PrefDiv: Efficient Algorithms for Effective Top-k Result Diversification

Xiaoyu Ge
University of Pittsburgh
xiaoyu@cs.pitt.edu

Panos K. Chrysanthis
University of Pittsburgh
panos@cs.pitt.edu

ABSTRACT

The ever-increasing supply of data is bringing renewed attention to result diversification, a technique usually studied with result relevance for a given task. Together, they produce a subset of results that are relevant to the user query and contain less redundant information. In this work, we formulate an extended version of the result diversification problem, considering three objectives—relevance, diversity, and coverage—and present a novel approach and algorithms that produce better-diversified results. Our approach takes a large set of possible answers generated from a user query and outputs a representative subset of results that are highly ranked according to the preference of the user. The data items contained in the representative set are diverse, such that each item is different from the rest and provides good coverage of the underlying aspects of the original results. Our approach also suggests a set of appropriate parameters for each user query to achieve a balance between our conflicting objectives and is efficient enough to ensure an interactive experience. We study the complexity of our algorithms and experimentally evaluate them in terms of normalized relevance, coverage, and execution time. Our evaluation indicates a speedup of up to 159x, and outperforms the state-of-the-art algorithms on multiple fronts.

1 INTRODUCTION

Motivation With the exponential increase in the amount of data being generated every second, the term "Big Data" that is adopted to represent the challenge of large-scale data processing is currently mentioned frequently in everyday life [20]. This reflects the fact that people are increasingly reliant on using data as an integral part of their daily activities (e.g., decisions and collaborations).

The challenge of scalable data processing can be examined from two viewpoints. Traditionally, scalability has been seen from a *systems point of view*, where challenges can be attributed to an increasing rate of data on the one hand, and network bandwidth, processing power, and storage limitation on the other hand. Scalability can also be viewed from a *human point of view* [23]. Given the expornential volume of data, the challenge here is how to avoid overwhelming users with irrelevant results.

Query personalization is a well-known technique for dealing with scalability challenges from a human point of view, which often happens at two different levels:

- *Ranking* – Ranking techniques utilize user *preferences* with the aim of providing the most relevant results to the users (e.g., [33]). These techniques can be distinguished as *quantitative-based*, *qualitative-based*, or *hybrid*, based on the type of user preferences that they can support.
- *Diversification* – Diversification techniques aim to reduce the amount of redundant information in the results. These techniques typically group data in sets that are most "dissimilar" with each other (e.g., [3, 12]).

Since highly-ranked items can be similar, result diversification [31] has recently drawn significant attention as a technique to facilitate applications such as keyword search, recommendation systems, and online shopping. The key idea of result diversification is to output a subset of representative results from the original in an informative way, since the user most probably will not view results beyond a small number. This requires the representative top-k results to be relevant, diverse, and maintain good coverage of the original answers (i.e., able to cover different underlying aspects of the original results). One thing to note is that the definitions of both relevance and diversity are subjective; thus, they can vary depending on the query and the user.

Goal In this paper, we present an approach to efficiently compute the representative result set for arbitrary top-k queries under user-definable relevance and diversity definitions. We name this as the *Diversified Top-k (DT-k)* problem.

Challenges Below, we will illustrate the challenges to our proposed approach by means of three examples.

Example 1.1. Assume a tourist who is currently visiting Athens wants to find an affordable restaurant with great taste. So she visits a publicly available database that contains the relation RESTAURANT (Name, Food Type, Cost, Score), where *Name* indicates the official name of the restaurant; *Food Type* indicates the type of food (e.g., Greek, Japanese, Chinese); *Cost* is the average expense per person, and *Score* is a numeric number between 1 and 10 that indicates the quality of the food and services offered at the restaurant. To find the ideal place for dinner, she, therefore, enters the following SQL-like query:

*SELECT * FROM RESTAURANT*
WHERE Score ≥ 6 *AND Cost* ≤ 20
ORDER BY Cost ASC;

However, these kinds of queries may produce thousands of results, among which the top 5 and bottom 5 results are listed in Table 1. The problem is that users are typically only interested in seeing a small portion of these results, not to mention many of these results are, in fact, redundant (e.g., differ only in the name). Simply fetching a certain top number (e.g., top 5) of results does not help improve their usefulness. Instead, the user might be better served with the right amount of diverse (i.e., dissimilar) items from the original answer with good coverage of different aspects (e.g., Food Type, Cost, Score). Furthermore, among those representative subsets with good diversity and coverage, the one that is most relevant to the user's interest should be preferred, such that the relevance refers to criteria that can be used to rank the answers. These may be obtained by interoperating the SQL-like query itself (e.g., through the "Order By" predicates), or derived from external user profiles (e.g., query histories, crowdsourcing).

One immediate challenge raised is how to define diversity, which clearly changes based on the user and the query being performed. In our work, we associate diversity with the similarity between pairs of answers (i.e., data items). To address this challenge, we propose a tunable definition that can be adjusted with a set of diversity thresholds DIV. Each threshold div in DIV is a real number between $[0, 1]$, which specifies the threshold between "similar" and "dissimilar" data items with respect to the normalized distance given by the specified distance measure (e.g., Euclidean, Manhattan, and Hamming) and attributes. $|DIV| = 0$ results in the traditional top-k query, while more diversity thresholds with higher values increase the diversity of the result set.

Name	Food Type	Cost	Score
McDonald	Fast Food	8	7
KFC	Fast Food	8	7
Burger King	Fast Food	8	7
Arby's	Fast Food	8	7
Oinomageireio H Epirus	Greek	8	9
.....			
Scala Vinoteca	Greek	20	9
Ta Karamanlidika tou Fani	Greek	20	10
A Little Taste of Home	Greek	20	9
Liondi Traditional Greek	Greek	20	9
Dio Dekares i Oka	Greek	20	9

Table 1: Top-5 and bottom-5 tuples with respect to the cost.

Name	Food Type	Cost	Score
McDonald	Fast Food	8	7
Beer Garden Ritterburg	German	8	9
Nolan	Japanese	9	8
Oinomageireio H Epirus	Greek	10	10
Dosirak	Korean	12	6

Table 2: Top-5 tuples based on cost that are diversified respect to attributes "Food Type" and "Score".

Example 1.2. With the above diversity parameter, the previous sample query in Example 1.1 could be expanded accordingly:

SELECT * *FROM* RESTAURANT
WHERE Score $\geq$ 6 *AND* (*Cost* $\leq$ 20)
ORDER BY Cost *DESC*
DIVERSE BY div = 0.2 *ON* 'Food Type' (Hamming)
AND div = 0.3 *ON* 'Score' (Euclidean) *LIMIT* 5*;*

where Food Type and Score are the attributes on which the diversity is calculated, and Hamming and Euclidean are the corresponding distance measures. The idea here is to generate a set of results that follow the diversity constraints *DIV* specified within the query [1]. The result of the above query is illustrated in Table 2. Although the above example produces some compelling results with its information representative subset, it could be difficult to see how the coverage is contributing differently to the results than the dissimilarity. To illustrate the importance of the coverage, let us consider a simple example:

Example 1.3. Consider the nodes in Figures 1 and 2. In these two figures, each node represents an item in the dataset, and an edge exists between a pair of nodes iff the similarity between these two nodes are close enough according to some pre-defined threshold. On the one hand, in Figure 2, a set of dissimilar items $\{v_5, v_4\}$ is selected. However, only $\{v_1, v_5, v_4\}$ are considered to be covered by $\{v_5, v_4\}$, as $\{v_2, v_3\}$ are not connected with either v_5 or v_4. On the other hand, in Figure 1, a single vertex v_1 is connected to all four vertices, hence achieving 100% coverage. In this case, one can see that vertex v_1 better represents the entire graph when compared with $\{v_5, v_4\}$, thus indicating coverage is another valuable aspect to the quality of the representative results.

The above two examples (i.e., Example 1.2 and 1.3) illustrate the key advantages and desired features of an effective approach that provides a meaningful and representative subset of the original query results. First, the representative subset is relevant to the intention of the query and contains items that would be ranked highly in the original results. Second, the chosen representative items are diverse, each contributing additional novelty to the answer. Third, the representative items are selected in a way that most items in the original answers are reachable with a small distance (i.e., change) from one of the representative answers. Clearly, simply applying ranking, diversification, or clustering on the original result sets could not achieve the above properties. Thus, techniques that clearly consider multiple aspects of the representative results are needed to address this challenge.

[1]Note that our PrefDiv algorithms take the set of diversity constraints DIV as one of their inputs, and it is up to the design of the actual system that integrates the PrefDiv to determine how DIV will be integrated with its user query.

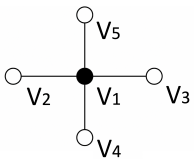

Figure 1: Single vertex v_1 with 100% coverage.

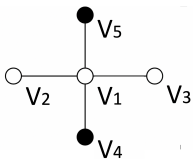

Figure 2: A set of vertices $\{v_4, v_5\}$ with 60% coverage.

Unfortunately, as we will discuss in more detail in Section 2.2.3, finding the optimal solution that maximizes both the "relevance" and "diversity" is an NP-Hard problem by itself, let alone with the addition of the other aspect "coverage" that should also be considered when producing the representative results.

