2025 IEEE 41st International Conference on Data Engineering (ICDE) | 979-8-3315-3603-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICDE65448.2025.00111

# Computing Shapley Values in Preference Queries

Jiayao Zhang*†, Chirong Zhang*‡, Jian Pei‡, Xuan Luo§, Jianliang Xu¶, Jinfei Liu‖†
†Zhejiang University, {jiayaozhang, jinfeiliu}@zju.edu.cn
‡Simon Fraser University, {chirong_zhang, jpei}@sfu.ca
§York University, xuanluo@yorku.ca
¶Hong Kong Baptist University, xujl@comp.hkbu.edu.hk

***bstract*—This paper tackles the novel problem of computing Shapley values when multiple data owners collaborate to answer preference queries. Despite extensive existing research on preference queries and Shapley value computation separately, the evaluation of data owners' contributions to cooperatively answering such queries has not been systematically explored. To address this gap, we first establish that, for a linear preference utility function with one data point per owner, the Shapley value can be computed in polynomial time. This finding is applicable to attribute weight spaces that are subsets of a simplex and represent various linear preference utility functions. For scenarios involving multiple data points per owner, we observe that only the locally optimal points from each data owner can make non-zero marginal contributions. Thus, we partition the attribute weight space into a polynomial number of subsets, ensuring that in each subset, only one data point per owner needs to be considered. Experimental results on real irbnb Listing data and synthetic data sets validate the effectiveness and efficiency of our algorithms, which significantly outperform baseline methods.**

***Index Terms*—Preference query, Data economy, Shapley value**

## I. Introduction

In the era of I and data science, many large-scale services only become feasible when data sources owned by different parties are utilized collaboratively. The success and development of a platform heavily rely on the participation of numerous providers, a phenomenon known as network effects in economics [21], [65], [74]. key challenge in building such cooperative service platforms, enabled by data collaboration, is the fair allocation of revenues to data contributors according to their contributions. This is crucial not only for encouraging participation but also for fostering a healthy ecosystem for collaborative innovation.

**Example 1.** Consider a B&B platform that connects many travelers with various hosts and their properties. One host may list multiple properties on the platform. traveler may search the platform and retrieve a ranked list of accommodations according to the traveler's preference, such as a weighted sum on attributes like price, location, and rating.

Table I shows three sets of accommodations $\{p_{1,1}, p_{1,2}, p_{1,3}\}$, $\{p_{2,1}, p_{2,2}\}$, and $\{p_{3,1}, p_{3,2}, p_{3,3}\}$, which are provided by hosts $D_1$, $D_2$, and $D_3$, respectively. Each accommodation has three attributes: price, location, and rating. For simplicity

*Joint first authors who contributed equally to this work.
‖Corresponding author.

T BLE I: ccommodations Contributed by 3 Hosts.

| ccommodation | | Price | Location | Rating | Score |
|---|---|---|---|---|---|
| | | | | | $\langle 0\ 2, 0\ 8, 0\rangle$ |
| $D_1$ | $p_{1,1}$ | 5 | 9 | 6 | 8.2 |
| | $p_{1,2}$ | 9 | 5 | 6 | 5.8 |
| | $p_{1,3}$ | 1 | 2 | 3 | 1.8 |
| $D_2$ | $p_{2,1}$ | 6 | 5 | 7 | 5.2 |
| | $p_{2,2}$ | 2 | 7 | 7 | 6.0 |
| $D_3$ | $p_{3,1}$ | 8 | 7 | 8 | 7.2 |
| | $p_{3,2}$ | 7 | 3 | 4 | 3.8 |
| | $p_{3,3}$ | 4 | 1 | 5 | 1.6 |

in this example, the value of each attribute ranges from 0 to 10, with 10 representing the highest level. traveler's preference for price, location, and rating can be specified by an attribute weight vector, such as $\langle 0.2, 0.5, 0.3\rangle$. The preference score of an accommodation is then calculated by $0.2 \times price + 0.5 \times location + 0.3 \times rating$.

The cost paid by a user of the B&B platform consists of two parts: the accommodation cost charged by the selected host and a service fee levied by the platform for offering a wide range of choices. question then arises: how should we allocate the revenue generated from the service charge among the hosts listing their properties on the platform?

Having many active hosts brings significant benefits and revenues to the platform. s the number of hosts increases and the quality of accommodations improves, the B&B platform can offer richer choices to cater to diverse preferences, draw in a larger user base, and build up a good reputation. The key to expanding a strong network of hosts for platforms is to incentivize the hosts proportionate to their contributions to satisfying travelers' search queries [33], [74]. Now, the challenge is how the platform can calculate a fair allocation of rewards to the hosts, which is the focus of our study.

While only a few hosts may achieve the highest score, the participation of all hosts determines the overall efficiency of the platform. Those who do not attain top ranks still play a critical role in shaping the actions of the leading hosts. Without sufficient participation from these hosts, dominant hosts can exploit their position to charge higher prices, offer decreased quality, and harm overall welfare. The presence of diverse hosts acts as a check on this behavior, fostering a healthier marketplace. For example, without hosts $D_2$ and $D_3$, host $D_1$ could raise prices or lower ratings significantly, potentially dropping its score to zero. Therefore, an effective design of rewards accounts for the contributions of all participants, even those who are not top performers, as they shape and drive the motivation and behavior of each other. □

In this paper, we investigate scenarios in which multiple parties provide services or data sources on a platform to collaboratively address a large number of preference queries. Given their essential contributions to the development of the platform ecosystem, we are interested in how the platform can allocate rewards to service providers fairly and constructively.

lthough preference queries have been extensively explored in the literature and have found many applications [19], [20], [42], [45], [62], [78], the issue of fair reward allocation for collaboratively answering preference queries involving multiple data owners remains unaddressed by prior research. The marginal revenue productivity theory of wages suggests that data owners should be compensated based on their marginal productivity, not their total productivity, to maximize profits [15], [29], [70]. This foundational principle is encapsulated by the Shapley value [66], which considers the marginal contribution by the data owner across all possible coalitions of data owners. The Shapley value is a unique metric for collective utility allocation that complies with all four desired fairness properties: allocation efficiency, symmetry, zero element, and additivity [66]. It has a great range of applications, such as in Bittensor, where Shapley values are utilized to determine and reward the contribution of each node in achieving consensus in the network [6], and in Google ds Data Hub, where Shapley values are utilized to model the contribution of advertising channels and touchpoints toward a conversion [32].

In the context of preference queries, we define the utility function of a coalition as the highest preference score achieved by the coalition [63], taking into account all possible user preferences represented within an attribute weight space, which can be obtained through surveys and statistical analysis conducted by the platform. This space captures the potential range of weights that users may assign to different attributes and provides a comprehensive model of user preferences. By leveraging the Shapley value based on the attribute weight space, it is ensured that compensation is fair and reflective of each data owner's contribution.

Computing exact Shapley values generally requires evaluating the utilities of an exponential number of coalitions due to its combinatoric nature, which is computationally prohibitive. To compute Shapley values efficiently in preference queries, we find that in the simple case where each data owner possesses only one data point, given an attribute weight vector, the number of utility values achieved by all possible coalitions is determined by the number of data owners. This observation obviates the need to enumerate an exponential number of coalitions. By sorting the data points according to their preference scores and examining a limited number of utilities, we can compute the exact Shapley values for a specified attribute weight vector in polynomial time.

We extend the result to an attribute weight space consisting of an infinite number of attribute weight vectors, each corresponding to a unique cooperative game. It is challenging to identify locally optimal points, partition and merge the attribute weight space into subspaces, where the ranking of data points remains consistent. To tackle this complexity, we exploit the arrangement structure to partition the space with lines or hyperplanes, such that each partitioned subspace shares the same polynomial expression of the Shapley value. Furthermore, to overcome the computational challenges posed by the curse of dimensionality, we propose a Monte Carlo simulation approach, where each sample of attribute weight vectors can be computed efficiently based on our earlier observation.

In general scenarios where each data owner possesses multiple data points, we first apply the concept of convex skyline points [28] to reduce the size of the data that needs to be processed. The convex skyline identifies non-dominated solutions in multi-objective optimization problems. Points not situated on the local convex skyline are excluded, as they cannot contribute to any coalition. Building on this, we note that only the locally optimal points from each data owner can make non-zero marginal contributions. We then partition the attribute weight space by analyzing the local convex skyline points and merging the resultant subspaces. Consequently, the problem of computing Shapley values for each data owner with multiple points is reduced to the problem of computing Shapley values for each data owner with a single locally optimal point.

To evaluate the effectiveness and efficiency of our approaches, we perform extensive experiments on a series of synthetic data sets and present a case study on the real irbnb Listing data. The experimental results show that our approaches not only ensure the accuracy of estimated Shapley values but also achieve efficiency improvements in computation when compared to baseline methods, including coalition enumeration method and permutation sampling method.

The rest of the paper is organized as follows. Section II reviews the related work. Section III formulates the problem of computing Shapley values of data owners for answering preference queries. We develop algorithms for computing Shapley values in a simple case where each data owner has one data point in Section IV and extend the result to a general case where each data owner may have multiple data points in Section V. We report the experimental results in Section VI. Finally, we conclude the paper in Section VII.

## II. Rel ted Work

Our work is related to the literature on Shapley value computation and preference queries.

### . *Shapley Value Computation*

Shapley value [66] has found many applications, such as cost sharing [47], [64], revenue allocation [55], [69], [79], query answering [18], [53], data/feature selection [22], [25], and data pricing [1], [48]. Computing the exact Shapley values was proved to be #P-hard for voting game [17]. Thus, significant research efforts have been dedicated to approximation or computation in various special cases [36], [38], [43], [75].

Several sampling techniques [8], [10], [11], [56], [58], [80] were developed to approximate Shapley values in generic games. Castro et al. [11] presented a Monte Carlo permutation sampling method that estimates Shapley values as the expectation of marginal contributions. Mitchell et al [58] improved the permutation sampling via Quasi Monte Carlo techniques. Maleki et al. [56] provided a stratified sampling algorithm that relies on an assumption about the range of utilities and gives the sample size of each stratum based on the Hoeffding bound [30], which was further improved by Castro et al. [10]. Burgess and Chapman [8] provided a stratified sampling algorithm that assumes the sample variance and sequentially chooses strata to sample based on an empirical bound. Most recently, Zhang et al. [80] proposed to sample complementary contributions that can be reused to estimate Shapley values for all players compared with marginal contributions.

Shapley value has been used to quantify the contributions of data points or features towards training models, and several special techniques were proposed in the context of machine learning [1], [25], [37], [84]. Lundberg and Lee [54] computed Shapley values for data features to explain a model. Ghorbani and Zou [25] computed Shapley values for data points to estimate the value of each data point towards a model. Jia et al. [37] focused on one family of models relying on $k$-nearest neighbors and showed that exact Shapley values of data points towards training a $k$-nn model can be computed in polynomial time. Farchi et al. [22] defined the utility function based on the number of misclassified data and used the exact Shapley values to rank validation data slices. Pastor et al. [61] analyzed classifier behavior and used Shapley values to measure the contribution of items to pattern divergence.

Shapley value has also been applied in the context of databases [9], [18], [53], [55]. Deutch et al. [18] provided several algorithms for computing Shapley values of endogenous facts in Boolean query answering. Luo et al. [55] enabled the efficient computation of the exact Shapley values for revenue sharing among multiple data owners under independent utility assumption.

This paper studies a new problem of Shapley value computation that has not been touched in literature. The study closest to ours is by Cabello and Chan [9], which studied the problem of computing Shapley values in the plane and provided polynomial-time algorithms for various geometric shapes. critical difference is that Cabello and Chan [9] focused on points in 2-dimensional space and defined the utility of a coalition by its area or perimeter, which is dramatically different from the utility addressed in our study.

### B. Preference Queries

Preference querying refers to the process of identifying a subset of data points from a data set that satisfies certain preference criteria, which can be based on a combination of multiple attributes. Top-$k$ queries [12], [31], [34], [71], [76], [77] and skyline queries [4], [39]–[42], [57] are two popular types of preference queries.

In a top-$k$ query, a user specifies a scoring function and retrieves $k$ data points with the highest scores in the data set. The class of linear scoring functions are most common, which assign a score to a data point by calculating a weighted sum of its attributes based on a given attribute weight vector [12], [31]. Ilyas et al. [34] presented a comprehensive survey on top-$k$ queries. Xin et al. [77] discussed ranking fragments for efficiently answering top-$k$ queries with multi-dimensional selections. Tao et al. [73] tackled the problem of subspace skyline/top-$k$ queries by converting multidimensional data into one-dimensional values and indexing them using a B-tree. To protect data security, Zhang et al. [81] proposed an approximate $k$-nearest neighbor query over a spatial database. He and Lo [27] introduced the concept of "why-not" questions, which aim to explain why expected points are absent from query results. These questions were further studied [13], [24]. Tang et al. [72] studied the problem of finding a preference space that allows given data records to be ranked in the top $k$. hmed et al. [2] turned to the reverse spatial top-$k$ queries, which find the neighborhoods of a given size where a given term is top-$k$ frequent.

The problem of computing skyline (Maxima) is a fundamental problem in the field of databases and has received extensive attention. Skyline computation was first studied in computational geometry [42]. Börzsönyi et al. [7] introduced the skyline operator in database. To facilitate skyline query answering, Liu et al. [52] proposed skyline diagram. To protect data privacy and query privacy, secure skyline query was studied [51], [82]. The number of skyline points in a large multidimensional data set can be exponentially high. Lin et al. [46] studied the problem of selecting $k$ skyline points to maximize the number of points dominated by at least one of the $k$ skyline points. Bai et al. [5] focused on selecting the $k$ skyline points over streaming data to represent the entire data set. Zhang et al. [83] showed a cone dominance definition which can control the size of returned points. Liu et al. [50] generalized 1NN and skyline queries and provided a more customizable query solution: eclipse.

lthough there have been extensive studies on preference queries, most of them focused on efficient query answering. To the best of our knowledge, the problem of valuing contributions to cooperation in answering preference queries has not been touched in literature, which is the topic of this paper.

