hits each w in Δ^2, i.e., $S \cap \mathcal{D}_\alpha(w) \neq \emptyset$, $\forall w \in \Delta^2$. First, show that S hits all w, if and only if $\mathcal{H}_\alpha(S) = 1$ under the uniform distribution on Δ^2. The "only if" case is obviously true.

Prove the "if" case by contradiction. Assume that $\mathcal{H}_\alpha(S) = 1$ but there is a $w^* \in \Delta^2$ such that $S \cap \mathcal{D}_\alpha(w^*) = \emptyset$. Suppose t^* is the top-scoring tuple w.r.t. w^*. Define a subspace P of Δ^2 by the following linear constrains:

- $\forall t \in D \setminus \mathcal{D}_\alpha(w^*), f_w(t) < \alpha f_w(t^*)$;
- $w \in \Delta^2$, i.e., $\sum_{i=1}^3 w[i] = 1$, and $w[i] \geq 0, \forall i \in [3]$.

Note that w^* is in P, and P is a convex polygon[1]. For any w in P and t in $D \setminus \mathcal{D}_\alpha(w^*)$, we have $f_w(t) < \alpha f_w(t^*) \leq \alpha f_w(D)$ and $t \notin \mathcal{D}_\alpha(w)$. $\mathcal{D}_\alpha(w)$ is a subset of $\mathcal{D}_\alpha(w^*)$. Thus S does not hit a w in P, i.e., $\mathcal{D}_\alpha(w) \cap S = \emptyset$, $\forall w \in P$. Under the uniform distribution on Δ^2, the probability of a weight vector lying in P is $\mathcal{A}(P)/\mathcal{A}(\Delta^2)$, where $\mathcal{A}(\cdot)$ denotes the area and $\mathcal{A}(\Delta^2) = \frac{1}{2}$. Then we have $\mathcal{H}_\alpha(S) \leq 1 - 2\mathcal{A}(P)$. Following the Shoelace formula [30], $\mathcal{A}(P) = \frac{1}{2}|\sum_{i=1}^{v}(v_i[1]v_{i+1}[2] - v_{i+1}[1]v_i[2])|$, where $v = O(n)$ is the number of vertices of P, v_i is the i-th vertex and $v_{v+1} = v_1$. If the attribute values of each tuple occupy constant bits, so does a vertex of P. $\mathcal{A}(P)$ is a strictly positive value with bounded bit complexity and $\mathcal{H}_\alpha(S) \leq 1 - 2\mathcal{A}(P) < 1$, introducing a contradiction.

Thus it is NP-hard for $d = 3$ to determine if there is a size r subset S of D with $\mathcal{H}_\alpha(S) = 1$ under the uniform distribution, a special decision version of $r\alpha$Hit. If $d > 3$, set the values of last $(d-3)$ attributes to 0. The results still hold. □

2.3 Candidate Tuples

Definition 2 (Dominance & Skyline). *Given two tuples $t, t' \in D$, t dominates t', if $\forall i \in [d]$, $t[i] \geq t'[i]$ and $\exists j \in [d]$, $t[j] > t'[j]$. The skyline of D, denoted by* Sky(D)*, is the set of tuples in D not dominated by any other tuple.*

Borzsony et al. [5] proposed the skyline, the set of optimal tuples for all possible monotonic scoring functions. The following theorem shows that it is sufficient to search for the solution of $r\alpha$Hit in the power set of Sky(D), i.e., $2^{\text{Sky}(D)}$, and filtering out non-skyline (i.e., "unimportant" in Introduction) tuples has no side effects on $r\alpha$Hit.

Theorem 2. *Given a set $R \subseteq D$, there is a set $S \subseteq$ Sky(D) with $|S| \leq |R|$ and $\mathcal{H}_\alpha(S) \geq \mathcal{H}_\alpha(R)$. Further, for all $S \subseteq$ Sky(D), $\mathcal{H}_\alpha(S)$ remains unchanged after removing non-skyline tuples in D.*

Proof. For each $t \in R$, if $t \notin$ Sky(D), there is a tuple $t' \in$ Sky(D) such that t' dominates t. For any w in Δ^{d-1}, $f_w(t') \geq f_w(t)$. Replace all such t in R with t', and obtain S. By definition, $f_w(S) \geq f_w(R)$, $\forall w \in \Delta^{d-1}$, and $\mathcal{H}_\alpha(S) \geq \mathcal{H}_\alpha(R)$. Further, S is no larger than R.

For all $S \subseteq$ Sky(D),

$$\mathcal{H}_\alpha(S) = \int_{w \in \Delta^{d-1}} \mathbb{I}(f_w(S) \geq \alpha f_w(D))\eta(w)dw.$$

Sky(D) contains the optimal tuple for any w in Δ^{d-1}, and $f_w(D) = f_w(\text{Sky}(D))$. Both $f_w(S)$ and $f_w(D)$ have nothing to do with non-skyline tuples. The theorem holds. □

[1] By definition, P cannot be a point or line segment.

Table 1: Notations

Notation	Description
D	Full dataset;
$t[i]$	Value of tuple t on i-th attribute;
$[i]$	Integer set $\{1, 2, \cdots, i\}$;
$f_w(t)$	Score of tuple t w.r.t. weight vector w;
$f_w(S)$	Top score of tuples in set $S \subseteq D$ w.r.t. w;
Δ^{d-1}	Preference space, i.e., standard $(d-1)$-simplex;
$\eta(\cdot)$	Probability density function of preference distribution Θ;
$\mathcal{D}_\alpha(w)$	Set of high-scoring tuples, i.e., $\{t \in D \mid f_w(t) \geq \alpha f_w(D)\}$;
$\mathcal{H}_\alpha(S)$	α-hit probability of set S;
Sky(D)	Skyline of D after deduplication;

Similarly, adding non-skyline (i.e., "unimportant" in Introduction) tuples has no effect. $r\alpha$Hit is *stable*. The algorithms for $r\alpha$Hit can run directly on top of the skyline operator, and be integrated into systems where the operator is available. Instead, rkHit [36] is not stable and non-skyline tuples have a severe impact on the solution of rkHit.

Computing the skyline requires $O(n \log s)$ time for $d = 2$ and $O(n \log^{d-2} s)$ time for $d \geq 3$ [16], where s is the skyline size. It is established that if the d attributes are statistically independent, s is $O(\ln^{d-1} n/(d-1)!)$ [6]. Chaudhuri et al. [8] found that in presence of anti-correlations not very large, the skyline size still grows as some power of $\log n$. Assume $r < s$. Otherwise, directly output Sky(D) as the solution. Further, if Sky(D) contains duplicate tuples, keep just one of them. In the following, the skyline has been deduplicated.

Summarize important notations in Table 1.

3 TOP & HIGH-SCORING SPACES

In this section, introduce the top-scoring and high-scoring spaces of a tuple, from which we can derive another insight into $r\alpha$Hit and build the foundation for subsequent algorithms.

Formally, the top-scoring space of a tuple $t \in D$ is defined to be $\mathcal{U}(t) = \{w \in \Delta^{d-1} \mid f_w(t) = f_w(D)\}$, i.e., the set of weight vectors for which t has the top score. Relax it to the high-scoring space $\mathcal{U}_\alpha(t)$, the set of vectors for which the score of t is at least an α-approximation of the top score, i.e.,

$$\mathcal{U}_\alpha(t) = \{w \in \Delta^{d-1} \mid f_w(t) \geq \alpha f_w(D)\}.$$

Let $\mathcal{P}(\mathcal{U}_\alpha(t))$ be the probability of a weight vector lying in $\mathcal{U}_\alpha(t)$, i.e., $\mathcal{P}(\mathcal{U}_\alpha(t)) = \int_{w \in \mathcal{U}_\alpha(t)} \eta(w) dw$. Then $r\alpha$Hit aims to find a fixed size set S maximizing

$$\mathcal{H}_\alpha(S) = \mathcal{P}(\cup_{t \in S} \mathcal{U}_\alpha(t)). \tag{1}$$

In the following, consider representing $\mathcal{U}(t)$ and $\mathcal{U}_\alpha(t)$ as the intersection of halfspaces.

For each $i \in [d]$, the i-th boundary point, denoted by b_i, is defined to be a d-dimensional point with $b_i[i] = 1$ and $b_i[j] = 0$ for $j \neq i$. Let B be the set $\{b_i\}_{i \in [d]}$ and $o = (0, \cdots, 0)$ be the origin. In geometry, the convex hull of D, denoted by CH(D), is the smallest convex set containing D. A point t in D is a *vertex* of D's convex hull, if t is not in the convex hull of $D \setminus \{t\}$. In this paper, use CH$^+(D)$ to represent the convex hull of Sky$(D) \cup B \cup \{o\}$, i.e.,

$$\text{CH}^+(D) = \text{CH}(\text{Sky}(D) \cup B \cup \{o\}).$$

For any $w \in \Delta^{d-1}$, there is a vertex of CH$^+(D)$ in Sky(D) realizing the top score $f_w(D)$. Let $\mathcal{N}_t$ be the set of neighboring vertices of a vertex t in CH$^+(D)$. Intuitively, t and $\mathcal{N}_t$ provide directive information to determine the top-scoring space of t.

Lemma 1. *Given a vector w in Δ^{d-1} and vertex t of* CH$^+(D)$*, either $f_w(t)$ is equal to $f_w(D)$, or there is a vertex v in $\mathcal{N}_t$ with $f_w(v) > f_w(t)$.*

A vertex t is top-scoring for w, if t has a score no less than that of its neighbors. Let $h^+(t, v)$ be the halfspace containing the vectors w such that $f_w(t) \geq f_w(v)$, i.e., $h^+(t, v) = \{w \in \mathbb{R}^d \mid w \cdot t \geq w \cdot v\}$. Then $\mathcal{U}(t) = (\cap_{v \in \mathcal{N}_t} h^+(t, v)) \cap \Delta^{d-1}$. Figure 1 shows an example of CH$^+(D)$ for $D = \{(0.1, 1), (0.5, 0.8), (0.6, 0.6), (0.8, 0.5), (1, 0.1)\}$. Only t_3 is not a vertex of CH$^+(D)$. The neighbor set of t_2 is $\mathcal{N}_{t_2} = \{t_1, t_4\}$, and then $\mathcal{U}(t_2) = h^+(t_2, t_1) \cap h^+(t_2, t_4) \cap \Delta^1$.

Let $\hat{t} = t/\alpha$ be the $1/\alpha$*-expanded* point of a tuple $t \in D$.

Lemma 2. *Given a vector w in Δ^{d-1} and tuple t in D, $f_w(t) \geq \alpha f_w(D)$, iff $f_w(\hat{t})$ is equal to $f_w(D \cup \{\hat{t}\})$.*

A tuple $t \in D$ is high-scoring for w, iff $\hat{t}$ is the top-scoring tuple of $D \cup \{\hat{t}\}$. $\mathcal{N}_{\hat{t}}$ denotes the set of neighboring vertices of $\hat{t}$ in CH$^+(D \cup \{\hat{t}\})$. If $\hat{t}$ is not a vertex of CH$^+(D \cup \{\hat{t}\})$, $\mathcal{N}_{\hat{t}}$ is empty. Combining Lemma 1 and 2, get the following.

Corollary 1. *Given a vector w in Δ^{d-1} and tuple t in D, if $\hat{t}$ is a vertex of* CH$^+(D \cup \{\hat{t}\})$*, either $f_w(t) \geq \alpha f_w(D)$, or there is a vertex v in $\mathcal{N}_{\hat{t}}$ with $f_w(v) > f_w(\hat{t})$.*

Represent $\mathcal{U}_\alpha(t)$ as the intersection of multiple halfspaces. Specifically, for a tuple $t \in D$, if $\hat{t} \notin$ CH$^+(D)$, then

$$\mathcal{U}_\alpha(t) = (\cap_{v \in \mathcal{N}_{\hat{t}}} h^+(\hat{t}, v)) \cap \Delta^{d-1}. \tag{2}$$

Otherwise[2], $\mathcal{U}_\alpha(t)$ is empty. In Figure 1, $\hat{t}_3$ is the $1/\alpha$-expanded point of t_3 with $\alpha = 0.95$, and is in CH$^+(D)$. Thus $\mathcal{U}_{0.95}(t_3) = \emptyset$. Figure 2 shows CH$^+(D \cup \{\hat{t}_2\})$ with $\alpha = 0.8$. The neighbor set $\mathcal{N}_{\hat{t}_2}$ is $\{b_2, t_5\}$. Then $\mathcal{U}_{0.8}(t_2) = h^+(\hat{t}_2, b_2) \cap h^+(\hat{t}_2, t_5) \cap \Delta^1$.