Our Approach To overcome these challenges, we propose an extremely efficient online algorithm, called *Preferential Diversity* (PrefDiv) [13] , for producing representative result sets with sufficient relevance, diversity, and coverage of the original answers. PrefDiv is a top-k bounded general diversification approach that can be applied to any existing relevance ranking model and datasets to retrieve a diversity-aware top-k representative subset of results. PrefDiv starts to construct the representative result set with the k most relevant results (according to the ranking method), then gradually refine this representative set by eliminating pairs of items that do not satisfy the constraints specified by the set of diversity thresholds DIV. This is achieved by identifying pairs of items in the representative result set that violate one or more diversity thresholds, and then, among the two items contained in the pair, one with lower relevance will be replaced with an item from the database that improves diversity and coverage. In the end, PrefDiv produces k representative results balanced between relevance, diversity, and coverage.

To the best of our knowledge, PrefDiv is the first general approach to deliver representative results that explicitly consider relevance, diversity, and coverage with an interactive speed that is independent of the underlying database and data set.

However, in order to optimize multiple conflicting objectives such as relevance and diversity, a common approach taken by most diversification algorithms, including our PrefDiv, is to utilize a number of tunable parameters. This could be a major drawback for an algorithm, because with the increase of the number of required parameters, the complexity of the algorithm increases as well, making it more difficult to use in real-world scenarios.

In this paper, we extend and present a family of PrefDiv algorithms based on the vanilla PrefDiv. These includes two novel algorithms that automatically determine: 1) the corresponding diversity thresholds $DIV = \{div_1, div_2, ..., div_n\}$ given the set of diversity constraints Ψ, and 2) the tunable parameters A that balance the trade-off between the relevance and diversity, respectively.

Contributions To achieve the solution as described above, this paper makes the following contributions.

- We formulate the *Diversified Top-k (DT-k)* problem, provide a theoretical analysis of its complexity, and show NP-hardness results. (Section 2)
- We provide a detailed description of the design of PrefDiv, which is an efficient online result diversification algorithm.

Table 3: LIST OF NOTATIONS USED IN THE PAPER

Symbol	Explanation
R_Q	a set of initial query results
R	a set of representative results
k	the number of item in the result-set
$\mathcal{L}$	the number of iterations to obtain R with PrefDiv
Ψ	a set of diversity constraints
ψ	the diversity constraint
div	a diversity threshold
div_{opt}	an optimal diversity threshold
Δ	a set of dimensions
A	a relevance parameter
υ	a self-adjustable relevance parameter
I_x	the intensity value of item x
$U(x)$	a utility function that produces the I_x
$sim_\Psi(x_i, x_j)$	x_i and x_j are similar w.r.t Ψ
$dissim_\Psi(x_i, x_j)$	x_i and x_j are dissimilar w.r.t Ψ

- We introduce the concept of Relevance Proportionality (RP), that dynamically balances the trade-off between relevance and diversity during the retrieval of the top-k representative results based on the given query and dataset.
- We propose a novel greedy algorithm that automatically finds the optimal diversity threshold, which maximizes the coverage of the representative result set produced by the PrefDiv.
- We perform extensive experimental evaluations with two real-world datasets, Cameras [10] and Foursquare [7]. Our experimental results show that PrefDiv and its optimizations outperform the most similar state-of-the-art competitors, as suggested in [34], by a significant margin. Compared to other alternatives (discussed in Section 5), our algorithm achieves up to 159x speedup and produces a representative subset that better covers the original answers with a negligible performance decrease in relevance. (Section 4)

2 PROBLEM FORMULATION

In our work, we assume that the database DB is composed of N data items over a D-dimensional space $(d_1, d_2, ..., d_D)$, where each dimension $d \in D$ can be either numerical or categorical attributes. Note that the above assumption enables us to handle any data type (e.g., structured, semi-structured, unstructured) as long as they are vectorized. The user specifies a query Q that aims to retrieve a set of k representative items from DB over a subset of dimensions S, such that $S \leq D$. The goal here is to produce a set of k items that maximizes the *relevance* while ensuring the *diversity* (i.e., ensuring each item is diverse with respect to one another). Below, we will first provide the necessary background and basic concepts of our problem, and then present our problem definition and analyze its complexity. The list of symbols used in the following sections of the paper is shown in Table 3.

2.1 Background

2.1.1 Relevance. The relevance of R represents the degree of the relevancy of each data item $x \in R$ and is typically represented with a utility function $U(x)$ that measures the "goodness" of each data item with respect to certain metrics.

Definition 1. *Relevance* – Given a database DB and a utility function $U(x)$, the relevance is measured as the outcome of $U(x)$, which is an *intensity value* $I_x \in (0,1) \subset \mathbb{R}$ for item $x \in$ DB that is used to express the degree of benefit from retrieving x.

A higher intensity value indicates that a data item is more desired than those items with a lower intensity value.

The intensity value (i.e., relevance score) enables database systems to produce a total order of each data item in a given data set and thus allow the extraction of top-k data items. This is simply achieved by retrieving k data items with the highest intensity value.

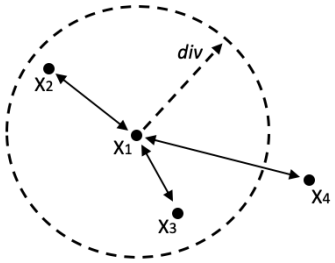

Figure 3: Illustration of similarity and dissimilarity.

2.1.2 Diversity. In our work, the diversity of a set of data items R is achieved by enforcing each pair of data items in R to be *dissimilar* with respect to each other, such that two data items x_i, and x_j are said to be dissimilar if for a given set of user-specified diversity constraints Ψ, x_i and x_j satisfy all constraints $\psi \in \Psi$. Formally, we can define a diversity constraint as follows:

Definition 2. *Diversity Constraint* – For a given pair of items x_i and x_j, a set of attributes Δ, a distance threshold div, and a distance function $dist(x_i, x_j, \Delta)$ that measures the distance between x_i and x_j with respect to the set of dimensions specified in Δ. A diversity constraint ψ is satisfied iff $dist(x_i, x_j, \Delta) > div$.

Based on the above definition of diversity constraints, we now define dissimilarity as:

Definition 3. *Dissimilarity* – Let X be a set of data items. For a given set of diversity constraints Ψ, two items x_i and $x_j \in X$ are dissimilar to each other, denoted as $dissim_\Psi(x_i, x_j)$, if they satisfy each diversity constraint ψ in Ψ.

Consequently, the similarity can simply be defined as the opposite of the dissimilarity, such that:

Definition 4. *Similarity* – Let X be a set of data items. For a given set of diversity constraints Ψ, two items x_i and $x_j \in X$ are similar to each other, denoted as $sim_\Psi(x_i, x_j)$, if they fail to satisfy at least one diversity constraint in Ψ.

Figure 3 illustrates the concept of similarity and dissimilarity with four 2-dimensional data items $x_1, .., x_4$, and a single diversity constraint that requires the Euclidean distance between each data object with respect to both dimensions (i.e., $\Delta = d_1, d_2$) to be at least div apart. Let us take point x_1 as an example. According to Definition 4, points $\{x_2, x_3\}$ are similar to x_1, since $dist(x_2, x_1, \Delta) \leq div$ and $dist(x_3, x_1, \Delta) \leq div$. In contrast, x_4 is dissimilar with respect to x_1, as $dist(x_4, x_1, \Delta) > div$.

2.1.3 Coverage. As pointed out in the previous literature [11], the coverage is another aspect that is important to the quality of the representative results. Since the size of the representative results is very restricted compared to the original answers, having a set of representative results with good coverage increases the chance for the user to get meaningful information from the selected representation items. Furthermore, coverage enables the system to organize all the answers in a cluster-like fashion, where each original answer of query Q can still be retrieved by "zoom-in" into one of the representative items. Such that the "zoom-in" operation will reveal all answers that are "similar" to the selected representative item. The actual implementation of this "zoom-in" operation has been well discussed in [11], thus it is omitted from the discussion of this paper.

Clearly, the coverage is defined completely based on the definition of the similarity and thus related heavily to the diversity constraints when the number of representative results is fixed

to a certain number k. When k is fixed, a set of more relaxed diversity constraints (i.e., with higher diversity threshold) will help the representative set include more original answers into its coverage, and a set of stricter diversity constraints will certainly decrease the coverage of the representative set. In particular, given the definition of similarity, if item x_j satisfies $sim_\Psi(x_i, x_j)$, x_j is said to be covered by the item x_i. Consequently, we can define the coverage of a set of items as follows:

Definition 5. *Coverage* – Given a set of original answers R_Q and a representative result set R, where $R \subseteq R_Q$, the coverage of R corresponds to the percentage of items in R_Q that satisfies $sim_\Psi(x_i, x_j)$, such that $x_i \in R$ and $x_j \in R_Q$.

2.2 Diversified Top-k (DT-k) Problem

Based on the above discussions and definitions, we name our problem the *Diversified Top-k (DT-k)* problem.

2.2.1 Problem Formulation. Consider a database DB that consists of N data items distributed over a multi-dimensional space with mixed numeric and categorical dimensions. Given a query Q and its corresponding initial results set R_Q over DB, the desired result cardinality of k, a utility function $U(x)$, and a set of diversity constraints Ψ, the solution of DT-k produces a k-sized representative subset R from the original results R_Q, whose *relevance*, according to $U(x)$ is maximum, while satisfying the set of diversity constraints Ψ.

We name the above k-sized subset of representative results as Diversified Top-k (DT-k) set.

2.2.2 Problem Complexity. Finding the optimal DT-k Set for the Diversified Top-k problem is computationally hard, which can be shown by mapping it to the well-known *Maximum-weight Independent Set* problem [1]. We can achieve the mapping by forming a graph of G that corresponds to the original results R_Q. Each data item x_i in R_Q maps to a vertex v_i in G. An edge e is added between two vertices v_i and v_j if the distance between these two vertices is close enough such that not all diversity constraints are satisfied, and the intensity value I_{x_i} of an item x_i represents the weights of the corresponding vertex in G. Some tractable solutions have been proposed in the literature [18, 21], but these solutions require either a very specific type of graph (e.g., Outerstring graphs) or have strict restrictions (e.g., sparsity, outcome degree of each vertex). Thus, they are not practical in our environment.