## III. PROBLEM FORMUL TION

Consider a set of $n$ data owners $\mathcal{D} = \{D_1, \ldots, D_n\}$, where $D_i = \{p_{i,1}, \ldots, p_{i,n_i}\}$ ($1 \leq i \leq n$ and $n_i \geq 1$) denotes a data owner as well as the set of data points owned by the owner. Here, we assume that all points are in a $d$-dimensional non-negative space of real numbers $\mathcal{R}^{+^d}$ ($d \geq 2$). **coalition** is a subset of data owners $S \subseteq \mathcal{D}$ that cooperate to form a data set $D(S) = \cup_{D_i \in S, 1 \leq i \leq n} D_i$. We assume a **utility function** $\mathcal{U} : 2^{\mathcal{D}} \rightarrow \mathcal{R}$ that evaluates the utility of the coalition, such that $\mathcal{U}(\emptyset) = 0$. Pair $(\mathcal{D},\mathcal{U})$ is a (**characteristic**) **game**, which is the most widely studied model in cooperative games.

**reward allocation function** $\psi : \mathcal{D} \to \mathcal{R}^+$ assigns a non-negative reward to each data owner. For the sake of simplicity, we often write $\psi_i = \psi(D_i)$ $(1 \leq i \leq n)$. Lloyd Shapley [66] laid out the fundamental requirements of fairness in reward allocation in cooperative games, including allocation efficiency, symmetry, zero element, and additivity.

**Shapley value** [66] is the expectation of marginal contributions by $D_i$ in all possible coalitions, that is,

$$\mathcal{SV}(D_i) = \frac{1}{n} \sum_{S \subseteq \mathcal{D} \setminus \{D_i\}} \frac{\mathcal{U}(S \cup \{D_i\}) \quad \mathcal{U}(S)}{\binom{n \ \ 1}{|S|})}. \tag{1}$$

We write $\mathcal{SV}(D_i)$ as $\mathcal{SV}_i$ for short when it is clear from the context. Shapley value is the only measure that satisfies all four fundamental requirements on fairness [66].

In this paper, we consider the utility of data in answering preference queries. For a point $p \in \mathcal{R}^{+d}$, we write $p[j]$ $(1 \leq j \leq d)$ the projection of $p$ on the $j$-th dimension. Given an attribute weight vector $\mathbf{w} = \langle \mathbf{w}[1], \ldots, \mathbf{w}[d] \rangle$ in a simplex, where $\mathbf{w}[j] \geq 0$ $(1 \leq j \leq d)$ and $\sum_{j=1}^{d} \mathbf{w}[j] = 1$, the **utility** of a point $p$ is $Q(p; \mathbf{w}) = \sum_{i=1}^{d} p[i]\mathbf{w}[i]$. We write $Q(p; \mathbf{w})$ as $Q(p)$ for short when $\mathbf{w}$ is clear from the context.

When a set of points is present, assuming, without loss of generality, that on a largervalue each attribute is more preferred, the point with the highest utility is considered the most preferred with respect to a given weight vector. user seeks a solution from the data owners that achieves the user's preference as much as possible. The gain of the user is the maximum value of any point in query results [63]. The data owners cooperate to contribute their data to provide the user with the most preferred solution.

Given an attribute weight vector $\mathbf{w}$, the utility of coalition $S \subseteq \mathcal{D}$ is $\mathcal{U}\ \ (S) = \max_{p \in D(S)}\{Q(p; \mathbf{w})\}$. n **attribute weight space** is a set of attribute weight vectors, which is a subset of a simplex. Given an attribute weight space $\mathcal{W} = \{\mathbf{w} | \mathbf{w}[i] \geq 0, 1 \leq i \leq d, \sum_{i=1}^{d} \mathbf{w}[i] = 1\}$, the **utility of coalition** $S \subseteq \mathcal{D}$ is defined as

$$\mathcal{U}_{\mathcal{W}}(S) = \int_{\ \in \mathcal{W}} \max_{p \in D(S)}\{Q(p; \mathbf{w})\} d\mathbf{w}. \tag{2}$$

The utility $\mathcal{U}_{\mathcal{W}}(S)$ evaluates the highest utility achievable by the data points contributed by the data owners in a coalition $S$, taking into account all possible user preferences represented by the attribute weight vectors in $\mathcal{W}$.

In this paper, without loss of generality, we focus on the special case where an attribute weight space is continuous. In general, our discussion holds on any user-specified attribute weight space that is Lebesgue-measurable.

*a) Problem Definition:* Given a set of $n$ data owners $\mathcal{D} = \{D_1, \ldots, D_n\}$ in a $d$-dimensional space and an attribute weight space $\mathcal{W}$, the problem of **computing Shapley values in preference queries** is to compute the Shapley value $\mathcal{SV}_i^{\mathcal{W}}$ $(1 \leq i \leq n)$ for all data owners $D_i$ in the characteristic game $(\mathcal{D}, \mathcal{U}_{\mathcal{W}})$.

## IV. Simple C se: One Point per Owner

Let us start with a simple case where each data owner has only one point, that is, $D_i = \{p_i\}$ $(1 \leq i \leq n)$. We present a polynomial-time method to compute Shapley values on an attribute weight vector and extend the result to an attribute weight space. The general case is handled in the next section.

### . Computing $\mathcal{SV}_i$ for an ttribute Weight Vector $\mathbf{w}$

Our first result is that the exact Shapley value $\mathcal{SV}_i$ in this simple case can be computed in polynomial time.

**Lemma 1.** *Given a set of $n$ data owners $\mathcal{D} = \{D_1, \ldots, D_n\}$ such that $D_i = \{p_i\}$ $(1 \leq i \leq n)$ and an attribute weight vector $\mathbf{w}$, let $L$ be the list of $p_i$'s in the ascending order of utility $Q(p_i; \mathbf{w})$, $L\ \ (j)$ $(1 \leq j \leq n)$ the $j$-th point in $L$, and $I\ \ (i)$ the position index of $p_i$ in $L$, that is, $L\ \ (I\ \ (i)) = p_i$. Set $Q(L\ \ (0)) = 0$. The Shapley value of $D_i$ is*

$$\begin{aligned}\mathcal{SV}_i &= \sum_{j=1}^{I\ \ (i)} \frac{Q(L\ \ (j)) \quad Q(L\ \ (j \quad 1))}{n \quad j+1} \\ &= \mathcal{SV}_{L\ \ (I\ \ (i)\ \ 1)} + \frac{Q(p_i) \quad Q(L\ \ (I\ \ (i) \quad 1))}{n \quad I\ \ (i)+1}.\end{aligned}$$

*Proof.* The utility of a coalition is the utility of the point that has the largest position in the sorted list $L$ . Therefore, the marginal contribution of $D_i$ with respect to a coalition $S$ is not zero only if the index of $p_i$ is larger than all indexes in the coalition. Thus, according to Equation 1,

$$\mathcal{SV}_i = \sum_{\substack{S \subseteq \mathcal{D} \setminus \{D_i\} \\ I\ \ (i) > \max_{D_j \in S}\{I\ \ (j)\}}} \frac{\mathcal{U}\ \ S \cup \{D_i\})}{n\ \binom{n\ \ 1}{|S|})} \qquad \sum_{\substack{S \subseteq \mathcal{D} \setminus \{D_i\} \\ I\ \ (i) > \max_{D_j \in S}\{I\ \ (j)\}}} \frac{\mathcal{U}\ \ S)}{n\ \binom{n\ \ 1}{|S|})}$$

Since there are $I\ \ (i) \quad 1$ points with a utility smaller than or equal to $Q(p_i)$, there are $\sum_{t=0}^{I\ \ (i)\ \ 1} \binom{I\ \ (i)\ \ 1}{t})$ coalitions with utility $Q(p_i)$. Thus, we have

$$\begin{aligned}\mathcal{SV}_i &= \sum_{t=0}^{I\ \ (i)\ \ 1} \frac{\binom{I\ \ (i)\ \ 1}{t})}{n\ \binom{n\ \ 1}{t})} Q\ \ p_i) \qquad \sum_{j=1}^{I\ \ (i)\ \ 1} \sum_{t=0}^{I\ \ (j)\ \ 1} \frac{\binom{I\ \ (j)\ \ 1}{t})}{n\ \binom{n\ \ 1}{t+1})} Q\ \ p_j) \\ &= \sum_{j=1}^{I\ \ (i)} \frac{Q\ \ L\ \ \ j)) \quad Q\ \ L\ \ \ j \quad 1))}{n \quad j+1} \\ &= \mathcal{SV}_{L\ \ (I\ \ (i)\ \ 1)} + \frac{Q\ \ p_i) \quad Q\ \ L\ \ \ I\ \ \ i) \quad 1))}{n \quad I\ \ \ i)+1} \qquad \square\end{aligned}$$

Using Lemma 1, we can compute $\mathcal{SV}_i$ $(1 \leq i \leq n)$ in $O(n \log n + nd)$ time by computing and sorting the utility of the $n$ points and iterating over them.

### B. Computing $\mathcal{SV}_i^{\mathcal{W}}$ for an ttribute Weight Space $\mathcal{W}$

Given an attribute weight space $\mathcal{W}$, how can we calculate the Shapley value $\mathcal{SV}_i^{\mathcal{W}}$ for data owner $D_i$ in characteristic game $(\mathcal{D},\mathcal{U}_{\mathcal{W}})$ based on the calculation of $\mathcal{SV}_i$ $(1 \leq i \leq n, \mathbf{w} \in \mathcal{W})$? ccording to the definition of Shapley value (Equation 1) and the utility function (Equation 2), we have

$$\begin{aligned}\mathcal{SV}_i^{\mathcal{W}} &= \frac{1}{n} \sum_{S \subseteq \mathcal{D} \setminus \{D_i\}} \frac{\int_{\ \in \mathcal{W}} \mathcal{U}\ \ \ S \cup \{D_i\})d \qquad \int_{\ \in \mathcal{W}} \mathcal{U}\ \ \ S)d}{\binom{n\ \ 1}{|S|})} \\ &= \int_{\ \in \mathcal{W}} \frac{1}{n} \sum_{S \subseteq \mathcal{D} \setminus \{D_i\}} \frac{\mathcal{U}\ \ \ S \cup \{D_i\}) \quad \mathcal{U}\ \ \ S)}{\binom{n\ \ 1}{|S|})} d \quad = \int_{\ \in \mathcal{W}} \mathcal{SV}_i\ \ d\end{aligned} \tag{3}$$

To solve the integration, we propose two approaches. The first partitions the space through arrangement and calculates the integral of a continuous integrand. The second directly calculates the integral of the Shapley value without arrangement.

*1) Integration with  rrangement:* Given two attribute weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$, let $L$  and $L_{\ \ 2}$ be the lists of all data points sorted in ascending order according to their values in $Q(p_i; \mathbf{w}_1)$ and $Q(p_i; \mathbf{w}_2)$, respectively.  ccording to Lemma 1, as long as $L \quad = L_{\ \ 2}$, the expression for $\mathcal{SV}_i$ is the same as that for $\mathcal{SV}_i^{\ \ 2}$ $(1 \leq i \leq n)$. This observation motivates us to partition $\mathcal{W}$ based on the order of points in $L$ .

Denote by $\mathcal{A}(\mathcal{W}) = \{\mathcal{W}_1, \dots, \mathcal{W}_a\}$ a partitioning of $\mathcal{W}$, such that in each subspace $\mathcal{W}_j$ $(1 \leq j \leq a)$, for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{W}_j$, $L \quad = L_{\ \ 2}$. Let $L_{\mathcal{W}_j}$ be the list of $p_i$'s in the $Q(p_i; \mathbf{w})$ $(\mathbf{w} \in \mathcal{W}_j)$ ascending order, $L_{\mathcal{W}_j}(i)$ $(1 \leq i \leq n)$ the $i$-th point in $L_{\mathcal{W}_j}$, and $I_{\mathcal{W}_j}(i)$ the position index of $p_i$ in $L_{\mathcal{W}_j}$. Set $L_{\mathcal{W}_j}(0) = (0, \dots, 0)$.  ccording to Equation 3, we have

$$\mathcal{SV}_i^{\mathcal{W}} = \sum_{j=1}^{a} \int_{\ \in \mathcal{W}_j} \mathcal{SV}_i \ d \quad = \sum_{j=1}^{a} \ \mathcal{SV}_{L_{\mathcal{W}_j}(I_{\mathcal{W}_j}(i) \quad 1)}^{\mathcal{W}_j} + \sum_{t=1}^{d} \frac{p_i[t] \quad L_{\mathcal{W}_j} \ I_{\mathcal{W}_j} \ i) \quad 1)[t]}{n \quad I_{\mathcal{W}_j} \ i) + 1} \int_{\ \in \mathcal{W}_j} \quad [t] d \ \Big). \tag{4}$$

The problem becomes how to obtain a partitioning $\mathcal{A}(\mathcal{W})$. Since there are only $n$ data points, in total there are at most $n$  different permutations of the points and thus $n$  different possible orders of $L$ . Thus, there exists at least an $\mathcal{A}(\mathcal{W})$ such that $|\mathcal{A}(\mathcal{W})| \leq n$ .