Both $\mathcal{U}(t)$ and $\mathcal{U}_\alpha(t)$ are $(d-1)$-dimensional convex polyhedra. For different tuples, their top-scoring spaces must be disjoint, but the high-scoring spaces may intersect.

4 ALGORITHM IN 2D

Start by considering the two-dimensional scenario. In this section, we design a 2D algorithm 2DRAH to return the optimal solution for $r\alpha$Hit. The algorithm works in two steps. 2DRAH first computes the high-scoring space $\mathcal{U}_\alpha(t)$ of each tuple t. Then, through dynamic programming, it finds a tuple set S maximizing $\mathcal{P}(\cup_{t \in S} \mathcal{U}_\alpha(t))$.

4.1 Preliminary

In 2D space, a vector w in Δ^1 is expressed in the form of $\langle \omega, 1 - \omega \rangle$ by setting $\omega = w[1]$. Then Δ^1 is equivalent to an interval, i.e., $\omega \in [0, 1]$. Let $f_\omega(\cdot)$ denote the scoring function $f_{\langle \omega, 1-\omega \rangle}(\cdot)$. The high-scoring space of a tuple $t \in D$ is

$$\mathcal{U}_\alpha(t) = \{\omega \in [0, 1] \mid f_\omega(t) \geq \alpha f_\omega(D)\}.$$

The density function $\eta(\cdot)$ is defined over ω. For example, when Θ is a uniform distribution over Δ^1, $\eta(\omega) = 1$, $\forall \omega \in [0, 1]$. Set $D =$

[2]Assume $\hat{t}$ does not lie on the boundary of CH$^+(D)$. Otherwise, there is a vertex v of CH$^+(D)$ in Sky(D) with $\mathcal{U}_\alpha(t) \subseteq \mathcal{U}(v)$. Removing t does not affect the problem.

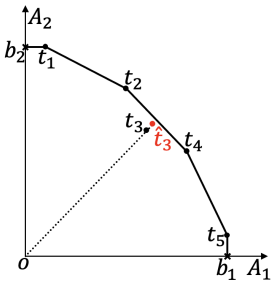

Figure 1: $\mathrm{CH}^+(D)$ **and** $\hat{t}_3$ **with** $\alpha = 0.95$

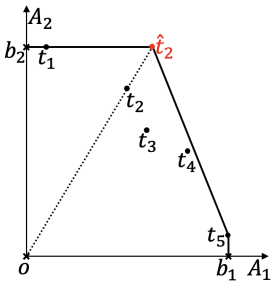

Figure 2: $\mathrm{CH}^+(D \cup \{\hat{t}_2\})$ **with** $\alpha = 0.8$

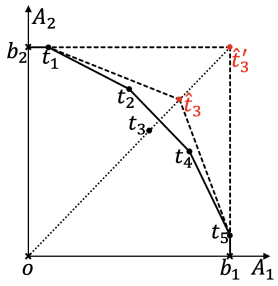

Figure 3: $\mathcal{N}_{\hat{t}_3}$ **with** $\alpha = 0.8$ **(or** 0.6**)**

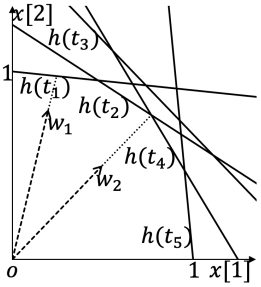

Figure 4: Example for ray shooting queries

$\mathrm{Sky}(D) = \{t_1, t_2, ..., t_s\}$. The tuples are sorted in ascending order by the first attribute, i.e., $t_i[1] < t_{i+1}[1], \forall i \in [s-1]$. The c vertices of $\mathrm{CH}^+(D)$ in D are $t_{\varphi_1}, t_{\varphi_2}, \cdots, t_{\varphi_c}$ with $1 = \varphi_1 < \cdots < \varphi_c = s$.

4.2 Compute High-Scoring Space

Consider computing the high-scoring space $\mathcal{U}_\alpha(t_i)$ of the i-th tuple t_i in D. First determine if the $1/\alpha$-expanded point $\hat{t}_i$ of t_i lies in $\mathrm{CH}^+(D)$. $\hat{t}_i \in \mathrm{CH}^+(D)$ implies that $\mathcal{U}_\alpha(t_i)$ is empty. If t_i is a vertex of $\mathrm{CH}^+(D)$, $\hat{t}_i$ must be outside $\mathrm{CH}^+(D)$. Assume t_i is not a vertex. Let t_{φ_j} be the j-th vertex with $\varphi_j < i < \varphi_{j+1}$. t_{φ_j}-$t_{\varphi_{j+1}}$ denotes the line segment with endpoints t_{φ_j} and $t_{\varphi_{j+1}}$. $\hat{t}_i$ is outside $\mathrm{CH}^+(D)$, if t_{φ_j}-$t_{\varphi_{j+1}}$ intersects the line segment o-$\hat{t}_i$. In Figure 1, when $\alpha = 0.95$, o-$\hat{t}_3$ is disjoint from t_2-t_4 and $\hat{t}_3$ is inside $\mathrm{CH}^+(D)$. In Figure 3, when $\alpha = 0.8$, o-$\hat{t}_3$ intersects t_2-t_4 and $\hat{t}_3$ is outside $\mathrm{CH}^+(D)$.

Consider the case of $\hat{t}_i \notin \mathrm{CH}^+(D)$, i.e., $\hat{t}_i$ is a vertex of $\mathrm{CH}^+(D \cup \{\hat{t}_i\})$. The size of neighboring vertex set $\mathcal{N}_{\hat{t}_i}$ is 2 when $d = 2$. Let $\mathcal{N}_{\hat{t}_i} = \{v_1, v_2\}$. The line v_1-$\hat{t}_i$ (resp., v_2-$\hat{t}_i$) is the supporting line and is tangent to $\mathrm{CH}^+(D)$ at v_1 (resp., v_2). Suppose that v_1, v_2 are located on the upper and lower sides of the line o-$\hat{t}_i$, respectively. Next, introduce how to obtain v_1 (locating v_2 is similar).

- If $\hat{t}_i[1] = 0$, then v_1 is the origin o.
- If $\hat{t}_i[1] > 0$ and $\hat{t}_i[2] \geq 1$, v_1 is the 2-nd boundary point b_2.
- If $\hat{t}_i[1] > 0$ and $\hat{t}_i[2] < 1$, v_1 is the point in $C = \{t_{\varphi_1}, \cdots, t_{\varphi_j}\}$ that maximizes $\Phi(v_1) = (v_1[2] - \hat{t}_i[2])/(\hat{t}_i[1] - v_1[1])$, where $\varphi_j \leq i < \varphi_{j+1}$.

For the third case, locate v_1 in a binary search manner. Specifically, at each iteration, check two consecutive median vertices t_{φ_κ} and $t_{\varphi_{\kappa+1}}$ in C ($\kappa = \lfloor (1+j)/2 \rfloor$ in the 1-st iteration). If $\Phi(t_{\varphi_\kappa}) = \Phi(t_{\varphi_{\kappa+1}})$, then v_1 is found and is t_{φ_κ}. If $\Phi(t_{\varphi_\kappa}) > \Phi(t_{\varphi_{\kappa+1}})$, C is updated to be the first half ($\{t_{\varphi_1}, \cdots, t_{\varphi_\kappa}\}$ in the 1-st iteration). If $\Phi(t_{\varphi_\kappa}) < \Phi(t_{\varphi_{\kappa+1}})$, C is updated to be the remaining half. This process continues until v_1 is found or C contains a single point. In Figure 3, when $\alpha = 0.8$, the $1/\alpha$-expanded point of t_3 is $\hat{t}_3 = (0.75, 0.75)$. The upper neighbor of $\hat{t}_3$ is a point in $C = \{t_1, t_2\}$. Due to $\Phi(t_1) > \Phi(t_2)$, it is t_1. When $\alpha = 0.6$, the $1/\alpha$-expanded point is $\hat{t}_3' = (1, 1)$. Then b_2 and b_1 are the two neighboring vertices of $\hat{t}_3'$ in $\mathrm{CH}^+(D \cup \{\hat{t}_3'\})$.

Afterwards, obtain $\mathcal{U}_\alpha(t_i)$ through v_1 and v_2. Construct $h^+(\hat{t}_i, v_1) = \{\omega \in \mathbb{R} \mid f_\omega(\hat{t}_i) \geq f_\omega(v_1)\}$ and $h^+(\hat{t}_i, v_2) = \{\omega \in \mathbb{R} \mid f_\omega(\hat{t}_i) \geq f_\omega(v_2)\}$. Then $\mathcal{U}_\alpha(t_i)$ is $h^+(\hat{t}_i, v_1) \cap h^+(\hat{t}_i, v_2) \cap [0, 1]$.

4.3 Dynamic Programming

For a tuple $t \in D$, $\mathcal{U}_\alpha(t)$ is empty or a sub-interval of $[0, 1]$. Just discard the empty ones. Sort non-empty $\mathcal{U}_\alpha(t)$ in non-decreasing order of their start locations, producing a list of sub-intervals, $\mathcal{I} = \{I_1, I_2, \cdots, I_{|\mathcal{I}|}\}$ with $|\mathcal{I}| \leq s$. Let $I_i = [p_i, q_i]$. Assume there are no two sub-intervals I_i and I_{i+1} with $p_i = p_{i+1}$. Otherwise, $I_i \subseteq I_{i+1}$ or $I_{i+1} \subseteq I_i$, and we remove one of them. Thus, $p_1 < p_2 < \cdots < p_{|\mathcal{I}|}$. Similarly, assume there are no I_i and I_{i+1} such that $q_i \geq q_{i+1}$. Then $q_1 < q_2 < \cdots < q_{|\mathcal{I}|}$. $\mathcal{P}(I_i)$ denotes the probability of ω lying in I_i, i.e., $\mathcal{P}(I_i) = \int_{\omega \in I_i} \eta(\omega) d\omega$. Consider finding a set T of r sub-intervals in $\mathcal{I}$ to maximize $\mathcal{P}(\cup_{I_i \in T} I_i)$. The r corresponding tuples constitute the solution to $r\alpha$Hit.

Let $\mathcal{I}_i = \{I_1, \cdots, I_i\}$ be the set of first i sub-intervals. $DP(i, j).set$ is the set of j sub-intervals in $\mathcal{I}_i$ maximizing the probability of ω lying in one of them, and $DP(i, j)$ denotes the corresponding probability. $DP(|\mathcal{I}|, r).set$ is the solution. The recursive formula for dynamic programming is as follows:

- $DP(i, 0) = 0$;
- If $i \leq j$, $DP(i, j) = \mathcal{P}([0, q_i])$;
- If $1 \leq j < i$, $DP(i, j) = \max(DP^*(i, j), DP(i-1, j))$.

The first case is due to $DP(i, 0).set = \emptyset$. In the second case, $DP(i, j).set = \mathcal{I}_i$ and $DP(i, j) = \mathcal{P}(\cup_{j \in [i]} I_j) = \mathcal{P}([0, q_i])$. The third case has two sub-cases. Let $DP^*(i, j).set$ be a set of j sub-intervals (one of which is I_i) in $\mathcal{I}_i$, to maximize the probability of ω included by one of them. $DP^*(i, j)$ denotes $\mathcal{P}(\cup_{I_i \in DP^*(i,j).set} I_i)$. If $I_i \in DP(i, j).set$, $DP(i, j) = DP^*(i, j)$. Otherwise, $DP(i, j) = DP(i-1, j)$.