2.2.3 Secondary Objective. As discussed above, coverage is another important aspect of result diversification, which is dependent completely on the diversity threshold specified inside each diversity constraint. Given that diversity constraints are typically defined by the user, this may lead to sub-optimal results if the user fails to define reasonable constraints. Consequently, our secondary objective is to address this challenge by automatically adapting the diversity constraints based on the type of the query being performed and the initial result set. Later, in Section 3.4, we will present a general optimization that helps determine the most suitable diversity constraints for different user queries.

3 PREFDIV ALGORITHMS

In this section, we introduce our solution to the Diversified Top-k problem. First, we start with the discussion of a naive approach to the problem and then propose our solution to this problem, namely, *Preferential Diversity* (PrefDiv) algorithm. Finally, we discuss some optimizations that improve the effectiveness of our proposed PrefDiv algorithm and reduce its number of tunable parameters.

3.1 Naive Solution

Before we discuss our solutions, one naive solution to the Diversified Top-k problem work as follows: given a new user query

ALGORITHM 1: PrefDiv

Require:
Initial result set R_Q, result cardinality k, relevance parameter A, a set of diversity constraints Ψ
Ensure:
One subset R of R_Q
$T \leftarrow \emptyset$
while exists unexamined items in R_Q and $|R| < k$ **do**
  $T \leftarrow$ Pick k items with highest intensity from R_Q
  for all $x_i \in T$ **do**
    if $Dissim_\Psi(x_i, x_j) : \forall x_j \in R$ **then**
      $R \leftarrow R \cup x_i$
    else
      Mark x_i as "redundant"
  while number of promoted items in R from $T < A * k$ **do**
    $R \leftarrow R \cup x_{max}$, s.t., x_{max} is marked & $\forall x_j \in T, I_{x_{max}} \geq I_{x_j}$
    $T \leftarrow T - x_{max}$
  $A \leftarrow A/2$
  $R_Q = R_Q - T$
Return R

Q, a k, a set of initial results $R_Q = \{x_1, ..., x_t\}$, a utility function $U(x)$ and a set of diversity constraints Ψ, for each item in R_Q of q, we first compute and sort each item in R_Q according to the intensity value computed by the $U(x)$. We pick the item $x_i \in R_Q$ with the highest intensity value; for each remaining items x_j in R_Q, we mark them as "Eliminated" if they are similar to the x_i (i.e., $sim_\Psi(x_i, x_j)$). We then add x_i into the final result set R and remove x_i from R_Q. Afterwards, a new unmarked item with the highest intensity value will be picked from R_Q, and the previous steps will be repeated until either $|R| = k$ or all remaining items in $|R_Q|$ are marked as "Eliminated".

This naive solution is a greedy approach that will eventually produce a set of items that satisfy all diversity constraints with relatively high-intensity values. Clearly, the naive solution is computationally expensive, especially when the size of R_Q is large. Furthermore, it does not guarantee the resulting set to contain at least k items. However, we use this naive solution as a foundation and propose an efficient online solution that achieves better performance with much less computational cost.

3.2 Preferential Diversity

Our Preferential Diversity algorithm is an online solution for the DT-k problem. As discussed in the previous section, finding the optimal solution to the DT-k problem is computationally expensive. Thus we chose a greedy approach in the PrefDiv design. To maximize the efficiency of PrefDiv, we need to develop it as an online algorithm that accesses database tuples (i.e., items) incrementally. The main idea underlying PrefDiv is minimizing as much as possible the number of data items being examined.

PrefDiv builds the DT-k set R by starting with a set of k highest ranked data item (with respect to the relevance score/intensity value), and then gradually replacing items that fail the diversity constraints with slightly less relevant but diverse items outside of R that satisfy the diversity constraints. This process continues until all items in R satisfy the specified diversity constraints.

One potential issue is that relevant items in the DT-k set tend to be similar to each other. Thus strictly enforcing diversity constraints may eliminate too many items that are highly beneficial to the user. To address this issue, we propose a *relevance parameter A* that allows PrefDiv to produce representative results with *partial* diversity. When $A = 1$, R would simply be the top k items from the initial set, i.e., the items with the k highest intensity

values. When $A = 0$, R contains k dissimilar items from the initial set. When A is between 0 and 1 and given that PrefDiv is an iterative algorithm, considering k objectives each iteration, the final result will have at least $A * k$ items from every iteration, and in each iteration A will be divided by half. For example, when $A = 0.5$ and $k = 20$, the first iteration will select at least $20 * 0.5$ items for the final result set, the second iteration will select at least $20 * (0.5 * 0.5)$ items, and so on. With this parameter, the user is able to control the trade-off between relevance *vs.* diversity by enabling partial diversity whenever necessary.

As illustrated in Algorithm 1, the basic logic of PrefDiv is as follows: PrefDiv takes as input, a set of initial results R_Q sorted according to the descending of their intensity value, the desired result cardinality of k, partial diversity parameter A, and a set of diversity constraints Ψ. It outputs a DT-k set that represents the original answers R_Q. In each iteration, PrefDiv fetches and removes k items with the highest intensity value from R_Q and places them in a temporary set T. Each of the items in T is then compared with items currently in R, such that any item in T that fails to satisfy all diversity constraints with respect to all items in R will be marked as "Redundant"; else, it will be added into R immediately. This process will continue until all items in T are either moved into R or marked as "Redundant". Once all k items fetched in the current iteration have been examined, PrefDiv will check if a sufficient number of items were moved into R according to parameter A. In case the number is not sufficient, the difference will be covered by the highest-ranked items (with respect to intensity value) that are marked as "Redundant" in the current iteration. Afterward, the above iteration will be repeated until k representative items are produced ($|R| = k$).

Time Complexity According to the above discussion, we can observe that the worst-case complexity of PrefDiv is $O(kN)$, since each of the N unlabeled items will be compared at most $k - 1$ times with the items that are currently in the result set before being included or discarded from the final result. Fortunately, as the size of k is usually a small number, PrefDiv should typically behave as a linear algorithm. Furthermore, as we will show in our empirical studies (Section 4), depending on the diversity constraints, PrefDiv typically does not need to examine all original items in R_Q. That is, a very small set of the item would be sufficient enough to produce R if Ψ are appropriately defined.

3.3 Relevance Proportionality

From the above discussions, it should be clear that having a good balance between relevance and diversity is important to the quality of the representative result set. In PrefDiv, we have introduced the relevance parameter A to enable the partial diversity, which helps preserve the relevance of the representative results. Our empirical study shows that such a parameter does help improve the quality of the result set R. However, it is up to the user to define A for any query, and this may increase user efforts when using our algorithm. This motivated us to introduce a new self-adjusted parameter υ to replace the manual relevance parameter A, which led to a new variation of PrefDiv called *Preferneral Diversity with Proportional Relevance* (PrefDiv-PR). As illustrated in Algorithm 2, the idea here is to automatically compute the right amount of items that should be promoted into the final result set based on the proportion of the relevance of each iteration.

In particular, υ adapts to the aggregated intensity value of all items in each iteration, and can be computed as follows: Assume a given set of original results R_Q, a DT-k subset $R \subseteq R_Q$ and a number of iterations $\mathcal{L}$ needed for PrefDiv to obtain the result set R. For each iteration ℓ., s.t. $\ell < \mathcal{L}$, a set of items with the highest intensity value from the remaining items of R_Q are reserved into a separated set B_ℓ, s.t. $B_\ell \subseteq R_Q$ and $|B_\ell| = k$. The υ_ℓ of an

ALGORITHM 2: PrefDiv-PR

Require:
Initial result set R_Q, result cardinality k, a set of diversity constraints Ψ

Ensure:
One subset R of R_Q
$\mathcal{L} \leftarrow 0$
$T \leftarrow \emptyset$
while exists unexamined items in R_Q and $|R| < k$ **do**
  $T \leftarrow$ Pick k items with highest intensity from R_Q
  for all $x_j \in T$ **do**
    if $Dissim_\Psi(x_j, x_t) : \forall x_t \in R$ **then**
      $R \leftarrow R \cup x_j$
  $R_Q = R_Q - T$
  $B_\mathcal{L} \leftarrow T - R$
  Increase $\mathcal{L}$ by one
for $\ell = 1 \rightarrow \mathcal{L}$ **do**
  $\upsilon_\ell \leftarrow$ Compute υ_ℓ according to Equation 1
  while the number of items in R from $B_\ell < \upsilon_\ell * |R|$ **do**
    $R \leftarrow R - x_j$, s.t. $x_j \in R, I_{x_j} < I_{x_k} : \forall x_k \in R$
    $R \leftarrow R \cup x_i$, s.t. $x_i \in B_\ell, I_{x_i} \geq I_{x_t} : \forall x_t \in B_\ell$
    $B_\ell \leftarrow B_\ell - x_i$
Return R

iteration ℓ is calculated through the following equation:

$$\upsilon_\ell = \frac{\sum_{x \in B_\ell} I_x}{\sum_{j=1}^{\mathcal{L}} \sum_{x_c \in B_j} I_{x_c}} \tag{1}$$

Recall from Section 2.1.1, I_x is the intensity value of data item x. The idea here is that at least υ_ℓ proportion (i.e., percent) of the item in the final representative result set should be extracted from iteration ℓ, as early iterations would always have a higher aggregated intensity value, and thus, would occupy a bigger portion of the final representative set R. Our empirical results show that by employing υ to compensate for the loss of relevance, we can prevent too many relevant results from being dropped.