*a) Partitioning in a 2-dimensional Data Space:* Let us first consider the scenario where all the points are in 2-dimensional space, i.e., $p_i \in \mathcal{R}^{+2}$ $(1 \leq i \leq n)$.  s illustrated in Figure 1, since each attribute weight vector $\mathbf{w}$ is a simplex, $\mathbf{w}$ must be in the southeast-northwest direction, $\langle 1, 0 \rangle$ and $\langle 0, 1 \rangle$ being two special cases. For an attribute vector space $\mathcal{W} \subseteq \{\mathbf{w} | \mathbf{w}[i] \geq 0, 1 \leq i \leq 2, \sum_{i=1}^{2} \mathbf{w}[i] = 1\}$ and any two different points $p_i$ and $p_j$ $(1 \leq i < j \leq n)$, $\mathcal{W}$ can be divided by vector $\mathbf{v}_{i,j} = \left\langle \frac{p_j[2] \quad p_i[2]}{p_i[1] \quad p_j[1] + p_j[2] \quad p_i[2]}, 1 \quad \frac{p_j[2] \quad p_i[2]}{p_i[1] \quad p_j[1] + p_j[2] \quad p_i[2]} \right\rangle$ into two subspaces $\mathcal{W}_{i>j} = \{\mathbf{w} \in \mathcal{W} | Q(p_i; \mathbf{w}) > Q(p_j; \mathbf{w})\}$ and $\mathcal{W}_{i \leq j} = \{\mathbf{w} \in \mathcal{W} | Q(p_i; \mathbf{w}) \leq Q(p_j; \mathbf{w})\}$.

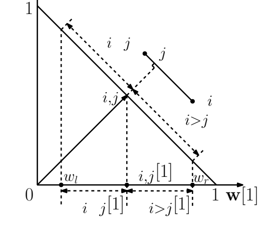
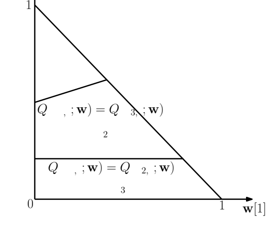


Fig. 1: Partitioning in 2D.



$\mathbf{w}[2]$
1
$Q \quad , ;\mathbf{w}) = Q \quad 3, ;\mathbf{w})$
2
$Q \quad , ;\mathbf{w}) = Q \quad 2, ;\mathbf{w})$
3
0
1
$\mathbf{w}[1]$

Fig. 2: Hyperplane  rrangement.

Since $\mathbf{w}[1] + \mathbf{w}[2] = 1$ for every attribute weight vector $\mathbf{w}$, every attribute weight vector $\mathbf{w}$ is uniquely determined by $\mathbf{w}[1]$. Correspondingly, an attribute vector space $\mathcal{W}$ can be represented as a one-dimensional space $\{\mathbf{w}[1] \mid \mathbf{w} \in \mathcal{W}\}$. In this representation, $\mathcal{W}_{i>j}[1]$ is a subset of the interval $(\mathbf{v}_{i,j}[1], 1] \cap [w_l, w_r]$, where $w_l = \min\{\mathbf{w}[1] \mid \mathbf{w} \in \mathcal{W}\}$ and $w_r = \max\{\mathbf{w}[1] \mid \mathbf{w} \in \mathcal{W}\}$.

 baseline method is to enumerate all pairs of points and obtain the corresponding vectors $\mathbf{v}_{i,j}$ $(1 \leq i < j \leq n)$. Then, we sort $\mathbf{v}_{i,j}[1]$ to obtain all intervals, and then sort the points based on their utility in each interval to have the sorted lists. This method takes $O(n^2 \log n)$ time.

To improve efficiency, we transform our problem from the primal space to the dual space by duality transform [16]. For a point $p = (p[1], p[2])$, its dual line is $x_2 = p[1]x_1 \quad p[2]$. In the dual space, the $x_1$-coordinate corresponds to the weight ratio $\quad \frac{[1]}{[2]}$. The proximity of a line to the $x_1$-axis corresponds to the utility of the point. The sorted list of points in utility ascending order changes only when the corresponding lines of these points intersect in the dual space.

**Example 2.** *In Table I, suppose that each data owner $D_i$ has only data point $p_{i,1}$ $(1 \leq i \leq 3)$ and an attribute weight space $\mathcal{W} = \{\mathbf{w} \mid \mathbf{w}[i] \geq 0, 1 \leq i \leq 2, \sum_{i=1}^{2} \mathbf{w}[i] = 1\}$, that is, we consider only first two attributes in Table I. The dual lines of $p_{1,1}$, $p_{2,1}$, and $p_{3,1}$ are $x_2 = 5x_1 \quad 9$, $x_2 = 6x_1 \quad 5$, and $x_2 = 8x_1 \quad 7$, respectively. Given $\mathcal{W}[1] = [0, 1]$, the dual interval is $( \quad \infty, 0]$, since $\quad \frac{w[1]}{w[2]} = 0$ when $w[1] = 0$, and $\lim_{w[1] \to 1} \quad \frac{w[1]}{w[2]} = \quad \infty$. The two intersections, $\langle \quad \frac{2}{3}, \quad \frac{37}{3} \rangle$ and $\langle \quad 4, \quad 29 \rangle$, partition $\mathcal{W}[1]$ into three intervals $[0, \frac{2}{5})$, $[\frac{2}{5}, \frac{4}{5})$, and $[\frac{4}{5}, 1]$. These intervals correspond to three sorted lists, $\langle p_{2,1}, p_{3,1}, p_{1,1} \rangle$, $\langle p_{2,1}, p_{1,1}, p_{3,1} \rangle$, and $\langle p_{1,1}, p_{2,1}, p_{3,1} \rangle$.* □

We can use a sweep-line algorithm to find the intersections and partitioning. The procedure is presented in  lgorithm 1. Given an interval $\mathcal{W}[1] = [w_l, w_r]$, we sweep a vertical line from $x_1 = \frac{w_r}{w_r \quad 1}$ to $x_1 = \frac{w_l}{w_l \quad 1}$. We first insert the dual line of $p_i$ $(1 \leq i \leq n)$ into a list of lines $\mathcal{T}$ (Lines 2-3). These lines in $\mathcal{T}$ are then sorted in the $Q(p_i; w_r)$ ascending order (Line 4). $\mathcal{T}$ is the sorted list of points in the utility ascending order under the current sweep line. The $x_1$-axis $e$ of the new intersection between line $\mathcal{T}(i)$ and line $\mathcal{T}(i \quad 1)$ $(2 \leq i \leq n)$ is inserted into a min-priority queue of events $\mathcal{Q}$ (Lines 5-6). We get the next $e$ by dequeuing the first element from $\mathcal{Q}$ (Line 9).  n interval $[\frac{e}{e \quad 1}, e']$ with sorted list $\mathcal{T}$ is inserted into the partitioning $\mathcal{A}$ and $e'$ is updated to $\frac{e}{e \quad 1}$ (Line 10). Then, we process event point $e$ following the standard sweep line algorithm [16]. It includes updating $\mathcal{T}$ and inserting new intersections into $\mathcal{Q}$ (Line 11). Finally, we insert the last interval $[w_l, e']$ with sorted list $\mathcal{T}$ into $\mathcal{A}$ (Line 12). It is easy to see that the time complexity of  lgorithm 1 is $O((n+s) \log n)$, where $s \leq \frac{n(n \quad 1)}{2}$ is the number of intersections.

*b) Integration in a 2-dimensional Data Space:* With $\mathcal{A}(\mathcal{W})$, we calculate the integral in each subspace, which is denoted by an interval, and aggregate the results to obtain the Shapley value in the entire space. The details are described in  lgorithm 2. We first partition the attribute weight space $\mathcal{W}$ based on points $p_1, \dots, p_n$ using  lgorithm 1. The integral $\int_{\ \in \mathcal{W}j} \mathbf{w}[t] d\mathbf{w}$ $(1 \leq t \leq 2, 1 \leq j \leq a)$ can be computed exactly through analytical integration (Line 4). Concretely, as $\mathcal{W}_j[1] = [w_l, w_r]$, $\int_{\ \in \mathcal{W}_j} \mathbf{w}[1] d\mathbf{w} =$

**lgorithm 1:** Line arrangement.

```
input  : points $p_1, \ldots, p_n$; an interval $\mathcal{W}[1] = [w_l, w_r]$;
output: $\mathcal{A}\ \mathcal{W})$;
1  initialize an empty partitioning $\mathcal{A}$, an empty list of lines $\mathcal{T}$,
     and an empty min-priority queue of event $\mathcal{Q}$;
2  for $i = 1$ to $n$ do
3      insert line $p_i : x_2 = p_i[1]x_1 \quad p_i[2]$ into $\mathcal{T}$;
4  sort lines in $\mathcal{T}$ in utility $Q\ p_i; w_r)$ ascending order;
5  for $i = 2$ to $n$ do
6      find the $x_1$-axis $e$ of the new intersection of $\mathcal{T}\ i \quad 1)$
         and $\mathcal{T}\ i)$ between $[w_l, w_r]$; insert $e$ into $\mathcal{Q}$;
7  $e' = w_r$;
8  while $\mathcal{Q} \neq \emptyset$ do
9      dequeue the first element from $\mathcal{Q}$ to get the next $e$;
10     insert $[\frac{e}{e\ 1}, e']$ with $\mathcal{T}$ into $\mathcal{A}$; $e' = \frac{e}{e\ 1}$;
11     process $e$ following the standard sweep line algorithm,
         including updating $\mathcal{T}$ and $\mathcal{Q}$;
12 insert $[w_l, e']$ with $\mathcal{T}$ into $\mathcal{A}$;
13 return $\mathcal{A}$.
```

**lgorithm 2:** Computing $\mathcal{SV}_i^{\mathcal{W}}$ with arrangement.

```
input  : points $p_1, \ldots, p_n$; an attribute weight space $\mathcal{W}$;
output: Shapley value $\mathcal{SV}_i^{\mathcal{W}}$ for each point $p_i \quad 1 \leq i \leq n)$;
1  compute the partitioning $\mathcal{A}\ \mathcal{W})$ with respect to $p_1, \ldots, p_n$;
2  $\mathcal{SV}_i^{\mathcal{W}} = 0\ (0 \leq i \leq n)$;
3  for $\mathcal{W}_j \in \mathcal{A}\ \mathcal{W})$ do
4      compute the integral $s_t = \int_{\ \in \mathcal{W}_j} \quad [t]d \quad (1 \leq t \leq d)$;
5      for $i = 1$ to $n$ do
6          $\mathcal{SV}_{L_{\mathcal{W}_j}(i)}^{\mathcal{W}_j} = \mathcal{SV}_{L_{\mathcal{W}_j}(i\ \ 1)}^{\mathcal{W}_j}$;
7          for $t = 1$ to $d$ do
8              $\mathcal{SV}_{L_{\mathcal{W}_j}(i)}^{\mathcal{W}_j} += \frac{L_{\mathcal{W}_j}(i)[t] \quad L_{\mathcal{W}_j}(i\ \ 1)[t]}{n \quad i+1} s_t$;
9          $\mathcal{SV}_{L_{\mathcal{W}_j}(i)}^{\mathcal{W}} += \mathcal{SV}_{L_{\mathcal{W}_j}(i)}^{\mathcal{W}_j}$;
10 return $\mathcal{SV}_1^{\mathcal{W}}, \ldots, \mathcal{SV}_n^{\mathcal{W}}$.
```

$\sqrt{2}\int_{w_l}^{w_r} \mathbf{w}[1]d\mathbf{w}[1] = \frac{\sqrt{2}}{2}(w_r^2 \quad w_l^2)$ and $\int_{\ \in \mathcal{W}_j} \mathbf{w}[2]d\mathbf{w} = \sqrt{2}\int_{w_l}^{w_r}(1 \quad \mathbf{w}[1])d\mathbf{w}[1] = \sqrt{2}(w_r \quad w_l) \quad \frac{\sqrt{2}}{2}(w_r^2 \quad w_l^2)$, from [67], the Shapley value then can be computed following the order of points based on Equation 4 (Lines 5-10).

*c) Partitioning in a General Multi-dimensional Data Space:* Now, let us tackle the scenario where all the points are in $d$-dimensional space, i.e., $p_i \in \mathcal{R}^{+d}$ ($1 \leq i \leq n$, $d > 2$). For any two points $p_i$ and $p_j$ ($1 \leq i < j \leq n$), an attribute weight space $\mathcal{W}$ can be divided by hyperplane $\{\mathbf{w}|Q(p_i; \mathbf{w}) = Q(p_j; \mathbf{w})\}$ into two subspaces $\mathcal{W}_{i>j} = \{\mathbf{w} \in \mathcal{W}|Q(p_i; \mathbf{w}) > Q(p_j; \mathbf{w})\}$ and $\mathcal{W}_{i \leq j} = \{\mathbf{w} \in \mathcal{W}|Q(p_i; \mathbf{w}) \leq Q(p_j; \mathbf{w})\}$. Therefore, the hyperplanes formed by all pairs of points divide the space into subspaces.