Let ϕ_i be the smallest index of sub-intervals in $\mathcal{I}_{i-1}$ that intersect with I_i, i.e., $\phi_i = \min_{j \in [i-1]: q_j \geq p_i} j$. Each I_j in $\mathcal{I}_{\phi_i - 1}$ is disjoint from I_i. Show the recursive formula of $DP^*(i, j)$:

- $DP^*(i, 1) = \mathcal{P}([p_i, q_i])$;
- If $i \leq j$, $DP^*(i, j) = \mathcal{P}([0, q_i])$;
- If $1 < j < i$,

$$DP^*(i, j) = \max \begin{Bmatrix} \mathcal{P}([q_{\phi_i}, q_i]) + DP^*(\phi_i, j-1), \\ \mathcal{P}([p_i, q_i]) + DP(\phi_i - 1, j-1) \end{Bmatrix}.$$

The first two cases are similar to $DP(i, j)$. The third has two sub-cases. One is that $DP^*(i, j).set$ contains a sub-interval I_j ($j \in [i-1]$) intersecting I_i. Since $(I_j \cup I_i) \subset (I_{\phi_i} \cup I_i)$, replace I_j with I_{ϕ_i}. Then $DP^*(i, j).set = \{I_i\} \cup DP^*(\phi_i, j-1).set$ and $DP^*(i, j) = \mathcal{P}(I_i \setminus I_{\phi_i}) + DP^*(\phi_i, j-1)$. Another sub-case is that I_i is disjoint from the other

Algorithm 1: 2DRAH

Input: D, $\eta(\cdot)$, r, α;
Output: a size r set S that maximizes $\mathcal{H}_\alpha(S)$;

Calculate Sky(D), $\text{CH}^+(D)$ and set $D = \text{Sky}(D)$;
foreach $t \in D$ **do** Compute $\mathcal{U}_\alpha(t)$;
Sort non-empty $\mathcal{U}_\alpha(t)$ and obtain $\mathcal{I}$;
foreach $i \in [|\mathcal{I}|]$ **do**
  $DP(i,0).set = \emptyset$, $DP(i,0) = 0$;
  $DP^*(i,1).set = \{I_i\}$, $DP^*(i,1) = \mathcal{P}([p_i, q_i])$;
foreach $j \in [r]$ and $i \in [j]$ **do**
  $DP(i,j).set = DP^*(i,j).set = \mathcal{I}_i$;
  $DP(i,j) = DP^*(i,j) = \mathcal{P}([0, q_i])$;
foreach $j \in [r]$ and $i \in [|\mathcal{I}|] \setminus [j]$ **do**
  if $j > 1$ **then**
    if $\mathcal{P}([q_{\phi_i}, q_i]) + DP^*(\phi_i, j-1) \geq \mathcal{P}([p_i, q_i]) + DP(\phi_i - 1, j-1)$ **then**
      $DP^*(i,j).set = \{I_i\} \cup DP^*(\phi_i, j-1).set$;
      $DP^*(i,j) = \mathcal{P}([q_{\phi_i}, q_i]) + DP^*(\phi_i, j-1)$;
    else
      $DP^*(i,j).set = \{I_i\} \cup DP(\phi_i - 1, j-1).set$;
      $DP^*(i,j) = \mathcal{P}([p_i, q_i]) + DP(\phi_i - 1, j-1)$;
  if $DP^*(i,j) \geq DP(i-1,j)$ **then**
    $DP(i,j).set = DP^*(i,j).set$, $DP(i,j) = DP^*(i,j)$;
  else $DP(i,j).set = DP(i-1,j).set$, $DP(i,j) = DP(i-1,j)$;
return set of r tuples corresponding to $DP(|\mathcal{I}|, r).set$;

$(j-1)$ sub-intervals in $DP^*(i,j).set$. Then they must be in $\mathcal{I}_{\phi_i - 1}$, and $DP^*(i,j).set = \{I_i\} \cup DP(\phi_i - 1, j-1).set$.

The pseudocode of 2DRAH is shown in Algorithm 1.

Simplified Storage. Note that $DP(i,j).set$ is obtained by comparing $DP^*(i,j).set$ and $DP(i-1,j).set$. $DP^*(i,j).set$ is derived in a similar way. It is unnecessary to spend $O(rs \times r)$ space and time to store them. Instead, replace each $DP(i,j).set$ (resp., $DP^*(i,j).set$) with a Boolean value to record whether it contains I_i (resp., I_{ϕ_i}). The only side effect is that the algorithm has to backtrack $O(|\mathcal{I}| + r)$ intermediate states to obtain the solution $DP(|\mathcal{I}|, r).set$.

4.4 Theoretical Results & Discussion

Theorem 3. *In 2D space,* 2DRAH *returns an optimal solution for the* $r\alpha$Hit *problem.*

Proof. Let $\Gamma(\mathcal{I}_i, j)$ be the problem of finding a set T of j sub-intervals in $\mathcal{I}_i$ to maximize $\mathcal{P}(\cup_{I_i \in T} I_i)$. Further, the problem $\Gamma^*(\mathcal{I}_i, j)$ aims to obtain a set T of j sub-intervals in $\mathcal{I}_i$ that contains I_i and maximizes $\mathcal{P}(\cup_{I_i \in T} I_i)$. The optimal solution to $\Gamma(\mathcal{I}, r)$ corresponds to that to $r\alpha$Hit. In the following, prove the correctness of dynamic programming. Both $\Gamma(\mathcal{I}_i, j)$ and $\Gamma^*(\mathcal{I}_i, j)$ have optimal substructure. Referring to the recursive formula of $DP(i,j)$, an optimal solution of $\Gamma(\mathcal{I}_i, j)$ can be constructed from that of $\Gamma(\mathcal{I}_{i-1}, j)$ and $\Gamma^*(\mathcal{I}_i, j)$. Similarly, an optimal solution of $\Gamma^*(\mathcal{I}_i, j)$ can be derived from that of $\Gamma^*(\mathcal{I}_{\phi_i}, j-1)$ and $\Gamma(\mathcal{I}_{\phi_i - 1}, j-1)$. Both $\Gamma(\mathcal{I}_i, j)$ and $\Gamma^*(\mathcal{I}_i, j)$ have overlapping subproblems, i.e., the problems can be broken down into subproblems which are reused several times. □

Theorem 4. 2DRAH *takes* $O(n \log s + rs)$ *time.*

Proof. It takes $O(n \log s)$ time to compute Sky(D) and $\text{CH}^+(D)$ in 2D space. Obtaining $\mathcal{U}_\alpha(t)$ for all $t \in \text{Sky}(D)$ requires $O(s \log c)$ time. Sorting them consumes $O(s \log s)$ time. All ϕ_i can be calculated by scanning $\mathcal{I}$ once, which takes $O(|\mathcal{I}|) = O(s)$ time. The dynamic programming contains $O(rs)$ intermediate states and each state runs in $O(1)$ time. 2DRAH totally runs in $O(n \log s + rs)$ time. □

In this paper, we assume that the calculation of $\mathcal{P}([p,q])$ for a sub-interval $[p,q]$ of $[0,1]$ takes constant time. It computes an integral $\int_{\omega \in [p,q]} \eta(\omega) d\omega$, i.e., $F(q) - F(p)$, where $F(\cdot)$ is the cumulative distribution function. The integral calculation relies on the specific distribution Θ.

5 ALGORITHMS IN MD

In the case of $d \geq 3$, $r\alpha$Hit is proved to be NP-hard. In this section, design approximation MD algorithms for $r\alpha$Hit.

As shown in Equation (1), computing the hit probability is equivalent to calculating a complex multiple integral over the union of multiple $(d-1)$-dimensional convex polyhedra. Therefore, we present a sampling-based algorithm, MDRAH. It estimates the hit probability through sampling, and returns a solution that is close to an $(1 - 1/e)$-approximation with arbitrary absolute error. Next, improve MDRAH and propose an algorithm combined with preference space partitioning, MDRAH^+. It requires fewer samples and thus has smaller time complexity than MDRAH. MDRAH^+ has no relative error of $1/e$ in the approximation guarantee.

According to Theorem 2, firstly set $D = \text{Sky}(D)$.

5.1 MDRAH: Sampling-Based Greedy Algorithm

In this section, we describe our sampling-based approximation algorithm, called MDRAH, for $r\alpha$Hit. It does not require complex integral operations, and instead estimates the hit probabilities of all candidate solutions through sampling. The estimate introduces a bounded absolute error. After sampling, the problem is formulated as an instance of the maximum coverage problem [14]. Through the naive greedy strategy, MDRAH obtains a theoretically guaranteed solution. Moreover, we accelerate MDRAH through preprocessing and indexing based entirely on the dataset D.

Sampling for Estimation. As shown in Section 3, given a set $S \subseteq D$, the α-hit probability of S is equal to the probability of a weight vector lying in $\cup_{t \in S} \mathcal{U}_\alpha(t)$, i.e., $\mathcal{H}_\alpha(S) = \mathcal{P}(\cup_{t \in S} \mathcal{U}_\alpha(t))$. Let $W = \{w_1, w_2, \cdots, w_m\}$ be a set of m weight vectors sampled from Δ^{d-1} w.r.t. the preference distribution Θ. Then the value

$$\mathcal{H}_\alpha(S, W) = \frac{1}{m} \times \left| \{ w \in W \mid w \in \cup_{t \in S} \mathcal{U}_\alpha(t) \} \right|$$

is an unbiased estimate of $\mathcal{H}_\alpha(S)$.

For each $i \in [m]$, let X_i be a random variable where $X_i = \mathbb{I}(w_i \in \cup_{t \in S} \mathcal{U}_\alpha(t))$ and $\mathbb{I}(\cdot)$ is the indicator function. Then $\mathcal{H}_\alpha(S, W) = \frac{1}{m} \sum_{i=1}^{m} X_i$. Further, the expected value of X_i is $\mathcal{P}(\cup_{t \in S} \mathcal{U}_\alpha(t))$. According to the well-known Chernoff-Hoeffding Inequality [24],

Algorithm 2: MDRAH

Input: D, Θ, r, α;
Output: a size r set $S \subseteq D$;
1 Calculate Sky(D) and set D = Sky(D);
2 Construct set W of m sampled weight vectors;
3 **foreach** $w \in W$ **do**
4 Traverse D to calculate $f_w(D) = \max_{t \in D} f_w(D)$;
5 Scan D to compute $\mathcal{D}_\alpha(w) = \{t \in D \mid f_w(t) \geq \alpha f_w(D)\}$;
6 Traverse $\{\mathcal{D}_\alpha(w)\}_{w \in W}$ to get $\{W_\alpha(t)\}_{t \in D}$;
7 **return** Greedy-MC(W, $\{W_\alpha(t)\}_{t \in D}$, r);
8 **def** Greedy-MC(W, $\{W_\alpha(t)\}_{t \in D}$, r):
9 $S = \emptyset$;
10 **while** $|S| < r$ **do**
11 Find t with largest $\left|W_\alpha(t) \setminus (\cup_{t \in S} W_\alpha(t))\right|$ and $S = S \cup \{t\}$;
12 **return** S;

with a probability at most $2e^{-2m\varepsilon^2}$, $|\mathcal{H}_\alpha(S, W) - \mathcal{H}_\alpha(S)| > \varepsilon$. The above is only true for the given set S. For the $r\alpha$Hit problem, both the approximate and optimal solutions are of size r.[3] Therefore, it is sufficient to ensure that for all r-size subsets of D, the estimated hit probability is close to the exact. Accordingly, the required sample size m is determined by the following lemma.

Lemma 3. *Given a sample size* $m = \frac{\ln 2 + r \ln s + \ln \frac{1}{\delta}}{2\varepsilon^2}$, *with probability at least* $1 - \delta$, $\forall S \subseteq D$ *with* $|S| = r$,

$$\left|\mathcal{H}_\alpha(S, W) - \mathcal{H}_\alpha(S)\right| \leq \varepsilon.$$

Proof Sketch. For a given S, with a probability at most δ / s^r, the absolute error overflows. Then for s^r sets, the probability of an overflow occurring is at most δ. □

Maximum Coverage. After sampling, $r\alpha$Hit is turned into an instance of the maximum coverage (MC) problem [14]. The corresponding set system[4] is $\Sigma_1 = (W, \{W_\alpha(t)\}_{t \in D})$, where $W_\alpha(t)$ is the set of sampled vectors lying in $\mathcal{U}_\alpha(t)$, i.e.,

$$W_\alpha(t) = \{w \in W \mid w \in \mathcal{U}_\alpha(t)\}.$$

The goal is to find a size r sub-collection of $\{W_\alpha(t)\}_{t \in D}$ that covers the maximum number of vectors in W. It corresponds to an r-sized set S maximizing $\mathcal{H}_\alpha(S, W) = \frac{1}{m} |\cup_{t \in S} W_\alpha(t)|$.