To actually generate the final representative results according to υ_ℓ, PrefDiv-PR needs to first run PrefDiv to obtain the initial representative set R, as well as records the number of iterations $\mathcal{L}$ taken to obtained R. During each iteration of PrefDiv, the items that are initially extracted from R_Q (before applying the diversity constraints) will also be recorded into a separate set B_ℓ. Afterward, PrefDiv-PR will examine the set of items in R that are extracted from each B_ℓ with the corresponding υ_ℓ to determine if additional items need to be extracted from B_ℓ and added into R. Note that if such extraction is necessary, depending on the number of items needed to be extracted, the set of items with the highest intensity value in B_ℓ that have not been included in R will be chosen from B_ℓ and placed in R. Finally, once all υ are satisfied for each iteration, the set of k items with the highest intensity value in R will be retrieved as the final results.

3.4 Optimize Diversity Constraints for Coverage

As discussed previously in Section 2.1.3, coverage is yet another important property of the representative set. It gives us two major benefits. First, it helps to ensure that the underlying data space (i.e., the original set) has been well represented by the selected representative items. Second, it enables the possibility for the user to retrieve items that are not in the representative result set by performing "zoom-in" operations—each representative item can be seen as the leader of a set of similar items, and by

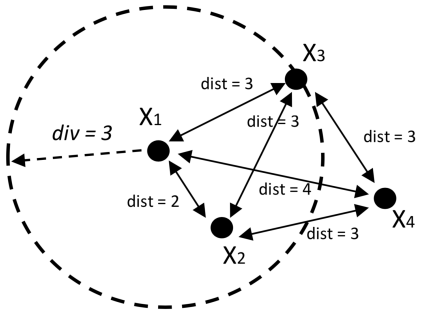

Figure 4: Illustration of the optimal radius, when $k = 2$

"zooming-in" to one of the leaders, similar items around the leader can be revealed.

Since the definition of coverage depends on the similarity between data objects, it is defined by the set of diversity constraints. In order to boost coverage, a set of appropriate diversity constraints must be defined. Below, we will discuss a general approach to determine such diversity constraints through an example.

Example 3.1. Consider a set of initial results R_Q that contains 100 items, each with two dimensions, $k = 30$, a single diversity constraint ψ that considers both dimensions, and the Euclidean distance as the diversity measure. Furthermore, assume that no partial diversity is allowed, meaning the final representative set produced must be a strict DT-k set. Obviously, whether it is possible to produce a DT-k set with 30 items is dependent on the definition of diversity constraints, such that if the diversity constraint consists of a diversity threshold that is beyond the maximum pair-wise distance between any pair of items in the original result set, then only a single item can be included in R, as the rest of the data items would be discarded due to the violation of the diversity constraint. Clearly, returning a result set with a single item when $k = 30$ is not ideal, and thus, the diversity threshold should be adjusted lower. In contrast, a minimum possible diversity threshold (i.e., 0) would lead to an arbitrary set of k items, which gives no guarantee of either diversity or coverage.

Clearly, from the example, an *optimal diversity constraint* should include a diversity threshold that exhibits the following properties: (1) be as large as possible to improve the coverage; and (2) be small enough to allow a strictly diverse representative set (i.e., DT-k set) with k mutually dissimilar items being formed.

With the above observations, we define the optimal diversity constraint as:

Definition 6. *Optimal Diversity Constraint.* For a given set of item R, an integer number k, and a distance function $dist(x_1, x_2)$, an optimal diversity constraint must contain the largest possible distance threshold, denoted as div_{opt}, that exists between a pair of items in R that can be used to generate a DT-k subset R_Q from R, such that $|R_Q| >= k$ and no two item in R_Q are similar according to $dist(x_1, x_2)$ and div_{opt}.

As illustrated in Figure 4, assume we have a set of points $P = \{x_1, ..., x_4\}$, such that each point consists of a 2-D coordinate and a diversity constraint ψ that consist of both dimensions and uses Euclidean distance. In such case, if $k = 2$, then $div = 3$ will be the div_{opt} (i.e., optimal diversity threshold) for ψ. If any value less than 3 is chosen to be the div_{opt}, then at least three points will remain after removing all similar points because only p_2 and p_3 are considered to be similar with respect to a diversity threshold of 2. If any value larger than 3 is chosen to be the div_{opt}, then only one point will remain in P after removing all similar points. Thus, 3 is the only option for div_{opt}, as no other distance threshold would be able to produce a result with three items. Unfortunately, finding the optimal diversity threshold for a given distance measure and a set of attributes is NP-hard.

ALGORITHM 3: SearchOptimalDiversityThreshold

Require:
1: A set of items R_Q, a size k, a set of attribute Δ, and a distance function $dist(x_1, x_2, \Delta)$
Ensure:
2: A diversity threshold div_{opt}
3: $S \leftarrow$ an initial item $x \in R_Q$
4: $div_{opt} \leftarrow \emptyset$
5: **while** $|S| < k$ **do**
6: $\quad x^* \leftarrow \arg\max_{x \in (R_Q - S)}(\min(dist(x, x_j, \Delta) : \forall x_j \in S))$
7: $\quad S \leftarrow S \cup x^*$
8: $\Theta \leftarrow$ minimum distance between any pair of items in S
9: $x^R \leftarrow \arg\max_{x \in (R_Q - S)}(\min(dist(x, x_j, \Delta) : \forall x_j \in S, s.t. dist(x, x_j, \Delta) < \Theta))$
10: $div_{opt} \leftarrow \min(dist(x^R, x_j, \Delta) : \forall x_j \in R_Q)$
11: **Return** div_{opt}

According to Definition 3, two items x_i, x_j are dissimilar iff they fail to satisfy a diversity constant Ψ with a diversity threshold div and distance function $dist(x_1, x_2)$. Since an item, x_i can be included in a DT-k subset R *if and only if* x_i satisfies all diversity constraints with respect to other items in R, the maximum distance d between any pair of items in R increases along with the diversity thresholds inside each diversity constraints. Based on the Definition 6 of the optimal diversity constraint, the problem of finding the optimal diversity thresholds (i.e., div_{opt}) for a given diversity constraint can be mapped to the MaxMin Diversity Problem, which aims to select a representative subset $R \subseteq R_Q$, such that $|R| = k$, and the minimum distance between any pair of items in R is maximized. As the MaxMin Diversity problem has been previously proven to be an NP-hard problem [4], finding the optimal diversity threshold is also an NP-hard problem.

Inspired by the MaxMin Diversity problem, we adopt a greedy heuristic (Algorithm 3), which automatically computes an approximation of the optimal diversity thresholds for a given set of diversity constraints. As illustrated in Algorithm 3, we first find a subset $S \subseteq R_Q$ that maximize the minimum distance between items in R (Lines 5 - 7). Then, we generate the optimal diversity threshold by comparing the pair-wise distance of all items that are in R_Q but not in S (Lines 8 - 11). The optimal diversity threshold is defined as the largest distance between a pair of items in R_Q that is smaller than the minimum distance between any pair of items in R. As proven in [30], the result produced by this greedy heuristic has a $\frac{1}{2}$ approximation of the optimal solution and a quadratic complexity, and no other polynomial algorithm can provide a better guarantee.

4 EXPERIMENTAL EVALUATION

To study the effectiveness of our PrefDiv and PrefDiv-PR algorithms, we compare them to the two most effective diversified top-k algorithms, namely *Swap* [37] and *MMR* [5], as suggested in [34]. We also compare them to four diversity-focused algorithms, *MaxSum, MaxMin, K-Medoids*, and *DisC Diversity*, to assess how well diversity has been preserved when relevance is taken into account. All the algorithms in our evaluation are discussed in Section 5.

4.1 Experimental Testbed

We implemented all of the algorithms with JDK 8.0 on an Intel machine with Core i7 2.5Ghz CPUs, 16GB RAM, and 512GB SSD.

Algorithms. We implemented MMR and Swap based on their published descriptions [5] and [37], respectively. The MaxMin and MaxSum algorithms used in our experiments are based on Definitions 8 and 9 (in Section 5), respectively, and DisC Diversity is taken directly from the original author [10]. However, DisC Diversity is not top-k bounded algorithms, and the size of the result set that DisC Diversity produces is heavily dependent on the radius. To allow for a comparison, we modified the DisC Diversity to stop when the size of the result set equals k. We also included one well-known clustering algorithm, K-Medoids [26], which aims to group a set of data objects into clusters through some distance measure, so objects within a cluster are close to each other and objects outside of the cluster are unrelated to the objects inside the cluster.

In our experiment, we implemented K-Medoids based on [26]. Since K-Medoids does not capture the relevance in any regard, we improved the performance of K-Medoids in balancing the relevance *vs.* diversity trade-off by choosing the object with the highest intensity value as the final recommendation from each of its k clusters. This improvement significantly enhances the performance of the K-Medoids with respect to relevance, while exhibiting the minimum decrease in diversity.

Most of the diversification techniques involved in our experiments require some parameters, since finding the best parameters for each technique that are optimum under all situations would be too difficult. For the purpose of comparison, in all of our experiments, there is only one diversity constraint ψ for any given set of experiments, which is used by all algorithms that require a diversity constraint during its execution. We fixed the diversity threshold div used in ψ for each set of the experiments, which is computed with the optimal radius by Algorithm 3. All other parameters except ψ and r are fixed for all runs and adjusted according to the suggestion of the original authors, or based on the best overall performance. For MMR, we set $\lambda = 0.3$, for Swap, we set the $UB = 0.1$, and for PrefDiv, we set $A = 0.6$.