**Example 3.** *Consider the same setting as Example 2 but an attribute weight space* $\mathcal{W} = \{\mathbf{w}|\mathbf{w}[i] \geq 0, 1 \leq i \leq 3, \sum_{i=1}^{3} \mathbf{w}[i] = 1\}$, *that is, considering all three attributes in Table I. s shown in Figure 2, two hyperplanes* $\{\mathbf{w}|\mathbf{w}[1] \quad 4\mathbf{w}[2] + 2 = 0\}$ *and* $\{\mathbf{w}|5\mathbf{w}[2] \quad 1 = 0\}$ *divide* $\mathcal{W} = \{\mathbf{w}|\mathbf{w}[1] \geq 0, \mathbf{w}[2] \geq 0, \mathbf{w}[1] + \mathbf{w}[2] \leq 1\}$ *into 3 subspaces* $\mathcal{W}_1$, $\mathcal{W}_2$, *and* $\mathcal{W}_3$, *with sorted lists* $\langle p_{2,1}, p_{3,1}, p_{1,1}\rangle$, $\langle p_{2,1}, p_{1,1}, p_{3,1}\rangle$, *and* $\langle p_{1,1}, p_{2,1}, p_{3,1}\rangle$, *respectively.* □

We can employ an output-sensitive algorithm [68] to enumerate the subspaces in the hyperplane arrangement. The detailed procedure is presented in lgorithm 3. It is easy to see that the number of subspaces in lgorithm 3 is $O(n^{2(d\ \ 1)})$.

**lgorithm 3:** Hyperplane arrangement.

```
input  : points $p_1, \ldots, p_n$; an attribute weight space $\mathcal{W}$;
output: $\mathcal{A}\ \mathcal{W})$;
1  initialize an empty hyperplane list $\mathcal{T}$;
2  for $1 \leq i < j \leq n$ do
3      insert hyperplane $Q\ p_i; \quad ) = Q\ p_j; \quad )$ into $\mathcal{T}$;
4  enumerate the subspaces in the arrangement of $\mathcal{T}$ in $\mathcal{W}$;
5  for each subspace do
6      record the sorted list of points in the $Q\ p_i; \quad )$ ascending
         order into $\mathcal{A}$;
7  return $\mathcal{A}$.
```

*d) Integration in a General Multi-dimensional Data Space:* fter getting $\mathcal{A}(\mathcal{W})$, we calculate the integral in each subspace and aggregate the results to obtain the Shapley value in the entire space. The algorithm is similar to the 2-dimensional case outlined in lgorithm 2. The main differences lie in the methods of partitioning and integration. Specifically, we partition the attribute weight space $\mathcal{W}$ based on points $p_1, \ldots, p_n$ using lgorithm 3. To approximate the integral $\int_{\ \in \mathcal{W}j} \mathbf{w}[t]d\mathbf{w}$ ($1 \leq t \leq d$ and $1 \leq j \leq a$), we employ the Monte Carlo integration.

The time complexity of lgorithm 2 is $O(\quad + n^{2(d\ \ 1)}(\beta d + nd))$, where is the time complexity for arrangement and $\beta$ is the time complexity of integration. For a 2-dimensional space, is $O((n + s)\log n)$ and $\beta$ is $O(1)$. For a general multi-dimensional data space, is $O(n^{2d})$ and $\beta$ is $O(\tau)$, where $\tau$ is the number of samples for the Monte Carlo integration.

*2) Integration without rrangement:* The challenge arises in the general case of multi-dimensional data spaces due to the curse of dimensionality. To tackle the problem, we propose an integration approach that does not require arrangement. In this approach, we sample attribute weight vectors and estimate the integral by calculating the average of the Shapley value integrand at these sampled vectors. Notably, each sample can be computed efficiently according to Lemma 1.

The details are given in lgorithm 4. We generate a hyperrectangle $\mathcal{H}$ with a known measure $M_c$ that encloses $\mathcal{W}$ and then randomly sample attribute weight vectors $\mathbf{w}_t$ from $\mathcal{H}$ (Lines 2-4). For each sample $\mathbf{w}_t \in \mathcal{W}$, we calculate and aggregate $\mathcal{SV}_i^{\ \ t}$ ($1 \leq i \leq n$) based on Lemma 1 (Lines 5-8). The measure of $\mathcal{W}$ is estimated based on the ratio of the number of samples falling in $\mathcal{W}$ to the total number of samples. By evaluating Shapley value $\mathcal{SV}_i^{\ \ t}$ on $\tau$ samples, the final estimate of Shapley value $\mathcal{SV}_i^{\mathcal{W}}$ is obtained by averaging $\mathcal{SV}_i^{\ \ t}$ and scaling it by the measure of $\mathcal{W}$ and $\sqrt{d}$ (Lines 9-10). lgorithm 4 is a Monte Carlo method and has the rate of convergence $O(N^{\ \ \frac{}{2}})$, where $N$ is the number of samples [60]. The rate of convergence can be improved to $O(N^{\ \ (\frac{}{2} + \frac{}{d})})$ using control neighbors [44].

**lgorithm 4:** Computing $\mathcal{SV}_i^{\mathcal{W}}$ without arrangement.

```
   input : points $p_1, \ldots, p_n$; an attribute weight space $\mathcal{W}$;
           sample size $\tau > 0$;
   output: Shapley value $\mathcal{SV}_i{}^{\mathcal{W}}$ for each point $p_i \;\; 1 \leq i \leq n)$;
 1 $\mathcal{SV}_i^{\mathcal{W}} = 0 \;\; 1 \leq i \leq n)$;
 2 let $\mathcal{H}$ be a hyperrectangle with a measure $M_c$ enclosing $\mathcal{W}$;
 3 for $t = 1$ to $\tau$ do
 4     let $\;_t$ be a sample in $\mathcal{H}$;
 5     if $\;_t \in \mathcal{W}$ then
 6         for $i = 1$ to $n$ do
 7             compute $\mathcal{SV}_i{}^{\;t}$ $(1 \leq i \leq n)$ based on Lemma 1;
 8             $\mathcal{SV}_i^{\mathcal{W}} += \mathcal{SV}_i{}^{\;t}$;
 9 for $i = 1$ to $n$ do
10     $\mathcal{SV}_i^{\mathcal{W}} = \mathcal{SV}_i^{\mathcal{W}} \cdot \frac{M_c\sqrt{d}}{\tau}$;
11 return $\mathcal{SV}_1^{\mathcal{W}}, \ldots, \mathcal{SV}_n^{\mathcal{W}}$.
```

## V. GENER L C SE: MULTIPLE POINTS PER OWNER

In this section, we tackle the general case where each data owner may have multiple points, i.e., $D_i = \{p_{i,1}, \ldots, p_{i,n_i}\}$ $(1 \leq i \leq n,\ n_i \geq 1)$.

### . n Insight

One major cost in computing Shapley values is that we have to calculate the scores of many points in each data coalition. If we can dramatically reduce the number of points needed to be considered and speed up the utility calculation, we can achieve faster Shapley value computation. In the context of preference queries, one fundamental insight is that *the utility of a set of data points depends only on the convex skyline points in the set*.

Let $p$ and $p'$ be two $d$-dimensional points. $p$ is said to *dominate* $p'$, denoted by $p' \prec p$, if $\forall i$ $(1 \leq i \leq d)$, $p[i] \geq p'[i]$, and there exists at least one $i$ for which $p[i] > p'[i]$. In a $d$-dimensional data set $D$, $p \in D$ is a *skyline point*, if it is not dominated by any other point in $D$ [7]. Denote by $Sky(D)$ the set of skyline points in $D$. Furthermore, $p$ is said to be a *convex skyline point*, if it is a skyline point in the set of vertices on the convex hull of $D \cup \{\;\}$, where is the origin of the $d$-dimensional space [28]. Denote by $Conv(D)$ the vertices on the convex hull of $D \cup \{\;\}$ and $CSky(D)$ the set of convex skyline points in $D$. We have:

**Lemma 2.** *Given a data set $D$ and an attribute weight space $\mathcal{W}$, $\mathcal{U}_{\mathcal{W}}(D) = \mathcal{U}_{\mathcal{W}}(CSky(D))$.*

*Proof sketch.* ccording to the fundamental theorem of linear programming [14], the highest score of a linear function over a convex hull attains at the hull's vertices, that is, $\mathcal{U}\;\;(D) = \mathcal{U}\;\;(Conv(D))$ for any $\mathbf{w} \in \mathcal{W}$. Thus, $\mathcal{U}_{\mathcal{W}}(D) = \mathcal{U}_{\mathcal{W}}(Conv(D))$. Furthermore, for two different points $p, p' \in Conv(D)$ such that $p' \prec p$, we have $Q(p; \mathbf{w}) \geq Q(p'; \mathbf{w})$ for any $\mathbf{w} \in \mathcal{W}$. Thus, $\mathcal{U}_{\mathcal{W}}(Conv(D)) = \mathcal{U}_{\mathcal{W}}(CSky(D))$. □

Using Lemma 2, we have the following result.

**Theorem 1.** *In characteristic games $(\{CSky(D_i)\}, \mathcal{U}_{\mathcal{W}})$ and $(\{D_i\}, \mathcal{U}_{\mathcal{W}})$, $\mathcal{SV}_{D_i}^{\mathcal{W}} = \mathcal{SV}_{CSky(D_i)}^{\mathcal{W}}$ $(1 \leq i \leq n)$.* □

Theorem 1 provides an efficient way to compute the Shapley value of a coalition. First, it allows us to eliminate the non-local convex skyline points from each data set in Shapley value computation.

Second, we only need to consider those points in the convex hulls of the skyline points of the data owners and their coalitions. This can be facilitated by the following result.

**Lemma 3.** *Let $D_1, \ldots, D_n$ be multiple sets of points. Then, $CSky(\cup_{i=1}^n D_i) = CSky(\cup_{i=1}^n CSky(D_i))$.*

*Proof sketch.* Since $\cup_{i=1}^n D_i \supseteq \cup_{i=1}^n CSky(D_i)$, $CSky(\cup_{i=1}^n D_i) \supseteq CSky(\cup_{i=1}^n CSky(D_i))$. We prove the equality by contradiction. Without loss of generality, assume that there is a point $x \in D_1$ such that $x \in CSky(\cup_{i=1}^n D_i)$ but $x \notin CSky(\cup_{i=1}^n CSky(D_i))$. Since $x \in CSky(\cup_{i=1}^n D_i)$, there does not exist any other point $y \in \cup_{i=1}^n D_i$ such that $x \prec y$ and $x$ is in the convex hull of $Sky(\cup_{i=1}^n D_i)$. Given $D \subseteq D'$, if $p \in D$ and $x \in CSky(D')$, then $p \in CSky(D)$. Therefore, $x \in CSky(D_1)$ and $x \in Sky(\cup_{i=1}^n CSky(D_i))$. Moreover, $x$ is in the convex hull of $\cup_{i=1}^n CSky(D_i)$. That is, $x \in CSky(\cup_{i=1}^n CSky(D_i))$. contradiction. □

### B. Computing $\mathcal{SV}_i^{\mathcal{W}}$ with Partitioning

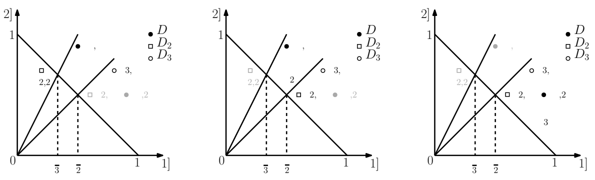
To compute Shapley values in an attribute weight space $\mathcal{W}$, our central idea is that we partition $\mathcal{W}$ into a series of subspaces $\{\mathcal{W}_1, \ldots, \mathcal{W}_b\}$ such that in each subspace $\mathcal{W}_j$ $(1 \leq j \leq b)$, the convex skyline of every data owner $D_i$ $(1 \leq i \leq n)$ contains only one point. Using the results in Section V- , we can reduce the computation of the Shapley value in each $\mathcal{W}_j$ into the situation addressed in Section IV.

Consider a set of local convex skyline points $CSky(D_i) = \{p_{i,1}, \ldots, p_{i,m_i}\}$ from a data set $D_i$ $(1 \leq i \leq n,\ m_i \geq 1)$. Denote by $\mathcal{P}(\mathcal{W}) = \{\mathcal{W}_1, \ldots, \mathcal{W}_b\}$ a partitioning of $\mathcal{W}$, that is, $\mathcal{W}_j \cap \mathcal{W}_{j_2} = \emptyset$ $(1 \leq j_1 < j_2 \leq b)$ and $\cup_{j=1}^b \mathcal{W}_j = \mathcal{W}$, such that in the subspace $\mathcal{W}_j$ $(1 \leq j \leq b)$, a local convex skyline point $p_{i,o_j} \in CSky(D_i)$ is the locally optimal point of $D_i$, i.e., $\forall \mathbf{w} \in \mathcal{W}_j$, $p_{i,o_j} = \arg\max_{p \in CSky(D_i)} Q(p; \mathbf{w})$. We have $\mathcal{SV}_i^{\mathcal{W}} = \sum_{j=1}^b \mathcal{SV}_i^{\mathcal{W}_j}$, where $\mathcal{SV}_i^{\mathcal{W}_j}$ is the Shapley value of $p_{i,o_j}$ in game $(\{p_{1,o_j}, \ldots, p_{n,o_j}\}, \mathcal{U}_{\mathcal{W}_j})$. To calculate $\mathcal{SV}_i^{\mathcal{W}_j}$, we can use either lgorithm 2 or 4 in Section IV.

The main idea to get $\mathcal{P}(\mathcal{W})$ is to partition $\mathcal{W}$ based on the local convex skyline points from each data owner, and then merge these individual partitionings. Let $\mathcal{W}_{p \geq p'} = \{\mathbf{w} \in \mathcal{W} | Q(p; \mathbf{w}) \geq Q(p'; \mathbf{w})\}$ be the attribute weight subspace where a point $p$ has a higher score than a point $p'$. point $p_{i,j} \in CSky(D_i)$ $(1 \leq i \leq n,\ 1 \leq j \leq m_i)$ is locally optimal in the subspace $\mathcal{W}_{i,j} = \cap_{1 \leq t \leq m_i, t \neq j} \mathcal{W}_{p_{i,j} \geq p_{i,t}}$. partitioning of $\mathcal{W}$ based on $CSky(D_i)$ can be $\mathcal{P}_i(\mathcal{W}) = \{\mathcal{W}_{i,1}, \ldots, \mathcal{W}_{i,m_i}\}$ $(1 \leq i \leq n)$. Finally, by merging these partitionings $\mathcal{P}_i(\mathcal{W})$ $(1 \leq i \leq n)$, we can construct the partitioning $\mathcal{P}(\mathcal{W})$.