Consider computing the collection $\{W_\alpha(t)\}_{t \in D}$. $W_\alpha(t)$ is the set $\{w \in W \mid t \in \mathcal{D}_\alpha(w)\}$. Instead of calculating $\{W_\alpha(t)\}_{t \in D}$ directly, it is better to compute $\{\mathcal{D}_\alpha(w)\}_{w \in W}$. For a sampled vector w, first traverse D to obtain the top score $f_w(D) = \max_{t \in D} f_w(t)$. Then, scan D for the 2-nd time to obtain the high-scoring tuples, i.e., all t in D with $f_w(t) \geq \alpha f_w(D)$. Eventually, all $W_\alpha(t)$ are obtained by a simple traversal of $\{\mathcal{D}_\alpha(w)\}_{w \in W}$. The above process just takes $O(sm)$ time.

[3] If the size is less than r, tuples are added until the size equals r.
[4] A set system consists of two parts, a ground set $E = \{e_1, \cdots, e_m\}$ and a set collection $\mathcal{T} = \{T_1, \cdots, T_n\} \subseteq 2^E$. E is the set of all elements that need to be covered. $\mathcal{T}$ contains all sets available for coverage.

Knowing $\{W_\alpha(t)\}_{t \in D}$, MDRAH adopts a greedy strategy to obtain an approximate maximum coverage, i.e., the subroutine Greedy-MC in Algorithm 2. Initially, set $S = \emptyset$. Then it iteratively chooses the set $W_\alpha(t)$ that covers the most uncovered vectors in W, and adds the corresponding tuple to S until $|S| = r$. Regarding $\mathcal{H}_\alpha(S, W)$, the greedy strategy brings an $(1 - 1/e)$-approximation [13].

Theoretical Results.

Theorem 5. *With probability at least* $1 - \delta$, MDRAH *outputs a tuple set* S *satisfying*

$$\mathcal{H}_\alpha(S) \geq (1 - 1/e)\mathcal{H}_\alpha(S^*) - (2 - 1/e)\varepsilon, \tag{3}$$

where S^* *is the optimal solution for* $r\alpha$Hit.

Proof Sketch. $\mathcal{H}_\alpha(S, W)$ is an $(1 - 1/e)$-approximation, i.e., $\mathcal{H}_\alpha(S, W) \geq (1 - 1/e)\mathcal{H}_\alpha(S^*, W)$. Based on Lemma 3, $\mathcal{H}_\alpha(S) \geq \mathcal{H}_\alpha(S, W) - \varepsilon$ and $\mathcal{H}_\alpha(S^*, W) \geq \mathcal{H}_\alpha(S^*) - \varepsilon$. □

Estimation by sampling introduces an absolute error. But it is adjustable. By adjusting ε, MDRAH infinitely approaches the approximation bound of $(1 - 1/e)$ in Section 2.2. Even with $\alpha = 0.99$, the worst-case $\mathcal{H}_\alpha(S)$ is greater than 0.3 in experiments. ε does not need to be very small.

Theorem 6. MDRAH *takes* $O(n \log^{d-2} s + rs \log s)$ *time.*

Proof. It takes $O(n \log^{d-2} s)$ time to calculate Sky(D) for $d \geq 3$. It requires $O(sm)$ time to compute $\{\mathcal{D}_\alpha(w)\}_{w \in W}$. The subroutine Greedy-MC (line 8-12) can be done in $O(sm)$ time [28]. Note that $m = O(\frac{r \ln s + \ln \frac{1}{\delta}}{\varepsilon^2})$. Then the time complexity of MDRAH is $O(n \log^{d-2} s + \frac{rs \ln s + s \ln \frac{1}{\delta}}{\varepsilon^2})$. Both $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$ in Lemma 3 are constants in practice. □

Speeding up with Preprocessing & Indexing. Consider accelerating MDRAH based on preprocessing and indexing on the dataset D, without any knowledge of the preference distribution Θ, threshold α and size bound r. First, precalculate Sky(D) and set D = Sky(D). The remaining most time-consuming step is to calculate $\{\mathcal{D}_\alpha(w)\}_{w \in W}$. Therefore, build an index to speed up it. As a result, the algorithm no longer traverses the complete D. The index has two parts.

The first part is used to obtain the top score $f_w(D)$. It makes use of the concept of *duality* and maps each tuple t in D to a hyperplane $h(t) = \{w \in \mathbb{R}^d \mid w \cdot t = 1\}$. Given a weight vector $w \in \Delta^{d-1}$, issue a ray from the origin o along the direction of w and determine the first hyperplane $h(t)$ hit by the ray. The above process is completed through a *ray shooting query* [20]. The corresponding tuple t has the top score, i.e., $f_w(t) = f_w(D)$. Figure 4 shows the dual of D = $\{(0.1, 1), (0.5, 0.8), (0.6, 0.6), (0.8, 0.5), (1, 0.1)\}$. The ray from o along the direction of $w_1 = \langle 0.2, 0.8 \rangle$ hits the line $h(t_1)$ first, and t_1 is top-scoring for w_1. The ray along $w_2 = \langle 0.5, 0.5 \rangle$ first hits $h(t_2)$ and $h(t_4)$. Both t_2 and t_4 realize the top score $f_{w_2}(D)$. With a data structure built in $O(s^{\lfloor d/2 \rfloor} \log^{O(1)} s)$ time, a ray shooting query can be answered in $O(\log s)$ time [20].

The second part is to calculate $\mathcal{D}_\alpha(w)$, given $f_w(D)$. $\mathcal{D}_\alpha(w)$ is the set of tuples in D lying in the halfspace $\{x \in \mathbb{R}^d \mid x \cdot w \geq \alpha f_w(D)\}$, and is returned by a *halfspace range query* [19]. With

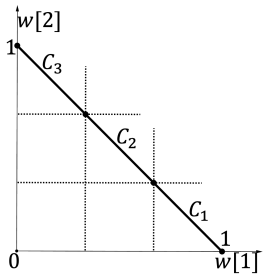

Figure 5: Partitioning of Δ^1

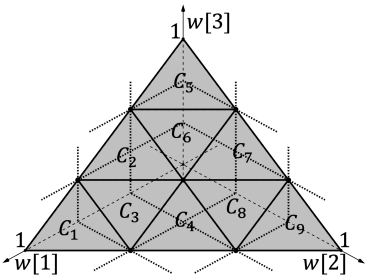

Figure 6: Partitioning of Δ^2

$O(s^{\lfloor d/2 \rfloor+\epsilon})$ time preprocessing on D, the query is answered in $O(\log s + |\mathcal{D}_\alpha(w)|)$ time [19], where ϵ is any positive constant.

Theorem 7. *With an index of size $O(s^{\lfloor d/2 \rfloor+\epsilon})$ constructed in $O(n \log^{d-2} s + s^{\lfloor d/2 \rfloor+\epsilon})$ time,* MDRAH *runs in $O(r \log^2 s + r^2\tau \log s)$ time, where ϵ is any positive constant and $\tau = \max_{w\in W} |\mathcal{D}_\alpha(w)|$.*

Proof. Index building includes precomputing Sky(D). Through the previously mentioned data structures for the ray shooting and halfspace range queries, it requires $O(m(\log s + \tau))$ time to compute all $\mathcal{D}_\alpha(w)$ for $w \in W$. Each iteration of the greedy process can be realized by scanning $\{\mathcal{D}_\alpha(w)\}_{w\in W}$. The process takes $O(rm\tau)$ time in total. Therefore. the time complexity is $O(r \log^2 s + r^2\tau \log s)$. □

When $r^2\tau$ is $O(n^{1-\epsilon})$ for a positive constant ϵ, MDRAH achieves sub-linear runtime. In experiments, both r and τ are constant w.r.t. n. The closer α is to 1, the smaller τ is.

5.2 MDRAH$^+$: Preference Space Partitioning

In this section, improve MDRAH and propose an algorithm based on space partitioning, MDRAH$^+$, to reduce the number of sample vectors actually processed. It divides the preference space into a bounded number of cells, to approximately describe the high-scoring spaces of all tuples. The partitioning is independent of $r\alpha$Hit's input. The algorithm also uses sampling for estimation, but the sample size drops from $O(r \ln s)$ in MDRAH to $O(1)$. After partitioning and sampling, the problem is converted into finding the weighted maximum coverage [14] on a fixed size set system (much smaller than Σ_1). By calling the exact solver for weighted MC, the algorithm has no relative error of $1/e$ in the approximation guarantee. Although the time complexity is theoretically near-linear (less than that of MDRAH), MDRAH$^+$ may be inefficient in practice, mainly due to the exact solver. Therefore, a practical adaptation of MDRAH$^+$ is considered by replacing it with a greedy approximate solver. At last, consider accelerating the algorithm through preprocessing and indexing. Relative to MDRAH, MDRAH$^+$ requires less preprocessing time but has smaller running time.

Preference Space Partitioning. MDRAH$^+$ partitions each dimension of the preference space into γ equal ranges, where γ is an application-specific parameter that controls the granularity of space partitioning. The imposed slicing decomposes Δ^{d-1} into a set Λ of cells. A cell $C \in \Lambda$ is associated with d positive integers $\beta_1, \beta_2, \cdots, \beta_d$ no larger than γ, where β_i corresponds to the range $[\frac{\beta_i-1}{\gamma}, \frac{\beta_i}{\gamma})$ on the i-th dimension. Formally, C is defined by the following $(d+1)$ constrains:

- $\sum_{i=1}^d w[i] = 1$,
- $\forall i \in [d]$, $\frac{\beta_i-1}{\gamma} \le w[i] < \frac{\beta_i}{\gamma}$.

The size of complete Λ is $\lambda = \binom{\gamma+d-1}{d} - \binom{\gamma}{d} = O(\gamma^{d-1})$ [21]. Define w_C to be the representative vector of C with

$$w_C[i] = \frac{\beta_i}{\gamma} - (\sum_{j=1}^d \frac{\beta_j}{\gamma} - 1)/d.$$

Assuming $\gamma = 3$, Figure 5 demonstrates the 3 cells (segments along the diagonal) when $d = 2$, and Figure 6 shows the 9 cells (triangles in the shaded surface) when $d = 3$. Cell C_7 corresponds to $[0, 1/3) \times [1/3, 2/3) \times [1/3, 2/3)$. Its representative vector is $\langle 1/9, 4/9, 4/9\rangle$.

A tuple $t \in D$ is said to *cover* a cell $C \in \Lambda$, if t is a high-scoring tuple under the representative vector w_C, i.e., $f_{w_C}(t) \ge \alpha f_{w_C}(D)$. Further, we say a set S *covers* C, if S contains a high-scoring tuple for w_C. Let $\Lambda_\alpha(S)$ denote the set of cells covered by S, i.e.,

$$\Lambda_\alpha(S) = \{C \in \Lambda \mid f_{w_C}(S) \ge \alpha f_{w_C}(D)\}.$$

For convenience, $\Lambda_\alpha(t) = \Lambda_\alpha(\{t\})$. Let $C_\alpha(S)$ be the probability of a weight vector lying in a cell covered by S, i.e.,

$$C_\alpha(S) = \sum_{C\in\Lambda_\alpha(S)} \mathcal{P}(C).$$

Next, show that $C_\alpha(S)$ can approximately replace $\mathcal{H}_\alpha(S)$.

Lemma 4. *Given a partitioning parameter $\gamma = O(\frac{1}{\epsilon})$, for all $t \in D$ and $C \in \Lambda$,*

- *if there is a w in C with $f_w(t) \ge (\alpha+\epsilon) f_w(D)$, then $f_{w_C}(t) \ge \alpha f_{w_C}(D)$, i.e., C is covered by t, and*
- *if C is covered by t, then $\forall w \in C$, $f_w(t) \ge (\alpha-\epsilon) f_w(D)$.*

Proof. W.l.o.g., consider the open-interval representation of C, i.e.,

- $\sum_{i=1}^d w[i] = 1$,
- $\forall i \in [d]$, $\frac{\beta_i-1}{\gamma} < w[i] < \frac{\beta_i}{\gamma}$.