Datasets. We ran our tests on two real-world datasets: Cameras [10], and Foursquare. We selected these datasets in order to experiment with two different distance functions, *Hamming* with Cameras and *Euclidean* with Foursquare. The Cameras dataset consists of 579 records and 7 attributes per record. The Foursquare dataset is collected from the major location-based social network, Foursquare. We obtained real-life user preferences, used Foursquare's public venue API, and queried information for 14,011,045 venues. In order to build realistic user profiles for our evaluations, we used a dataset collected by Cheng et al. [7] that includes geo-tagged user-generated content from a variety of social media between September 2010 and January 2011. This dataset includes 11,726,632 check-ins generated by 188,450 users. Accordingly, each reading in our Foursquare dataset has the following tuple format: <ID, latitude, longitude, # check-ins, # unique users>. In our experiments, we consider only data items (i.e., venues) from New York City (NYC), which consists of 10912 items and San Francisco (SF), which consists of 7859 items.

User Preferences. The intensity values (I) for each individual dataset is generated as follows:

For the Cameras dataset, we generated 100 different sets of user preferences, such that in each profile the preference intensity value for each individual camera is generated based on a uniform distribution, and each individual user preference is represented as one unique query.

For the Foursquare dataset, we obtained the *real-life* user preferences based on the hierarchy of the Foursquare dataset, such that every individual venue v in the dataset is associated with a type T_v. For example, an Italian restaurant belongs to the category "Italian restaurant", which can belong to the higher level category "Restaurants", which can itself belong to the category "Food", and so on. In order to build highly personalized and specific profiles, we use the bottom layer of the hierarchy, as well as the specific venues visited. In particular, given the set of check-ins C_u of user u, we build a hierarchical profile $\mathcal{P}$ where at the top level, the preferences of the user are expressed in terms of the (normalized) frequencies of this user's visitations with respect to the types of venues. The second layer of the user profiles further provides the normalized frequencies of venues for the different types of locations visited by u. Since our user profile is sparsely gathered during a short period of time, to resemble a real-world user profile, we merged the 1000 sparse Foursquare user profiles to create one superuser profile. We performed our experiments by randomly selecting 50 query points from each city (100 query points in total). For each query point, we considered all venues located within a 1.5 kilometer radius of the query location.

4.2 Evaluation Metrics

In our experimental evaluation, we evaluate the performance of all models based on three well-known and commonly used metrics: *Normalized Relevance* [35], *Coverage* [10] (Definition 5), and *Execution Time*. Note, there are two other commonly used metrics for evaluating ranking algorithms such as DCG and Spearman rho. However, both metrics focus on measuring the correctness of the order of the results produced by the ranking algorithm. Thus, they are not the ideal evaluation metrics for evaluating the effectiveness of result diversification algorithms. As stated in previous sections, our proposed PrefDiv algorithms are post-processing steps of initial query results, which does not impact the relative order of the original result set. In other words, the produced representative result set of PrefDiv algorithms essentially follows the original order of the initial results. Thus, metrics that focus on the correctness of the ranking order do not fit the context of this evaluation.

Definition 7. *Normalized Relevance.* Let S be a set of items and $S_k^* \subseteq S$ such that $|S_k^*| = k$. The Normalized Relevance of a subset S_k^* is defined as the sum of the intensity value of items in S_k^* over the sum of k items with highest intensity value in S.

$$nRev(S_k^*) = \frac{\sum_{x \in S_k^*} I_x}{\max_{S_k \subseteq S, |S_k|=k} \sum_{x \in S_k} I_x} \tag{2}$$

In order to calculate the coverage for the Foursquare dataset, given that it is difficult to find a fixed radius that would work with any query location, we calculate the coverage with respect to the optimal radius generated by Algorithms 3 for every output size k. In the case of a Camera dataset, where the hamming distance is employed, the coverage is calculated with a fixed diversity threshold/radius $div = 3$ (which is the mid-point of the maximum distance allowed). For a fair comparison, all algorithms are evaluated with respect to the same diversity threshold/radius. Note that Normalized Relevance and Normalized Intensity Value would be used interchangeably in the following sections.

4.3 Experimental Results

Here we report the findings of our experimental evaluation.

4.3.1 Normalized Relevance. As demonstrated in Figures 5, 8, and 11, we can see a clear separation between two groups of algorithms for all datasets, where PrefDiv, PrefDiv-PR, Swap, MMR, and K-Medoids tend to group together, and MaxMin, MaxSum, and DisC Diversity form another group. The reason for this is that the second group does not take relevance into account; hence, it would be unlikely for them to retrieve a representative subset that has a high total intensity value. In contrast, the first group of algorithms takes relevance into account, and, as such, it

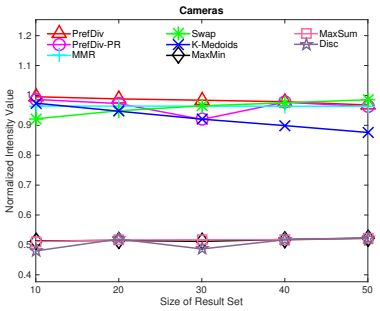

Figure 5: Normalized Intensity Value of Cameras

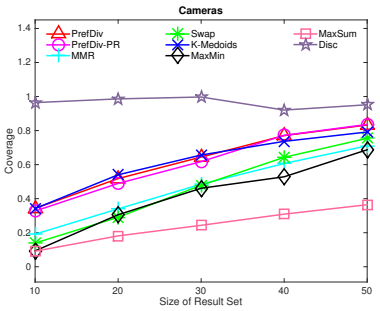

Figure 6: Coverage of Cameras.

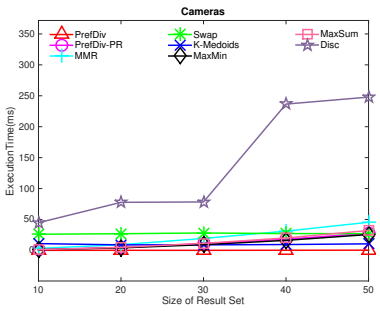

Figure 7: Execution Time of Cameras.

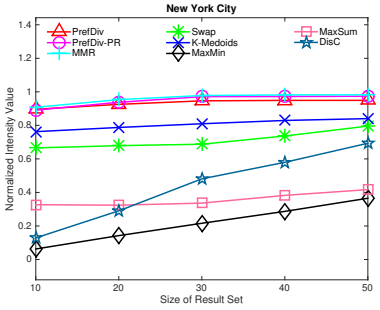

Figure 8: Normalized Intensity Value for the Foursquare, NYC

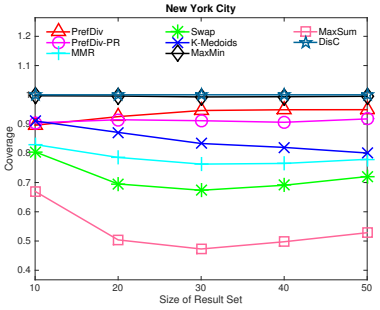

Figure 9: Coverage of Foursquare, NYC

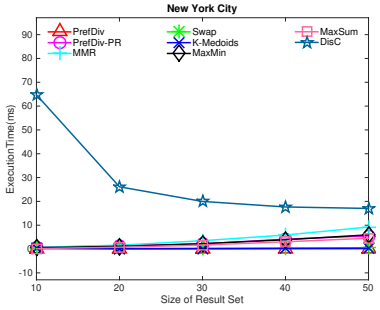

Figure 10: Execution Time for the Foursquare, NYC

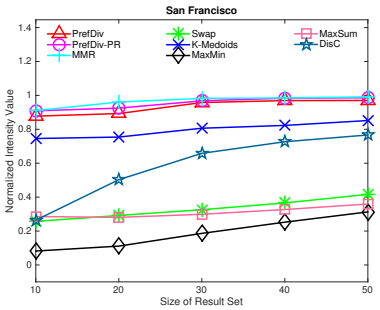

Figure 11: Normalized Intensity Value of Foursquare, SF

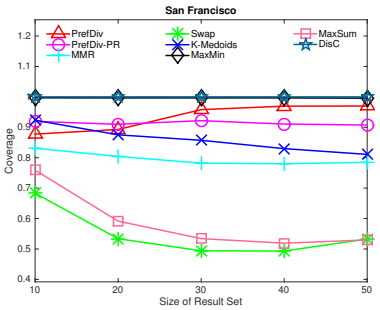

Figure 12: Coverage of Foursquare, SF

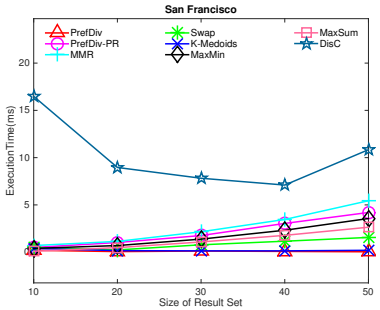

Figure 13: Execution Time of Foursquare, SF

achieves a significantly higher performance in terms of retrieving relevant items.

4.3.2 Coverage. Figures 6, 9, and 12 show that our PrefDiv and PrefDiv-PR exhibit better coverage on average when compared with MMR and Swap by 20% and 42%, respectively. The reason could be because both MMR and Swap optimize dissimilarity as their definition of diversity. In contrast, both PrefDiv and PrefDiv-PR are coverage-aware algorithms, which seek an optimal radius that directly improves the coverage of the representative result set. Therefore, both PrefDiv and PrefDiv-PR can perform much better than Swap and MMR. We also observed that on average PrefDiv is able to outperform MaxSum in terms of coverage by 160%, which could be explained because MaxSum as pure dissimilarity-based algorithm fails to cover the entire space of the dataset. K-Medoids demonstrates good coverages for both datasets. However, both PrefDiv and PrefDiv-PR still exhibit slightly better coverage in general.