*1) Partitioning in a 2-dimensional Data Space:* Let us first consider the scenario where all the points are in 2-dimensional space, i.e., $p_{i,j} \in \mathcal{R}^{+2}$ $(1 \leq i \leq n,\ 1 \leq j \leq m_i)$. In this case, $\mathcal{P}(\mathcal{W})$ can be obtained by sorting, as demonstrated in the following example.

**Example 4.** *Consider data owners in Table I and an attribute weight space* $\mathcal{W} \subseteq \{\mathbf{w} \mid \mathbf{w}[i] \geq 0, 1 \leq i \leq 2, \sum_{i=1}^{2} \mathbf{w}[i] = 1\}$*.  s shown in Figures 3(a)(b)(c),* $\mathcal{P}_1(\mathcal{W}) = \{\mathcal{W}_{1,1}, \mathcal{W}_{1,2}\}$*, where* $\mathcal{W}_{1,1} = \{\mathbf{w} \in \mathcal{W} \mid 0 \leq \mathbf{w}[1] \leq 1, 5\mathbf{w}[1] + 9(1 \quad \mathbf{w}[1]) \geq 9\mathbf{w}[1] + 5(1 \quad \mathbf{w}[1])\} = \{\mathbf{w} \in \mathcal{W} \mid 0 \leq \mathbf{w}[1] \leq \frac{1}{2}\}$*,* $\mathcal{W}_{1,2} = \{\mathbf{w} \in \mathcal{W} \mid \frac{1}{2} < \mathbf{w}[1] \leq 1\}$*.* $\mathcal{P}_2(\mathcal{W}) = \{\mathcal{W}_{2,1}, \mathcal{W}_{2,2}\}$*, where* $\mathcal{W}_{2,1} = \{\mathbf{w} \in \mathcal{W} \mid \frac{1}{3} < \mathbf{w}[1] \leq 1\}$ *and* $\mathcal{W}_{2,2} = \{\mathbf{w} \in \mathcal{W} \mid 0 \leq \mathbf{w}[1] \leq \frac{1}{3}\}$*.* $\mathcal{P}_3(\mathcal{W}) = \{\mathcal{W}_{3,1}\} = \{\mathcal{W}\}$*. We first sort the lower bounds and upper bounds of* $\mathbf{w}[1]$*'s in* $\mathcal{W}_{j \ ,j_2}$ $(1 \leq j_1, j_2 \leq 2)$ *into a list* $\langle 0, \frac{1}{3}, \frac{1}{2}, 1\rangle$*. Naturally,* $\mathcal{P}_1(\mathcal{W})$*,* $\mathcal{P}_2(\mathcal{W})$*, and* $\mathcal{P}_3(\mathcal{W})$ *are merged to a partitioning* $\mathcal{P}(\mathcal{W}) = \{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3\}$*, where* $\mathcal{W}_1 = \{\mathbf{w} \in \mathcal{W} \mid 0 \leq \mathbf{w}[1] \leq \frac{1}{3}\}$*,* $\mathcal{W}_2 = \{\mathbf{w} \in \mathcal{W} \mid \frac{1}{3} < \mathbf{w}[1] \leq \frac{1}{2}\}$*, and* $\mathcal{W}_3 = \{\mathbf{w} \in \mathcal{W} \mid \frac{1}{2} < \mathbf{w}[1] \leq 1\}$*. In* $\mathcal{W}_1$*, the locally optimal points from the three data owners are* $\{p_{1,1}, p_{2,2}, p_{3,1}\}$*, in* $\mathcal{W}_2$*,* $\{p_{1,1}, p_{2,1}, p_{3,1}\}$*, and in* $\mathcal{W}_3$*,* $\{p_{1,2}, p_{2,1}, p_{3,1}\}$*.* □



(a) $[0, \frac{1}{3}]$.   (b) $(\frac{1}{3}, \frac{1}{2}]$.   (c) $(\frac{1}{2}, 1]$.

Fig. 3: Partitioning in a 2-dimensional Space.

The detailed algorithm is presented in  lgorithm 5. For each point $p_{i,j} \in CSky(D_i)$ $(1 \leq i \leq n,\ 1 \leq j \leq m_i)$, we first compute the upper bound of the interval where $p_{i,j}$ is the locally optimal point. By sorting $CSky(D_i)$ in the $p_{i,j}[1]$ ascending order into $\pi_i$, we can determine the upper bound of the interval for $\pi_i(j)$ by its successor in $\pi_i$ (Lines 3-5). To merge all $\mathcal{P}_i$'s, the upper bounds are inserted into the list $ws$, and the upper bounds as well as the associated points achieving the bounds are inserted into local partitioning $\mathcal{P}_i$ (Lines 6-13). We sort the list of upper weight bounds $ws$ in ascending order (Line 15). The position index $I(i)$ $(1 \leq i \leq n)$ points to the current locally optimal point in $D_i$ (Line 16). To obtain the partitioning $\mathcal{P}(\mathcal{W})$, the interval $[temp\_w, w]$ with corresponding locally optimal points from each data owner is inserted into $\mathcal{P}$ (Lines 17-25). The time complexity of  lgorithm 5 is $O(\sum_{i=1}^{n} m_i \log m_i + (\sum_{i=1}^{n} m_i) \log n)$.

*2) Partitioning in a General Multi-dimensional Data Space:* We consider the scenario where all the points are in $d$-dimensional space, i.e., $p_{i,j} \in \mathcal{R}^{+d}$ $(1 \leq i \leq n,\ 1 \leq j \leq m_i,\ d > 2)$. In this case, the subspaces in partitionings are defined by a set of linear inequalities. Thus, to merge these partitionings, we utilize a linear programming solver [23] to detect whether two subspaces intersect. If an infeasible solution is returned by the linear programming solver, the two subspaces cannot be merged. The detailed algorithm is given in  lgorithm 6. We first compute $P_i$ $(1 \leq i \leq n)$ (Lines 1-6) and then merge $P_i$ $(1 \leq i \leq n)$ by checking each pair of subspaces. Subspace $\mathcal{P}_j.weight \cap \mathcal{P}_t.weight$ with points $\mathcal{P}_j.point \cup \mathcal{P}_t.point$ will be inserted into $\mathcal{P}'$ if $\mathcal{P}_j.weight \cap \mathcal{P}_t.weight$ is not empty. If $\mathcal{P}_j.weight \cap \mathcal{P}_t.weight = \mathcal{P}_j.weight$, it indicates that $\mathcal{P}_j.weight$ cannot

**lgorithm 5:** Partitioning in a 2-dimensional space.

```
input : local convex skyline points $CSky\ D_1)$, ...,
        $CSky\ D_n)$; an interval $\mathcal{W}[1] = [w_l, w_r]$;
output: $\mathcal{P}\ \mathcal{W})$ with locally optimal points;
1  initialize an empty list $ws$ and empty partitionings $\mathcal{P}_i$
     $(1 \leq i \leq n)$;
2  for $i = 1$ to $n$ do
3      let $\ _i$ be the list of points $CSky\ D_i)$ in the $p_{i,j}[1]$
         ascending order;
4      for $j = 1$ to $m_i \quad 1$ do
5          $w = \frac{\ _i(j+1)[2] \quad _i(j)[2]}{\ _i(j)[1] \quad _i(j)[1] \quad _i(j)[2] + \ _i(j)[2]}$;
6          if $w \leq w_l$ then continue;
7          if $w \geq w_r$ then $j \quad = 1$; break;
8          insert $w$ into $ws$;
9          $temp.weight = w$; $temp.point = \{\ _i\ j)\}$;
10         insert $temp$ into $\mathcal{P}_i$;
11     insert $w_r$ into $ws$;
12     $temp.weight = w_r$;
         $temp.point = \{\ _i\ \min\{j+1, m_i\})\}$;
13     insert $temp$ into $\mathcal{P}_i$;
14 initialize an empty partitioning $\mathcal{P}$;
15 sort $ws$ in ascending order;
16 $I\ i) = 0$ $(1 \leq i \leq n)$;
17 $temp\_w = w_l$;
18 for $w \in ws$ do
19     if $temp\_w \neq w$ then
20         $temp.weight = [temp\_w, w]$;$temp.point = \emptyset$;
21         for i=1 to n do
22             $temp.point = temp.point \cup \mathcal{P}_i[I\ i)].point$;
23             if $\mathcal{P}_i[I\ i)].weight = w$ then $I\ i) += 1$;
24         insert $temp$ into $\mathcal{P}$;
25         $temp\_w = w$;
26 return $\mathcal{P}$.
```

intersect with any other subspaces, so the loop terminates. If $\mathcal{P}_j.weight \cap \mathcal{P}_t.weight = \mathcal{P}_t.weight$, it indicates that $\mathcal{P}_t.weight$ cannot intersect with any other subspaces, so $\mathcal{P}_t$ is deleted from $\mathcal{P}_i$ (Lines 12-16). It is easy to see that the number of subspaces in  lgorithm 6 is $O((\sum_{i=1}^{n} m_i)^{2(d \quad 1)})$.

## *C. Computing* $\mathcal{SV}_i^{\mathcal{W}}$ *without Partitioning*

 lgorithm 4 can be easily extended to the scenario where each data owner has multiple points. The detailed procedure without partitioning is presented in  lgorithm 7. To start, we randomly sample attribute weight vectors $\mathbf{w}_t$ from a flat Dirichlet distribution in $d$ dimensions, i.e., a Dirichlet distribution $Dir(d)$ with parameters $\langle \frac{1}{d}, \ldots, \frac{1}{d}\rangle$ (Line 3). For each sample $\mathbf{w}_t \in \mathcal{W}$, we identify the locally optimal point in each owner's data set and then compute $\mathcal{SV}_i^{\ \ t}$ based on Lemma 1 $(1 \leq i \leq n)$ (Line 7). By evaluating the Shapley value $\mathcal{SV}_i^{\ \ t}$ on $\tau$ samples, the final estimate of the Shapley value $\mathcal{SV}_i^{\mathcal{W}}$ is obtained by averaging $\mathcal{SV}_i^{\ \ t}$ and scaling it by the measure of $\mathcal{W}$ and $\sqrt{d}$ (Line 10). This algorithm has time complexity $O(\tau(\sum_{i=1}^{n} m_i d + n \log n))$.

## VI. Experiment l Results

In this section, we report our extensive experimental results on the effectiveness and efficiency of the proposed algorithms.

**lgorithm 6:** Partitioning in a $d$-dimensional space.

```
input : local convex skyline points $CSky\ D_1), \ldots,$
        $CSky\ D_n)$; an attribute weight space $\mathcal{W}$;
output: $\mathcal{P}\ \mathcal{W})$ with locally optimal points;
initialize empty partitionings $\mathcal{P}_i$ $(1 \leq i \leq n)$;
for $i$ = 1 to $n$ do
    for $j$ = 1 to $m_i$ do
        compute a subspace $\mathcal{W}_{i,j}$ for $p_{i,j}$ by computing
          $\mathcal{W} \cap\ \cap_{t=1}^{m_i} \mathcal{W}_{p_{i,j} \geq p_{i,t}})$;
        $temp.weight = \mathcal{W}_{i,j}$; $temp.point = \{p_{i,j}\}$;
        insert $temp$ into $\mathcal{P}_i$;
$\mathcal{P} = \mathcal{P}_1$;
for $i$=2 to $n$ do
    initialize an empty partitioning $\mathcal{P}'$;
    for $\mathcal{P}_j \in \mathcal{P}$ do
        for $\mathcal{P}_t \in \mathcal{P}_i$ do
            if $\mathcal{P}_j.weight \cap \mathcal{P}_t.weight \neq \emptyset$ then
                $temp.weight = \mathcal{P}_j.weight \cap \mathcal{P}_t.weight$;
                  $temp.point = \mathcal{P}_j.point \cup \mathcal{P}_t.point$;
                insert $temp$ into $\mathcal{P}'$;
                if $\mathcal{P}_j.weight \cap \mathcal{P}_t.weight = \mathcal{P}_j.weight$
                  then break;
                if $\mathcal{P}_j.weight \cap \mathcal{P}_t.weight = \mathcal{P}_t.weight$
                  then delete $\mathcal{P}_t$ from $\mathcal{P}_i$;
    $\mathcal{P} = \mathcal{P}'$;
return $\mathcal{P}$.
```

**lgorithm 7:** Computing $\mathcal{SV}_i^{\mathcal{W}}$ without partitioning.

```
input : local convex skyline points $CSky\ D_1), \ldots,$
        $CSky\ D_n)$; an attribute weight space $\mathcal{W}$; sample
        size $\tau > 0$;
output: Shapley value $\mathcal{SV}_i{}^{\mathcal{W}}$ for $D_i\ \ 1 \leq i \leq n)$;
$\mathcal{SV}_i^{\mathcal{W}} = 0\ \ 1 \leq i \leq n)$; $\tau' = 0$;
for $t = 1$ to $\tau$ do
    let $\ _t$ be a sample from $Dir\ d)$;
    if $\ _t \in \mathcal{W}$ then
        $\tau' += 1$;
        for $i = 1$ to $n$ do
            compute $\mathcal{SV}_i{}^{t}$ $(1 \leq i \leq n)$ based on Lemma 1;
            $\mathcal{SV}_i^{\mathcal{W}} += \mathcal{SV}_i{}^{t}$;
for $i = 1$ to $n$ do
    $\mathcal{SV}_i^{\mathcal{W}} = \mathcal{SV}_i^{\mathcal{W}} \cdot \frac{\tau' \sqrt{d}}{\tau^2 (d\ \ 1)}$;
return $\mathcal{SV}_1^{\mathcal{W}}, \ldots, \mathcal{SV}_n^{\mathcal{W}}$.
```

### . *Experiment Setup*

Our experiments were conducted on a machine comprising two 40-core Intel(R) Xeon(R) Platinum 8383C CPU@2.70GHz, running the Ubuntu 20.04.6 LTS 64-bit operating system with 1TB main memory. We implemented the algorithms using Python for the integral components and C++ for the remaining components. To ensure efficient processing, we ran all experiments in parallel using 40 threads. The source code is available at [3].