Let $\kappa = \sum_{i=1}^d \beta_i - \gamma$. Due to $\sum_{i=1}^d \frac{\beta_i-1}{\gamma} < 1$ and $\sum_{i=1}^d \frac{\beta_i}{\gamma} > 1$, κ is in $[d-1]$. For a w in C, the L_1-norm of $w - w_C$, denoted by

$$\|w - w_C\|_1 = \sum_{i=1}^d \left|w[i] - w_C[i]\right|,$$

takes the maximum, iff the κ dimensions of w approach the left boundaries of ranges and the remaining $(d-\kappa)$ dimensions are close to the right boundaries. Assume the first κ approach the left boundaries. Then $\forall w \in C$,

$$\begin{aligned}\|w - w_C\|_1 &< \sum_{i=1}^\kappa \left|\frac{\beta_i - 1}{\gamma} - w_C[i]\right| + \sum_{i=\kappa+1}^d \left|\frac{\beta_i}{\gamma} - w_C[i]\right| \\ &= \kappa \times \frac{d-\kappa}{d\gamma} + (d-\kappa) \times \frac{\kappa}{d\gamma} = \frac{2\kappa(d-\kappa)}{d\gamma}.\end{aligned}$$

Due to $\kappa(d-\kappa) \le d^2/4$, $\|w - w_C\|_1 < \frac{d}{2\gamma}$. The range of each attribute of D is normalized to $[0, 1]$. Thus $\forall t \in D$ and $\forall w \in C$,

$$\begin{aligned}\left|f_w(t) - f_{w_C}(t)\right| &= \left|\sum_{i=1}^d t[i](w[i] - w_C[i])\right| \\ &\le \sum_{i=1}^d t[i]\left|w[i] - w_C[i]\right| \\ &\le \sum_{i=1}^d \left|w[i] - w_C[i]\right| < \frac{d}{2\gamma}.\end{aligned}$$

For each $i \in [d]$, there is a tuple $t \in D$ with $t[i] = 1$. Each $w \in \Delta^{d-1}$ has $\sum_{i=1}^{d} w[i] = 1$. Thus $f_w(D) \geq 1/d$ for all $w \in \Delta^{d-1}$.

Prove the first sentence. Let t^* be the top-scoring tuple for w_C. We have

$$f_w(D) \geq f_w(t^*) > f_{w_C}(t^*) - \frac{d}{2\gamma} = f_{w_C}(D) - \frac{d}{2\gamma}.$$

Then

$$\begin{aligned} f_{w_C}(t) &> f_w(t) - \frac{d}{2\gamma} \geq (\alpha+\epsilon) f_w(D) - \frac{d}{2\gamma} \\ &> (\alpha+\epsilon)(f_{w_C}(D) - \frac{d}{2\gamma}) - \frac{d}{2\gamma} \\ &\geq \alpha f_{w_C}(D) + \frac{\epsilon}{d} - \frac{(\alpha+\epsilon+1)d}{2\gamma}. \end{aligned}$$

The last "$\geq$" is due to $f_{w_C}(D) \geq 1/d$. Let γ be no less than $\frac{(\alpha+\epsilon+1)d^2}{2\epsilon}$. Then $f_{w_C}(t) > \alpha f_{w_C}(D)$.

Prove the second. Similarly, we have $f_{w_C}(D) > f_w(D) - \frac{d}{2\gamma}$ and $f_w(D) \geq 1/d$. Then

$$\begin{aligned} f_w(t) &> f_{w_C}(t) - \frac{d}{2\gamma} \geq \alpha f_{w_C}(D) - \frac{d}{2\gamma} \\ &> \alpha(f_w(D) - \frac{d}{2\gamma}) - \frac{d}{2\gamma} \\ &\geq (\alpha-\epsilon) f_w(D) + \frac{\epsilon}{d} - \frac{(\alpha+1)d}{2\gamma}. \end{aligned}$$

If $\gamma \geq \frac{(\alpha+1)d^2}{2\epsilon}$, then $f_w(t) > (\alpha-\epsilon) f_w(D)$. □

Recall that the high-scoring space of a tuple t is $\mathcal{U}_\alpha(t) = \{w \in \Delta^{d-1} \mid f_w(t) \geq \alpha f_w(D)\}$. Then we have the following.

Corollary 2. $\forall t \in D$, $\mathcal{U}_{\alpha+\epsilon}(t) \subseteq \cup_{C \in \Lambda_\alpha(t)} C \subseteq \mathcal{U}_{\alpha-\epsilon}(t)$.

Proof Sketch. Based on Lemma 4, if a tuple t satisfies $f_w(t) \geq (\alpha+\epsilon) f_w(D)$ for a w, i.e., $w \in \mathcal{U}_{\alpha+\epsilon}(t)$, then the cell C containing w is covered by t. Further, if a cell C is covered by the tuple t, then $C \subseteq \mathcal{U}_{\alpha-\epsilon}(t)$. □

Due to $\mathcal{U}_{\alpha+\epsilon}(t) \subseteq \mathcal{U}_\alpha(t) \subseteq \mathcal{U}_{\alpha-\epsilon}(t)$, the cells covered by a tuple t together form an approximation of $\mathcal{U}_\alpha(t)$. In this way, the cells in Λ can approximately describe all high-scoring spaces just like pixels. Knowing $\mathcal{H}_\alpha(S) = \mathcal{P}(\cup_{t \in S} \mathcal{U}_\alpha(t))$, show that $C_\alpha(S)$ can bound the hit probability of S.

Corollary 3. $\forall S \subseteq D$, $\mathcal{H}_{\alpha+\epsilon}(S) \leq C_\alpha(S) \leq \mathcal{H}_{\alpha-\epsilon}(S)$.

Proof Sketch. Based on Lemma 4, if S $(\alpha+\epsilon)$-hits a w in the cell C, i.e., $f_w(S) \geq (\alpha+\epsilon) f_w(D)$, then C is covered by S. Further, if a cell C is covered by S, all $w \in C$ are $(\alpha-\epsilon)$-hit by S. □

$C_\alpha(S)$ is the upper bound of $\mathcal{H}_{\alpha+\epsilon}(S)$ and lower bound of $\mathcal{H}_{\alpha-\epsilon}(S)$. Assuming $1/\epsilon$ is a constant, both the partitioning parameter γ and number λ of cells are also constants. In the following, consider finding a size r set S maximizing $C_\alpha(S)$.

Algorithm 3: MDRAH$^+$

Input: D, Θ, r, α;
Output: a size r set $S \subseteq D$;
1 Calculate Sky(D) and set D = Sky(D);
2 Construct set W of m sampled weight vectors;
3 $\Lambda = \emptyset$;
4 **foreach** $w \in W$ **do**
5 Calculate cell C to which w belongs;
6 **if** C is in Λ **then** $\omega(C) = \omega(C) + 1$;
7 **else**
8 Compute $\mathcal{D}_\alpha(w_C)$;
9 Insert C into Λ and set $\omega(C) = 1$;
10 Traverse $\{\mathcal{D}_\alpha(w_C)\}_{C \in \Lambda}$ to get $\{\Lambda_\alpha(t)\}_{t \in D}$;
11 Remove duplicate sets from $\{\Lambda_\alpha(t)\}_{t \in D}$ and delete corresponding tuples from D;
12 **return** Exact-Weighted-MC(Λ, $\{\Lambda_\alpha(t)\}_{t \in D}$, $\omega(\cdot)$, r);

Sampling for Estimation. The exact calculation of $C_\alpha(S)$ still involves complex multiple integrals over cells (i.e., $\mathcal{P}(C)$). Therefore, similar to MDRAH, MDRAH$^+$ estimates $C_\alpha(S)$ based on sampling. W is the set of m sample vectors drawn from Δ^{d-1} based on the preference distribution Θ.

$$C_\alpha(S, W) = \frac{1}{m} \times \left|\{w \in W \mid w \in \cup_{C \in \Lambda_\alpha(S)} C\}\right|$$

is the proportion of vectors in W falling into a cell covered by S, an unbiased estimate of $C_\alpha(S)$. Like Lemma 3, sufficient samples can ensure that the estimation error is small enough.

Lemma 5. *Given a sample size* $m = \frac{\ln 2 + \min(r \ln s, \lambda \ln 2) + \ln \frac{1}{\delta}}{2\varepsilon^2}$, *with probability at least* $1-\delta$, $\forall S \subseteq D$ *with* $|S| = r$,

$$\left|C_\alpha(S, W) - C_\alpha(S)\right| \leq \varepsilon.$$

The above sample size is reduced from $O(r \ln s)$ in Lemma 3 to $O(1)$. Lemma 3 guarantees that the estimation error is small for all r-size subsets of D, the number of which is at most s^r. It is sometimes redundant for MDRAH$^+$. $C_\alpha(S)$ is determined by $\Lambda_\alpha(S)$, which is a subset of Λ and has 2^λ different cases. If $2^\lambda < s^r$, it is enough to ensure that the error is bounded for all cases. Note that if $m = (\ln 2 + \lambda \ln 2 + \ln \frac{1}{\delta})/2\varepsilon^2$ is directly set, then the error is bounded for all subsets of D (regardless of size).

Weighted Maximum Coverage. After sampling, the problem is equivalent to an instance of the weighted maximum coverage problem [14]. The corresponding set system is $\Sigma_3 = (\Lambda, \{\Lambda_\alpha(t)\}_{t \in D})$. Every element $C \in \Lambda$ has a weight

$$\omega(C) = \left|\{w \in W \mid w \in C\}\right|.$$

Find a maximum coverage with maximum weight, corresponding to a set S with highest $C_\alpha(S, W) = \frac{1}{m} \sum_{C \in \Lambda_\alpha(S)} \omega(C)$.

Consider computing Λ and $\{\Lambda_\alpha(t)\}_{t \in D}$. The cells C with $\omega(C) = 0$ are redundant. Therefore, MDRAH$^+$ does not pre-divide Δ^{d-1} to obtain the complete Λ. Instead, construct Λ online. Starting from $\Lambda = \emptyset$, it continuously inserts cells C with $\omega(C) > 0$ into Λ by traversing W (line 3-9 in Algorithm 3). In the following, Λ

refers to the set of cells with a weight greater than 0, and its size satisfies $|\Lambda| \le \min(\lambda, m)$, bounded by a constant. Note that $\Lambda_\alpha(t) = \{C \in \Lambda \mid t \in \mathcal{D}_\alpha(w_C)\}$. Similar to MDRAH, MDRAH$^+$ obtains $\{\Lambda_\alpha(t)\}_{t\in D}$ by computing $\{\mathcal{D}_\alpha(w_C)\}_{C\in\Lambda}$. The number of $\mathcal{D}_\alpha(w_C)$ is much smaller than the size of $\{\mathcal{D}_\alpha(w)\}_{w\in W}$ in MDRAH. The dataset and set system can be further simplified. If multiple t in D has the same $\Lambda_\alpha(t)$, D retains just one of them and deletes the others. Correspondingly, $\{\Lambda_\alpha(t)\}_{t\in D}$ is also simplified. After simplifying, the dataset D and collection $\{\Lambda_\alpha(t)\}_{t\in D}$ are of constant size (i.e., $\min(2^{|\Lambda|}, s)$).

The problem is transformed into a weighted maximum coverage instance on a constant-sized set system. Therefore, although the weighted MC problem is NP-hard [14], the optimal solution for such an instance is, theoretically, possible to obtain in constant time. MDRAH$^+$ adopts the exact solver for weighted MC (i.e., Exact-Weighted-MC in Algorithm 3) to find the maximum coverage with the maximum weight. The corresponding tuple set S has the highest $C_\alpha(S, W)$. To get the optimal coverage, Exact-Weighted-MC traverses all r-sized sub-collections of $\{\Lambda_\alpha(t)\}_{t\in D}$, whose number is at most $\min(2^{\min(2^{|\Lambda|},s)}, (\min(2^{|\Lambda|}, s))^r)$.

Theoretical Results.

Theorem 8. *With probability at least* $1-\delta$, MDRAH$^+$ *outputs a tuple set* S *satisfying*

$$\mathcal{H}_{\alpha-\epsilon}(S) \ge \mathcal{H}_{\alpha+\epsilon}(S^*) - 2\varepsilon, \tag{4}$$

where S^* *is the optimal solution for* $r\alpha$Hit.