The MaxMin algorithm performs well in terms of the Foursquare dataset, although the performance dropped significantly for the cameras dataset. The reason could be because in the Foursquare dataset the average number of venues around each query point is about 90 venues. In contrast, the Cameras dataset consists of 579 tuples. This shows that MaxMin is able to obtain good coverage with a relatively small dataset and Euclidean distance that takes a wide range of values as distance, but fails to cover the space with large datasets and hamming distance that only takes the number of attributes + 1 distinct values as distance. In both datasets, DisC Diversity demonstrated the highest coverage, which is to be expected since DisC Diversity is the only algorithm in the experiment that directly optimizes coverage as the only objective.

Another interesting observation is that, in the Foursquare dataset, except PrefDiv, PrefDiv-PR, and DisC Diversity, other algorithms appear to have a drop in coverage when the value of k increases, although, in general, with the increase in result size the coverage should increase as well. The reason for such behavior is that in the Foursquare dataset, we employed the optimal radius as the criterion for determining the similarity between items. Therefore, with the increase in result size, the optimal radius becomes smaller, thus leading to a decrease of coverage for some algorithms.

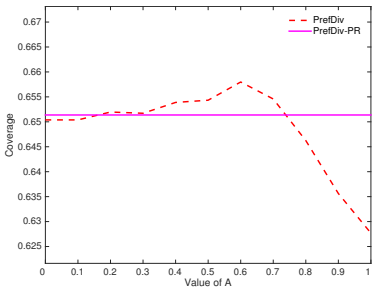

Figure 14: Coverage of different settings of A, with optimal radius and k = 30.

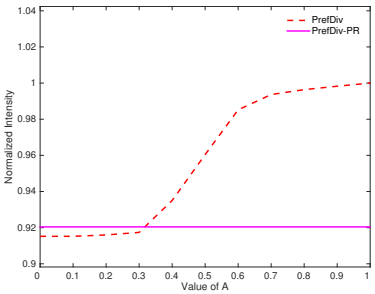

Figure 15: Normalized Relevance of different settings of A, with optimal radius and k = 30.

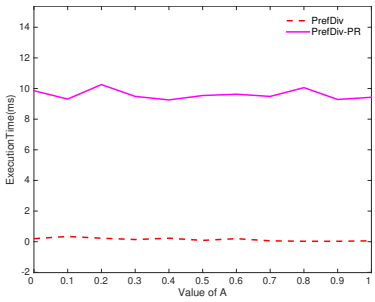

Figure 16: Execution Time of different settings of A, with optimal radius and k = 30.

4.3.3 Execution Time. We have measured the execution time required by all algorithms. As shown in Figures 7, 10, and 13, our PrefDiv and PrefDiv-PR appears to be the overall fastest algorithm when compared to all other alternatives. In general, PrefDiv and PrefDiv-PR perform near identically in terms of runtime, which is expected as the additional computation overhead introduced in PrefDiv-PR is negligible. To illustrate the efficiency of our proposed greedy heuristic for searching the optimal diversity threshold, we have included its runtime in the PrefDiv-PR, so the runtime difference between PrefDiv and PrefDiv-PR reflects the runtime of the search optimal diversity threshold algorithm. With that in mind, PrefDiv-PR is still on average faster than both MMR and Swap by up to 72% and 116%, respectively. Specifically, in the Camera dataset, PrefDiv is able to execute 57 times faster than K-Medoid, 127 times faster than MMR, and 159 times faster than Swap. In the Foursquare dataset, most algorithms tend to be faster when compared with Cameras, because the number of venues near each query point in Foursquare is much smaller than in the Cameras dataset. However, we still observed that, on average, PrefDiv is able to outperform MMR and Swap by 30 and 36 times, respectively. When compared to K-Medoid, which is also very efficient when dealing with this type of dataset, PrefDiv still appears to be 2.7 times faster than K-Medoid on average. As mentioned previously, if the optimal radius for most frequent queries is stored, PrefDiv-PR would not need to calculate the optimal radius for these queries again. Furthermore, for the fairness of the comparison, all of the algorithms run in a single-threaded mode. Since the optimal radius computation method that we adopted is fully parallelizable, it can take advantage of the modern multi-threaded CPU architecture to speed up the computations. In fact, we have observed linear speed up with respect to the number of CPU cores in the system up to 16 cores (the highest we have experimented). One interesting remark here is that, in the Foursquare dataset, the execution time of DisC Diversity drops when k increases from 10 to 20. The reason is that the runtime of DisC Diversity is also affected by the length of the radius, therefore, with smaller output sizes, the optimal radius will become larger, which leads to the relativity longer execution time of DisC.

4.3.4 Parameter A of PrefDiv. As illustrated in Figures 6 to 11, a performance difference between PrefDiv and PrefDiv-PR exists due to the existence of the accuracy parameter A in the PrefDiv algorithm. In this section, we conducted an experiment to study the effect of parameter A in PrefDiv with the Camera dataset and $k = 30$. As shown in Figures 14, 15, and 16, when A increases from 0 to 1, we observed an improvement of normalized relevance, albeit with a decrease in coverage. This is as expected since, with higher values of A, PrefDiv will select more relevant items, and with lower values of A, more diverse items will be selected that lead to an increase in coverage. However, this would be at the expense of lower relevance. The execution time appears to be stable regardless of the value of A. This is because, for each iteration, PrefDiv only requires a tiny amount of execution time. Therefore, the additional iterations introduced by the low value of A would not have a large impact on the overall runtime.

4.3.5 Relevancy vs. Diversity. Lastly, as a summary, we present three scatterplots that capture the trade-off between relevance and diversity. Each point in Figures 17 and 18 are corresponding to the average of over 50 different query locations with one value of k, and each point in Figures 19 are corresponding to the average of over 100 different user profiles with one value of k. As shown in the figures, we have Normalized Intensity Value as the y-axis and Coverage as the x-axis. Algorithms located in the upper left corner of the figure exhibit the best coverage result, while those in the lower right corner have the highest relevance scores. As we can observe, both PrefDiv and PrefDiv-PR are located towards the upper right corner (circled) for all three scatter plots, which indicates that both PrefDiv and PrefDiv-PR exhibit better ability to handle the trade-off between relevance and diversity with respect to both datasets and distance measures. One may notice that in the Camera dataset, the advantage of PrefDiv and PrefDiv-PR with respect to other alternatives is relatively smaller compared to that in the Foursquare dataset. This is because the Cameras dataset uses the Hamming distance as the distance measure, which has a much smaller domain than the Euclidean distance used in the Foursquare dataset, thus weakening the benefit of the optimal diversity threshold. These results also indicate that the relational proportionality introduced in PrefDiv-PR does effectively improve the quality of the result, since PrefDiv-PR is able to outperform (although slightly) the PrefDiv with manually configured relevance parameter A.

4.3.6 Additional Observations. Despite the fact that both PrefDiv and PrefDiv-PR run up to 159 times faster than other alternatives, the greedy heuristic (Algorithm 3) that we proposed for finding the optimal diversity threshold/radius runs at a quadratic time complexity, and thus, it is much slower. Although it is not required to run this heuristic before each execution of the PrefDiv/PrefDiv-PR, it certainly helps improve its performance. However, this is not an issue with our PrefDiv/PrefDiv-PR algorithm because other algorithms (e.g., DisC Diversity) also benefit from the optimal diversity threshold as much as PrefDiv/PrefDiv-PR.

Fortunately, this greedy heuristic only needs to run once for each query, and thus, it can simply be cached to boost the runtime of frequent queries significantly.

5 RELATED WORKS

In this section, we discuss works that are closely related to ours from multiple aspects.

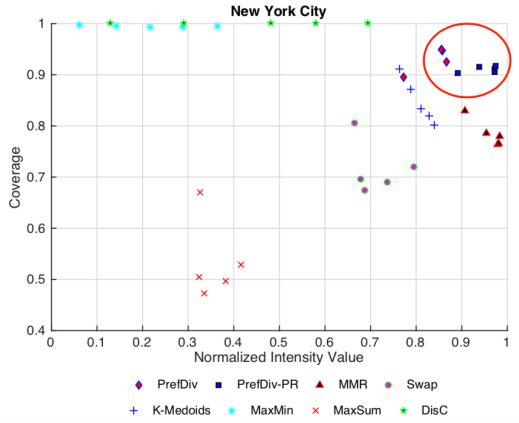

Figure 17: Relevance VS. Diversity (NYC).

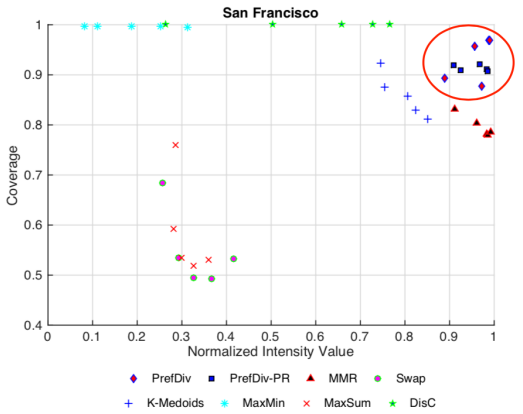

Figure 18: Relevance VS. Diversity (SF).