*1) Methods:* We evaluate the following proposed algorithms.

- **P RR**: The **p**artitioning & **arr**angement algorithm ( lgorithm 2). **P RR** computes Shapley values with partitioning ( lgorithms 5 or 6) and arrangement ( lgorithms 1 or 3), depending on the dimensionality. **P RR** computes the exact Shapley values in a 2-dimensional data space and approximates Shapley values when the dimensionality is over 2.
- **PSSV**: The **p**artitioning & **s**ampling **S**hapley **v**alue computation algorithm ( lgorithm 4) approximates Shapley values with partitioning ( lgorithms 5 or 6, depending on the dimensionality) but without arrangement.
- **WSSV**: The **w**hole space & **s**ampling **S**hapley **v**alue computation algorithm ( lgorithm 7) approximates Shapley values without partitioning or arrangement.

We compare with two baselines, **ENUM** and **PERM**.

- **ENUM**: The coalition **enum**eration baseline computes Shapley values by computing utilities of $2^n\ \ 1$ coalitions according to Equation 1, where $n$ is the number of data owners. In the 2-dimensional case, **ENUM** computes the exact Shapley value. In a $d$-dimensional data space ($d > 2$), **ENUM** approximates Shapley values as the utility of coalition $S$ is approximated as $\frac{\sqrt{d}}{(d\ \ 1)} \sum_{t=1} \max_{p \in CSky(D(S))}\{Q(p; \mathbf{w}_t)\}$, where $\tau$ is the sample size and $\mathbf{w}_t$ is sampled from $Dir(d)$, a flat Dirichlet distribution in $d$ dimensions.
- **PERM**: The **perm**utation-based sampling baseline approximates Shapley values by sampling permutations and computing average marginal contributions [11]. Denote by **PERM**-$k$X sampling $kn$ permutations. The utilities are computed in the same way as **ENUM**.

*2) Data Sets:* In our experiment, we use both synthetic and real data sets. We defer the discussion about the real data set to Section VI-F.

We generated synthetic data sets in two steps. First, we generated independent (INDEP), correlated (CORR), and anti-correlated ( NTI) data sets following the procedures in [7].

Second, we randomly assigned the data points in a data set produced in the first step to $n$ data owners. Each data point was assigned to only one data owner. Two different scenarios were implemented: uniform (UN) and round-robin (RR). UN assigned data points to data owners following a uniform distribution. Each data owner had an equal probability of getting every data point. RR assigned data points to data owners based on skyline layers [49] with a Pareto distribution [59]. Let $l$ be the number of skyline layers in a data set. The data points were divided into $n$ groups, where the $i$-th ($1 \leq i \leq n$) group contained the points from the $\Big((i\ \ \ 1)\lfloor\frac{l}{n}\rfloor + \min\{i\ \ \ 1, l\%n\} + 1\Big)$-th to the $\Big(i\lfloor\frac{l}{n}\rfloor + \min\{i, l\%n\}\Big)$-th layers. In each group, the data points were distributed among $n$ data owners using a Pareto distribution. That is, a small number of data owners held the majority of data points in this group. We applied Round Robin to the Pareto distribution to ensure that each data owner held a majority of data points in a certain group.

In total, we have six different settings for synthetic data sets, namely INDEP-UN, INDEP-RR, CORR-UN, CORR-RR,  NTI-UN, and  NTI-RR. These settings allow us to thoroughly evaluate the impact of the data distribution and data owner distribution on the performance of methods.

Consistent performance trends were observed among all methods when applied to the INDEP-UN, CORR-UN, and  NTI-UN data sets, as well as with the INDEP-RR, CORR-RR, and  NTI-RR data sets. Due to space limitations, we present only the experimental results for the  NTI-UN and  NTI-RR settings to illustrate the performance of the methods. We chose the  NTI setting because it poses the greatest challenge to our proposed methods, given that the increased number of local convex skyline points leads to higher computational costs in partitioning and arrangement.

*3) Evaluation Metrics:* We used the output computed by **ENUM** as the benchmark. Given a benchmark Shapley value $\mathcal{SV}_i$ and an approximated Shapley value $\overline{\mathcal{SV}_i}$ $(1 \leq i \leq n)$, the *relative difference* (RD) is defined as $RD = \frac{{}^{n}_{i}|\mathcal{SV}_i \quad \overline{\mathcal{SV}_i}|}{{}^{n}_{i}\mathcal{SV}_i}$. For ease of comparison, we normalized Shapley values such that the sum of Shapley values of all players is 1. Then, the relative difference can be rewritten as $RD = \sum_i^n |\mathcal{SV}_i \quad \overline{\mathcal{SV}_i}|$. In the 2-dimensional case, since the integral can be calculated exactly, we call the relative difference the *relative error* (RE).  dditionally, we report the standard deviation (STD).

*4) Parameter Settings:* We set the attribute weight space to be the entire simplex in experiments on the synthetic data sets and the case study, i.e., $\mathcal{W} = \{\mathbf{w} \mid \mathbf{w}[j] \geq 0 \ (1 \leq j \leq d), \sum_{j=1}^{d} \mathbf{w}[j] = 1\}$. In a multi-dimensional data space, all algorithms compute integrals by Monte Carlo sampling [26]. The number of samples for integrals affects both the runtime and degree of convergence. We employed the relative deviation to evaluate the degree of convergence. Given a set of estimated Shapley values $\{\overline{\mathcal{SV}_i^1}, \ldots, \overline{\mathcal{SV}_i^{\tau}}\}$ $(1 \leq i \leq n)$ obtained by sampling $\tau$ times using the same algorithm under the same setting, where $\overline{\mathcal{SV}_i^j}$ is the $j$-th estimated Shapley value of $D_i$ computed by the algorithm, the *relative deviation* is defined as $RelativeDeviation = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}\frac{|\overline{\mathcal{SV}_i^j} \ - \ {}_{j=}\overline{\mathcal{SV}_i^j}|}{- \ {}_{j=}\overline{\mathcal{SV}_i^j}}$. In all experiments, $\tau$ was set to 10 and the sample size for integrals was set to the minimum number of samples required to achieve $RelativeDeviation < 0.01$. The settings of the parameters are shown in Table II.

T BLE II: Parameter Settings (defaults are in bold).

| Parameter | Settings |
|---|---|
| # of Data Owners | 5, **10**, 15, 20, 25, 30, 40, 50 |
| # of Data Points | 500, 1k, 2k, 5k, **10k**, 20k, 50k, 100k, 200k, 500k |
| # of Dimensions | **2**, **3**, 4, 5, 6, 7, 8, 9, 10 |

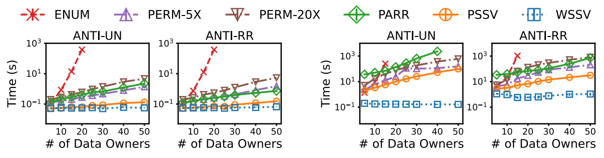
## *B. Scalability on Number of Data Owners*

We first evaluate the scalability with respect to the number of owners. The results are shown in Figure 4. In the 2-dimensional cases (Figure 4(a)), **PERM** with sample size $5X$ and $20X$ outperforms the **ENUM** method. **PERM** exhibits poor approximation quality with these two sample sizes, with the average relative error ranging from 18.62% to 37.05% and the average standard deviation from 0.03% to 0.33%, as detailed in Table III. The errors are significant. In this table, the statistics are computed across all the cases with various numbers of owners.

The runtime of **P  RR** is very close to that of **PERM-**5X in  NTI-UN and outperforms **PERM** with both sample sizes in  NTI-RR. **P  RR** computes the exact Shapley values, but **PERM** does not. **PSSV** and **WSSV** outperform all other methods by orders of magnitude in terms of runtime and maintain a consistently low relative error. Compared with **PSSV**, **WSSV** is faster but has a higher relative error. **PSSV** and **WSSV** present a tradeoff between the quality of the approximation and the computational efficiency.

In the 3-dimensional cases (Figure 4(b)), when the number of data owners is small, **P  RR** is slower than **ENUM** due to the extra computational cost in the arrangement. The time complexity of the arrangement method is polynomial in the number of data owners. Thus, when there are more data owners, **P  RR** outperforms **ENUM** as expected. Different from the 2-dimensional case, **P  RR** is slower than **PERM** in the 3-dimensional case, but still the average relative difference and standard deviation of **P  RR** are substantially lower than those of **PERM**, as detailed in Table IV, where the statistics are computed across all the cases with various numbers of owners. **ENUM** cannot finish in one hour when the number of data owners is over 15, thus we only report the relative differences with respect to no more than 15 owners. **PSSV** and **WSSV** consistently outperform all other methods and obtain much lower relative differences compared with **PERM**.

In summary, our **P  RR** method computes the exact Shapley values and outperforms **ENUM** in the 2-dimensional cases. While in higher dimensions where Shapley values need to be approximated, our proposed **PSSV** and **WSSV** outperform **PERM**, excelling in terms of both runtime and approximation quality. Moreover, **PSSV** and **WSSV** provide a tradeoff between runtime efficiency and approximation quality.



(a) 2-dimensional cases. (b) 3-dimensional cases.

Fig. 4: Scalability with respect to Number of Data Owners.

T BLE III:  verage RE and STD with Different Numbers of Data Owners in 2-dimensional Cases.

| Setting | NTI-UN | | NTI-RR | |
|---|---|---|---|---|
| | vg. RE | vg. STD | vg. RE | vg. STD |
| PERM 5X | 37.05% | 0.06% | 35.66% | 0.33% |
| PERM 20X | 19.65% | 0.03% | 18.62% | 0.25% |
| PARR | 0% | 0% | 0% | 0% |
| PSSV | 0.02% | 0.01% | 0.03% | 0.01% |
| WSSV | 0.11% | 0.05% | 0.30% | 0.12% |

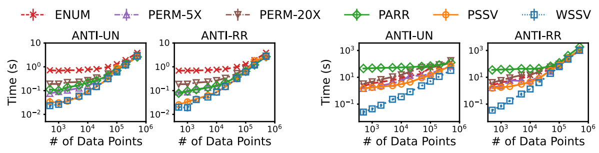
## *C. Scalability on Number of Data Points*

Figure 5 shows the results on the scalability with respect to the number of data points in the log-log scale. In both

T BLE IV: verage RD and STD with Different Numbers of Data Owners in 3-dimensional Cases.

| Setting | NTI-UN | | NTI-RR | |
|---|---|---|---|---|
| | vg. RD | vg. STD | vg. RD | vg. STD |
| PERM 5X | 44.13% | 0.51% | 42.60% | 0.22% |
| PERM 20X | 20.22% | 0.46% | 18.87% | 0.40% |
| PARR | 1.13% | 0.26% | 0.53% | 0.09% |
| PSSV | 1.14% | 0.26% | 0.54% | 0.10% |
| WSSV | 1.24% | 0.37% | 0.69% | 0.17% |

the 2- and 3-dimensional cases, as the number of data points increases, the runtime of all algorithms tends to converge to the same trend. This occurs because the cost of identifying local convex skyline points, which is consistent across all algorithms, gradually dominates the total runtime. In the 2-dimensional cases, **P RR**, which computes the exact Shapley values, outperforms **ENUM** and **PERM** in runtime. In both the 2- and 3-dimensional cases, **PSSV** and **WSSV** outperform all other methods and achieve much lower relative differences compared with **PERM**, as detailed in Tables V and VI.



(a) 2-dimensional cases. (b) 3-dimensional cases.

Fig. 5: Scalability with respect to Number of Data Points.

T BLE V: verage RE and STD with Different Numbers of Data Points in 2-dimensional Cases.

| Setting | NTI-UN | | NTI-RR | |
|---|---|---|---|---|
| | vg. RE | vg. STD | vg. RE | vg. STD |
| PERM 5X | 43.37% | 0.13% | 41.66% | 0.39% |
| PERM 20X | 19.76% | 0.05% | 19.12% | 0.13% |
| PARR | 0% | 0% | 0% | 0% |
| PSSV | 0.03% | 0.01% | 0.06% | 0.03% |
| WSSV | 0.12% | 0.08% | 0.29% | 0.13% |

T BLE VI: verage RD and STD with Different Numbers of Data Points in 3-dimensional Cases.

| Setting | NTI-UN | | NTI-RR | |
|---|---|---|---|---|
| | vg. RD | vg. STD | vg. RD | vg. STD |
| PERM 5X | 42.88% | 0.23% | 40.82% | 0.44% |
| PERM 20X | 19.79% | 0.26% | 18.35% | 0.38% |
| PARR | 1.30% | 0.23% | 1.53% | 0.30% |
| PSSV | 1.30% | 0.23% | 1.52% | 0.30% |
| WSSV | 1.37% | 0.29% | 1.60% | 0.33% |

## D. Scalability on Dimensionality

Figure 6 shows the scalability with respect to dimensionality. The runtime of all methods increases as the dimensionality increases. **P RR** and **PSSV** exhibit the greatest sensitivity to increases in dimensionality, leading to timeouts when dealing with dimensions of 4 and 5, respectively. The exponential surge in the number of subspaces after partitioning is the underlying reason for these timeouts in these specific methods.