Proof Sketch. Referring to Lemma 3, $\mathcal{H}_{\alpha-\epsilon}(S) \ge C_\alpha(S)$ and $C_\alpha(S^*) \ge \mathcal{H}_{\alpha+\epsilon}(S^*)$. Based on Lemma 5, $C_\alpha(S) \ge C_\alpha(S, W) - \varepsilon$ and $C_\alpha(S^*, W) \ge C_\alpha(S^*) - \varepsilon$. Due to an exact solver for weighted MC, $C_\alpha(S, W) \ge C_\alpha(S^*, W)$. □

By adjusting α, we can replace Formula (4) with

$$\mathcal{H}_\alpha(S) \ge \mathcal{H}_{\alpha+2\epsilon}(S^*) - 2\varepsilon \ \text{ or } \ \mathcal{H}_{\alpha-2\epsilon}(S) \ge \mathcal{H}_\alpha(S^*) - 2\varepsilon.$$

Compared to Theorem 5, Theorem 8 relaxes the threshold α but has no relative error of $1/e$. The relaxation is also adjustable and can be arbitrarily small.

Theorem 9. MDRAH$^+$ *takes* $O(n\log^{d-2} s)$ *time.*

Proof. It takes $O(n\log^{d-2} s)$ time to calculate Sky(D) for $d \ge 3$. It requires $O(m + s|\Lambda|)$ time to compute $\{\mathcal{D}_\alpha(w_C)\}_{C\in\Lambda}$ and obtain $\Sigma_3 = (\Lambda, \{\Lambda_\alpha(t)\}_{t\in D})$ (including simplification). The simplified set system has $|\Lambda|$ items and at most $2^{|\Lambda|}$ sets. The subroutine Exact-Weighted-MC takes $O(2^{2^{|\Lambda|}}2^{|\Lambda|}|\Lambda|)$ time, where $|\Lambda| \le \lambda = O(\gamma^{d-1}) = O(\frac{1}{\epsilon^{d-1}})$. As shown in Lemma 5, m is $O((\frac{1}{\epsilon^{d-1}} + \ln\frac{1}{\delta})\frac{1}{\varepsilon^2})$. Note that $\frac{1}{\epsilon}$, $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$ in Theorem 8 are constants in practice. Then both $|\Lambda|$ and m are $O(1)$. Thus MDRAH$^+$, theoretically, takes $O(n\log^{d-2} s)$ time. □

Regardless of calculating Sky(D), MDRAH$^+$ requires $O(s)$ time, much smaller than MDRAH's $O(rs\log s)$ time.

Practical Implementation. Although MDRAH$^+$ has near-linear time complexity in theory, it can be inefficient in practice. The inefficiency mainly comes from Exact-Weighted-MC. To make MDRAH$^+$ practical, consider replacing it with a naive greedy method like Greedy-MC in MDRAH. Starting from $S = \emptyset$, iteratively add the tuple t covering the most uncovered weights, i.e, maximizing

$$\sum\nolimits_{C\in\Lambda_\alpha(t)\setminus\Lambda_\alpha(S)} \omega(C),$$

to S until $|S| = r$. Regarding $C_\alpha(S, W)$, the greedy method guarantees an $(1 - 1/e)$-approximation. It adds another level of approximation and turns Formula (4) in Theorem 8 into

$$\mathcal{H}_{\alpha-\epsilon}(S) \ge (1 - 1/e)\mathcal{H}_{\alpha+\epsilon}(S^*) - (2 - 1/e)\varepsilon, \tag{5}$$

which is similar to Formula (3) in Theorem 5. The greedy method takes $O(\min(2^{|\Lambda|}, s)|\Lambda|)$ time. It does not improve the time complexity in Theorem 9, but makes MDRAH$^+$ efficient in practice. As shown in experiments, the practical implementation of MDRAH$^+$ is much faster than MDRAH, but has similar output quality.

Speeding up with Preprocessing & Indexing. Like MDRAH, use indexing and preprocessing to speed up MDRAH$^+$. First, precalculate Sky(D) and set D = Sky(D). Second, consider accelerating the calculation of $\{\mathcal{D}_\alpha(w_C)\}_{C\in\Lambda}$. Note that the complete Λ and all w_C can be obtained in advance, which are independent of the input of $r\alpha$Hit. Therefore, build λ ordered lists as index. Each list is obtained by sorting the tuples in D w.r.t. $f_{w_C}(\cdot)$ for a cell $C \in \Lambda$. To obtain $\mathcal{D}_\alpha(w_C)$, MDRAH$^+$ just traverses the first $(|\mathcal{D}_\alpha(w_C)| + 1)$ tuples of the list, taking $O(|\mathcal{D}_\alpha(w_C)|)$ time.

Theorem 10. *With an* $O(s)$*-size index constructed in* $O(n\log^{d-2} s)$ *time,* MDRAH$^+$ *runs in* $O(\tau)$ *time, where* $\tau = \max_{C\in\Lambda} |\mathcal{D}_\alpha(w_C)|$.

Proof. It takes $O(n\log^{d-2} s)$ time to calculate Sky(D) for $d \ge 3$. It takes $O(s\lambda\log s)$ time to computing λ sorted lists. Due to λ is a constant, the processing time is $O(n\log^{d-2} s)$.

It requires $O(m+\lambda\tau)$ time to compute $\{\mathcal{D}_\alpha(w_C)\}_{C\in\Lambda}$ and obtain $\Sigma_3 = (\Lambda, \{\Lambda_\alpha(t)\}_{t\in D})$ (including simplification). Both λ and m are $O(1)$. The maximum coverage process takes $O(1)$ time. Therefore, the runtime of MDRAH$^+$ is $O(\tau)$. □

Note that preprocessing and indexing are completely dependent on the dataset D, without any knowledge of the preference distribution Θ, threshold α and size bound r. Referring to Theorem 7, compared with MDRAH, MDRAH$^+$ requires less preprocessing time but has smaller running time.

6 EXTENSIONS

6.1 αHit Permutation

Consider some cases where the size bound r cannot be easily determined in advance. For an online shop, users may not be interested in the products shown on the first page. Then it must continue to determine which products to display on the second page and so on. Further, the online shop can be browsed on different devices. The number of products viewed at one time on a mobile phone is less than that on a laptop, thus requiring a smaller r. The goal of the above should be to produce a total order of the tuples in D, such that a solution of $r\alpha$Hit for any r is simply the set of the first r tuples in that order. This leads to an extended problem definition.

Problem 2 (αHit Permutation Problem). *Given a threshold $\alpha \in [0, 1]$, sort the tuples in D to maximize $\sum_{i=1}^{n} \mathcal{H}_\alpha(D_i)$, where D_i refers to the set of first i tuples in the sorted list.*

Note that $\mathcal{H}_\alpha(\text{Sky}(D)) = 1$. Put the non-skyline tuples at the end of list and take the ordering of skyline tuples into account. With simple modifications, both MDRAH and MDRAH$^+$ can obtain provably good solutions to the permutation problem.

For MDRAH, first change the sample size m in Lemma 3 to $(\ln 2 + s \ln 2 + \ln \frac{1}{\delta})/2\varepsilon^2$. It ensures that with high probability, the estimation error for all subsets of Sky(D) is bounded. Second, modify the termination condition of the greedy process to S covering all vectors in W, i.e., $\cup_{t \in S} W_\alpha(t) = W$. In the sorted list, the tuples in Sky(D) $\setminus$ S are placed after the tuples in S. As a result, with a probability at least $1 - \delta$, Formula (3) holds for every prefix D_i. Then the approximation guarantee for the hit probability transfers to the accumulated hit probability. At the cost, the time complexity of MDRAH increases to $O(n \log^{d-2} s + s^2)$.

Modify the practical implementation of MDRAH$^+$ in a similar way. The new sample size is $(\ln 2 + \min(\lambda, s) \ln 2 + \ln \frac{1}{\delta})/2\varepsilon^2$. The new termination condition is that S covers all cells in Λ. Then, with high probability, for every prefix D_i, Formula (5) holds. Differently, the time complexity of MDRAH$^+$ is still $O(n \log^{d-2} s)$.

6.2 Continuous $r\alpha$Hit

Consider the case where the dataset D is a continuous convex set, in particular a convex polytope, represented by its set A of vertices, i.e., D is the convex hull CH(A). An example is multi-objective linear programming (MOLP), which maximizes a set of d linear functions subject to a set of linear constraints. The d function values for each feasible solution form a tuple. Several algorithms [12, 40] have been developed for MOLP to generate all vertices of the tuple space.

Define the continuous $r\alpha$Hit problem on CH(A).

Problem 3 (Continuous (r, α)-Hit Problem). *Given a set A of vertices, a size bound $r \geq 1$ and a threshold $\alpha \in [0, 1]$, compute a size r subset of* CH(A) *that maximizes the $\mathcal{H}_\alpha(S)$. Denote the optimal solution to this problem by $S^*(\text{CH}(A), r)$.*

Note that $\forall w \in \Delta^{d-1}$, $f_w(A) = f_w(\text{CH}(A))$. Thus,

$$\mathcal{H}_\alpha(S) = \int_{w \in \Delta^{d-1}} \mathbb{I}(f_w(S) \geq \alpha f_w(A)) \times \eta(w) dw.$$

Relative to discrete $r\alpha$Hit on A (i.e., $D = A$), the continuous problem on CH(A) approximates the same top score, but has a larger candidate tuple space. Let $S^*(A, r)$ be the optimal solution of discrete $r\alpha$Hit on A. The following theorem shows that using only tuples in A instead of CH(A) requires at most d times more points to achieve the same hit probability.

Theorem 11. $\mathcal{H}_\alpha(S^*(A, dr)) \geq \mathcal{H}_\alpha(S^*(\text{CH}(A), r))$.

Proof. Let $S^*(\text{CH}(A), r) = \{t_i^*\}_{i \in [r]}$. W.l.o.g., assume t_i^* is on the boundary of CH(A). Otherwise, t_i^* is dominated by a tuple t on the boundary. The set $S^*(\text{CH}(A), r) \setminus \{t_i^*\} \cup \{t\}$ has the highest hit probability but satisfies the assumed property.

Then, t_i^* belongs to a facet of CH(A). The facet is a $(d-1)$ dimensional convex polytope. Following Carathéodory's theorem, t_i^* can be expressed as a convex combination of (at most) d vertices in A. The union of these d-sets of vertices is a solution to the discrete problem. □

Therefore, our algorithms in this paper can be invoked to obtain theoretically guaranteed solutions to continuous $r\alpha$Hit, by running directly on A. The design of algorithms specifically for the continuous problem is our future work.

7 EXPERIMENTS

7.1 Experimental Setting

All experiments are conducted on a Core-I9 machine, running CentOS 7 with 256GB main memory and 1TB hard disk. All algorithms are implemented in C++.

The experiments use both synthetic and real datasets. The anti-correlated dataset (Anti) is generated by the generator proposed by Borzsony et. al [5]. Specifically, select a hyperplane perpendicular to the diagonal of the data space $[0, 1]^d$ using a normal distribution with a small standard deviation. Within the hyperplane, the new tuple is generated using a uniform distribution. Anti has a massive skyline demanding a concise representation, and is most interesting [28, 36]. Due to space limitations, the results on the correlated and independent datasets are not shown. The real data includes four datasets. Island [22, 36, 37] contains 63,383 2D geographic positions. Colors [22, 23, 34] is a collection of color moments in 9 dimensions collected from 68,040 images. NBA [23, 34, 38] includes $d = 4$ statistics for 21,960 NBA players. Household [23, 31, 38] contains 127,931 tuples of household expenditures on $d = 6$ aspects. Following [3, 10, 33, 36, 38], the range of each dimension is normalized to $[0, 1]$.

The experiments compare the following seven algorithms:

- 2DRAH: the 2D exact algorithm for $r\alpha$Hit in Section 4.
- 2DRKH [36]: the 2D exact algorithm for rkHit.
- MDRAH: the sampling-based multidimensional (MD) approximation algorithm for $r\alpha$Hit in Section 5.1.
- MDRAH$^+$: the MD approximation algorithm combined with preference space partitioning in Section 5.2. In its practical implementation, apply the greedy approximation method to solve the weighted MC problem.
- MDRKH [36]: the MD approximation algorithm for rkHit.
- HDRRM [35]: the MD approximation algorithm for the rank-regret minimizing (RRM) query. It requires $r \geq d$. Thus we relax its output size from r to $(r + d)$.
- MDRMS [2]: the MD approximation algorithm designed for the regret minimization set (RMS) query.