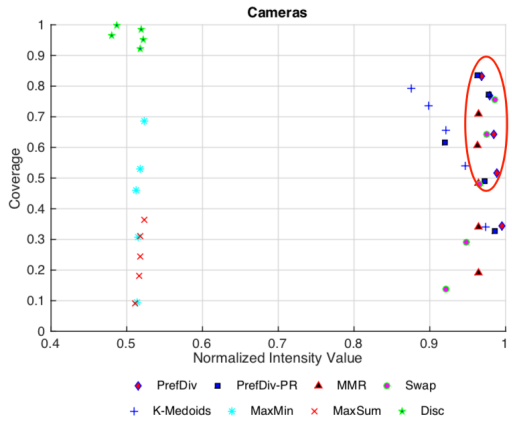

Figure 19: Relevance VS. Diversity (Cameras).

5.1 Relevance Ranking Techniques

Many ranking techniques using preference have been proposed. These are comprehensively surveyed in Stefanidis et al. [33]. As mentioned above, these techniques can be distinguished based on the type of preferences they support for filtering and ordering data. These techniques primarily handle only one type of preference, either *quantitative* or *qualitative*. However, each preference type has its own advantages and disadvantages. Hybrid schemes that support both qualitative and quantitative preferences have been proposed in an attempt to exploit the advantages of both types of preferences while eliminating their disadvantages [17, 22]. In this work, our proposed algorithms can work with any existing relevance ranking model that returns a set of sorted tuples/objects along with their scores/intensity values.

More recently, in [6], the author studied the problem of producing rankings while preserving a given set of fairness constraints. In particular, the proposed algorithm takes as input, a utility function, a collection of sensitive attributes (e.g., gender, race), and a collection of fairness constraints that restrict the number of items with each sensitive attribute that are allowed to appear in the final results. It outputs a ranking that maximizes the relevance with respect to the given utility function while respecting the fairness constraints. As mentioned previously, our proposed PrefDiv algorithms can leverage any existing relevance ranking model. Therefore, in the case where the required sensitive attributes and fairness constraints can be provided by the user, PrefDiv can be used in conjunction with the ranking produced in [6].

5.2 Diversity Techniques

Result diversification has been studied in many different contexts and with various definitions [10], such as similarity, semantic coverage [2], and novelty [8]. In our work, we focus on the similarity definition and use MaxMin and MaxSum, which are two widely used diversification models, as baselines.

The goal of these two diversification models is to select a subset S from the object space R, so that the minimum or the total pairwise distances of objects in S is maximized. Recently, a number of variations of the MaxMin and MaxSum diversification models have also been proposed (e.g., [9, 25]) to address the problem of diversifying continuous data. Formally, MaxMin and MaxSum are defined as follows:

Definition 8. *MaxMin* generates a subset of R with maximum $f = min_{p_i,p_j \in S} dt(p_i,p_j)$ where dt is some distance function $p_i \neq p_j$ for all subsets with the same size.

Definition 9. *MaxSum* generates a subset of R with maximum $f = \Sigma_{o_i,o_j \in S} dt(o_i,o_j)$ where dt is some distance function $o_i \neq o_j$ for all subsets with the same size.

DisC Diversity [10] is the most recently proposed diversity framework and solves the diversification problem from a different perspective. In DisC Diversity, the number of retrieved diverse results is not an input parameter. Instead, users define the desired degree of diversification in terms of a tuning parameter r (radius). DisC Diversity considers two objects o_i and o_j in the query result R to be similar objects if the distance between o_i and o_j is less than or equal to a tuning parameter r (radius). It selects the representative subset $S \in R$ according to the following conditions: (1) For any objects in R, there should be at least one similar object in S; and (2) All objects in S should be dissimilar to each other. These two conditions ensure both the coverage and the dissimilarity property of a diverse result set.

In addition, DisC Diversity also introduces two problems, *Covering* and *CoveredBy* [11]. These can be used to model the issue of generating a representative result set that is both diverse and relevant to a user's individual preference (without using preferences). The Covering problem is used to model the case where users want highly relevant items to cover a large area around them. In order to achieve this goal, a relatively larger radius is assigned to items with larger weights. The CoveredBy problem is used to model a case where a user wants to see more relevant objects. In that case, a smaller radius is assigned to items with larger weights. These two problems together illustrate the possibility of using DisC to handle relevance together with diversity.

The key differences between PrefDiv algorithms and DisC Diversity are: (1) PrefDiv algorithms follow the Top-k paradigm, which provides users with the option to specify the size of the final result set by assigning a value to parameter k, whereas DisC Diversity adjusts the size of the result set by changing its radius parameter r. (2) The PrefDiv algorithms focus on both the *relevance* of the result set with respect to the users' preference and the *diversity* of the result set. DisC Diversity focuses mainly on the most diverse representative subset with two scenarios that only illustrate the possibility of using DisC Diversity to handle such relevance-aware diversity requests; however, they do not mention any specific strategies on how one can dynamically change r with respect to Covering or CoveredBy. In addition, our implementation of PrefDiv-PR eliminates the need for identifying r (i.e., diversity threshold) manually by automatically finding the most suitable diversity threshold under any given situation.

Another way to generate a diverse, representative set of results is through clustering. One example of this would be *k-Medoids*, which is a well-known clustering algorithm that attempts to minimize the distance between points in a cluster and the center point (medoid element) of that cluster. The k-Medoids algorithm can be classified into two stages: In its first stage, it generates a set of k clusters C = {c1, c2, ..., ck} based on some distance function dt. In the second stage, one element from each cluster is selected to be part of the result set R. Several strategies for selecting an element from each cluster could be employed. For instance, one strategy is to choose the center point of each cluster that is expected to deliver high diversity, and another strategy would be to choose the point that has the highest intensity value for each cluster. However, since there is no parameter that can be tuned, either manually or automatically, to balance the trade-off between relevance and diversity, k-Medoids is unable to balance such a trade-off in fine granularity.

5.3 Multi-Criteria Objective Optimization

In the past, diversification and retrieval of relevant results have often been studied together as a multi-objective optimization problem with two objectives, where the first objective is relevance, and the second objective is dissimilarity [38]. The following are some representative techniques that are related to our work.

In [27], the authors considered the optimization of the diversified Top-K problem as finding the optimal solution for the maximum weight independent set problem, which has been proven to be an NP-hard problem. The authors proposed an approach, called *div-astar*, which uses a diversity graph that consists of N nodes, where each node corresponds to one item in the original data. This diversity graph is sorted according to the relevance score, and an a^* algorithm is used to find the optimal solution for diversifying Top-K Results. In addition to the div-astar solution, two enhancements have also been proposed, called *div-dp* and *div-cut*: div-dp takes advantage of dynamic programming to divide the initial graph into disconnected components, and div-cut is a cutpoint-based approach that further decomposes each disconnected component based on loosely connected sub-graphs. PrefDiv algorithms are different from div-astar [27] (Section 5.3), in that the main objective of div-astar is to find the exact solution for the maximum weight independent set; hence, even with all the enhancements and decompositions, each sub-problem is still NP-hard. On the other hand, although PrefDiv algorithms also consider the maximum weight independent set problem as part of the algorithm, they take advantage of greedy approximation with a relaxed constraint, which allows similar items to be included in the result set if the relevance distribution of the original data can be better reflected in the resulting set. Furthermore, such relaxed constraints allow PrefDiv to be more practical for border usage, especially for tasks that require a short response time.

One widely used approach that was targeted directly at optimizing the trade-off between diversity and relevance was introduced by [5]. In this work, the authors proposed the famous twin-objective function called *Maximal Marginal Relevance* (MMR), which combines both relevance and diversity aspects in a single, comprehensive objective function. Formally, MMR defines its objective function as:

$$\arg\max_{D_i \in R \setminus S} [\lambda(Sim_1(D_i, Q) - (1-\lambda) \max_{D_j \in S} Sim_2(D_i, D_j))] \quad (3)$$

Where λ is a scaling factor that specifies the preference between relevance and diversity. When $\lambda = 1$, the MMR function equals a standard relevance ranking function. When $\lambda = 0$, it computes a maximal diversity ranking. Comparing PrefDiv to MMR [5] approach, one can clearly see the difference: there are no comprehensive objective functions being used in the PrefDiv algorithms. Our approach addresses the combined problem of relevance and diversity through a combination of multiple steps, rather than solving it in one single function.

Recently, a new bi-criteria objective optimization approach based on MMR has been proposed [19]. This approach integrates *regret minimization* with traditional MMR to generate a new relevance score that takes into consideration the case of minimizing the disappointment of users when they see k representative tuples rather than the whole database. In this work, the authors proposed two approximation algorithms called *ReDi-Greedy* and *ReDi-SWAP*, which find the set of items consisting of k items having the highest score with respect to their MMR function.

In [32], the author has conducted a study on personalized, keyword-based search over relational databases, which includes the notion of diversity and coverage. Specifically, the author provided good discussions on modeling the relevance, user preferences, diversity, and coverage for keyword-based searches over relational databases by means of Join Tree of Tuples. Join Tree are trees of tuples connected through primary to foreign key dependencies. However, PrefDiv algorithms assume that a utility function F is given in advance to reflect the relevance and user preference, and thus does not focus on modeling the relevance and user preferences. Furthermore, PrefDiv algorithms are general, post-processing techniques for result diversification, and hence, do not restrict themselves to the keyword-based search over relational database settings. As long as proper utility functions and distance measures are given, PrefDiv algorithms can be applied to any data types (e.g., structured, unstructured, semi-structured). Consequently, the definition of coverage in [32] is also different than the definition of coverage in this work. Wheres [32] focuses on covering more user intents based on user profiles, PrefDiv algorithms focus on the proximity between the representative results and original results.