Conversely, our proposed method, **WSSV**, demonstrates remarkable scalability, outperforming **ENUM** and **PERM** by at least one order of magnitude. It also achieves significantly lower relative differences compared with **PERM**. Due to space limitations, we omit the details here.

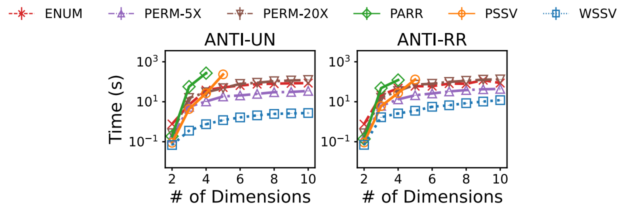


Fig. 6: Scalability with respect to Dimensionality.

## E. Extension of the Utility Function

Our approaches can be readily adapted to scoring functions that are linear in weight and monotonically increase with each attribute. Linearity in weights ensures the space can be partitioned by hyperplanes, and monotonicity enables pruning points using the convex skyline. The independence or correlation of the attributes has no impact on the applicability of our approaches. To validate the extension capability of our approaches, Table VII shows the experiment result under the default setting of CORR data sets with an exponential scoring function of $Q(p;\mathbf{w}) = \mathbf{w}[1]e^{p[1]p[2]} + \mathbf{w}[2]e^{p[2]p[3]} + \mathbf{w}[3]e^{p[1]p[3]}$. Similar as before, **WSSV** achieves the highest efficiency, while **P RR** offers the highest accuracy. **PSSV** strikes a balance between accuracy and efficiency.

T BLE VII: verage RE and verage Runtime for $Q(p;\mathbf{w}) = \mathbf{w}[1]e^{p[1]p[2]} + \mathbf{w}[2]e^{p[2]p[3]} + \mathbf{w}[3]e^{p[1]p[3]}$

| Setting | CORR-UN | | CORR-RR | |
|---|---|---|---|---|
| | Time (s) | vg. RE (%) | Time (s) | vg. RE (%) |
| ENUM | 0.47±0.12 | / | 0.47±0.03 | / |
| PERM 5X | 0.26±0.02 | 37.30±0.39 | 0.17±0.02 | 27.64±0.94 |
| PERM 20X | 0.67±0.15 | 17.26±0.18 | 0.38±0.05 | 13.66±0.47 |
| PARR | 1.94±0.37 | **0.76±0.42** | 2.09±0.05 | **0.96±0.37** |
| PSSV | 0.23±0.01 | 0.82±0.45 | 0.24±0.01 | 1.04±0.39 |
| WSSV | **0.02±0.00** | 1.34±0.52 | **0.03±0.01** | 1.33±0.18 |

## F. Case Studies

To test the performance of our proposed methods in real-world applications, we conducted case studies on a real data set, irbnb Listing [35], which contains listings from 114 countries/cities and 108,819 hosts. Each listing has 6 attributes: review scores for `ccur cy`, `cle nliness`, `check-in`, `communic tion`, `loc tion`, and `v lue`. To ensure the reliability of reviews, we filtered out hosts with fewer than 100 reviews.

In our case studies, we derived data owner details directly from the dataset, treating hosts as owners and their listings as data points. Then, for each country/city, we ran all methods to compute the Shapley values for these data owners. cross the countries/cities, the number of data owners ranges from 38 to 6,442, the number of data points varies from 83 to 15,987, and the number of dimensions is fixed at 6.

The formidable challenges posed by the high dimensionality of the data space and the significant number of data owners render **P RR**, **PSSV**, and **ENUM** incapable of completing any Shapley value computations for data owners. **PERM** also encounters timeouts in about 32% of the country/city cases. Remarkably, **WSSV** is the only method that accomplishes all Shapley value computations across all the country/city scenarios, with runtime ranging from 0.04 seconds to 0.51 seconds. s anticipated, **WSSV** shows scalability with respect to the number of data owners, data points, and dimensions. To further show the effectiveness of **WSSV**, we conducted experimental analysis on 15 districts/boroughs with 5 to 18 data owners such that **ENUM** can complete the calculation to obtain benchmark Shapley values for comparison. The average relative difference of **WSSV** is only 0.08%, significantly outperforming **PERM-5X** and **PERM-20X**, which exhibited average relative differences of 33.19% and 16.72%, respectively. This demonstrates the effectiveness and efficiency of **WSSV** in real-world applications.

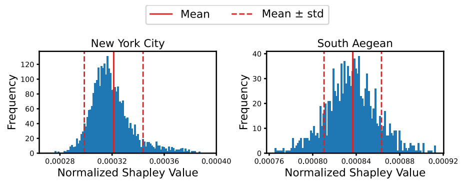


Fig. 7: The Shapley Value Distribution of Two Cities.

We also make some interesting observations regarding the distribution of Shapley values among all data owners across different countries/cities. mong them, we identify two distinct patterns, exemplified by the two selected cities shown in Figure 7. In the equi-width histograms, the Shapley values are normalized so that the sum of the Shapley values of all hosts is 1. Moreover, the width of the bucket is set to 6% of the standard deviation. The Shapley value distribution in New York City is right-skewed, which indicates that the majority of hosts provided a lower-than-average contribution to the overall satisfaction of the accommodation experience, while a few hosts contributed significantly more to the satisfaction. The host with the highest Shapley value provides three listings, whereas the host with 11 listings has a Shapley value below the average. This observation suggests that offering more listings does not necessarily lead to a higher Shapley value. The Shapley value distribution in South egean is symmetric, suggesting a balanced distribution of hosts contributing above and below the average. Our efficient method **WSSV** enables the discovery of these patterns.

**Comparison with Other llocation Method.** We compare the Shapley value based allocation method to the Leave-One-Out (LOO) based allocation method. The LOO method measures the contribution of a data owner as the utility difference between the grand coalition and the grand coalition excluding that owner: $LOO(D_i) = \mathcal{U}_{\mathcal{W}}(\mathcal{D}) \quad \mathcal{U}_{\mathcal{W}}(\mathcal{D}\setminus D_i)$. We compute the values of the hosts, remove them one by one in descending value order, and record the utility of the remaining hosts. Figure 8 shows the impact of removing high-value owners on the grand utility using two methods in New York City and South egean datasets. The Shapley value method exhibits a significantly faster decrease in utility compared with the LOO method. This is because the LOO method allocates zero value to hosts who do not achieve the highest utility, regardless of how close their utility is to the highest utility. The Shapley value method effectively assigns value by accounting for all marginal contributions and better differentiates the value of the hosts.

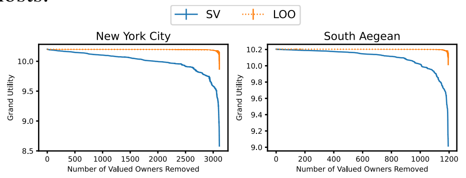


Fig. 8: Impact of Removing High-Value Owners.

### G. *Discussion*

The experimental results show that **P RR**, **PSSV**, and **WSSV** have tradeoffs between accuracy and efficiency. **P RR** offers the highest accuracy, delivering exact results in 2-dimensional scenarios and performing efficiently with a large number of data owners, whereas **WSSV** excels in speed and addresses the complexity posed by the partitioning of **P RR** and **PSSV** in high-dimensional spaces. **PSSV** offers a balance between the two. lthough **P RR** and **PSSV** are slower than **WSSV**, they provide additional insights through space partitioning. For example, they can identify subspaces in which a data owner contributes the most or least.

## VII. Conclusion

In this paper, we studied the novel problem of data valuation in cooperative preference query answering. We leveraged the well-established Shapley value and defined the utility function as the highest score achieved by the assembled data set considering all potential user preferences. Remarkably, we found that Shapley values can be computed in polynomial time for any given linear preference function as long as each data owner possesses only a single data point. Building on this nice base case, we proposed algorithms **P RR**, **PSSV**, and **WSSV** to compute Shapley values with any subsets of linear preference functions, allowing each data owner to have multiple data points. Our approaches also balance computational efficiency and approximation accuracy. Using a series of synthetic data sets and real irbnb Listing data, we systematically confirmed the effectiveness and efficiency of our approaches in comparison to baseline methods. Our strategies prove to be both efficient and applicable in real-world scenarios. Our approaches can be readily adapted to suit linear-in-weight monotone functions and provide insights into broader applications with different utility functions. Exploring nonlinear-in-weight utility functions presents a promising direction for further work.

## cknowledgment

We thank the anonymous reviewers for their helpful comments. J. Zhang and J. Liu are supported in part by the National Key RD Program of China (2022YFB3103401, 2021YFB3101100), NSFC (62102352, 62472378, U23 20306), and the Zhejiang Province Pioneer Plan (2024C01074). Xuan Luo is supported in part by a startup grant from York University.

## References

[1] . garwal, M. . Dahleh, and T. Sarkar. marketplace for data: n algorithmic solution. In . Karlin, N. Immorlica, and R. Johari, editors, *Proceedings of the 2019 CM Conference on Economics and Computation, EC 2019, Phoenix, Z, US , June 24-28, 2019*, pages 701–726. CM, 2019.

[2] P. hmed, . Eldawy, V. Hristidis, and V. J. Tsotras. Reverse spatial top-k keyword queries. *VLDB J.*, 32(3):501–524, 2023.

[3] nonymous. Code repository. https://anonymous.4open.science/r/PreferenceShapley/readme.md, 2024.

[4] . sudeh, S. Thirumuruganathan, N. Zhang, and G. Das. Discovering the skyline of web databases. *Proc. VLDB Endow.*, 9(7):600–611, 2016.

[5] M. Bai, J. Xin, G. Wang, L. Zhang, R. Zimmermann, Y. Yuan, and X. Wu. Discovering the k representative skyline over a sliding window. *IEEE Trans. Knowl. Data Eng.*, 28(8):2041–2056, 2016.

[6] Bittensor. What is the consensus mechanism of tao bittensor? https://bittensor.org/. ccessed: 2024-10-27.

[7] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In D. Georgakopoulos and . Buchmann, editors, *Proceedings of the 17th International Conference on Data Engineering, pril 2-6, 2001, Heidelberg, Germany*, pages 421–430. IEEE Computer Society, 2001.

[8] M. . Burgess and . C. Chapman. pproximating the shapley value using stratified empirical bernstein sampling. In Z. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on rtificial Intelligence, IJC I 2021, Virtual Event / Montreal, Canada, 19-27 ugust 2021*, pages 73–81. ijcai.org, 2021.

[9] S. Cabello and T. M. Chan. Computing shapley values in the plane. In G. Barequet and Y. Wang, editors, *35th International Symposium on Computational Geometry, SoCG 2019, June 18-21, 2019, Portland, Oregon, US* , volume 129 of *LIPIcs*, pages 20:1–20:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[10] J. Castro, D. Gómez, E. Molina, and J. Tejada. Improving polynomial estimation of the shapley value by stratified random sampling with optimum allocation. *Comput. Oper. Res.*, 82:180–188, 2017.

[11] J. Castro, D. Gómez, and J. Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & OR*, 36(5):1726–1730, 2009.

[12] Y. Chang, L. D. Bergman, V. Castelli, C. Li, M. Lo, and J. R. Smith. The onion technique: Indexing for linear optimization queries. In W. Chen, J. F. Naughton, and P. . Bernstein, editors, *Proceedings of the 2000 CM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, US* , pages 391–402. CM, 2000.

[13] Z. Chen, P. Manolios, and M. Riedewald. Why not yet: Fixing a top-k ranking that is not fair to individuals. *Proc. VLDB Endow.*, 16(9):2377–2390, 2023.

[14] E. K. P. Chong and S. H. Żak. *Introduction to Linear Programming*, chapter 15, pages 297–331. John Wiley & Sons, Ltd, 2008.

[15] J. B. Clark. *The distribution of wealth: a theory of wages, interest and profits*. Macmillan, 1908.

[16] M. de Berg, M. van Kreveld, M. Overmars, and O. C. Schwarzkopf. *Computational Geometry*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[17] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19(2):257–266, 1994.

[18] D. Deutch, N. Frost, B. Kimelfeld, and M. Monet. Computing the shapley value of facts in query answering. In Z. Ives, . Bonifati, and . E. bbadi, editors, *SIGMOD '22: International Conference on Management of Data, Philadelphia, P , US , June 12 - 17, 2022*, pages 1570–1583. CM, 2022.

[19] S. D. C. di Vimercati, S. Foresti, G. Livraga, V. Piuri, and P. Samarati. Supporting user requirements and preferences in cloud plan selection. *IEEE Trans. Serv. Comput.*, 14(1):274–285, 2021.

[20] B. Ding, B. Zhao, C. X. Lin, J. Han, C. Zhai, . N. Srivastava, and N. C. Oza. Efficient keyword-based search for top-k cells in text cube. *IEEE Trans. Knowl. Data Eng.*, 23(12):1795–1810, 2011.

[21] T. Eisenmann, G. Parker, and M. W. Van lstyne. Strategies for two-sided markets. *Harvard business review*, 84(10):92, 2006.