The experiments do not take into account the hit_Alg algorithm [23] proposed for the $(r, 1)$-hit (rHit) problem. When $\alpha = 1$, the process of MDRAH is exactly the same as hit_Alg.

The experiments use two measurements to measure the performance of each algorithm, namely the execution time and α-hit probability of the output. Section 5 considers accelerating MDRAH and MDRAH$^+$ based on indexing and preprocessing. However, for the sake of fairness, no indexing is assumed in experiments, and the execution time includes the preprocessing time. The experiments randomly draw 10,000 scoring functions based on the given preference distribution Θ and use them to estimate hit probabilities.

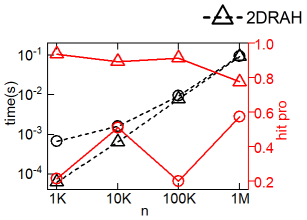

Figure 7: Anti, 2D: vary n

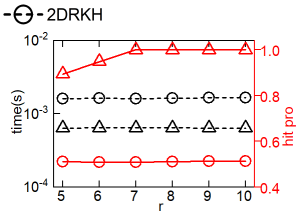

Figure 8: Anti, 2D: vary r

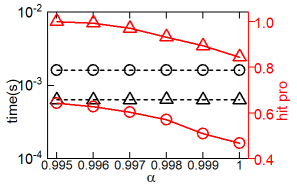

Figure 9: Anti, 2D: vary α

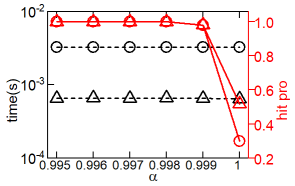

Figure 10: Island, 2D: vary α

7.2 2D Experimental Result

The performance of two 2D algorithms is studied on the anti-correlated (Anti) and Island datasets. The default values for dataset size, size bound and threshold are $n = 10\text{K}$ ($\text{K} = 10^3$), $r = 5$ and $\alpha = 0.999$. The preference distribution adopted is a uniform distribution on Δ^1. In Figures 7 to 10, the triangle and circle markers represent algorithms 2DRAH and 2DRKH. The black dashed and red solid lines show the runtime and hit probability, respectively.

7.2.1 Impact of the dataset size (n). Vary n from 1K to 1M (i.e., 1000K). Figure 7 show the results for 2D Anti. As the data volume increases, the runtime of algorithms increases significantly. 2DRAH is efficient and always faster than 2DRKH. When $n = 1\text{K}$, it is at least one order of magnitude faster. The *first step* of 2DRAH (resp., 2DRKH) is to calculate the skyline (resp., k-skyband [36]), and the *remaining steps* are performed on the skyline (resp., k-skyband). The skyline is a much smaller subset of the k-skyband. However, as n grows, the gap in time becomes smaller. This is because the sizes of skyline and k-skyband grow sub-linearly with n. For a larger n, the first step takes up a higher proportion of the overall runtime, and the remaining steps take up less. Calculating the skyline and k-skyband are both based on sorting the dataset and therefore take similar time. As n increases, the output quality of 2DRAH roughly decreases, still much better than that of 2DRKH. 2DRKH does not aim to maximize the α-hit probability, so the output quality fluctuates up and down. When $n = 1\text{K}$ or 100K, its hit probability is less than 0.22.

7.2.2 Impact of the size bound (r). r is varied from 5 to 10. Figure 8 shows the results for the 2D anti-correlated dataset. 2DRAH is faster than 2DRKH. The running time of both is not affected by the size bound. For 2DRAH, both the skyline and high-scoring spaces are calculated independently of r. As r increases, the hit probability of 2DRAH improves and is close to 1, indicating that nearly all users (or scoring functions) are covered by the solution. The hit probability of 2DRKH is basically unchanged, about 0.51.

7.2.3 Impact of the threshold (α). Vary α from 0.995 to 1. Figures 9 and 10 show the results for Anti and Island. The runtime of both algorithms is unaffected by α. As α grows, the hit probabilities decrease. For Anti, 2DRAH has much better output quality than 2DRKH. For Island, 2DRAH has a hit probability about 0.517 when $\alpha = 1$. It increases to 0.98 for $\alpha = 0.999$. Relaxation of α is necessary.

7.3 MD Experimental Result

Figures 11 to 30 show the performance of five MD algorithms on Anti and three real datasets (Colors, NBA and Household). The default values for dataset size, dimensionality, size bound and threshold are set to $n = 10\text{K}$, $d = 4$, $r = 5$ and $\alpha = 0.99$. Based on Lemma 3, the sample size of MDRAH is $(\ln 2 + r \ln s + \ln \frac{1}{\delta})/2\varepsilon^2$, where by default, $\varepsilon = 0.03$ and $\delta = 0.01$. Based on Lemma 5, the sample size of MDRAH$^+$ is $(\ln 2 + \min(r \ln s, \lambda \ln 2) + \ln \frac{1}{\delta})/2\varepsilon^2$, where $\lambda = \binom{\gamma+d-1}{d} - \binom{\gamma}{d}$. Set $\gamma = 12$ by default. The default preference distribution is a uniform distribution on Δ^{d-1}. In Figures 11 to 30, the triangle, inverted triangle, circle, rhombus and rectangle markers represent MDRAH, MDRAH$^+$, MDRKH, HDRRM and MDRMS. The runtime and hit probability are shown in two figures, respectively.

7.3.1 Impact of the dataset size (n). n is varied from 1K to 1M. Figures 11 and 12 show the results for 4D Anti. As n increases, the runtime grows. MDRAH$^+$ is the fastest. Compared with MDRAH, it requires fewer samples, calculates a smaller number of $\mathcal{D}_\alpha(w)$, and operates on a smaller set system. MDRAH is also faster than the others. Even when $n = 1\text{M}$, the runtime of both algorithms is just a few seconds. As n grows, the time gap between MDRAH and MDRAH$^+$ becomes smaller. The reason is that both have to calculate the skyline, which takes up most of the runtime when n is large. This is similar to 2DRAH vs 2DRKH. HDRRM is the slowest, at least two orders of magnitude slower than MDRAH$^+$. MDRAH has the best output quality. MDRAH$^+$ gets the second, very close to the best. Neither HDRRM nor MDRMS maximize the hit probability, so the output quality is terrible. When $n = 10\text{K}$, their hit probabilities are 0.269 and 0.171.

7.3.2 Impact of the number of attributes (d). We vary d from 3 to 6. Figure 13 shows the runtime for Anti. As d increases, the runtime grows. MDRAH$^+$ has the shortest time, and MDRAH gives the 2-nd. Their running time is about 1 second even for 6D. MDRMS is most sensitive to the increase of d. MDRAH and MDRAH$^+$ are less sensitive. Although the complete Λ contains $O(\gamma^{d-1})$ cells, MDRAH$^+$ just processes the cells C with $\omega(C) > 0$, whose number is not greater than the sample size. Figure 14 shows the hit probability. MDRAH and MDRAH$^+$ have the best and second-best hit probabilities, respectively, which roughly decrease as d grows.

7.3.3 Impact of the size bound (r). Vary r from 5 to 10. Figures 15 and 16 show the results for 4D Anti. As the size bound increases, MDRAH runs slightly slower. There are more samples in W and MDRAH computes $\mathcal{D}_\alpha(w)$ for each $w \in W$. MDRAH$^+$ is less sensitive to the increase of r. The number of cells processed by MDRAH$^+$ (i.e., the number of $\mathcal{D}_\alpha(w_C)$ being calculated) is basically not affected by r. MDRAH$^+$ and MDRAH are the fastest. The hit probabilities approximately grow as r increases. MDRAH has the best one. MDRAH$^+$ has one near-best.

7.3.4 Impact of the threshold (α). α is varied from 0.95 to 1. Figure 17 shows the runtime for 4D Anti. As α increases, the execution

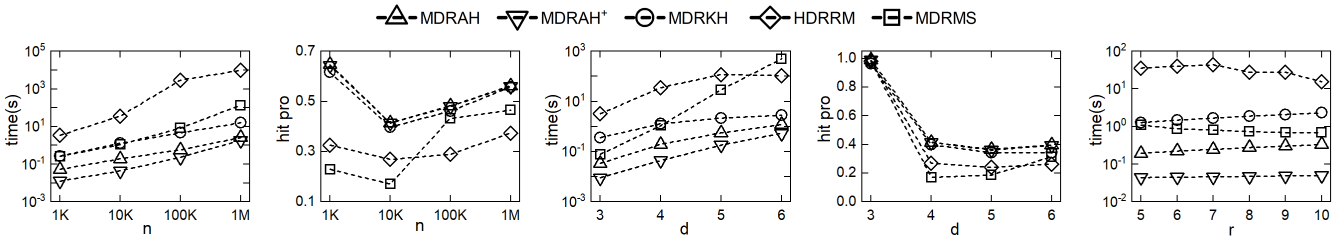

Figure 11: Time, Anti: vary n

Figure 12: HP, Anti: vary n

Figure 13: Time, Anti: vary d

Figure 14: HP, Anti: vary d

Figure 15: Time, Anti: vary r

Figure 16: HP, Anti: vary r

Figure 17: Time, Anti: vary α

Figure 18: HP, Anti: vary α

Figure 19: Time, Anti: vary γ

Figure 20: HP, Anti: vary γ

time of MDRAH and MDRAH$^+$ decreases. A larger α results in a smaller set system in the greedy process. Compared with MDRAH, MDRAH$^+$ is less affected because its greedy process takes up a smaller proportion of the total time. Both algorithms are still much faster than the others. Figure 18 shows the hit probability. As α grows, the output quality of algorithms decreases. MDRAH and MDRAH$^+$ have the two highest hit probabilities. When $\alpha = 0.95$, the hit probability of MDRAH is 0.787. It decreases to 0.261 for $\alpha = 1$. This illustrates the necessity for relaxation on α.

7.3.5 Impact of the partition parameter (γ). Vary γ from 2 to 20. Figures 19 and 20 show the impact of γ on MDRAH$^+$ for 4D Anti. As γ increases, the runtime and output quality roughly increase. When γ is large enough, the hit probability basically remains unchanged but the time continues to increase. Looking at the figures, $\gamma = 12$ seems appropriate.

7.3.6 Impact of the sample size. The sample size is closely related to ε. Vary ε from 0.01 to 0.1 and study the effect of ε on MDRAH and MDRAH$^+$. Figures 21 and 22 show the results for 4D Anti. As ε grows, the sample size becomes smaller and thus the runtime decreases. When $\varepsilon \leq 0.1$, varying ε has little impact on the hit probability. Setting $\varepsilon = 0.03$ seems appropriate. Further decreasing ε will hardly improve the output quality, but will significantly increase the runtime.

7.3.7 Other preference distribution. Following [28, 36], consider a preference distribution in which each dimension has an independent normal distribution with mean 0.5 and standard deviation 0.25. Additionally, negative values are directly converted to the opposite. Vary α from 0.95 to 1. Figure 23 show the runtime on 4D Anti. MDRAH and MDRAH$^+$ are faster than the others. Figure 24 show the results for output quality. The hit probability decreases as α grows. The hit probability of MDRAH$^+$ is close to that of MDRAH, i.e., the best. When $\alpha = 0.95$, MDRAH has a hit probability about 0.898. It falls off to 0.38 for $\alpha = 1$. It is necessary to decrease α.

7.3.8 Real datasets. Evaluate the performance of MD algorithms over three real datasets. α is varied from 0.95 to 1. Figures 25 and 26 show the results on Colors. We set $n = 68{,}040$ and $d = 9$. The time complexity of MDRMS is exponentially related to d, and the memory will overflow when running MDRMS. So there are no results for MDRMS. MDRAH and MDRAH$^+$ are the fastest, almost 4 orders of magnitude faster than HDRRM. As the threshold increases, the output quality decreases. MDRAH and MDRAH$^+$ have the best hit probabilities. MDRKH has the worst one, due to a different optimization goal. In Figures 27 and 28, we set $n = 21{,}960$ and $d = 4$ for NBA. The results are basically the same as that on Colors. Regardless of runtime or output quality, MDRAH and MDRAH$^+$ are the best. HDRRM is the slowest and MDRKH has the worst output quality. When $\alpha \geq 0.97$, the hit probability of MDRKH is less than 0.5. Figures 29 and 30 set $n = 127{,}931$ and $d = 6$ for Household. The output quality is worse than that on the first two datasets. MDRAH has the highest hit probability, which is 0.635 when $\alpha = 0.95$. It decreases to 0.253 for $\alpha = 1$. Decreasing α is necessary.