Swap is another recent Top-K diversification technique that is related to ours [37]; Swap starts with K items with the highest relevance scores. Among these K items, Swap picks an item with the lowest contribution to the diversity of the entire set, then swaps this item with the item that has the next highest relevance score. A candidate is successfully swapped with one of the items in the Top-K set if and only if it can contribute more in terms of the overall diversity of the result set. In order to preserve the relevance aspect, Swap introduces an optional pre-defined threshold called UB that specifies how much decrease in relevance can be tolerated. UB can serve as a terminal condition that stops the algorithm when the item with the highest relevance among the remaining set is no longer high enough for the algorithm to perform a swap operation. Our PrefDiv is different from the Swap, as Swap seeks diversity through pairwise distances of items among the result set, filters out items that contribute less to diversity, and ensures relevance by throwing out items that cause the relevance to drop below the pre-defined threshold. In contrast, PrefDiv algorithms seek diversity by eliminating similar items and ensuring relevance by using a relevance-focused

greedy algorithm along with proportionality, which can reflect the relevance distribution of the original domain.

5.4 Data Summarization

Some recent works [24, 36] have studied the problem of providing interactive exploration and summarization support for tuples in a given table. The goal of this type of approach is to produce an informative hierarchy that organizes the underlying tuples essentially in k clusters. In order to display tuples as clusters, each cluster is folded into a single, representative tuple, with only the common attribute values among all members of the cluster being displayed. The rest of the attributes are shown as "?", which indicates that there are objects with different values with respect to these attributes inside the cluster. To explore each cluster, the user can gradually expand each "?" symbol contained in the current representative tuple of a cluster. Each time the user expands a "?" symbol, more tuples that contain a different value with respect to the corresponding attributes will be displayed. Clearly, these works are different than ours. We focus on producing a representative subset that is most informative to the user with adjustable size, rather than summaries of subsets of a table.

5.5 Impacts of PrefDiv

The efficiency of PrefDiv and its ability to balance the trade-offs between relevance, diversity, and coverage have already benefited the design of some real-world systems that need to produce highly informative representative subsets, or require interactive efficiency in producing the representative results (e.g., [14–16, 28, 29]) .

One example is a novel mobile recommendation service that provides a set of diverse points-of-interest (POI's) recommendations [14–16], where the interactive efficiency has been weighted equally important as the quality of the produced recommendations.

Another example is in the scientific domain and dimensionality reduction, which PrefDiv has been employ as a novel way to select subsets of highly informative dimensions for high-dimensional gene expression datasets [28, 29]. Those selected dimensions will then be used to enable effective downstream analysis in a variety of medical and bioinformatics researche.

6 CONCLUSIONS

Scalability from a human point of view is a very challenging problem as it consists of finding the perfect balance between the conflicting objectives of relevance and diversity. Traditional top-k result diversification approaches focus on producing a subset of results that balance the trade-off between selecting highly relevant items and items that are dissimilar with respect to each other. In order to achieve the above-mentioned objectives, most algorithms rely on a number of tunable control parameters, making them harder to configure (and be adopted). Coverage is another important factor of diversity, which has been mostly ignored in previous top-k result diversification algorithms.

In this work, we addressed these problems and proposed an efficient online solution called *Preferential Diversity* (PrefDiv). PrefDiv produces a set of high-quality representative items from a large set of initial answers, where each representative item is chosen to optimize both the *relevance* and *diversity* (i.e., dissimilarity, and coverage). We also proposed a number of optimizations that further improve PrefDiv's usability, efficiency, and effectiveness.

We theoretically analyzed and experimentally compared our algorithms to the state-of-the-art, top-k diversification algorithms. Our evaluation showed that our algorithms achieve similar performance in terms of normalized relevance, but outperforms the state-of-the-art algorithms in terms of coverage by a noticeable margin, while achieving a speedup of the runtime up to two orders of magnitude.

REFERENCES

[1] Independent set (graph theory). https://en.wikipedia.org/wiki/Independent_set_(graph_theory), 2018.
[2] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM*, pages 5–14, 2009.
[3] A. Angel and N. Koudas. Efficient diversity-aware search. In *ACM SIGMOD*, pages 781–792, 2011.
[4] J. Carbonell and J. Goldstein. The discrete p-dispersion problem. *EJOR*, 46:48–60, 1990.
[5] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *ACM SIGIR*, pages 335–336, 1998.
[6] L. E. Celis, D. Straszak, and N. K. Vishnoi. Ranking with fairness constraints. In *ICALP*, pages 28:1–28:15, 2018.
[7] Z. Cheng, J. Caverlee, K. Lee, and D. Sui. Exploring millions of footprints in location sharing services. In *ICWSM*, 2011.
[8] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. BÃijttche, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *ACM SIGIR*, pages 659–666, 2008.
[9] M. Drosou and E. Pitoura. Dynamic diversification of continuous data. In *EDBT*, pages 216–227, 2012.
[10] M. Drosou and E. Pitoura. Result diversification based on dissimilarity and coverage. In *VLDB*, pages 13–24, 2012.
[11] M. Drosou and E. Pitoura. Multiple radii disc diversity: Result diversification based on dissimilarity and coverage. *ACM TODS*, 40(1), 2015.
[12] P. Fraternali, D. Martinenghi, and M. Tagliasacchi. Top-k bounded diversification. In *ACM SIGMOD*, pages 421–432, 2012.
[13] X. Ge, P. K. Chrysanthis, and A. Labrinidis. Preferential diversity. In *ExploreDB*, 2015.
[14] X. Ge, P. K. Chrysanthis, and K. Pelechrinis. Mpg: Not so random exploration of a city. In *IEEE MDM*, 2016.
[15] X. Ge, A. Daphalapurkar, M. Shimpi, D. Kohli, K. Pelechrinis, P. K. Chrysanthis, and D. Zeinalipour-Yazti. Data-driven serendipity navigation in urban places. In *IEEE ICDCS*, pages 2501–2504, June 2017.
[16] X. Ge, S. R. Panati, K. Pelechrinis, P. K. Chrysanthis, and M. A. Sharaf. In search for relevant, diverse and crowd-screen points of interests. In *EDBT*, 2017.
[17] R. Gheorghiu, A. Labrinidis, and P. K. Chrysanthis. A user-friendly framework for database preferences. In *CollaborativeCom*, pages 205–214, 2014.
[18] M. Grohe. Descriptive and parameterized complexity. In *Computer Science Logic, 13th Workshop, number 1683 in LNCS*, pages 14–31, 1999.
[19] Z. Hussain, H. A. Khan, and M. A. Sharaf. Diversifying with few regrets, but too few to mention. In *ExploreDB*, pages 27–32, 2015.
[20] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7):86–94, July 2014.
[21] J. M. Keil, J. S. B. Mitchell, D. Pradhan, and M. Vatshelle. An algorithm for the maximum weight independent set problem on outerstring graphs. In *CCCG*, 2015.
[22] W. Kiessling and G. Kostler. Preference SQL: design, implementation, experiences. In *VLDB*, pages 990–1001, 2002.
[23] A. Labrinidis. The big data - same humans problem. In *Proc. of Conference of Innovative Data Systems Research*, 2015.
[24] A. P. Manas Joglekar, Hector Garcia-Molina. Interactive data exploration with smart drill-down. In *IEEE ICDE*, 2016.
[25] D. Panigrahi, A. D. Sarma, G. Aggarwal, and A. Tomkins. Online selection of diverse results. In *WSDM*, pages 263–272, 2012.
[26] H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, 2009.
[27] L. Qin, J. X. Yu, and L. Chang. Diversifying top-k results. In *VLDB*, pages 1124–1135, 2012.
[28] V. K. Raghu, X. Ge, A. Balajee, D. J. Shirer, I. Das, P. V. Benos, and P. K. Chrysanthis. A pipeline for integrated theory and data-driven modeling of genomic and clinical data. In *ACM BioKDD*, 2019.
[29] V. K. Raghu, X. Ge, P. K. Chrysanthis, and P. V. Benos. Integrated theory-and data-driven feature selection in gene expression data analysis. In *IEEE ICDE*, pages 1525–1532, 2017.
[30] S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Facility dispersion problems: Heuristics and special cases. In *WADS*, 1991.
[31] R. L. T. Santos, C. Macdonald, and I. Ounis. Search result diversification. In *Foundations and Trends in Inf Retrieval*, volume 9, pages 1–90, 2015.
[32] K. Stefanidis, M. Drosou, and E. Pitoura. Perk: personalized keyword search in relational databases through preferences. In *EDBT*, 2010.
[33] K. Stefanidis, G. Koutrika, and E. Pitoura. A survey on representation, composition and application of preferences in database systems. *ACM TODS*, 36(19), 2011.
[34] D. C. Thang, N. T. Tam, N. Q. V. Hung, and K. Aberer. An evaluation of diversification techniques. *LNCS*, 9262:215–231, 2015.
[35] H. Tong, J. He, Z. Wen, R. Konuru, and C.-Y. Lin. Diversified ranking on large graphs: an optimization viewpoint. In *ACM KDD*, pages 1028–1036, 2011.
[36] Y. Wen, X. Zhu, S. Roy, and J. Yang. Interactive summarization and exploration of top aggregate query answers. In *VLDB*, 2018.
[37] C. Yu, L. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: Diversification in recommender systems. In *EDBT*, pages 368–378, 2009.
[38] C.-N. Ziegler, J. A. K. Sean M. McNee, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW*, pages 22–32, 2005.