[22] E. Farchi, R. Narayanam, and L. Nagalapatti. Ranking data slices for ML model validation: shapley value approach. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, pril 19-22, 2021*, pages 1937–1942. IEEE, 2021.

[23] K. Fukuda. CDD: Computational Geometry lgorithms Library. 2022.

[24] Y. Gao, Q. Liu, G. Chen, B. Zheng, and L. Zhou. nswering why-not questions on reverse top-k queries. *Proc. VLDB Endow.*, 8(7):738–749, 2015.

[25] . Ghorbani and J. Y. Zou. Data shapley: Equitable valuation of data for machine learning. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, US* , volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR, 2019.

[26] G. H. Givens and J. . Hoeting. *Computational statistics*, chapter 6, pages 151–200. John Wiley & Sons, Ltd, Hoboken, NJ, US , 2 edition, 2012.

[27] Z. He and E. Lo. nswering why-not questions on top-k queries. In . Kementsietsidis and M. . V. Salles, editors, *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, US ( rlington, Virginia), 1-5 pril, 2012*, pages 750–761. IEEE Computer Society, 2012.

[28] J. Heo, K. Whang, M. Kim, Y. Kim, and I. Song. The partitioned-layer index: nswering monotone top-k queries using the convex skyline and partitioning-merging technique. *Inf. Sci.*, 179(19):3286–3308, 2009.

[29] J. Hicks. *The theory of wages*. Springer, 1963.

[30] W. Hoeffding. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[31] V. Hristidis, N. Koudas, and Y. Papakonstantinou. PREFER: system for the efficient execution of multi-parametric ranked queries. In S. Mehrotra and T. K. Sellis, editors, *Proceedings of the 2001 CM SIGMOD international conference on Management of data, Santa Barbara, C , US , May 21-24, 2001*, pages 259–270. CM, 2001.

[32] G. . D. Hub. Shapley value analysis. https://developers.google.com/ads-data-hub/guides/shapley. ccessed: 2024-10-23.

[33] M. Iansiti and K. R. Lakhani. Managing our hub economy. *HBR'S*, 10:117, 2017.

[34] I. F. Ilyas, G. Beskales, and M. . Soliman. survey of top-*k* query processing techniques in relational database systems. *CM Comput. Surv.*, 40(4):11:1–11:58, 2008.

[35] Inside irbnb. Inside irbnb: dding data to the debate. http://insideairbnb.com. ccessed: 2023-06-01.

[36] N. Jethani, M. Sudarshan, I. C. Covert, S. Lee, and R. Ranganath. Fastshap: Real-time shapley value estimation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, pril 25-29, 2022*. OpenReview.net, 2022.

[37] R. Jia, D. Dao, B. Wang, F. . Hubis, N. M. Gürel, B. Li, C. Zhang, C. J. Spanos, and D. Song. Efficient task-specific data valuation for nearest neighbor algorithms. *Proc. VLDB Endow.*, 12(11):1610–1623, 2019.

[38] B. Karlas, D. Dao, M. Interlandi, B. Li, S. Schelter, W. Wu, and C. Zhang. Data debugging with shapley importance over end-to-end machine learning pipelines. *CoRR*, abs/2204.11131, 2022.

[39] F. B. Kashani, P. Ghaemi, B. Movaqar, and S. J. Kazemitabar. Efficient maximal reverse skyline query processing. *GeoInformatica*, 21(3):549–572, 2017.

[40] I. Keles and K. Hose. Skyline queries over knowledge graphs. In C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. F. Cruz, . Hogan, J. Song, M. Lefrançois, and F. Gandon, editors, *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, uckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, volume 11778 of *Lecture Notes in Computer Science*, pages 293–310. Springer, 2019.

[41] N. Kumar, B. Raichel, S. Sintos, and G. V. Buskirk. pproximating distance measures for the skyline. In P. Barceló and M. Calautti, editors, *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, volume 127 of *LIPIcs*, pages 10:1–10:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[42] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. CM*, 22(4):469–476, 1975.

[43] Y. Kwon, M. . Rivas, and J. Zou. Efficient computation and analysis of distributional shapley values. In . Banerjee and K. Fukumizu, editors, *The 24th International Conference on rtificial Intelligence and Statistics, IST TS 2021, pril 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 793–801. PMLR, 2021.

[44] R. Leluc, F. Portier, J. Segers, and . Zhuman. Speeding up monte carlo integration: Control neighbors for optimal convergence. *CoRR*, abs/2305.06151, 2023.

[45] Y. Li, X. Yu, and N. Koudas. Top-k queries over digital traces. In P. . Boncz, S. Manegold, . ilamaki, . Deshpande, and T. Kraska, editors, *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, msterdam, The Netherlands, June 30 - July 5, 2019*, pages 954–971. CM, 2019.

[46] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: The k most representative skyline operator. In R. Chirkova, . Dogac, M. T. Özsu, and T. K. Sellis, editors, *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, pril 15-20, 2007*, pages 86–95. IEEE Computer Society, 2007.

[47] S. C. Littlechild and G. Owen. simple expression for the shapley value in a special case. *Management Science*, 20(3):370–372, 1973.

[48] J. Liu, J. Lou, J. Liu, L. Xiong, J. Pei, and J. Sun. Dealer: n end-to-end model marketplace with differential privacy. *Proc. VLDB Endow.*, 14(6):957–969, 2021.

[49] J. Liu, L. Xiong, J. Pei, J. Luo, and H. Zhang. Finding pareto optimal groups: Group-based skyline. *Proc. VLDB Endow.*, 8(13):2086–2097, 2015.

[50] J. Liu, L. Xiong, Q. Zhang, J. Pei, and J. Luo. Eclipse: Generalizing knn and skyline. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, pril 19-22, 2021*, pages 972–983. IEEE, 2021.

[51] J. Liu, J. Yang, L. Xiong, and J. Pei. Secure skyline queries on cloud platform. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, C , US , pril 19-22, 2017*, pages 633–644. IEEE Computer Society, 2017.

[52] J. Liu, J. Yang, L. Xiong, J. Pei, and J. Luo. Skyline diagram: Finding the voronoi counterpart for skyline queries. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, pril 16-19, 2018*, pages 653–664. IEEE Computer Society, 2018.

[53] E. Livshits, L. E. Bertossi, B. Kimelfeld, and M. Sebag. The shapley value of tuples in query answering. In C. Lutz and J. C. Jung, editors, *23rd International Conference on Database Theory, ICDT 2020, March 30- pril 2, 2020, Copenhagen, Denmark*, volume 155 of *LIPIcs*, pages 20:1–20:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[54] S. M. Lundberg and S. Lee. unified approach to interpreting model predictions. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *dvances in Neural Information Processing Systems 30: nnual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, C , US* , pages 4765–4774, 2017.

[55] X. Luo, J. Pei, Z. Cong, and C. Xu. On shapley value in data assemblage under independent utility. *Proc. VLDB Endow.*, 15(11):2761–2773, 2022.

[56] S. Maleki, L. Tran-Thanh, G. Hines, T. Rahwan, and . Rogers. Bounding the estimation error of sampling-based shapley value approximation with/without stratifying. *CoRR*, abs/1306.4265, 2013.

[57] X. Meng, H. Zhu, and G. Kollios. Top-k query processing on encrypted databases with strong security guarantees. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, pril 16-19, 2018*, pages 353–364. IEEE Computer Society, 2018.

[58] R. Mitchell, J. Cooper, E. Frank, and G. Holmes. Sampling permutations for shapley value estimation. *J. Mach. Learn. Res.*, 23:43:1–43:46, 2022.

[59] M. E. Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005.

[60] . B. Owen. *Monte Carlo theory, methods and examples*. 2013.

[61] E. Pastor, L. de lfaro, and E. Baralis. Looking for trouble: nalyzing classifier behavior via pattern divergence. In G. Li, Z. Li, S. Idreos, and D. Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 1400–1412. CM, 2021.

[62] F. Psallidas, B. Ding, K. Chakrabarti, and S. Chaudhuri. S4: top-k spreadsheet-style search for query discovery. In T. K. Sellis, S. B. Davidson, and Z. G. Ives, editors, *Proceedings of the 2015 CM SIGMOD International Conference on Management of Data, Melbourne, Victoria, ustralia, May 31 - June 4, 2015*, pages 2001–2016. CM, 2015.

[63] J. Qi, F. Zuo, H. Samet, and J. C. Yao. K-regret queries using multiplicative utility functions. *CM Trans. Database Syst.*, 43(2):10:1–10:41, 2018.

[64] . E. Roth and R. E. Verrecchia. The shapley value as applied to cost allocation: a reinterpretation. *Journal of ccounting Research*, pages 295–303, 1979.

[65] C. Shapiro and H. R. Varian. *Information rules: strategic guide to the network economy*. Harvard Business Press, 1999.

[66] L. S. Shapley. value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[67] J. Shurman. Multivariable calculus. *Found at http://people. reed. edu/jerry/211/vcalc. pdf*, pages 111–119, 2010.

[68] N. H. Sleumer. Output-sensitive cell enumeration in hyperplane arrangements. *Nordic journal of computing*, 6(2):137–147, 1999.

[69] T. Song, Y. Tong, and S. Wei. Profit allocation for federated learning. In C. Baru, J. Huan, L. Khan, X. Hu, R. k, Y. Tian, R. S. Barga, C. Zaniolo, K. Lee, and Y. F. Ye, editors, *2019 IEEE International Conference on Big Data (IEEE BigData), Los ngeles, C , US , December 9-12, 2019*, pages 2577–2586. IEEE, 2019.

[70] D. Stewart. *Lectures on political economy*, volume 2. T. &T. Clark, 1877.

[71] B. Tang, K. Mouratidis, and M. Han. On m-impact regions and standing top-k influence problems. In G. Li, Z. Li, S. Idreos, and D. Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 1784–1796. CM, 2021.

[72] B. Tang, K. Mouratidis, and M. L. Yiu. Determining the impact regions of competing options in preference space. In S. Salihoglu, W. Zhou, R. Chirkova, J. Yang, and D. Suciu, editors, *Proceedings of the 2017 CM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, US , May 14-19, 2017*, pages 805–820. CM, 2017.

[73] Y. Tao, X. Xiao, and J. Pei. Efficient skyline and top-k retrieval in subspaces. *IEEE Trans. Knowl. Data Eng.*, 19(8):1072–1088, 2007.

[74] M. W. Van lstyne, G. G. Parker, and S. P. Choudary. Pipelines, platforms, and the new rules of strategy. *Harvard business review*, 94(4):54–62, 2016.

[75] T. Wang, Y. Yang, and R. Jia. Learnability of learning performance and its application to data valuation. *CoRR*, abs/2107.06336, 2021.

[76] D. Wu, Y. Li, B. Choi, and J. Xu. Social-aware top-k spatial keyword search. In . B. Zaslavsky, P. K. Chrysanthis, C. Becker, J. Indulska, M. F. Mokbel, D. Nicklas, and C. Chow, editors, *IEEE 15th International Conference on Mobile Data Management, MDM 2014, Brisbane, ustralia, July 14-18, 2014 - Volume 1*, pages 235–244. IEEE Computer Society, 2014.

[77] D. Xin, J. Han, H. Cheng, and X. Li. nswering top-k queries with multi-dimensional selections: The ranking cube approach. In U. Dayal, K. Whang, D. B. Lomet, G. lonso, G. M. Lohman, M. L. Kersten, S. K. Cha, and Y. Kim, editors, *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 463–475. CM, 2006.

[78] Z. Yu, X. Yu, N. Koudas, Y. Liu, Y. Li, Y. Chen, and D. Yang. Distributed processing of k shortest path queries over dynamic road networks. In D. Maier, R. Pottinger, . Doan, W. Tan, . lawini, and H. Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, US ], June 14-19, 2020*, pages 665–679. CM, 2020.

[79] J. Zhang, Y. Bi, M. Cheng, J. Liu, K. Ren, Q. Sun, Y. Wu, Y. Cao, R. C. Fernandez, H. Xu, R. Jia, Y. Kwon, J. Pei, J. T. Wang, H. Xia, L. Xiong, X. Yu, and J. Zou. survey on data markets. *arXiv preprint arXiv:2411.07267*, 2024.

[80] J. Zhang, Q. Sun, J. Liu, L. Xiong, J. Pei, and K. Ren. Efficient sampling approaches to shapley value approximation. *Proc. CM Manag. Data*, 1(1):48:1–48:24, 2023.

[81] K. Zhang, Y. Tong, Y. Shi, Y. Zeng, Y. Xu, L. Chen, Z. Zhou, K. Xu, W. Lv, and Z. Zheng. pproximate k-nearest neighbor query over spatial data federation. In X. Wang, M. L. Sapino, W. Han, . E. bbadi, G. Dobbie, Z. Feng, Y. Shao, and H. Yin, editors, *Database Systems for dvanced pplications - 28th International Conference, D SF 2023, Tianjin, China, pril 17-20, 2023, Proceedings, Part I*, volume 13943 of *Lecture Notes in Computer Science*, pages 351–368. Springer, 2023.

[82] S. Zhang, S. Ray, R. Lu, Y. Zheng, Y. Guan, and J. Shao. Towards efficient and privacy-preserving interval skyline queries over time series data. *IEEE Trans. Dependable Secur. Comput.*, 20(2):1348–1363, 2023.

[83] Z. Zhang, H. Lu, B. C. Ooi, and . K. H. Tung. Understanding the meaning of a shifted sky: a general framework on extending skyline query. *VLDB J.*, 19(2):181–201, 2010.

[84] S. Zheng, Y. Cao, and M. Yoshikawa. Secure shapley value for cross-silo federated learning. *Proc. VLDB Endow.*, 16(7):1657–1670, 2023.