7.4 Summary

We conduct various experiments on synthetic and real data, showing the effectiveness and efficiency of our algorithms. 2DRAH not only returns the optimal solution, but is also efficient. Even when $n = 1$M, its running time is less than 1/10 second. Among all MD algorithms, MDRAH$^+$ is the fastest and the runtime is less than 2 seconds for $n = 1$M. MDRAH is also faster than the others. The output quality of MDRAH is always the best. MDRAH$^+$ gets a very close one. 2DRKH and MDRKH fail to achieve reasonable output quality in some cases, indicating that maximizing the k-hit probability [36] does not typically maximize the α-hit probability. The experiments show that it is necessary to relax the threshold α. When $\alpha = 1$, the solution just covers a small proportion of users. The α-hit probability increases significantly as α decreases.

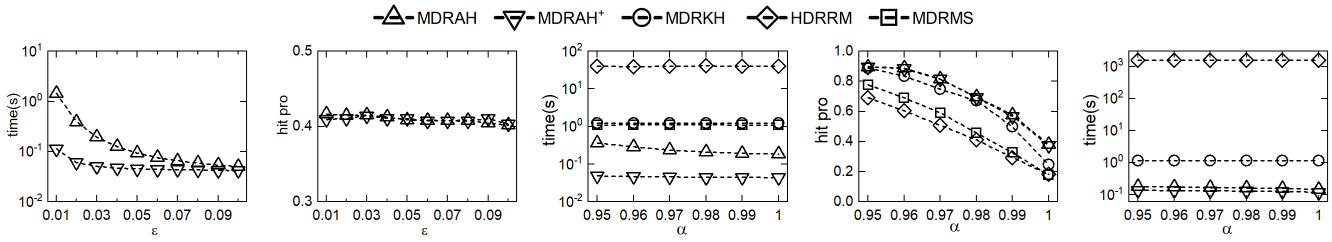

Figure 21: Time, Anti: vary ε

Figure 22: HP, Anti: vary ε

Figure 23: Time, Anti, Normal Distri: vary α

Figure 24: HP, Anti, Normal Distri: vary α

Figure 25: Time, Colors: vary α

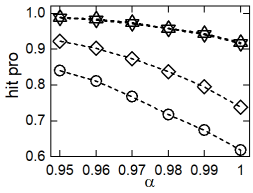

Figure 26: HP, Colors: vary α

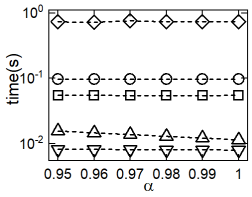

Figure 27: Time, NBA: vary α

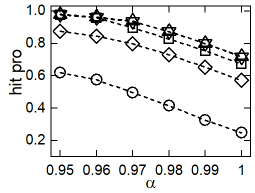

Figure 28: HP, NBA: vary α

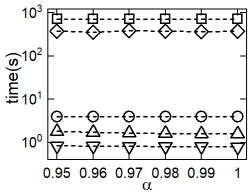

Figure 29: Time, Household: vary α

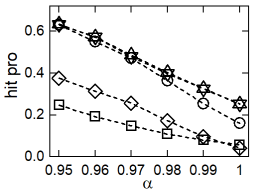

Figure 30: HP, Household: vary α

8 CONCLUSION

In this paper, we propose the $r\alpha$Hit query, which incorporates a given preference distribution and aims to find a fixed size representative set. We get some theoretical and practical results for $r\alpha$Hit. We devise an exact algorithm 2DRAH for $r\alpha$Hit when $d = 2$, and prove that $r\alpha$Hit is NP-hard when $d \geq 3$. Then we provide an approximation algorithm MDRAH, applicable to arbitrary dimension. Further, we improve it and propose an algorithm with smaller time complexity, MDRAH$^+$. Comprehensive experiments verify the efficiency and output quality of our proposed algorithms.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (NSFC) Grant NO. 61832003.

REFERENCES

[1] Pankaj K. Agarwal, Nirman Kumar, Stavros Sintos, and Subhash Suri. 2017. Efficient Algorithms for k-Regret Minimizing Sets. In *16th International Symposium on Experimental Algorithms (SEA 2017) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 75. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 7:1–7:23.

[2] Abolfazl Asudeh, Azade Nazi, Nan Zhang, and Gautam Das. 2017. Efficient Computation of Regret-Ratio Minimizing Set: A Compact Maxima Representative. In *Proceedings of the 2017 ACM International Conference on Management of Data* (Chicago, Illinois, USA) *(SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 821–834.

[3] Abolfazl Asudeh, Azade Nazi, Nan Zhang, Gautam Das, and H. V. Jagadish. 2019. RRR: Rank-Regret Representative. In *Proceedings of the 2019 International Conference on Management of Data* (Amsterdam, Netherlands) *(SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 263–280.

[4] Avrim Blum, Jeffrey Jackson, Tuomas Sandholm, and Martin Zinkevich. 2004. Preference elicitation and query learning. *Journal of Machine Learning Research* 5, Jun (2004), 649–667.

[5] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. 2001. The Skyline Operator. In *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*. IEEE Computer Society, 421–430.

[6] Christian Buchta. 1989. On the average number of maxima in a set of vectors. *Inform. Process. Lett.* 33, 2 (1989), 63–65.

[7] Wei Cao, Jian Li, Haitao Wang, Kangning Wang, Ruosong Wang, Raymond Chi-Wing Wong, and Wei Zhan. 2017. k-regret minimizing set: Efficient algorithms and hardness. In *20th International Conference on Database Theory (ICDT 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[8] Surajit Chaudhuri, Nilesh N. Dalvi, and Raghav Kaushik. 2006. Robust Cardinality and Cost Estimation for Skyline Operator. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*. IEEE Computer Society, 64.

[9] Li Chen and Pearl Pu. 2004. *Survey of Preference Elicitation Methods*. Technical Report. http://infoscience.epfl.ch/record/52659

[10] Sean Chester, Alex Thomo, S Venkatesh, and Sue Whitesides. 2014. Computing k-regret minimizing sets. *Proceedings of the VLDB Endowment* 7, 5 (2014), 389–400.

[11] Wei Chu and Zoubin Ghahramani. 2005. Preference Learning with Gaussian Processes. In *Proceedings of the 22nd International Conference on Machine Learning* (Bonn, Germany) *(ICML '05)*. Association for Computing Machinery, New York, NY, USA, 137–144.

[12] Matthias Ehrgott. 2005. *Multicriteria optimization*. Vol. 491. Springer Science & Business Media.

[13] Uriel Feige. 1998. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)* 45, 4 (1998), 634–652.

[14] Dorit S Hochba. 1997. Approximation algorithms for NP-hard problems. *ACM Sigact News* 28, 2 (1997), 40–52.

[15] Neil Houlsby, Ferenc Huszar, Zoubin Ghahramani, and Jose Hernández-lobato. 2012. Collaborative gaussian processes for preference learning. *Advances in neural information processing systems* 25 (2012).

[16] David G Kirkpatrick and Raimund Seidel. 1985. Output-size sensitive algorithms for finding maximal vectors. In *Proceedings of the First Annual Symposium on Computational Geometry*. 89–96.

[17] Xuemin Lin, Yidong Yuan, Qing Zhang, and Ying Zhang. 2007. Selecting Stars: The k Most Representative Skyline Operator. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*. IEEE Computer Society, 86–95.

[18] Matteo Magnani, Ira Assent, and Michael L Mortensen. 2014. Taking the big picture: representative skylines based on significance and diversity. *The VLDB journal* 23, 5 (2014), 795–815.

[19] Jiri Matousek. 1992. Reporting points in halfspaces. *Computational Geometry* 2, 3 (1992), 169–186.

[20] Jiří Matoušek and Otfried Schwarzkopf. 1993. On ray shooting in convex polytopes. *Discrete & Computational Geometry* 10 (1993), 215–232.

[21] Kyriakos Mouratidis, Keming Li, and Bo Tang. 2023. Quantifying the competitiveness of a dataset in relation to general preferences. *The VLDB Journal* (2023), 1–20.

[22] Danupon Nanongkai, Atish Das Sarma, Ashwin Lall, Richard J. Lipton, and Jun Xu. 2010. Regret-Minimizing Representative Databases. *Proc. VLDB Endow.* 3, 1–2 (sep 2010), 1114–1124.

[23] Peng Peng and Raymong Chi-Wing Wong. 2015. K-Hit Query: Top-k Query with Probabilistic Utility Function. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (Melbourne, Victoria, Australia) *(SIGMOD '15)*. Association for Computing Machinery, New York, NY, USA, 577–592.

[24] Jeff M. Phillips. 2012. Chernoff-Hoeffding Inequality and Applications. *CoRR* abs/1209.6396 (2012). arXiv:1209.6396 http://arxiv.org/abs/1209.6396

[25] Li Qian, Jinyang Gao, and HV Jagadish. 2015. Learning user preferences by adaptive pairwise comparison. *Proceedings of the VLDB Endowment* 8, 11 (2015), 1322–1333.

[26] Paul Resnick and Hal R Varian. 1997. Recommender systems. *Commun. ACM* 40, 3 (1997), 56–58.

[27] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*. Springer, 1–35. https://doi.org/10.1007/978-0-387-85820-3_1

[28] Atish Das Sarma, Ashwin Lall, Danupon Nanongkai, Richard J. Lipton, and Jun (Jim) Xu. 2011. Representative skylines using threshold-based preference distributions. In *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*. IEEE Computer Society, 387–398.

[29] Malene Søholm, Sean Chester, and Ira Assent. 2016. Maximum coverage representative skyline. In *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016*. OpenProceedings.org, 702–703.

[30] Mohamed A. Soliman, Ihab F. Ilyas, Davide Martinenghi, and Marco Tagliasacchi. 2011. Ranking with Uncertain Scoring Functions: Semantics and Sensitivity Measures. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data* (Athens, Greece) *(SIGMOD '11)*. Association for Computing Machinery, New York, NY, USA, 805–816.

[31] Bo Tang, Kyriakos Mouratidis, and Mingji Han. 2021. On M-Impact Regions and Standing Top-k Influence Problems. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 1784–1796.

[32] Yufei Tao, Ling Ding, Xuemin Lin, and Jian Pei. 2009. Distance-Based Representative Skyline. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*. IEEE Computer Society, 892–903.

[33] Weicheng Wang, Raymond Chi-Wing Wong, and Min Xie. 2021. Interactive Search for One of the Top-k. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 1920–1932.

[34] Yanhao Wang, Michael Mathioudakis, Yuchen Li, and Kian-Lee Tan. 2021. Minimum Coresets for Maxima Representation of Multidimensional Data. In *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (Virtual Event, China) *(PODS'21)*. Association for Computing Machinery, New York, NY, USA, 138–152.

[35] Xingxing Xiao and Jianzhong Li. 2022. Rank-Regret Minimization. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 1848–1860.

[36] Xingxing Xiao and Jianzhong Li. 2023. rkHit: Representative Query with Uncertain Preference. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–26.

[37] Min Xie, Raymond Chi-Wing Wong, Peng Peng, and Vassilis J. Tsotras. 2020. Being Happy with the Least: Achieving α-happiness with Minimum Number of Tuples. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 1009–1020.

[38] Min Xie, Raymond Chi-Wing Wong, and Ashwin Lall. 2019. Strongly Truthful Interactive Regret Minimization. In *Proceedings of the 2019 International Conference on Management of Data* (Amsterdam, Netherlands) *(SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 281–298.

[39] Min Xie, Raymond Chi-Wing Wong, Jian Li, Cheng Long, and Ashwin Lall. 2018. Efficient k-regret query algorithm with restriction-free bound for any dimensionality. In *Proceedings of the 2018 international conference on management of data*. 959–974.

[40] Milan Zeleny. 1974. Linear Multiobjective Programming. *Lecture Notes in Economics and Mathematical Systems* (1974).